Designing End-user Creation Tools for Immersive Experiences

by

Lei Zhang

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Information)
in the University of Michigan
2024

Lei Zhang

raynez@umich.edu

ORCID iD: 0000-0003-3584-6754

ACKNOWLEDGEMENTS

The work presented in this dissertation would not be possible without the guidance and support from many people. First and foremost, I would like to express my deepest gratitude to my advisors Steve Oney and Anhong Guo. Steve, thank you for always believing in me even when I was filled with self-doubt, for guiding me through every aspect of conducting research, and for always supporting me no matter how unusual my research interests may have appeared. Anhong, thank you for always inspiring me with your passion and creativity in research, for encouraging me to think more deeply about the impact of my work, and for always advocating for me. I have learned so much from both of you about not just how to conduct high-quality and tasteful research, but also how to lead with creativity, thoughtfulness, and kindness.

I would like to thank my dissertation committee members – Anıl Çamcı, Andrés Monroy-Hernández, and Michael Nebeling – for their invaluable feedback on my dissertation. I would like to thank Anıl, who has brought valuable perspective to this dissertation from arts and immersive media literature. I feel fortunate to have collaborated with Andrés on several chapters of my dissertation. His guidance and mentorship during my research internships were invaluable, and I cannot thank him enough. In addition, I am lucky to have Michael as a role model, who is not only a great person to chat with but has also taught me so much about the research landscape in the XR space. I am extremely privileged to have had such a fantastic committee.

I have been incredibly fortunate to have many amazing collaborators and colleagues over the years, including but not limited to Mark Ackerman, Ashutosh Agrawal, Tianying Chen, Youjean Cho, Tim Chong, Wai-tat Fu, Jacob Gettig, Margaret Hedstrom, Daekun Kim, Sven Kratz, Fannie Liu, Cyrus Omar, Jin Pan, Ava Robinson, Olivia Seow, Brian Smith, Yu Jiang Tham, Rajan Vaish, Xu Wang, and Xubo Yang. Some of them are great mentors who have helped me become a better researcher in different stages of my PhD. Some are wonderful mentees who have graciously tolerated my flaws and helped me refine my mentoring style. Others are fantastic colleagues, and I feel incredibly fortunate to be able to work with them and constantly learn from their expertise.

During my PhD, I was surrounded by a caring and supportive community of fellow students in the School of Information and the Computer Science and Engineering department.

Many thanks to Ruei-Che Chang, Xinyue Chen, Yan Chen, Heeryung Choi, Gregory Croisdale, John Joon Young Chung, Gabriel Grill, Licia He, Jaylin Herskovitz, Jordan Huffaker, Jane Im, Aarti Israni, Alicia Kathleen, Rebecca Krosnick, Linfeng Li, Warren Li, Chen Liang, Xinyi Lu, Mauli Pandey, Jiaxin Pei, Shwetha Rajaram, Hari Subramonyam, April Wang, Jingying Wang, Qingyi Wang, Zhuofeng Wu, Zihan Wu, Xinghui Yan, Shiyan Yan, Ashley Zhang, Jiacheng Zhang, Qiaoning Zhang, Yichi Zhang, Yixin Zou, and many more. I'm also thankful for the friendships I cultivated through music in Ann Arbor. Special thanks to Keera Feng, Jerry Liu, Stone Liu, Boyang Yu, and the Annappella group for making my time in Ann Arbor unforgettable.

Finally, my deepest appreciation goes to my family for their endless love and care. To my parents, who are far away in China but close to my heart, thank you for supporting me in every decision I have made in my life. To Simone, my partner and best friend whom I was lucky to meet during my PhD, thank you for your patience, kindness, and support. I could not have completed this journey without you all.

TABLE OF CONTENTS

[1]Portions of this chapter were adapted from [285]
[2]Portions of this chapter were adapted from [286]

[3]Portions of this chapter were adapted from [282]

[4]Portions of this chapter were adapted from [287]
[5]Portions of this chapter were adapted from [283]

[6]Portions of this chapter were adapted from [284]

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AR	Augmented Reality
VR	Virtual Reality
3D	three-dimensional
2D	two-dimensional
VPL	Visual Programming Language
IoT	Internet of Things
WIMP	Window Icon Menus Pointer
FRP	Functional Reactive Programming
API	Application Programming Interface
HMD	Head-Mounted Display
UI	User Interface
IDE	Integrated Development Environment
VCS	Version Control System
HG	History Graph
GUI	Graphical User Interface
WIM	World in Miniature
DAG	Directed Acyclic Graph
AI	Artificial Intelligence

ABSTRACT

Although immersive experiences powered by Augmented Reality (AR) and Virtual Reality (VR) devices are becoming increasingly engaging, democratizing their creation to empower end-users, particularly those with limited technical skills, to create, share, and communicate, remains a significant challenge. This dissertation explores the design and development of end-user creation tools for immersive experiences, focusing on enhancing expressiveness, supporting design exploration and collaboration, and fostering meaningful social interactions. My work investigates the challenges faced by novice programmers and designers in creating immersive content and proposes innovative solutions to democratize the creation process.

To raise the ceiling of expressiveness, the dissertation introduces FlowMatic, an immersive authoring tool that enables novice programmers to define reactive behaviors through visual representations of Functional Reactive Programming (FRP) in VR. This tool allows users to create interactive virtual experiences with ease. To support design exploration and collaboration, the dissertation introduce a new collaborative version control system named VRGit, which facilitates design exploration and rapid iterations by providing rich history-keeping and workspace awareness in VR. Furthermore, the dissertation explores the integration of generative AI models in immersive authoring through VRCopilot, enabling creative design explorations via human-AI co-creation. Lastly, the dissertation also examines the impact of content created with these tools on social interactions, presenting Auggie, an AR-based communication tool, and Jigsaw, an AR and IoT-based storytelling tool. These tools aim to encourage effort in remote communication and engagement in immersive stories, with positive impact on people's social interactions via immersive content creation.

Throughout the dissertation, I demonstrate that by incorporating novel visualization and interaction techniques in immersive environments, we can design end-user creation tools for immersive experiences that can express reactive behaviors, facilitate design exploration and collaboration, and foster meaningful social interactions.

CHAPTER 1

Introduction

For decades, the idea of being able to immerse ourselves in computer-generated world has fascinated the public, as depicted in science fictions (e.g., [240]) and movies (e.g., [236]). Advances in technologies including Augmented Reality (AR) and Virtual Reality (VR), have brought this idea closer to reality. AR and VR originally emerged as two distinct concepts: AR *combines* the physical environment with virtual information while VR *replaces* the physical environment with virtual information. Recent hardware advancements have merged these concepts into a single consumer-ready computational device, known as a Head-Mounted Display (HMD), which can render 3D virtual content that either replaces (VR) or blends with (AR) users' physical surroundings. Using such devices, users can interact intuitively with the virtual content with relatively high accuracy and low latency. We refer to content or applications powered by AR and VR devices as *immersive experiences.* Immersive experiences have been shown to provide benefits in numerous domains including education [208], mental health [82], social interactions [201], and many others.

Despite the proliferation of commercial AR/VR devices and the widespread availability of immersive experiences, democratizing the creation of immersive experiences to empower a broad range of people, particularly those with limited technical skills, to create, share, and communicate, remains a formidable challenge. For novice programmers, immersive applications are particularly difficult to develop because they require advanced knowledge in areas such as imperative programming, 3D modeling, and computer graphics. For example, while tools such as Unity [260] and Unreal [261] are comprehensive and enable the creation of high-fidelity interactive AR/VR applications, they require navigating a complex visual editor and using programming languages like C#. This makes them more suitable for expert programmers rather than non-technical end-users, including novice programmers and designers [185, 14]. Even desktop-based Visual Programming Languages (VPLs) such as Unreal Blueprints [2] are designed for non-experts and still require users to mentally translate between 2D and 3D representations and predict how their code will execute in VR. This calls for new paradigms for end-users to create immersive experiences.

One possible solution to these challenges is a paradigm called *immersive authoring*, in which users can create, edit, and evaluate 3D content directly while immersed in the 3D environment. Immersive authoring tools offer a What You See Is What You Get (WYSIWYG) experience by enabling an intuitive and efficient workflow for end-users to create immersive experiences via direct manipulation [223] and to view immediate results as they design and program a virtual scene. In alignment with the application of WYSIWYG principles in domains such as text and image editing, prior research has developed several immersive authoring tools for both AR (e.g. [153, 267, 266, 280]) and VR (e.g. [239, 172, 74, 182]). Some of these tools allow end-users to create static 3D models or sketches, while others allow novice programmers to define dynamic relationships between 3D objects. However, most immersive authoring tools have a low ceiling of expressiveness since they cannot express *reactive* behaviors, a fundamental requirement for practical immersive applications. Reactive behaviors are behaviors where the application responds to events like user actions and system events. Defining such behaviors are uniquely difficult in the 3D environment as it typically requires writing text-based imperative event-action code. In addition, most immersive authoring tools provide very limited support for exploring design alternatives, which is essential to prototyping creative content [185]. Exploring design alternatives requires rich history-keeping of versions, frequent communication with stakeholders, and access to creative inspirations. However, these aspects are poorly supported in existing tools, underscoring the need to rethink history tracking, workspace awareness, and creativity support in 3D immersive environments. Lastly, while most immersive authoring tools have focused on offering high ceilings and low thresholds for end-users, less research has explored the meaningful impact of the various content produced with these tools (i.e., wide walls [213]).

To address these limitations of authoring immersive experiences, my argument in the remainder of this chapter, and throughout this dissertation, contains three main components:

- By incorporating visualizations of declarative programming paradigms such as FRP, we can make immersive authoring tools more expressive by allowing novice programmers to define reactive behaviors.

- We can enable the exploration of design alternatives for novice designers in immersive authoring, by carefully design a version control system that provides rich history-keeping and collaboration support, and by integrating generative Artificial Intelligence (AI) models that intelligently suggest creative inspirations.

- The content produced with end-user creation tools for immersive experiences could have a meaningful impact in social interactions by encouraging effortful communication or facilitating co-located storytelling experiences.

1.1 A Paradigm for Expressing Reactive Behaviors

Immersive authoring can take advantage of many of the features that make immersive applications intuitive and natural to use - users can manipulate programming primitives through direct manipulation, immediately see the output of their programs, and use their innate spatial reasoning capabilities when viewing a program. This dissertation starts with an empirical study of the benefits and challenges of the state-of-the-art immersive authoring tools, in Chapter 3, with a focus on programming paradigms such as dataflow programming. I will first present an immersive authoring tool that enables dataflow programming in VR. I then will describe findings and design implications gained from a qualitative user study.

To extend the ceiling of expressiveness of immersive authoring, I will describe a paradigm that allows novice programmers to specify *reactive* behaviors—behaviors that react to discrete events such as user actions, system timers, or collisions. Specifically, I introduce an immersive authoring tool named FlowMatic based on this paradigm. FlowMatic also includes primitives for programmatically creating and destroying new objects, for abstracting and re-using functionality, and for importing 3D models. Importantly, FlowMatic uses novel visual representations to allow these primitives to be represented directly in VR. I will also describe a comparative study that illustrates the advantages of FlowMatic compared to a two-dimensional (2D) authoring tool. The study results reveal that participants were able to build the reactive behaviors using FlowMatic without writing code, and that the immediate feedback in VR makes it intuitive for programming VR applications and fun to play with. I also demonstrate its expressiveness through several example applications that would be impossible to implement with existing immersive authoring paradigm.

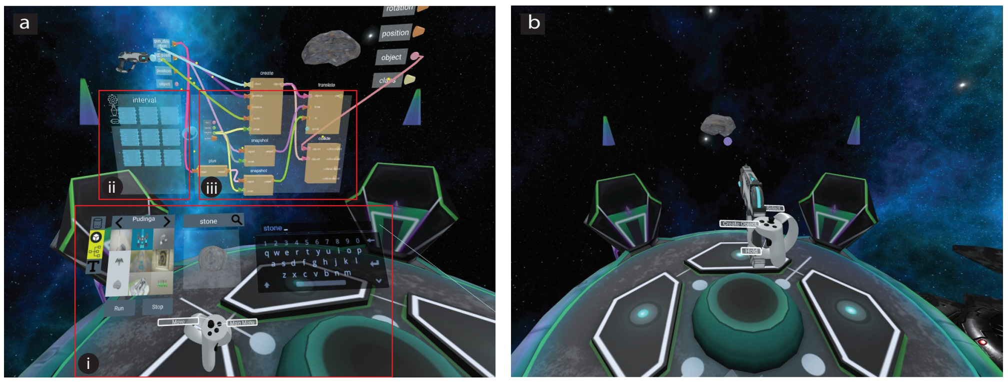


Figure 1.1: System preview of FlowMatic. (a) shows the edit mode. (i) is a palette menu that allows users to browse, search, and import 3D models into the scene. (ii) is a toolbox for the user to add programming primitives such as operators and data sources into the Functional Reactive Programming (FRP) Diagram in (iii). (b) shows the run mode where users can evaluate the application in real time by hiding all the programming primitives.

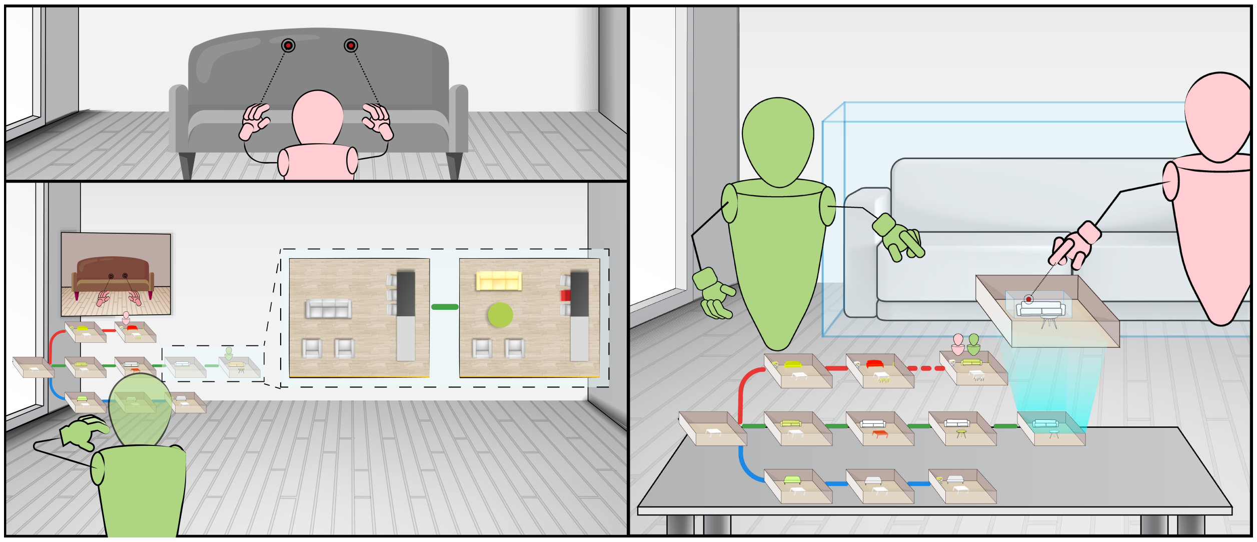


Figure 1.2: System Overview of VRGit. A History Graph (HG) that represents non-linear version history is anchored on the user's left arm, where each node is a 3D miniature of that version. Inside each miniature, objects are highlighted using color coding if they are changed compared to the previous version. Mini avatars are anchored in the HG to represent which version users are in. Users can also create portals to monitor other users' first-person views. A shared history visualization facilitates group discussion by anchoring the HG on a surface and allowing users to preview a version and reuse objects collaboratively.

1.2 Supporting Collaboration and Design Exploration

Another important consideration of designing end-user creation tools for immersive experiences is to enable explorations of design alternatives, which requires rich history-keeping, collaboration with stakeholders, and access to design inspirations [226]. This dissertation presents two systems for this purpose: VRGit and VRCopilot.

Authoring immersive experiences is a creative process that involves numerous iterations, explorations of design alternatives, and frequent communication with collaborators. Version Control Systems (VCSs) help users achieve this by keeping track of the version history and creating a shared hub for communication. However, most VCSs are unsuitable for managing the version history of VR content because their underlying line differencing mechanism is designed for text and lacks the semantic information of 3D content; and the widely adopted commit model is designed for asynchronous collaboration rather than real-time awareness and communication in VR. To this end, we propose VRGit, a new collaborative VCS that visualizes version history as a directed graph composed of 3D miniatures, and enables users to easily navigate versions, create branches, as well as preview and reuse versions directly in VR. Beyond individual uses, VRGit also facilitates synchronous collaboration in VR by providing awareness of users' activities and version history through portals and shared history visualizations. Through an exploratory lab study, I demonstrate that VRGit enables users to easily manage non-linear version histories, communicate with collaborators, and maintain workspace awareness in VR.

VRCopilot is another immersive authoring tool that takes advantage of recent advances in generative AI that enables the automatic creation of realistic 3D layouts. Via this artefact, I explore how capabilities of generative AI can be used in immersive authoring to support design exploration, user agency, and creativity. VRCopilot is a mixed-initiative system that integrates pre-trained generative AI models into immersive authoring to facilitate human-AI co-creation in VR. VRCopilot presents multimodal interactions to support rapid prototyping and iterations with AI, and intermediate representations such as wireframes to augment user controllability over the created content. I will present two rounds of comparative studies that evaluates the potential and challenges of human-AI co-creation including manual, scaffolded, and automatic creation in immersive authoring. Overall, our results show that, when co-creating with AI in VR, the design of intermediate representation in scaffolded creation can enhance user agency, and offering multiple suggestions can increase user creativity.

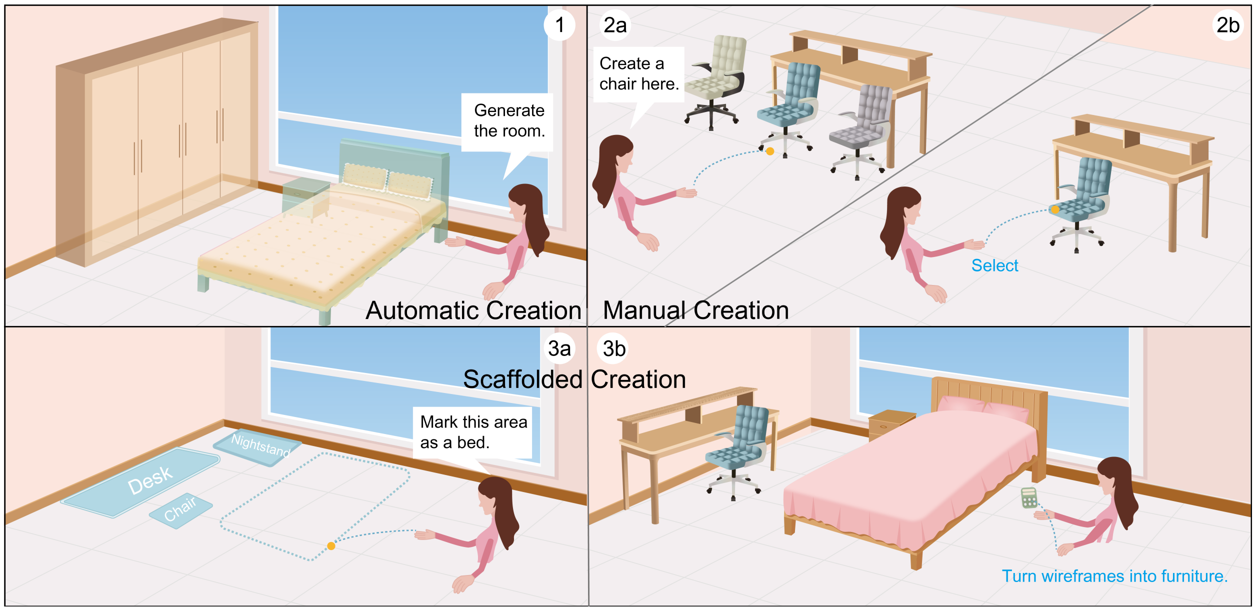


Figure 1.3: System Overview of VRCopilot. 1) Automatic Creation: Users can use voice commands to ask the generative model to generate a full-room layout based on an empty room. 2) Manual Creation: Users can use multimodal specification by speaking with simultaneous pointing to ask the system to suggest a chair (a); they can select from one of the three suggestions offered by the system (b). 3) Scaffolded Creation: Users can create *wireframes* by drawing on the floor while speaking, in addition to automatically generated wireframes (a); They can then turn the wireframes into specific furniture (b).

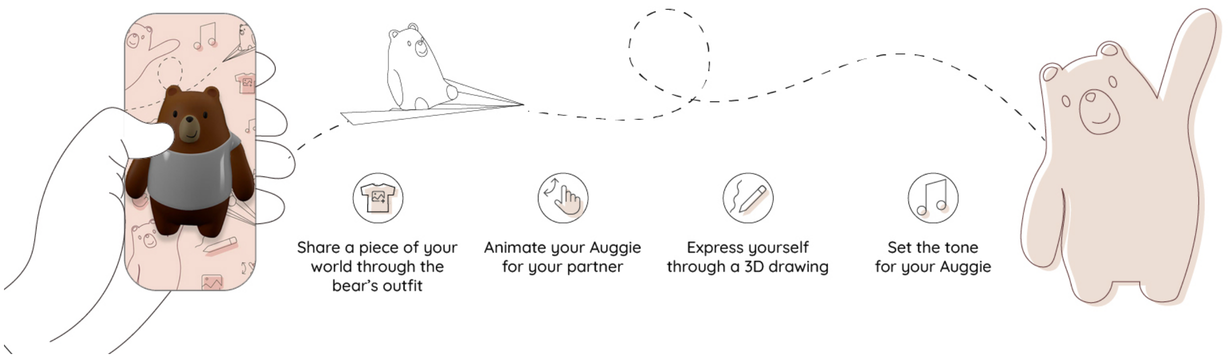

Share a piece of your world through the bear's outfit
Animate your Auggie for your partner
Express yourself through a 3D drawing
Set the tone for your Auggie

Figure 1.4: Overview of Auggie, an app that enables partners to create handcrafted AR experiences for each other, based on customizing a 3D character.

1.3 Social Interactions via End-user Creation

In addition to building end-user creation tool for immersive experiences, my dissertation explores the meaningful impact of content created with these tools on social interactions. Specifically, I will present a field study where we deploy an end-user creation tool for immersive communication, as well as a lab study focused on a tool for creating immersive multi-user storytelling experiences.

I first explore how the end-user creation process of immersive experiences could encourage effortful communication between partners such as friends and families. I will describe Auggie, an iOS app that encourages partners to create digitally handcrafted AR experiences for each other. Auggie is centered around crafting a 3D character with photos, animated movements, drawings, and audio for someone else. I will present a two-week-long field study during which participants used Auggie with their partners remotely. Our findings reveal that the end-user creation tool, i.e. Auggie, can encourage users to engage in meaningful effort during remote communication by evoking a sense of agency that enables them to craft gift-like personal stories. These immersive experiences subsequently have the potential to enable authentic, lightweight connection and feelings of presence between partners. I also discuss design implications and future directions for content creation tools that encourage effortful communication.

Lastly, I will describe Jigsaw, a system that empowers beginners to both experience and craft immersive AR stories, blending virtual and physical elements. This is motivated by the challenge that crafting immersive narratives is complex and generally beyond the reach of amateurs due to the need for advanced technical skills. Jigsaw uniquely combines mobile AR with readily available Internet-of-things (IoT) devices. I will present a qualitative study to assess Jigsaw's effectiveness in both consuming and creating immersive narratives among

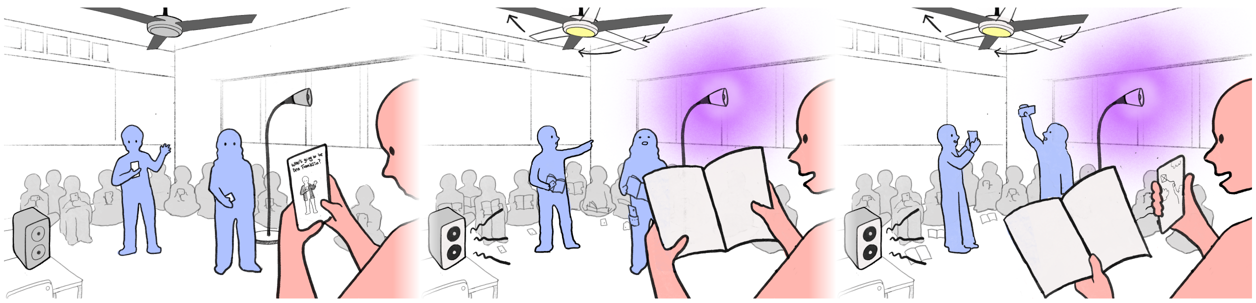


Figure 1.5: An example story of Benjamin Franklin's Kite Experiment from Jigsaw that combines AR and IoT devices into one immersive experience: (a) The participants can assign themselves a character by waving. (b) As the narrator reads the story, trigger words from the narration enact changes in the physical environment (shown with a smart light, smart fan, and smart speaker). (c) Virtual effects or objects are shown in the AR view, such as a kite, clouds, and sparks.

groups. The results from the study reveal that the end-user creation tool, i.e. Jigsaw, can keep users feel engaged, immersed, and connected with each other in the co-located storytelling experiences. However, sensory overload poses a primary challenge. I will also discuss design trade-offs and considerations for future endeavors in both the consumption and the creation of immersive stories involving AR and IoT.

1.4 Thesis Statement and Contributions

Thesis Statement:
By incorporating novel visualization and interaction techniques in immersive environments, we can design end-user creation tools for immersive experiences that can express reactive behaviors, facilitate design exploration and collaboration, and foster meaningful social interactions.

To achieve this, I make the following contributions in this dissertation:

- A qualitative study that evaluates the challenges and benefits of the state-of-the-art immersive dataflow programming tools.
- A set of techniques to raise the ceiling of the expressiveness of immersive authoring tools, including the ability to create reactive behaviors and to programmatically create and destroy objects in a scene.
- An immersive authoring tool named FlowMatic that integrates the above techniques and provides the first visual representation of Functional Reactive Programming (FRP) in immersive environments.

- A qualitative comparison study of FlowMatic and a desktop-based authoring tool demonstrating its usability and benefits and a suit of example applications demonstrating its expressiveness.
- A new multi-user multi-branch VCS named VRGit that facilitates design exploration, rapid iterations, and collaborative content creation in VR.
- Results and design insights gained from an exploratory lab study that evaluated the usability and utility of the VCS for content creation in VR.
- VRCopilot, an immersive authoring system that integrates a set of techniques enabling users to interact and co-create with pre-trained generative AI models in virtual immersive environments.
- Empirical results gained from two user studies that provide insights on user experiences such as perceived agency and creativity, as well as potential and challenges of human-AI co-creation in immersive authoring workflows.
- An end-user creation tool named Auggie that aims to encourage effortful communication via digital handcrafting in AR.
- Results and design implications gained from a two-week field study that evaluated this system's potential to encourage effort on an immersive medium and create meaningful interactions.
- An end-user creation tool named Jigsaw that makes it easier to create immersive stories combining AR and IoT.
- Findings from a lab study that highlight the advantages and difficulties of this multi-user immersive storytelling experience.

1.5 Outline

The rest of the dissertation is structured as follows: Chapter 2 provides background information on end-user creation tools for immersive experiences and visualization and interaction techniques in immersive environments in general. Chapter 3 introduces a qualitative user study of the state-of-the-art immersive authoring tools that utilize the visualization of dataflow programming. Chapter 4 introduces an immersive authoring tool that raises the ceiling of the expressiveness by allowing users to edit reactive behaviors of object via visual

FRP in VR. Chapter 5 presents a version control system called VRGit that provides content management and exploration, and workspace awareness during collaborative content creation in VR. Chapter 6 presents how current generative models can be integrated to improve the overall content creation experience, discusses users' strategies of collaborating with AI during immersive content creation. Chapter 7 presents an AR-based communication system called Auggie for end-users to to encourage effortful communication via handcrafting AR experiences for their partners. Chapter 8 examines an end-user creation tool named Jigsaw for creating immersive stories that combine AR and IoT, and discusses its potential benefits and challenges for immersive multi-user storytelling experiences.

CHAPTER 2

Background

In this chapter, I begin by positioning this dissertation within the context of prior research. Each subsequent chapter addresses specific related work in detail, so to avoid redundancy, this chapter focuses on providing a high-level overview of end-user creation tools for immersive experiences and visualization and interaction techniques in immersive experiences.

2.1 Authoring Tools for Immersive Experiences

Despite the rapidly increasing interest in developing AR/VR applications, previous research has revealed that creators frequently encounter various difficulties in AR/VR authoring [14]. There has been a long body of work on authoring tools that allow end-users to build the virtual world. We first focus on immersive tools and distinguish between immersive *modeling* tools for creating static 3D scenes and immersive *authoring* tools for creating dynamic 3D scenes, both within immersive environments. In addition, we discuss 2D and hybrid authoring interfaces for authoring immersive experiences.

2.1.1 Immersive Modeling Tools

Commercial 2D modeling software (e.g., Maya, Blender, Reality Composer) has been popular for creators of different levels of technical experiences to create 3D models for years. However, a wealth of spatial information is lost since users are constrained to view and interact through a 2D window. Previous work has thus explored immersive modeling for building static 3D scenes directly in the immersive environment by proposing intuitive interaction techniques that can leverage users' spatial reasoning skills [31, 172, 206, 173, 124, 95, 171, 188]. One of the earliest attempts to achieve this was 3DM [31], a HMD-based modeler that allows users to build and view 3D models in the 3D virtual environments. Along this line of research, ISAAC [172] introduces more intuitive forms of interactions in 3D and adds constraints to the interactions for accomplishing more precise work. Mine et al. later proposed an approach that combines precise 2D touch surfaces and 3D bimanual spatial inputs to build complex 3D models in VR [173]. Lift-Off [124] explored generating 3D models from mid-air 2D sketches drawn by users. Commercial applications such as Google Tilt Brush [95] have enabled compelling immersive 3D sculpting experiences for end-users to craft static 3D models. In addition, platforms such as Minecraft have enabled multi-player collaboration for crafting 3D scenes directly in VR [53]. More recently, XRDirector supports collaborative creation of immersive experiences among multiple designers in both AR and VR, and uses demonstrations and simulates interactive behaviors in a Wizard of Oz style [182]. However, a key limitation of the above systems is that they cannot create interactive experiences (that are not Wizard of Oz), where virtual objects can be automated to react to users' actions or system events in the final experience.

2.1.2 Immersive Authoring Tools

Immersively adding interactivity and functionality to objects in VR can be difficult since it normally requires writing text-based code to define logic, such as triggering reactions in response to system events. Text-based programming languages are difficult to present in VR, because (1) many VR systems have lower resolution displays that are acceptable for graphics but not pages of text and (2) text entry in VR can be challenging. Further, text-based languages usually have a steep learning curve for end users. Therefore, some previous work has explored incorporating VPLs, especially dataflow programming languages, into immersive authoring systems [239, 153, 151, 74, 285]. Steed et al. [239] were among the earliest to introduce the concept of using a visual dataflow representation within the virtual immersive environment to define behaviors of objects. In their system, users can draw wires to connect virtual objects and virtual representations of input devices in the immersive environment. The data would then propagate along the wires across different objects in the scene, thus specifying their configurations. Lee et al. took the idea of immersive dataflow programming further by providing different properties of objects and computational primitives in the virtual dataflow representation [153, 151] and coined the term "immersive authoring". More recently, Ens et al. [74] built an immersive authoring system using a visual dataflow representation in VR for specifying behaviors of Internet of Things (IoT) devices. This pattern of embedding dataflow programming languages in the 3D immersive environments was found to be intuitive and easy for both novice programmers and end users [153]. However, by using basic dataflow programming, these immersive authoring tools can only express a limited set of static relationships among pre-defined objects in a scene. More importantly, they cannot define reactive behaviors of objects that come with a rich set of system events (e.g. collisions, user actions, and state changes) and behaviors triggered by those events (e.g. color changed when being selected). More recently, there are systems proposed that use a visual event-trigger representation for authoring reactive immersive experiences focusing on specialized applications such as IoT and free-hand gestures (e.g. [267, 266, 290]). One system introduced in this dissertation, FlowMatic, extends prior work by integrating concepts from Functional Reactive Programming (FRP) and providing a rich set of programming primitives and intuitive interactions for authoring general immersive experiences [286].

2.1.3 2D and Hybrid Authoring Tools

Besides immersive tools mentioned in previous sections, another line of work for authoring immersive experiences focuses on building 2D or hybrid interfaces for people to create immersive experiences. Typically, creating an immersive experiences requires creating virtual objects with their properties, arranging them in the scene, and programming their behaviors in each frame. While some afromentioned 2D tools such as Blender can support the creation and manipulation of 3D models, game engines such as Unity [260] and Unreal [261] are currently the most popular tools for programming immersive experiences. In recent years, the advances of WebVR have also given rise to libraries and frameworks such as Three.js [33] and AFRAME [1], which enable developers to build VR scenes as web applications that can be loaded by web browsers. However, all the above tools for programming VR scenes involve arguably complex user interfaces and imperative programming languages such as C# and JavaScript, which require extensive training. Therefore, much of the efforts in the HCI community focus on how to make programming immersive experiences easier, which has helped transform what was considered the "expert-only" task of programming to something that anyone can do (e.g., [200, 131, 220, 2]). For example, earlier systems such as VRML97 [37] and X3D [28] also provide a declarative format for the description of 3D content and utilize event passing mechanisms to define user interaction. Alice [200] is a 2D block-based programming environment that enables users to rapidly prototype 3D animations. Since then, many commercial applications have sprung up (e.g., graph-based programming in Lens Studio [121], Blender shader graph [21], Unreal Blueprints [2]) with similar goals in mind. While the above tools mostly focus on novice programmers with some technical experiences, another line of research is focuses on designers to craft immersive experiences using various modalities such as sketches [155, 183] and clay [184]. For example, 360Proto supports sketches on paper as background for authoring 360 experiences [183]. Pronto enables capturing 3D information that allow designers to navigate a 2D video frame in a 3D space and create of spatial layers for sketching and drawing [155]. ProtoAR combines modalities including paper, Play-Doh, and mobile devices to enable early prototyping of AR experiences [184]. However, all of the above systems produce a non-interactive video prototype as their output. Recent authoring tools, along this line, have therefore explore visual representations that allows designers to create interactive immersive experiences [220, 249, 154, 280]. For example, Saquib et al. [220] developed a 2D authoring tool that uses a dataflow representation to bind user inputs with graphical effects for AR presentations. RealitySketch analyzes and visualizes responsive graph plots of behaviors of physical objects and bind them to digital sketches created by end-users [249]. More recently, ProInterAR [280] enables users to construct interaction scenes by creating immersive content from the view of an AR-HMD and to script interactive behaviors by stacking blocks from a tablet UI.

2.2 VR Visualization and Interaction Techniques

Our work also builds on prior visualization and interaction techniques for object manipulation and navigation in VR. Prior work has proposed several techniques for interaction and navigation in 3D scenes of large distances [168, 207, 203, 242, 146, 23]. For instance, Mackinlay et al. propose the using teleportation to navigate large virtual workspaces [168]. Kunert et al. use photos of 3D scenes as portals that allow users to navigate in space and time [146]. To interact with objects at a large distance, "go-go" interaction uses the metaphor of interactively growing the user's arm to interact with distant objects in a virtual environment [207]. Stoakley et al. introduced the concept of World in Miniature (WIM), which enables both navigation and interaction in a large VR scene. A WIM represents the virtual environment and allows users to manipulate objects offered by the miniature, or rapidly teleport in the virtual environment by selecting locations directly in the miniature [242]. It also has the benefit of allowing users to see a preview of the immersive virtual environment without having to travel back and forth between different views. In this dissertation, we contribute to the literature by using the techniques of WIM and portals in a few designed artefacts such as VRGit and VRCopilot. We also extend the concept of portals to communicating and sharing views between collaborators in VRGit.

CHAPTER 3

Benefits and Challenges of Immersive Authoring: A Qualitative Study[1]

3.1 Introduction

VR applications can enable more natural human-computer interactions by matching the computer's representation of an environment with the spatial processing capabilities that humans have evolved over thousands of years [128]. By giving users a sense of presence and immersion, VR applications can nearly eliminate the gulfs of execution (how users translate intent into action) and evaluation (how users understand the state of a system) [187]. Although using VR applications can be natural and intuitive, creating VR applications requires specialized knowledge including advanced knowledge of imperative programming languages, 3D modeling, reactive programming, and geometry. One possible solution to the challenges of authoring VR content is to create VPLs for immersive 3D environments—to allow programmers to create content directly while immersed in VR. This paradigm is called *immersive authoring* [153, 152]. Immersive authoring tools have several potential advantages over traditional tools for creating VR content. First, immersive authoring environments can be intuitive, as they allow users to manipulate programming primitives through direct manipulation [225]—reducing the gulf of execution. Second, immersive authoring environments allow users to evaluate their code as they write it in the VR environment [152]—reducing the gulf of evaluation. Finally, by situating programs in an easily navigable 3D world, immersive authoring tools can leverage our natural spatial reasoning capabilities [128].

In this chapter, we evaluate the challenges and benefits of immersive dataflow programming tools. One of the earliest attempts to achieve this was Steed et al.'s dataflow representation for customizing behaviors [239]. Researchers have since built several immersive authoring systems, including iaTAR for creating AR scenes [153, 151], Ivy [74] for program-

[1]Portions of this chapter were adapted from [285]

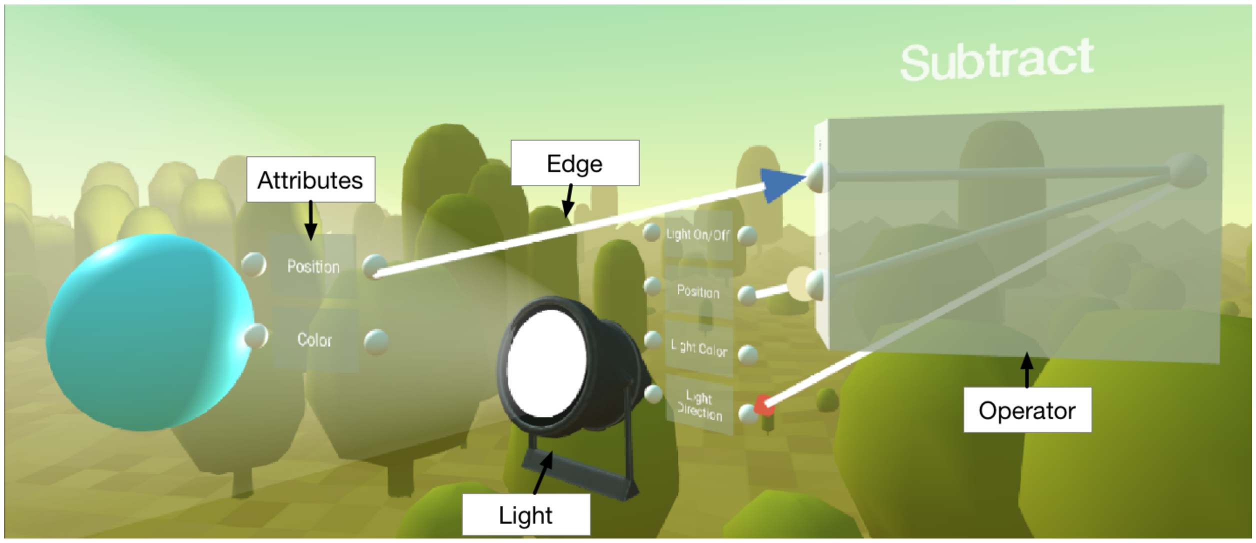


Figure 3.1: A screenshot of our immersive dataflow programming tool. The directed arrows specify edges that determine the direction of data propagation. Operators accept any number of inputs and produce one or more outputs. The operator shown subtracts the position of the light from the position of the sphere, producing a direction from the light to the sphere. The result of the operator goes to the *direction* attribute of the light. This program thus creates a scene where the light always shoots at the sphere, even as the sphere and light move.

ming IoT devices, and Soundstage for creating music [191]. Each of these systems uses dataflow to represent behaviors. However, none of this prior work has studied the usability of dataflow in their immersive authoring tool, which is the focus of this chapter. Of prior immersive authoring systems, only two (Ivy and Soundstage) run on modern VR hardware and only one (Soundstage) is publicly available. However, Soundstage was designed for authoring music. Thus, we built a new VR immersive dataflow programming language to use in our evaluation[2]. However, our findings are generalizable to other immersive dataflow authoring systems, which use similar paradigms and interactions. The results of our evaluation provide design insights that have implications for future immersive authoring tools.

3.2 Immersive Authoring System

In our immersive authoring system, each operator is represented as a translucent box that takes inputs and produces outputs (Fig. 3.2d, 3.2e, 3.2f). Each input has a connector on the left side of the box and each output has a connector on the right side. For example, the *Subtract* operator (Fig. 3.2d) has two connectors on the left: + (plus) and - (minus),

[2]Our immersive authoring tool is open source and publicly available: http://raynezhang.me/files/ImmersiveAuthoring.zip

where data inputted through the + connector will be added to the result and data inputted through the - connector will be subtracted from the result. The final result will be propagated through the output connector on the right.

Users can also create behaviors that depend on the position of their headset and controllers through *avatars*, which are proxies of their headset and controllers (Fig. 3.2c). Avatars contain output nodes for the position, rotation, and each button on these devices. Each avatar can be viewed as a node and by drawing edges between the avatar and other virtual object in the scene, users can create interactive scenes where the attributes of the virtual objects will depend on the the user's tracked devices (i.e. the headset and the controllers).

Users use two controllers to interact with the immersive authoring tool. Users can create objects, avatars, and operators through a palette tool menu, which is "attached" to the user's left hand controller and whose items can be selected by pointing (via raycast) and pressing a trigger on the right hand controller. The application employs a drag-and-drop interaction for drawing edges using the raycast and provides straight arrows as intermediate feedback (Fig. 3.3 C).

3.3 Method

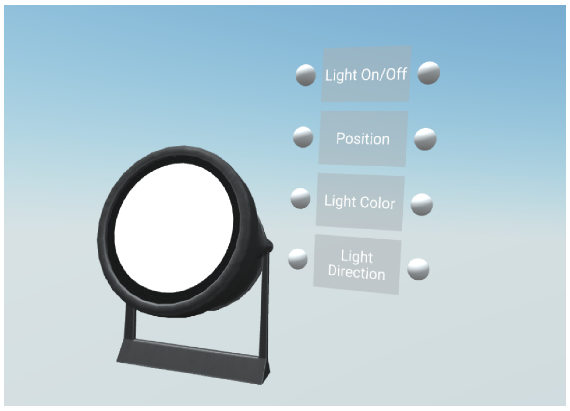
To better understand the benefits and challenges of immersive dataflow programming, we conducted a user study with our immersive authoring tool. Although this user study was only conducted with our tool, we believe our findings are representative of other immersive authoring systems [239, 153, 151, 74, 191], which also employ similar dataflow metaphor, visualizations, and node placement features in VR.

We recruited 7 participants (2 male, 4 female, and 1 prefer not to say), ages 20–27 ($\mu = 24.1$), from the authors' university. All participants had at least basic programming experience (having completed at least one programming class). Two participants also had experience creating VR applications. We compensated every participant with $25 in cash for their participation. Our application was run on the Oculus Rift on a Windows 10 system with an Nvidia GTX 1080 GPU, Core i7 CPU and 16 GB RAM.

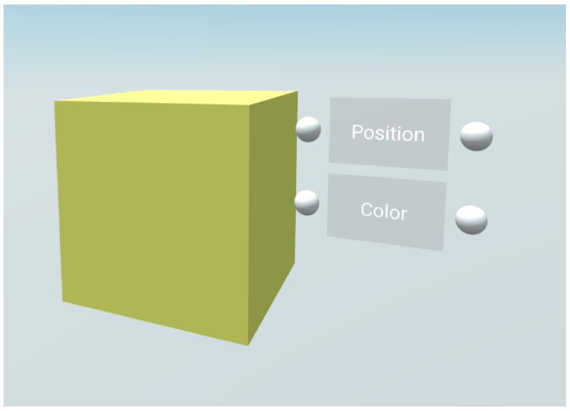
Every study lasted 90 minutes. Participants spent the first 30 minutes in a tutorial that walked them through a series of small tasks including creating and manipulating objects, drawing edges between nodes, and using different operators. The tutorial also gave participants a chance to ask questions and experiment on their own.

We then asked participants to perform two tasks:

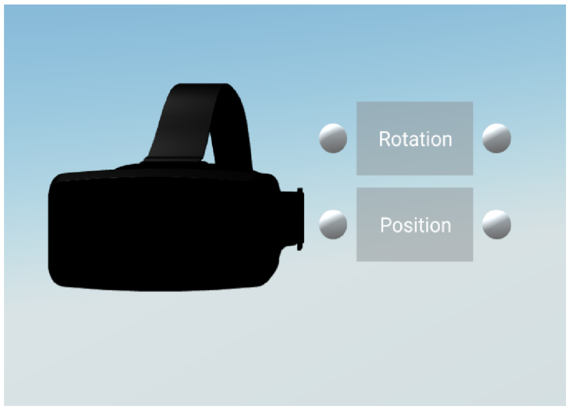
- *Task 1*: Create three spotlights and make them shoot at and follow the user. The three spot lights should emit light of random colors.



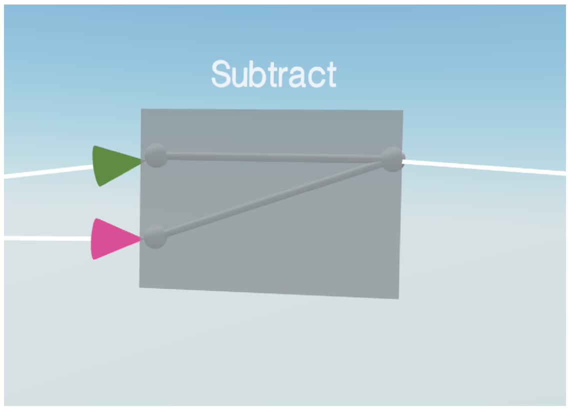
(a) Object: Light



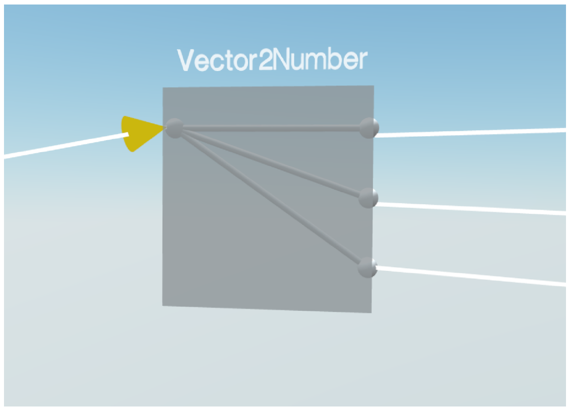
(b) Object: Cube



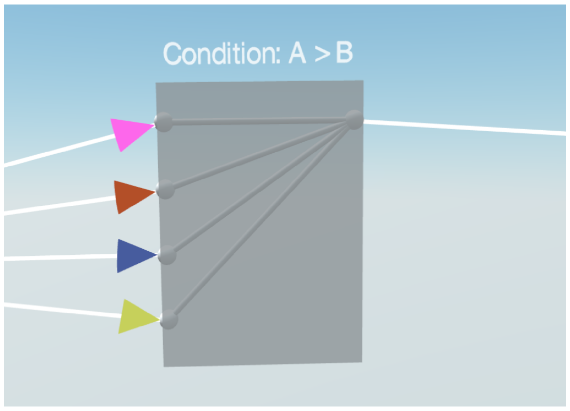
(c) Avatar: Headset



(d) Operator: Subtract



(e) Operator: Vector2Number



(f) Operator: Condition $A > B$

Figure 3.2: Examples of objects, avatars, and operators. Objects (a&b) have several attributes listed next to them that can be modified. Avatars (c) represent virtual proxies of users' inputs. Operators (d, e, f) are computational units that take inputs produce the results as outputs.

- *Task 2*: Create a scene where a spot light will shoot at a cube when the user's left hand is higher than the user's right hand, and the spot light will shoot at a sphere when the user's left hand is lower than the user's right hand.

We gave participants 15 minutes for each task and did not give them further instructions on how to complete the tasks unless they specifically requested help. We then conducted a one-on-one retrospective interview with each participant.

3.4 Results

Most participants expressed that the application is fun to play with:

> P3: *"I really like to play with (it). I think it's really good to explore. It was just really fun."*

Participants also generally enjoyed the immersive feeling of being able to place things freely and naturally in the 3D virtual space as if they were manipulating them in the real world:

> P2: *"I really enjoyed having stuff in a three-dimensional space... it just feels so real. "*

When being asked about any confusion about the application, all participants commented that drawing edges between nodes is difficult and annoying since they often missed the target connector when it became too small and hard to aim at in the scene:

> P1: *"The aiming of putting a line on a circle was annoying. I messed up five times or something..."*

Another difficulty from most participants is the struggle to follow the execution of the program when it gets more complicated and the lines get cluttered. Based on that, some participants expressed desire for adding secondary notations (e.g. comments, annotations, etc.) or grouping nodes into sub programs:

> P4: *"I think it will be helpful to let the user to group things together. For example, for all the condition patch(es), I can group them together and have a note like "this is gonna compare positions" "*

When being asked to compare the application with their previous experiences in text-based programming, most participants commented positively on this application and all participants expressed that the application is easier for beginners and is therefore suitable for educational use:

> P2: *"This tool reminds me of this thing called Alice. I can definitely see it being a potential educational tool. I think it can make things so much easier."*

3.5 Discussion

In this section, we discuss several challenges and benefits of immersive dataflow programming based on the aggregated results above. For each challenge, we provide our insights and design implications based the study.

1) Layout Management: From the results, users were not strategic about where they placed the nodes in their dataflow programs. However, they typically placed related nodes next to each other, either horizontally or vertically. They normally placed nodes in a from-left-to-right order based on the direction of the data propagation. At the same time, users indicated that lines became clustered and hard to follow when the program became more complicated. This is also a challenge in general dataflow programming languages but a compounding problem was that edges could be occluded and hidden because of the wealth of depth information. One design implication for managing the dataflow program layout is to automatically sort and place the nodes and edges, according to some participants. Another implication is to support objects customization for the users. Specifically, participants commented that being able to group related nodes and edges together is helpful for keeping track of the dataflow program. They also expressed that being able to add annotations or comments would be helpful for understanding each part of the dataflow program.

2) Drawing Edges: During the retrospective interviews, most participants preferred drawing edges through direct manipulation (Fig. 3.3 A), as opposed to using raycast to aim at target connectors (Fig. 3.3 C). There are two reasons for this. One reason is that through direct manipulation users can feel more immersed—as if the wire is in their hands. The other reason is that it is easier to aim and connect when the connector is close to the users. This is a challenge that is specific to 3D immersive environments since users draw edges in a 3D space using the controllers held in their hands as opposed using the 2D Window Icon Menus Pointer (WIMP) interfaces.

We therefore propose four mechanisms for drawing edges, as shown in Fig. 3.3. Each mechanism has its own tradeoffs. Users can draw edges through direct manipulation as if they are holding the wires. The first method (Fig. 3.3 A) allows users to draw edges through direct manipulation as if they are holding the wires. One benefit of this method is that it is intuitive by allowing users to connect edges in the same way that they do in the physical environment. Another benefit is that it allows customized shapes of the edges, which is helpful in avoiding occluding edges with each other. The drawback of this method is that it is hard to draw an edge between two objects that are far away from each other without moving in the virtual world. The second method (Fig. 3.3 B) allows users to draw edges using a proxy at the fixed distance to the controller. This method offers custom edges but

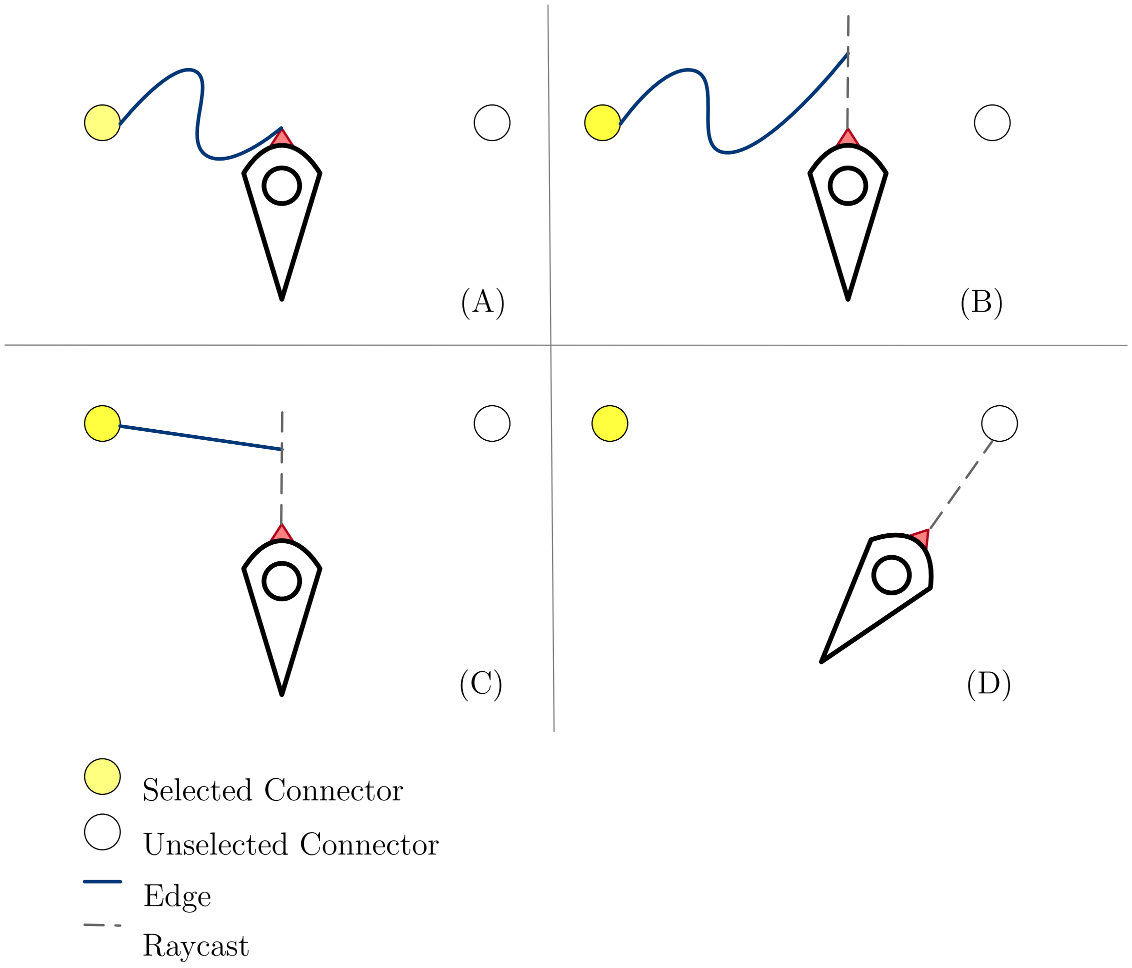


Figure 3.3: An illustration of four alternative techniques for specifying edges between nodes in VR using a controller. Edges can either be drawn as custom shapes (A&B) or as a straight line between nodes (C&D). Users could draw edges directly with their controller (A); along two dimensions with a depth that the system computes (B); as straight edges between nodes while showing intermediate feedback (C); or by selecting source and target nodes with no intermediate feedback (D).

also requires them to move around the scene to connect distant nodes. The third method (Fig. 3.3 C) allows users to draw straight edges using the raycast and shows the straight edge as intermediate feedback. The benefit of this approach is allowing users to draw lines beyond their reach without moving in the virtual world. However, it is hard to aim at the connector that is too far away and appears very small in the scene. The last method (Fig. 3.3 D) is similar to mouse-clicking, where users will click at the first connector and then the second connector in order to draw an edge. This method produces no intermediate feedback and avoids the drag-and-drop interaction. However, the drawback of this approach is the lack of direct manipulation.

3) Navigation: Changing the user's viewpoint of the dataflow program is also challenging when the whole program is embedded in an immersive 3D space. This is also different from dataflow programming languages on 2D WIMP interfaces where users can pan and zoom with the mouse. Instead, in the immersive virtual environments, users have to move around in the virtual world in order to navigate through the program. This usually causes the feeling of lack of control of their own bodies when the locomotion of self affects the viewpoint of the dataflow program. To cope with this challenge, we propose a 2D map-like dataflow representation design, where users can zoom in, zoom out, and move the whole diagram without changing their positions in the world.

4) Immersion and Natural Interaction: Immersive dataflow programming tools allow users to manipulate and place things freely in the 3D space in the same way that they interact with the physical world. Multiple users expressed that being able to create and place objects (such as cubes and lights) wherever they want is the most enjoyable part of the immersive dataflow programming tool. Participants' sense of immersion was also enhanced by the immediate feedback they received. Users can see the changes instantly after drawing an edge between two nodes, which makes it faster to prototype VR scenes and faster to make changes to the existing program.

5) Potential for Educational Use: Most participants agree that this tool is easier for beginners and has the potential for educational use. This may be due to that existing imperative programming tools have steeper learning curve than immersive dataflow programming tools. Some participants also expressed that they found the immersive tool to be fun and engaging to interact with.

Based on the results of our study, we do not see an advantage in allowing users to place dataflow objects themselves in 3D space. Users in our study did not tend to spend the time to organize their dataflow nodes and occlusion could make it difficult for them to follow the edges. However, we do see advantages in including dataflow languages inside of immersive authoring environments. Participants specifically liked the quick feedback loop and the ability to connect dataflow outputs and inputs directly to objects in their environment and objects that represented the user.

Thus, we feel that dataflow is still an effective option for immersive authoring. However, there is little benefit to giving users complete 3D freedom in dataflow authoring. Instead, we propose dataflow interactions that happen in 2D—where all of the nodes and operators are visible on a 2D plane but can be connected to objects in the 3D space. For example, one could imagine a "breadboard" metaphor where users can see their dataflow program on a 2D plane but they can connect the output of their dataflow diagrams to objects in the virtual world.

3.6 Limitations & Conclusion

There are several limitations of the presented study. First, we only recruited participants who were willing and able to use VR, which may bias the kinds of feedback that participants give. Second, we performed a short-term study and received only the first impression of the immersive dataflow programming tool on the participants. Despite having the training session, some participants expressed that it was difficult to get familiar with all the features in a short time. A longitudinal study would be necessary to better understand participants' learning curve.

In this chapter we presented an exploratory study analyzing the benefits and challenges of immersive dataflow programming tools. We also proposed several design implications based on the results. We believe that dataflow programming is an effective approach for immersive authoring and that there is ample room for future improvement to address the design challenges brought by the freedom of 3D space.

CHAPTER 4

FlowMatic: Raising the Expressiveness of Immersive Authoring[1]

4.1 Introduction

VR applications are uniquely compelling for end users because they enable natural and intuitive interactions in a 3D space. For novice developers, however, VR applications are uniquely difficult to program because they require advanced knowledge of imperative programming, 3D modeling, geometry, and reactive programming. Even desktop-based VPLs built for non-experts, such as Unreal Blueprints [2], require that users mentally translate between 2D and 3D representations and predict how their code will execute in VR. Immersive authoring enables a faster, more intuitive workflow by allowing users to manipulate programming primitives in VR through direct manipulation [223]. Immersive authoring also enables a more fluid workflow by offering immediate feedback as the user designs and programs a virtual scene. Several immersive authoring tools allow users to create static 3D models or sketches [206, 13, 95, 188], and others allow programmers to declare dynamic relationships between object properties [239, 152, 74, 285]. However, existing immersive authoring systems cannot express *reactive* behaviors, a fundamental requirement for most practical VR applications. Reactive behaviors are behaviors where the application responds to events like user actions, system events, or collisions. Typically, programming reactive behaviors requires writing text-based imperative event-action code, which is difficult to represent effectively in VR.

In order to raise the ceiling of what immersive authoring systems can express, we introduce FlowMatic, which allows novice programmers to define reactive behaviors and prototype interactive VR scenes. Our techniques build on FRP, a declarative paradigm that treats discrete events (e.g., user input) as first-class data that can be referenced, managed, and

[1]Portions of this chapter were adapted from [286]

transformed along with “signals”, which represent continuous values (e.g., the user’s position). We also introduce techniques for dynamically creating new objects in the environment, abstracting and re-using functionality, visually representing types, and easily importing 3D models into a scene. We also propose interaction techniques that take advantage of the intuitive and direct interactions that VR affords.

4.2 Related Work

In addition to related work in immerisve authoing, as described in Chapter 2, we describe in this section dataflow programming and FRP that are related to our approach.

4.2.1 Dataflow Programming

Dataflow programming languages have a long history, beginning with Bert Sutherland’s Ph.D. thesis [248]. The dataflow model is represented by a directed graph, consisting of data sources, data sinks and nodes. The nodes are primitive operations such as arithmetic and comparison operations. The direction of each edge represents the direction of the data propagation across different nodes. Researchers have then improved and applied dataflow programming in various domains [136, 180, 114, 149, 239, 152, 74]. For example, Show and Tell [136] was one of the earliest *visual* dataflow languages targeted primarily at elementary school children. Lau-Kee et al. [149] built a visual programming tool and environment for interactive image processing. Successful commercial software that incorporates visual dataflow programming—such as LabView [123] and Max MSP [5]—has also been popular in the domains of hardware and music respectively.

Despite being intuitive for end users, basic dataflow programming has several weaknesses of expressiveness such as visual cluttering when scaling to complex dataflow graphs with lots of nodes and edges [233], and lack of support for control issues such as state transition and initialization [279]. We address the visual cluttering issue in the domain of authoring VR scenes by introducing techniques of abstracting and re-using behaviors. We further adopt the FRP model and make explicit controls for defining reactive behaviors and initializing 3D objects at run time.

4.2.2 Functional Reactive Programming

The concept of reactive programming has been proposed for implementing event-driven applications based on synchronous dataflow programming paradigms but with relaxed real-time constraints [17]. FRP [72] integrates reactive programming into functional programming.

The basic primitives in FRP are *behaviors*, which refer to different states of the system defined by continuous time-varying values, and *event streams*, which refer to infinite streams of discrete values. Users can define the behaviors by specifying how they should change in response to the incoming events. Event-driven FRP (E-FRP) [263] is considered a more efficient variant of FRP that can guarantee real-time execution of FRP programs using two phases: 1) comparing the current state with the prior state of the computation to see whether they are the same, and 2) updating the current state. The FRP approach is suitable for a variety of areas such as interactive 2D animations [72], web applications [55], and data visualizations [221]. To our knowledge, FlowMatic is the first to exploit the expressiveness of FRP in authoring VR scenes and further visualize FRP semantics in the 3D space. FlowMatic also proposes a set of domain-specific operators for programming VR applications.

4.3 System Design

We have three primary design goals for FlowMatic:

- To raise the ceiling of the expressiveness of immersive authoring tools.
- To minimize its learning curve by relying on a small number of conceptual primitives that behave consistently.
- To build visualizations and controls that are appropriate for state-of-the-art VR systems.

FlowMatic starts with a standard dataflow model that allows users to define relationships between objects in the scene and the state of the user. The top-level primitive of FlowMatic is called a *scene* (analogous to a “program”). Every scene can contain any number of elements from the following list:

- **Models** represent 3D shapes in the scene that are visible to users. Every model contains:
  - **Attributes**, which control how the model is displayed. For example, a model representing a stereo box might have attributes controlling its volume, position, and color. Attributes can reference and be referenced by other elements in the scene.
  - **Methods**, which are discrete actions that a model might take, such as animations. Like attributes, methods can reference or be referenced by other elements in the scene.

- **Data sources** are ways for data to enter the application. Like models, data sources contain attributes that can be referenced by other elements in the scene. Unlike models, however, data sources are *only* visible to the programmer—not for users of the applications they create. FlowMatic contains three kinds of data sources:
  - **Avatars** provide information about the state of the user, as first introduced by Steed and Slater [239]. For example, avatars allow programmers to reference the position of the user's headset and controllers in the virtual scene, the buttons the user is pressing, and more.
  - **Constants** represent values that never change but need to be referenced as part of the scene. For example, in the expression "`x+5`", `5` is a constant.
  - **Randomized Generators** provide randomized data that can be referenced for non-deterministic programs. This is analogous to how Unix systems can pipe data from `/dev/random`.
- **FRP diagrams** represent the logic of the scene—how elements change dynamically and react to user and system events. The FRP diagram contains four kinds of elements:
  - **Operations**, which transform and manipulate the data[2]. Operations can accept any number of arguments and produce any number of outputs.
  - **Nodes**, which are elements that can be referenced in the FRP diagram. Every node is part of an operation, model, or data source.
  - **Edges** between nodes that specify how data flows within the FRP diagram.
  - **Abstract models**, which are models that can programmatically be added and removed from the scene at runtime. Abstract models have the same attributes and methods as normal models but are not instantiated until the abstract model is passed into the `create()` operation.

Although FlowMatic is an immersive authoring tool, several aspects of immersive authoring are beyond its scope—particularly the ability to define new 3D models and to define rendering functions that specify how to display a given model given its attribute states. However, future iterations of FlowMatic could incorporate such features by building on prior work, such as TiltBrush [95] and Medium [188].

FlowMatic consists of three User Interface (UI) components, as Figure 3.1 (a) shows. The palette menu (Figure 3.1 (i)) allows users to search, select, and import 3D models to the

[2]The operations that FlowMatic includes are based on those of the RxJS JavaScript library [218].

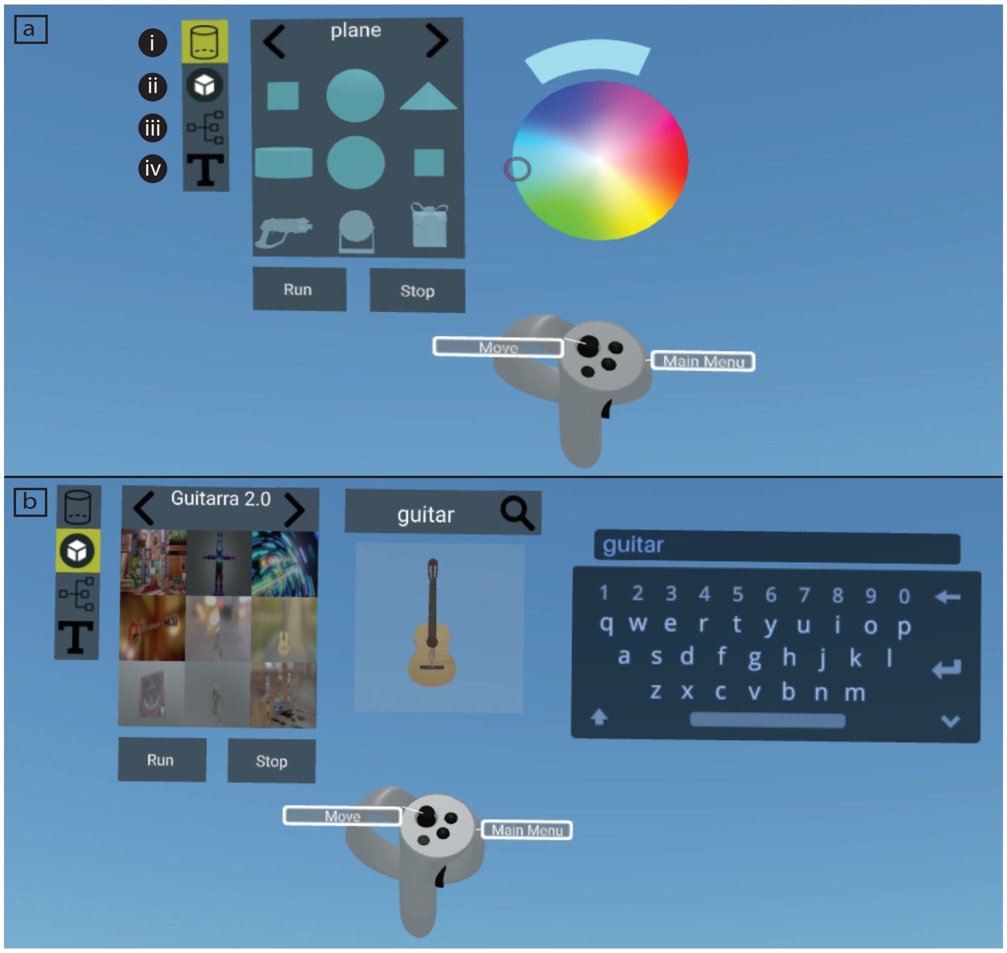


Figure 4.1: The palette menu of FlowMatic. (a)/(i) is the interface that allows users to import primitive models and select their colors using the color palette. (b)/(ii) is the interface that allows users to browse, search for, and import models from online. (iii) is a toggle for displaying the FRP diagram. (iv) allows users to create text elements in the scene.

scene. The canvas (Figure 3.1 (iii)) shows the FRP diagram that the user creates. The toolbox (Figure 3.1 (ii)) allows users to select from a set of data sources and operators used for the diagram. We will delve into the design of each component and use an interactive example to showcase the workflow of FlowMatic.

4.3.1 Palette Menu

Palette Tool Menus are widely adopted in popular immersive authoring tools such as Google Tilt Brush [95] and Microsoft Maquette [171]. In FlowMatic, the palette tool menu is always attached to the left controller, as Figure 4.1 shows.

4.3.1.1 Import 3D Objects

FlowMatic allows users to import both primitive models (e.g. cubes, spheres) and models from Sketchfab[3], a popular library of 3D models, as Figure 4.1 shows. Users use a raycast shooting from one controller to select different models and then import them to the VR world by pressing a button on the controller. Any animations associated with imported models are represented as methods in FlowMatic, which execute the animation when called. FlowMatic also includes models for displaying text (for example, to display a user's current score in a game).

4.3.1.2 Stop Mode and Run Mode

Although users enjoy *liveness* (where they can see the output immediately after they write part of the program), prior work has found that they prefer having a button that allows them to switch between running and editing the program [285]. This is partly because they may accidentally trigger the actions in their application while they are using the controllers to manipulate the programming primitives. As Figure 4.1 shows, users can switch between *stop* mode and *run* mode, where *stop* mode shows both 3D models and the programming primitives (e.g. operators, data sources, or attributes), whereas *run* mode only shows the 3D models.

4.3.2 Functional Reactive Programming Diagram

In order to represent discrete events, we incorporate concepts from FRP. We choose FRP as a complement to the state-of-the-art immersive authoring tools for several reasons. First, VR applications, like most graphical applications, are typically event-driven [73]. Although previous immersive authoring tools cannot express various system events, FRP models event streams as a first-class abstraction. Second, from an end user's perspective, the benefits of FRP are similar to those in favor of declarative programming over imperative programming—ease of construction, composability, clean semantics, etc. Especially for immersive authoring systems, FRP enables end users to focus on "what" to present instead of "how" to present on the HMD, which they have neither expertise nor interest in. Lastly, FRP fits within the dataflow model but also provides more expressive functionality, such as the abstractions of event streams. In FlowMatic, the user edits the FRP diagram using a canvas, as Figure 3.1 (iii) shows. Users can also toggle whether the FRP diagram is shown (developer mode) or not (user mode).

[3] https://sketchfab.com/

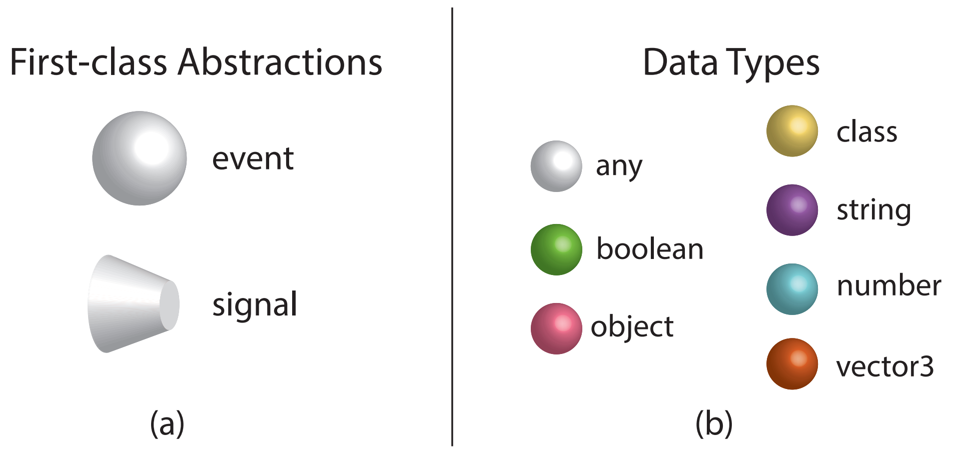


Figure 4.2: The type visualization in FlowMatic. We use shapes to represent first-class abstractions (a). We use colors to represent data types (b). The combination of shape and color represents the abstraction and data type.

4.3.2.1 Signals & Events

The essence of FRP is the notion of *signals* for representing continuous time-varying values (e.g. time, position, and rotation) and *event streams* for representing discrete events (e.g. button presses and collisions). These are the only two first-class and composable abstractions. Although signals are widely supported by the state-of-the-art immersive authoring tools, event streams are not. For example, using current immersive authoring tools, users cannot specify that something should happen *when* the user presses a button[4]. FlowMatic addresses this by modeling behaviors of objects using FRP's notion of signals and event streams.

Event streams can be attached to model methods (to specify when to call a particular method) or composed and manipulated into more complex event streams or signals. They can be emitted from the user's input devices (e.g., controller button presses) or from system events (e.g., collisions, timed intervals, or animations).

4.3.2.2 Type Visualization and Constraints

To minimize the learning curve for adopting the concept of FRP, FlowMatic also enables type visualizations and constraints. Specifically, we represent data types using color and first-class abstractions using shapes, as shown in Figure 4.2. In addition, we allow type constraints on the connections so that an edge can only establish when the types of the two ports are compatible, as shown in Figure 4.3 (a) & (b). This visual feedback can help users avoid type errors effectively. We also introduce polymorphic ports, as shown in Figure 4.3

[4]This is different from expressing a relationship that holds *while* the user presses a button, a continuous event.

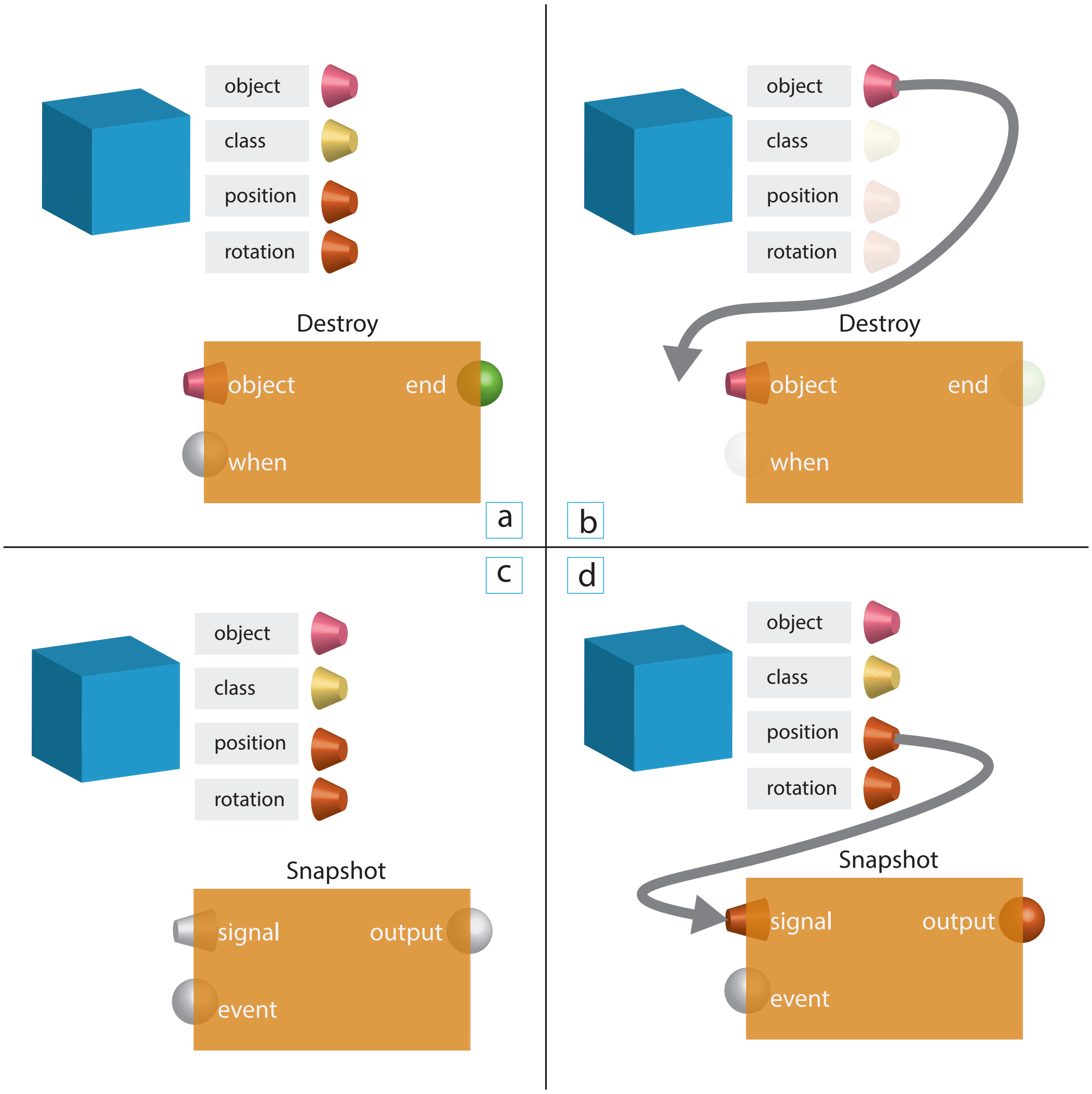


Figure 4.3: The type constraints in FlowMatic. (a) shows the state before making a connection. (b) shows that when making a connection of type *object*, all ports of incompatible types will be transparent and unconnectable. (c) & (d) demonstrate polymorphic ports. The snapshot operator takes a snapshot of the input signal whenever the event is fired and outputs event streams of the same type as the input signal.

(c) & (d), where the type of a connector can be polymorphic according to the incoming data types.

4.3.3 Dynamic Operations: An Interactive Example

While the state-of-the-art immersive authoring tools allow users to define the behaviors of existing objects in the scene, they cannot dynamically operate on 3D objects, which means that users are not able to author scenes that can programmatically create or destroy objects, react to system events, or perform discrete actions. This is difficult in previous immersive authoring tools because they define behaviors using the dataflow model, which specifies continuous relationships and typically needs the objects to exist in the scene—dataflow connections rely on the object being visible.

In order to programmatically create objects in the scene at runtime, FlowMatic includes *abstract nodes*. Users can create abstract models as placeholders for objects that will be created in the scene at some point in the future. They can draw edges to and from these abstract models to specify dependencies and behaviors (for example, to specify the dynamics of where it should appear in the scene when it shows up). Finally, they can use the `create()` operator to create any number of instances of these abstract models. Conversely, the `destroy()` operator removes an existing object from the scene. It can destroy models that were created dynamically or 'regular' models that the user manually placed in the scene. We will illustrate this with the example of how Bob uses FlowMatic to progressively program a simple shooting game, which is impossible to build with previous immersive authoring tools. Note that in the example we cover only a subset of all the operators in FlowMatic but we demonstrate how a limited number of operations are sufficient for building such interactive behaviors.

The first feature towards a shooting game is to specify that a bullet should be instantiated at the gun tip whenever the player presses the *trigger* button on the controller. In order to achieve that, Bob first wants to get the position value of the gun tip every time the player presses the *trigger*. The first operator being used is called `snapshot()`. The `snapshot()` operator takes two inputs—an event and a signal—and produces one output. The functionality of the operator is to always take a "snapshot" of the signal's current value whenever the event fires. After dynamically getting the position value of the gun tip, Bob uses the `create()` operator to dynamically instantiate a sphere (that represents a bullet) at that position. The `create()` operator thus takes several inputs including a class that comes from an *abstract model* indicating what type of objects to create, an event that defines when to create it, and several parameters for the instantiation such as *position* and *scale*. The output of the `create()` operator is the instantiated object (an individual bullet).

The second step is to set the bullet's trajectory so it shoots along the gun direction and then destroy it when it collides with an object. Bob gets the instantiated object from the output of the `create()` operator and uses a `translate()` operator to translate the bullet. The `translate()` operator takes four inputs: an object that specifies which instantiated

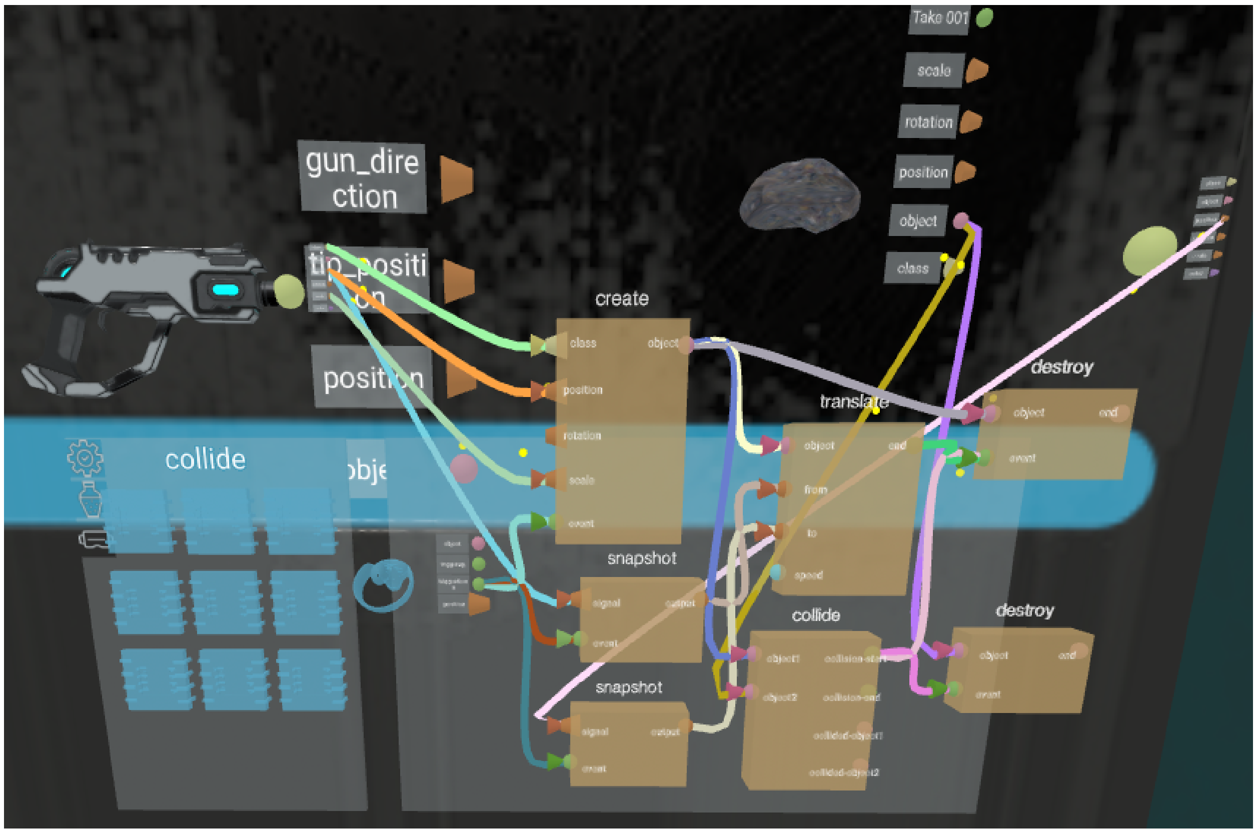


Figure 4.4: Resulting scene of a simple interactive example.

object to operate on, a *from_position* that defines the start position, a *to_position* that defines the end position, and a *speed* that defines the speed of the translation. Bob sets *from_position* to the position of the gun tip and *to_position* to the position of the destination (an abstract node). The output of the operator is an event that will emit when the translation completes.

As a final step, Bob wants to specify that when the bullet collides with an obstacle, both the bullet and the obstacle should be destroyed. The `collide()` operator is designed to detect collisions between two objects in the scene. It thus takes two inputs that specify the two objects respectively and produces four outputs—an event that emits when the collision starts, an event that emits when the collision ends, and the ids of the two collided objects. Bob uses the `destroy()` operator to specify that when a bullet collides with an asteroid, both should be destroyed. Figure 4.4 shows the scene resulting from this example.

4.3.4 Interaction Design

4.3.4.1 Direct Manipulation of Objects

We iterated our design to directly manipulate objects in VR by matching the direct manipulations that people perform physically in real life and preliminary feedback we gathered from user tryouts. Users use a raycast shooting from the right controller to aim and use the *trigger* button on the controller to select items on the menu, analogous to conventional WIMP interactions. Users also use the raycast to aim at connectors, press the *trigger* button to start drawing an edge, and release it when aiming at the target connector, analogous to conventional drag-and-drop interactions. Users can directly move an object and place it in the 3D space by holding and releasing the *grip* button on the controller. Users can remove an object or an edge by aiming at it with the raycast and pressing a button on the controller. They can also rotate and scale the object using the *thumbstick* on the right controller while holding the object. This is especially useful when creating *abstract nodes* as placeholders, since users can inspect the positions and scales of the models directly in VR.

4.3.4.2 Abstracting and Re-using Behaviors

FlowMatic enables users to define and re-use customized operators by taking abstractions of basic operators or data sources, as Figure 4.5 shows. To initiate an abstraction, users can 'pull' an existing operator closer to them using the thumbstick on the controller. The abstraction then contains the target operator. Users can continue to 'absorb' additional operators into the abstraction. The abstraction can dynamically update its inputs and outputs based on the operators being abstracted. Users can 'push' the abstraction back to the FRP diagram again using the thumbstick once the abstraction is done. Users can also save the abstraction in the toolbox for future use by pressing a button on the controller. These abstractions make complex FRP diagrams easier to read and allow users to build up higher levels of abstraction.

4.3.5 Implementation

FlowMatic is open source and publicly available for other researchers to build on and evaluate[5]. The front-end of FlowMatic builds on AFRAME, which in turn builds on Three.js and WebVR. The back end of FlowMatic uses Node.js and RxJS [218] to handle FRP logic.

[5]https://github.com/RayneZhang/FlowMatic

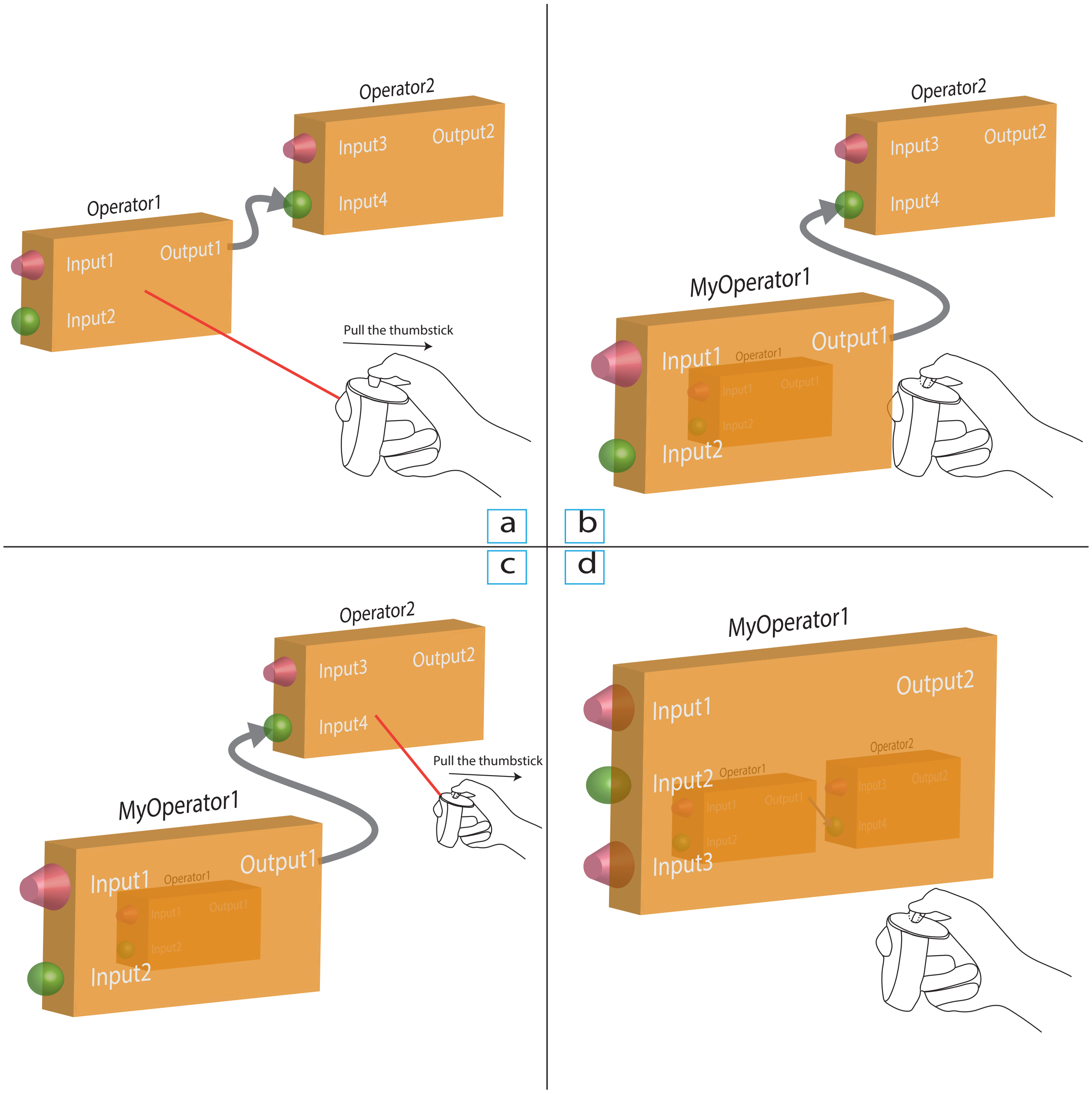


Figure 4.5: In FlowMatic, users can create and re-use new operators as abstractions. By 'pulling' an operator closer to the user using the thumbstick (a), they create a new abstraction (b). As they pull additional operators into the abstraction (c), FlowMatic automatically updates its inputs and outputs (d).

4.4 Study

We conducted a user evaluation to evaluate the learnability, efficiency, and usability of FlowMatic. Specifically, we designed our evaluation (1) to see whether participants are able to build VR applications with FlowMatic, and (2) to gain insights into the advantages, disadvantages, and usability of immersive authoring systems.

Table 4.1: The first task given to the participants in the evaluation: Stage Animation

Steps	**Description**
Step 1	Create three light models in the scene
Step 2	Place the lights: 1. Each light must be next to each performer on the stage 2. The lights should appear on the stage floor
Step 3	The light in the middle should always aim at the user even if the user is moving
Step 4	The light in the middle should be turned off when the user presses the trigger button on the left controller, and be turned on when the user releases the trigger button on the left controller

Table 4.2: The first task given to the participants in the evaluation: Shooting Game

Steps	**Description**
Step 1	Create a gun model in the scene
Step 2	To dynamically create a sphere at the gun tip, when the user presses the trigger button on the right controller.
Step 3	The created sphere should translate from the tip position to the destination that is along the gun direction.
Step 4	The created sphere should disappear after translating to the destination

Because there is no existing immersive authoring tool that allows users to program reactive behaviors, we chose AFRAME [1], a popular JavaScript framework for programming web-based VR content, as a representation of conventional desktop methods for authoring VR. There are two reasons we chose AFRAME as a comparison. First, AFRAME uses entity-component architecture and an event-handler mechanism for programming VR applications, which have been a fundamental feature of developing 3D user interfaces including VR applications [73]. Second, AFRAME has been used for several research projects [184, 183], including FlowMatic, and has proven usable and capable of authoring event-driven behaviors.

4.4.1 Participants

We recruited 8 participants (6 female, 1 male, and 1 non-binary, age 20–26) to evaluate our system. All participants had at least a beginner level of JavaScript (have taken at least one web programming class) and half of them identified themselves as experts (have experience building websites using JavaScript and are very familiar with syntax frequently used in JavaScript). All participants reported having no or very limited experience in programming VR applications before. Participants were paid $25 USD for an approximately 120-minute study.

4.4.2 Procedure

We used two different study tasks and two systems with which to implement them (AFRAME or FlowMatic), all counterbalanced to control for learning effects. The study procedure consisted of three sessions: 50 minutes for training and experimenting with the first system, 50 minutes for training and experimenting with the second system, and 15 minutes for retrospective interviews and post-task questionnaires. In each 50-minute session, we spent the first 20 minutes helping the participants go through a tutorial of the system and then gave the participants 30 minutes to implement the task. The tutorial for AFRAME was a document that introduced the syntax and Application Programming Interfaces (APIs) necessary for programming VR applications. The tutorial for FlowMatic contained basic concepts and operators necessary for the experiment. Participants also did some exercises with each system to write basic features in addition to the tutorials, but none of the features were the same as the actual tasks. The task descriptions were the same regardless of the implementation system. Table **??** shows the descriptions for each task, which consisted of four steps.

After the training, we gave participants 30 minutes for the task in each condition and did not give them further instructions on how to complete the task unless they specifically asked for help or we noticed they had been stuck for more than 1 minute. Participants were allowed to freely use the tutorials for reference. In the AFRAME condition, the participants were allowed to copy the APIs from the document directly to the Integrated Development Environment (IDE). In both conditions, the participants were allowed to ask for clarifications on specific concepts, APIs, or operators covered in the tutorials. The researchers would then give the clarifications verbally. Questions that were unrelated to the contents of the tutorials, such as what the next step in the task should be, were counted as asking for help.

To make the comparison with AFRAME fair, we tried to provide the same level of abstractions of APIs. For example, when implementing the feature of translation from one position to another, the participants only needed to specify the *entity* (which will translate), the *from position*, and the *to position* in both systems. We also provided the same resources in terms of digital models. During the tasks, we observed how often participants made errors and what types of errors they made in both conditions.

After completing both tasks, we conducted a one-on-one retrospective interview with each participant to obtain feedback on how the tools compared and the usability issues. Participants next filled out a questionnaire about the systems. The questionnaire used a 5-point Likert scale (1–Strongly Disagree, 5–Strongly Agree), assessing the learnability, usability, and other metrics of FlowMatic.

4.5 Results

All participants were able to implement the given tasks using both systems. We observed that participants made more errors when using AFRAME and normally spent more time on fixing errors, though we did not quantitatively measure the time spent on error fixing. Some common errors when using AFRAME were binding event listeners to wrong entities, confusing `init()` with the `tick()` function, and forgetting to attach components to entities.

4.5.1 Retrospective Interviews

During the retrospective interviews, we asked participants about the comparison between the two authoring tools and their preferences. We also asked about the advantages and disadvantages of each tool.

No Code Required. All eight participants mentioned that one of the advantages of FlowMatic is that users can do programming tasks without any experience in coding. Similar feedback also included that FlowMatic would be suitable for teaching people to program.

> P8: *"[FlowMatic] would be a lot easier for artists, beginners to coding, and kids who don't know actual coding because you have to learn the basics of Javascript to use [AFRAME]."*

Correspondence between Programs and Objects. Participants also described FlowMatic as more "direct" and "intuitive". More specifically, participants thought that it is "always easier to know which object and which event you are operating on" with FlowMatic (P2), while it is "hard to keep track of what you are doing and which variable you are working on" using AFRAME (P3). This is because FlowMatic allows users to operate directly on the objects in the scene so that they can establish correspondences between the program and the objects more easily. With 2D authoring tools, on the other hand, users have to establish the correspondences between the scripts and the objects, which requires more mental effort.

Liveness. Participants pointed out that the immediate feedback of FlowMatic, its *liveness* [254], helped them to be "more efficient" (P8). While using AFRAME, they had to "spend time on compiling and running" (P8). Another participant also mentioned that the *liveness* gave her "more sense of accomplishment" (P6).

Context Switching. Six out of eight participants mentioned that with AFRAME they had to do "context switch" or "switching back and forth" between the HMD and the IDE. They also preferred FlowMatic for being easier and more convenient, since "everything is in VR" (P6).

> P4: *"To test what I was doing [in AFRAME], I had to compile, reload the webpage, wear the headset, and head back to the IDE to do bug fixing, which is very time-consuming."*

Syntax. Participants also said that FlowMatic is easier because the syntax for specifying behaviors is more concise and intuitive than in AFRAME.

> P7: *"It seems like there is a lot of syntax involved [in AFRAME], such as having to get the position of an object, and having to instantiate things, which I think would be easier to get the trigger [button] for those [in FlowMatic]."*

Another example of this is that participants made a lot of mistakes related to the syntax of AFRAME, such as attaching a component, binding an event listener, etc.

Debugging. Debugging was an interesting topic throughout the interviews. It turned out that each system has its own benefits and drawbacks in terms of debugging. Participants who thought debugging was easier in FlowMatic claimed that "it is easier to see how I did wrong since everything is visual (in FlowMatic) and one node is connected to another"(P8), and that bug fixing is harder in AFRAME due to frequently switching the context, according to P4. Participants who thought debugging was easier in AFRAME commented that it was easier to utilize simple functions like `console.log()` to see whether an event had been triggered or not, according to P2. It was also easier because users can go through the code and see where is wrong, while using FlowMatic it is hard for the user to check if she connected a wrong arrow and to know how to change it, according to P5.

4.5.2 Questionnaire

Figure 4.6 shows the aggregated results from the usability questionnaire. Six out of eight participants agreed that FlowMatic is easy to learn (5 Strongly Agree, 1 Agree), while only one participant strongly agreed that AFRAME is easy to learn. Six participants agreed that FlowMatic is easy to use (3 Strongly Agree, 3 Agree), while five participants agreed that AFRAME is easy to use (1 Strongly Agree, 4 Agree). All participants agreed that the design of FlowMatic is good (5 Strongly Agree, 3 Agree). Six participants agreed that FlowMatic is helpful in programming VR applications and that it would be easy to become skillful at FlowMatic. Seven out of eight participants thought FlowMatic was fun to play with (6 Strongly Agree, 1 Agree), which corresponds to the interview feedback that it is interesting to use FlowMatic.

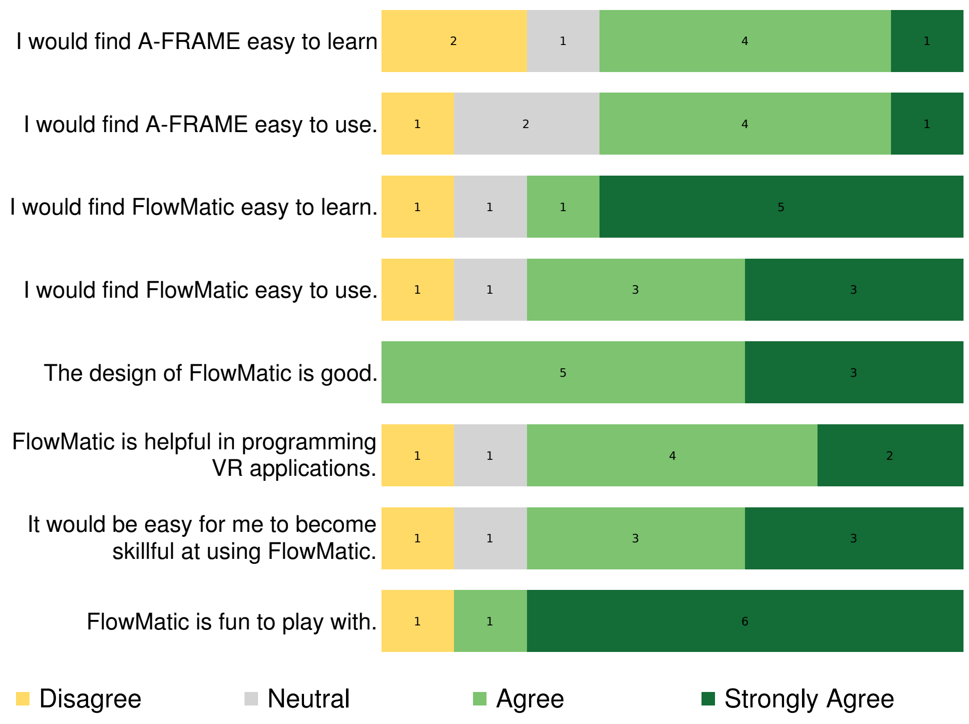


Figure 4.6: Results of the usability questionnaire.

4.6 Discussion

Our results indicate that our immersive authoring system, FlowMatic, is generally easy to learn and use. We observed that participants tended to make more errors when using desktop authoring tools, though we did not quantify them. Even though all participants had programming experience, the errors they made were heavily linked to the programming paradigm of each authoring system, which was outside the scope of their previous experience. More specifically, in the AFRAME condition, the errors were mostly linked with the entity-component system and the event-handler mechanism, which are the core of current VR application development tools.

Participants were also able to build target reactive behaviors using FlowMatic and all of them believed that FlowMatic was more beneficial for non-programmers and novice programmers. Most of them agreed that FlowMatic was more intuitive and direct and they generally preferred the liveness of FlowMatic. Participants also thought FlowMatic was interesting and fun to play with. Using desktop authoring methods, users often need to deal with a lot of “chores” that are less related to their authoring intentions, such as dealing with frame-rate-driven architecture and repeatedly switching between developing and testing. Those chores are also less interesting to users. By integrating FRP and fusing developing and test-

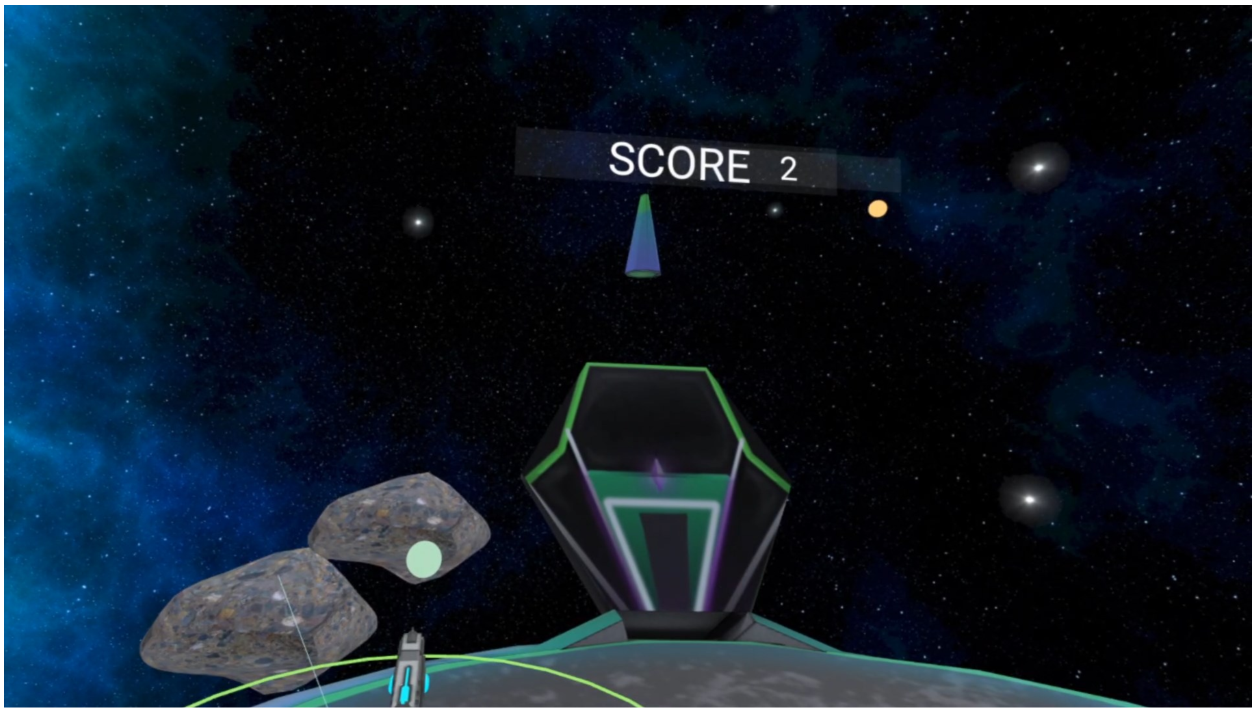


Figure 4.7: A Blaster Game created using FlowMatic, where the player presses the *trigger* button on the controller to shoot spheres out and earns scores by hitting the target asteroids.

ing in the immersive environment, FlowMatic allows users to directly map their intentions into operations and receive immediate feedback in the 3D world.

4.7 Example Applications

To demonstrate the expressiveness of FlowMatic, we use *replicated examples* [150] and create three example applications. The first application, a blaster game, is one of the most common gaming mechanisms in VR where the user shoots bullets using the controller to try to hit the targets. The VR Whac-A-Mole game asks users to use a hammer to hit the moles that are generated in random positions on the ground. In the last application, we replicated Beat Saber—one of the most popular VR games, where users swing their controllers to slash the blocks moving towards them in a certain rhythm.

4.7.1 Blaster Game

The tricky parts of creating a blaster game include dynamically creating bullets at the gun tip position at run time, detecting collisions between targets and each bullet, and dynamically destroying objects when they collide. As partly covered in the previous example, we

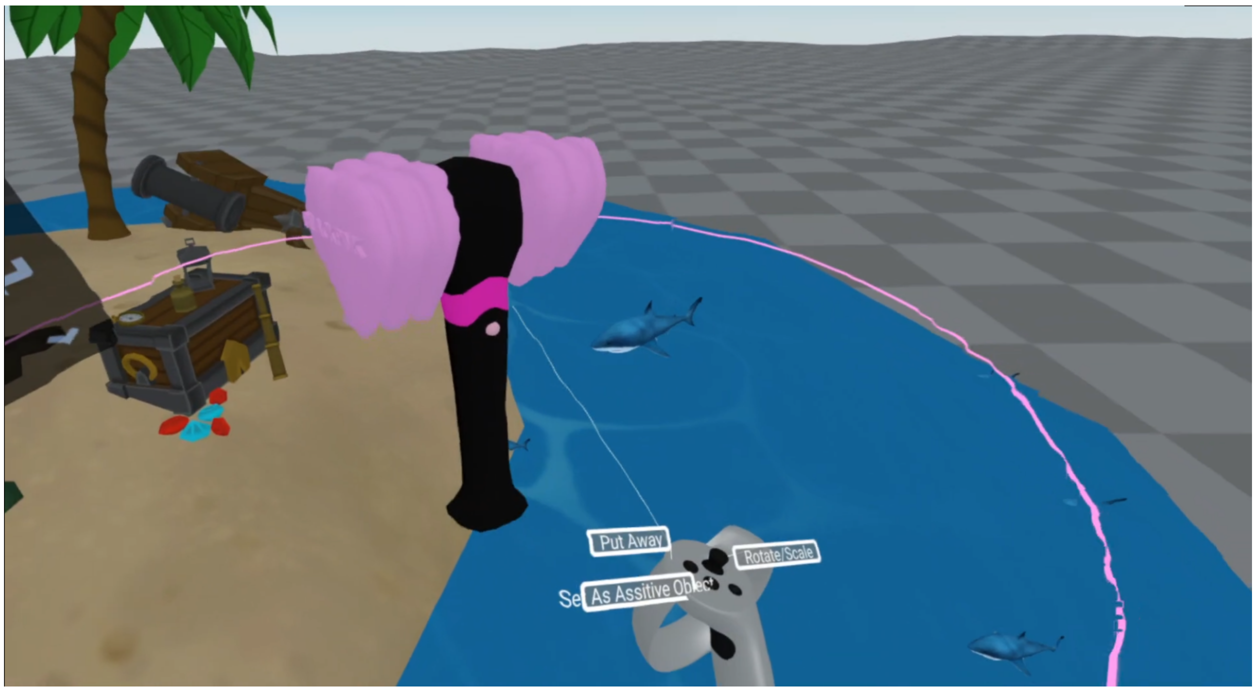


Figure 4.8: A VR Whac-A-Mole game where the user holds a hammer and tries to hit the shark models randomly generated on the water's surface.

accomplish this by using FRP operators that can dynamically create, translate, and destroy a bullet (represented as a sphere but can be any model from an online library) according to different system events.

4.7.2 VR Whac-A-Mole

Whac-A-Mole is an arcade game where the user uses a hammer to hit the randomly generated moles and earns scores. Again, this application requires dynamically creating/destroying objects and reacting to system events such as collisions between the hammer and the moles. In addition, this application was previously hard to replicate because it requires the system to generate objects at random positions in the space. We address this by using an aforementioned *abstract node* to directly specify an area and a random generator as an operator to generate random data of type *vector3* as positions within the area.

4.7.3 Beat Saber

Beat Saber is a popular rhythm game in VR. The difficult aspect of replicating this game is to arrange the timing of each moving block according to the rhythm. We address this by using an interval operator that can fire events based on a specified interval value. The create operator can then subscribe to the internal clock events and generate blocks accordingly.

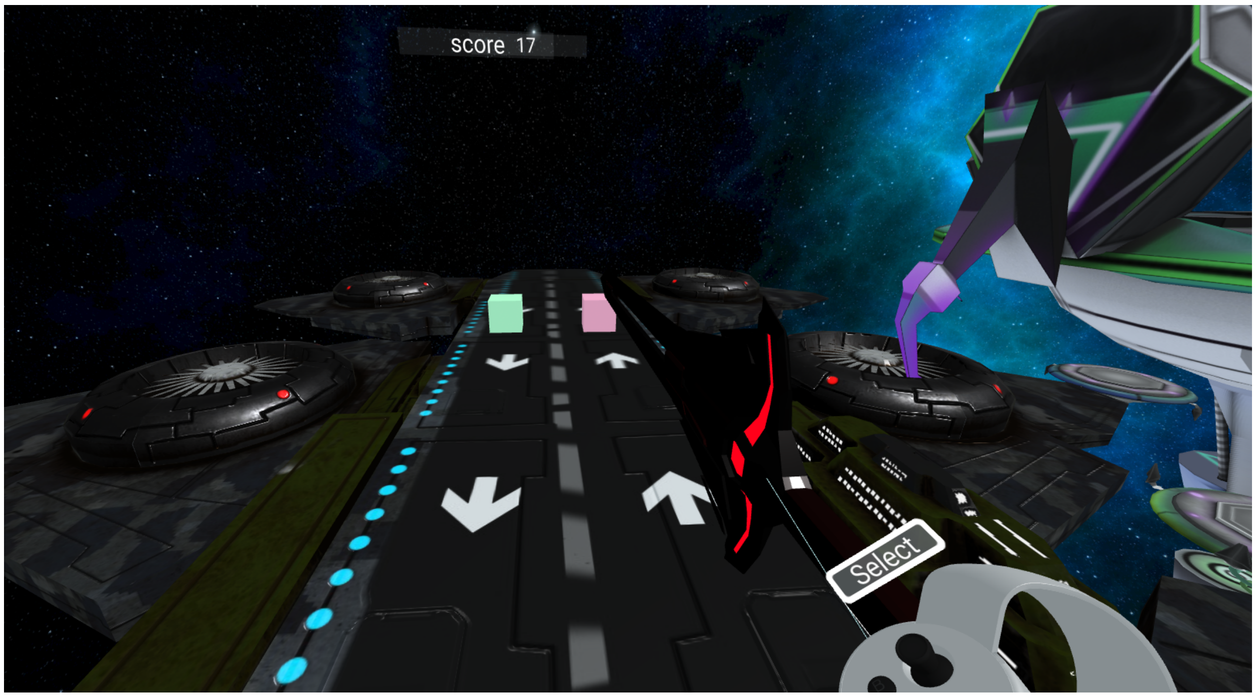


Figure 4.9: A basic implementation of Beat Saber, where the blocks are created at certain intervals and the player swings the sword and hits the cubes to earn scores.

4.8 Conclusion

Our current study has several limitations. First, we have only performed a short-term study comparing FlowMatic and AFRAME for beginners. A long-term study or a study with experts in VR will be needed to see whether the conclusion still holds for people who are familiar with VR programming tools. Second, since all of our participants had an interest in VR and FlowMatic appeared new to them, their feedback may be biased. People who are experts in JavaScript may feel more comfortable using imperative programming languages rather than FRP. Finally, while FlowMatic is expressive for creating different reactive behaviors, there are several components of VR applications that are not supported in FlowMatic yet, such as authoring particle systems and complex algorithms. However, recreating a complex authoring tool such as game engines is beyond the scope of this chapter. Instead, we explored the possibility of making programming VR applications easier and proved that FlowMatic is capable of allowing novices to build relatively complex VR scenes.

In this chapter, we presented a novel immersive authoring system named FlowMatic that raises the ceiling of the expressiveness of immersive authoring tools. To enable that, we integrated concepts of FRP and modeled reactive behaviors of objects as time-varying signals or event streams. FlowMatic introduces a set of dynamic operations, intuitive interactions, and visual representations for defining reactive behaviors, reducing complexity, and programmatically creating/destroying objects in a scene. We conducted a comparative study with AFRAME, a desktop authoring method, to evaluate the usability, advantages and disadvantages of FlowMatic. Our study results show that participants were able to build the target reactive behaviors using FlowMatic, and that it is intuitive, fun to play with, and helpful for programming VR applications. We also demonstrate the expressiveness of FlowMatic by replicating three relatively complex applications that were impossible to build using prior immersive authoring systems.

CHAPTER 5

VRGit: Supporting Design Exploration and Collaboration in Immersive Content Creation[1]

5.1 Introduction

By leveraging people's spatial capabilities, VR provides an effective approach for users to formulate and evaluate ideas of 3D content. Research has also shown that VR provides an effective tool for users to evaluate ideas for 3D spatial content in multiple creative domains such as game design [261], architecture [84, 199], urban planning [245], and interior design [119]. Furthermore, as modern workforces in these domains are becoming diverse in terms of their skill sets and backgrounds, better collaborative content creation support is needed for coordination among various roles including designers, developers, and customers/end-users [14, 143].

Collaborative content creation is an iterative process in which users may perform numerous editing operations, explore various design alternatives, communicate with collaborators, and shift between individual and shared activities frequently [109, 257, 104]. Keeping track of version history in this process is important for providing the ability to revert to previous states if necessary. In addition, providing rich history-keeping can help users explore different design alternatives in the task of creative content production [226]. In collaborative settings, keeping track of version history is even more challenging since users may also need to maintain awareness of collaborators' activities. For example, imagine you are collaborating on designing a VR scene, and you would like to explore a design variant of the current scene without interfering with your collaborators' design. Moreover, when you and your collaborators are working on different design variants, you would like to know which versions your collaborators are working on and communicate ideas with them. If all the

[1]Portions of this chapter were adapted from [282]

versions of the scene, including different branches that collaborators are working on, are preserved and visualized in VR, you could easily "travel" between versions and communicate with your collaborators across versions or branches. While existing systems enable compelling experiences for creating and manipulating 3D content in VR, most only enable basic history-keeping (e.g., a linear timeline) for single users in VR.

VCSs have been used widely for keeping track of version history of digital content among collaborators. Most current VCSs, however, are designed for text rather than spatial data such as 3D scenes. Research in VCS for 3D modeling has explored some effective mechanisms for one or two features of a VCS such as comparing and merging scenes or models on 2D displays [66, 63, 38, 219, 34], but still lacks knowledge regarding how to enable multiple users to track version history without breaking the fluidity of immersive authoring. On the other hand, although collaboration systems in VR have long been an exploration in the area of HCI and CSCW [205, 204, 278, 111], most of them lack version control capabilities and keep only one version of 3D scenes at a time for all users. In this work, we aim to explore providing visualization and interactions of version history that are appropriate for collaborative immersive environments. We provide a complete, standalone design and implementation for version control in VR in order to avoid breaking the immersion and the workflow of collaborative content creation, to leverage intuitive interactions that VR affords, and to harness people's spatial skills for understanding and navigating 3D environments. We also take a different approach by enabling collaborators to stay in different versions of 3D scenes and explore supporting awareness and communication among collaborators across different versions.

We introduce *VRGit*, a new VCS for collaborative content creation in VR. VRGit enables novel visualization and interactions for version control commands such as history navigation, commits, branching, previewing, and re-using. VRGit is also designed to facilitate real-time collaboration by providing workspace awareness, whether users are working on the same version or different versions. More specifically, when users are in different versions, our system enables shared views for understanding where collaborators are and what they are doing. VRGit also introduces a shared visualization to reduce friction during group discussions when users are in the same version, by providing awareness related to version control operations such as navigating version history and re-using 3D content. Finally, we describe an exploratory lab study with 14 participants in which we evaluate the usability and utility of VRGit. Results show that it enables users to easily keep track of non-linear version histories and improves the collaborative workflow of content creation in VR.

The contributions of this chapter include: *(i)* the design and implementation of a new VCS for collaborative content creation in VR, and *(ii)* results and design insights gained from an exploratory lab study that evaluated the usability and utility of the VCS for content creation in VR.

5.2 Related Work

Our work draws inspiration from prior literature in VCSs for 3D scenes, graphical history visualization, collaborative virtual environments, and visualization and interaction techniques in VR.

5.2.1 Version Control Systems for 3D Scenes

A Version Control System, also known as a Revision Control System, enables users to keep historical versions of digital content. VCSs such as Git [89] and Subversion [12] have been popular in the domain of software engineering to help developers keep track of the history of source code by committing changes along with text messages that describe the changes. Most VCSs also support asynchronous collaboration among multiple developers at remote locations by allowing them to manually create and merge branches. However, most existing VCSs are unsuitable for tracking and understanding changes of 3D scenes because the underlying line differencing mechanism is designed for tracking changes of text files and thus lacks high-level semantic information of spatial data such as 2D images and 3D scenes. Recent work, primarily from the Computer Graphics community, has thus explored techniques for tracking changes in media files such as 2D images [41] and 3D scenes [66, 67, 63, 38, 219, 34], which can be categorized into two approaches: state-based and operation-based.

State-based approaches aim to build effective mechanisms that can automatically derive changes by comparing two states, e.g. a version and its successor, after the changes occur. Prior work focusing on state-based approaches has strived to derive changes at different levels of granularity [66, 67, 63, 38]. For example, Doboš and Steed version 3D assets at a coarse granularity of individual nodes of a scene graph such as individual meshes [66, 67]. SceneGit can derive changes at a finer granularity of vertices and faces [38]. In the domain of drawing, techniques such as object-oriented drawing can also preserve states of individual attributes and allow users to revert to previous states of an attribute without interfering with other attributes [277]. The other approach is operation-based, which records changes while they occur. This approach typically records editing operations that users make, and then applies the operations to a state to transform it to the successor state [219, 34, 41]. For example, MeshHisto stores and transmits mesh difference by encoding them as sequences of primitive editing operations [219]. In our work, VRGit uses operation-based change tracking since it

Table 5.1: Summary of prior VCSs and VRGit. VRGit contributes to a full VCS in collaborative immersive environments.

System	**Content Type**	**Version Control Features**					**State/Operation**	**Real-time Collaboration**	**Immersive**
		Navigation$^+$	Commit	Branch	Diff	Merge			
Git [89]	Text	✓	✓	✓	✓	✓	State-based		
Chen et al. [41]	Image	✓	✓	✓	✓	✓	Operation-based		
Dobõs et al. [67]	3D Scene	✓	✓	✓	✓	✓	State-based		
MeshGit [63]	3D Scene				✓	✓	State-based		
SceneGit [38]	3D Scene				✓	✓	State-based		
MeshHisto [219]	3D Scene				✓	✓	Operation-based	✓	
CSculpt [34]	3D Scene				✓	✓	Operation-based	✓	
Spacetime [278]	3D Scene				✓*		State-based	✓	✓
Lilija et al. [159]	Spatial Recordings	✓					State-based		✓
VRGit (this work)	3D Scene	✓	✓	✓	✓*	✓**	Operation-based	✓	✓

Navigation$^+$: History visualization and navigation.

✓*: The system allows users to compare different versions instead of calculating the differences.

✓**: The system allows users to reuse content from different versions instead of merging all changes.

is more precise and efficient in determining the difference between two states and provides functionality such as replay and undo. It has also been applied to prior content creation tools such as 3D modelling [219] and sculpting [34].

Our work contributes to existing literature in VCSs for 3D scenes by introducing a full VCS in *collaborative*, and *immersive* environments (Table 5.1). While prior work has been focused on one or two features of a VCS such as diffing and merging and has targeted VCSs on 2D screens, VRGit aims to explore novel visualization and interactions of a VCS and provide real-time workspace awareness in collaborative content creation in VR.

5.2.2 Graphical History Visualization

Enabling intuitive graphical visualization and interactions for version or operation history has long been an area of exploration in HCI. Prior work has explored visualization of history using representations such as layers of operations [178], snapshots of before-and-after states [147], and timeline views of history [210]. Later work has then built on these representations and explored techniques that enable users to better understand and interact with operation history. For example, Klemmer et al. built upon snapshots of states of collaborative web editing sessions and embedded non-linear branches in the timeline view [139]. Nakamura and Igarashi captured the Graphical User Interface (GUI) input and output history of graphical documents and visualized the snapshots with annotations of detailed operations [181]. Chronicle instead captures the video history of graphical documents and provides users with a set of probes to filter the revision history [101]. More recently, Chen et al. explored using a DAG for versioning image editing operations [41]. However, all the above systems for visualizing and interacting with version history are designed for text or 2D content such as paintings and images.

Most graphical history representations for 3D scenes today have primarily focused on viewing and interacting through 2D displays. For example, commercial Computer-Aided Design tools such as Autodesk Maya or Vistrails are able to record modelling history and provide a list of operation history in the editor. Another line of research in this space is focused on interactive summary of long sequences of editing operations. For example, MeshFlow [62] and 3DFlow [64] are proposed to summarize the history of mesh editing by clustering editing operations. Closer to our work that visualizes history in VR, Lilija et al. introduced techniques of visualizing objects' trajectory in 3D scenes and allowing users to view the history of spatial recordings in VR [159], though not in the context of a VCS. Our work expands on prior work to explore graphical representation and interactions of non-linear history (i.e. branching) in a VCS for immersive VR authoring.

5.2.3 Collaborative Virtual Environments

Researchers have acknowledged the importance of designing real-time collaborative systems that support workspace awareness, i.e. understanding of other collaborators' interaction with the shared workspace [104]. Prior work has explored various techniques for supporting awareness of other users in collaborative virtual environments such as the use of gaze [205], gestures [276, 189, 204], and pointers [71]. Recent advances of the underlying sensing technologies have also allowed for capturing and rendering full bodies of users via 2D projection (e.g. Room2Room [201]) or 3D hologram (e.g. Holoportation [194]). Beyond the awareness of other users, workspace awareness also involves understanding of collaborators' workspace context. To achieve that, sharing views of the workspace has shown to be an effective technique [88, 81, 186, 258]. For instance, Fraser et al. proposed using peripheral views to support peripheral awareness of other users in collaborative virtual environments [81].

A common limitation of the above techniques is that they are designed for providing awareness when there is only one version of 3D scenes and all users are virtually co-located and therefore visible to each other in that version. Highly relevant to our work, Spacetime proposes the concept of parallel objects that allow users to create parallel versions of the same object similar to branching [278]. However, their technique still requires users to land on one final version and renders all parallel designs using different levels of transparency in the same version. In this work, we take a different approach that allows users to manually branch into different variants of the 3D scenes and make changes in those branches while still maintaining workspace awareness when they are located in different versions or design variants (i.e., branches) of 3D scenes.

5.3 VRGit

VRGit is a VCS for VR that enables users to keep track of multiple versions of 3D scenes, to create and navigate different branches, and to preview and reuse content from different versions. Beyond supporting individual uses, VRGit also supports real-time workspace awareness of users' activities and version history by integrating synchronous communication and enabling a shared history visualization. To instantiate the visualization, interaction, and collaboration design of VRGit, we first build an immersive authoring system that enables users to create and manipulate 3D scenes directly in VR. We choose an example of designing the floor plan of an apartment for evaluation with end-users, because it has been a key VR application since it requires users' spatial capabilities and has been used to evaluate prior collaborative VR systems [119, 195]. We also believe the design concepts behind VRGit are

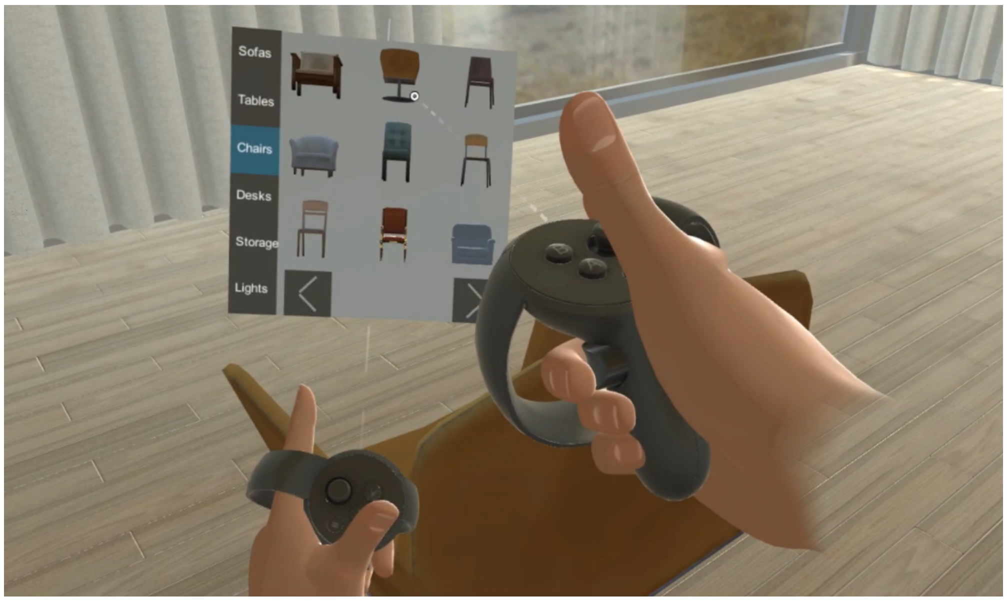


Figure 5.1: The immersive authoring environment. A menu is always attached to the left controller. Users can select pre-made furniture models of different categories and place them in the VR scene.

generalizable to other application domains (e.g., game scenes design) as well, because the immersive authoring operations (e.g., manipulation of 3D objects), version control mechanism (e.g., commit and branching), and collaborative support (e.g., portals and shared visualization) are independent of the task context.

5.3.1 Immersive Authoring Environment

We build an immersive authoring environment that allows users to design the spatial layout of an apartment, as a foundation for instantiating the concepts of VRGit. Users can place and manipulate pre-made furniture models in an empty apartment, as seen in Fig. 5.1. Our system is scoped to authoring VR scenes at a fixed scale (e.g., city scales in urban planning or room scales in interior design) to better focus on the challenges of visualizing and interacting with version histories in VR, and keeping workspace awareness among collaborators across different versions.

5.3.2 Version Control System

VRGit supports full version control of the 3D scenes during the editing process. We use an operation-based history model that records users' edit operations and then applies the operations to a state to transform it to the successor state. We use this model because *(i)*

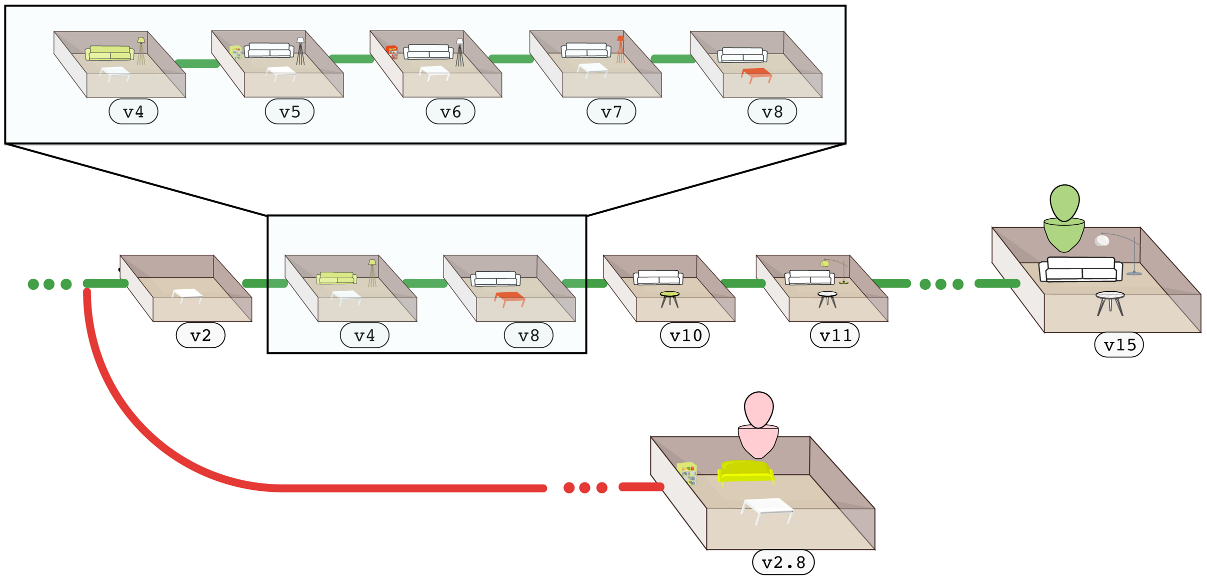


Figure 5.2: The summarized version history. The upper area refers to part of the DAG that generate a node whenever there is a new operation. The middle area shows part of the summarized HG where *V4-8* are clustered based on operation dependencies. The bottom area shows the constraint of workspace awareness, where a version that another user is working in will always be shown.

it has been utilized in prior editing tools such as desktop-based 3D modeling [219] and 2D images [41], and *(ii)* it can be more precise and efficient in determining the difference between two states [219] and provides functionalities such as replay and undo [41]. Our current system supports common immersive authoring operations including creation, transformation, and deletion. We use a Directed Acyclic Graph (DAG) as the underlying data structure where each node represents an editing operation with its relevant parameters and each edge along with its direction represents the temporal order between two operations.

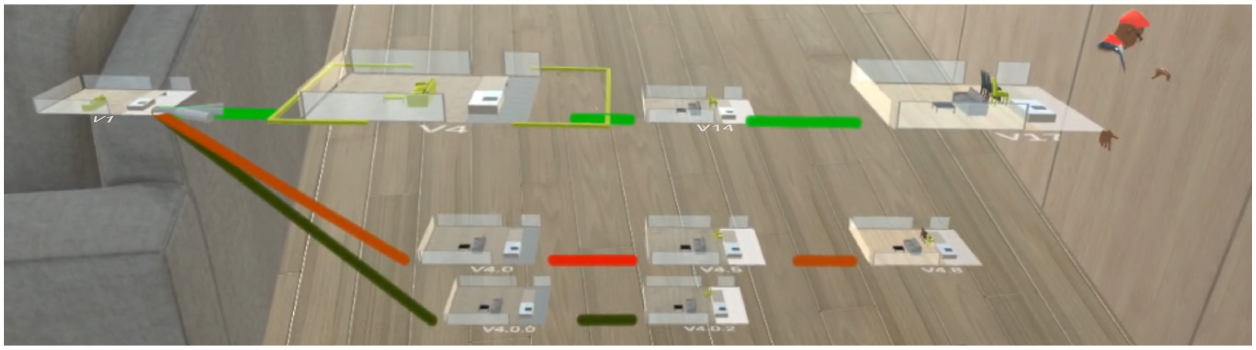
5.3.2.1 History Visualization and Navigation.

We visualize a directed graph on the users' left arm to represent version history as a History Graph (HG). Each node is visualized using a miniature of the version state and each edge represents the temporal order between two versions. We also show the version number in text in each node. Inside each miniature, we highlight the difference with its prior version by changing the material of the furniture with color coding: we use green to represent an added object, yellow to represent transformation of an existing object, and red to represent a deleted object. Because of the potential complexity and large size of the underlying DAG, the HG only shows users the summarized version history, as shown in Fig. 5.2. Our summarization techniques take into account three parameters: *(i)* operation dependencies, *(ii)* spatial constraints, and *(iii)* workspace awareness. We use a simple timeout mechanism to determine operation dependencies, which generates a new node in the history graph after the user has been idle for a specified amount of time (10 seconds when working alone, 15 seconds when working collaboratively—defaults that we found to work anecdotally). Spatial constraints allow us to determine the number of miniatures and branches to display given the amount of available space in a visualization anchor (e.g. users' arms or a tabletop). VRGit also always shows which versions users' collaborators are working in, as a way to improve workspace awareness. Users can select the previous or next version in the history graph by pushing the thumbstick on the left controller horizontally. A cursor of a yellow square is then shown around the miniature when a version is selected. To enter a version, users can select the version and press a button on the controller. The layout of the environment will then change to the state of that version. When there are multiple branches at a node (shown as parallel siblings in the HG), users can also switch branches by pushing the thumbstick vertically.

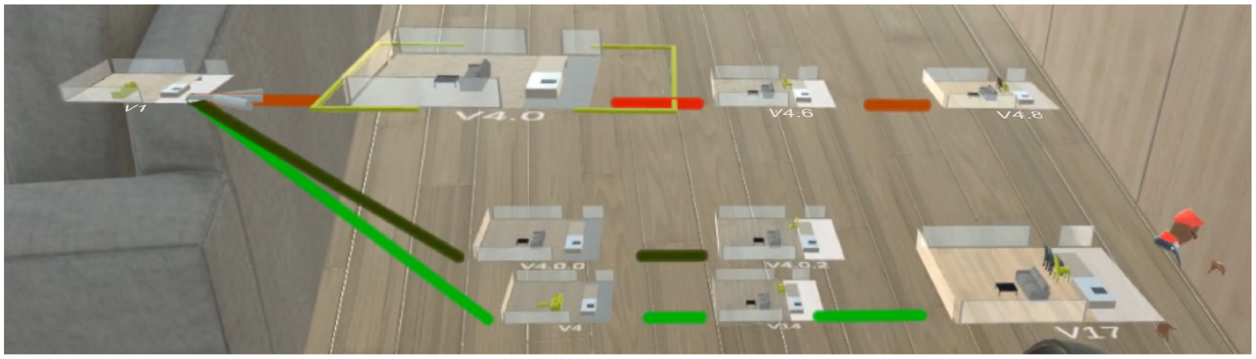
5.3.2.2 Commits and Branching

In VRGit, commits (checkpoints in the repository) are made automatically by the system whenever the user performs a new operation. VRGit does not require that users explicitly perform commits, which could interrupt their workflow, while allowing users to revert back to previous versions if any misoperations occur. When a new commit is made, the system will append a new node that represents the editing operation to the underlying DAG. The summarized HG will in turn be updated based on the new DAG.

VRGit enables visualization of multiple branches and intuitive interactions for creating, updating, and navigating different branches in the visualization, as shown in Fig. 5.3. Creative tasks such as content creation usually involve numerous trial-and-error experiments and design variants (branches) [109, 257] and the ability to keep multiple branches has shown to be an important building block of creativity support tools [226]. In our system, users can easily create a branch based on a historical version by first entering that version and then pressing a button on the controller. The system will then create a copy of the node that represents the historical version, and append the copy to its parent node in the underlying DAG. The HG will then be updated by laying out the branches in a circular path that takes advantage of depth afforded by VR displays. The user will be automatically switched to the new branch once it is created and can update the new branch modifying the 3D scene. Users can use the thumbstick to navigate different existing branches as mentioned above.



(a) Visualization of three branches.



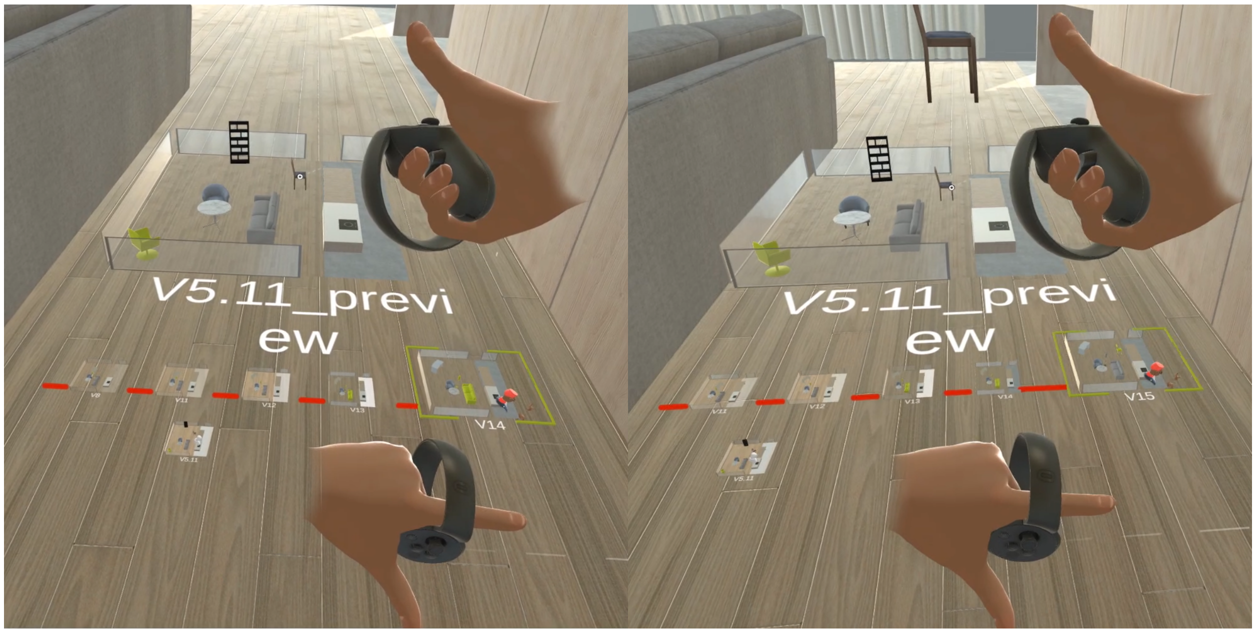
(b) Visualization after switching to the next branch.

Figure 5.3: Screenshots of the History Graph (HG) that visualizes multiple branches. The user stays in the version of *V17*. In 5.3a, the user is selecting *V4* which belongs to the branch highlighted in light green. Users can switch to other branches. After switching branches (5.3b), the user is selecting *V4.0* which belongs to the branch highlighted in dark red.

5.3.2.3 Previewing and Reusing

A preview allows users to easily examine the state of a historical version without actively entering that version, typically known as "snapshots" or "thumbnails" in 2D graphical editing tools [147]. In our system, as the miniatures in the HG can be small for inspection depending on the anchor of the HG, we enable users to preview of a version in the HG by showing an expanded miniature of the version. Users can open a preview of a version by using the raycast to aim at the version and pressing a button on the controller, as shown in Fig. 5.4a. They can also resize the preview to better inspect changes in the version. Multiple previews can exist at the same time and users can directly manipulate the preview for the convenience of inspection.

Another novel functionality that previews afford is reusing spatial design from one or more previews. In conventional VCSs, reusing or combining data between versions is done via *merging*, which applies all changes of one branch to another. Along this line, prior



(a) Preview. (b) Reuse.

Figure 5.4: Screenshots of previewing and reusing in VRGit. In 5.4a, the user is in *V14* and opening a preview of *V5.11*. In 5.4b, the user reuses the chair in the preview by aiming at it using the raycast shooting from the controller and pressing a button on the controller. The chair then appears in the environment and the system automatically creates a commit that brings the user to *V15*.

work has also proposed mechanisms for merging changes and resolving conflicts of two 3D scenes [66, 63, 38]. However, merging is typically limited to fusing all changes between only two versions at a time. Content creation, on the other hand, is often an open-ended design process that is subject to unexpected changes of directions or goals [51]. It is therefore common for users conducting creative tasks to explore different design variants by selectively mixing-and-matching changes from multiple sources [156, 39, 40, 246]. For instance, imagine an architect who wants to mix-and-match designs of the balcony from one version, the furniture from a second version, and decoration from a third version. It would be useful if they can open previews of the three versions and directly reuse these specific parts of the 3D scene. Therefore, instead of fusing all changes between two versions at a time, our system allows users to reuse objects from multiple versions. In VRGit, users can reuse spatial design of different versions and selecting the target item in the previews, as shown in Fig. 5.4b. The selected furniture will then appear with the same transformations (i.e., position, rotation, and scale) in the user's current version. To incorporate designs from multiple versions, users can create multiple previews and reuse objects in those previews accordingly.

5.3.3 Collaboration in VRGit

Collaboration is an important component in most VCSs. For instance, VCSs such as Git and Subversion are designed for multiple developers at remote locations to collaborate with each other, by allowing them to sync changes (e.g. commits or branches) to the repositories through a central server. The collaboration in such VCSs is asynchronous and users typically work in separate editing environments. Numerous synchronous 3D scene editing tools exist (e.g. [219, 34]), but they do not support active branching or navigating history in their VCSs. In VRGit, users can collaboratively author the 3D scene when they are in the same version, analogous to a synchronous editing tool. They are able to see each other's edits and avatars, point to objects with raycasts, and talk to each other via audio communication. VRGit also allows users to create branches and navigate to different versions, in order to explore different design variants without interfering with each other. In this scenario, they can work on their own branches but are not able to see each other's avatars in the environment, analogous to an asynchronous editing tool. In all, collaborators can easily separate or reconcile by navigating and branching into the same or different versions.

In this work, we address the unique challenges of supporting communication and workspace awareness in immersive authoring through a VCS. We consider two primary scenarios: *(i)* when users are working in different versions and *(ii)* when users are working in the same version. In most existing VCSs, being in different versions means users are working in separate workspaces and there is little support or need for real-time workspace awareness since there is no presence of users or activities in the workspace [252]. However, as in VRGit, users can easily switch and work in different versions or branches, so there should be a consistent presence of collaborators and their activities as well as a convenient way to communicate in order to maintain workspace awareness. We incorporate mini-avatars in the HG that indicate in which version (*where*) collaborators are located. We also integrate the concept of portals and shared history visualizations in our VCS that help users understand *what* collaborators are doing and communicate with each other when they are working in different and the same version respectively. We detail the design of these two features below.

5.3.3.1 Portals

In collaborative authoring, it is common for collaborators to adopt a 'multi-synchronous' collaboration styles in which they work simultaneously in isolation and subsequently integrate their contribution [69]. VRGit enables this by allowing users to create and work in different branches, and combine their work together by reusing designs from multiple branches. An important aspect in this process is how to enable communication between collaborators and

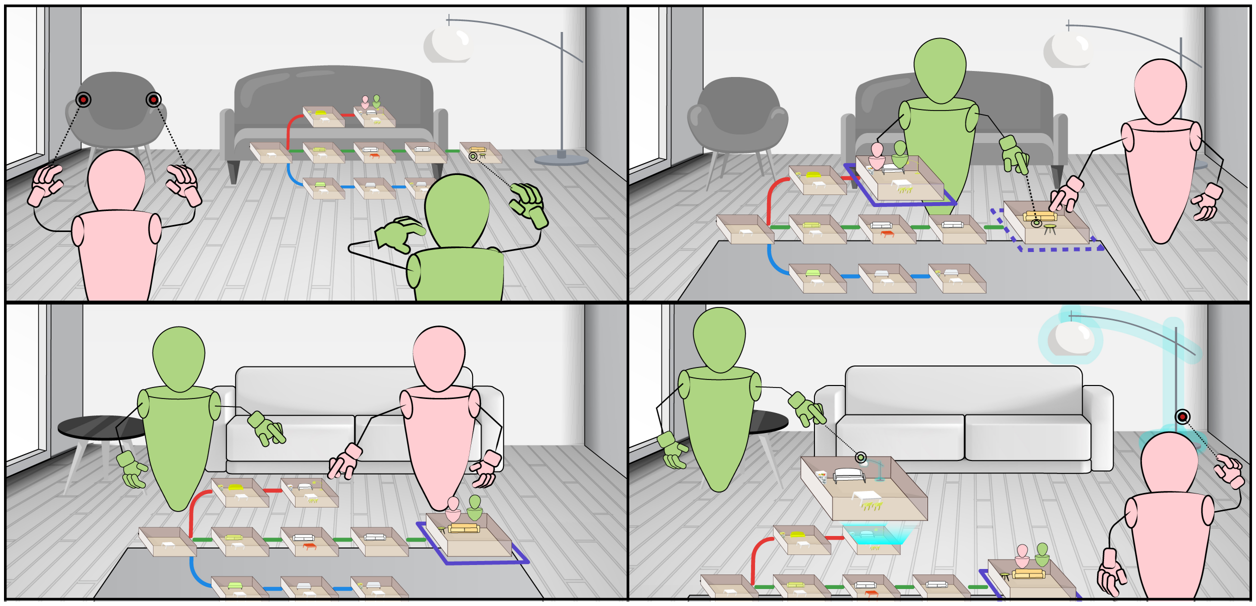


Figure 5.5: The workflow of shared history visualizations. When two users are working in the same version, one user can start a shared history by pointing at the HG on the left arm and pressing a button (top left). The shared history visualization will then be anchored on the table and the sharer can select a version (highlighted in dashed line) for all users (top right). They can then enter that version collaboratively (bottom left). Finally, all users can collaboratively preview and reuse the lamp from another version (bottom right).

awareness of their activities in order to ensure common ground [49] and to avoid redundant work [120]. VRGit addresses this by integrating portals into the HG that allow users to easily monitor and communicate with each other when they work in different branches. Portals are 2D video streams from collaborators' first-person view, which have been shown to be useful in understanding collaborator's activities [141, 11]. Users can create a portal of another user by using the raycast from the controller to aim at the mini-avatar appearing in the HG and pressing a button on the controller. A 2D plane showing the target user's first-person view will be created next to the mini-avatar, as shown in Fig. **??** (left). Users can directly manipulate and place the portal.

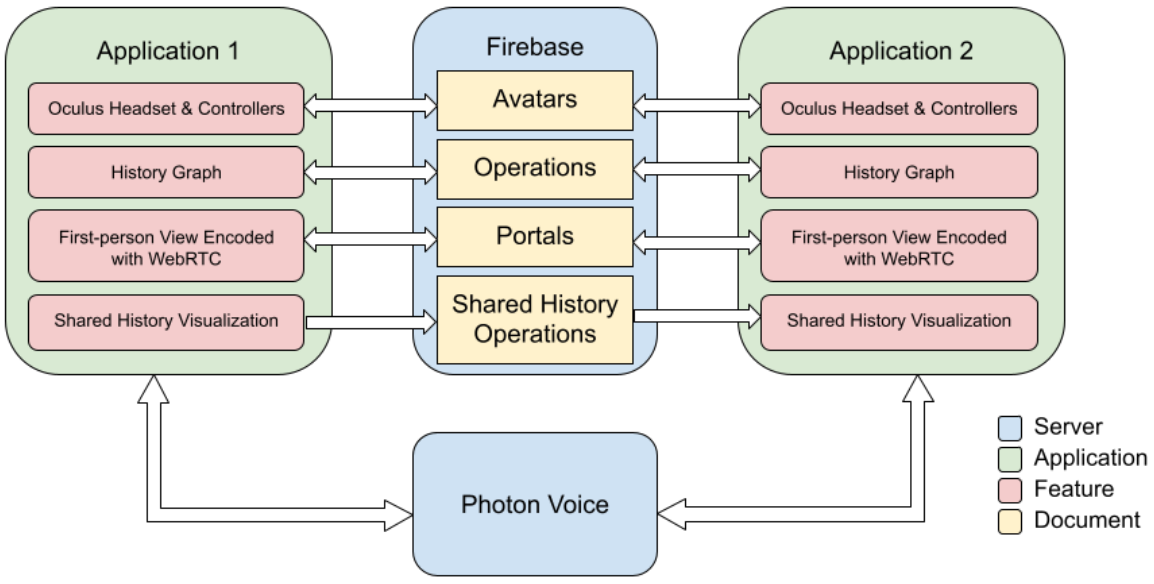
5.3.3.2 Shared History Visualization.

In VRGit, shared history visualization is designed to facilitate discussions about multiple versions (e.g., to compare features) or about the edit history itself, as shown in Fig. 5.5. A unique challenge for the VCS under this setting is how to maintain awareness of version history and version control operations from collaborators. As each user keeps a local HG on their left arm, similar to a local copy of the entire repository in VCSs such as Git, their views of the HG may be different due to navigating versions and switching branches. Therefore, there is little awareness of collaborators' version control operations such as navigating and entering different versions, and it is difficult to refer to specific versions during discussion. VRGit addresses this by introducing shared history visualization where the local HG originally anchored on each user's arm moves to a shared location in the virtual scene. One sharer is required to create the shared history visualization by using the raycast to aim at the HG anchored on the arm and pressing a button on the controller. Then every user in the same version will be able to see an animation of the HG moving from their arms to a shared location in the virtual scene. Similar to screen sharing, the sharer can interact with the shared visualization to navigate history, switch branches, and create previews. The sharer can also move the shared history visualization through direct manipulation. The operations on the shared history visualization are synced for all sharees who are in the same version to ensure they have the same view and understanding of the shared history visualization. For instance, when the sharer navigates and enter an older version, all sharees are able to see the navigation in the shared history visualization and enter the version with the sharer. When the sharer creates a preview and reuses objects from the preview, all sharees can see the preview being created and the objects being reused in the current version.

5.3.4 System Implementation

VRGit is implemented using Unity 2020.2.7 and runs on Oculus Quest or Rift headsets. The overall architecture of our system is shown in Fig. 5.6. We use the Photon Voice[2] plugin in Unity to enable voice chat among users. The rest of our functionalities are mainly synced through a Firebase server.[3] More specifically, the application encodes the animation of avatars that is tracked in Oculus in binary and updates the document linked to the user ID on Firebase. In VRGit, operations of each user are synced through the operation document on Firebase and used to update the local copies of HGs in each application. We currently support five operation types: creation, transformation, deletion, entering, and branching. The first three are immersive authoring operations and the rest are manual operations in the HG that need to be updated. When users start a portal to share their first-person views and communicate with each other, we encode the video streams in binary using WebRTC[4] and sync the streams through Firebase. Finally, while all of the above functionalities are synced in both directions, the shared history visualization is in one direction. When one user becomes the sharer by starting a shared history visualization, all the operations from the sharer such as navigating history, switching branches, and previewing and reusing are

[2]https://www.photonengine.com/voice
[3]https://firebase.google.com/
[4]https://webrtc.org/



Application 1
Oculus Headset & Controllers
History Graph
First-person View Encoded with WebRTC
Shared History Visualization
Firebase
Avatars
Operations
Portals
Shared History Operations
Application 2
Oculus Headset & Controllers
History Graph
First-person View Encoded with WebRTC
Shared History Visualization
Photon Voice
Server
Application
Feature
Document

Figure 5.6: The system architecture of two applications running VRGit. VRGit uses two servers: Photon Voice and Google Firebase. All applications write and read audio data from the Photon Voice server to sync voice communication. Applications sync the movement of users' avatars by writing and reading data of Oculus headset and controllers from the *Avatars* document on Firebase. Similarly, applications sync their version control operations through the *Operations* document and their video data through the *Portals* document. Finally, for shared history visualization, only the sharer (i.e., App 1) writes data to the *Shared History Operations* document and only the receiver (i.e., App 2) reads data from the document.

sent to the server. All the sharees then pull those shared history operations and update their shared history visualization based on the operations.

5.4 Study

We conducted a qualitative user evaluation of VRGit with two goals: *(i)* to evaluate the usability and utility of the visualization and interaction of the Version Control System, and *(ii)* to assess how well the system support people's communication and awareness in a collaborative task.

5.4.1 Participants

We recruited 14 participants in seven groups (10 female and 4 male, age 20–28) from a university in the United States through public email lists of a department. All participants had prior experience using VR devices. Three groups of participants were friends and four groups were strangers. Each participant was compensated with $30 USD Amazon gift cards for an approximately 120-minute study. Our study was approved by our institution's IRB.

5.4.2 Procedure

Our study was divided into two sessions. The first session was completed by participants individually and the second session was completed collaboratively by three people (two participants and one researcher). To simulate remote collaboration settings, participants were separated in two different rooms where they cannot verbally communicate unless using our system. The first session began with an introduction and a walkthrough of the system that lasted approximately 30 minutes. During the walkthrough, participants were shown individual features and asked to complete some atomic tasks to get familiar with those features. After the walkthrough, participants were asked to complete an individual task to iteratively design the spatial layout of an empty apartment. The individual design iteration task was designed to evaluate how well our system could help users navigate, understand, create branches, and reuse designs across versions. The task took approximately 15 minutes to complete and will be detailed in the next section. After the individual task, participants were asked to take off their headsets and fill out a survey that examines the usability and utility of the VCS, where participants were asked to rate, on a 7-point Likert scale, statements such as "I found it easy to use" and "I think it would be useful." We also examined how well the system helped users make sense of version histories. Participants were asked to rate on 7-point Likert scale statements such as "I found it easy to know what has been changed between two consecutive versions." After the survey, participants were given a 5-minute break.

After the break, participants were given debriefs on the collaboration task (also detailed in the next section), which lasted about 10 minutes. After the instruction, participants were connected with each other online. They were reminded of the main communication features of the system such as the usage of portals and the shared history visualization. In the second session, participants were asked to complete a collaboration task that simulates the communication process between a client (or buyer), acted by the first author, and two interior designers (or sellers), acted by the two participants. The furniture designers needed to collaboratively come up with two variants of the apartment. This collaborative task lasted about 30 minutes.

At the end of the collaboration task, participants filled out another 5-minute survey that focuses on our collaboration features by examining users' workspace awareness and their communication experience. Finally, we conducted a semi-structured interview with the participants individually that asked about the benefits and challenges of using VRGit.

The interview lasted about 15 minutes. We aggregated all the survey data, transcribed and coded the recordings of interview.

5.4.3 Task

In this section, we describe in detail the individual task in the first study session, and the collaboration task in the second study session. In both study sessions, we asked participants to act as interior designers working remotely to design the furniture layout of a new apartment.

For the individual task, we aim to examine our VCS by asking participants to iteratively design the layout of the living room. The design of the task aims to emphasize the usage of the VCS including history navigation, branching, previewing and reusing content. At the beginning of the individual task, participants were asked to act as an interior designer working in a company and to design the room layout by following the instructions given by the experimenters. The first design iteration was shown as a miniature in VR and participants were asked to recreate the layout according to the miniature. Participants were asked to verbally notify the experimenter once they finished the first iteration. After the first iteration, participants were then asked to undo by going back to a historical version. Based on that version, participants were then asked to create two design variants (branches). After the two design variants were created, the experimenter would pick a random object in a random branch in the scene and ask the participants to reuse the object in another branch. After the object was reused, the participants then completed the individual task.

For the collaboration task, participants were placed in two different branches and asked to freely design the room for two minutes. After that, we aim to prompt the usage of portals between users by giving them information access to different aspects of the design principles. More specifically, one participant was given access to principles of furnishing styles (e.g. color matching, shape). The other participant was given access to principles of furniture arrangement (e.g. required furniture, placement). The principles were incrementally shown every two minutes and displayed on the wall in their individual virtual environments. Participants were asked to act as if these principles were their areas of expertise. Their goal was to make sure their designs satisfy both people's expertise. For example, participant A should not only make sure that the layout satisfied the arrangement principles, but also need to communicate with participant B to make sure that the design satisfied the styling principles. In this way we created scenarios for the participants to ask each other for design feedback through the portal. The design principles were also flexible enough that resulted in design alternatives that were sufficiently different from the two participants, which were spread across two separate branches from participants A and B. Finally, the client (i.e. the researcher) joined the session and started a group discussion involving three people. The goal of the client was to ask for a new design option that incorporated the designs from both participants and to make sure that a shared history visualization is created by either participant A or B. In this way, we encouraged group discussions via shared history visualization.

5.5 Results

All participants were able to complete the tasks using VRGit and thought the system was highly useful for managing version history in iterative design tasks and maintaining awareness and communication with collaborators. Participants also complimented that the system was fun and cool to use: *"I thought the system is really cool; It felt like I was in a futuristic movie"* (P2a). In the following sections, we provide more detailed insights gained from the usability and utility survey and the retrospective interview. We center our findings around the two evaluation goals, *(i)* the usability and utility of VRGit, and *(ii)* the communication and awareness support in the collaboration task.

5.5.1 Usability and Utility of VRGit

5.5.1.1 History visualization and interaction required low effort.

Participants felt that the design of VRGit made it easy to understand historical changes and allowed easy access to specific versions. Participants reported in the survey (on a Likert scale of 1-7) that it was easy to track the evolution of design across various versions (avg=6.2, sd=0.6) and see the changes between two consecutive versions (avg=5.9, sd=1.0). Participants compared to existing VCSs and thought that the miniature representation of nodes in the HG costed low effort to understand the history. *"I do think it is really useful. When I use Github or Photoshop, for the previous versions you have to open up the files and it is not very visual so you have to go through each one."* (P1a) *"Compared to something like Git, where I have struggled to make sense of things like which branch I'm on or how it differs from other branches, I really like the visualization of the VR system, and being able to see little preview models of the rooms and the way they branch."* (P7a) The color highlighting mechanism also made it easy to quickly understand *"what has been added and what has been deleted"* (P5a) between two consecutive versions.

Participants also found it useful to navigate and enter different versions (avg=6.4, sd=0.7). Participants commented on the similarity to existing VCSs like Git, in which users can easily revert to previous states if necessary. *"I think the version history is very similar to Git...like* *on GitHub you can return back to previous version. That's really easy in case you make a mistake."* (P6a) However, the interactions of navigating and entering different versions (avg=4.4, sd=1.7) introduced a learning curve that participants sometimes could not recall which buttons to press on the controller for certain actions. *"Some of the controls were hard to remember, you know, there was a lot of learning in that. But the guide with most of them written out helped."* (P5a) Another reason they found it difficult to navigate to different versions was the potential large size of the HG. *"I felt when looking at the version history, especially when it becomes longer, it is harder to see and navigate to previous versions."* (P1a)

5.5.1.2 Branching allowed easy exploration of multiple design variants

Participants generally thought it was easy (avg=5.6, sd=1.2) and useful (avg=6.1, sd=1.1) to create multiple branches. Participants thought that branching was particularly useful when users wanted to experiment and make changes based on earlier versions. *"If you want to make a little bit of change of something, or if I proceed too much but I want to go back and change that one a little bit. This kind of thing would be very hard for me to keep track of the hierarchy...but if I have a branch, I can actually make different version in the same workspace and show it to other people in that workspace."* (P7b) Users thought the visualization of multiple branches next to in a semi-circular shape allowed easy comparison of different design variants. *"I think it [branches] is really helpful because there are projects where I want to compare different versions. I wanna have those on hand to visualize for clients... I think it's really more persuasive to have the comparative visuals right next to each other."* (P5a)

5.5.1.3 Previewing and reusing were useful for comparison and efficiency.

Participants generally found it easy (avg=5.2, sd=1.7) and useful (avg=6.3, sd=0.6) to see the preview of a version. Previews are suitable for closer inspection than the nodes, allowing users to control the viewing angle by direct manipulation and to get a holistic understanding of the floor plan. This has been mentioned by participants in comparison to portals, in which users did not have control over the views and could only see part of the floor plan. *"I feel like for me it was a lot more useful if I pulled up a preview of their version rather than looking through the portal because I have the bird-eye view and I can move it [the preview] around."* (P7a) Opening previews of multiple versions were also useful for comparison. *"I don't know if it is a feature but if we were able to make previews of multiple versions at one time and put them next to each other. I think it would be pretty useful to compare and* *get the bird-eye view of all of them next to each other."* (P2a) Participants also thought it was easy (avg=5.6, sd=1.4) and useful (avg=6.5, sd=0.7) to reuse objects in the preview. Reusing objects can save time because accurate object manipulation is time-consuming in VR [110] and reusing objects can put those objects at the same location and rotation. One participant also reported to prefer reusing to merging in conventional VCSs such as Git because it allowed them to utilize part of the changes. *"I like it better than, say, like Git or Google Doc because you could pull it [the preview] up and then take in just the pieces that you wanted, versus like Git you have to kind of merge all the updates."* (P2a)

5.5.2 Communication and Awareness in Collaboration Tasks

Overall, participants felt that their overall communication with collaborators was natural (avg=6.4, sd=0.6) and enjoyed the collaboration in the task (avg=5.6, sd=1.3). Participants thought that the portal (avg=6.3, sd=0.9) and the shared visualization (avg=6.4, sd=0.8) with other users were very useful.

5.5.2.1 Portals allowed easy communication on ideas

Participants found it easy to understand what their collaborators were doing (avg=6.1, sd=0.8) and to communicate feedback with collaborators (avg=6.4, sd=0.7) through the portal. Although we mentioned above that previews were suitable for understanding holistic layout of the floor plan, participants thought that being able to see the first-person view of their collaborators made it easy to understand their collaborators activities based on their own experience. *"I thought the portal was pretty good to use because I could see exactly what they were doing...and since I had the same experience too I knew what they were doing and why they decided to do those things."* (P1a) They thought it was easy to communicate feedback because they could easily refer to items (avg=5.9, sd=1.4) through the portal. *"It (the portal) was especially good for, like, if I wanted to get a really close view of something or if I wanted to make sure if it was the right item or the right color."* (P2a) In addition, participants thought the design of the portal can save the effort of leaving for other users' branches. *"[With portals] you don't have to step out from your room to actually go to the second room, you can just be in your room, and just from there you can see what is happening in the other room."* (P6b)

5.5.2.2 Shared history visualization provided common ground for discussion

Participants generally applauded the ability to see a larger scale HG laid out in the physical environment. *"It is nice that the structure of versions is larger when it is on the table. I* *can actually see how many furniture inside that version and decide whether or not to go there."* (P4a) Participants found it easy to refer to a specific version (avg=6, sd=0.8) or object (avg=6.4, sd=0.7) in the shared history visualization. This consensus on versions and objects then helped facilitate participants' discussion on their design. *"...if I actually didn't agree with my collaborator's final design, like for example version 5, but I see in version 3 there is something interesting. I can say 'Okay, actually I like what you did in version 3 here, can we branch off of version 3 and explore something else here?' and the fact that you have this shared vision collectively, it gives that opportunity to branch off of the design."* (P9b) However, the shared history visualization could be confusing sometimes when the user was unclear who was the sharer. *"The shared visualization didn't communicate to me if I had control of it or not. If it said that I'm observing [partner's] visualization, I would know that it's her thing and I wouldn't do anything."* (P4b)

5.6 Discussion and Future Work

Our results demonstrate that users were able to use VRGit for version control in both individual and collaborative content creation in VR. We found that VRGit offered intuitive visualization and interaction for understanding non-linear version history, creating branches of design variants, and previewing and reusing design from different versions in VR. In addition, VRGit facilitated communication and awareness in collaboration with portals and a shared history visualization. Although the system was generally considered easy to use and useful for most tasks, VRGit also introduced challenges of navigating longer histories more effectively and workspace awareness understanding sharing and control among collaborators. In this section, we detail the design considerations and future directions of building VCSs for collaborative content creation in VR. We also discuss the current limitations of our work.

5.6.1 Lowering the Effort for Using VCS in VR

As they are originated from managing source code among developers [259], existing VCSs could require advanced knowledge and skills of both using the system (e.g., shell commands) and interpreting the visualization (e.g., differences between two versions and structures of non-linear version history). Content creation in VR, however, is targeted on a diverse population including non-technical users such as designers and customers/end-users [14, 143]. In our research, we aimed to reduce the barrier of entry to VCSs by designing intuitive visualization and interaction of version history in VR, and found that even people without prior programming experience were able to use VRGit to manage the versions history of their work. This demonstrates the benefits of building graphical representations in VR that end-users are familiar with for tracking version history. This also aligns with the spirit of What-You-See-Is-What-You-Get in immersive authoring research, where many tools and techniques were explored to lower the floor for novices to create content directly in VR [172, 286]. Future work may also draw inspiration from past research that has investigated intuitive visualization and interactions to lower the barrier for using VCS, such as by leveraging people's spatial abilities using the virtual body resizing technique [144] to enable intuitive navigation of version histories. In addition, VRGit only uses controllers as input, which has limited capabilities, making it effortful for users to recall which buttons to press. Future work might also draw inspiration from past research that has investigated richer input modalities in VR such as gestures [13] and cross-device interactions [247, 70] for more precise control and lower mental or physical effort.

5.6.2 Providing Efficient Comparison and Fusion of Design Variants

We found that VRGit enabled users to easily compare different versions through a combination of visualizing variants using branches, highlighting changes in colors, and previewing versions in miniatures. This extends prior work on VCS for 3D scenes by seamlessly providing the comparison in the content creation process without explicit commands such as *diff*. Our study also suggests that the design of reusing content in VRGit was helpful in quickly copying objects from different versions and fusing ideas of multiple collaborators. This aligns with prior work on creativity support where enabling easy exploration of possible creative solutions such as mix-and-match is a key component [226]. This is different from prior work on VCS that uses the *merge* command to combine the whole scenes of two branches instead of part of the scene [63]. Both approaches could become inefficient when users have to manually deal with a large number of content that they do or do not want to fuse (e.g., manually resolving conflicts after merging). Therefore, future work could explore more efficient ways of fusing multiple design variants, such as providing suggestions based on other design variants during the creation process, or more efficient group selection techniques to facilitate fusing.

5.6.3 Leveraging System Support for Richer Workspace Awareness

Our results showed that various channels of workspace awareness were helpful for a VCS to support fluid collaboration experiences. While portals allowed users to understand their collaborators' activities and communicate ideas, when they are working in different versions, participants still sometimes preferred using previews or going directly to the collaborator's version for communication. This is partially because portals were only 2D video streams and did not offer the spatial context of the layout that the other collaborators were working on. In addition, users did not have control over the viewing angle through the portal, which introduced friction when users were looking at their collaborator's environment. Future work could investigate ways to fuse different views of the collaborators' environment in order to support better workspace awareness (e.g. [141]).

Another challenge we found in the user study is the awareness of users' control over the shared history visualization. For instance, in VRGit, only the sharer had control over the shared history visualization, but it is unclear to users in the immersive environment who is controlling and who is not, thus resulting in less engagement by the participants. On the other hand, introducing simultaneous control could introduce potential conflicts over the shared history visualization. Future work should thus investigate ways to balance control in a shared history visualization while maintaining consistency and users' awareness of the version history.

5.6.4 Integrating Version Control in the VR Content Creation Process

Shneiderman suggests that rich history-keeping is an essential feature of creativity support tools [226], and more recently, Sterman et al. [241] also suggest that version control for creative domains should be designed based on the needs of creative processes as creative practitioners in these domains could prioritize different values over efficiency and fidelity as those in the software engineering domain. In line with prior research, our study suggests that our design of history visualization and interaction that is appropriate for VR authoring can help users revert to prior versions, compare and explore design variants, and communicate with collaborators. Besides designing VCSs that can suit the needs of creative tasks, our results also suggest that the design of VRGit can help shape collaboration in content creation such as understanding collaborators' activities without leaving their workspace and sparking design suggestions during discussion based on shared visualization. Future researchers and designers should thus consider users' creative and collaborative needs that are specific to the workflow of collaborative VR content creation when designing VCSs. For example, future research could further explore a hybrid VCS that can bridge different devices and platforms as many researchers have explored cross-device and cross-platform development of VR and AR experiences [143, 234]. More specifically, numerous aspects of VR content creation might take place outside of the immersive environments, from 3D modeling to software engineering, and debugging. Future research could thus explore how to integrate the design of VRGit into current version control practices of the above aspects.

5.6.5 Limitations

Though our findings showed that our VCS is useful for collaborative content creation tasks in VR, our work has several limitations. First, we ran an exploratory lab study of the system with teams of three users for no more than two hours. The way that participants performed the tasks in the lab setting could be different from that in the real-world setting. It is also unclear how well it works for larger collaboration teams, which might reveal additional challenges for collaboration awareness and scalability. In addition, although we did not observe any specific novelty effect during the study, it is possible that it played a role, and a longitudinal study would help us understand how people would use VRGit in practice. A participant response bias could also exist when the participants thought the system was developed by the researchers [61].

Second, we mainly evaluated the usability and utility of VRGit in the context of interior design. Future work could further evaluate other parameters such as correctness, consistency, and scalability and potentially compare VRGit with a control condition where users are provided with the same tasks without using VRGit. It is also worth investigating how different task contexts and populations could affect the usage of VRGit. For example, future work could evaluate the usage of VRGit in contexts such as game design and architecture, and investigate how people's relationships (e.g. friends, colleagues) and skills (e.g. high VCS literacy but low VR literacy) could affect their usage of VRGit.

Finally, our system was designed for synchronous collaboration in VR. It is unclear how well the system can be extended to asynchronous collaboration where multiple users are offline but still need to communicate and maintain awareness asynchronously. VRGit was designed based on a limited set of content creation operations that take effect on immersive 3D scenes, which could limit its uses. Future work could look into how to extend it to support more operations on different granularities of 3D scenes. For example, it would be useful to be able to store operations on vertices and meshes of a 3D model, which is common in tasks such as 3D modeling.

5.7 Conclusion

In this chapter, we presented a new Version Control System (VRGit) for collaborative content creation in VR. VRGit enables novel visualization and interactions of History Graphs in VR that allows users to easily view and navigate version history, create branches, preview and reusing objects from multiple versions. Our system also facilitates communication and awareness between collaborators and the version history, whether users are working in the same version or different versions. Through an exploratory lab study, we found that our system enabled users to easily manage non-linear version histories, communicate with collaborators, and maintain workspace awareness in VR.

CHAPTER 6

VRCopilot: Intelligent Support via Generative AI in Immersive Content Creation[1]

6.1 Introduction

While existing immersive authoring tools make it intuitive for users to visualize their design concepts for 3D scenes in VR, most current 3D layouts such as architectural designs and game scenes are laboriously created through manual placement of 3D models. This manual process is not only tedious and time-consuming, but can also limit the user's ability to explore a diverse range of ideas [126]. In recent years, generative Artificial Intelligence (AI) models have emerged as powerful means for automatic generation of intelligible text [193], photorealistic images [209], videos [7], music [164], and 3D layouts [198, 157, 264]. By leveraging generative models, we can potentially provide users with automatically generated 3D layouts during the process of immersive content creation, enabling users to save time and effort while exploring alternative design possibilities.

Prior work has demonstrated promising results in generating realistic 3D layouts [198, 157, 264] and text-to-layout generation [167, 78]. However, integrating these models into immersive authoring workflows poses unique challenges of how users can collaborate and interact with generative models—specifically, understanding, controlling, and refining model outputs in immersive virtual environments. This difficulty is compounded by generative AI models' well-known issues with transparency, controllability, and user agency. Current generative models for 3D layouts use either room sizes (e.g., [198]) or text captions (e.g., [78]) as prompts. It is difficult for users to define their design objectives, such as requesting layout designs with elements in particular locations or sizes (as specified in Fig. 1.3.b).

[1]Portions of this chapter were adapted from [287]

In this chapter, we introduce VRCopilot, a mixed-initiative system that integrates pre-trained generative models into immersive authoring workflows. VRCopilot is instantiated in the context of layout design for indoor scenes, where users are able to co-create with generative models via requesting, controlling, and refining generative models' outputs in VR. VRCopilot introduces two key interaction techniques: *(1) multimodal specification* and *(2) intermediate representation.* Inspired by multimodal interactions such as "Put-that-there" [22], our system enables users to use speech and simultaneous pointing to specify their creation needs, increasing the naturalness and economy of language description in the immersive environments. For instance, users can point to a location in the room while saying "create a wooden chair here". As a response, the system will generate three suggested objects for the user to choose from. Besides, to help users co-create with the generative model in a more transparent and controllable way, VRCopilot proposes the notion of *wireframes* as intermediate representations for the generated outcomes. Inspired by the concept of low-fidelity prototyping in Human-Computer Interaction [214, 32, 230], wireframes are 2D representations of 3D layouts similar to floor plans in interior design. These representations can be hand-drawn by users together with speech specifying the their types, or suggested by generative models. VRCopilot allows users to iteratively refine the design with generative AI by enabling them to convert between intermediate representations and 3D layouts.

Taking the above techniques together, we propose three ways of human-AI co-creation in VR enabled by VRCopilot: (1) manual creation, where users create individual objects to complete a layout design by creating from a catalog menu and multimodal specification; (2) automatic creation, where users request suggestions from generative models for full-room layouts and refine their outputs; and (3) scaffolded creation, where users co-create intermediate representations with generative AI for guiding the final layout design.

To provide an in-depth understanding of the human-AI co-creation process in VR, we conducted two rounds of user studies. Our first study aimed to compare user experiences of creating 3D layouts with and without AI. Specifically, we compared creation without AI using manual placement and creation with AI using generative models. We found that co-creating 3D layouts with generative models is generally more preferable as it could save users' effort while resulting in 3D layouts with more complete *functionality* and diverse *color palette.* However, users struggled with the generative model's non-deterministic output, where the generated results might misalign with the user's design goals due to the lack of controllability of the generative model.

Based on the insights and challenges from the first study, we further evaluated VRCopilot by comparing different levels of AI automation in the creation process, by comparing manual creation, automatic creation, and scaffolded creation. We found that users' sense of agency significantly decreases in the order of manual creation, scaffolded creation, and automatic creation. We also found that users felt significantly higher creativity in manual creation, compared either scaffolded creation or automatic creation, with no significant difference found between the latter two conditions. Our qualitative interviews reveal that the design of wireframes can increase users' sense of agency and that having multiple suggestions in multimodal specification can make users feel more creative.

In sum, our chapter makes the following contributions: 1) VRCopilot, an immersive authoring system that enables users to interact and co-create with generative AI models in virtual immersive environments; and 2) empirical results gained from two user studies that provide insights on user experiences such as perceived agency and creativity, as well as potential and challenges of human-AI co-creation in immersive authoring workflows.

6.2 Related Work

VRCopilot draws inspiration from prior literature on 3D scene synthesis using generative models and creativity support with generative design.

6.2.1 Generative Models for 3D Scenes

The demand for automatically generating 3D scenes has never been higher in the domain of gaming, AR & VR, architecture and interior design. In the field of computer vision, this topic named 3D scene synthesis is gaining popularity and prior researchers have explored generating new 3D scenes via various input including images [163, 87], text [281, 60], or room shape [198]. A key line of work is 3D *indoor* scene synthesis, which refers to the task of automatically generating a set of 3D furniture objects along with their positions and orientations, given a room layout [288]. Some of the early work in this space offered suggestions using hardware-accelerated Monte Carlo sampler based on interior design guidelines [170]. Follow up work has been focused on data-driven approaches, given the rise of large 3D object datasets such as SUNCG [232] and 3D-FRONT [85]. The data-driven approaches can be approximately categorized into graph-based [264] and autoregression-based approaches [265, 216, 268, 198]. Graph-based approaches encode 3D layouts as scene graphs, where objects are nodes, and the spatial relationship between objects are edges. This method treats the task of generating 3D scenes as generating directional graphs. The main motivation behind this is to process it with graph convolutional networks. Most notably, Ritchie et al. [216] introduced a CNN-based architecture that operates on a top-down image-based representation of a scene and inserts objects in it sequentially by predicting their category, location, orientation, and size. More recently, autoregression-based approaches have been introduced. Wang et al. introduced SceneFormer [268], a series of transformers that autoregressively add objects in a scene. ATISS [198] simplifies the process by proposing a single model trained end-to-end to predict all attributes. Most notably, ATISS encodes 3D objects' positions, rotations, and scales in transformers for training. More recently, DiffuScene utilizes a denoising diffusion model that is able to generate more plausible and diverse indoor scenes [253].

6.2.2 Creativity Support via Steering Generative Models

The acceleration of AI capabilities has enabled human-AI co-creation in domains such as drawing [59, 75], creative writing [48], video game content creation [105], and music composition [117, 164]. For example, Bach Doodle [117] is able to complete a music composition in the style of J.S. Bach by requiring users to only write a few notes. While recent research has focused on building co-creation experiences in 2D interfaces, there has been relatively little HCI work examining how to design interactions with these state-of-the-art generative models to ensure they are effective for co-creation in the immersive environments. Our research contributes an understanding of how interactions with these AI models can be designed, how they affect the immersive authoring experience, and users' attitudes towards AI co-creation in VR.

Integrating existing generative AI models into creative work presents unique challenges in itself such as adapting actions of AI based on users' preferences [47, 130, 250]. Research has also observed that users desire to take initiative in their partnership with AI, and thus sought to provide steering tools to make AI align with users' creative goals. For example, TaleBrush [47] uses a combination of line sketching and natural language narration to create stories. DreamSketch [130] uses sketches as input for the generative design of 3D models. In the domain of 2D layouts, Scout [250] uses high-level constraints based on design concepts to generate multiple designs. Building on this need, our work investigates how users express their preferences to generative AI through multimodal creation and intermediate representations in VR.

6.3 VRCopilot

VRCopilot is a mixed-initiative immersive authoring system that enables users to co-create 3D layouts with pre-trained generative models in VR. Users can ask generative models to generate full-room layouts or use multimodal specification to create individual objects. They can also manually place objects from a catalog menu or request suggestions from the system using multimodal interactions (i.e., pointing and speaking). VRCopilot further allows users to create *wireframes* — intermediate representations that help guide and refine the layout generation process. We detail our system design in the sections below.

6.3.1 Scope

We situate our design of VRCopilot in the context of interior design tasks, where users can place pre-made 3D furniture models in a virtual apartment. Interior design requires balancing constraints (e.g., functional requirements and space limitations) with aesthetic preferences. It has been the application domain of many prior immersive authoring tools [119, 45, 282] and is a common use case for Mixed Reality. For example, several popular home goods stores, including IKEA[2], integrate features that allow customers to virtually preview furniture arrangements in their own homes before making a purchase. VRCopilot includes $7,302$ furniture models from 3D-FRONT [85], a large open-source dataset of furniture objects and textures.

Designing VRCopilot for interior design allows us to evaluate it in a realistic domain with demonstrated utility. However, we believe many of the concepts behind our design could generalize to other spatial design tasks, as the low-level tasks (e.g., object instantiation, customization, and manipulation) and multimodal interactions with generative models are broadly applicable across domains.

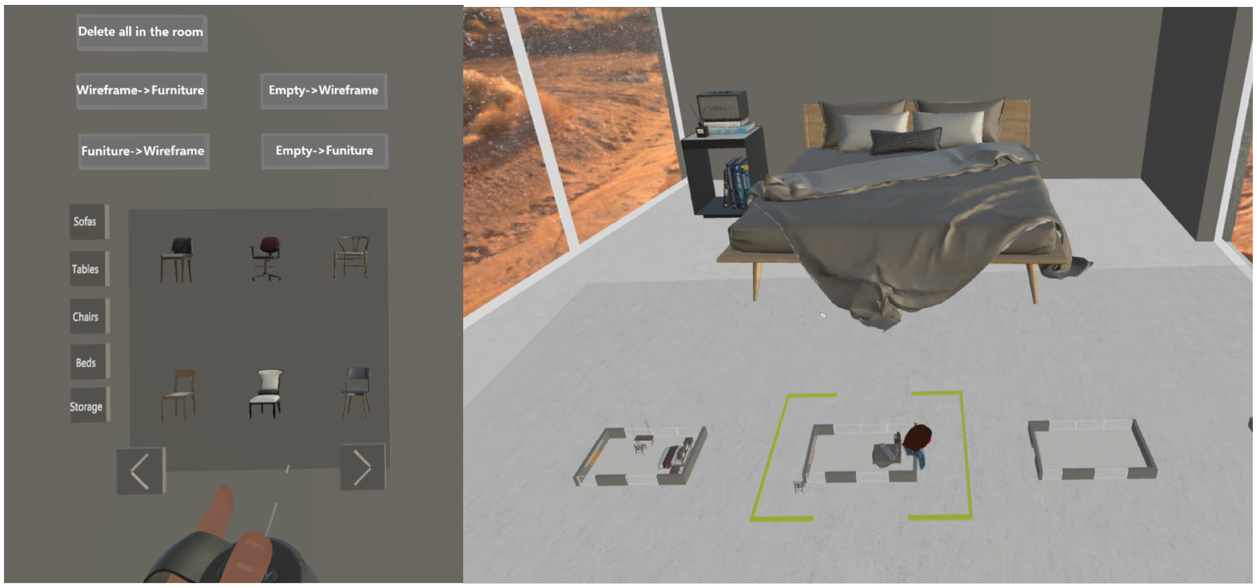
6.3.2 Immersive Authoring Features

VRCopilot is designed as an *immersive authoring* tool, meaning users design a room layout while immersed in that room (in VR).

6.3.2.1 Importing Models

Users can manually import furniture models into the virtual environment from a catalog menu bound to their non-dominant hand, as seen in Fig. 6.1a. Using the catalog menu, users are able to choose from different categories of furniture such as tables and chairs from a sub-menu. Each page on the catalog menu contains six furniture items and they can turn pages to navigate more items. After a furniture model is imported, users can manipulate and place the model via direct manipulation using the VR controllers.

[2]https://www.ikea.com/global/en/newsroom/innovation/ikea-launches-ikea-place-a-new-app-that-allows-people-to-virtually-place-furniture-in-their-home-170912/



(a) Palette Menu. (b) Multi-workspace.

Figure 6.1: User interfaces in VRCopilot including (a) a palette menu where users can select and furniture from the catalog, and (b) a multi-workspace visualization that allows users to work and switch between multiple versions.

6.3.2.2 Design Exploration

The ability to explore multiple alternatives is crucial to supporting creativity in design tasks [226, 109]. For example, in the realm of interior design, designers typically develop a variety of versions to present to clients or stakeholders. To facilitate the exploration of multiple design variations, VRCopilot offers multiple empty workspaces or templates for users to work on (as seen in Fig. 6.1b). Users can easily switch between workspaces to work on different versions by navigating a list of miniatures in VR. This creativity support is inspired by the concept of "World in Miniature" (WIM) [242] and recent work on version control in VR [282]. To help users reuse partial layouts across different versions of the designs, VRCopilot also includes a copy & paste feature, shown as additional buttons bound to the handheld menu. This feature allow users to copy multiple objects and paste them either in the same workspace or other workspaces.

6.3.3 Generative Model in VRCopilot

We used ATISS [198], an open-source generative model for indoor scene synthesis using autoregressive transformers, as our generative model. ATISS is trained using an open-source dataset of 3D models called 3D-FRONT [85], from which we also build our furniture catalog. This model takes room dimensions parameters as prompts and generates reasonable furniture arrangements of the full-room layout. It is also versatile for user inputs such as asking for a suggested placement of a given furniture item, or asking for a suggested furniture item for a given location. We chose this model because it has been used as baseline models for work in the domain of indoor scene synthesis (e.g., [78, 269]).

We integrated ATISS in VRCopilot and can generate suggestions for full-room layouts on demand. In VRCopilot, users can access the generative model via either voice commands or the catalog menu (as shown in Fig. 6.1). Upon receiving the request, our system can fill the entire room by placing suggested furniture in the user's current workspace. Users can delete the suggestions and also run the generative model repeatedly. Our system supports multiple room sizes, shapes, and types (e.g., bedrooms and living rooms), and can be easily extended to support arbitrary room sizes and shapes (e.g., users can draw the room) and the backend generative model can adapt to these specifications automatically.

6.3.4 Multimodal Specification

Existing immersive authoring tools enable users to directly manipulate virtual objects similar to how they would manipulate them in the physical world. However, direct manipulation does not suffice for all needs during immersive authoring. One clear weakness of direct manipulation is that it makes it difficult to identify or manipulate a potentially large sets of objects. For example, there is a massive number of objects and styles in our furniture catalog (e.g., the catalog is based on 3D-FRONT that contains 7,302 furniture items with textures). It is difficult for users to specify generating a chair with *minimalist* style via direct manipulation. On the other hand, the inherent ambiguity of natural language instructions makes it difficult to use pronominal reference to objects in the scene [50]. For example, it is hard for the system to understand which location the user is referring to when the user describes "generate a chair here" without additional contextual information.

Inspired by multimodal specifications in graphical interfaces such as "Put-that-there" [22], VRCopilot allows users to use speech and simultaneous pointing to specify their creation needs, increasing the naturalness and utility of language description in the immersive environments. Our system can process users' natural language voice commands and categorize them into several possible intents:

- *Object Generation*: generating individual objects;
- *Object Regeneration*: regenerate individual objects;
- *Object Duplication*: duplicate selected objects;
- *Scene Completion*: initiating a request for generating the full-room layout;

- *Wireframe Generation*: initiating a request for generating the (see specifics in Section 6.3.5);
- *Wireframe Labelling*: assigning an object type to a wireframe (see specifics in Section 6.3.5);
- *Deletion*: delete selected objects.

Users can point to any location in the scene while verbally requesting that VRCopilot suggest furniture to be placed at the designated point. In their specifications, users can use pointing to specify the *location* and voice to specify the *object type* and its *style* and *material* (e.g., "Generate a minimalist wooden chair here"), as seen in Fig. 6.2. Since its furniture catalog is built on 3D-FRONT, VRCopilot contains a limited number of 21 object types such as beds and chairs, 19 unique styles such as Modern and Japanese, and 15 unique materials such as Wood and Metal. Our system's language processing currently ignores out-of-range intents, such as indication of color or shape, due to the repository's limitation of not supporting color or shape labels. For example, when users indicated a *red* chair, the system would retrieve a chair of any color. VRCopilot also does not include other natural language intents such as repositioning and object selection, as these operations can be more easily achieved via direct manipulation as observed in our pilot tests. Upon parsing the user's request, three suggested furniture items fitting the user's provided criteria are visualized in front of the user (as seen in Fig. 6.2), and the user can choose one of the three to become a part of the scene (as seen in Fig. 6.2).

6.3.5 Intermediate Representation

One of the key challenges of human-AI co-creation is the lack of transparency, control, and user agency [10, 30]. To help users co-create with the generative model in a more transparent and controllable way, VRCopilot proposes the notion of *wireframes* that is used as intermediate representations for the generated outcomes. We took inspirations from low-fidelity prototyping that is commonly used in Human-Computer Interaction [214, 32, 230]. For example, prior work has explored using low-fidelity prototypes such as paper prototypes to quickly scaffold user interface design [230] or Play-Doh as intermediate representations to represent high-quality 3D models [184]. In VRCopilot, wireframes are designed as 2D representations of 3D layouts such as floor plans in interior design. These representations can be hand-drawn by users together with speech specifying their types. For instance, users can use the cursor of the raycast from the controller as the pen tip. They can place the cursor on the floor and start drawing by pressing a button on the controller, while saying "Mark this

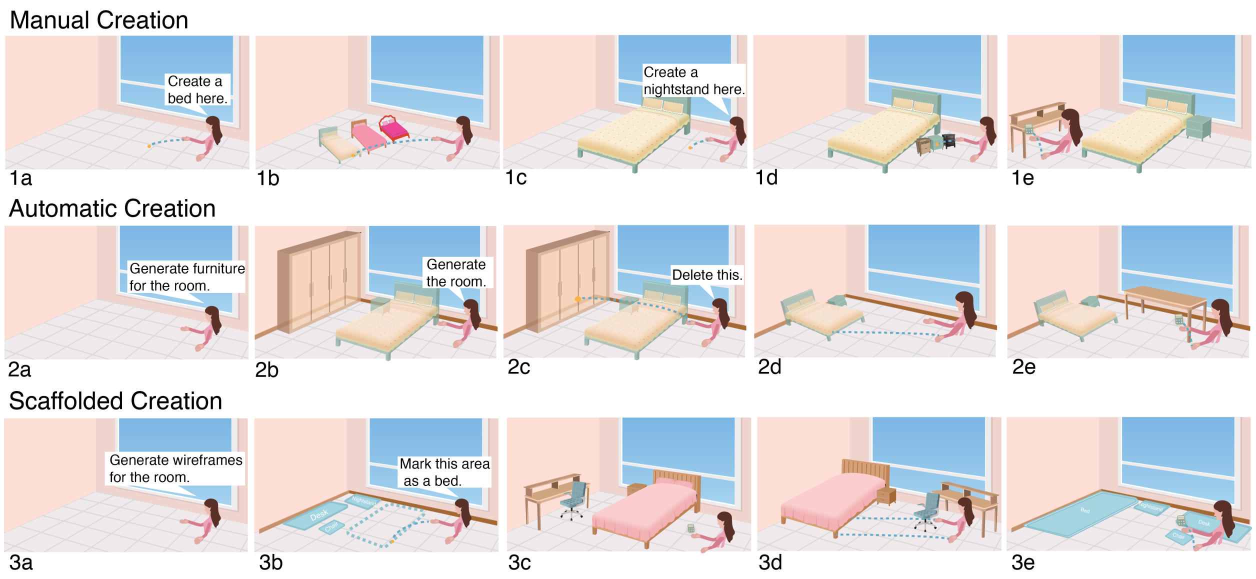


Figure 6.2: VRCopilot proposes three ways of human-AI co-creation in virtual immersive environments: manual creation (1a-e), automatic creation (2a-e), and scaffolded creation (3a-e).

area as a bed." Upon the intent is recognized, the system will normalize the drawing into a rectangular plane with a text label of the object type (e.g. "Bed") attached to it. Users can further adjust the placement and dimension of the wireframe using direct manipulation, similar to manipulating furniture models. Users can build up intermediate representation of the full-room layout design by creating multiple wireframes in the room. Alternatively, users can ask the generative model to offer suggestions of wireframes by initiating a request similar to generating full-room layout. The system can then generate the intermediate representation of the full-room layout and visualize all generated wireframes in the room.

In addition, VRCopilot allows users to iteratively refine the design with generative AI by enabling them to convert between intermediate representations and 3D layouts. For example, users can use voice commands or button presses to turn their intermediate representations into actual furniture models. The system can then interpret the labels and populates the scene with detailed furniture pieces corresponding to the *object type*, *size*, and *orientation* as specified using each wireframe. For objects that are not placed on the floor, such as ceiling lamps, users can draw wireframes on the floor similarly to how they create other objects. The system will then automatically set the y attribute, representing the height, for these objects when populating the scene. Users can also switch back to intermediate representations from detailed furniture design, enabling an iterative design that leverages both lo-fi and hi-fi representations of the layout.

6.3.6 Ways of Human-AI Co-creation in VR

With the above generative model and interaction techniques, VRCopilot supports three ways of human-AI co-creation in VR.

6.3.6.1 Manual Creation

Manual creation enables users to manually create a 3D layout design by creating each furniture item and its placement one after another. Such creation method uses a bottom-up approach, where users start by creating specific furniture items either via the catalog menu or multimodal specification. Once the central pieces are selected (e.g., beds, sofas), users consider how other components can be arranged within the room including placement of furniture, the flow of circulation, and how spaces will be utilized.

Figure 6.2 1a-e showcases a typical workflow of how users can create layout designs using manual creation: a) Users first use multimodal specification to ask the system to generate a bed. b) They can then pick from one of the suggestions as a central piece in the room. c) Users use multimodal specification to create a nightstand, and d) pick the one that best matches the style of the bed. e) Then they start using the catalog menu to create other furniture models such as desks to complete the layout design.

6.3.6.2 Automatic Creation

Automatic creation enables users to ask the generative model to generate full-room layouts. After the suggested layout is given, users can modify the layout design based on their design goals. This could include adjusting the placement of objects to avoid overlapping objects or to remove unwanted objects.

Figure 6.2 2a-e demonstrates a typical workflow of how users can create layout designs using automatic creation: a) Users start out with no concrete design goals and ideas so decide to ask the system to generate the full-room layout for them. b) After the system processes the request and visualize the layout suggestion to the user, c) they can manipulate furniture models such as deleting the wardrobe to fit their own preferences. d) They decide to move the bed to be further away from the window. e) They also decide to add additional furniture models from the catalog menu that are missing from the generated layout such as additional desks.

6.3.6.3 Scaffolded Creation

Scaffolded creation enables users to create intermediate representations, i.e., wireframes, to scaffold their designs. Such a creation method uses a top-down approach where users begin with a broad, overarching vision of the floor plan by creating wireframes in the immersive environments. They can draw their own wireframes and ask for generated wireframes. They can also modify the placements and sizes of wireframes, and convert between wireframes and furniture layouts.

Figure 6.2 3a-e a typical workflow of how users can iteratively create layout designs using scaffolded creation: a) Users first ask for generated wireframes from the system. b) Upon getting the results from the generative models, they can draw their own wireframes such as a bed and rearrange the wireframes. c) They can turn the wireframes into furniture layouts via a button press. d) They can then manipulate the furniture models to further fine-tune the design. e) Once they want to explore an alternative design, they can switch back to wireframes for generating another layout option.

6.3.7 Implementation

VRCopilot is developed using Unity (version 2021.3.20f1) and integrates plugins from Meta Oculus and the Microsoft Mixed Reality Toolkit (MRTK), enabling operation on Meta Quest and Rift VR headsets. The application incorporates advanced voice recognition and response capabilities through integration with the ChatGPT Audio Model (whisper-1) and Chat Model (gpt-4-turbo), with the latter hosted on a dedicated GPU server equipped with an Nvidia RTX 4090 graphics card. A comprehensive system architecture is depicted in Figure 6.3.

6.3.7.1 Integration with ChatGPT Models

Interaction with ChatGPT models is facilitated through voice commands. The system captures user voice input via the microphone, converting the audio to an .mp3 format. This file is then translated into text by the ChatGPT SpeechToText model (whisper-1) through an HTTP request. The resulting text is processed by the ChatGPT Chat Model (gpt-4-turbo), which identifies the user's intent from the predefined categories and extracts relevant parameters such as furniture styles or categories. The responses, formatted as JSON, are parsed by the Unity client to execute the corresponding actions. While most actions are deterministic, actions requiring the generation of new items (e.g., "generate a chair in a modern style") involve a selection process from a set of items meeting the specified criteria.

6.3.7.2 Communication with the Generative Model

For tasks that involve the generation of new furniture, VRCopilot employs socket communication with a generative AI model, ATISS. Furniture attributes (unique ID, position,

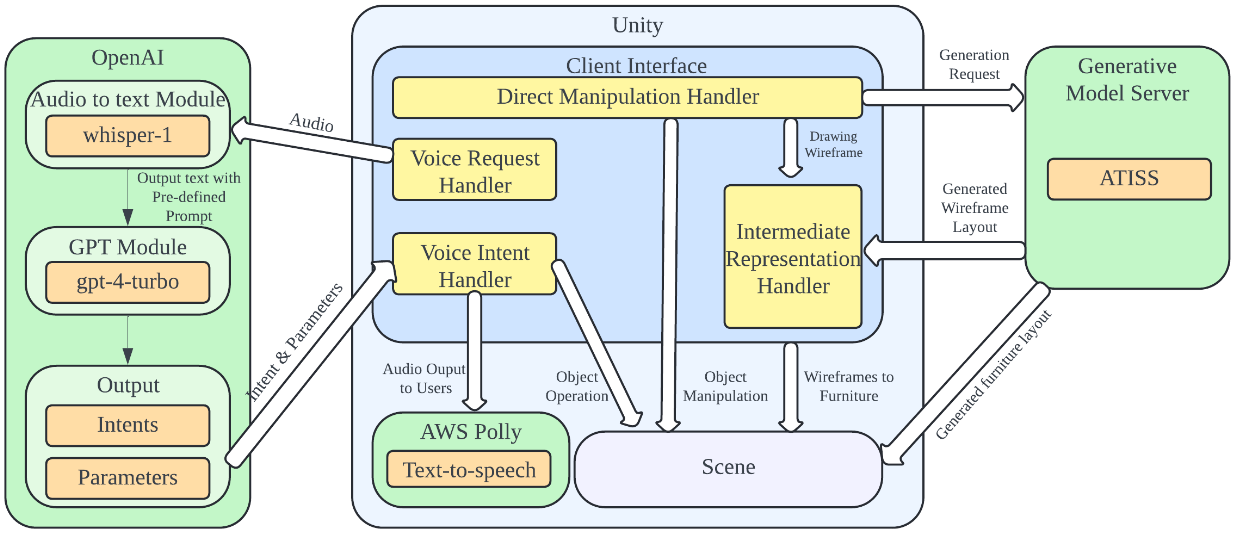


Figure 6.3: The system architecture of VRCopilot.

rotation, scale) are encoded in JSON and sent to the server. Upon completion, the server returns a JSON response with the furniture items that meet the established criteria, which the Unity client then processes and renders in the virtual environment.

6.3.7.3 Multimodal Feedback Module

To enhance user interaction, VRCopilot integrates a feedback loop through AWS Polly Text-ToSpeech model. After processing an intent, the system generates textual feedback corresponding to the user's request, which is then converted into speech. This multimodal feedback mechanism provides real-time auditory confirmation of actions taken within the virtual environment, enriching the user experience.

6.4 User Study 1

To understand the effectiveness and challenges of co-creating with generative AI in immersive environments, we fist sought to compare immersive authoring with and without AI. Prior research has provided some insights on how people collaborate with generative AI in creative domains (e.g. music [164] and painting [46]). We extend this line of work by understanding people's behaviors and attitudes when working with generative AI in virtual immersive environments. We conducted a qualitative comparison study between two conditions: 1) immersive authoring using the conventional interfaces (e.g., via direct manipulation and menu selection), 2) immersive authoring with conventional interfaces and generative AI models. We use this study as the first stop to eliciting the challenges that users perceived when co-creating with AI in VR.

6.4.1 Participants

We recruited 14 participants (10 women and 4 men, age 20—28) from a university through public email lists. All participants had prior experience using VR devices and were compensated with $30 USD Amazon gift cards for two hours of their time.

6.4.2 Procedure

During the study, users were first given a tutorial of the system that covered individual features of the system including the control of direct manipulation and the usage of the generative model. The tutorial lasted about 30 minutes. Then, participants were asked to design an empty apartment, consisting of two bedrooms and one living room, under two conditions: 1) with conventional immersive authoring interface, 2) with the conventional interface and the generative AI model. The room sizes and types were pre-configured, in order to encourage participants to focus on the co-creation process. The order of the conditions was counterbalanced. Each condition took about 15 minutes to complete. In each condition, participants were asked to aim for finishing three versions of the apartment with at least three items in each room. This instruction was not a strict requirement, but rather a means to encourage participants to design multiple variations of the apartment. After both conditions were finished, we conducted a retrospective interview with participants. Our study protocol was approved by our institution's IRB.

6.4.3 Analysis

We transcribed and conducted a thematic analysis [24] of the interview data. To assess the creation results from participants, we designed an evaluation that elicits emerging patterns of users' creation through an evaluation workshop. One design expert, a full-time architect with 2 years of working experience, was invited to participate an evaluation workshop with one experimenter that took 90 minutes. The evaluation workshop was held remotely where the experimenters screen shared to the expert. The expert then went through the top down images of the creation results from all participants under each condition. The order of showing the creation results is completely randomized and the expert was not informed of how the design were created under each condition. Then the expert were asked to use an inductive approach to observe the top down images under each condition and use open-coding to elicit emerging patterns in each condition.

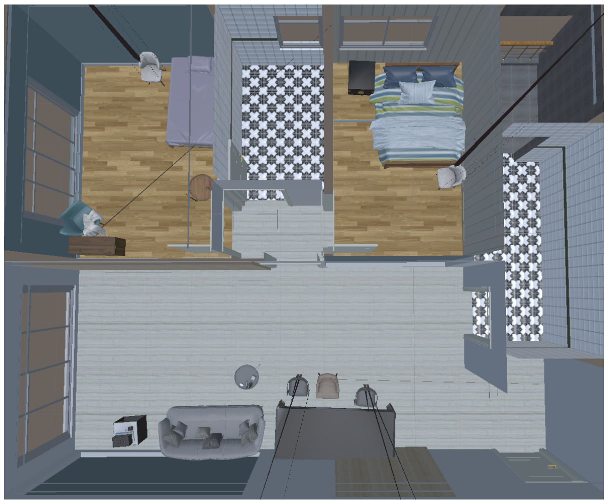
6.4.4 Results and Insights

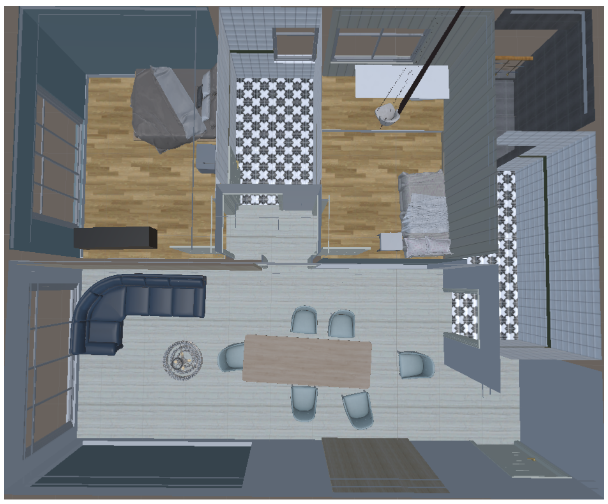
Below are the insights gained from the qualitative user study and the expert evaluation:

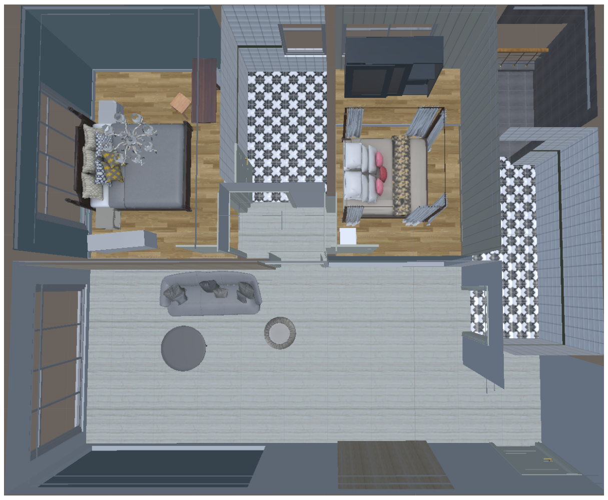
Generative models provide less user agency. Agency refers to the awareness and control over one's action and their results [273]. We found that participants reported feeling less agency over the creation results when co-creating with AI. While the generative AI models could make meaningful layout suggestions that help users explore different ideas, the generated suggestions sometimes misaligned with users preferences in terms of the functionality and other considerations of the layout design. For instance, P2 said "I really have no idea what was going to come out when I did it [using the generative model], like I did not at all expect a bookcase in the middle of the living room, even if it would make sense for that room." P7 also commented on their agency when comparing creating with and without AI: "When there's no AI intervention in the process it is just me thinking about what is the best circulations? What is the best looking furniture to be placed in the room? Those are my primary concern when I was doing it. So I will say I was the most in control when I was doing the first task [without generative AI models]."

Generative models are useful for sparking different ideas. One key dimension of creativity is the ability to explore different ideas [44]. Participants reported that the results from generative AI models can provide inspirations for the layout design that they did not come up with. For example, P8 commented that "It brings up new ideas I hadn't thought about before... also because when I first create the room, I pretty much put in what my favorite idea is for so when I create or generate a new room, it adds more inspiration than what I already had started off with."

Creating with generative models can lead to more diverse functionality and color palette. Functionality refers to the ability of a space or its components to serve a specific purpose or function effectively and efficiently. We found that creation results with the help of AI encompass more diverse functionalities. Specifically, the expert observed more diverse object types in each room that can support different activities. For example, the bedrooms shown in both Fig. 6.4c and Fig. 6.4d include desks (for working), wardrobes (for clothes), and bookshelves (for storage). Color palette refers to the selection of colors used in a design, including primary, secondary, and accent colors, which contribute to the overall mood and atmosphere of a space. We found that creation results generated with AI generally have a richer color palette (seen in Fig. 6.4), which contributes to the expert commenting the creation "more exciting."


(a) Creation Results without generative AI (P7).


(b) Creation Results without generative AI (P10).


(c) Creation Results with generative AI. Two bedrooms are created with generative AI while the living room is unfinished (P6).

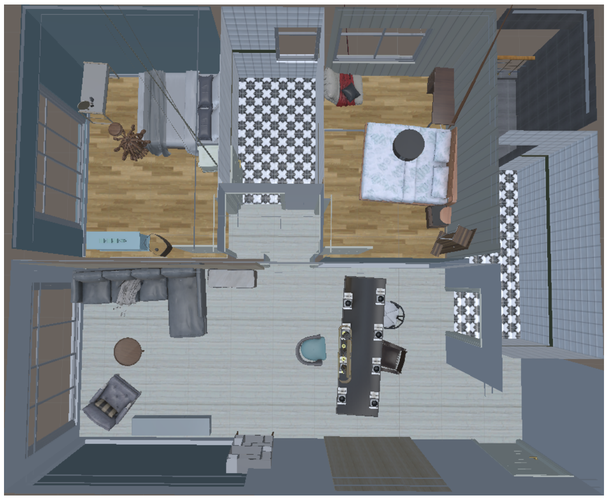

(d) Creation Results with generative AI. All rooms are created with generative AI (P13).

Figure 6.4: Exemplary top-down view comparison of participants' creation results with and without the assistance of generative AI in Study 1.

Creating with generative models can lead to poor considerations of circulation and daylighting. Circulation refers to the flow or movement of people within a space. It encompasses the pathways, routes, and patterns that individuals follow as they navigate and move through an interior environment. We found that creation results generated with AI generally have a poorer circulation. For example, one of the bedrooms shown in Fig. 6.4c includes a nightstand that is blocking the doorway. The dining table in the living/dining room in Fig. 6.4d does not allow for much movement between the two sides due to its close placement to the walls. This is because our underlying generative model (i.e., ATISS) that we utilize does not take doorway or room of movements into consideration when generating. Daylighting in interior design is a design strategy that focuses on harnessing and optimizing natural daylight to illuminate interior spaces. We found that creation results generated with AI generally have a poorer consideration of daylighting. For example, both bedrooms shown in Fig. 6.4c have furniture blocking the windows, making it difficult to harness daylight. This is due to the underlying generative model (i.e., ATISS) that we utilize does not take window placement, size and shape into consideration.

Based on these findings around using generative models, our research team investigated further in the second round of study, that was specifically focused on the mitigation of the issue of user agency and the comparison across different ways of human-AI co-creation (as described in Section 6.3.6). We were also able to design tasks for the second study based on the patterns drawn from the expert evaluation session to further develop our ideas. We describe the second user study in the following section.

6.5 User Study 2

We conducted a second user study to compare three conditions: 1) manual creation using catalog menus and multimodal specification, 2) scaffolded creation using wireframes, and 3) automatic creation using generative AI. We aimed to compare user perceived effort, creativity, and agency, and to elicit potential and challenges that users perceived when co-creating with AI in VR.

6.5.1 Participants

We recruited another 15 participants (5 women and 10 men, ages 19—26) through university email lists. All participants had prior experience using VR devices and did not participate in the previous study (Section 6.4). We labeled the participants as P15-29 below. Each participant was compensated with a $30 USD Amazon gift card for two hours of their time.

6.5.2 Procedure

We designed a within-subject study where each participant experienced all three conditions during the study. Balanced Latin-Square was used to determine the order of the conditions for each participant. For the study setup, we used the Meta Quest 2 connected to a laptop that was running our system in Unity 3D game engine. Each study session began with an introduction of the study and a tutorial of the system that lasted about 30 minutes. During the introduction, participants were introduced to the study and were informed of all the data that would be collected during the study. Participants were then given a tutorial of individual features of VRCopilot. They were given an atomic task after learning each feature to familiarize themselves with the system.

After the tutorial, participants performed a design task under each condition, where they furnished an empty bedroom in VR. In each condition, they were asked to come up with three design solutions for the same room within 15 minutes. If more than three versions were created, they would be asked to turn in the three versions that they were most satisfied with. The following design goals were given to participants for each condition:

- There should be at least 4 furniture types in the bedroom.
- Make sure the top of the window is not blocked by wardrobes / shelves / bookcases.
- There should be enough space for users to navigate in the room.
- There should be a sofa to accommodate seating.
- Try to make the three versions different in both layouts and appearance.

The design goals were created based on design considerations drawn from the expert evaluation in the previous study (Section 6.4) including functionality, day-lighting, navigation, seating, and diversity. Participants were encouraged to design multiple variations of the room based on the design goals. They were notified every five minutes during the task. They were also free to ask for time remaining as well as clarification questions related to the task or the system. However, experimenters were not allowed to give any instructions on how to design the room. After finishing each task, participants were asked to fill out a questionnaire while we saved the resulted scenes and the screen recordings from that condition.

After experiencing all conditions, we conducted a semi-structured interview with participants to ask about user experience and their perceptions of each condition. Each interview lasted about 20 minutes. Our study protocol was approved by our institution's IRB.

6.5.3 Measures and Analysis

We evaluated the following metrics via post-task surveys. Participants rated the items on a 7-point Likert scale (1=Strongly Disagree, to 7=Strongly Agree). To answer the research questions, we measured the following aspects: (1) user perceived **effort** via the NASA-TLX [108] questionnaire; (2) user perceived **creativity** from the Creativity Support Index [44], with an emphasis on how well our system help users explore different ideas; and (3) user perceived **agency** adapted from prior work (e.g., Tapal et al. [255] and Lukoff et al. [165]).

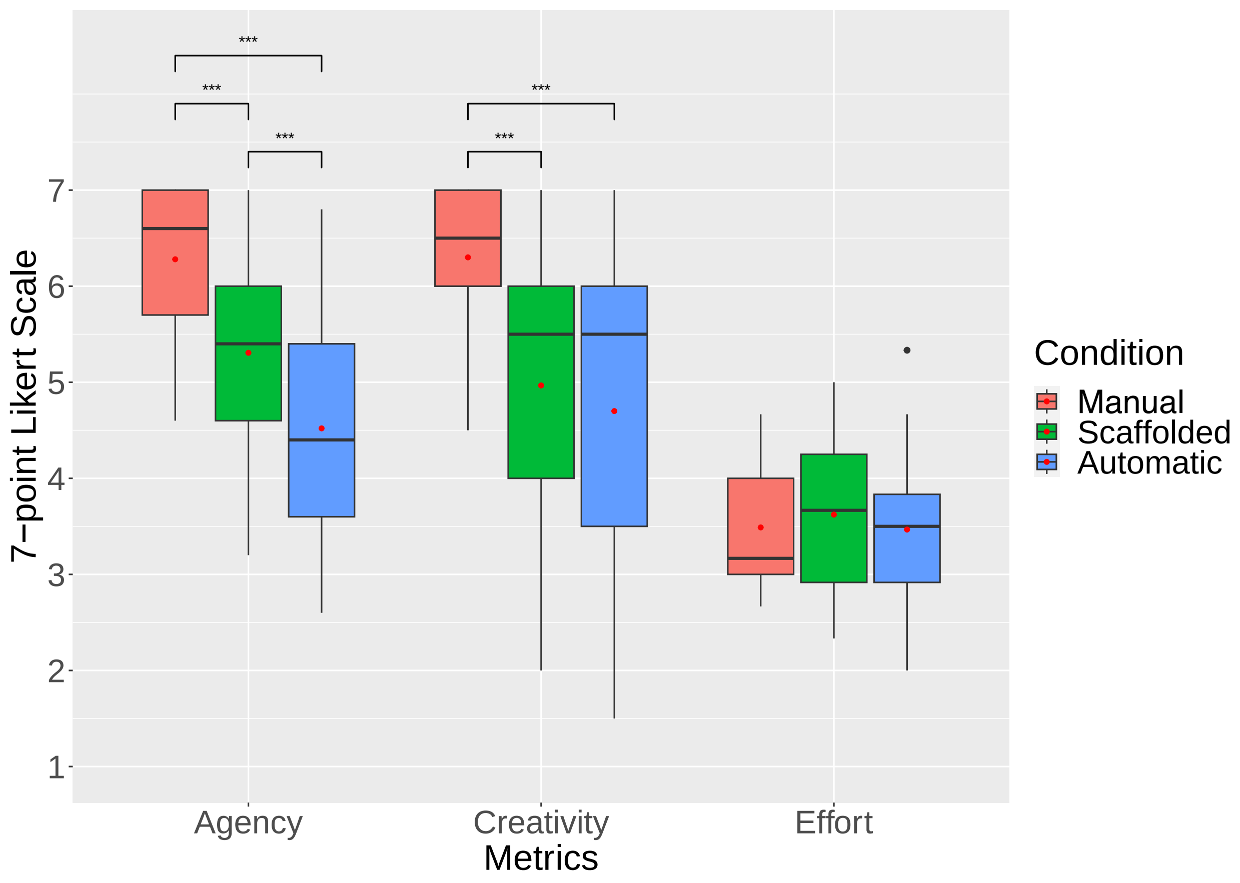


Figure 6.5: Results from post-task survey comparing three conditions in Study 2.

We first performed a Friedman test, to analyze the non-parametric within-subject survey data. For the potential post-hoc analyses, we conducted pairwise comparisons using Conover's test. For qualitative results, we transcribed and conducted a thematic analysis [24] of the interview data.

6.5.4 Quantitative Results

The results of the post-task questionnaire and the Conover's test are aggregated and shown in summarized in Figure 6.5.

A Friedman test was conducted to evaluate differences in participants' perceived agency across three conditions. The analysis revealed a statistically significant difference in the sense of agency across the three conditions ($\chi^2(2) = 20.11, df = 2, p < .001$). Post-hoc analyses with Conover's pairwise comparisons were performed with a Bonferroni correction. We found that participants' perceived control was significantly higher in the manual creation condition ($p < .001$ compared to the scaffolded creation condition and $p < .001$ compared to the automatic creation condition). We also found that participants' perceived control was significantly higher in the scaffolded condition compared to the automatic condition ($p < .001$). These results suggest that the design of wireframes was effective in increasing users' sense of control compared to fully automatic generation from AI.

In regards to users' perception of creativity, we found a significant effect of ways of creation on the sense of creativity ($\chi^2(2) = 17.633, df = 2, p < .001$). Further post-hoc analysis revealed that users felt significantly higher sense of creativity in the manual condition compared to both the scaffolded condition ($p < .001$) and the automatic condition ($p < .001$), with no significant difference found between the latter two. These findings show that participants felt they were the most creative when they were working in the manual creation condition, but similarly creative in the scaffolded and automatic creation condition.

For users' perceived effort, we did not find a significant effect of ways of creation on users' perceived effort ($\chi^2(2) = 0.915, df = 2, p = 0.63$). This indicates that users felt similar levels of effort across three conditions.

6.5.5 Qualitative Results

All participants were able to finish three design variants of the room by the end of the task. To further investigate the reasons behind users' perceptions in subjects such as agency and creativity, we analyzed our interview data and solicited users' qualitative feedback. Overall, our results suggest that users' sense of agency can be enhanced by offering greater control during the human-AI co-creation process, such as control over object types and sizes through wireframes or object styles through multimodal specification. In addition, providing multiple suggestions via multimodal specification can increase users' creativity. We center our findings below around the topics of user agency and creativity in the contexts of human-AI co-creation, interior design, and immersive authoring.

6.5.5.1 Offering greater control could enhance the sense of agency and ownership.

Participants reported having higher agency over the created content in scaffolded creation compared to in automatic creation. Specifically, participants felt that they have control over aspects such as furniture size and placement compared to automatic creation.

> "I felt like I had the most control with the wireframe because in addition to what types of furniture I could also decide what size and how it's positioned. Whereas the others, I think, particularly lost out on the sizing component. Because there

> were a few times, after I tried the wireframe, that I did try to resize furniture, but then realized that that wouldn't work [in the other conditions]." -P23

In addition, participants reported having higher agency in manual creation since they have additional control over the furniture styles.

> "For example, I can pick, only leather chairs, leather sofas, and then have a bed that matches that style... you just got more control over the style itself, rather than just the layout" -P21

We also found that users felt the least agency in the automatic creation since the generated furniture is already fleshed out and decreases their willingness to control or manipulate things, which further decreased the sense of ownership in the created content.

> "Because it feels like that's already there. So it looks like it already looks pretty good. So I wouldn't want to move it too much, and definitely I have less control with it. Because the furniture and everything were chosen by AI, I feel like it doesn't feel fully like *I* designed it." -P28

6.5.5.2 Manual creation can spark creativity via multiple suggested options.

We found that participants felt the most creative in the manual creation, mostly because the multiple suggestions offered via multimodal specification can give users inspirations. Having the options from the system could inspire participants to keep building the room centering around the piece that they chose from the options.

> "When I saw the bed [from the three suggestions], and it's like bright green, yellow, I was like, 'maybe I can make this the theme of this room.' And I was trying to go with this style when I was choosing the other furniture. Then when I saw a bunk bed, I was like, 'maybe this could be a bunk bed for two children,' and I'm styling the room in that way.... I think the [Manual Creation] condition facilitates creativity a bit more just because you can choose between the three options." -P24

6.5.5.3 Users tend to follow the designs generated in scaffolded or automatic creation, leading to a reduced sense of creativity.

While the scaffolded creation and the automatic creation also suggest furniture to the user, participants reported less creativity mostly because they tended to follow the layout that the generative model suggests to them. We found that users tended to feel fixation when the system generated the full-room layout compared to the system suggested individual furniture items.

> "I think having everything laid out for you already, it decreases your creativity. Because you'll have that bias towards the way that it just puts everything. So it's like the bed's here, I might just keep it there." -P21

6.5.5.4 Scaffolded creation enables high-level and unbiased design thinking.

Participants mentioned that scaffolded creation, specifically the design of wireframe, allowed them to focus on the functionality over the styles and enabled them to think "in the layout sense" (P24).

> "I feel like, by creating all those wireframes, I'm actually doing the job of an interior designer, because I'm not the one who's purchasing the actual furniture for the household. I'm just designing how to maximize the utility of the whole space for this household." -P20

> "I think just where things are and how you move around the room. I think that's very important...if the room is cramped or awkward, it's not going to be as good even if it looks really nice. So wireframe, I think, is very good for that just to see it completely unbiased. Because if I just build using the voice or the menu, I can already see things. Like if it looks good, but it's not really functional, I might be biased just because it looks good, and just go with something that doesn't really work. But wireframe kind of takes away from that. And it really lets me focus on the function. And just making sure that everything flows together nicely." -P19

6.5.5.5 The design of wireframes in scaffolded creation enables easy manipulation in VR

We found that the design of wireframes in scaffolded creation made it easier for users to navigate the layout and manipulate distant objects due to the reduced occlusion of 2D planes, compared to handling a full layout with 3D furniture.

> "I think it [wireframe] is useful in, getting through the layout, because with all the objects already in the environment, it's been hard to see around and if there's something behind the big cabinet, you can't reach it. But with wireframes, you can see everything at once." -P18

6.5.5.6 Expectation mismatches with system suggestions reduce user control

We found that users sometimes felt that the system's suggestions, either suggested via multimodal specification or generative AI models, did not match their design expectations, and thus reduced their sense of control. For example, participants were not able to specify the color or the relative size (e.g. big or small) of the objects either through multimodal specification or wireframes. This kind of mismatch is often due to the lack of understanding of the capabilities of the underlying AI models.

> "When I was trying to create a side table to place next to a sofa as a coffee table, either it was not picking up or it was going for more desk or larger-size tables. Even though I switched back and forth between saying small table and side table, it still took a while before it generated something I was happy with." -P23

6.6 Discussion

In our first study, we found that generative models are helpful for idea inspirations. Through the followup expert elicitation study, we found that when co-creating with generative AI models, users can create 3D layouts with more diverse functionality and color palette, but with poorer consideration of circulation and daylighting. Furthermore, we found that generative models could result in lower user agency when it comes to human-AI co-creation in immersive environments. However, this could be mitigated via the design of wireframes as found in our second study.

Our second study demonstrated that among the three ways of human-AI co-creation, manual creation offers users the most sense of agency and creativity. By visualizing multiple furniture suggestions, manual creation can offer design inspirations. Scaffolded creation offers users higher agency compared to automatic creation. This is because users have additional control over aspects such as furniture size and placement via scaffolded creation. Users also found that scaffolded creation can enable un-biased, higher-level of thinking when designing layouts. In automatic creation, users tended to follow what the system suggested to them and not to make changes, leading to the least sense of creativity and agency among the three conditions.

6.6.1 Design Implications

Through the lens of creativity and agency, we highlight the opportunities and challenges of human-AI co-creation in immersive virtual environments, and discuss design recommendations drawn from our results.

6.6.1.1 Offering results of generative AI via intermediate representations

Our design of wireframes offers higher user controllability when working with generative models. Specifically, in the task of creating 3D layouts, users are granted more control over the size and placement of object and think they can view the design in an unbiased way. Besides, the design of wireframes offers unique affordances in VR by making it easier for users to navigate layouts and manipulate distant objects due to less occlusions compared to handling a fully populated 3D scene. This aligns with prior work that utilizes low fidelity representation when working the generative designs (e.g. [130]). Similarly, there has been also a long-standing body of work in Sketch Based Interfaces for Modeling that utilizes both the coarseness and the expressiveness of sketches to guide the detailed generation of 3D models [190]. This demonstrates the benefits of designing low fidelity representations that can prompt more controllable and sophisticated generated content. We therefore encourage future researchers and designers to consider using more advanced intermediate representations of the generated outcome beyond 2D planes on the floor. These representations should capture richer properties of 3D content, such as color and shape, in the immersive environment while still allowing users to easily manipulate the objects and navigate the scene. The note of intermediate representations could even go beyond immersive environments. The concept of intermediate representations can extend beyond immersive environments. For instance, rather than generating lengthy text, Large Language Models could produce an outline as an intermediate representation, allowing users to make adjustments before finalizing the text. Similarly, other generative models could use intermediate representations like image skeletons for pictures or key frames for videos.

6.6.1.2 Offering multiple generated suggestions for inspirations

Our study shows that participants felt more creative and more easily inspired when they can choose from multiple generated suggestions. Users tend to get inspirations from suggestions when they don't have a concrete idea in mind or when they don't want to spend too much time on browsing the catalog menu. Contrarily, when given one suggestion in automatic creation, users tended to follow what the system generates, leading to fixation of thinking. Thus, future researchers and practitioners might consider offering user the ability to choose from multiple generated suggestions, in order to enhance users' sense of creativity. For example, generative AI systems can offer parallel comparison by visualizing potentially diverse generated results for users.

6.6.1.3 Addressing expectation mismatch between users and generative AI

A common challenge across all conditions, based on the study, is the expectation mismatch when unexpected output was generated by AI. Through the expert evaluation, we found that although by co-creating generative AI models users can create 3D layouts that are diverse in aspects such as functionality and color palette, users generally have preferences of the layout design that fall outside of the capabilities of generative AI models. For example, in study 1, layouts co-created with AI showed poorer consideration of circulation and daylighting because the underlying generative AI model was not trained with those criteria in mind. Additionally, users lacked a sufficient understanding of the system's capabilities. This highlights the need for more transparent communication between users and generative AI regarding the system's capabilities and limitations. This aligns with the Explainable AI (XAI) research (e.g., [158, 65, 102]), where researchers aim to provide more transparent explanations of decision-making process of the AI model, with an emphasis on text or images. However, there has been little explorations in the visualization and interaction techniques for making AI models more understandable in the immersive environments. Therefore, future researchers and practitioners should consider designing human-AI systems that can visualize how the generative AI model perceives and completes the user's design.

6.6.2 Limitations

This chapter explored ways of human-AI co-creation in virtual immersive environments and showed empirical results on the comparison among various ways of creation. However, our work has several limitations. First, both of our studies took place in a lab setting with the participants engaged with the system in a short amount of time. The way that participants design 3D layout with a time constraint in the lab setting could be different from how they would design without time constraint outside the lab. Our studies also had a relatively small sample size, which could reduce the validity of our quantitative results. Second, our system was specifically tailored for interior design tasks and had several technical limitations. For instance, as mentioned in Section 3, users could only draw wireframes on the floor, and for objects not placed on the floor (e.g., ceiling lamps), the system automatically set their heights when converting to furniture. The underlying AI models of VRCopilot also had limitations, such as misidentifying voice intents or not supporting color or size in multimodal specifications. Participants occasionally had to retry their intents or regenerate in a few cases when the system misidentified voice commands. Future work should seek to provide clearer system status and offer alternatives for misidentified or unsupported intents, as well as further extend the model's capabilities and supported attributes. Lastly, the generalizability of our findings to domains or contexts other than interior design necessitates further investigation. This chapter provides insights into user creativity, agency, and strategies in human-AI co-creation in general. Some findings, however, are more specific to interior design or the immersive virtual environments. Future research should evaluate the adaptability and utility of the system across diverse application domains to determine its broader applicability.

6.7 Conclusion

In this chapter, we presented a mixed-initiative system named VRCopilot that integrates pre-trained generative models into immersive authoring workflows. We introduce three ways of human-AI co-creation in the immersive virtual environment including Manual Creation, Automatic Creation, and Scaffolded Creation. We conducted two rounds of comparative studies that evaluates the potential and challenges of co-creating with generative AI in VR and user perceived creativity, effort, and agency. Our first study revealed that generative AI could offer design inspirations to users but decrease their sense of agency. Our second study suggested that when users use the wireframes in Scaffolded Creation, they felt higher sense of agency compared to Automatic Creation. Manual Creation offers users the most creativity and agency. We provide insights on the opportunities and challenges around human-AI co-creation in the immersive environments and make recommendations for future research and design.

CHAPTER 7

Auggie: Encouraging Effortful Communication via Authoring Augmented Reality Experiences[1]

7.1 Introduction

The strategies originally developed to help computer users rapidly navigate the digital world [18] have strongly influenced the design of digital social platforms. For instance, single-click reaction buttons frequently accompany or replace prompts for free response that could include personalized voice, text, and images [224]. Autocomplete features are also increasingly used, wherein fixed presets are selected over custom expressions [68]. While low barrier-to-action digital interaction techniques can allow users to quickly initiate social interaction, too little effort can lead to concerns around authenticity [176], trust [125], and feelings of support [274, 237]. These are critical elements of *meaningful* social connections, which involve emotional impact that can improve the quality of people's lives and relationships [162]. For example, "like" buttons are less associated with well-being improvements than more *effortful* counterparts, including tailored expressions of support [106, 29]. Moreover, both researchers [91] and popular media [262, 80] have noted that simply triggering more interactions does not necessarily lead to more meaningful interactions.

To encourage meaningful digital interactions, we draw inspiration from the analog world. People express genuine affection by putting in effort to handcraft letters, scrapbooks, and gifts [161, 86]. By virtue of the medium alone, the effort in analog expressions translates relatively well to the receiver. Handwritten letters, for example, are perceived as more effortful than typing a long email, even if they take the same amount of time and mental energy: they are unique, personal, physically written and received, and, if properly cared for,

[1]Portions of this chapter were adapted from [283]

even "last centuries" [215]. Better yet, if the creation process is documented and conveyed along with the analog gift, they can evoke even more meaningfulness [217].

Yet, as digital communication becomes increasingly mainstream, handwritten letters and handcrafted scrapbooks might feel tedious and antiquated, while modern lightweight and automated communications can feel impersonal in comparison. Research investigating effort-provoking forms of digital communication, such as lengthy text messages, has had limited results, and findings show that effort "can be resented just as easily as it could be valued, depending on the manner in which it arises" [134].

We attempt to address this by lowering procedural barriers to make effort more approachable, designing a communication system that enables versatile, personal expressions of effort. In this research, we study the feasibility of encouraging effortful digital communication and its ability to facilitate meaningful remote interactions. We designed and evaluated Auggie, an iOS app that encourages partners to create digitally handcrafted Augmented Reality (AR) experiences for each other (see Fig. **??**). Auggie encourages effortful communication by making handcrafting for someone playful, personal, and engaging. Each experience, or *auggie*, contains a 3D character that people can send to partners, along with custom photos, animated movements, drawings, and audio. We designed Auggie's interface and easy-to-use features to reduce the tedium and perceived expertise associated with handcrafting while enabling a wide range of creative possibilities for custom expressions and storytelling. All auggies also contain a "behind-the-scenes" view where recipients can see the process a sender took to craft it for them, and thus, the effort they put in. We chose mobile AR because it offers immersive qualities [222] on ubiquitous personal devices, giving us a widely accessible means to approximate and build on the experience of analog handcrafting on a digital medium.

This chapter describes the design and system architecture of Auggie, along with results from a two-week-long field deployment with 30 participants (15 pairs) who used Auggie remotely with close others. Our results show that, on aggregate, Auggie successfully encouraged participants to put effort into crafting visual narratives beyond what is possible in physical reality. Participants found themselves inspired to digitally handcraft supportive, funny, and authentic expressions, which helped facilitate meaningful connection and feelings of presence. At the same time, this research also uncovered some unresolved challenges of effort, such as creativity blocks and appropriateness in certain situations.

Taken together, this chapter contributes: 1) the design and implementation of an effortful communication system using digital handcrafting in AR; 2) results from a two-week field study that evaluated this system's potential to encourage effort on a digital medium through handcrafting and create meaningful interactions; and 3) implications for future design and research on fostering effortful communication in relationships.

7.2 Related Work

7.2.1 Defining Effort

Effort is sometimes referred to as "the work...done to cope with and overcome demand on physical and mental capacities" [134]. However, in interpersonal contexts, defining effort in only this mechanical sense does not capture the nuances of human connections, including the meaningfulness of effort for the people involved [134]. For instance, researchers have identified the importance of effort in relationship maintenance, i.e., the strategies that people employ to maintain and enhance their relationships, such as cheerful attempts to make interactions enjoyable and mutual disclosure [36, 35, 238]. Markopoulos breaks down the effort invested in communication within these relationships, particularly over technology, into *procedural* effort and *personal* effort. *Procedural* effort is the effort required to complete a task (e.g., opening a messaging app), which aligns with the traditional mechanical perspective. On the other hand, *personal* effort is what makes the task special for the recipient (e.g., thinking about something to say that they would like). Personal effort is usually more appreciated than procedural effort [169], but procedural effort can also be valuable when it reflects the personal effort involved (i.e., one's willingness to take on procedural tasks) [132].

Kelly et al. further categorized different qualities of personal effort that people value in social interactions: (1) going out of one's way for the recipient of their effort, (2) crafting for them, (3) overcoming personal challenges for them, (4) devoting time for them, and (5) accounting for them and their beliefs in communication with them [132]. While people can engage in these qualities within existing communication channels (e.g., sending a personalized text message), few prior researchers and social platform designers have *intentionally* designed for these qualities. We still have limited understanding on both how to encourage meaningful, personal effort in digital communication systems, and its subsequent effects on relationships. The present research aims to address this and expand on the design space for effortful communication, exploring the potential to integrate meaningful qualities of effort, specifically "crafting," "devoting time," and "accounting for the recipient and their beliefs," into communication technology.

7.2.2 Supporting Personal Effort in the Digital Realm

Though HCI researchers have acknowledged the need for communication systems to incorporate effort to improve and enhance relationships [137, 161, 215], few works have explored this space. In one recent work, researchers created a physically effortful device to browse one's Twitter feed – a hand crank that one must turn to scroll. They found that with the manual scroll, people focused more on the meaningfulness of the posts they read, while also becoming more aware of their Twitter usage [231]. Although researchers built this project using a social platform, the user's effort led to an individual rather than an interpersonal outcome. Moreover, this research increased the *procedural* effort, rather than the *personal* effort, of a task.

Effort explorations in direct interpersonal communication have primarily focused on text messaging. "Message Builder," for example, is a text-based messaging app that encourages exchanging increasingly long messages. However, the authors found that while effortfully written lengthy messages produced some meaningful interactions, participants were frustrated by the inconvenience of the required procedural effort to access the system and read/write messages [134]. This work highlights the need to balance design goals with users' expectations when increasing procedural effort in a task like text-based messaging. Other research examples in this space include "Lily" [135], which encourages people to manually revise text messages based on lyrical inspirations, and "DearBoard" [98], where communication partners can collaboratively customize message keyboards, though increasing meaningful effort was not the primary goal of either system.

Our research expands on text-based messaging to explore new modes and opportunities for designing around *personal* effort in communication. In particular, we focus on taking an analog task known to require a high level of *personal* effort, and make it less *procedurally* effortful. We do this through Auggie, a playful system focused on the crafting quality of effort [132], engaging people in effort through digitally handcrafting gifts. As opposed to "digital craftsmanship", where a craftsperson may use software to aid in the fabrication of a physical item [138], our use of "digital handcrafting" refers to the process of investing personal effort to produce a digital gift-like experience for another person. In the present work, we use AR as the medium for digital handcrafting.

We chose to focus on the effort conveyed by "crafting for others" as handicrafts are often perceived as authentic and "made with love," which contributes to their value and attractiveness [83, 86]. Love and authenticity are key elements of interpersonal relationships [27]. Therefore, it is not surprising that handcrafting something for another person contributes to relationship maintenance [174]. Moreover, gifting handicrafts has long been a key way to convey personal effort [238]. By investing effort into creating something personal, the craft giver is often viewed as actively investing effort into the relationship [174]. More uniquely, a handcrafted artifact highlights two layers of meaningfulness: in the end product itself, and the process that the creator took to bring it to life [166]. For example, when knitting a sweater, the creator may enjoy the process of picking out the yarn, measuring the size, and finishing and wrapping up the gift as a whole experience that represents their effort towards the relationship [174]. The final handcrafted product is imbued with love from the creator's hands, which personally and physically interacted with it [86].

In the present work, we investigate whether the meaningfulness of physically handcrafting for others can effectively translate to the *digital* realm via an AR representation. We study how we can encourage personal effort through handcrafting in a digital communication tool. We explore digital handcrafting as a means to facilitate meaningful social interactions, and employ a qualitative approach to understand its potential for encouraging effort through Auggie, a mobile application for handcrafting AR experiences. By evaluating Auggie, we derive design implications for future researchers and practitioners to create digital systems that provoke personal effort to improve interpersonal outcomes.

7.3 Preliminary Study

To guide our design of an effortful digital communication system, we first sought to understand how effort is currently conveyed online. Prior research has provided some evidence on how people invest effort in their online interactions (e.g., carefully written text messages or social media posts) [132]. We expand on this work by surveying people more broadly to understand their perceptions of effort on a wide range of digital platforms. We aimed to narrow in on characteristics of online effort that could inform our design direction, such as the frequency at which people engage in it and its current limitations. We use this preliminary study as a first step to define a set of design goals for our system.

We conducted a survey with 70 participants who spoke fluent English using the online participant pool Prolific[2]. Participants were compensated through Prolific for completing the survey following Prolific's recommended compensation guideline[3]. Participants were asked to describe the last time they put in extra effort for someone else online (as a "sender" of effort), the last time someone else put in extra effort for them (as a "receiver" of effort), and if there was anything they wished was different about those situations. They were also asked about their relationship with the people they mentioned, and how often they put in extra effort for them and vice versa. After removing extremely short responses (e.g., one word answers for every question), we obtained 49 valid responses. The remaining participants included 35 male, 13 female, and 1 non-binary person, with ages ranging between 18 to 50 (M=25.02 and SD=7.29). 74% of participants identified as White, 10% as Hispanic, Latino,

[2]https://prolific.co/

[3]https://researcher-help.prolific.co/hc/en-gb/articles/4407695146002-Prolific-s-payment-principles

or Spanish, 8% as Black or African American, 4% as Asian, and 4% as mixed or multi-racial (self-described as White/Asian and Caucasian/African-American).

We analyzed responses to understand when, with whom, and how people engage in effortful communication online and its current limitations. We open coded the responses according to the effort qualities proposed by Kelly et al. [132], as well as the context (e.g., planned or unplanned) and social goals (e.g., comfort, celebration, thinking of you). Two coders coded a subset of the data independently until they reached 80% agreement, and then independently coded the rest of the data. Participants most commonly reported effort conveyed between close friends (38.8%), followed by spouse and significant others (20.4%), friends (20.4%), family members (10.2%), strangers online (8.0%), and acquaintances (2.0%). The following insights from the survey guided the design of Auggie:

Personalization is the most memorable type of effort The most memorable effortful interactions for both senders and receivers are those where the sender tailored the end product to the receiver's interest. This aligns with the "accounting for the recipient and their beliefs" quality of meaningful effort described in prior work [132]. The most common forms for this were gifting, sharing memories, and music. For gifting, senders bought gifts online (e.g., an online game, model kits) that they knew the receiver would enjoy. Shared memories were strengthened by exchanging photos and videos of such experiences. Senders would also put effort into personalizing music for the receiver. For instance, one respondent said their friend sang their favorite song over a call to them to cheer them up. Based on this finding, we conceptualize a system that allows for personal effort through high levels of customization [169].

Effort is demonstrated day-to-day, not just on special, planned occasions We found that most effortful digital interactions occurred in everyday occasions (e.g., sending sentimental songs when one party was *thinking of the other*), while only 30% and 15% of responses for effort conveyed and received, respectively, were for special, planned occasions (e.g., sending a recorded video message on a birthday). Based on this result, we believe that effortful communication should not only be designed for special occasions, but also be applicable on day-to-day occasions. As such, we designed the system to support versatile crafting, where people have the choice to craft simple (e.g., just saying hi) or complex (e.g., elaborate stories) content that they can contextualize for different occasions with images, audio, and animations.

Sense of presence is lacking in remote effort Participants highlighted a deep desire to feel a greater sense of presence from their remote partners. This aligns with prior literature on social presence, or "the sense of being with another," which is well-known to be missing in remote communication [20]. Respondents remarked that they particularly appreciate the effort from remote partners in attempting to be "present" through video chats, voice calls, or simply being available on a digital platform (e.g., a participant noting a friend who was "with them" on a Discord call for several hours to cheer them up). However, there is still a clear demand for an increased sense of tangible presence in any form of remote communication. For example, the aforementioned participant who described their friend singing to them over a call continued to say, "I would like her to come to my place so I can hug her but she couldn't cause we live in different cities." To address this, we incorporate AR in our system to make use of its immersive qualities and ability to simulate physical presence [222], further described in Section 7.4.1.

Based on these findings, our research team underwent several rounds of brainstorming to generate ideas for an effortful communication system. This involved each team member individually presenting ideas, followed by group discussion and iteration, and finally narrowing down on specific concepts that we felt best aligned with the design insights from our preliminary study and prior work. We also built and user tested several low-fidelity prototypes to further develop our ideas. We describe the final system design in the following section.

7.4 Auggie System

Auggie is an iOS app that allows partners to send digitally handcrafted AR experiences (i.e., auggies) to each other in the form of an animated 3D bear. The bear character was inspired by teddy bears, which are common gifts among close partners. The character also facilitates the creation of expressive and embodied animation for shared narratives, which can increase feelings of personalization and presence. This represents one implementation by which we can study the impact of effort conveyed through an activity typically of high *procedural* effort (in this case handcrafting, which usually require time, material costs, and skill) made more *personal* and approachable.

Senders can craft an AR experience by animating the 3D bear, as well as personalizing its outfit and adding 3D drawings, background music, and voice notes. The bear then "travels" via an AR paper airplane to its designated recipient, who can view the AR experience in their own physical environment. The recipient can also view the sender's "behind-the-scenes" creation process, which is automatically compiled by the system.

7.4.1 Situating Auggie in the Physical World

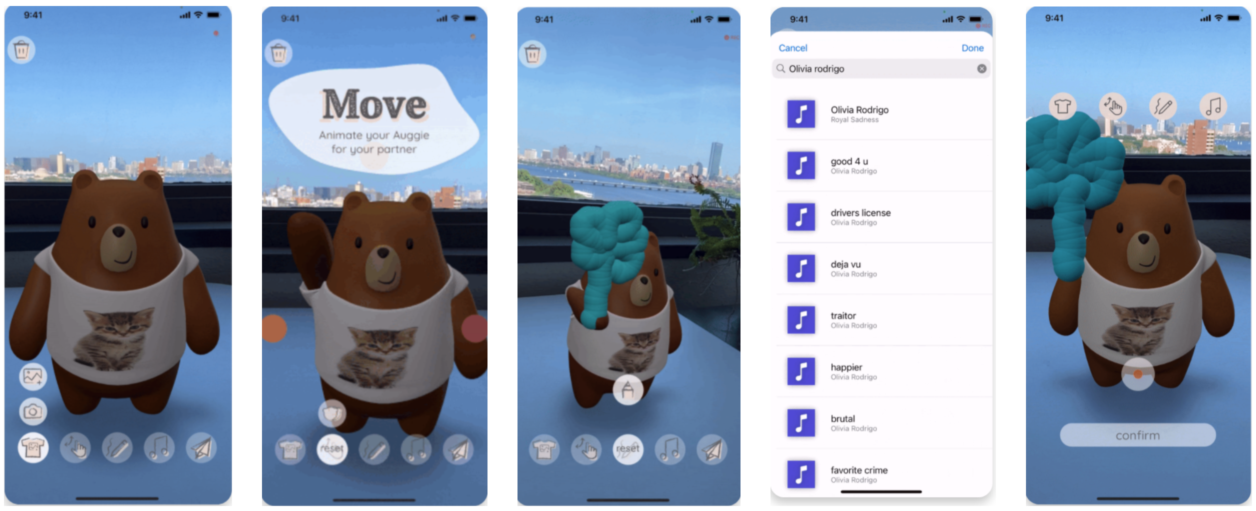
Auggie integrates AR by enabling users to point their phone camera to a surface in their physical environment to craft or view the 3D bear and other elements of the experience. We chose AR as a medium for crafting for several reasons. First, creating and interacting with virtual objects in AR could stimulate the sense of handcraftedness, since a user is personally and physically interacting [86] with an object that feels real [145, 113]. We also aim to make the handcrafted digital experience more engaging using AR due to its unique and promising potential to facilitate playfulness [56] and users' creativity [291]. In addition, the physical and spatial affordances of AR could enable a heightened sense of presence [222], which is often lacking in remote effortful communication, as found in our preliminary study. By overlaying the virtual bear on users' physical environments, Auggie could provide a subjective sense of a partner "being present" in the experience, where the bear represents the partner in one's space. Finally, AR can enhance expressiveness by enabling users to imagine and create experiences for each other that are otherwise not possible in the physical world. The ability of AR to superimpose virtual information onto the physical world has enabled experiences beyond the physical world such as storytelling [16], virtual structure building [103], and museum touring [57]. We aim to leverage the expressiveness and versatility of AR experiences to enable high levels of personalization in Auggie, which our preliminary study suggests is the most memorable type of effort. Altogether, we introduce a unique avenue for handcrafted digital experiences.

7.4.2 Crafting Personalized Auggies

Our survey results and prior work both indicate that people value effort in personalization, such as of gifts, photos, and music. Auggie encourages users to engage in such effortful communication through crafting personalized AR experiences (or auggies) for another person, as if creating a gift. To support different occasions and desired levels of effort, users can personalize the experience as much or as little as they would like, where they could even send "blank" auggies that simply show the bear. The personalization process consists of five components: outfit, animation, 3D drawing, background music, and voice note (as seen in Fig. 3.1).

7.4.2.1 Outfit, music, and voice note

Our preliminary study revealed that people appreciated the effort made to curate photos and hand select music for them. On Auggie, users can select from the photo gallery or take a photo (Fig. 7.1a) to personalize the bear's outfit. Selecting from existing photos



(a) Outfit (b) Animation (c) 3D Drawing (d) Background Music (e) Voice note & Confirm

Figure 7.1: Components of crafting personalized auggies. Users can personalize each component of an auggie by selecting from a set of buttons on the bottom - (a) when *outfit* is selected, users can attach a picture to the bear's t-shirt by selecting from the image gallery or taking a photo; (b) when *animation* is selected, users can author the animation of the bear or toggle the partner bear button to animate two bears together; (c) users can create a 3D drawing by holding the pen button and moving the phone in the physical world; (d) when *music* is selected, users can search and select a background music for the auggie; (e) while reviewing the experience and before sending it, users can add a voice note by pressing the record button.

allows users to engage in digital handcrafting with shared memories or shared interests (e.g., inside jokes). Taking a photo can also draw a connection between the digital and physical worlds, allowing users to integrate elements from the physical into the digital world and potentially feel more present with each other [270], further addressing concerns described in the preliminary study. Users can also set the mood of the experience by selecting background music from the iTunes music library (Fig. 7.1d). In addition to adding existing songs, users can attach a voice note to incorporate more personal and expressive audio (Fig. 7.1e). We included these audio options based on our preliminary study results and prior work, which shows that music influences positive emotion, engagement, and relationships [54].

7.4.2.2 Animation

We included animation to enable a wide range of expressions (e.g., waving, sad, jumping) while fostering a sense of handcraftedness. Animations and animated images are frequently used in the digital communication for storytelling and self-expression through the forms of GIFs, emojis, and memes [99]. These formats not only facilitate shared expressions and address the lack of facial and vocal exchanges in text-based communication channels, but also serve as important strategies to support connectedness and relationship building [244, 116]. Users animate the 3D bear's joints by dragging three circles on the screen corresponding to the bear's spine, left arm, and right arm (Fig. 7.1b). Users can also make the bear jump using a single finger vertical swipe, and rotate the bear using a single finger horizontal swipe. The bear can also walk in the physical environment, moving when a user taps on a destination on the phone screen. To further increase perceptions of presence, we also included a feature where users can toggle the appearance of a partner or companion bear, whose actions are synchronized with the main bear when they are in close proximity, enabling effects like dancing or hugging.

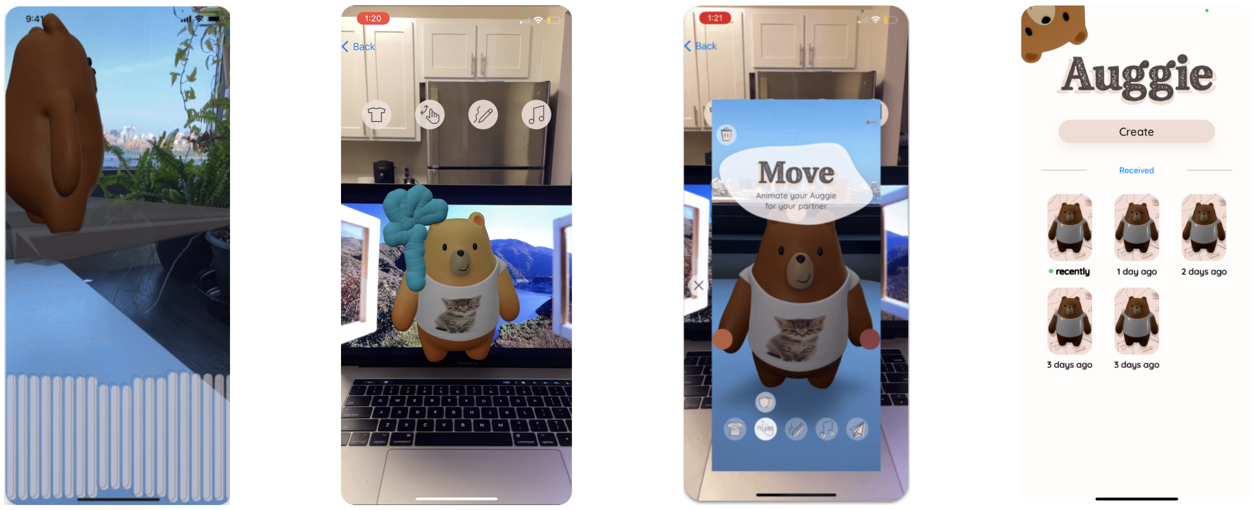
In other digital animation tools like Autodesk Maya and Cinema 4D, animation is an investment of time and procedural effort in many aspects, such as learning how to model 3D objects, encode 3D movements, and set up lighting effects. Comparatively, our system has a small set of intuitive controls, aiming to reduce the amount of procedural effort required to produce aesthetically pleasing digital animations. At the same time, users are able to create animations that are more complex and personal compared to selecting from a fixed library of generic emojis or animated stickers. By creating animations from scratch in their physical environment, users can invest and communicate effort that feels handcrafted and personal.

7.4.2.3 3D drawing

Drawing is known to be an expressive way of creating personal and unique experiences in AR, such as for performances [220] and storytelling [155]. In Auggie, users can press the draw button and move the phone freely in their physical space to create 3D drawings (Fig. 7.1c). We incorporate this feature to allow further personalization as well as the ability to interact with the physical space. This feature also enables users to create virtual props (e.g., a hat for the bear) as part of the handcrafted AR experience.

7.4.3 Sending and Receiving Auggies

After crafting a personalized auggie and before sending it, users may review the whole experience. To send an auggie, users press the "confirm" button (as seen in Fig. 7.1e) and view the bear climbing on a paper airplane in AR (see Fig. 7.2a). Users then need to blow wind into the phone's microphone for the paper plane to take off (see Fig. 7.3). When receiving an auggie, users can point the rear camera to a window or mid-air, and the paper airplane will fly into view. Users can then guide the airplane to land by guiding the camera



(a) Send-off Scene (b) Viewing Auggie in AR (c) "Behind-the-scenes" (d) Inbox

Figure 7.2: Sending and receiving auggies: (a) after confirming, users can blow wind to the microphone to send off the bear and see the real-time audio wave; (b) receivers can view the auggie in AR and see available "behind-the-scenes" videos through the top buttons; (c) receivers can also click on one of the four categories (outfit, animation, drawing, and music) from the top buttons and see a screen recording pop-up of the sender's creation process for the selected category; (d) Received auggies are saved in an inbox for users to rewatch.

to a surface. The auggie is then presented in the receiver's physical environment (see Fig. 7.2b). All the received auggies are saved on the phone and can be rewatched at any time through Auggie's inbox feature (see Fig. 7.2d).

Prior work highlights the importance of incorporating analog approaches to gifting. In particular, "wrapping" a gift in a physical object and mimicking the unwrapping ritual can harbor a sense of physical presence [148]. Therefore, we intentionally incorporated interactions and visual representations similar to the physical world (i.e., blowing wind to make a paper plane fly, guiding its landing) into the sending and receiving process (see Fig. 7.3).

7.4.3.1 "Behind-the-scenes" videos

The perception of the effort one puts into a relationship is essential in determining the satisfaction that one has towards the relationship [238]. Additionally, demonstrating the creation process of a handmade gift and sending it alongside the gift itself can allow receivers to interpret the story behind the gift and increase meaningfulness [217].

However, the effort put into crafting a *digital* artifact may not be easily deduced from

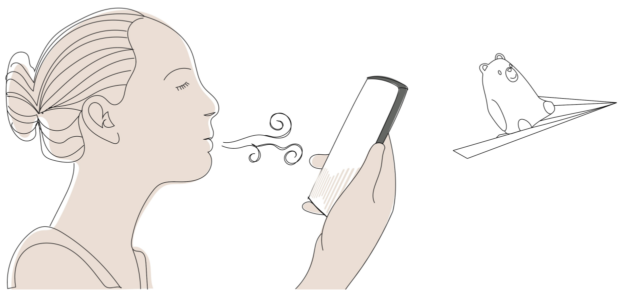


Figure 7.3: To send an auggie, users blow wind at their phone, which triggers an AR paper plane to fly into the distance, carrying a bear over to their partner.

the output. For example, a person may go through a lengthy process of trial-and-error to craft the artifact, which is lost in the final result. Prior work has thus explored several design concepts around revealing effort, centered primarily on text communication, and shown that information about effort can foster reflection and appreciation [132, 133]. To reveal effort in the realm of AR handcrafting, Auggie screen records the sender's "behind-the-scenes" creation process and automatically creates a timeline consisting of four videos (corresponding to outfit, music, animation, and drawing) that document the sender's effort (see Fig. 7.2c). We chose to reveal effort through screen recordings since prior work has shown that replaying creation processes can demonstrate authenticity behind effort (i.e., whether the effort is invested willfully by the sender). That is, recordings can provide details behind effort such as pauses, which can help receivers "think what senders are thinking" and feel closer to them [133]. For privacy, we blur out aspects of the "behind-the-scenes" video that are external to the Auggie app, such as photos in a user's gallery that may appear during the outfit selection process.

7.4.4 Implementation

Auggie is implemented using SwiftUI[4], RealityKit[5], and ARKit[6]. Messages are sent and received through HTTP requests and stored on an AWS[7] server. Each message consists of six components: animation, 3D drawing, images, videos, music, and audio. Animation is encoded using transforms of the auggie bear's skeleton on a frame-to-frame basis. 3D drawing is encoded using positions of spheres generated in each frame. Images, videos, music, and audio are encoded in binary and uploaded to the server. To replay a message on the receiver's

[4]https://developer.apple.com/documentation/swiftui/
[5]https://developer.apple.com/documentation/realitykit/
[6]https://developer.apple.com/documentation/arkit
[7]https://aws.amazon.com/

Table 7.1: Participant Demographics

Pair IDs		**Gender**		**Age**		**Occupations**		**Relationship**
1a	1b	M	M	21	20	Student	Student	Friend
2a	2b	F	F	22	22	Software Engineer	Student	Friend
3a	3b	F	M	25	25	Research Intern	Planning Manager	Significant Other
4a	4b	F	F	33	30	Marketing Manager	Account Director	Friend
5a	5b	M	F	33	28	Product Manager	Property Manager	Sibling
6a	6b	F	M	26	28	Student	Product Manager	Significant Other
7a	7b	F	F	24	24	UX Architect	Student	Friend
8a	8b	F	F	25	25	Student	Data Engineer	Friend
9a	9b	M	M	30	26	Student	Pharmacist	Sibling
10a	10b	F	F	23	24	Student	Bather	Friend
11a	11b	F	F	22	22	UX Lead	Technical Lead	Friend
12a	12b	F	M	20	20	Student	Student	Significant Other
13a	13b	F	F	22	22	Software Engineer	Equity Analyst	Friend
14a	14b	F	F	26	20	Student	Student	Sibling
15a	15b	F	F	25	20	Software Engineer	Student	Sibling

end, the system downloads and decodes each component.

7.5 Methods

We conducted a field study with 30 participants (15 pairs) who used Auggie for two weeks. We interviewed participants at the middle and end of the study in order to understand the details of their experience using Auggie.

7.5.1 Participants

We recruited participants from a technology company and four universities in the United States through Slack posts and e-mails. Each participant was asked to select a partner with whom they are close but not currently cohabiting. We chose this population in order to investigate circumstances in which digital communication plays an essential role. By disallowing cohabiting participants, we focus on participants whose main method of communication is digital and reduce the impact that in-person interactions might have between the pairs. We recruited 50 participants (25 pairs); however, 10 pairs were removed from the study in the first week due to failing to meet the minimum study requirements (e.g., cohabitation status, not showing up to interview sessions, and technical issues such as incompatible iOS versions). This left us 15 pairs of participants in total (n=30). Table 7.1 shows the demographic information of participants. We compensated each participant with $25 Amazon gift cards at the middle and end of the study, for a total of $50 worth of gift cards.

7.5.2 Procedure

Participants first joined an hour-long onboarding session over Google Meet to receive study instructions. Prior to this session, they also completed a short entry survey about their background and relationship with their Auggie partner. During the onboarding, participants were first introduced to the study and were informed of all the data that would be collected during the study. They were also informed that participation was voluntary and that they could drop out at any time without repercussion. All participants signed an informed consent. After obtaining consent, a member of the study team then introduced Auggie's functionalities and guided the participants through the process of creating an auggie. They ensured that participants understood how to use the application and were able to send and receive auggies during the onboarding session.

During the two-week study, the participants were asked to send at least one auggie to their partner daily. This instruction was not a strict requirement, but rather a means to encourage participants to send auggies to each other. All participants were compensated fully regardless of the number of auggies they sent, and were only excluded from our analyses in extreme cases (e.g., only sending one default auggie with no changes). While more freely using the app could potentially reveal how it fits naturally into participants' lifestyles, we encouraged minimal usage to ensure enough engagement during the study to inform our understanding of participants' experiences with the app.

To understand participants' experiences, we asked participants to complete brief daily surveys and two semi-structured interviews during the study. The daily surveys included open-ended questions about the auggies that they sent and received, their perceptions of effort using Auggie, and any technical issues they faced (see supplementary materials). We asked participants to fill out a minimum of three surveys per week. After the first week, the participants completed a 30-60 min semi-structured mid-study interview over Google Meet, which included questions about participants' experiences creating and receiving auggies, their perceptions of effort and AR while using Auggie, and comparisons between Auggie and other communication tools that they used (see supplementary material). The interviews were also used to clarify and follow-up on responses to their daily surveys for more detail. At the conclusion of the study, participants were offboarded and completed an exit interview and survey, which were similar to the mid-study interview (with additional questions about any changes in their usage and perceptions of Auggie) and entry survey, respectively. Throughout the study, we also recorded each user interaction in Auggie with the corresponding user id, timestamp, and interaction event to track their overall usage of Auggie. Our study proposal and protocol were reviewed internally by a panel consisting of independent privacy engineering and legal teams to ensure that the subject matter and approach complied with ethical standards and that participants' data was processed appropriately.

7.5.3 Data Analysis

We analyzed users' behavioral data and the transcriptions of the mid-study and exit interviews. For the behavioral data, we parsed and cleaned the system event data and the message content from the server, forming descriptive results of the usage of Auggie. For the qualitative analysis of the interview data, we took a grounded theory approach [243]. Two researchers first performed open-coding on three randomly selected participants' transcripts, identifying and labeling similarities across participants' experiences with Auggie in order to develop a codebook. Three researchers then refined the codebook on another subset of participant transcripts, individually coding the same transcripts. The researchers validated the codebook on two additional participants' transcripts, meeting frequently to discuss disagreements. After achieving high inter-rater reliability, with a Krippendorff's α above 0.8, they divided and coded the rest of the transcripts in parallel. Finally, we performed axial coding to group related codes into themes according to our research questions.

7.6 Results

Participants' experiences using Auggie during the study suggest that they indeed engaged in effort while crafting animated AR experiences for their partner. As Table 7.2 shows, participants sent an average of 11.2 auggies, with animations as the most frequently used component (10/11 auggies on average), but the number of auggies sent varied widely across participants. Based on the interviews, this was partly due to different perceptions of Auggie usage scenarios, where some participants perceived everyday usage while others preferred it for special occassions. For instance, P13b, who sent 25 auggies in total, described Auggie as "similar to Messenger or Snapchat where we're just sending little bits of our updates every day." On the other hand, P11b sent 3 auggies in total, and described using it for special occasions "like a birthday or an anniversary," rather than daily, viewing Auggie as "mostly about creating something new" and not "sharing anything about [themselves]."

Additionally, in line with our design motivations, we found that participants perceived how much effort their partners put into crafting through "behind-the-scenes" videos. As one participant pointed out, with this feature, "you know where this person is...you could kind

Table 7.2: Auggie Usage Data

Pair IDs	**# of Auggies**	**# of Animation**	**# of Voicenote**	**# of Drawing**	**# of Outfit***	**# of Music**
1a, 1b	8, 8	8, 7	8, 5	7, 0	7, 5	5, 3
2a, 2b	8, 6	8, 6	4, 6	1, 0	0, 0	0, 0
3a, 3b	16, 17	11, 8	11, 17	5, 2	2, 11	0, 1
4a, 4b	12, 8	11, 8	9, 5	8, 3	3, 1	7, 5
5a, 5b	8, 11	6, 10	6, 4	1, 3	1, 4	0, 0
6a, 6b	6, 8	6, 8	6, 4	3, 2	4, 3	1, 1
7a, 7b	7, 9	7, 8	3, 9	0, 3	2, 1	1, 0
8a, 8b	14, 13	14, 11	5, 10	6, 5	2, 5	0, 2
9a, 9b	13, 7	9, 7	8, 1	4, 3	2, 0	0, 0
10a, 10b	17, 15	17, 12	12, 15	17, 14	16, 13	3, 2
11a, 11b	17, 3	15, 3	13, 3	4, 1	2, 1	6, 0
12a, 12b	11, 16	10, 16	1, 0	8, 16	4, 7	6, 4
13a, 13b	21, 25	20, 25	5, 9	5, 6	3, 0	5, 1
14a, 14b	8, 8	6, 4	0, 2	4, 5	2, 4	3, 3
15a, 15b	7, 10	7, 10	7, 4	2, 9	4, 9	1, 3
Mean	11.2	9.9	6.4	4.9	3.9	2.1
SD**	5.0	4.8	4.3	4.4	3.9	2.2

* Outfit: t-shirt image.

** SD: standard deviation.

of see if they put effort into making these auggies or if they didn't" (P6b).

In the following subsections, we provide more detailed insights into the participants' usage and perceptions of Auggie. We center our findings around two major themes: 1) how digital handcrafting encouraged people to engage in effortful communication, and 2) the effects of handcrafting in the digital space.

7.6.1 Encouraging Effort through Handcrafted Digital Experiences

Overall, our results suggest that the design of Auggie encouraged effort by promoting a sense of agency, creating the feeling of gift-giving, and enabling digital storytelling experiences through the process of crafting auggies. Participants felt they had control over the creation process, which encouraged them to invest effort into crafting unique personal experiences for their partner as if it was a gift. Participants also highlighted the ability to craft stories that allowed them to express themselves beyond the possibilities of the physical world.

7.6.1.1 Agency over handcrafting encouraged effort

Agency refers to the awareness and control over one's action [272]. Participants reported having agency over crafting their auggies. The design of Auggie provided participants with a variety of crafting options such as animation to orchestrate a story, 3D drawing to embellish the environment, music to set the mood, and voice note to express themselves. These elements together provided wide control over what the final experience would look like and further enabled users to craft experiences that feel personal:

> "It's a lot more personal compared to all the other apps that are out there, especially because you control the bear and because you're able to modify all of its functions and all the other features. It's basically, *you* create it." -P2b

In addition, participants' agency over auggies evoked a sense of accomplishment and ownership. More specifically, participants felt proud of themselves after investing effort and using creativity in the crafting process, especially given that many of them did not have artistic backgrounds. Through this process, they could feel like *creators*, and the experiences they created belonged to them:

> "I'm not an artist at heart or anything like that, so drawing a 3D flower and having it visibly come off as that was really interesting to see. I was really proud of that to the point I took screenshots myself of it when I finished because I was really proud of it." -P10a

"I think it's just that you have this idea in mind and you pretty much just bring it into a world by either making the bear do something or by drawing it. It really brings your thoughts into the real world. I think that that's also a very big component and what makes it meaningful is that you created it, so really, it is yours." -P12b

With the benefits of agency, the digital handcrafting process became meaningful to participants, and sparked joy in the process of creating for someone else. As such, participants became more willing to invest effort into creation:

"I think the most meaningful part was when I saw it come together as it was loading in for me to send to my partner. That was the part where I realized that I had created every single bit of that message...the drawing and the shirt and that storyline, I created without my own knowledge. That was really meaningful to me because I'm not outwardly a creative person. I found that in this app, I was able to explore those features a little more and find some joy in what I created, even if it wasn't perfect." -P10a

7.6.1.2 Treating the craft as a gift encouraged effort

Many participants treated a crafted auggie like a gift or a hand-drawn or handwritten card. In particular, participants drew from the agency they had over handcrafting to treat the craft like a gift. Since they were given several crafting options, they described the process as similar to spending effort in preparing a gift or a gifting a card to their partner.

"I feel like it's like you want to make a gift. Auggie is like you're trying to create something. Because the bear is like a small board, and you're trying to add image, motions, some actions, musics, voice message, so it's like you're trying to create something... It's like when you create something, you will take more effort to do that... to send something that's very caring for them." -P8b

Participants also described the process of sending their auggies to their partner as a special experience, where blowing on the AR airplane felt like taking a "moment" of care:

"I think of when you're sending it off there's a really the intentional blow into your microphone. You send off a little bear, a little airplane. It's like, 'Oh yes, it's come all the way to me.' It's a lot different than this is a text message it's going to loop like right across the wireless networks and everything like that. You got a little considerate moment. That's like, 'Oh yes, I'm sending off the little thing. Goodbye."' -P12a

Participant's perceptions of auggies as a gift were further reinforced by their experiences receiving auggies from their partner. Participants reported feeling a sense of anticipation and gratitude when receiving an auggie, similar to feelings they would have when receiving a gift or card in real life. For example, one participant talked about the excitement of not knowing what the auggie would bring, comparing the process of viewing it to unwrapping a gift:

> "...really just seeing the little story unfold. My partner doesn't talk to me before I view it to say, this is what I said or anything. Obviously, it's a surprise when [I watch] it. This also goes into gift-giving, it's almost like unwrapping a present. It's a little bit of suspense and seeing like, oh, what's going to happen here?" -P12b

Participants also described feeling of gratitude when receiving auggies, similar to receiving a gift, due to the effort they perceived was involved in crafting it:

> "I feel like there's a lot of similarities [between an Auggie and a real-life gift]. I feel like one of the biggest ones is whenever I receive a gift, I feel like a little bit grateful, that bit of gratitude because you think to yourself like, 'Oh, man.' The person who's giving you a gift or whatever, they've put in a lot of work into it." -P12a

Finally, prior work has found that the motivations of gift-giving include not only social norms such as birthdays, but also altruism (e.g. showing comfort) [275]. Aligned with this work, we found that participants felt encouraged to invest effort into crafting auggies like gifts for not only planned occasions, such as birthdays, but also day-to-day occasions, such as providing emotional support to their partner:

> "As well as it allows for a fun personalized message, which could be used in that gifting aspect because you could use this incentives to create a personalized Happy Birthday to someone that you might not see all the time or be able to express some emotion a little more directly and more personable than via social media." -P10a

> "Both of us are actually sick at the moment so she stayed home the other day, so that's when I sent that [auggie]. I know that that had made her laugh, to see it jumping around and moving so quickly. I know that it was nice for her to hear that I missed her even though she was sick." -P10b

7.6.1.3 Crafting stories beyond physical limits encouraged effort

Participants were also motivated to invest effort due to the ability to create stories beyond their physical limits. As mentioned in the previous section, the design of Auggie provides a set of crafting tools that opens up creative possibilities. Some participants saw the virtual bear character as a representation of themselves that they could equip with new abilities or actions that would be physically difficult for them to achieve in real life. With the new capabilities, participants were encouraged to tell imaginary or past stories. For example, one participant described an auggie that they created for their friend, where they animated the bear to go out into the rain when they could not:

> "...my friend just moved from [my city] to another city and it is raining. I [made the] auggie, I [recorded the environment] outside of my window and I made the bear walk around the grass and I [recorded] a video and sent it... I can't go out there if it is raining but the bear can... If I want to jump [on the table], I can't jump on the table but the bear can jump on the table." -P8a

In addition to physically difficult experiences, participants felt encouraged to invest effort since they could express themselves in new ways that they might not feel comfortable doing in real life. For example, one participant created an auggie in which the bear banged its head against a wall to show that they were cringing after a meeting:

> "It could be comic relief, and it's easier to convey the head banging against a wall with the bear instead of me doing that because it will look strange if I show a video of myself doing that." -P2b

Lastly, by blending virtual content with the physical environment, AR offered a sense of embodiment and situatedness, such that participants felt that the virtual content actually exists within the physical world and could interact with the environment in an unconventional way. Such interactions triggered serendipitous experiences that motivated them to invest effort in order to tell a unique story:

> "I went to the kitchen, and we [have a] cast iron, so we don't move it very often. It was just sitting on the stove, and I pointed the camera at it, it takes a second, and the [bear] rendered, and it just rendered it directly standing in the pan and I started laughing. I was like, 'I'm just–great, I'm going to send this, this is hilarious, it's just standing in my pan.' Yes, so, I drew a heart, I was laughing, I was like, I swear I'm not cooking the auggie...was like my audio note that went with it. It was just funny. Again, it was that real interaction with the environment I think is what made it special." -P5a

This potential for unexpected and physically impossible interactions with the physical environment (e.g., a bear standing on a pan) can trigger users to invest more effort into creating a whole story and providing context around that interaction, such as adding 3D drawing and voicenotes while expressing their surprise and joy.

7.6.1.4 Limits of expressiveness and creativity support discouraged effort.

While the design of Auggie encouraged effort through its gift-like and storytelling crafting process, some participants reported their unwillingness to invest effort with Auggie due to limitations in its expressiveness for certain contexts. For example, one participant commented that Auggie is not suitable for serious situations:

> "My friend was going through some family stuff the last two weeks. Her dad was in the hospital, so that's more of a serious and emotional topic. It didn't feel very personal or sensitive to send my communications about that with her through Auggie, mainly probably because it feels, although Auggie's an extension of me, its like coming from an AR bear versus me personally" -P4a

In this case, crafting with our tool could be inappropriate and undesirable to users, where a different level of personal communication may be necessary. In addition, our system can only send the virtual bear character in AR, excluding any physical objects from the sender's environment. This design discouraged some participants from investing effort into crafting as they were unable to accomplish the expressions they wanted with their surrounding objects, which they felt were essential to creating personal stories. For example, one participant described her frustration with the inability for the bear to interact with their household objects or her cat:

> "I would like [to] dance the little bear around my wineglass or have the bear petting my cat but the cat or the wineglass doesn't really go to her [experience]. I'm having the bear do these funny things, but she's not seeing it in my environment." -P4b

Finally, several participants faced eventual creativity blocks in the crafting process. This discouraged effort as participants did not know what to create. For example, one participant reported having a difficult time coming up with ideas for an auggie, heightened by their static physical environment over the course of the study:

> "I think my problem was I ran out of ideas and I've also ran out of new scenery...I think if I had new scenery, it'd give me more ideas how to use it, but

I'm mostly just sitting in front of my desk every single day so as a result, my auggies, I just haven't gotten any new inspiration." -P1b

7.6.2 The effects of digital handcrafting

We also sought to understand how the effort encouraged through digital handcrafting might affect participants' perceptions of the experience (both as a sender and receiver) and connections with their partner. We found that the design of Auggie had the potential to influence more authentic social connection between partners that felt effortful yet lightweight. Participants also reported feelings of social support and presence through handcrafted digital experiences. We describe these in detail below.

7.6.2.1 Personal and authentic connection.

Our results suggest that effort through digital handcrafting can connect people in more personal and authentic ways, specifically by reflecting the creator's personal expressions. In Auggie, the bear's animations and recorded audio felt especially personal, because the creator's postural and auditory characteristics were embodied in the bear. As previously mentioned, participants often viewed the bear as representative of themselves; thus, they could animate it to express their internal feelings or simulate their actions. One participant compared this process to existing social platforms:

> "There is a lot of toxicity found in social media, and this false portrayal of people's lives sometimes. With Auggie, I think that removes those from the equation, and actually allows you to just be yourself by creating what you want to or creating something for your partner that will be meaningful, without feeling the pressure of society to do something a certain way or make it look a certain way or anything like that. You can just purely have fun with it and enjoy your time using it without those pressures present.... No auggie will be like another auggie. Whereas there are Snapchat filters where it will make you look like someone that you've never met. With this, you really get to just be yourself and have fun with it." -P10a

Some also described Auggie's potential for inside jokes and comic relief (P2a, P12a), enabling expressions that are comfortably authentic representations of themselves. For example, one participant created an animation of the bear leaving work and walking out of the lab, as if it was themselves:

> “The benefit that I really like is the movement of auggies, which is a personified version of you. It shows your partner or whoever you’re sending it to what you’re doing. It already feels a little bit more personal [than other communication tools].” -P2b

7.6.2.2 Lightweight connection

We found that participants viewed their communication through digital handcrafting as low-stress and low-urgency due to its resemblance to a letter, greeting card, or postcard (as described in section 7.6.1.2). Similar to receiving letters and cards in real life, receiving an auggie is exciting and meaningful, but does not require an immediate response through Auggie or other communication means (e.g., text, phone call). Aligned with this, 70% of participants reported in their surveys that they did not expect an immediate response from their partner for the auggies they sent. Therefore, despite the effort necessary to create auggies, the asynchronicity of the handcrafting experience made it feel *lightweight*, an informal and playful way to convey something interesting or spontaneously show “I’m thinking of you.”

> “I guess it’s just the fact that I’m thinking about a lot throughout the day and they’re very small, not very important thoughts, they’re random thoughts or information that I read about online. Just these very small things that maybe are not important enough to save for later to discuss or stuff can be shown through this Auggie since they’re a really quick and low commitment.” -P11a

This shows that the feeling of lightweight and low commitment resulted in a casual way of using our system, similar to a conversation starter. This approach demonstrates further resemblance to how a postcard is used in communication: effortful in the creation and sending process yet lightweight enough to serve as a catalyst to deeper conversations. For P11a, they were able to express random thoughts such as a song that got stuck in their head or a meme they saw online that they wanted to share with their partner, using tools such as Auggie’s music and customizable outfit features. While participants could have expressed these ideas through other communication means, such as messaging apps, P3b commented on the value of crafting them through Auggie:

> “[Auggie] may be more like a postcard. [It’s] not quite an instant message. When you sign it, your partner may check it after a while and your message in Auggie is quite vivid, not like other instant communicate [sic] apps like WeChat or Facebook or something.” -P3b

This aspect of Auggie's design consequently lead some participants to feel closer to each other at the end of the study. For example, one participant expressed a greater appreciation between her and her partner because of these lightweight "conversation starters" (P2b):

> "We've been getting closer... I definitely felt like she appreciates our time more or our interactions more and I do too... I think it does add something to our relationship in an interesting sense. We've gotten to talk a lot more because of it." -P2b

7.6.2.3 Feelings of presence

Participants felt that the situatedness of handcrafting in AR and inclusion of the partner bear character helped enable a sense of presence. Similar to how participants viewed the bear as a self-representation, participants viewed the partner bear as a representation for their partner. By superimposing the virtual representations of participants into their partner's physical world, the design of Auggie helped create a feeling of "being there":

> "To me, it's the idea of putting some avatar of yourself for the recipient in the scene. You know a good analogy is when you want to be with your friends but one of them is missing and you take a group photo and you hold up your hand and then you're like, 'You should be here, your head should be where my hand is.' It's like that. To me, it's sort of that idea. It's like some sense of you should be here, you would enjoy this or like – yes, that's really what it resonates as for me... you make them dance or move around together, or you try to make them hug or something... that's where it gives me that connection of like my sister and I were sitting together in person somewhere...." -P5a

This highlights how the bear character could be used as a proxy for the recipient, enabling creators to incorporate their partners in the stories they were creating. This included scenarios where the bear representations of both people could interact with each together, thus contributing to feelings of togetherness. At the same time, this feeling of presence can have an emotional and social impact, which helped participants feel closer:

> "One thing [about Auggie] was since we stay far apart, it is easier to connect. I feel like I'm right there with her, so it's kind of give me the warmth or– When I miss her, it's like I'm right there with her. It's kind of an intimate or warm feeling that I get from that. That is what I feel like." -P14b

7.7 Discussion

Our results demonstrate that the design of Auggie offers a sense of handcraftedness that encourages digital effort, and subsequently has the potential to encourage feelings of personal connection and presence. At the same time, our findings suggest that effortful communication should be carefully designed, as effort in the wrong contexts could lead it to become meaningless and undesirable.

As prior research has pointed out, effort is widely understood as a desirable concept in interpersonal relationships, as it is interpreted as a sign of mutual affection and care [29]. However, *personal* effort invested in communicating within these relationships is frequently entangled with the concept of *procedural* effort, referring to the work required to operate an interactive system [169]. This nuance can be lost in the design of communication systems, where *any* effort is systematically perceived as difficult to use, negative, and thus minimized in a system [76]. Message Builder, for example, involves both personal and procedural effort in accessing and writing lengthy text messages, but some users perceived the system as demanding and inconvenient [134].

In comparison, our approach attempts to provoke effort in a digital context by making the highly *personal* effort task of handcrafting less *procedurally* effortful. Our findings showed that the experience of using Auggie was indeed perceived as effortful on a personal level, and, despite investments in time and experimentation, still described as lightweight and approachable on a procedural level. The results suggest that designers can leverage playfulness, flexibility, and asynchronicity to integrate effort into communication for greater social connection.

Based on the concept of procedural and personal effort, we highlight the opportunities and challenges of effortful communication over technology, and discuss design recommendations drawn from our results. We focus our discussions on these two types of effort within social communication systems: on the *procedural* side, making effort approachable by reducing the barrier of interacting with the system; and on the personal side, making effort meaningful through creating unique and personal content and expressions for communication.

7.7.1 Designing for approachable effort

While prior work suggests that procedural effort can be valuable when conveying personal commitment [132], excessive procedural effort can have significant impact on a user's motivation to interact with a system [134, 231]. Based on our findings, we discuss implications for design that can make effort more *approachable* to people, focused on reducing procedural barriers associated with interacting with a digital communication system.

7.7.1.1 Masking the heaviness of effort with playfulness

Excessive procedural effort can have an innate sense of heaviness or undesirability [169]. We found that the design of Auggie's effortful experience, including playful animations and engaging visuals, helped make the procedural aspects of crafting (i.e., the step-by-step of putting the craft together) fun and enjoyable. This demonstrates how playful design can distract people from the potential tedium of required effort with a joyful, more lightweight experience. This aligns with game design research, where designers who aim to change perceptions or teach about conventionally "heavy" or "boring" topics, like unconscious biases or cybersecurity, face resistance from users. To solve this, researchers have created interventions that use playful content, such as entertaining storylines or cute imagery, to distract from the heaviness of the topic [129, 43]. We suggest that future researchers and practitioners exploring effortful communication integrate playfulness alongside effort to reduce the barriers of undesirable procedural work. For example, future effortful communication systems might employ similar storylines to generate or deliver the content being communicated (e.g., through whimisical characters or objects), or utilize "gamification" attributes such as stickers or reward systems based on effort. At the same time, we highlight the importance of carefully designing such playful features so as not to take away from intrinsic, personal motivations behind effort.

7.7.1.2 Reducing pressure with asynchronicity.

The asynchronous design of Auggie helped reduce the pressure to communicate on both the sender's and receiver's side. As suggested by prior work [134], asynchronicity can ensure that a sender has sufficient time to invest effort in their communication. Compared to a more synchronous exchange, such as a back-and-forth conversation, participants could take the time to carefully craft an experience for their partner. Similarly, on the receiver's end, participants naturally related receiving auggies to receiving a greeting card, i.e., something that one can treasure and keep but not necessarily respond to immediately. Consequently, the crafted content can become a precious moment to be treasured, rather than hogging on one's mental capacity. Thus, future researchers and practitioners might consider optimizing for asynchronicity, where people have time to craft and reflect on the content of their communication. For example, this could include an asynchronous co-creation process, where people can collaboratively craft content based on their own time and reflect on each others' additions to the craft, or virtual "mailbox" spaces that people can join at any time and create content to leave behind for others.

7.7.2 Creating meaning in effort

In this section, we highlight design implications for heightening meaningful *personal* effort in communication, which, as our results suggest, has the potential to facilitate feelings of connection and presence. In particular, we focus on characteristics that can help frame communication as a *gifting experience*. From our study, participants described finding auggies meaningfully effortful because they felt like gifts. Interestingly, this is distinct from prior work suggesting that digital gifts lack specific elements that make physical gifts more meaningful in comparison [148]. Building on this work, our results suggest that the design of Auggie incorporated key attributes and rituals of gifting that can build meaningfulness, such as personalization, "physical" wrapping and exchange, and visible effort [148]. We detail three potential design directions along these dimensions.

7.7.2.1 Providing channels for personalization

Spence suggests that efforts to personalize a gift and create shared memories can provide a sense of *inalienability*, or the sense that the "spirit" of the gift giver remains with the gift even after it's sent, making it unique from simply shared content in communication [235]. Our study suggests that a variety of creation channels that act as a scaffold to guide effort [134] can help to achieve personalization. The variety of channels allows the participants to create personalized communication content for the recipient that feel less like a "message" and more like a story through combinations of different elements, such as characters, animations to represent actions, drawings for props, and background music for mood.

We encourage future researchers and designers to consider a range of effortful "building blocks," where people can invest effort into assembling content that is unique and personal for someone else. For example, in the context of social media, Snapchat and Instagram emulate this by providing the function for the users to add in stickers, text, sound, and filters to the photos they share. For text messaging applications, this personalization could happen in not only the content of the message, such as through kinetic typography or font options, but also the application background, message bubbles, and sticker options, and keyboards [99, 100]. Going beyond communication applications that use 2D content (e.g. images and texts), future work can also explore possible channels of personalizing 3D content (e.g. 3D scenes and characters) for others, such as customizing facial and postural expressions of virtual characters or customizing elements of the virtual environment like the background or surrounding objects.

7.7.2.2 Bridging physicality

Prior work suggests that physicality can bring meaning to gifting, particularly in the rituals of wrapping and co-located exchange [148, 140]. While Auggie did not involve physical gift wrapping or co-location, we found that the situatedness and physical interactions incorporated in the design helped convey a sense of physicality. For example, our results suggest the *process* of sending and receiving the content acted as a "hybrid wrapping," or the wrapping of digital gifts in physical material and vice versa [140]. Senders took a "considerate moment" in blowing wind to send their auggie off on an airplane, and receivers landing the airplane and opening the auggie felt like they were unwrapping a gift. Future work might consider designing similar physical interactions to emulate this effortful wrapping process, such as body gestures (e.g., tossing the content into the air) or facial expressions (e.g., opening it with with a smile).

For the content itself, auggies were played in the receiver's environment, but also included the sender's environment as part of the content through the "behind-the-scenes" recordings, which participants enjoyed viewing for additional context (such as in the case of P5a showing the bear on a frying pan). Since the physical environment is fluid and unique for every user, the ability to incorporate any surrounding physical object could be beneficial for encouraging people to craft more personalized and creative experiences that embed aspects of their environment. At the same time, nudging senders to consider crafting the content according to the receiver's environment (e.g., knowing that the receiver usually sits at their desk during the day and would view the content there) could demonstrate more personal effort through the quality of accounting for the receiver's context.

7.7.2.3 Explicitly revealing effort

Kwon et al. suggest that people may not value digital gifts when the effort behind them is not evident [148]. Even if a lot of effort was invested, people may not recognize this effort when only the final content is visible, rather than the holistic experience of creating it. Aligned with this, our results suggest that explicitly visualizing effort (e.g., through a "behind-the-scenes" screen recording) as opposed to expecting recipients to implicitly recognize effort (e.g., through knowledge about how much effort a system involves) can strengthen the meaningfulness of the communication content and encourage reciprocation.

Researchers and practitioners should thus consider ways to explicitly reveal sender and receiver efforts involved in communication. For example, platforms like Honk[8] use live messaging to make the thoughtful composition of a message and revision process more explicit.

[8]https://honk.me

Kelly et al. proposed a concept called Craft Box that uses a video replay of the sender's text composition process to reveal their interactions in the message creation [132]. Similarly, Auggie uses a screen recording to show the exact creation process, given the variety of visual elements involved. While making it easier to understand "authentic" effort, such full recordings may not be desirable for lengthier or more complex effortful processes, as they may introduce "higher costs" for recipients to interpret and acknowledge the effort [133]. Senders may also have as privacy concerns with recordings, as they might not feel comfortable exposing the whole creation context to recipients. A less invasive approach could involve aggregate statistics that quantify the effort process, such as the time spent, the number of interactions (i.e., procedural tasks), the number of revisions, and so on to summarize the effort invested. Visualizing traces of effort, such as strokes indicating gestures used on an interface, could provide a more abstract view into the content creation process, while also potentially providing a sense of presence [175].

7.7.3 Limitations

Our results demonstrate that Auggie encourages effortful handcrafting of digital artifacts, which subsequently has the potential to support meaningful interactions with close others. However, our work is not without limitations.

First, while deploying Auggie in participants' everyday lives enabled evaluating the system *in situ*, the nature of a field study limits our ability to control for external variables, such as other communication channels or effortful practices the participants might have engaged in during the study. This can subsequently affect our ability to isolate the specific effects of our system. Future work should consider more controlled experiments or the inclusion of comparison groups to better measure the impact of effort and digital handcrafting on social connection and interpersonal relationships.

Second, requesting participants to create at least one auggie a day allowed us to understand the perceptions of the system under frequent usage. However, this requirement may have limited the scenarios in which effort was meaningful, as participants noted that this frequency did not necessarily align with how they believed they would normally use the system. This may explain the "creativity blocks" discussed in Section 7.6.1.4 that some participants experienced during the crafting process, where participants described their everyday environment as unchanging. Future work should explore how the regularity of such interactions and greater variety of social situations might affect people's perception and desire to engage in effort, such as through longer studies across several months. Additionally, while we did not observe specific novelty effects during the study, it is possible that it played a role, and a longer study would be beneficial to understand how people would use systems like Auggie in steady state.

Finally, participants primarily participated with close study partners (i.e., friends, family, significant others). While we still gleaned valuable and diverse insights from their usage of Auggie, participants in different relationship types may have different perceptions of the application. For example, early stage romantic couples may inherently feel motivated to invest more effort more frequently, as part of strengthening their relationship. Future researchers should thus consider investigating the usage and value of effortful communication for different relationship types.

7.8 Conclusion

We introduced Auggie, an iOS app for engaging in effortful communication through digitally handcrafted AR experiences. Through a two-week long field deployment with 30 participants (15 pairs), we explored how to encourage meaningful, personal effort in digital communication systems, and its subsequent effects on relationships. Our research revealed that the design of Auggie inspired participants to invest effort in crafting personalized experiences by evoking a sense of agency that enabled them to craft gift-like personal stories beyond physical limits. These immersive, digitally handcrafted experiences subsequently had the potential to enable authentic, lightweight connection and feelings of presence between partners. At the same time, we found that limited expressiveness and creativity support for different situations discouraged people in engaging in effortful communication. We yield insights and recommendations that provide a starting point for researchers and practitioners to further explore meaningful personal effort in digital communication.

CHAPTER 8

Jigsaw: Authoring Immersive Storytelling Experiences with Augmented Reality and Internet of Things[1]

8.1 Introduction

As technology develops, the media used for storytelling evolves too, from traditional methods, such as printed books and spoken tales, to digital formats like photos, videos, and animations. A key goal in these developments is to make stories more immersive and engaging. AR opens new opportunities for immersive storytelling by showing 3D content in a spatial manner and allowing people to interact directly with it. However, AR's immersiveness is typically limited to pixels on a display. Beyond visuals, storytellers in theatres and performing venues have explored making experiences more immersive by adding elements that stimulate multiple senses, including sight, sound, touch, and smell. This approach appears in, for example, "4D" movies at immersive movie theaters and amusement parks. Also, plays like "Sleep No More" enhance engagement by allowing audiences to move around the performance space and connect with actors, making the experience more interactive. These immersive experiences, though captivating, often require specialized setups limited by cost, skills, and locations.

We introduce Jigsaw, a novel system that democratizes creating and enjoying immersive, multi-sensory stories. Jigsaw uses common mobile AR devices—like iPhones or Android phones–and household IoT devices— such as smart lights and fans. This setup allows Jigsaw to display virtual 3D content and mimic environmental effects, like wind and light changes. Unlike previous systems, Jigsaw does not require specialized IoT devices or advanced programming skills [90]. Jigsaw lets people author stories with multiple scenes. Drawing inspiration from slide-based presentation software like PowerPoint, people can add AR elements, such as 3D models and animations, and control IoT devices, like dimming

[1]Portions of this chapter were adapted from [284]

lights or adjusting fan speed. They can set triggers, such as a spoken word or touch, to move to the next scene. Additionally, multiple people can experience the story, each from their own unique perspective.

We evaluated Jigsaw with 20 participants in 10 groups, with each group creating three specific stories. We found that Jisgsaw's stories were immersive, engaging, and memorable. The authoring mode was user-friendly and versatile for creating various stories. Our study also highlighted some trade-offs of this new storytelling modality, including the tension between engagement and sensory overload.

This work presents three key contributions: 1) an AR + IoT tool that makes it easier to create immersive stories; 2) three immersive stories that showcase the possibilities of this approach; 3) findings from a study that highlight the advantages and difficulties of this innovative design approach.

8.2 Related Work

8.2.1 Immersive Storytelling

Immersion is an important aspect in any storytelling to engage the audience and to convey the emotion of the performance. Therefore, many fields of performing arts have adopted novel forms of performance to enhance immersion. Immersive theater is a prominent example "which use installations and expansive environments, which have mobile audiences, and which invite audience participation" [271]. It can involve great levels of intimacy [92, 93, 94] or put more distinction between the audience and the actors [211, 212]. The experience also involves multiple spatial awareness, touch, and smell [271]. Sleep No More, an immersive theater performance based on Shakespeare's Macbeth, enables the audience to be embedded within the performance, inviting them to walk around the stage and touch the props [115]. While there are some examples of technology used in this type of performance, e.g., headphones and speakers in Rotozaza's Autoteatro [107], it is rather limited, with many available technologies like projection mapping, AR, and robotics rarely being used if ever. We envision that technology has a great potential to enhance immersion and give a sense of magical events occurring in the world around us, as seen in literary genres like magical realism [179, 8].

Outside of theater, we see more frequent use of technology in storytelling. For instance, the Weather Channel frequently uses AR in their weather forecast programs [77]. Additionally, theme parks, such as Disneyland, use projection mapping, robots, and mechanically moving rides to foster a high sense of immersion in the storytelling of their rides [196, 197]. Projection mapping, in particular, is a popular method for AR to convert a room or a building into a magical space filled with eye-catching visualizations (e.g., [256, 96, 227]). However, these storytelling experiences require a team of expert designers and engineers to configure, along with an expensive set of equipment. Therefore, they are difficult to access for most people and can only be experienced in special venues. In contrast, our work aims to democratize immersive storytelling to everyday users.

The emergence of AR-enabled mobile devices created a platform to tackle this issue. Wonderscope [122] is a prime example that uses mobile AR to recreate immersive theaters for children in their own rooms. Using Wonderscope, children can be immersed in the magical storytelling experience, presented as ready-made narratives. SceneAR achieves a similar immersive storytelling experience but adds the ability to remix the stories [42]. There are also examples of AR use for educational purposes, such as teaching chemistry [58]. While these systems provide a starting point and an inspiration for our work, they are largely limited to augmenting the world with pixels on a screen. This fundamentally limits the level of immersion compared to previous examples of immersive theater and higher-fidelity experiences that give a sense of magic throughout the entire surroundings. In this work, we want to incorporate both virtual and physical augmentations ("pixels and atoms") to foster a more complete immersion. Therefore, we examine the current landscape of ubiquitous computing and the possibility of using smart IoT devices as a latent infrastructure of immersive storytelling.

8.2.2 Ubiquitous Computing and Physical Augmentation

With the growing popularity of smart homes and IoT devices, ubiquitous computing is becoming a part of everyone's day-to-day life. Part of this effort aims to bring virtual augmentation to the world around us. For instance, RoomAlive [127] and Room2Room [202] achieve room-scale projection mapping with commercially available devices such as consumer-grade projectors and Microsoft Kinect. Moreover, Olwal and Dementyev [192] showcase an ambient display hidden under common surfaces that can be seen in household furniture. As for use cases that focus on storytelling, Healey et al. [112] demonstrate that mixed-reality systems can enable remote storytelling and that it helps promote engagement among children.

However, as discussed above, we wish to go beyond pixel-based augmentation and leverage devices that physically augment the environment. Examples include smart lights, smart fans, speakers, smart thermostats, vacuum robots, and wearable devices like smart watches or smart glasses. Additionally, we want to seamlessly integrate the virtual and physical

Table 8.1: Design considerations for each immersive story.

Story	IoT Augmentation	User Role	Interaction
Benjamin Franklin	Background	Narrators and actors	N, T*, M
The Wind and The Sun	Foreground	Audience	T**
Goodnight Moon	Foreground	Narrators and audience	N

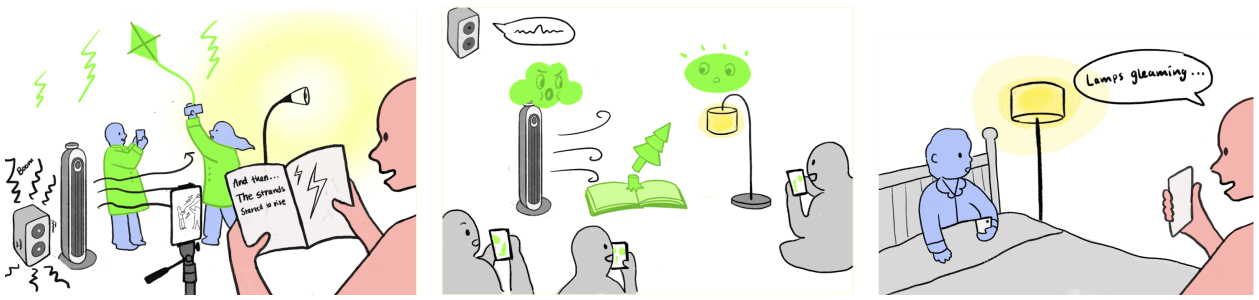
N: Narration
M: Moving in the space
T*: Touching
T**: Tapping on the phone

augmentations to create a holistic narrative. There are previous works that leverage conversational agents on smart speakers [52, 160], robots [118, 142], and tangible interfaces [251] to foster effective storytelling with physical augmentation, but they largely focus on one or a handful of IoT device types, and often do not incorporate virtual augmentation. Lin et al. [160] showcase a more comprehensive, low-cost system that recreates 4D theater experience using home appliances like the ones mentioned before. However, the system nonetheless does not incorporate immersive virtual augmentations like mobile AR or AR glasses, and the system's authoring environment is not suited for novice users, which fails to meet our goal of enabling anyone to create immersive experiences at home. StoryMakAR [90] enables children to create an immersive storytelling experience using robots and mobile AR, which aims to achieve similar goals as our work but still does not incorporate other physical devices like fans, lights, thermostats, etc., to create a truly immersive experience that transforms the entire surroundings. We examine existing tools with similar goals to envision how to empower anyone to author this type of experience.

8.3 Jigsaw System

We designed Jigsaw to enable novices to create and consume immersive stories using a mix of mobile AR and off-the-shelf IoT devices. Our research process was divided into two steps: exploring *what* immersive stories are, and investigating *how* to create these immersive stories. We executed our research in this order since to understand what the possible resulting experiences in this paradigm are before building an authoring system for creating these experiences.

For the first step, we used two motivating questions to guide our design of immersive storytelling experiences: "how to seamlessly integrate both virtual and physical augmentation to these immersive stories?" and "what are the levels of participation for users in interacting



(a) Benjamin Franklin. (b) The Wind and the Sun. (c) Goodnight Moon.

Figure 8.1: Three immersive stories designed via a mix of mobile AR and IoT devices. (a) The Benjamin Franklin story leverages IoT as background augmentation and enables users to actively participate as different roles in the story; (b) The Wind and the Sun story uses IoT as foreground augmentation and pictures users as the audience who watch the story proceeds; (c) The Goodnight Moon story uses IoT as foreground elements in the story, each of which responds to users' greeting.

with each other and with the stories?" We identified three design considerations for immersive stories that involve AR and IoT devices. Based on these design considerations, we then conducted a series of brainstorming sessions to discuss potential stories via sketches and low-fidelity prototypes. Lastly, we prototyped three representative stories (i.e., Benjamin Franklin, The Wind and the Sun, and Goodnight Moon) that cover different aspects of the design considerations. The design considerations for each story are specified in Table 8.1. During the prototyping process, the research team met regularly to discuss design iterations and ran pilot tests frequently with end-users outside of our team using the prototypes.

For the second step, we designed and built an authoring system for creating these immersive stories. We had three design goals in mind: *(i)* to lower the barrier of entry for novices to create these stories, *(ii)* to make the system generalizable for creating various stories including the three representative stories, and *(iii)* to make the system consistent with existing interaction modalities in mobile AR applications. We drew inspiration from presentation tools such as Google Slides and enabled novices to create a linear storyline by creating a series of scenes. Within each scene, users can edit the triggers for the scene and the behaviors of AR content and IoT devices, inspired by existing event-trigger models (e.g. as seen in [90]). Finally, users can play the story that they create in the same system. We detail the design and implementation of the immersive stories and the authoring system in the following sections.

8.3.1 Jigsaw's Immersive Stories

Inspired by the style of comic strips in storytelling and slide decks in presentation, we represent each immersive story using a sequential number of scenes. The story proceeds as the system transitions from the previous to the next scene. The transition of scenes can be triggered by either keyword detection (e.g., when the user says a keyword) or touch interaction (e.g., when the user's hand collides with an AR object in the physical world), similar to how the transition in Google slides is triggered by pressing a button on the mouse. Each scene contains states of AR content (e.g., appearing/disappearing of a 3D model) and IoT devices (e.g., turning on/off a smart device). We used three types of off-the-shelf IoT devices in these immersive stories: smart lights, smart fans, and smart speakers. In the below sections, we present the design considerations of each immersive story.

8.3.1.1 Benjamin Franklin Kite Experiment

The Benjamin Franklin Kite Experiment is a story of Benjamin Franklin and his son taking a kite out during a storm to see if a key attached to the string would draw an electric charge (as seen in Fig. 8.1a). To play the Benjamin Franklin story, three users are required to be co-located with each user holding a phone. At the beginning of the experience, users coordinate according to the app and assign each user a different role: a narrator, an actor of Benjamin Franklin, and an actor of Benjamin Franklin Jr. After assigning the roles, users can see the two actors' bodies augmented with virtual costumes on them. The app then asks the narrator to narrate the story by reading a physical book out loud. As the narrator reads the story on the physical book, the app uses voice recognition to transcribe what the narrator says and automatically detect keywords in the transcription. A list of keywords that are selected based on the physical book and their corresponding scenes are predefined in the app. As one keyword is detected, the app will jump to its corresponding scene and wait for the next trigger. In this story, we defined a total of six pairs of triggers and scenes, with the last trigger being a touch event between users' hands and a virtual key in AR.

In this experience, we focused on using IoT devices as background augmentation, having human actors interacting with the story, and having a narrator reading from a physical book. This augmentation is considered to be in the background since IoT devices add to the story's ambience, such as dimming the lights to simulate cloudy weather. The costumes and key touching interaction are intended to facilitate the sense of participation. Lastly, the design of reading a physical book is to increase the sense of physicality, where the physical interactions of flipping pages and reading printed text can be meaningful in the experience.

8.3.1.2 The Wind and The Sun

The Wind and the Sun story is one of Aesop's Fables, wheretwo characters quarrel about which of them is stronger (as seen in Fig. 8.1b). Any number of users can play the story with each user holding a phone. The system narrates the story through the smart speaker. Users can tap on the screen to make the story proceed. In this story, the IoT devices act as the driving actors of the story with visual augmentations from the phone's AR, where the system superimposes the Wind on the smart fan and the Sun on the smart light.

Compared to the previous story, this experience is similar to a 4D movie with less focus on interactivity. The users take a passive role, and the primary user who would have been the narrator before now only has control of moving to the subsequent scene.

8.3.1.3 Goodnight Moon

We recreated Goodnight Moon [26], an American children's story written by Margaret Wise Brown in 1947. The narrator greets objects around the room (e.g., "Goodnight, red balloon. Goodnight lamp. Goodnight fan"). The greeting triggers an appropriate change in the greeted AR or IoT device (e.g., the balloon goes down; the smart lamp turns off; and the smart fan stops blowing wind). The story progresses with keyword detection as the narrator speaks, similar to the Benjamin Franklin story. However in this experience, the story is shown on the screen rather than on a physical book (as seen in Fig. 8.1c).

Because the narrator explicitly greets the IoT devices, the devices take on more of a foreground role. While the audience still has a passive role, the narrator is now an active, interactive participant in driving the story. The one-to-one mapping of each narration line with an effect on some object (AR or IoT) creates a simple but magical experience.

8.3.1.4 Implementation

We implemented the three stories using Lens Studio[2], which provides AR tracking, multi-user co-located experience, body pose detection and speech-to-text/text-to-speech capabilities. These features are necessary to enable the rich set of interaction techniques of the system, and motivated our choice for Lens Studio. We connect and interact with the smart light using Kasa Smart's API [4]. We used an IR-controlled fan, along with Bond Bridge [3] for networked control of IR devices. Lastly, we used Amazon's Echo smart speaker [9] connected to the narrator's phone via Bluetooth.

[2]https://ar.snap.com/lens-studio

8.3.2 Jigsaw's Authoring Tool

Following the *trigger-and-scene* framework that was used to build different immersive stories, we introduce the authoring tool of Jigsaw which allows novices with little to no technical background to create these immersive stories with mobile AR and IoT devices. We first present a system walkthrough of a step-by-step implementation of the Goodnight Moon story using Jigsaw's authoring tool. Then we introduce the underlying techniques including our scene-based authoring approach and in-situ editing in AR. Lastly, we demonstrate the generalizability of our authoring system by illustrating how to replicate the other stories, i.e., Bejamin Franklin and the Wind and the Sun.

8.3.2.1 Scene-based Authoring and In-situ Editing

Jigsaw enables novices to author immersive stories by creating a sequence of scenes in the editor, which draws inspiration from prior work on scene-based authoring (e.g., [42]) and presentation tools such as Google Slides.

In Jigsaw, users are able to define the following primitives of immersive stories via Jigsaw's authoring tool:

- **Behaviors**: A behavior represents the state of either a virtual object in AR or an IoT device (e.g., appearing/disappearing of a 3D model, on/off of a smart light).
- **Scenes**: A scene represents the state of the immersive story and can consist of zero to multiple behaviors.
- **Triggers**: A trigger is used as the transition from the current scene to the next scene. Each scene progresses to the next through exactly one trigger that the user defines.

In Jigsaw, three trigger types are available: tapping on the screen, keyword recognition via speech, and touching a virtual object in AR. Keyword triggers, as those in the Goodnight Moon story, allows users to define custom keywords for entering each scene. This was incorporated since verbal narration is a key component of storytelling experiences. The touching trigger enables users to enter a new scene when their hands collide with an AR object. This is one of the basic interaction in AR applications in order to add to the interactivity of the stories. Lastly, tapping is the default trigger for stories that are narrated by the system where the user enters a new scene by tapping on the screen.

In addition to defining *triggers* to different scenes, Jigsaw allows novices to edit *behaviors* of AR content and IoT devices within each scene directly in AR. This is aligned with the spirit of What-You-See-Is-What-You-Get that numerous authoring tools employ to lower

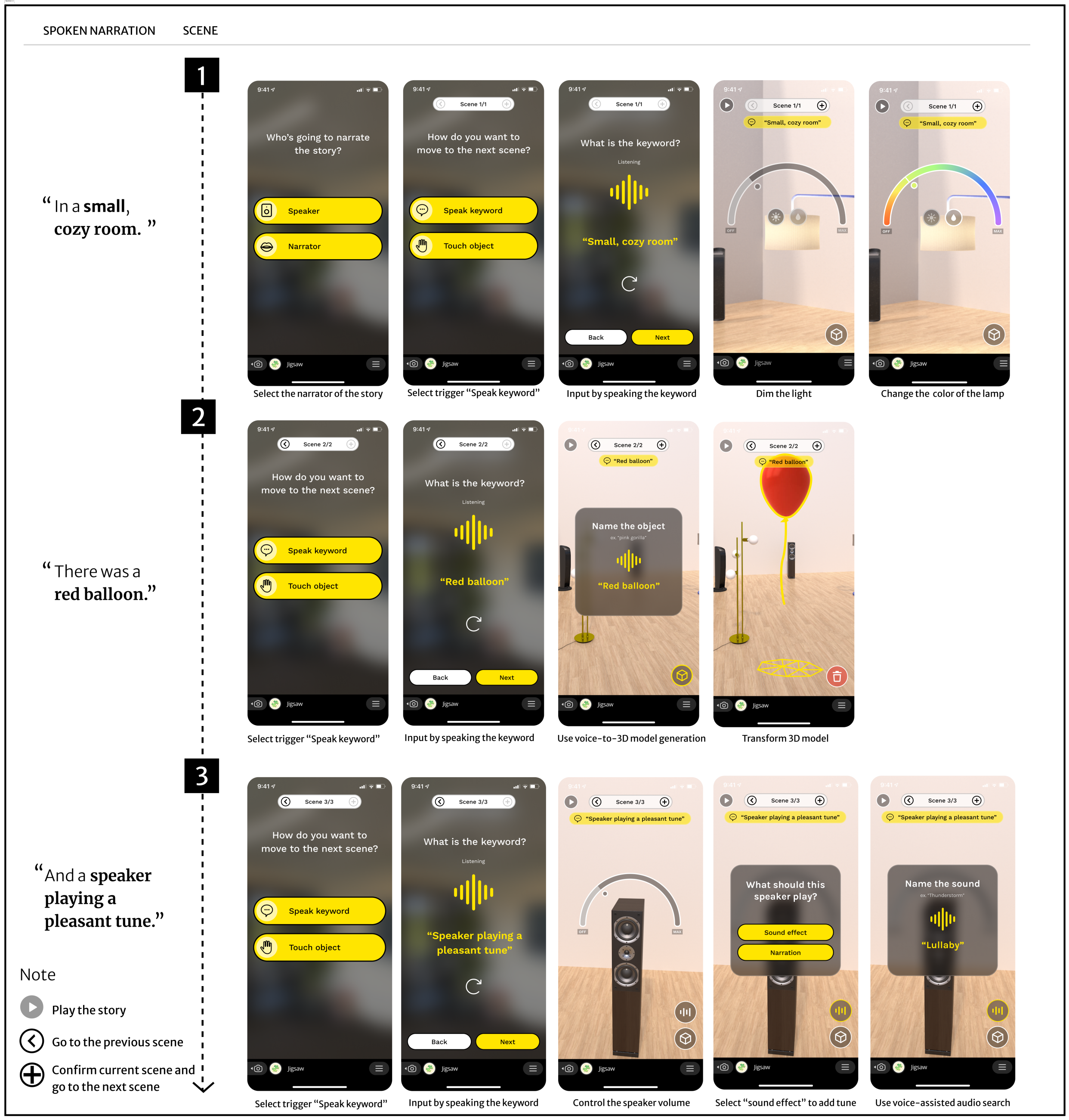

SPOKEN NARRATION
SCENE
1
“In a small, cozy room.”
Who's going to narrate the story?
Speaker
Narrator
Select the narrator of the story
How do you want to move to the next scene?
Speak keyword
Touch object
Select trigger “Speak keyword”
What is the keyword?
Listening
“Small, cozy room”
Back
Next
Input by speaking the keyword
Scene 1/1
“Small, cozy room”
Dim the light
Change the color of the lamp
2
“There was a red balloon.”
Scene 2/2
Select trigger “Speak keyword”
“Red balloon”
Input by speaking the keyword
Name the object
ex.“pink gorilla”
Use voice-to-3D model generation
Transform 3D model
3
“And a speaker playing a pleasant tune.”
Scene 3/3
Select trigger “Speak keyword”
“Speaker playing a pleasant tune”
Input by speaking the keyword
Control the speaker volume
What should this speaker play?
Sound effect
Narration
Select “sound effect” to add tune
Name the sound
ex.“Thunderstorm”
“Lullaby”
Use voice-assisted audio search
Note
Play the story
Go to the previous scene
Confirm current scene and go to the next scene

Figure 8.2: A step-by-step demonstration of creating the first three scenes of the Goodnight Moon story using Jigsaw's authoring tool.

the barrier for end-users to create immersive content [? ?]. An interface for importing 3D assets is shown on the screen when the user edits the scene. By saying the name of the 3D asset that they want, users can add the 3D asset to the AR scene. They can then adjust the placement of the asset by tapping and dragging the asset on the screen; they can also adjust the depth dimension by moving the phone in the physical environment while dragging the

3D asset. Jigsaw also enables in-situ editing of IoT devices including smart lights, smart fans, and smart speakers. A corresponding editing interface is pops up when the distance between the user's phone and the IoT device is below a pre-defined proximity threshold. For the smart light, users can edit its color, brightness, and special effects. The color is visualized in a semi-circular palette and the user can change its color by tilting the phone. Users can change the brightness using the same control, where the value of 0 will turn off the light. Lastly, users can add special effect of smart lights by saying the name (e.g. flickering). We envision delegating the low-level implementation of special effects to advanced developers and only expose the effect name to novice creators. Similarly, users can edit the on/off and intensity of the smart fan, and the on/off, volume, and special sound effects of the smart speaker.

8.3.2.2 System Walkthrough: Building the Goodnight Moon Story

We present a system walkthrough involving a general end-user who has little to no technical background and is motivated to use Jigsaw to create the Goodnight Moon story that end-users can play with their families and friends (as seen in Fig. 8.2).

Opening Jigsaw's editor, the user first selects the narrator of the story to be users rather than the system. After selecting the narrator, the user then enters the initial scene which is an empty scene. The first scene that the user is going to build for the story is dimming and changing the color of the smart lights when the user says "In a small, cozy room..." From the initial scene, the user can create a new scene. Before entering the new scene, the system prompts the user to define a keyword; in this case, the user speaks out the keyword "small" and confirms.

After the trigger is defined, the user can edit the new scene by defining behaviors of the smart light. As the user walks closer to one of the lights, an editing interface of the smart lights is shown in AR based on the user's proximity to the lights and allows the user to edit the status of the lights. The user can toggle between changing different attributes of the smart light including the brightness and the color. They can then turn down the brightness and change the color of the smart light. Jigsaw allows users to define multiple behaviors in a scene (e.g., they can walk up to other IoT devices to edit additional behaviors) though in this specific story the scene only contains the behavior of one device, i.e. the smart lamp. In this way, the first pair of trigger (i.e., the keyword "small, cozy room") and scene (i.e., changing the brightness and color of smart lights) is recorded in the system.

Upon finishing the first scene, the user can add a new scene, which saves the existing scene and jumps to the definition of a new trigger. The second scene that the user is going to build for the story is making a red balloon appear when the user says "There was a red balloon."

The user then defines the corresponding keyword as the trigger and starts editing the new scene. In addition to changing the status (e.g., on/off and intensity) of IoT devices, Jigsaw also allows users to create, manipulate, and delete 3D models or animation. In this case, to add a balloon model, the user can tap on the corresponding button and say the model name (e.g., "red balloon") directly to the system. The system then dictates the model name spoken by the user and shows the corresponding model in AR.

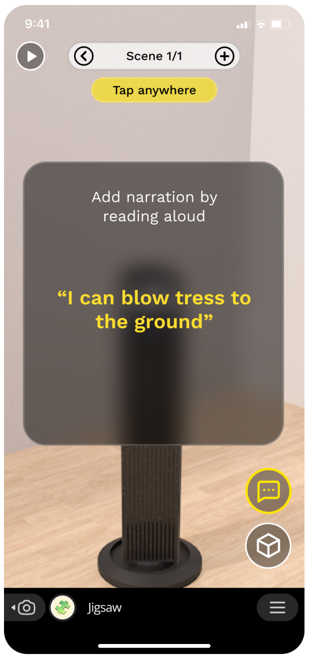
Next, the user places the content in the physical space through a simple drag-and-drop interaction; alternatively, they can delete it by dropping it to the trash icon on the screen. The system can create a fade-in, fade-out, and moving animation by default. For example, when scene 1 does not contain a balloon and scene 2 has one, the system can create a fade-in effect for the balloon in scene 2 during playback. Similarly, the system automatically applies a translation animation to a 3D object by moving from the position of the previous scene, to the position of the next scene during playback. In this way, the user can author scenes such as a moon model appears and disappears.

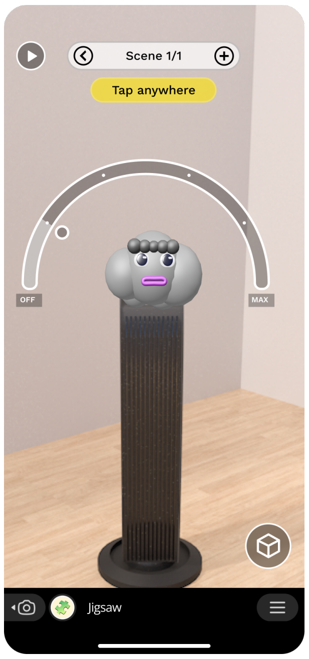
After creating the two scenes, the user can create the following scenes in a similar manner as the story proceeds. Note that the Goodnight Moon story contains 11 pairs of triggers and scenes in total. Jigsaw allows users to navigate and edit previous scenes. For example, the user can navigate to the first scene and change the color of the lamp. Finally, using Jigsaw, the user can play the storytelling experience immediately in the authoring environment.

8.3.2.3 Generalizability of the Authoring System

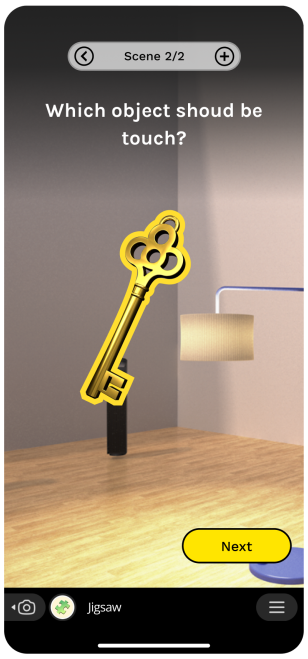
A key factor that makes our authoring system generalizable is that the the triggers and scenes that users create are open-ended and not coupled with any specific story. For example, a user can define any keyword or touching with any AR object as the trigger and use the brightness of smart lights for any story. To demonstrate the generalizability of the authoring system, we use replicated examples [150] by replicating the other two immersive stories using the same system.

To replicate the Benjamin Franklin story, users can define pairs of triggers and scenes like they do in creating the Goodnight Moon story. The unique challenges of creating this particular story include allowing multiple behaviors in one scene and defining touch triggers. Different from the Goodnight Moon story, the Benjamin Franklin contains multiple behaviors triggered by one keyword in a scene. For example, when the user says "when the clouds first passed over", the smart fan will turn on and the AR cloud model will appear. To achieve that, users can walk up to the smart fan and adjust its state, and then import the 3D model of a cloud while staying in the same scene. After editing all the behaviors in a scene, the user can proceed to add a new scene, which indicates finishing the current scene. The

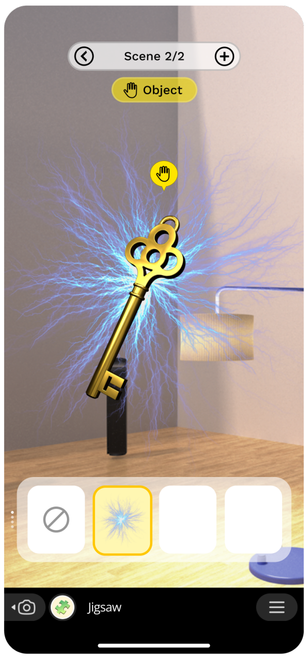

9:41
Scene 1/1
Tap anywhere
Add narration by reading aloud
“I can blow tress to the ground”
Jigsaw
(a) Adding narration.



(b) Editing the intensity of the smart fan.



(c) Defining the object for touch interactions.



(d) Selecting effects of models.

Figure 8.3: Authoring interfaces for adding narration, editing smart fans, and defining touch interactions. (a) For self-narrated stories like the Wind and the Sun, the default trigger for each scene is tapping on the screen and users can add narration to each scene. (b) Users can edit the intensity of the smart fan with the Wind character anchored to it in The Wind and the Sun story. (c) When defining the touch interaction in the Benjamin Franklin story, users can select which object to touch by tapping on the object. (d) Users can also choose built-in effects of 3D models when the touch interaction is triggered.

Benjamin Franklin story also incorporates touch triggers, where the user touches the virtual key model and triggers the flickering effect of smart lights, as seen in Fig. 8.3c & 8.3d. Jigsaw accomplishes this by allowing users to select trigger types besides keyword recognition when adding a new scene. After selecting touching as the trigger type, the user can then select the object for collision detection by touching it directly in AR.

The Wind and the Sun story is different from the Goodnight Moon example since it requires the system instead of the user to narrate the story. Jigsaw enables this by allowing users to select narrators at the start of the authoring experience, as seen at the beginning of Fig. 8.2. After selecting the system as the narrator, the user can add a narration to each scene by saying it directly to the system. The system can then transcribe the narration and play it through the speaker during the playback, as seen in Fig. 8.3a. When the system is assigned as the narrator, the system will omit trigger selection when adding a new scene and the trigger type goes to default, i.e. tapping on the screen. Users can import 3D assets such as the Wind character and place it in the physical environment such as anchored to the smart fan. They can also edit the intensity of the smart fan as seen in Fig. 8.3b.

8.3.2.4 Implementation

The authoring system is implemented using Lens Studio for displaying and interacting with AR content. The connection with IoT devices is built using the same technique as the implementation of the immersive stories mentioned above. We used the built-in voice recognition and hand tracking component in Lens Studio to enable users to define keyword and touch triggers. The playback of the final experience among multiple users is implemented using the built-in connected lens component.

8.4 Study

We conducted an in-lab user evaluation with 20 participants (10 groups) with two goals: *(i) to investigate the benefits and challenges of immersive storytelling experiences with both virtual and physical augmentation, and (ii) to assess the usability and utility of the authoring system.*

8.4.1 Participants

We recruited participants from a technology company in the United States through Slack posts and e-mails. Participants (ages 21 to 36) were randomly paired into 10 groups and volunteered for the one-hour study. Our study was approved by the privacy & legal team in our institution. Participants had either strong ties (e.g. friends) or weak ties (e.g. colleagues) within each group. All but one of the participants had prior experience of using IoT devices such as Amazon Alexa. All but two of the participants had used AR products such as mobile AR or AR glasses before the study.

8.4.2 Apparatus

The immersive experiences used three iPhone 13 Pro Max devices for AR visualizations and interactions. The same devices described in section 8.3.1.4 are used to enable physical augmentations, such as lightning and storm effects.

Table 8.2: The first three scenes of the Goodnight Moon story for the authoring task.

Scenes	**Keywords**	**Behaviors**
Scene 1	“small, cozy room”	The brightness of smart lights becomes 20%.
Scene 2	“red balloon”	A red balloon model appears in the room.
Scene 3	“speaker playing a pleasant tune”	The smart speaker plays the sound “lullaby”.

8.4.3 Study Procedure

Our study was divided into two sessions, each lasting 30 minutes. Prior to the study, participants first signed an agreement form that confirms their consent for our data collection and usage. They also completed a short entry survey about their demographic information and their prior experience of using creation tools for content such as image/video (e.g., Adobe Photoshop) and presentations (e.g., Google Slides).

8.4.3.1 Experience session

In the first session, we asked the participants to play the three immersive stories that we created in the order of Goodnight Moon, the Wind and the Sun, and Benjamin Franklin. We started with Goodnight Moon for its simplicity, where each trigger links to one behavior, and concluded with the more complex Benjamin Franklin story involving multiple behaviors and interactions. We chose this story order to progressively familiarize users with the immersive storytelling landscape, aligning with our goal of exploring this emerging paradigm of immersive stories. Before playing each story, participants were debriefed about the story background and their roles in the story. Since the Benjamin Franklin story requires three people to experience, one member from our study team acted as the Benjamin Franklin Jr. In this story, one participant acted as the narrator holding the physical book and the other participant acted as the character listening to the story and touching the key. Our team member filled this audience role their role was only listening to the story, which overlapped with existing participant roles, ensuring all participation aspects were experienced. After experiencing the three stories, we conducted a semi-structure interview with both participants to ask about their experiences and the benefits and challenges of the immersive stories.

8.4.3.2 Authoring session

In the second session, we spent the other half an hour evaluating the authoring system with one randomly picked participant from the group. We decided to ask participants to first experience and then recreate as we believed it is essential for them to understand this new paradigm of immersive stories (i.e. "what" immersive stories are) before coming up ways to craft them (i.e., "how" to create immersive stories). The participant was first given a 10-minute tutorial introducing the functionalities of Jigsaw's authoring tool, including defining triggers, creating scenes, and editing behaviors of 3D models and IoT devices. After ensuring the participant understood how the authoring system works, we then asked the participant to complete the task of recreating the Goodnight Moon story that they experienced in the first session. We asked participants to replicate existing stories rather than creating their own since having one consistent story across all participants can help us determine whether participants were able to create immersive stories using Jigsaw's authoring tool. In the evaluation of Jigsaw's authoring tool, we also emphasized the creation process rather than the story content. During the task, the participant was provided with the complete script on a piece of printed paper including keywords and their corresponding behaviors. An example of the first three scenes in the script can be seen in Table 8.2. We provided this so that the participant could focus on *how* to create the story using the system instead of *what* to create. During the task, the study team was not allowed to provide any help unless the participant explicitly asked for help. Any question that the participant asked during the task was noted down. The task included setting up 11 pairs of keywords and behaviors and took 10 minutes in total. Finally, we conducted a 10-minute semi-structure interview with the participant to ask about the usability and utility of the authoring system.

8.4.4 Data Analysis

We aggregated the entry survey data and conducted a thematic analysis [24] of the transcriptions of the retrospective interviews for both the consumption and the creation of immersive stories via Jigsaw. Two researchers first developed a codebook based on four randomly selected participants' transcripts, including two from the consumption interview and two from the creation interview, identifying and labeling similarities across participants' experiences with Jigsaw. The researchers discussed and refined the codebook on another subset of participant transcripts. We then validated the codebook on two additional participants' transcripts, individually coding the same transcripts. After achieving high inter-rater reliability, with a Krippendorff's α above 0.8, they divided and coded the rest of the transcripts in parallel. Finally, we discussed and grouped related codes into themes according to our research questions.

8.5 Results

Participants' experiences playing immersive stories during the study suggested that Jigsaw's storytelling experiences with both virtual and physical augmentation were immersive, engaging, and memorable. In the evaluation of the authoring system, all participants were able to complete the creation task and applauded the system for being easy to use and generalizable for creating a wide variety of stories. We center our findings around the two evaluation goals, *(i) benefits and challenges of immersive stories*, and *(ii) the usability and utility of the authoring system*

8.5.1 Benefits and Challenges of Immersive Stories

Overall, our results suggest that the design of Jigsaw made the storytelling experience immersive, engaging, and memorable by adding augmentation of IoT devices. Participants felt they had control over the storytelling experiences via various forms of participation and interactions with elements of the story. Participants also highlighted the versatility of these immersive stories for various population and scenarios.

8.5.1.1 IoT devices made the experience immersive, engaging, and memorable.

Immersion is often associated with *extensiveness* (i.e., the range of sensory modalities accommodated) and *surroundings* (i.e., the extent to which the experience is panoramic rather than limited to a narrow field) of the experience [229]. Our results suggested that, in Jigsaw, the augmentation via IoT devices increases the *extensiveness* to make the storytelling experience more immersive by invoking more senses such as haptics from the smart fan.

> "if you just had AR simulated wind, there's something that's not as exciting about the fact of having real wind and having that effect come together...the wind being real because you can feel it."-P9B

In line with our motivation to create AR experiences beyond just "pixels" on the screen, participants also reported the storytelling experience being immersive as it extends from the screen to the IoT devices in the *surroundings.*

> "it's not only are you seeing it on the screen, it had panned out a little bit further and not only do you see it on the screen, but you're feeling it too, so it's 4D as opposed to just the AR experience of seeing it." -P11B

With the enhanced immersion, the storytelling experiences became memorable and engaging to participants due to the various neural stimuli involved in the experiences.

> "I think it (without IoT) would've been less of an immersive experience and also probably little less memorable. So you remember, "Okay, at some point the moon went to sleep, the lights went out," and it's a bit easier to remember visual cues than it is just reading it off. And then also staying with the story was a lot easier because things are changing around and keeping me engaged." -P2A

8.5.1.2 Users' participation and interaction improves the sense of immersion and agency.

We also found that allowing users to act as different roles (e.g., narrators or characters in the stories) in the stories enhanced the sense of immersion. This is because when users act as characters in the stories and their interaction with the story elements triggers certain effects, it reinforces the impression that they are embodied as the character; when users act as narrators, they tend to focus more on the storyline and feel immersed.

> "It makes you feel like you are in a theatre with the whole background noise, the light changes, and the fact that you can touch things virtually... I was very involved in everything going on, so I was definitely very immersed." -P5A

> "I think being the narrator helped me be immersed in the story more. I think maybe because I had to focus on the story more... Since I'm narrating, I have to think of what's going to happen next or think what I have to say next. Whereas when I'm an audience member, I think it's a little easier to get distracted by what's going on and then lose what the computer's saying." -P7A

Agency refers to "the satisfying power to take meaningful action and see the results of our decisions and choices" [177]. In addition to prior digital storytelling experiences that explore the sense of agency via questions and answers [289], we found that being the narrator in the immersive stories can give participants the sense of agency since they feel that they have control over how the story goes, by closely examining the mapping between triggers and scenes based on their own paces.

> "I definitely enjoyed being a narrator more because I felt more in control of the actions that I was committing throughout the story. So let's say I was saying the story and then something came on the AR screen, I was able to look around and say, okay, I'm done looking at that. I'm ready for the next thing." -P7A

8.5.1.3 Sensory overload is the primary challenge in immersive stories.

While Jigsaw's immersive stories are immersive, engaging, and memorable, we found that our design of immersive stories also comes with the challenge of sensory overload. We found that participants' attention was occupied by participation as different roles, AR effects on the phone, and IoT effects in the surroundings, making it difficult for them to catch up with all the changes. For example, participants reported missing the IoT behaviors easily when they were focusing on the AR effects on the phone.

> "I think if the experience doesn't happen at the same time... that would make sense, because I feel like if you're already dealing with something flashing in your phone and you're going to blink and miss what's in your world, because you're really focusing on what's on your phone there." -P11A

In addition to the tension between AR content on the screen and IoT behaviors in the surroundings, participants also expressed sensory overload of both reading the physical book and paying attention to the changes on the phone.

> "I love reading books. I love reading to children. I didn't need to have a phone. The reader in that case doesn't need to have any devices. I don't need to be a part of the immersive experience. I can just be the reader reading to the people in the room. So I didn't really like having the phone sitting beside me because I'm just like, now I'm tempted to pick up the phone, but I'm trying to read this story in a really nice way. So it just felt like there was this weird divide and I felt like I was being pulled in two different directions and not really being able to fully be in the moment." -P4B

8.5.1.4 Immersive stories are versatile for various population and scenarios.

Lastly, we found that Jigsaw's immersive stories are versatile for both ambient and interactive scenarios, and for both kids and adults. This corresponds to our motivated design explorations of using IoT devices for both foreground and background augmentation (Table 8.1). For example, participants commented that the foreground augmentation of IoT devices could be used for stimulating children to keep them engaged and for offering interactive gaming experiences such as escape rooms for adults.

> "because children have low attention spans, they're all looking at screens all day long, watching those 15-second unboxing YouTube videos... they need to be stimulated to be engaged, so that's why I see this being working wonders with children" -P4A

> “I can’t tell you of how many escape rooms I know where you trigger one thing happen in the room, like flickering lights. That would be perfect for it.” -P11B

In addition, we found that when Jigsaw’s immersive stories can also be used for ambient scenarios, when IoT devices were used as background augmentation to set the mood of the experience.

> “I think one thing that I would love to see, so maybe for the adult story part, would be more horror type of stories. I could see a potential there because you have lights flickering, you have sort of sounds that come out of nowhere or wind that starts blowing.” -P8A

> “I have a set bedtime and 30 minutes before that I’m supposed to do this end of the day meditation, where I watch a sunset. And I just thought it reminded me of that, because it would be really cool to see that in AR and then maybe you power down your devices, because I’m supposed to be off my phone until bedtime. You power down your TV and electronics and screens and stuff, lower the lights... that type of transition.” -P10A

8.5.2 Usability and Utility of the Authoring System

We also sought to understand the usability and utility of Jigsaw’s authoring tool, as we were curious whether users are able to create these new immersive stories. We found that the design of Jigsaw’s authoring tool has a low entry of barrier and is generalizable due to the open-endedness of the triggers. However, the Jigsaw’s authoring tools face challenges of creating more complex stories such as creating custom AR animation, coming up with triggers and behaviors, and authoring longer stories. We describe these in detail below.

8.5.2.1 The authoring system has a low barrier of entry.

Participants commented that the authoring system is easy to learn, partly because the in-situ interface that Jigsaw’s authoring tool offers allows users to directly edit behaviors of AR and IoT devices in AR with little technical literacy such as imperative programming.

> “I think if I was a non-technical person, it was pretty easy to use because you just point at an object and then you can interact with it. So that mechanic, rather than me specifying things through a complicated user interface or a bunch of text, makes it a lot easier to just as a lay person to be, ”Oh, I want the lights to turn on or to be red,” and then you can just do it...I’ve actually made a lens

> that controls Internet of Things before, but I had to do all of this through script and code, and it's not simple to use in that sense." -P8A

This is in line with the benefits of prior authoring tools for immersive experiences that take advantage of directness (e.g., [153, 286]). In addition, participants reported that the authoring system makes it available to settings other than theaters due to less production cost to create immersive storytelling experiences.

> "For me, I almost immediately thought of preschools and kindergartens... especially at younger ages, you don't have that much production... so, just having, I would say, devices take cares of a lot of that."-P2A

8.5.2.2 Open-ended triggers make the system generalizable and fun to use.

Participants commented that the keyword triggers were open-ended, which made the authoring tool generalizable and fun to use. The ability to define custom keyword triggers, powered by Jigsaw's speech recognition capabilities, made users feel like they could "do anything with it" (P10B).

> "I guess the benefits are with the keyword part that's super open-ended versus just on a Google Slides, you know, have only preset options, and so you're somewhat limited to the options that they provide you with. But the keyword element being something where you could just really make it any word that you want to say, it makes that part fun for the interactions themselves"-P9A

The flexibility to specify any number and any types of behaviors triggered also enhanced the open-ended nature of the triggers.

> "It's actually a story so that I can see this stuff can put me into a situation where I can probably trigger multiple things following the story as long as the story is intuitive and easy to follow. So that's definitely a lot add on there." -P5A

> "It would really be able to do a whole lot for each keyword or throughout the entire story that it's really going through a full experience. I like that it's not just one thing, so not just one keyword equals one action. One keyword can equal ten actions if you have it set up correctly." -P3A

8.5.2.3 Creating complex stories is challenging using Jigsaw.

We found that the expressiveness of our authoring system is limited when participants wanted to author special animation and more complex behaviors of objects such as making a shaking animation or setting the duration for the status of objects' behaviors. Authoring these behaviors typically requires being able to edit lower-level attributes of the experience such as time, mesh, and position.

> "Maybe if you wanted to make it feel like the ground was shaking or something like that, that would be hard."-P9A

> "If you wanted to have a flicker or setting a duration of the fan so it's like, oh, it's on for 10 [seconds] off for 10 [seconds], back on for another 30 [seconds]. Something like that, having more complex actions for each object would be really interesting." -P3A

8.5.2.4 Authoring long stories is challenging due to the limit of creation support.

Finally, we found that Jigsaw's authoring tool is suitable for authoring short stories rather than long stories due to the lack of support for duplicating triggers, scenes, or behaviors, and for navigating longer sequence of scenes.

> "it would be too much work to copy from one scene to the next and let's say you want to have duplicated scenes or something like that. But yeah, I feel like children stories because they're kind of simple to create." -P7A

> "I think this one, like I said, was a bit more tedious because editing on for example, Adobe Premier Pro, I was able to just pick exactly what parts I want and put them next to each other whereas with this one, it's based on the keyword's is the next line. The more I have, the more work it's going to be, whereas the more clips I have might not necessarily be a linear growth at the time." -P2A

Participants also commented that they might encounter challenges of coming up with pairs of triggers and scenes if they were going to create their own stories and would like more creation support during this process.

> "I feel like it should have said something like, "Okay, now do you want to trigger the lights, the audio, or the fan?" and then you could decide. You already gave me the script and the script told me exactly what to do, but if I was trying to just create my own story from scratch for the very first time, at least once when you first make your first story there should be more prompts." -P4A

8.6 Discussion

Overall, we found that Jigsaw enables immersive, engaging, and memorable storytelling experience through the combination of mobile AR and IoT devices. We found the augmentation via IoT devices made the stories more immersive by adding more senses to the stories and expanding beyond “pixels” on the screen. Users’ participation that Jigsaw enables in the immersive stories also enhances users the sense of immersion and agency. In addition, our findings suggest that immersive stories should be carefully designed as mobile AR, IoT effects, and participation can cause sensory overload. We found that the immersive stories are versatile for various population and scenarios. Our results also revealed that participants were able to use Jigsaw’s authoring tool to create immersive stories. Our authoring tool can also enable users to create diverse stories due to the open-endedness of the keyword triggers. However, our authoring tool has challenges in expressiveness for creating complex behaviors of objects or effects. We highlight the opportunities and challenges of immersive storytelling experiences using mobile AR and IoT devices, and discuss design implications and limitations based on our results.

8.6.1 Design Implications and Future Directions

Based on the results of evaluating Jigsaw’s immersive stories and authoring tool, we highlight the opportunities and challenges of both consuming and creating immersive storytelling experiences via AR and IoT. We focus our discussions on how to make digital storytelling experiences more immersive via physical augmentation and user participation, how to create immersive storytelling experiences by enabling open-ended triggers and scenes, and how to support authoring more complex behaviors of AR and IoT.

8.6.1.1 Adding immersion via physical augmentation and user participation

We found that the augmentation of IoT devices and user participation enhanced the sense of immersion during storytelling experiences. This extends prior effort of bringing in physical augmentation and participation for immersive experiences such as 4-D theatres and participatory plays [197]. Our study indicates that enhancing readily available IoT devices, instead of using specialized, costly equipment, can effectively promote immersive storytelling experiences. Our results also align with prior exploration of dimensions of immersion including *extensiveness* and *surroundings* [229]. We found that Jigsaw’s immersive stories can effectively increase the *extensivenss* of immersion by adding in sensory augmentations and the *surroungdings* by extending the storytelling experience beyond just the content on the screen.

We encourage future researchers and practitioners exploring immersive stories integrate elements of physical augmentation and user participation to add to the immersion of stories. For example, future immersive stories might explore how to extend the experience beyond visual displays on the screen via more IoT devices such as robots and projectors which can be used for diverse tasks and content. They may also explore immersive stories that involve other human senses such as haptics [6] and tastes [25]. They can also explore more means of participation beside being narrators, actors, and audience and investigate how they can affect users' sense of immersion during storytelling experiences. For example, prior work has explored the involvement of generative AI for storytelling experiences [47, 289] and future research can explore how these new roles of user collaboration with AI could affect their sense of immersion in storytelling experiences. In addition, another key insight that we gained from the results is that participants faced challenges of catching up with everything in the immersive stories due to sensory overload. Therefore, future research should explore how to design immersive storytelling experiences that could balance various sensory augmentations and modalities of participation.

8.6.1.2 Authoring immersive stories via open-ended triggers and behaviors

Our findings show that Jigsaw's open-ended approach to trigger and behavior definition enables it to be widely adaptable for authoring diverse stories. Users can assign a single keyword (e.g. "balloon") or behavior (e.g. smart light brightness change) to serve different narrative functions across multiple stories. We found that this flexibility can empower participants to get creative and develop novel ways of linking triggers with behaviors for diverse immersive experiences. This aligns with prior work in creativity support tools that aims to enable people to concentrate on creative designs by removing low-level implementation and supplying appropriate building blocks [97]. We suggest future research and design in authoring immersive storytelling to consider a range of triggers and behaviors that are essential and open-ended for building immersive stories. For example, in the context of authoring context-aware IoT applications, prior research has explored triggers such as the user's location and the state of IoT devices to activate events of IoT devices [267]. While being intuitive and versatile for authoring immersive stories, Jigsaw's authoring tool also has challenges of authoring long stories. Participants also reported challenges coming up with the appropriate triggers and their corresponding behaviors. Future research could thus explore ways to automate the process of defining triggers, scenes, and behaviors. For example, one can imagine narrating the story verbally for the first time and the system could intelligently detect and record triggers and behaviors on the fly.

8.6.1.3 Supporting end-user creation of complex behaviors of AR and IoT

While being generalizable for creating various stories, one limitation of Jigsaw's authoring tool is its expressiveness for defining complex behaviors such as interactivity. Such complex behaviors typically require coding to accomplish. We encourage future researchers to explore ways to raise the ceiling of expressiveness of the authoring tool for immersive stories while keeping the low barrier of entry for end-users. For example, in the context of editing animation, visual programming platforms such as Alice [200] and Scratch [?] could be incorporated into authoring tools like Jigsaw to enhance its expressiveness. In-situ visual programming approaches (e.g., [290, 153, 286]) can also be considered as a way to enhance the expressiveness while augmenting the existing intuitive in-situ controls of IoT devices in Jigsaw. For example, FlowMatic [286] utilized a visual flow-based diagram that enables end-users to create complex behaviors of virtual objects in VR. In addition, future researchers can consider ways for reusing or remixing [?] to make the authoring experience even more expressive for adapting to different storytelling contexts. The functionality of remixing and reusing could also foster a broader and more vibrant community for exchanging storytelling ideas.

8.6.2 Limitations

Our results demonstrate the benefits and challenges of both the consumption and the creation of immersive stories. However, our work is not without limitations. First, Jigsaw currently incorporates a constrained set of modalities including mobile AR, smart lights, fans, and speakers. Further research should explore additional modalities like head-worn AR and diverse smart devices to fully uncover the possibilities of immersive storytelling. Second, our qualitative evaluation focused on adults; given children represent a key demographic for engaging with and creating stories, future studies could examine their perceptions and involvement. Our particular order of experiencing and re-creating immersive stories in the study could also introduce bias such as learning effects [79]. For example, users might have learned what the expected behaviors are prior to the authoring task. One way to avoid this in the future could be evaluating how end-users can create their own immersive stories, which can offer additional insights on the types of immersive stories that Jigsaw's authoring tool enables as well. Follow-up work could also undertake more quantitative investigations into how immersion is impacted by blending virtual and physical augmentations. Finally, while designed for storytelling, Jigsaw's adaptability suggests applications in other domains like education, entertainment, and communication. Further research might examine how combining IoT and AR could play a role in these and other contexts. In summary, our work provides initial insights but warrants expanded investigations into users, technologies, and applications to elucidate the full potential of immersive storytelling.

8.7 Conclusion

In this chapter, we investigated immersive experiences created by integrating mobile AR with off-the-shelf IoT devices. We also introduced an authoring tool for crafting these immersive narratives. Our qualitative lab study revealed augmenting IoT devices and enabling user participation can enhance immersion in storytelling, though sensory overload poses a primary challenge. We also found our tool's open-ended approach to defining keywords and attributes enabled authoring diverse stories. These insights highlight opportunities and obstacles around AR+IoT storytelling. Our recommendations for future research and design include: furthering immersion via physical augmentation and participation; enabling immersive story creation by editing adaptable narrative components; and supporting end-user authoring of interactive behaviors.

CHAPTER 9

Conclusion and Future Work

In this dissertation, I argued that by incorporating novel visualization and interaction techniques in immersive environments, we can design end-user creation tools that can define reactive behaviors, support design exploration and collaboration, and have meaningful impact on social interactions. In this chapter, I summarize how my work contributes to prior literature, followed by potential directions for future work.

9.1 Thesis Contributions

Through system building and a series of in-depth user evaluations including both qualitative and quantitative methods, my dissertation has made the following contributions:

- In Chapter 3, I document a qualitative study that evaluates the challenges and benefits of the state-of-the-art immersive dataflow programming tools. Specifically, we found that dataflow is still an effective option for immersive authoring, but there is little benefit to giving users complete 3D freedom in the manipulation of programming primitives, which affected our design of FlowMatic in Chapter 4.
- In Chapter 4, I present a novel immersive authoring tool named FlowMatic that raises the ceiling of the expressiveness of immersive authoring tools, by allowing users to define reactive behaviors and to programmatically create and destroy objects in a scene. Through a qualitative comparative study of FlowMatic and conventional desktop-based authoring tool, I demonstrate its usability and benefits such as low thresholds for novice programmers and the immediate feedback. I also present a series of example applications prototyped with FlowMatic, demonstrating its expressiveness.
- In Chapter 5, I approach design explorations powered by version control systems in immersive environments. I introduce the design and implementation of a new VCS named VRGit for collaborative content creation in VR. Results and design insights

gained from an exploratory lab study show that VRGit can support history tracking, design exploration, and workspace awareness in collaborative content creation among end-users.

- In Chapter 6, I present VRCopilot, an immersive authoring system that enables users to interact and co-create with generative AI models in virtual immersive environments. Through a series of user studies, we evaluated the potential and challenges in procedural, scaffolded, and automatic creation in immersive authoring. We found that scaffolded creation using wireframes enhanced the user agency compared to automatic creation. We also found that procedural creation via multimodal specification offers the highest sense of creativity and agency.

- In Chapter 7, I present the design and implementation of an effortful communication system via digital handcrafting in AR. Results from a two-week field study show that end-user creation of AR experiences helped participants engage in meaningful effort through the handcrafting process, and feel closer to their partners, although the tool may not be appropriate in all situations.

- In Chapter 8, I present a system named Jigsaw that empowers beginners to both experience and craft immersive stories, blending virtual and physical elements. Findings from a study show that participants not only successfully created their own immersive stories but also found the playback of such immersive stories kept them immersed, engaged, and connected with other users.

9.2 Reflection and Future Directions

9.2.1 Reflection on Lessons learned

Resnick et al. identify a set of design principles for end-user creation tools including low threshold, high ceiling, and wide walls, which refers to supporting a wide range of creative explorations [213]. My dissertation advances end-user creation tools for immersive experiences based on the following principles: 1) VRCopilot lowers the threshold by automating the creation of immersive experiences using generative AI models; 2) while keeping the low threshold, FlowMatic and VRGit raise the ceiling by allowing end-users to define reactive behaviors and collaboratively explore design alternatives for immersive experiences; 3) Auggie and Jigsaw widen the walls by expanding the communicative and storytelling capabilities in immersive authoring tools.

The paradigm of immersive authoring started out with the goal of lowering the threshold for creating immersive experiences. By leveraging intuitive interactions such as direct manipulation and visualizations such as visual programming languages, the target audience can be shifted from experts to novices. However, this still requires the manual arrangement of 3D objects and the construction of visual programs. With rapid advances in AI, particularly generative AI models, it is now possible to automate much of this process, further lowering the threshold. The challenge, however, lies in balancing automation with maintaining users' sense of creativity and agency. Through our work with VRCopilot, we have found that using interaction techniques like intermediate representation and multimodal specification can provide more effective intelligent support. For future AI-powered creation tools, not only for immersive experiences but broadly, it is thus crucial to consider what kinds of interactions and controls we should expose to users to enhance their sense of creativity and agency throughout the creation process.

Raising the ceiling of end-user creation tools has been a persistent goal for researchers. In the realm of immersive authoring, tools like FlowMatic, which visualizes advanced programming paradigms such as functional reactive programming (FRP), and VRGit, which visualizes a collaborative version control system (VCS), elevate the ceiling by enabling users to create more complex immersive experiences and explore design alternatives. However, a significant challenge with these and other immersive authoring tools is that they often reach a point where their development is abandoned. Additionally, there is minimal skill transition between tools designed for end-users and those for experts. This is akin to the gap between visual programming languages like Scratch and imperative programming languages like Python or C++. The critical question is therefore to develop a framework that determines the appropriate abstraction or ceiling for immersive authoring tools to ensure they remain useful and can facilitate skill progression.

Finally, building wider walls means that end-user creation tools must support a broad spectrum of different types of experiences. Through Auggie and Jigsaw, I have demonstrated that these tools can facilitate meaningful social interactions in communication and storytelling. Prior research has also explored authoring tools for various domains such as education [290] and IoT [74] However, these tools are often mutually exclusive, with widgets and components not being shared across different platforms. The key idea for future researchers is to identify building blocks that can be combined in various ways to support more general immersive experiences. This approach will help create a more versatile and interconnected framework of end-user creation tools, enabling a wider range of applications and enhancing user creativity.

Next, I will highlight specific opportunities that I am excited to explore in the future.

9.2.2 Future Directions

Collaborative Immersive Programming at Scale. By building FlowMatic, I have raised the ceiling of expressiveness of immersive authoring via visual programming and demonstrated the advantages of building "live" programs directly in VR. In the future, how can we enable multiple users to collaborate on building immersive programs? How can we design a platform that allows teams to jointly exchange ideas, and build and manipulate code in real-time? Pair programming has been explored for computer science education. What are the unique advantages of immersive technologies in fields like education, engineering, and digital art? In addition, visual programming tools usually fall short on creating complex and large-scale applications. Both FlowMatic and VRGit have explored graph visualization in the immersive space. FlowMatic allows users users to reuse functionalities by combining nodes in the FRP diagram. VRGit introduces summarization techniques that can cluster operations and reduce the visual cluttering by clipping the HG based on the constraints. These were all sufficient for lab study tasks, but the graph visualization can become complex and hard to navigate when creating more complex 3D scenes. Future work could investigate techniques that allow visualization and interaction with the immersive graph visualization at larger scales. For instance, there are techniques that allow users to cluster operations manually [181]. One can also consider employing more sophisticated ways of automatic clustering operations or interactions, e.g., by leveraging techniques in interactive summarization of videos [15, 19] and 3D models [62, 64]. While the former is based on image analysis and the latter is based on geometry analysis of single objects, future work could extend this line of work by exploring interactive summarization techniques based on content creation operations clustering.

AI-empowered Creativity Support for Immersive Content Creation. One of my research goals to build a future where everyone can create and express themselves via 3D spatial media as easily as they can today with 2D media such as images and videos. One approach to achieving that is to provide AI-empowered creativity support for content creation while lowering the barrier to create. While recent advancements of generative models have revolutionized the creation process of traditional media including text, images, and videos, it is underexplored how generative AI can help users create immersive content. VRCopilot acts as the first attempt to explore human-AI co-creation with generative AI in the immersive space. However, a challenge we found is the expectation mismatch when unexpected output was generated by AI. For example, users generally have preferences of the layout design that fall outside of the capabilities of generative AI models. However, users do not have sufficient understandings of the capabilities of the system. This demonstrates that there should be a more transparent communication between users and generative AI in terms of the capabilities and limitations of generative AI. This aligns with the Explainable AI (XAI) research (e.g., [158, 65, 102]), where researchers aim to provide more transparent explanations of decision-making process of the AI model, with an emphasis on text or images. However, there has been little explorations in the visualization and interaction techniques for making AI models more understandable in the immersive environments. Therefore, future researchers and practitioners should consider designing human-AI systems that can visualize how the generative AI model perceives and completes the user's design. For example, future work might draw inspiration from past XAI research that has investigated techniques such as saliency maps that visualize how features of an instance contribute to the decision-making of AI models [228]. My prior work in content creation tools and human-AI co-creation provides grounding for this research.

Immersive Content Creation for Meaningful Social Interactions Another research goal of my dissertation is to contribute to the human process by empowering people to engage in meaningful and fun social interactions. Auggie and Jigsaw are two representative works that examine people's social interactions via immersive experiences in the topics of communication and storytelling. Social interactions inherently involves the generation of content to be communicated, a process that could involve a lot of skill depending on the content or background of the sender. Both Auggie and Jigsaw were aimed to reduce the barrier of entry to content creation by providing a small and compact set of intuitive creation tools, and found that even people who self-described as non-artistic were able to create immersive experiences that were engaging and aesthetically pleasing. This demonstrates the benefits of designing simplified, more widely accessible tools that can create sophisticated immersive content for social interactions Future work might continue to investigate how to lower the barrier for creating immersive content for social interactions, particularly involving system support. At the same time, polished content could feel less personal if the system has a high degree of control over the output compared to the user, replacing the user's agency [176, 125]. Future research should further explore the appropriate balance for system support versus user agency that can lower barriers to content creation while retaining authenticity. In addition, I am curious to continue to explore the unique affordances of immersive technologies and how those affordances could benefit social interactions among users. For example, I am interested in the unique aspects of immersive technologies such as the physicality of AR, the spatial visualization of VR, and the expressions via face, body, and hand could improve users' social interactions and impact their social behavior.

9.3 Summary

Current end-user creation tools for immersive experiences have many limitations, such as low ceilings of expressiveness, lack of support in design exploration and collaboration, and underexplored impact in social interactions. In this dissertation, I have described a suite of end-user creation tools that leverage novel visualization and interactions in the immersive environment to address these limitations. By incorporating the paradigm of Functional Reactive Programming (FRP) in 3D environments, we enable end-users to define reactive behaviors in immersive authoring, a challenging yet essential task for creating interactive immersive experiences. By integrating visualizations and interactions with version control systems (VCS) and generative AI models, we can facilitate the exploration of design alternatives through intuitive history tracking, enhanced workspace awareness for collaborators, and the creative support provided by generative AI. Lastly, I suggest that efforts to make authoring immersive experiences accessible to end-users have led to meaningful impacts on social interactions, such as fostering meaningful remote communication and co-located playful storytelling experiences. While many of the artifacts were developed for specific scenarios such as interior design or game scene design, the visualization and interaction techniques demonstrated could have broader generalizability. I believe that these techniques will become integral components of future end-user creation tools for immersive experiences. Ultimately, I hope that this dissertation represents a significant step towards the vision of democratizing content creation for immersive experiences to a broad range of people.

APPENDIX A

Study Surveys: VRGit

A.1 Pre-task Survey

1. What is your self-rated experience level in using VR? Select only one option below.

    - Beginner
    - Intermediate
    - Expert
    - Other:_______

2. What have you used VR for in the past? Check all that apply.

    ☐ Gaming and Entertainment

    ☐ Art and Design

    ☐ Architecture

    ☐ Education

    ☐ VR Development

    ☐ VR Product Design

    ☐ VR Research

    ☐ Healthcare

    ☐ Other:_______

3. Have you used any of the following tools/software to track multiple versions of your work? Check all that apply.

    ☐ Version Control System (e.g., Git, SVN)

☐ Adobe Creative Suite (e.g., Photoshop, Illustrator)

☐ Figma

☐ Google Doc

☐ Other:_______

A.2 Survey Part I

Participants answered the following questions using 7-point likert scale (i.e., "1" means strongly disagree and "7" means strongly agree).

Usability and Utility of Features

1. I found it easy to create a branch of the history.
2. I think it would be useful to create multiple branches.
3. I found it easy to navigate and enter different versions.
4. I think it would be useful to navigate and enter different versions.
5. I found it easy to see the preview (i.e. a larger miniature) of a version.
6. I think it would be useful to see the preview (i.e. a larger miniature) of a version.
7. I found it easy to merge objects from one version to another.
8. I think it would be useful to merge objects from one version to another.

System suitability

1. By looking at the history visualization, I can see the changes between two consecutive versions.
2. By looking at the history visualization, I can track the evolution of my design across various versions.
3. By looking at the history visualization, I can refer to other versions to inform my current design.

A.3 Survey Part II

Participants answered the following questions using 7-point likert scale (i.e., “1” means strongly disagree and “7” means strongly agree).

Communication and Awareness across Different Versions (using Portals)

1. I found it easy to see in which version my collaborator was located through the history visualization.
2. I found it easy to understand my collaborator’s work progress through the history visualization.
3. I found it easy to understand my collaborator’s design through the portal.
4. I found it easy to share my design with my collaborators through the portal.
5. It was easy to understand which items my collaborator was referring to through the portal.
6. I found it easy to understand what my collaborator was doing through the portal.
7. I found it easy to communicate feedback with my collaborators through the portal.
8. I think the portal with another user would be useful.

Communication and Awareness in the Same Version (using Shared Visualization)

1. I was aware of the presence of my collaborators when we are in the same version.
2. It was easy to understand my collaborator’s actions when they are in the same version as me.
3. It was easy to collaboratively go to a different version using the shared visualization.
4. During the group discussion, it was easy for me to refer to a specific version in the shared visualization.
5. During the group discussion, it was easy for me to refer to a specific object in the scene.
6. During the group discussion, it was easy to understand which version my collaborator was referring to in the shared visualization.

7. During the group discussion, it was easy to understand which object my collaborator was referring to in the scene.

8. It was easy to merge objects from another version using the shared visualization.

9. During the group discussion, I think the shared visualization would be useful.

General Experience

1. My overall communication with my collaborators felt natural.

2. To what extent did you experience that you and your partner collaborated?

3. Think of some previous time (before today) when you enjoyed collaborating with someone. To what extent did you enjoy collaborating with your partner in today's task?

4. To what extent would you, on another occasion, like to carry out a similar task with your partner?

BIBLIOGRAPHY

[1] A-frame, 2015.

[2] Blueprints visual scripting, 2022.

[3] Bond, 2023.

[4] Kasa smart, 2023.

[5] Cycling '74. Max 8, 2018.

[6] Parastoo Abtahi and Sean Follmer. Visuo-haptic illusions for improving the perceived performance of shape displays. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2018.

[7] Open AI. Sora, 2024.

[8] Isabel Allende. *La casa de los espíritus*. Plaza & Janés, 1982.

[9] Amazon. Echo, 2023.

[10] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N Bennett, Kori Inkpen, et al. Guidelines for human-ai interaction. In *Proceedings of the 2019 chi conference on human factors in computing systems*, pages 1–13, 2019.

[11] Judith Amores, Xavier Benavides, and Pattie Maes. Showme: A remote collaboration system that supports immersive gestural communication. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, pages 1343–1348, 2015.

[12] Apache. Subversion, 2022.

[13] Rahul Arora, Rubaiat Habib Kazi, Danny M Kaufman, Wilmot Li, and Karan Singh. Magicalhands: Mid-air hand gestures for animating in vr. In *Proceedings of the 32nd annual ACM symposium on user interface software and technology*, pages 463–477, 2019.

[14] Narges Ashtari, Andrea Bunt, Joanna McGrenere, Michael Nebeling, and Parmit K Chilana. Creating augmented and virtual reality applications: Current practices, challenges, and opportunities. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pages 1–13, 2020.

[15] Jackie Assa, Yaron Caspi, and Daniel Cohen-Or. Action synopsis: pose selection and illustration. *ACM Transactions on Graphics (TOG)*, 24(3):667–676, 2005.

[16] Zhen Bai, Alan F Blackwell, and George Coulouris. Exploring expressive augmented reality: The fingar puppet system for social pretend play. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 1035–1044, 2015.

[17] Engineer Bainomugisha, Andoni Lombide Carreton, Tom van Cutsem, Stijn Mostinckx, and Wolfgang de Meuter. A survey on reactive programming. *ACM Computing Surveys (CSUR)*, 45(4):52, 2013.

[18] Xinlong Bao, Jonathan L. Herlocker, and Thomas G. Dietterich. Fewer clicks and less frustration: Reducing the cost of reaching the right folder. In *Proceedings of the 11th International Conference on Intelligent User Interfaces*, IUI '06, page 178–185, New York, NY, USA, 2006. Association for Computing Machinery.

[19] Connelly Barnes, Dan B Goldman, Eli Shechtman, and Adam Finkelstein. Video tapestries with continuous temporal zoom. In *ACM SIGGRAPH 2010 papers*, pages 1–9. 2010.

[20] Frank Biocca, Chad Harms, and Judee K Burgoon. Toward a more robust theory and measure of social presence: Review and suggested criteria. *Presence: Teleoperators & virtual environments*, 12(5):456–480, 2003.

[21] Blender. Shader editor, 2022.

[22] Richard A Bolt. "put-that-there" voice and gesture at the graphics interface. In *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pages 262–270, 1980.

[23] Evren Bozgeyikli, Andrew Raij, Srinivas Katkoori, and Rajiv Dubey. Point & teleport locomotion technique for virtual reality. In *Proceedings of the 2016 annual symposium on computer-human interaction in play*, pages 205–216, 2016.

[24] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2):77–101, 2006.

[25] Jas Brooks, Noor Amin, and Pedro Lopes. Taste retargeting via chemical taste modulators. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–15, 2023.

[26] Margaret Wise Brown. *Goodnight Moon.* Harper Collins, New York, NY, 50 edition, September 1947.

[27] Amy B Brunell, Michael H Kernis, Brian M Goldman, Whitney Heppner, Patricia Davis, Edward V Cascio, and Gregory D Webster. Dispositional authenticity and romantic relationship functioning. *Personality and Individual Differences*, 48(8):900–905, 2010.

[28] Don Brutzman and Leonard Daly. *X3D: extensible 3D graphics for Web authors.* Elsevier, 2010.

[29] Moira Burke and Robert E. Kraut. The Relationship Between Facebook Use and Well-Being Depends on Communication Type and Tie Strength. *Journal of Computer-Mediated Communication*, 21(4):265–281, 05 2016.

[30] Daniel Buschek, Lukas Mecke, Florian Lehmann, and Hai Dang. Nine potential pitfalls when designing human-ai co-creative systems. *arXiv preprint arXiv:2104.00358*, 2021.

[31] Jeff Butterworth. 3dm: a three-dimensional modeler using a head-mounted display. 1992.

[32] Bill Buxton. *Sketching user experiences: getting the design right and the right design.* Morgan kaufmann, 2010.

[33] Ricardo Cabello. three.js, 2010.

[34] Claudio Calabrese, Gabriele Salvati, Marco Tarini, and Fabio Pellacini. csculpt: a system for collaborative sculpting. *ACM Transactions on Graphics (TOG)*, 35(4):1–8, 2016.

[35] Daniel J Canary and Laura Stafford. Relational maintenance strategies and equity in marriage. *Communications Monographs*, 59(3):243–267, 1992.

[36] Daniel J Canary and Young-Ok Yum. Relationship maintenance strategies. *The international encyclopedia of interpersonal communication*, pages 1–9, 2015.

[37] Rikk Carey and Gavin Bell. *The annotated VRML 2.0 reference manual.* Number BOOK. Addison-Wesley, 1997.

[38] Edoardo Carra and Fabio Pellacini. Scenegit: a practical system for diffing and merging 3d environments. *ACM Transactions on Graphics (TOG)*, 38(6):1–15, 2019.

[39] Siddhartha Chaudhuri, Evangelos Kalogerakis, Leonidas Guibas, and Vladlen Koltun. Probabilistic reasoning for assembly-based 3d modeling. In *ACM SIGGRAPH 2011 papers*, pages 1–10. 2011.

[40] Siddhartha Chaudhuri and Vladlen Koltun. Data-driven suggestions for creativity support in 3d modeling. In *ACM SIGGRAPH Asia 2010 papers*, pages 1–10. 2010.

[41] Hsiang-Ting Chen, Li-Yi Wei, and Chun-Fa Chang. Nonlinear revision control for images. *ACM Transactions on Graphics (TOG)*, 30(4):1–10, 2011.

[42] Mengyu Chen, Andrés Monroy-Hernández, and Misha Sra. Scenear: Scene-based micro narratives for sharing and remixing in augmented reality. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 294–303, 2021.

[43] Tianying Chen, Margot Stewart, Zhiyu Bai, Eileen Chen, Laura Dabbish, and Jessica Hammer. Hacked time: Design and evaluation of a self-efficacy based cybersecurity game. In *Proceedings of the 2020 ACM Designing Interactive Systems Conference*, pages 1737–1749, 2020.

[44] Erin Cherry and Celine Latulipe. Quantifying the creativity support of digital tools through the creativity support index. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 21(4):1–25, 2014.

[45] Kevin Chow, Caitlin Coyiuto, Cuong Nguyen, and Dongwook Yoon. Challenges and design considerations for multimodal asynchronous collaboration in vr. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–24, 2019.

[46] John Joon Young Chung and Eytan Adar. Promptpaint: Steering text-to-image generation through paint medium-like interactions. *arXiv preprint arXiv:2308.05184*, 2023.

[47] John Joon Young Chung, Wooseok Kim, Kang Min Yoo, Hwaran Lee, Eytan Adar, and Minsuk Chang. Talebrush: Sketching stories with generative pretrained language models. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–19, 2022.

[48] Elizabeth Clark, Anne Spencer Ross, Chenhao Tan, Yangfeng Ji, and Noah A Smith. Creative writing with a machine in the loop: Case studies on slogans and stories. In *23rd International Conference on Intelligent User Interfaces*, pages 329–340, 2018.

[49] Herbert H Clark and Susan E Brennan. Grounding in communication. 1991.

[50] Philip R Cohen. The role of natural language in a multimodal interface. In *Proceedings of the 5th annual ACM symposium on User interface software and technology*, pages 143–149, 1992.

[51] Robin George Collingwood. *The principles of art*, volume 11. Oxford University Press, 1958.

[52] Richard Cook. *Connecting the Echo Dots: An Exploratory Ethnographic Study of 'Alexa' in the Classroom.* PhD thesis, 05 2021.

[53] Mojang AB. TM Microsoft Corporation. Minecraft in virtual reality, 2024.

[54] Adam M Croom. Music practice and participation for psychological well-being: A review of how music influences positive emotion, engagement, relationships, meaning, and accomplishment. *Musicae Scientiae*, 19(1):44–64, 2015.

[55] Evan Czaplicki and Stephen N Chong. Asynchronous functional reactive programming for guis. In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation-PLDI'13.* ACM Press, 2013.

[56] Ella Dagan, Ana María Cárdenas Gasca, Ava Robinson, Anwar Noriega, Yu Jiang Tham, Rajan Vaish, and Andrés Monroy-Hernández. Project irl: Playful co-located interactions with mobile augmented reality. *Proceedings of the ACM on Human-Computer Interaction*, 6(CSCW1):1–27, 2022.

[57] Areti Damala, Pierre Cubaud, Anne Bationo, Pascal Houlier, and Isabelle Marchal. Bridging the gap between the digital and the physical: design and evaluation of a mobile augmented reality guide for the museum visit. In *Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts*, pages 120–127, 2008.

[58] DAQRI. Elements 4d interactive blocks, Jan 2014.

[59] Nicholas Davis, Chih-PIn Hsiao, Kunwar Yashraj Singh, Lisa Li, and Brian Magerko. Empirically studying participatory sense-making in abstract drawing with a co-creative cognitive agent. In *Proceedings of the 21st International Conference on Intelligent User Interfaces*, pages 196–207, 2016.

[60] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36, 2024.

[61] Nicola Dell, Vidya Vaidyanathan, Indrani Medhi, Edward Cutrell, and William Thies. " yours is better!" participant response bias in hci. In *Proceedings of the sigchi conference on human factors in computing systems*, pages 1321–1330, 2012.

[62] Jonathan D Denning, William B Kerr, and Fabio Pellacini. Meshflow: interactive visualization of mesh construction sequences. In *ACM SIGGRAPH 2011 papers*, pages 1–8. 2011.

[63] Jonathan D Denning and Fabio Pellacini. Meshgit: Diffing and merging meshes for polygonal modeling. *ACM Transactions on Graphics (TOG)*, 32(4):1–10, 2013.

[64] Jonathan D Denning, Valentina Tibaldo, and Fabio Pellacini. 3dflow: Continuous summarization of mesh editing workflows. *ACM Transactions on Graphics (TOG)*, 34(4):1–9, 2015.

[65] Shipi Dhanorkar, Christine T Wolf, Kun Qian, Anbang Xu, Lucian Popa, and Yunyao Li. Who needs to know what, when?: Broadening the explainable ai (xai) design space by looking at explanations across the ai lifecycle. In *Designing Interactive Systems Conference 2021*, pages 1591–1602, 2021.

[66] Jozef Doboš and Anthony Steed. 3d diff: an interactive approach to mesh differencing and conflict resolution. In *SIGGRAPH Asia 2012 Technical Briefs*, pages 1–4. 2012.

[67] Jozef Doboš and Anthony Steed. 3d revision control framework. In *Proceedings of the 17th International Conference on 3D Web Technology*, pages 121–129, 2012.

[68] Wendy Doubé and Jeanie Beh. Typing over autocomplete: Cognitive load in website use by older adults. In *Proceedings of the 24th Australian Computer-Human Interaction Conference*, OzCHI '12, page 97–106, New York, NY, USA, 2012. Association for Computing Machinery.

[69] Paul Dourish. The parting of the ways: Divergence, data management and collaborative work. In *Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work ECSCW'95*, pages 215–230. Springer, 1995.

[70] Tobias Drey, Jan Gugenheimer, Julian Karlbauer, Maximilian Milo, and Enrico Rukzio. Vrsketchin: Exploring the design space of pen and tablet interaction for 3d sketching in virtual reality. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pages 1–14, 2020.

[71] Thierry Duval, Thi Thuong Huyen Nguyen, Cédric Fleury, Alain Chauffaut, Georges Dumont, and Valérie Gouranton. Improving awareness for 3d virtual collaboration by embedding the features of users' physical environments and by augmenting interaction tools with cognitive feedback cues. *Journal on Multimodal User Interfaces*, 8(2):187–197, 2014.

[72] Conal Elliott and Paul Hudak. Functional reactive animation. In *ACM SIGPLAN Notices*, volume 32, pages 263–273. ACM, 1997.

[73] Carmine Elvezio, Mengu Sukan, and Steven Feiner. Mercury: A messaging framework for modular ui components. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 588. ACM, 2018.

[74] Barrett Ens, Fraser Anderson, Tovi Grossman, Michelle Annett, Pourang Irani, and George Fitzmaurice. Ivy: Exploring spatially situated visual programming for authoring and understanding intelligent environments. In *Proceedings of the 43rd Graphics Interface Conference*, pages 156–162, 2017.

[75] Judith E Fan, Monica Dinculescu, and David Ha. Collabdraw: an environment for collaborative sketching with an artificial agent. In *Proceedings of the 2019 on Creativity and Cognition*, pages 556–561. 2019.

[76] Ben Fehnert and Alessia Kosagowsky. Measuring user experience: Complementing qualitative and quantitative assessment. In *Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '08, page 383–386, New York, NY, USA, 2008. Association for Computing Machinery.

[77] Brian Feldman. The best use of augmented reality right now is the weather channel's, Jan 2019.

[78] Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[79] Ronald Aylmer Fisher. Design of experiments. *British Medical Journal*, 1(3923):554, 1936.

[80] Glenn Fleishman. How facebook devalued the birthday, Apr 2018.

[81] Mike Fraser, Steve Benford, Jon Hindmarsh, and Christian Heath. Supporting awareness and interaction through collaborative virtual interfaces. In *Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 27–36, 1999.

[82] Daniel Freeman, Sarah Reeve, Abi Robinson, Anke Ehlers, David Clark, Bernhard Spanlang, and Mel Slater. Virtual reality in the assessment, understanding, and treatment of mental health disorders. *Psychological medicine*, 47(14):2393–2400, 2017.

[83] Francielle Frizzo, Helison Bertoli Alves Dias, Nayara Pereira Duarte, Denise Gabriela Rodrigues, and Paulo Henrique Muller Prado. The genuine handmade: How the production method influences consumers' behavioral intentions through naturalness and authenticity. *Journal of Food Products Marketing*, 26(4):279–296, 2020.

[84] Peter Frost and Peter Warren. Virtual reality used in a collaborative architectural design process. In *2000 IEEE Conference on Information Visualization. An International Conference on Computer Visualization and Graphics*, pages 568–573. IEEE, 2000.

[85] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10933–10942, 2021.

[86] Christoph Fuchs, Martin Schreier, and Stijn MJ Van Osselaer. The handmade effect: What's love got to do with it? *Journal of marketing*, 79(2):98–110, 2015.

[87] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. *Advances In Neural Information Processing Systems*, 35:31841–31854, 2022.

[88] William W Gaver, Abigail Sellen, Christian Heath, and Paul Luff. One is not enough: Multiple views in a media space. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, pages 335–341, 1993.

[89] Git. Git, 2022.

[90] Terrell Glenn, Ananya Ipsita, Caleb Carithers, Kylie Peppler, and Karthik Ramani. Storymakar: Bringing stories to life with an augmented reality & physical prototyping toolkit for youth. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2020.

[91] Kristine Gloria. Lessons in loneliness. Technical report, Facebook and Aspen Institute, 2020.

[92] Ontroerend Goed. 'the smile off your face (2004)' - projects: Ontroerend goed, 2004.

[93] Ontroerend Goed. 'internal (2007)' - projects: Ontroerend goed, 2007.

[94] Ontroerend Goed. 'a game of you' - projects: Ontroerend goed, 2010.

[95] Google. Google tilt brush, 2016.

[96] Digital Graffiti. Alys beach – where immersive technology meets art and architecture., 2022.

[97] Saul Greenberg. Toolkits and interface creativity. *Multimedia Tools and Applications*, 32:139–159, 2007.

[98] Carla Griggio, Arissa Sato, Wendy Mackay, and Koji Yatani. Mediating intimacy with dearboard: a co-customizable keyboard for everyday messaging. 05 2021.

[99] Carla F Griggio, Joanna Mcgrenere, and Wendy E Mackay. Customizations and expression breakdowns in ecosystems of communication apps. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–26, 2019.

[100] Carla F Griggio, Arissa J Sato, Wendy E Mackay, and Koji Yatani. Mediating intimacy with dearboard: a co-customizable keyboard for everyday messaging. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–16, 2021.

[101] Tovi Grossman, Justin Matejka, and George Fitzmaurice. Chronicle: capture, exploration, and playback of document workflow histories. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 143–152, 2010.

[102] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

[103] Anhong Guo, Ilter Canberk, Hannah Murphy, Andrés Monroy-Hernández, and Rajan Vaish. Blocks: Collaborative and persistent augmented reality experiences. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(3):1–24, 2019.

[104] Carl Gutwin and Saul Greenberg. A descriptive framework of workspace awareness for real-time groupware. *Computer Supported Cooperative Work (CSCW)*, 11(3):411–446, 2002.

[105] Matthew Guzdial, Nicholas Liao, Jonathan Chen, Shao-Yu Chen, Shukan Shah, Vishwa Shah, Joshua Reno, Gillian Smith, and Mark O Riedl. Friend, collaborator, student, manager: How design of an ai-driven game level editor affects creators. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pages 1–13, 2019.

[106] Blake Hallinan. Like: The informatization of affect. In *Companion of the 2018 ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '18, page 73–76, New York, NY, USA, 2018. Association for Computing Machinery.

[107] Ant Hampton and Silvia Mercuriali. Autoteatro, 2020.

[108] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988.

[109] Björn Hartmann, Loren Yu, Abel Allison, Yeonsoo Yang, and Scott R Klemmer. Design as exploration: creating interface alternatives through parallel authoring and runtime tuning. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 91–100, 2008.

[110] Devamardeep Hayatpur, Seongkook Heo, Haijun Xia, Wolfgang Stuerzlinger, and Daniel Wigdor. Plane, ray, and point: Enabling precise spatial manipulations with shape constraints. In *Proceedings of the 32nd annual ACM symposium on user interface software and technology*, pages 1185–1195, 2019.

[111] Zhenyi He, Ruofei Du, and Ken Perlin. Collabovr: A reconfigurable framework for creative collaboration in virtual reality. In *2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 542–554. IEEE, 2020.

[112] Jennifer Healey, Duotun Wang, Curtis Wigington, Tong Sun, and Huaishu Peng. A mixed-reality system to promote child engagement in remote intergenerational storytelling. In *2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 274–279, 2021.

[113] Juan Luis Higuera-Trujillo, Juan López-Tarruella Maldonado, and Carmen Llinares Millán. Psychological and physiological human responses to simulated and real environments: A comparison between photographs, 360 panoramas, and virtual reality. *Applied ergonomics*, 65:398–409, 2017.

[114] Daniel D Hils. Datavis: a visual programming language for scientific visualization. In *Proceedings of the 19th annual conference on Computer Science*, pages 439–448. ACM, 1991.

[115] McKittrick Hotel. *Sleep No More.* 2011.

[116] Sara H Hsieh and Timmy H Tseng. Playfulness in mobile instant messaging: Examining the influence of emoticons and text messaging on social interaction. *Computers in Human Behavior*, 69:405–414, 2017.

[117] Cheng-Zhi Anna Huang, Curtis Hawthorne, Adam Roberts, Monica Dinculescu, James Wexler, Leon Hong, and Jacob Howcroft. The bach doodle: Approachable music composition with machine learning at scale. *arXiv preprint arXiv:1907.06637*, 2019.

[118] Layne Jackson Hubbard, Yifan Chen, Eliana Colunga, Pilyoung Kim, and Tom Yeh. Child-robot interaction to integrate reflective storytelling into creative play. In *Creativity and Cognition*, C&C '21, New York, NY, USA, 2021. Association for Computing Machinery.

[119] Hikaru Ibayashi, Yuta Sugiura, Daisuke Sakamoto, Natsuki Miyata, Mitsunori Tada, Takashi Okuma, Takeshi Kurata, Masaaki Mochimaru, and Takeo Igarashi. Dollhouse vr: a multi-view, multi-user collaborative design workspace with vr technology. In *SIGGRAPH Asia 2015 emerging technologies*, pages 1–2. 2015.

[120] Claudia-Lavinia Ignat, Stavroula Papadopoulou, Gérald Oster, and Moira C Norrie. Providing awareness in multi-synchronous collaboration without compromising privacy. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, pages 659–668, 2008.

[121] Snap Inc. Lens studio, 2022.

[122] Within Unlimited Inc. Wonderscope, 2018.

[123] National Instruments. Labview, 2017.

[124] Bret Jackson and Daniel F Keefe. Lift-off: Using reference imagery and freehand sketching to create 3d models in vr. *IEEE transactions on visualization and computer graphics*, 22(4):1442–1451, 2016.

[125] Maurice Jakesch, Megan French, Xiao Ma, Jeffrey T Hancock, and Mor Naaman. Ai-mediated communication: How the perception that profile text was written by ai affects trustworthiness. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2019.

[126] David G. Jansson and Steven M. Smith. Design fixation. *Design Studies*, 12(1):3–11, 1991.

[127] Brett Jones, Rajinder Sodhi, Michael Murdock, Ravish Mehra, Hrvoje Benko, Andrew Wilson, Eyal Ofek, Blair MacIntyre, Nikunj Raghuvanshi, and Lior Shapira. Roomalive: Magical experiences enabled by scalable, adaptive projector-camera units. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, page 637–644, New York, NY, USA, 2014. Association for Computing Machinery.

[128] Roy S Kalawsky and Roy S Kalawsky. *The science of virtual reality and virtual environments: a technical, scientific and engineering reference on virtual environments.* Addison-wesley Workingham, 1993.

[129] Geoff Kaufman and Mary Flanagan. A psychologically "embedded" approach to designing games for prosocial causes. *Cyberpsychology: Journal of Psychosocial Research on Cyberspace*, 9(3), 2015.

[130] Rubaiat Habib Kazi, Tovi Grossman, Hyunmin Cheong, Ali Hashemi, and George W Fitzmaurice. Dreamsketch: Early stage 3d design explorations with sketching and generative design. In *UIST*, volume 14, pages 401–414, 2017.

[131] Annie Kelly, R Benjamin Shapiro, Jonathan de Halleux, and Thomas Ball. Arcadia: A rapid prototyping platform for real-time tangible interfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 409. ACM, 2018.

[132] Ryan Kelly, Daniel Gooch, Bhagyashree Patil, and Leon Watts. Demanding by design: Supporting effortful communication practices in close personal relationships. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, pages 70–83, 2017.

[133] Ryan Kelly, Daniel Gooch, and Leon Watts. Designing for reflection on sender effort in close personal communication. In *Proceedings of the 30th Australian Conference on Computer-Human Interaction*, pages 314–325, 2018.

[134] Ryan Kelly, Daniel Gooch, and Leon Watts. 'it's more like a letter' an exploration of mediated conversational effort in message builder. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):1–23, 2018.

[135] Taewook Kim, Jung Lee, Zhenhui Peng, and Xiaojuan Ma. Love in lyrics: An exploration of supporting textual manifestation of affection in social messaging. *Proceedings of the ACM on Human-Computer Interaction*, 3:1–27, 11 2019.

[136] Takayuki Dan Kimura, Julie W Choi, and Jane M Mack. A visual language for keyboardless programming. 1986.

[137] Simon King and Jodi Forlizzi. Slow messaging: intimate communication for couples living at a distance. In *Proceedings of the 2007 conference on Designing pleasurable products and interfaces*, pages 451–454, 2007.

[138] Tobias Klein. Digital craftsmanship. pages 643–654, 08 2015.

[139] Scott R Klemmer, Michael Thomsen, Ethan Phelps-Goodman, Robert Lee, and James A Landay. Where do web sites come from? capturing and interacting with design history. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1–8, 2002.

[140] Boriana Koleva, Jocelyn Spence, Steve Benford, Hyosun Kwon, Holger Schnädelbach, Emily Thorn, William Preston, Adrian Hazzard, Chris Greenhalgh, Matt Adams, et al. Designing hybrid gifts. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 27(5):1–33, 2020.

[141] Ryohei Komiyama, Takashi Miyaki, and Jun Rekimoto. Jackin space: designing a seamless transition between first and third person view for effective telepresence collaborations. In *Proceedings of the 8th Augmented Human International Conference*, pages 1–9, 2017.

[142] Jacqueline Kory and Cynthia Breazeal. Storytelling with robots: Learning companions for preschool children's language development. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 643–648, 2014.

[143] Veronika Krauß, Alexander Boden, Leif Oppermann, and René Reiners. Current practices, challenges, and design implications for collaborative ar/vr application development. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–15, 2021.

[144] Andrey Krekhov, Sebastian Cmentowski, Katharina Emmerich, Maic Masuch, and Jens Krüger. Gullivr: A walking-oriented technique for navigation in virtual reality games based on virtual body resizing. In *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play*, pages 243–256, 2018.

[145] Eleni Kroupi, Philippe Hanhart, Jong-Seok Lee, Martin Rerabek, and Touradj Ebrahimi. Predicting subjective sensation of reality during multimedia consumption based on eeg and peripheral physiological signals. In *2014 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2014.

[146] André Kunert, Alexander Kulik, Stephan Beck, and Bernd Froehlich. Photoportals: shared references in space and time. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, pages 1388–1399, 2014.

[147] David Kurlander and Steven Feiner. Editable graphical histories. In *VL*, pages 127–134. Citeseer, 1988.

[148] Hyosun Kwon, Boriana Koleva, Holger Schnädelbach, and Steve Benford. " it's not yet a gift" understanding digital gifting. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, pages 2372–2384, 2017.

[149] David Lau-Kee, Adam Billyard, Robin Faichney, Yasuo Kozato, Paul Otto, Mark Smith, and Ian Wilkinson. Vpl: an active, declarative visual programming system. In *Proceedings 1991 IEEE Workshop on Visual Languages*, pages 40–46. IEEE, 1991.

[150] David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg, and Saul Greenberg. Evaluation strategies for hci toolkit research. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–17, 2018.

[151] Gun A Lee and Gerard J Kim. Immersive authoring of tangible augmented reality content: A user study. *Journal of Visual Languages & Computing*, 20(2):61–79, 2009.

[152] Gun A Lee, Gerard J Kim, and Mark Billinghurst. Immersive authoring: What you experience is what you get (wyxiwyg). *Communications of the ACM*, 48(7):76–81, 2005.

[153] Gun A Lee, Claudia Nelles, Mark Billinghurst, and Gerard Jounghyun Kim. Immersive authoring of tangible augmented reality applications. In *Third IEEE and ACM international symposium on mixed and augmented reality*, pages 172–181. IEEE, 2004.

[154] Germán Leiva, Jens Emil Grønbæk, Clemens Nylandsted Klokmose, Cuong Nguyen, Rubaiat Habib Kazi, and Paul Asente. Rapido: Prototyping interactive ar experiences through programming by demonstration. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 626–637, 2021.

[155] Germán Leiva, Cuong Nguyen, Rubaiat Habib Kazi, and Paul Asente. Pronto: Rapid augmented reality video prototyping using sketches and enaction. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2020.

[156] Jingyi Li, Wilmot Li, Sean Follmer, and Maneesh Agrawala. Automated accessory rigs for layered 2d character illustrations. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 1100–1108, 2021.

[157] Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao Zhang. Grains: Generative recursive autoencoders for indoor scenes. *ACM Transactions on Graphics (TOG)*, 38(2):1–16, 2019.

[158] Q Vera Liao, Hariharan Subramonyam, Jennifer Wang, and Jennifer Wortman Vaughan. Designerly understanding: Information needs for model transparency to support design ideation for ai-powered user experience. In *Proceedings of the 2023 CHI conference on human factors in computing systems*, pages 1–21, 2023.

[159] Klemen Lilija, Henning Pohl, and Kasper Hornbæk. Who put that there? temporal navigation of spatial recordings by direct manipulation. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–11, 2020.

[160] Yi-Bing Lin, Ming-Ta Yang, and Yun-Wei Lin. Low-cost four-dimensional experience theater using home appliances. *IEEE Transactions on Multimedia*, 21(5):1161–1168, 2019.

[161] Siân E Lindley, Richard Harper, and Abigail Sellen. Desiring to be in touch in a changing communications landscape: attitudes of older adults. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1693–1702, 2009.

[162] Eden Litt, Siyan Zhao, Robert Kraut, and Moira Burke. What are meaningful social interactions in today's media landscape? a cross-cultural survey. *Social Media+ Society*, 6(3):2056305120942888, 2020.

[163] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9298–9309, 2023.

[164] Ryan Louie, Andy Coenen, Cheng Zhi Huang, Michael Terry, and Carrie J Cai. Novice-ai music co-creation via ai-steering tools for deep generative models. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pages 1–13, 2020.

[165] Kai Lukoff, Ulrik Lyngs, Himanshu Zade, J Vera Liao, James Choi, Kaiyue Fan, Sean A Munson, and Alexis Hiniker. How the design of youtube influences user sense of agency. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–17, 2021.

[166] Marketta Luutonen et al. Handmade memories. *Trames*, 12(3):331–341, 2008.

[167] Rui Ma, Akshay Gadi Patil, Matthew Fisher, Manyi Li, Sören Pirk, Binh-Son Hua, Sai-Kit Yeung, Xin Tong, Leonidas Guibas, and Hao Zhang. Language-driven synthesis of 3d scenes from scene databases. *ACM Transactions on Graphics (TOG)*, 37(6):1–16, 2018.

[168] Jock D Mackinlay, Stuart K Card, and George G Robertson. Rapid controlled movement through a virtual 3d workspace. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 171–176, 1990.

[169] Panos Markopoulos. A design framework for awareness systems. In *Awareness systems*, pages 49–72. Springer, 2009.

[170] Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. *ACM transactions on graphics (TOG)*, 30(4):1–10, 2011.

[171] Microsoft. Microsoft maquette, 2018.

[172] Mark Mine. Isaac: A virtual environment tool for the interactive construction of virtual worlds. 1995.

[173] Mark Mine, Arun Yoganandan, and Dane Coffey. Making vr work: building a real-world immersive modeling application in the virtual world. In *Proceedings of the 2nd ACM symposium on Spatial user interaction*, pages 80–89, 2014.

[174] Yuko Minowa and Stephen J Gould. Love my gift, love me or is it love me, love my gift: A study of the cultural construction of romantic gift giving among japanese couples. *ACR North American Advances*, 1999.

[175] Beatrice Monastero and David K McGookin. Traces: Studying a public reactive floor-projection of walking trajectories to support social awareness. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2018.

[176] Andrés Monroy-Hernández, Benjamin Mako Hill, Jazmin Gonzalez-Rivero, and Danah Boyd. Computers can't give credit: How automatic attribution falls short in an online remixing community. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3421–3430, 2011.

[177] Janet H Murray. *Hamlet on the Holodeck, updated edition: The Future of Narrative in Cyberspace*. MIT press, 2017.

[178] Brad A Myers, Ashley Lai, Tam Minh Le, YoungSeok Yoon, Andrew Faulring, and Joel Brandt. Selective undo support for painting applications. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 4227–4236, 2015.

[179] Gabriel García Márquez. *Chronicle of a Death Foretold*. La Oveja Negra, 1981.

[180] Marc A Najork and Eric Golin. Enhancing show-and-tell with a polymorphic type system and higher-order functions. In *Proceedings of the 1990 IEEE Workshop on Visual Languages*, pages 215–220. IEEE, 1990.

[181] Toshio Nakamura and Takeo Igarashi. An application-independent system for visualizing user operation history. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 23–32, 2008.

[182] Michael Nebeling, Katy Lewis, Yu-Cheng Chang, Lihan Zhu, Michelle Chung, Piaoyang Wang, and Janet Nebeling. Xrdirector: A role-based collaborative immersive authoring system. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2020.

[183] Michael Nebeling and Katy Madier. 360proto: Making interactive virtual reality & augmented reality prototypes from paper. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 596. ACM, 2019.

[184] Michael Nebeling, Janet Nebeling, Ao Yu, and Rob Rumble. Protoar: Rapid physical-digital prototyping of mobile augmented reality applications. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2018.

[185] Michael Nebeling and Maximilian Speicher. The trouble with augmented reality/virtual reality authoring tools. In *2018 IEEE international symposium on mixed and augmented reality adjunct (ISMAR-Adjunct)*, pages 333–337. IEEE, 2018.

[186] Cuong Nguyen, Stephen DiVerdi, Aaron Hertzmann, and Feng Liu. Collavr: collaborative in-headset review for vr video. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pages 267–277, 2017.

[187] Donald A Norman and Stephen W Draper. *User centered system design: New perspectives on human-computer interaction*. CRC Press, 1986.

[188] Oculus. Oculus medium, 2016.

[189] Ohan Oda, Carmine Elvezio, Mengu Sukan, Steven Feiner, and Barbara Tversky. Virtual replicas for remote assistance in virtual and augmented reality. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 405–415, 2015.

[190] Luke Olsen, Faramarz F Samavati, Mario Costa Sousa, and Joaquim A Jorge. Sketch-based modeling: A survey. *Computers & Graphics*, 33(1):85–103, 2009.

[191] Logan Olson. Soundstage vr, 2018.

[192] Alex Olwal and Artem Dementyev. Hidden interfaces for ambient computing: Enabling interaction in everyday materials through high-brightness visuals on low-cost matrix displays. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, New York, NY, USA, 2022. Association for Computing Machinery.

[193] OpenAI. Gpt4, 2023.

[194] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, et al. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th annual symposium on user interface software and technology*, pages 741–754, 2016.

[195] Yun Suen Pai, Benjamin I Outram, Benjamin Tag, Megumi Isogai, Daisuke Ochi, Hideaki Kimata, and Kai Kunze. Cleavr: collaborative layout evaluation and assessment in virtual reality. In *ACM SIGGRAPH 2017 Posters*, pages 1–2. 2017.

[196] Disneyland Paris. Star tours: The adventures continue, 2022.

[197] Disneyland Park. Space mountain, 2022.

[198] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. *Advances in Neural Information Processing Systems*, 34:12013–12026, 2021.

[199] Navinchandra K Patel, Simon P Campion, and Terrence Fernando. Evaluating the use of virtual reality as a tool for briefing clients in architecture. In *Proceedings Sixth International Conference on Information Visualisation*, pages 657–663. IEEE, 2002.

[200] Randy Pausch, Tommy Burnette, AC Capeheart, Matthew Conway, Dennis Cosgrove, Rob DeLine, Jim Durbin, Rich Gossweiler, Shuichi Koga, and Jeff White. Alice: Rapid prototyping system for virtual reality. *IEEE Computer Graphics and Applications*, 15(3):8–11, 1995.

[201] Tomislav Pejsa, Julian Kantor, Hrvoje Benko, Eyal Ofek, and Andrew Wilson. Room2room: Enabling life-size telepresence in a projected augmented reality environment. In *Proceedings of the 19th ACM conference on computer-supported cooperative work & social computing*, pages 1716–1725, 2016.

[202] Tomislav Pejsa, Julian Kantor, Hrvoje Benko, Eyal Ofek, and Andy Wilson. Room2room: Enabling life-size telepresence in a projected augmented reality environment. In *CSCW '16 Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, pages 1716–1725. ACM, February 2016.

[203] Jeffrey S Pierce, Brian C Stearns, and Randy Pausch. Voodoo dolls: seamless interaction at multiple scales in virtual environments. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 141–145, 1999.

[204] Thammathip Piumsomboon, Gun A Lee, Jonathon D Hart, Barrett Ens, Robert W Lindeman, Bruce H Thomas, and Mark Billinghurst. Mini-me: An adaptive avatar for mixed reality remote collaboration. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pages 1–13, 2018.

[205] Thammathip Piumsomboon, Youngho Lee, Gun Lee, and Mark Billinghurst. Covar: a collaborative virtual and augmented reality system for remote collaboration. In *SIGGRAPH Asia 2017 Emerging Technologies*, pages 1–2. 2017.

[206] Kevin Ponto, Ross Tredinnick, Aaron Bartholomew, Carrie Roy, Dan Szafir, Daniel Greenheck, and Joe Kohlmann. Sculptup: A rapid, immersive 3d modeling environment. In *2013 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 199–200. IEEE, 2013.

[207] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. 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*, pages 79–80, 1996.

[208] Jaziar Radianti, Tim A Majchrzak, Jennifer Fromm, and Isabell Wohlgenannt. A systematic review of immersive virtual reality applications for higher education: Design elements, lessons learned, and research agenda. *Computers & education*, 147:103778, 2020.

[209] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1060–1069, New York, New York, USA, 20–22 Jun 2016. PMLR.

[210] Jun Rekimoto. Time-machine computing: a time-centric approach for the information environment. In *Proceedings of the 12th annual ACM symposium on User interface software and technology*, pages 45–54, 1999.

[211] Joanna Resnick, Ruth Saxton, Matt Angove, and Alan Lane. Anthology, 2010.

[212] Joanna Resnick, Ruth Saxton, Matt Angove, and Alan Lane. They only come at night: Resurrection, 2012.

[213] Mitchel Resnick, Brad Myers, Kumiyo Nakakoji, Ben Shneiderman, Randy Pausch, Ted Selker, and Mike Eisenberg. Design principles for tools to support creative thinking. 2005.

[214] Marc Rettig. Prototyping for tiny fingers. *Communications of the ACM*, 37(4):21–27, 1994.

[215] Yann Riche, Nathalie Henry Riche, Petra Isenberg, and Anastasia Bezerianos. Hard-to-use interfaces considered beneficial (some of the time). In *CHI'10 Extended Abstracts on Human Factors in Computing Systems*, pages 2705–2714. 2010.

[216] Daniel Ritchie, Kai Wang, and Yu-an Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6182–6190, 2019.

[217] Daniela K Rosner and Kimiko Ryokai. Spyn: augmenting the creative and communicative potential of craft. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 2407–2416, 2010.

[218] RxJS. Rxjs, 2015.

[219] Gabriele Salvati, Christian Santoni, Valentina Tibaldo, and Fabio Pellacini. Meshhisto: Collaborative modeling by sharing and retargeting editing histories. *ACM Transactions on Graphics (TOG)*, 34(6):1–10, 2015.

[220] Nazmus Saquib, Rubaiat Habib Kazi, Li-Yi Wei, and Wilmot Li. Interactive body-driven graphics for augmented video performance. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2019.

[221] Arvind Satyanarayan, Ryan Russell, Jane Hoffswell, and Jeffrey Heer. Reactive vega: A streaming dataflow architecture for declarative interactive visualization. *IEEE transactions on visualization and computer graphics*, 22(1):659–668, 2015.

[222] Joachim Scholz and Andrew N. Smith. Augmented reality: Designing immersive experiences that maximize consumer engagement. *Business Horizons*, 59(2):149–161, 2016.

[223] Udo Schultheis, Jason Jerald, Fernando Toledo, Arun Yoganandan, and Paul Mlyniec. Comparison of a two-handed interface to a wand interface and a mouse interface for fundamental 3d tasks. In *2012 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 117–124. IEEE, 2012.

[224] Lauren Scissors, Moira Burke, and Steven Wengrovitz. What's in a like? attitudes and behaviors around receiving likes on facebook. In *Proceedings of the 19th acm conference on computer-supported cooperative work & social computing*, pages 1501–1510, 2016.

[225] Ben Shneiderman. Direct manipulation: A step beyond programming languages. In *ACM SIGSOC Bulletin*, volume 13, page 143. ACM, 1981.

[226] Ben Shneiderman. Creativity support tools: Accelerating discovery and innovation. *Communications of the ACM*, 50(12):20–32, 2007.

[227] Massimiliano Siccardi. Van gogh exhibit los angeles: The immersive experience, Jun 2022.

[228] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[229] Mel Slater and Sylvia Wilbur. A framework for immersive virtual environments (five): Speculations on the role of presence in virtual environments. *Presence: Teleoperators & Virtual Environments*, 6(6):603–616, 1997.

[230] Carolyn Snyder. *Paper prototyping: The fast and easy way to design and refine user interfaces*. Morgan Kaufmann, 2003.

[231] Katherine W Song, Janaki Vivrekar, Lynn Yeom, Eric Paulos, and Niloufar Salehi. Crank that feed: A physical intervention for active twitter users. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–6, 2021.

[232] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1746–1754, 2017.

[233] Tiago Boldt Sousa. Dataflow programming concept, languages and applications. In *Doctoral Symposium on Informatics Engineering*, volume 130, 2012.

[234] Maximilian Speicher, Brian D Hall, Ao Yu, Bowen Zhang, Haihua Zhang, Janet Nebeling, and Michael Nebeling. Xd-ar: challenges and opportunities in cross-device augmented reality application development. *Proceedings of the ACM on Human-Computer Interaction*, 2(EICS):1–24, 2018.

[235] Jocelyn Spence. Inalienability: Understanding digital gifts. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2019.

[236] Steven Spielberg. Ready player one, 2018.

[237] Erin Spottswood and Donghee Yvette Wohn. Beyond the "like": How people respond to negative posts on facebook. *Journal of broadcasting & electronic media*, 63(2):250–267, 2019.

[238] Laura Stafford and Daniel J Canary. Maintenance strategies and romantic relationship type, gender and relational characteristics. *Journal of Social and Personal relationships*, 8(2):217–242, 1991.

[239] Anthony Steed and Mel Slater. A dataflow representation for defining behaviours within virtual environments. In *Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium*, pages 163–167. IEEE, 1996.

[240] Neal Stephenson. *Snow crash*. Penguin UK, 1994.

[241] Sarah Sterman, Molly Jane Nicholas, and Eric Paulos. Towards creative version control. In *CM Conference on Computer-Supported Cooperative Work and Social Computing*, 2022.

[242] Richard Stoakley, Matthew J Conway, and Randy Pausch. Virtual reality on a wim: interactive worlds in miniature. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 265–272, 1995.

[243] A Strauss and J Corbin. Basics of qualitative research techniques. 1998.

[244] Satomi Sugiyama. Kawaii meiru and maroyaka neko: Mobile emoji for relationship maintenance and aesthetic expressions among japanese teens. *First Monday*, 2015.

[245] Kaj Sunesson, Carl Martin Allwood, Dan Paulin, Ilona Heldal, Mattias Roupé, Mikael Johansson, and Börje Westerdahl. Virtual reality as a new tool in the city planning process. *Tsinghua Science & Technology*, 13:255–260, 2008.

[246] Minhyuk Sung, Hao Su, Vladimir G Kim, Siddhartha Chaudhuri, and Leonidas Guibas. Complementme: Weakly-supervised component suggestions for 3d modeling. *ACM Transactions on Graphics (TOG)*, 36(6):1–12, 2017.

[247] Hemant Bhaskar Surale, Aakar Gupta, Mark Hancock, and Daniel Vogel. 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*, pages 1–13, 2019.

[248] William Robert Sutherland. *The on-line graphical specification of computer procedures.* PhD thesis, Massachusetts Institute of Technology, 1966.

[249] Ryo Suzuki, Rubaiat Habib Kazi, Li-Yi Wei, Stephen DiVerdi, Wilmot Li, and Daniel Leithinger. Realitysketch: Embedding responsive graphics and visualizations in ar through dynamic sketching. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pages 166–181, 2020.

[250] Amanda Swearngin, Chenglong Wang, Alannah Oleson, James Fogarty, and Amy J Ko. Scout: Rapid exploration of interface layout alternatives through high-level design constraints. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2020.

[251] Cristina Sylla. Designing a tangible interface for collaborative storytelling to access 'embodiment' and meaning making. In *Proceedings of the 12th International Conference on Interaction Design and Children*, IDC '13, page 651–654, New York, NY, USA, 2013. Association for Computing Machinery.

[252] James Tam and Saul Greenberg. A framework for asynchronous change awareness in collaboratively-constructed documents. In *International Conference on Collaboration and Technology*, pages 67–83. Springer, 2004.

[253] Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20507–20518, 2024.

[254] Steven L Tanimoto. A perspective on the evolution of live programming. In *Proceedings of the 1st International Workshop on Live Programming*, pages 31–34. IEEE Press, 2013.

[255] Adam Tapal, Ela Oren, Reuven Dar, and Baruch Eitam. The sense of agency scale: A measure of consciously perceived control over one's mind, body, and the immediate environment. *Frontiers in psychology*, 8:1552, 2017.

[256] teamLab. Sketch ocean, 2021.

[257] Michael Terry and Elizabeth D Mynatt. Recognizing creative needs in user interface design. In *Proceedings of the 4th Conference on Creativity & Cognition*, pages 38–44, 2002.

[258] Balasaravanan Thoravi Kumaravel, Fraser Anderson, George Fitzmaurice, Bjoern Hartmann, and Tovi Grossman. Loki: Facilitating remote instruction of physical tasks using bi-directional mixed-reality telepresence. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, pages 161–174, 2019.

[259] Walter F Tichy. Rcs—a system for version control. *Software: Practice and Experience*, 15(7):637–654, 1985.

[260] Unity. Unity, 2005.

[261] Unreal. Unreal editor vr mode, 2022.

[262] Nick Vega. I just lost a 159-day snapchat streak and i couldn't be happier, Aug 2017.

[263] Zhanyong Wan, Walid Taha, and Paul Hudak. Event-driven frp. In *International Symposium on Practical Aspects of Declarative Languages*, pages 155–172. Springer, 2002.

[264] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X Chang, and Daniel Ritchie. Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Transactions on Graphics (TOG)*, 38(4):1–15, 2019.

[265] Kai Wang, Manolis Savva, Angel X Chang, and Daniel Ritchie. Deep convolutional priors for indoor scene synthesis. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[266] Tianyi Wang, Xun Qian, Fengming He, Xiyun Hu, Yuanzhi Cao, and Karthik Ramani. Gesturar: An authoring system for creating freehand interactive augmented reality applications. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 552–567, 2021.

[267] Tianyi Wang, Xun Qian, Fengming He, Xiyun Hu, Ke Huo, Yuanzhi Cao, and Karthik Ramani. Capturar: An augmented reality tool for authoring human-involved context-aware applications. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pages 328–341, 2020.

[268] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. Sceneformer: Indoor scene generation with transformers. In *2021 International Conference on 3D Vision (3DV)*, pages 106–115. IEEE, 2021.

[269] Qiuhong Anna Wei, Sijie Ding, Jeong Joon Park, Rahul Sajnani, Adrien Poulenard, Srinath Sridhar, and Leonidas Guibas. Lego-net: Learning regular rearrangements of objects in rooms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19037–19047, 2023.

[270] David West, Aaron Quigley, and Judy Kay. Memento: A digital-physical scrapbook for memory sharing. *Personal Ubiquitous Comput.*, 11(4):313–328, apr 2007.

[271] GARETH WHITE. On immersive theatre. *Theatre Research International*, 37(3):221–235, 2012.

[272] George Wilson and Samuel Shpall. Action. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Winter 2016 edition, 2016.

[273] George Wilson and Samuel Shpall. The nature of action and agency. *Stanford Encyclopedia of Philosophy*, 2016.

[274] Donghee Yvette Wohn, Caleb T Carr, and Rebecca A Hayes. How affective is a "like"?: The effect of paralinguistic digital affordances on perceived social support. *Cyberpsychology, Behavior, and Social Networking*, 19(9):562–566, 2016.

[275] Mary Finley Wolfinbarger. Motivations and symbolism in gift-giving behavior. *ACR North American Advances*, 1990.

[276] Nelson Wong and Carl Gutwin. Where are you pointing? the accuracy of deictic pointing in cves. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1029–1038, 2010.

[277] Haijun Xia, Bruno Araujo, Tovi Grossman, and Daniel Wigdor. Object-oriented drawing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 4610–4621, 2016.

[278] Haijun Xia, Sebastian Herscher, Ken Perlin, and Daniel Wigdor. Spacetime: Enabling fluid individual and collaborative editing in virtual reality. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, pages 853–866, 2018.

[279] Baltasar Trancón y Widemann and Markus Lepper. Foundations of total functional data-flow programming. In *MSFP*, pages 143–167, 2014.

[280] Hui Ye, Jiaye Leng, Pengfei Xu, Karan Singh, and Hongbo Fu. Prointerar: A visual programming platform for creating immersive ar interactions. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–15, 2024.

[281] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[282] Lei Zhang, Ashutosh Agrawal, Steve Oney, and Anhong Guo. Vrgit: A version control system for collaborative content creation in virtual reality. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2023.

[283] Lei Zhang, Tianying Chen, Olivia Seow, Tim Chong, Sven Kratz, Yu Jiang Tham, Andrés Monroy-Hernández, Rajan Vaish, and Fannie Liu. Auggie: Encouraging effortful communication through handcrafted digital experiences. *Proceedings of the ACM on Human-Computer Interaction*, 6(CSCW2):1–25, 2022.

[284] Lei Zhang, Daekun Kim, Youjean Cho, Ava Robinson, Yu Jiang Tham, Rajan Vaish, and Andrés Monroy-Hernández. Jigsaw: Authoring immersive storytelling experiences with augmented reality and internet of things. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2024.

[285] Lei Zhang and Steve Oney. Studying the benefits and challenges of immersive dataflow programming. In *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 223–227. IEEE, 2019.

[286] Lei Zhang and Steve Oney. Flowmatic: An immersive authoring tool for creating interactive scenes in virtual reality. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pages 342–353, 2020.

[287] Lei Zhang, Jin Pan, Jacob Gettig, Steve Oney, and Anhong Guo. Vrcopilot: Authoring 3d layouts with generative ai models in vr. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–14, 2024.

[288] Song-Hai Zhang, Shao-Kui Zhang, Yuan Liang, and Peter Hall. A survey of 3d indoor scene synthesis. *Journal of Computer Science and Technology*, 34:594–608, 2019.

[289] Zheng Zhang, Ying Xu, Yanhao Wang, Bingsheng Yao, Daniel Ritchie, Tongshuang Wu, Mo Yu, Dakuo Wang, and Toby Jia-Jun 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*, CHI '22, New York, NY, USA, 2022. Association for Computing Machinery.

[290] Zhengzhe Zhu, Ziyi Liu, Youyou Zhang, Lijun Zhu, Joey Huang, Ana M Villanueva, Xun Qian, Kylie Peppler, and Karthik Ramani. Learniotvr: An end-to-end virtual reality environment providing authentic learning experiences for internet of things. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–17, 2023.

[291] Fabio Zünd, Mattia Ryffel, Stéphane Magnenat, Alessia Marra, Maurizio Nitti, Mubbasir Kapadia, Gioacchino Noris, Kenny Mitchell, Markus Gross, and Robert W Sumner. Augmented creativity: Bridging the real and virtual worlds to enhance creative play. In *SIGGRAPH Asia 2015 Mobile Graphics and Interactive Applications*, pages 1–7. 2015.