# Building Immersive Worlds: Authoring Content and Interactivity in Virtual Reality

Von der Fakultät für Informatik der RWTH Aachen University
zur Erlangung des akademischen Grades

einer Doktorin der Naturwissenschaften

genehmigte Dissertation

vorgelegt von

Sevinc Eroglu, M.Sc.

Berichter:
Univ.-Prof. Dr. Torsten W. Kuhlen
Univ.-Prof. Dr. Benjamin Weyers

Tag der mündlichen Prüfung: 9. Januar 2026

**Abstract**

Despite the growing adoption of Virtual Reality (VR) across various fields, the complexity of creating interactive virtual environments limits its wider use and impact. Traditional content creation workflows often require technical expertise in desktop-based tools, which poses a barrier for many potential users. Even with expertise, switching between VR and desktop environments disrupts immersion and workflow. This challenge has motivated the development of immersive authoring systems that enable users to create and modify virtual environments directly within VR. However, due to diverse use cases and application domains, no single authoring technique can meet all requirements; while some require speed and accuracy, others might prioritize expressive capabilities. In this thesis, we therefore investigate and propose user-friendly immersive authoring techniques that simplify the creation of interactive virtual environments across various use cases. Creating such environments involves two main challenges: creation of content and definition of its behavior. Thus, our contributions are divided into two parts.

In the first part, we focus on immersive content authoring, i.e., the creation and arrangement of 3D scene elements. We present three novel authoring approaches at increasing levels of abstraction, each tailored to different use cases and interaction modalities. First, we present an artistic workflow that enables users to transform 2D paintings into expressive 3D artworks through a novel sculpting technique that enables intuitive shape manipulation with minimal effort. Second, we introduce a sketching-based procedural method for creating fluid-like 3D content, allowing manipulation via mid-air gestures and a novel blowing-based interaction to support expressive creation. Third, we propose a high-level scene authoring technique in a simulation context, enabling fast and precise object placement for road network creation via novel indirect free-hand interactions.

In the second part, we focus on authoring interactivity, i.e., defining how virtual objects respond to user actions or environmental triggers. We introduce simplified visual programming approaches designed for immersive use and further investigate the positioning of the interface elements to improve user experience. In this context, we present a block-based visual programming approach where users define behavior by combining conditions and actions via drag-and-drop interactions. We further introduce a behavior authoring technique inspired by dataflow programming, in which users visually connect input sources to object parameters. In addition, we show how the spatial positioning of the interface elements affects usability and performance.

In summary, this research contributes valuable insights into the design of user-friendly immersive authoring techniques and VR interfaces that facilitate the effective creation of interactive virtual environments for a broad range of users.

## Zusammenfassung

Ungeachtet der wachsenden Verbreitung der virtuellen Realität (VR) erschwert die Komplexität der Erstellung interaktiver virtueller Umgebungen deren umfassende Nutzung. Die Erstellung virtueller Inhalte erfordert traditionell technisches Know-how in Desktop-basierten Werkzeugen, welches vielen Nutzern fehlt. Weiterhin unterbrechen Desktop-basierte Werkzeuge die Immersion, was eine umständliche Arbeitsweise zur Folge hat. Dies motivierte die Entwicklung immersiver Autorenwerkzeuge, mit denen virtuelle Welten innerhalb der VR erstellt werden können. Da Anwendungsfälle und Zielsetzungen variieren, kann keine einzelne Technik alle Anforderungen erfüllen; manche Szenarien erfordern Schnelligkeit und Präzision, andere kreative Ausdrucksfähigkeit. Diese Arbeit untersucht und entwickelt daher benutzerfreundliche immersive Autorenwerkzeuge, die die Erstellung interaktiver virtueller Umgebungen für verschiedene Anwendungsfälle vereinfachen. Die Erstellung dieser Umgebungen umfasst zwei wesentliche Aufgaben: Inhaltserstellung und Verhaltensdefinition, anhand derer diese Arbeit gegliedert ist.

Im ersten Teil liegt der Fokus auf der Inhaltserstellung, also der Erzeugung und Anordnung von 3D-Objekten. Wir stellen drei neuartige Ansätze auf unterschiedlichen Abstraktionsebenen vor, jeweils angepasst an verschiedene Anwendungsfälle und Interaktionsformen. Zuerst stellen wir ein künstlerisches Werkzeug vor, das mittels einer neuartigen Interaktionstechnik ermöglicht 2D-Gemälde mit minimalem Aufwand in expressive 3D-Kunstwerke zu formen. Weiterhin führen wir eine prozedurale Methode zur Erstellung flüssigkeitsähnlicher 3D-Kunstwerke ein, die expressive Manipulationen mittels Gesten sowie Pusten als neuartiger Interaktionsform ermöglicht. Zuletzt präsentieren wir eine Technik im Simulationskontext, die eine schnelle und präzise Objektplatzierung für die Erstellung von Straßennetzen durch indirekte Freihandinteraktionen ermöglicht.

Der zweite Teil behandelt die Verhaltensdefinition, die beschreibt wie virtuelle Objekte auf Nutzeraktionen oder Umwelteinflüsse reagieren. Wir stellen vereinfachte grafische Programmieransätze für die immersive Nutzung vor und untersuchen weiter die Positionierung der Interface-Elemente zur Verbesserung des Nutzererlebnisses. Wir präsentieren einen blockbasierten visuellen Programmieransatz, bei dem Nutzer das Verhalten anhand von Bedingungen und Aktionen per Drag-and-Drop-Interaktion definieren. Weiterhin führen wir eine Datenfluss-basierte Technik ein, bei der Nutzer Eingabequellen visuell mit Objektparametern verbinden. Zudem zeigen wir, wie die räumliche Anordnung der Interface-Elemente die Nutzbarkeit und Effizienz beeinflusst.

Zusammenfassend trägt diese Forschung neue Erkenntnisse zur Gestaltung benutzerfreundlicher immersiver Autorenwerkzeuge bei, die die effektive Erstellung interaktiver virtueller Umgebungen für ein breites Nutzerspektrum erleichtern.

**Acknowledgments**

I would like to thank my scientific advisor, Torsten W. Kuhlen, for his trust, guidance, valuable feedback, support, and especially for giving me the freedom to explore new and creative approaches throughout this research. And of course, I want to thank my second scientific advisor, Benjamin Weyers, for his constant support from the very beginning. Thank you for always making time for long discussions, for your patience, for listening to my ideas, and for helping me shape and refine them. I am sincerely grateful for your encouragement, insight, motivation, and push to publish. I feel truly lucky to have had the support of two advisors whose guidance complemented each other so well.

I want to extend my gratitude to Sascha Gebhardt, my first and excellent mentor, who guided me through my initial steps into the field. I am grateful for his encouragement and support in helping me bring a creative perspective to VR, and for always being generous with his time and advice. I am also thankful to my dear colleagues from the Virtual Reality and Immersive Visualization Group, with whom I had the pleasure to work over the years: Martin Bellgardt, Andrea Bönsch, Aliki Charalabidou, Ali Can Demiralp, Jonathan Ehret, Sebastian Freitag, Tim Gerrits, David Gilbert, Dirk Helmrich, Kris Tabea Helwig, Jens Koenen, Marcel Krüger, Yuen Law, Fabian Lennartz, Aleksandra Lukic, Jan Frieder Milke, Jan Müller, Simon Oehrl, Heiko Overath, Sebastian Pape, Till Petersen-Krauß, Faysal Qurabi, Timon Römer, Daniel Rupp, Andrea Schnorr, Tim Weißker, Viktor Wolf, and Daniel Zielasko. Thank you all for your feedback, support, for taking the time to participate in studies, and for the inspiring discussions throughout these years. I am deeply grateful for the great environment that we shared. I am also grateful to the project partners, students, and student assistants who contributed to this work, namely David Anders, Alain Chevalier, Philipp Ljubarskij, Carlos Aguilera Martinez, Jana Rusch, Kilian Sinke, Frederic Stefan, and Arthur Voigt.

My deepest thanks go to Patric Schmitz, who is my friend, my love, my partner in crime, my mentor, and collaborator. Thank you for always being by my side, for the endless conversations, for listening, questioning, debating ideas, and for helping me see aspects I might have overlooked. I am especially grateful for your support during long nights and stressful moments, for cheering me up, and for inspiring me to keep growing.

Most importantly, I want to thank my wonderful parents, Hamdi and Maksude Eroglu, and my brother, Semih Eroglu. Thank you for always making me smile, for lifting my mood, for helping me see the positive side, and for always supporting and believing in me. No words can fully express my gratitude for everything you have done for me. I love you and thank you so much!

Fakultät für Mathematik, Informatik und Naturwissenschaften | RWTH AACHEN UNIVERSITY

**Eidesstattliche Erklärung**
**Declaration of Authorship**

I, Sevinc Eroglu

declare that this thesis and the work presented in it are my own and has been generated by me as the result of my own original research.

Hiermit erkläre ich an Eides statt / I do solemnly swear that:

1. This work was done wholly or mainly while in candidature for the doctoral degree at this faculty and university;

2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this university or any other institution, this has been clearly stated;

3. Where I have consulted the published work of others or myself, this is always clearly attributed;

4. Where I have quoted from the work of others or myself, the source is always given. This thesis is entirely my own work, with the exception of such quotations;

5. I have acknowledged all major sources of assistance;

6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;

7. Parts of this work have been published before as listed below.

Aachen, 09. January 2026

**Pre-Released Publications**

1. **Eroglu, S.**, Gebhardt, S., Schmitz, P., Rausch, D., and Kuhlen, T. W. (2018). Fluid Sketching—Immersive Sketching Based on Fluid Flow. In: *IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, doi:10.1109/VR.2018.8446595

2. **Eroglu, S.**, Schmitz, P., Martinez, C. A., Rusch, J., Kobbelt, L., and Kuhlen, T. W. (2020). Rilievo: Artistic Scene Authoring via Interactive Height Map Extrusion in VR. In: *Leonardo Journal*, doi:10.1162/leon_a_01933 & In: *ACM SIGGRAPH Art Papers*, doi:10.1145/3386567.3388577

3. **Eroglu, S.**, Stefan, F., Chevalier, A., Roettger, D., Zielasko, D., Kuhlen, T. W., and Weyers, B. (2021). Design and Evaluation of a Free-Hand VR-based Authoring Environment for Automated Vehicle Testing. In: *IEEE Virtual Reality and 3D User Interfaces (VR)*, doi:10.1109/VR50410.2021.00020

4. **Eroglu, S.**, Voigt, A., Weyers, B., and Kuhlen, T. W. (2024). VRScenarioBuilder: Free-Hand Immersive Authoring Tool for Scenario-based Testing of Automated Vehicles. In: *IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, doi:10.1109/VRW62533.2024.00040

5. **Eroglu, S.**, Schmitz, P., Sinke, K., Anders, D., Kuhlen, T. W., and Weyers, B. (2024). Choose Your Reference Frame Right: An Immersive Authoring Technique for Creating Reactive Behavior. In: *30th ACM Symposium on Virtual Reality Software and Technology (VRST)*, doi:10.1145/3641825.3687744

For my contribution to each of these papers, see Section 1.3.

# Contents

# List of Figures

# List of Tables

# List of Acronyms

| | |
|---|---|
| **3DUI** | 3D User Interface |
| **ANOVA** | Analysis of Variance |
| **AR** | Augmented Reality |
| **CAVE** | Cave Automatic Virtual Environment |
| **DOF** | Degree of Freedom |
| **HMD** | Head-Mounted Display |
| **IVE** | Immersive Virtual Environment |
| **SD** | Standard Deviation |
| **SSQ** | Simulator Sickness Questionnaire |
| **SUS** | System Usability Scale |
| **UEQ** | User Experience Questionnaire |
| **VPL** | Visual Programming Language |
| **VR** | Virtual Reality |
| **WIM** | World-In-Miniature |

CHAPTER 1

# INTRODUCTION

## 1.1 Motivation

Virtual Reality (VR) can provide realistic and interactive environments that support skill development through training [Abich IV et al., 2021; Mao et al., 2021] and enable the safe testing of dangerous scenarios in controlled settings [Pedram et al., 2020; Riegler et al., 2021; Adami et al., 2021]. VR can further introduce new ways of learning by enabling students to experience and interact directly with the subject in ways that would not be possible with traditional methods, such as traveling through blood vessels and combating pathogens to better understand the human immune system [Zhang et al., 2019]. In fields like architecture [Hsu et al., 2020] and art[1], VR enables creators to work at life-size scales, providing a realistic sense of space and depth. This enables them to easily explore, evaluate and interact with their designs before they are fully completed, which can lead to a more effective design process [Freeman and Salmon, 2017]. Additionally, VR provides new ways of artwork creation by freeing artists from the constraints of two-dimensional (2D) media and enabling them to draw, and shape their creations directly in three-dimensional (3D) space [Keefe et al., 2001; TiltBrush, 2016; Adobe, 2020].

However, creating interactive virtual environments can still be challenging. Existing 2D desktop-based creation tools often require technical expertise, which can create difficulties for users who lack programming or 3D modeling. Even with expertise, using these tools also require users to frequently switch between VR and desktop environments, which disrupts immersion and workflow. To overcome these limitations, *immersive au-*

[1]IEEE VR 2025 XR Gallery, https://ieeevr.org/2025/program/xrgallery, last-visited: 2024-04-14

*thoring* provides a promising solution by enabling users to design, modify, and interact with VR content directly within the virtual environment [Zhang and Oney, 2020; Nebeling et al., 2020; Eroglu et al., 2021].

*Immersive authoring* [Lee, 2009] refers to the process of creating and modifying content fully within an immersive virtual environment (IVE), thereby eliminating the necessity for conventional desktop-based tools and frequent transitions between VR and external platforms. By allowing creators to design and evaluate their work in the immersive environment, this method can shorten turnaround times. When users manipulate elements such as objects, lighting, or textures inside VR, they can observe the results instantaneously. This real-time feedback enables rapid issue identification and quick refinements, which is particularly useful for iterative design and testing [Delgado et al., 2020].

Additionally, manipulating objects within VR improves spatial perception, enabling creators to better understand scale, depth, and spatial relationships [Paes et al., 2017]. This improved spatial perception can support the creation of more realistic and engaging content since designers experience perspectives firsthand, and make informed decisions regarding placements and interactions. Overall, immersive authoring can improve the effectiveness of content creation by maintaining a consistent mental model of the virtual environment.

To create an interactive immersive environment, two main steps must be considered: the creation of content, and the definition of the content behavior [Lee et al., 2005]. To achieve interactivity, it is not sufficient to create static objects within the scene; it is also necessary to define how these objects will behave when interacting with the user or other objects. Thereby, in this work, we identify two major challenges:

1. **Content Creation:** To construct an immersive environment, we first need to create and arrange content in the scene. In this work, content creation specifically refers to defining 3D scene elements rather than texts, images, or audio. Content creation can be achieved through various methods, ranging from low-level geometric modeling for detailed control, to high-level scene authoring, which may utilize pre-made objects for faster scene assembly. However, as mentioned before, traditional desktop-based modeling and authoring tools often require significant technical expertise, making them inaccessible to non-technical users. Additionally, these tools rely on 2D interfaces and mouse-based manipulation, which are not suitable for the immersive nature of VR. As a result, traditional design workflows cannot be directly adapted for such environments. To address this, alternative interaction techniques and interface designs are needed to enable users to create, modify, and arrange 3D content effectively, regardless of their technical expertise.

   Existing approaches have explored various methods for creating objects in VR. On a low level, some focus on detailed geometric control by extending desktop-based

3D modeling tools into VR [Arqueros et al., 2012; Takala et al., 2013; Mine et al., 2014a], while others use spatial input for more creative forms of direct shape manipulation, such as virtual sculpting [Ponto et al., 2013; Jackson and Keefe, 2016; Dashti et al., 2022], or 3D sketching [Keefe et al., 2001; Rosales et al., 2019; Yu et al., 2021b]. For high-level scene authoring, previous approaches focus on arranging objects rather than modeling them from scratch by using input modalities such as gloves and free-hand interaction [Mapes and Moshell, 1995; De Leon et al., 2016], voice control [Barot et al., 2013] or a tracked tablet [Wang et al., 2013], while some also use a World-In-Miniature (WIM) interface to improve the arrangement at different scales [Mine, 1995; Trueba et al., 2009; Jin et al., 2020]. The variety of content creation approaches designed for specific needs highlights the diversity of demands in immersive authoring. Ongoing research in this area indicates that no single approach can address the full range of VR content authoring needs. The research gap identified in Section 2.1 highlights the need for further development and evaluation of techniques that simplify content creation processes for a broader range of users, tasks, and domains.

2. **Behavior Creation:** The second essential step for creating immersive interactive environments is content behavior creation. Behavior creation involves defining how objects respond to user actions, environmental changes, or other in-world triggers. This step is particularly important because it transforms static elements into interactive, responsive entities that can enhance the realism and engagement of the virtual experience. Typically, behavior creation is achieved by programming, where developers write scripts to specify object interactions. However, this requires expertise in programming, which can pose a challenge for non-programmers as well as the previously outlined context-switch.

   Visual Programming Languages (VPLs) have been used as a potential solution, using graphical interfaces to reduce the dependence on textual coding. VPLs have also been integrated into immersive authoring tools to author object behavior in VR. Some approaches represent the code structure using graphical blocks [Sun et al., 2021; Zhu et al., 2023; Hedlund et al., 2023], while others represent the code as a graph of interconnected nodes that can be distributed in the scene [Ens et al., 2017; Solirax, 2018a] or placed on a panel [Zhang and Oney, 2020; Murray, 2022]. Despite their advantages, VPLs can still be challenging for users who are unfamiliar with programming concepts. Further simplification of these approaches is necessary to make behavior creation accessible to a wider range of users.

   In addition to the complexity of VPLs, programming interfaces in VR may lead to visual clutter and usability issues [Ens et al., 2017; Genz et al., 2021]. Since the user is co-located with the scene object when defining its behavior, it is important to consider how these interfaces are designed and positioned within the scene to improve the user experience. While the effects of different placements in information displays [Feiner et al., 1993; Billinghurst et al., 1998; Ens et al.,

2014a] and menu interactions [Lediaeva and LaViola, 2020; Das and Borst, 2010] have been studied, their role on the immersive behavior creation using visual programming is not investigated. This reveals a research gap that highlights the need for novel approaches that eliminate the need for programming expertise while providing user-friendly VR interfaces. Addressing these challenges is important for facilitating the adoption of immersive authoring and enabling a broader range of creators to develop engaging and dynamic virtual environments.

The challenges stated above highlight the central research question guiding this thesis:

*How can user-friendly VR interfaces be designed to enable the authoring of virtual environments while being immersed?*

To address this question, it is important to understand that a one-size-fits-all approach is not feasible when creating immersive authoring of interactive environments as designing effective, user-centered interfaces is heavily dependent on the specific use case and the users involved [Abras et al., 2004; LaViola Jr et al., 2017, Sec. 3.1]. Different tasks, and application domains require diverse authoring techniques, each demanding tailored interaction methods and interface designs to create effective and user-friendly VR experiences. The goal of this thesis is to explore novel, user-friendly immersive authoring techniques that enable users to create and modify VR content. These techniques aim to support a range of authoring needs while remaining accessible to a broad spectrum of users. This research focuses on techniques and workflows for both the creation and arrangement of scene elements, as well as the definition of their interactive behaviors.

## 1.2 Contributions and Thesis Structure

This section presents the main contributions of the thesis and outlines its structure. The thesis is divided into two sequential parts, each addressing a key challenge in the creation of interactive immersive environments: the authoring of 3D content and the definition of content behavior. Each part begins by reviewing relevant literature to contextualize the contributions, identify research gaps, and motivate the design decisions.

In the first part, Chapter 2, we tackle content generation as our first challenge. We investigate how immersive content authoring techniques can be designed to enable users to create, modify, and arrange 3D content effectively, regardless of their technical background. To address this, we divide the techniques into three categories, each with increasing levels of abstraction in content generation to support a range of authoring needs. Specifically, we present three novel approaches that are in the context of 3D modeling, sketching-based procedural generation, and high-level scene authoring. Before detailing

our contributions in these areas, we first define the respective approaches, highlight their strengths, review relevant existing works, and identify research gaps (Section 2.1).

Our first contribution (as outlined in Section 2.2) focuses on enabling users to interactively convert 2D images into 3D artworks with low effort. To this end, we present an artistic workflow and introduce a novel manipulation technique for virtual sculpting. Our approach focuses on enabling creative professionals, especially those without a technical background, to create complex, visually interesting 3D shapes in an intuitive and expressive way. Next, in Section 2.3.2, we introduce a novel approach for creating 3D fluid-like artworks through immersive procedural sketching. We further device a novel blowing-based interaction technique to manipulate such artworks and present a qualitative study demonstrating the effectiveness of our approach. Afterwards, we present our contribution, in Section 2.4, on high-level scene authoring in which we focus on efficient object placement techniques for constructing virtual environments. Specifically, in the context of automated vehicle testing, we present a VR-based authoring environment for authoring road networks. To enable users to author the environment in a fast and precise manner, we introduce a novel indirect free-hand interaction technique for object placement and selection through a 2D WIM. We present the result of a comparative user study, showing our interaction technique outperformed existing approaches regarding precision and task completion time. In addition, we demonstrate the effectiveness of the authoring environment by a qualitative user study with domain experts.

In the second part, Chapter 3, we investigate how to author object behavior to create interactivity while immersed, addressing our second research direction. In this chapter, we introduce two approaches for creating interactive scenarios and examine how the design and positioning of spatial interfaces can improve the user experience in scenario creation. Our approaches build on simplified visual programming paradigms to enable a broader range of users to create interactivity. Before presenting our contributions, we provide detailed insights into visual programming paradigms, focusing on block-based and dataflow-based approaches. We discuss their strengths and challenges, review how these approaches have been adopted in VR, and present existing studies on spatial positioning of interfaces (Section 3.1).

In Section 3.2, we present our first contribution to this area, where we investigate block-based interfaces for the creation of automated driving scenarios. This type of interface uses the sequential arrangement of blocks, which fits well with the step-by-step structure of driving scenarios. To facilitate fast and intuitive scenario authoring, we enable users to create the scenarios through free-hand interactions using a drag-and-drop mechanism with condition and action blocks. We demonstrate the effectiveness of our approach through a user study, and identify research directions based on our observations and participant feedback. The finding of this study highlights the importance of the spatial arrangement of the interface elements.

Following this, our second contribution investigates how to adapt the dataflow paradigm

for immersive object behavior authoring. While the hierarchical arrangement of the block-based paradigm aligns well with linear scenario creation like driving simulation, a dataflow-based interface can be more suitable for other scenario types [Altendeitering and Schimmler, 2022]. In Section 3.3, we present a simplified dataflow-based visual programming technique. We further investigate the role of the reference frame in which the programming elements are positioned. Therefore, we implemented and compared two interface layouts and examined them regarding visual clutter, cognitive load, task completion time, and how easy to use and learn. Based on the study results, we contribute initial design implications for the design and optimization of immersive dataflow VPL interfaces for immersive authoring.

As a conclusion, in Chapter 4, we provide a concise summary of our work and discuss potential directions for future research.

## 1.3 Contribution to Publications

Parts of the research presented in this thesis have already been published in peer-reviewed articles and a journal. This section provides a list of these publications, along with a brief description of how the original publications differ from the work included in this thesis. Additionally, the contributions of the author of this thesis and the respective coauthors to each publication are outlined.

The immersive authoring technique for converting 2D images into 3D objects, presented in Section 2.2, has been published in [Eroglu et al., 2020]. The method has been designed, developed, and evaluated by the author of this thesis together with Carlos Aguilera Martinez in the scope of his Master's thesis, under close guidance throughout the entire development by the author and Patric Schmitz. Jana Rusch provided valuable feedback in the initial conceptualization and throughout the development as a target user. Torsten W. Kuhlen and Leif Kobbelt provided guidance in writing the publication.

The approach for authoring 3D fluid-like artworks within VR, presented in Section 2.3.2, has been published in [Eroglu et al., 2018]. It has been designed, developed, and evaluated by the author of this thesis within the scope of a Master's thesis. Torsten W. Kuhlen has supervised the thesis. Sascha Gebhardt and Dominik Rausch helped with the supervision of the thesis, and provided valuable feedback during the development and discussions. Furthermore, Sascha Gebhardt, Patric Schmitz, and Torsten W. Kuhlen provided guidance in writing the publication.

The free-hand, VR-based high-level scene authoring technique for fast and precise object placement, presented in Section 2.4, has been published in [Eroglu et al., 2021]. The method has been designed, developed, and evaluated by the author of this thesis.

Frederic Stefan, Alain Chevalier, and Daniel Roettger provided valuable feedback in the initial conceptualization and throughout the development as target users. Benjamin Weyers provided valuable feedback in discussions during the development of the study design. Daniel Zielasko supported the evaluation of the study data. Benjamin Weyers and Torsten W. Kuhlen provided guidance in writing the publication.

The block-based approach for linear scenario authoring, presented in Section 3.2, has been published in [Eroglu et al., 2024b]. The initial conceptualization of the technique, the study design, and the data analysis were carried out by the author of this thesis. Arthur Voigt implemented the approach and conducted the study as part of his Master's thesis, under the close guidance of the author throughout the entire development. Benjamin Weyers and Torsten W. Kuhlen provided guidance in writing the publication.

The dataflow-based approach for object behaviour authoring along with its two spatial interfaces, presented in Section 3.3, has been published in [Eroglu et al., 2024a]. In this thesis, some additional study results are reported. The method was designed and developed by the author of this thesis in collaboration with David Anders as part of his Master's thesis, under the close guidance of the author and Patric Schmitz throughout the entire development. The two spatial interface layouts were designed, developed, and evaluated by the author of this thesis. Kilian Sinke contributed with implementation support. Benjamin Weyers provided valuable feedback in discussions during the development. In addition, Benjamin Weyers and Torsten W. Kuhlen provided guidance in writing the publication.

To acknowledge the contributions of my co-authors, I use the first-person plural 'we' rather than the singular 'I' throughout this thesis.

CHAPTER 2

# CONTENT AUTHORING TECHNIQUES

As mentioned in Chapter 1, creating content is an essential step for building virtual environments. However, conventional 2D design workflows are unsuitable for authoring content while immersed. Therefore, alternative authoring techniques and interface designs are needed. These should enable users to create, modify, and arrange 3D content effectively while being accessible to users without a technical background.

Content creation in VR can be approached through various methods at different levels of abstraction, ranging from 3D modeling to the arrangement of pre-made objects, depending on specific needs across different fields. To cover a wide spectrum of content creation strategies that can be applied to various use case scenarios, this thesis investigates authoring techniques in three categories of increasing levels of abstraction: geometric modeling, sketching-based procedural generation, and high-level scene authoring. Each approach is characterized by unique strengths and challenges depending on the context of use.

An important aspect in effective content creation is the design of interaction techniques. These techniques must align with the specific demands of the content being developed, whether that involves precision for technical tasks or fostering creativity for artistic work. Designing such interactions requires understanding not only the general principles of VR but also the unique needs of the different types of content being created.

In the following section, we outline the three categories that are investigated in this thesis in more detail.

## 2.1 Content Authoring at Different Levels of Abstraction

The application use cases for immersive content creation can vary significantly regarding the meaningful level of abstraction of the provided authoring techniques. While some require precise and manual control, others aim to foster creativity and expressiveness. Yet some approaches use pre-made objects to rapidly author virtual test scenes by leveraging efficient interaction and navigation techniques. Therefore, we selected the categories to reflect the diverse ways users interact with and create 3D content in immersive environments. The following sections provide a detailed overview of each category. We define the respective approaches, highlight their strengths, review relevant existing works, identify research gaps, and outline the contributions made in this thesis.

### 2.1.1 3D Modeling

Geometric modeling plays a crucial role in 3D content creation, where detailed control over the geometry and structure of objects is necessary. It focuses on defining the shape of objects through geometric elements, such as vertices, edges, and faces. It enables users to create intricate shapes and forms, addressing the demand for detailed control in creative and professional fields.

However, the process of creating and refining low-level 3D meshes can be difficult for non-technical users due to the need for knowledge in mesh processing or modeling tools like 3ds Max, Maya or Blender. Additionally, users primarily interact through a 2D interface, which might cause them to lose important spatial information essential for 3D modeling. These limitations of traditional workflows can hinder the accessibility to specific user groups, particularly in the creative field where users may lack formal training. To address these issues, previous work proposes VR-based 3D modeling, enabling users to interact directly with objects in 3D space using natural interactions to simplify creation and manipulation of geometric shapes.

Early examples of such approaches include the voxel-based sculpting tool introduced by Galyean and Hughes [1991], which mimics working with clay or wax and provides tools to add or cut away material. Sachs et al. [1991] presented a CAD system utilizing six degrees of freedom (6DOF) input for both hands, enabling users to design 3D shapes by drawing, editing, fitting surfaces to groups of linked curves and deforming these surfaces to obtain the desired detail. Subsequent research has extended desktop-based 3D modeling tools into VR environments. Takala et al. [2013] presented a Blender extension enabling users to perform mesh extrusion and editing using 6DOF handheld controllers, while Mine et al. [2014b] developed an immersive version of *SketchUp*, combining 3D input for manipulation with touchscreen input for precise positioning.

Furthermore, some VR-based modeling approaches support Boolean operations for creating and manipulating 3D models. *MakeVR* by Jerald et al. [2013] introduced a two-handed interface that enables users to modify objects by stretching, cutting, and combining. Similarly, Mendes et al. [2017] explored mid-air modeling using hand and arm tracking with Microsoft Kinect and Myo armbands to perform Boolean operations, complemented by a menu-based alternative. Building on these ideas, Fu et al. [2022] introduced *EasyVRModeling*, leveraging a precomputed component dataset with discrete signed distance functions to enable the interactive creation of complex shapes through Boolean operations using motion controllers.

In addition, there exist immersive 3D sculpting tools. *SculptUp* by Ponto et al. [2013] enables users to compose volumetric models from cubes and spheres, which can be merged, edited, or colored. The commercial tool Medium by Adobe [2020] provides a library of approximately 300 template shapes, allowing users to build volumetric models and apply sculpting operations, such as elastically moving mesh vertices without distorting the volume. Recent advancements, like the interactive mesh sculpting tool by Zhu and Yang [2024], leverage octree data structures to efficiently organize vertex data during deformation, enabling users to refine rough geometries as if sculpting clay.

Furthermore, there is a line of research further focus on image-based modeling that involves lifting 2D sketches into 3D while immersed. In this context, *Lift-Off* is introduced by Jackson and Keefe [2016], an immersive 3D modeling interface that utilizes image detection algorithms to identify curves in hand-drawn 2D sketches. These curves can then be selectively extruded, and additional surfaces can be created through sweeping gestures with a stylus. However, it is reported that prolonged use of the stylus may lead to arm fatigue. Another notable example is *PanoTrace*, developed by Sayyad et al. [2017], which enables the creation of virtual environments from panoramic images. Users trace lines and draw polygons by snapping the controller's pointer to edges detected by an algorithm. These polygons can then be extruded into 3D models, with features allowing users to modify vertices, clone, scale, reposition models, and adjust the projected texture behind objects for a seamless appearance. However, the system is limited to simple, uniform extrusions, resulting in faces being extended to a single height each time. This restricts their ability to produce intricate surface details or varied topographies without requiring manual adjustments or additional tools.

Building on previous image-based modeling approaches, we aim to further simplify the modeling process while addressing their limitations. By introducing heightmap-driven extrusions, our method, *Rilievo*, enables the creation of complex, non-uniform surface shapes directly from 2D images. The sculpting is achieved by interactively blending heightmaps, which are automatically generated from the input image's structure and appearance. We further introduce a novel manipulation technique, the *drawbar* metaphor, which enables users to perform virtual sculpting intuitively by dragging drawbar handles. Our approach removes the need for prior knowledge of mesh processing or modeling, making 3D object creation more accessible to artists while providing detailed manual

control. It enables them to bring their paintings into VR and transform them into 3D objects with low effort.

Further details regarding the workflow of Rilievo and the manipulation technique are presented in Section 2.2.

### 2.1.2 Procedural Sketching

In the previous section, we discussed low-level approaches to object creation, where users have full control over the details of their intended creations. In this section, we explore procedural techniques, where the definition of objects' shape and appearance is guided by user input but partly automated by generative rules or algorithms.

Procedural generation is commonly utilized for tasks such as game design [Hendrikx et al., 2013; Rocha and Prada, 2025], virtual testing environment creation [Gambi et al., 2019; Li et al., 2022], and architectural modeling [Müller et al., 2006; Schwarz and Müller, 2015], where it enables the efficient generation of complex structures or large amounts of content that would be impractical to produce manually. However, achieving a desired outcome by manipulating the parameters of a procedural algorithm can be challenging, particularly for users without technical expertise. To address this issue, sketching-based procedural generation have been explored [Smelik et al., 2011].

Sketching, a skill developed from childhood, is a natural and intuitive way to quickly define a desired shape. It has been used as an input to guide procedural generation algorithms and explore the effects of their parameters. Sketch-based interfaces have been applied to tasks such as creating trees [Chen et al., 2008; Okabe et al., 2006; Longay et al., 2012], plants [Ijiri et al., 2006], terrains [Gasch et al., 2020], urban layouts [Benes et al., 2021] and virtual worlds [Smelik et al., 2011; Emilien et al., 2015]. However, these approaches rely on sketching on desktop or tablet screens, constrained by 2D input, which limits users' ability to perceive and manipulate content in 3D space. To enable users to create content while immersed, 3D input techniques are needed.

Immersive sketching interfaces have been explored to draw and shape objects directly in VR using 3D input techniques. The foundation for this approach was established by *Holosketch* [Deering, 1995]. It is the first sketching system designed for a head-tracked stereo VR environment, which enables users to create and manipulate 3D geometries using a hand-held six-axis wand. Following this approach, *CavePainting* [Keefe et al., 2001] is introduced as the first sketching system designed for CAVE environments. It offers a range of brush stroke types that user can select by dipping the tracked brush into a cup placed a painting table. Additionally, a tracked bucket is provided to simulate throwing paint onto walls or the floor. In this simulation, the paint behaves realistically by flying in the direction of the bucket's movement and falling under the influence

of gravity. Another approach, *FreeDrawer* [Wesche and Seidel, 2001], enables users draw and manipulate spline-based curves within a virtual environment using a tracked stylus. Based on experience with this system, the authors argue that the absence of force feedback is a drawback for drawing in immersive environments. To overcome this limitation, Keefe et al. [2007] introduced a haptic-aided input technique called *Drawing on Air* for drawing 3D curves within a fishtank VR setup. They further provide a feature to dynamically adjust line attributes, such as orientation, thickness, and color, during the drawing process. Rausch et al. present a sketching interface tailored for architectural modifications in IVEs [Rausch and Assenmacher, 2008]. They enable users to create line-based annotations and incorporate new objects into existing architectural models. The authors further enhanced their approach with a sketch-recognition framework [Rausch et al., 2010], which introduces a library of 28 predefined commands and associated symbols.

Researchers have also investigated accuracy issues in mid-air strokes for immersive sketching by automatically evaluating shape quality using mathematically defined measures [Machuca et al., 2018; Yu et al., 2021a; Arora and Singh, 2021; Yu et al., 2021b; Rosales et al., 2021]. However, these advances are beyond the scope of this thesis. For further details, we refer the reader to the literature reviews on 3D immersive sketching techniques [Machuca et al., 2023].

Using immersive sketching as an input for generating procedural content has been explored in various studies. One such approach is *BuildingSketch* by Liu et al. [2021], which interprets mid-air strokes using a deep neural network to create detailed procedural building models in VR. For procedural 3D tree creation in VR, Zhang et al. [2021] investigated sketching input via mid-air strokes, while Wu et al. [2024] used controller input in their *VRTree* system. Furthermore, Hu et al. [2024] introduce a VR-based terrain authoring system, where mid-air sketches are transformed into heightmaps enhanced with Perlin Noise [Perlin, 1985] and processed by CGANs to produce detailed terrains. While these approaches have studied immersive sketching for procedurally generating environmental content, less attention has been given to exploring more artistic and expressive content creation. To address this, we introduce a novel approach for creating 3D fluid-like content, enabling users to create fluid artworks through immersive sketching. The fluid-like behavior and appearance are achieved using a curl-noise based procedural technique [Bridson et al., 2007]. Additionally, we introduce a novel blowing-based interaction metaphor for the manipulation of such artwork.

Further details regarding our approach, along with a qualitative user study are provided in Section 2.3.

### 2.1.3 High-Level Scene Authoring

In the previous section, we explored content creation via procedural sketching, where content generation is automated by generative rules and guided by free-form sketching. In this section, we focus on high-level scene authoring, which involves creating immersive environments by arranging pre-made objects, rather than constructing every element from scratch.

High-level scene authoring is an important aspect of game design and the rapid prototyping of interactive environments, as it enables quick iteration and testing while maintaining control over the final result. This approach may simplify the authoring process and make scene authoring more accessible for non-technical users. To enable users to author their scenes effectively, it is important to provide intuitive interaction and navigation techniques.

Building on these principles, several approaches have been developed to facilitate immersive scene authoring through novel interaction and navigation techniques. For instance, Mine [1995] introduce *ISAAC*, a system designed as a testbed to enhance the composition of interactive virtual scenes while immersed. It incorporates both direct (raycasting) and indirect (WIM) selection techniques, object manipulation with constrains, travel techniques (flying), and a menu that provides conventional editing functions (cut, copy, duplicate, delete, and group). To further improve scene authoring, Mapes and Moshell [1995] present MultiGen's *SmartScene*, derived from *PolyShop*, studied two-handed interaction techniques using tracked data gloves for object manipulation and viewpoint control. Another approach to immersive scene authoring is proposed by Wang et al. [2013] with *DIY World Builder*, a hybrid system that combines a wand and forearm-mounted interaction tablet interface to construct the scene. The system offers features to modify terrain, place and texture objects, and control lighting while immersed. The authors state that while most users responded positively, some experienced disruptions in their sense of presence and interaction flow when switching focus between the tablet and the head-mounted display (HMD). Additionally, users found it challenging to judge scale when adding elements through the tablet due to the absence of reference objects.

To further investigate on interaction techniques for immersive authoring, Barot et al. [2013] introduce *Wonderland Builder* that enables users to create and manipulate objects using multi-modal interactions, combining voice and tracked hand inputs. The system also enables users to adjust their avatar's scale via voice commands for efficient navigation and precise object transformations. Furthermore, De Leon et al. [2016] studied scene authoring using free-hand gestural input. They proposed *Genesys*, a VR scene builder that enables users to load and manipulate assets from a built-in content browser in VR. While the system received positive feedback, they noted that two-handed interaction for rotating and scaling objects could be improved.

Another approach for rapid scene composition is presented by Ichikawa et al. [2018] in *VR Safari Park*. It enables users to design a virtual environment by attaching blocks, representing objects such as animals, to a tree interface with branches correspond to different scene components like weather, flying animals, terrestrial animals, and terrain. Once connected, objects appear automatically, and users can refine their positions and orientations through direct interaction. The authors note that this approach is best suited for world building scenarios where elements are not bound by strict spatial relationships.

Alternatively, *VRCopilot* by Zhang et al. [2024] explore the usage of generative AI models to author immersive environments. The authors compares three modes of human-AI co-creation for room layout design: manual creation, where the user selects object from a catalog menu; automatic creation, in which the user dictates the intent and generative AI suggests full-room layouts for further refinement; and scaffolded creation, where the user and AI collaboratively define intermediate representations to guide the final design. The study revealed that the sense of agency, the awareness and control over one‘s action, significantly increases from automatic to scaffolded to manual creation. They further note that while generative AI provides inspiration for scene creation, users often struggle with its non-deterministic output, which leads to misalignment with their design goals due to limited control.

Building on these approaches, we investigate high-level scene authoring in the context of immersive road network creation for the simulation of traffic scenarios, which has not been addressed in previous works. This aspect is critical for engineers who must efficiently and precisely design and manipulate complex scenes for safe automated vehicle testing in VR. To address this research gap, we propose a VR-based authoring environment that supports an iterative workflow for creating road networks and simulating automated driving in an immersive setting. While road geometry is procedurally generated, users defines the structure by placing control points. Users author the road network and traffic scenarios via placement and manipulation of objects at run-time using free-hand gestural interactions. For efficient and precise object selection and manipulation in our context, we introduce a novel 3D interaction technique. It is an indirect interaction technique that is employed for placement and selection of objects on the 2D projected terrain surface. This technique is integrated with a scene control panel consisting of a 2D minimap and a navigation panel.

Similar to our approach, Côté and Beaulieu [2019] present a VR system that enables users to create and modify roads using free-hand gestures. Their method enables users to place control points directly on the terrain and adjust the road shape by dragging these points. Regarding navigation, they use the two-handed viewport control Mapes and Moshell [1995] in which is performed by scaling, or dragging the virtual world through a gesture-based interaction. To evaluate our technique, we implemented their direct interaction and travel techniques and compared with our indirect interaction approach for road creation and virtual navigation.

Further details on our proposed authoring technique and the user study can be found in Section 2.4.

## 2.2 From 2D Paintings to 3D Artistic Content: Intuitive Surface Modeling

*The contents of this section are based on – and taken in part from – work previously published in [Eroglu et al., 2020].*

**Figure 2.1:** Top left: "twisted and urban", 190 x 190 cm, mixed media on canvas (© Jana Rusch, 2019). Top center: an interpretation created in VR. Other: artworks created with our system.

Content creation entirely inside VR is an exciting area that is also explored for new forms of artistic creation. For this reason, powerful tools are being developed, ranging from 3D drawing applications to virtual sculpting and 3D modeling, such as Google Tilt Brush, Gravity Sketch, Facebook Quill and Oculus Medium. While these tools offers exciting new possibilities for the creation of 3D artworks, artists also want to bring existing 2D artworks into VR. An example is to turn existing paintings on canvas into plausible 3D objects to present in an immersive environment. This way art pieces can be experienced from new perspectives and dynamic interactions with the static creations become possible. However, transitioning 2D artwork into VR poses significant challenges. Current workflows involve manual segmentation, 3D modeling, and content alignment, which requires training with geometric modeling tools and considerable manual work

[Kottlowski, 2019]. There is a clear need for tools that simplify this process while keeping the expressiveness of the original 2D artwork.

Therefore, in this section, this work investigates *how can an immersive authoring technique be designed to enable artists to interactively convert 2D paintings into 3D objects with low effort?*

To answer this question, we developed a VR authoring environment to create a virtual 3D composition from 2D material. Our system enables artists to define their individual collection of compositional elements based on images and, optionally, relief height maps. To create the elements, the artist first isolates desired parts of the provided image using a segmentation tool. Instances of these elements are dragged into the scene from a palette and arranged spatially using direct manipulation for rigid transformations and a bimanual controller gesture for scaling.

To give each element instance an individual 3D appearance, its surface is extruded by mixing different height profiles with a novel direct manipulation technique we call the *drawbar* metaphor. It enables simplified virtual sculpting without knowledge about mesh processing or 3D modeling by dragging out *drawbar* handles. Plausible height profiles are inferred from the input image or user-provided graphics. Relief height maps that were manually sculpted in an analog process and digitized using a 3D scanner can be used. In the following sections, we detail the segmentation, feature-based height map extraction and interactive extrusion steps.

### 2.2.1 Segmentation and Height Map Extraction

To create a compositional element, the artist first defines a region in the input image that contains the desired structure. To reduce manual labor and speed up the artist's workflow, we perform a segmentation based on the GrabCut algorithm [Rother et al., 2004]. The artist sparsely labels regions as foreground or background pixels by drawing onto the input image with a variable-size brush. The algorithm then infers the optimal labeling of foreground and background by solving an iterated graph cut problem. Figure 2.2 shows our interactive segmentation tool.

The resulting segmentation mask serves as the basis for our first height map. We perform a distance transform on the outline, which results in a height profile that increases with distance from the object contours. Since many objects exhibit a profile that bulges outward from the center and falls off toward the sides, this yields a plausible basic 3D shape in many cases. In addition to the height profile inferred from the contour, we enable the artist to add higher-frequency detail. To this end, we perform edge detection on the input image using a Sobel filter for intensity gradient estimation, followed by non-maximum suppression and hysteresis thresholding, similar to the Canny edge detection

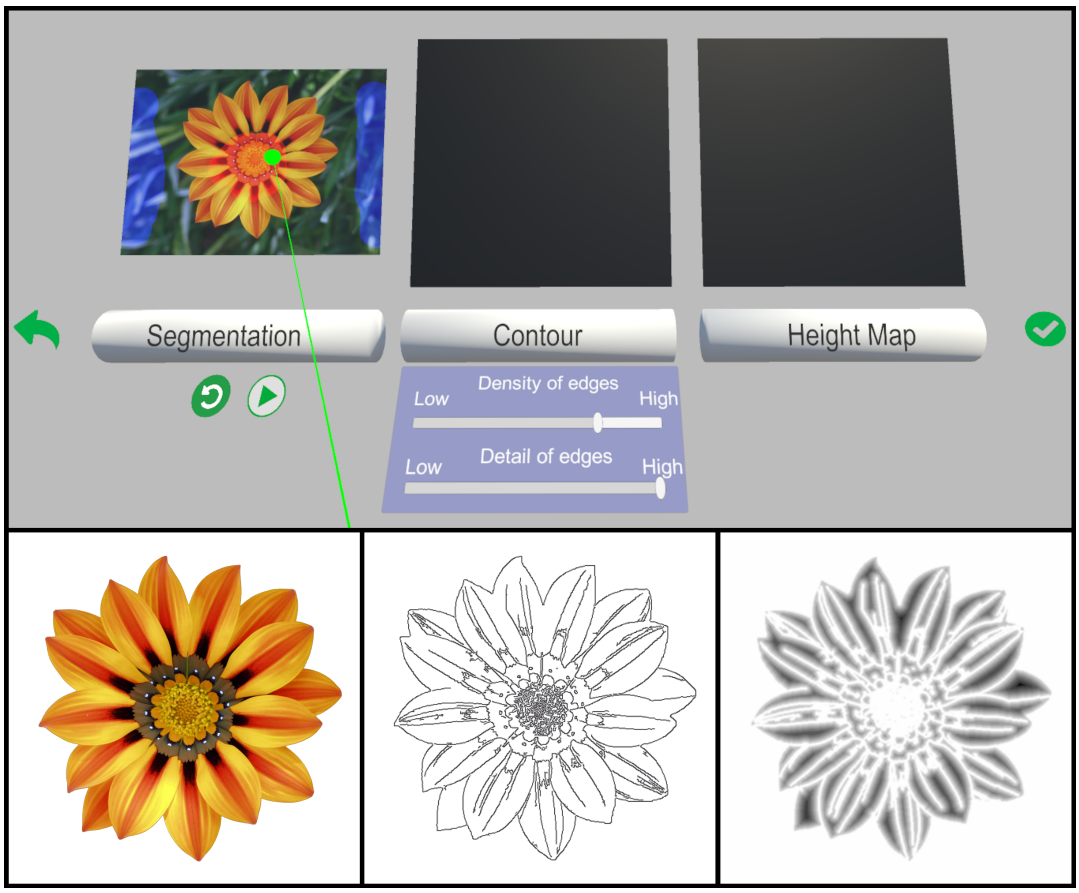



**Figure 2.2:** Top: 3D user interface for segmentation, feature extraction and height map generation. Bottom: a segmentation result, the extracted feature edges and the inferred height map. (Source flower image © Sam Oth via Creative Commons, CC BY-SA-2.5.)

algorithm [Canny, 1986]. On the resulting edge image, we apply a distance transform, yielding a height map that grows with the distance from salient image features. We provide the artist with a simple user interface to adjust the hysteresis thresholds until the desired feature edge density is achieved. The resulting edge image and corresponding height map are updated interactively while the artist adjusts the sliders, enabling a fast and intuitive workflow. Lastly, we generate height maps based on the luminance intensity in the input image or any user-provided image file. This enables the artist to overlay the height profile with the object's own apparent brightness, or emboss arbitrary structure from regular or stochastic input textures, such as brick walls, cell structures or marble, into the element's final extruded height profile.

### 2.2.2 Interactive Height Map Extrusion

Our tool presents a palette of compositional elements as a menu that is attached to the non-dominant hand. It shows the image from which the object was created next to the generated height maps. A submenu enables the selection of additional height maps supplied by the artist (see Figure 2.3 Bottom left). After an element is selected, a preview floats above the menu, which can then be instantiated by dragging it into the scene. We provide a height map extrusion panel based on the *drawbar* metaphor to create an individual 3D appearance for each element. It is inspired by vintage tone wheel organs, where the final sound is mixed from individual registers by pulling out sliding drawbars. The surface is displaced symmetrically in opposing directions to create a closed surface mesh that can be observed from any direction.

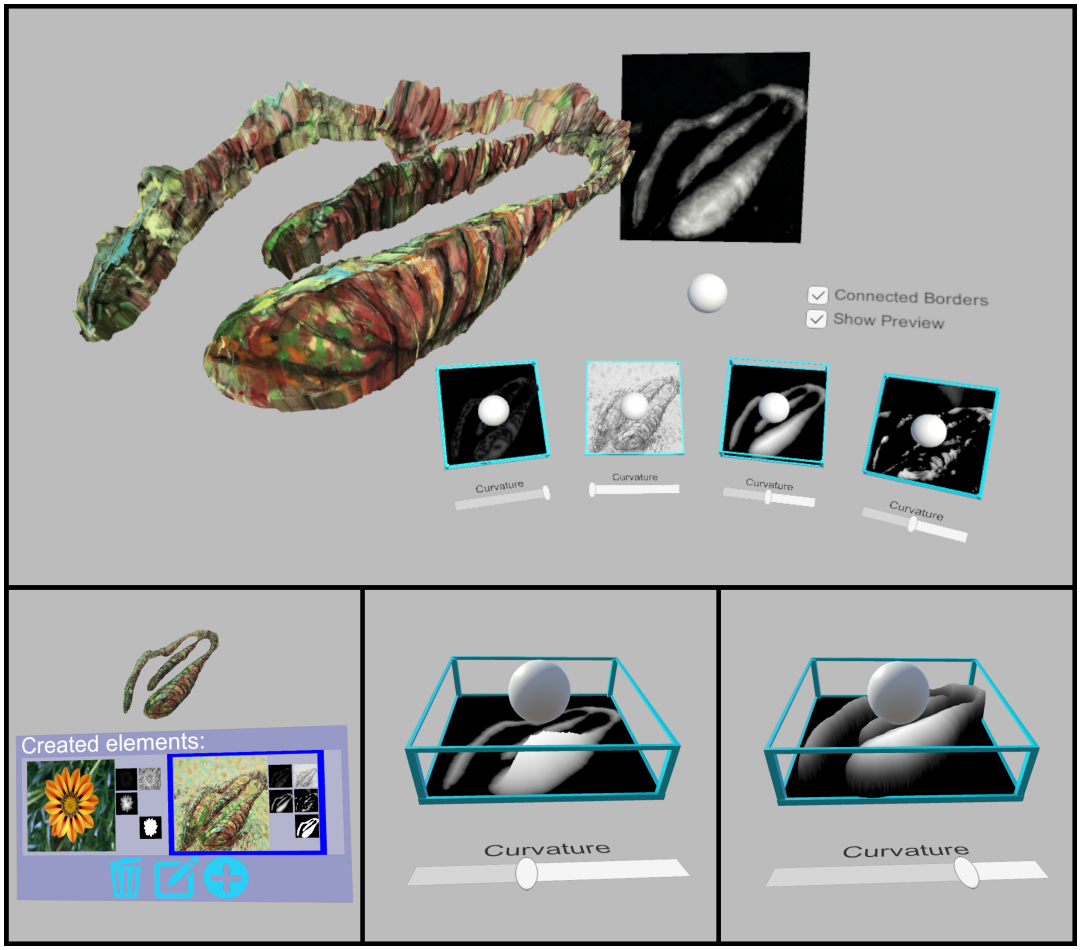



**Figure 2.3:** Top: 3D user interface for interactive height map extrusion using *drawbar*. Bottom left: element palette. Bottom center and right: close-up of the beveling tool. (Flower image © Sam Oth via Creative Commons, CC BY-SA-2.5.)

In the extrusion panel, the height maps are laid out in an arc shape as shown in Figure 2.3

Top. Each height map is depicted as a grayscale image and features a drawbar handle that is dragged outward to perform an extrusion. The individual grayscale images are displaced to visualize the relative extrusion depth of each component. A large grayscale image in the center shows the final mixture for additional visual reference. We offer an option to hide the preview image to create an unobstructed view of the element. For non-smooth height maps, the opposing surfaces can detach visually. While this can be a desired effect, we provide an option to connect the parts by automatically adding appropriate faces. While sculpting the height map mixture using the drawbars (see Figure 2.4), the element's 3D shape is continuously updated. The height maps generated via distance transforms vary linearly with distance to contours or feature edges. This results in a pyramidal shape of the height profile, with a sharp ridge at the medial axis. We offer a "beveling" tool to add positive or negative curvature to the slope of the height profile. This enables the artist to continuously vary the profile from sharp and pointy to smooth and round, which Figure 2.3 illustrates.

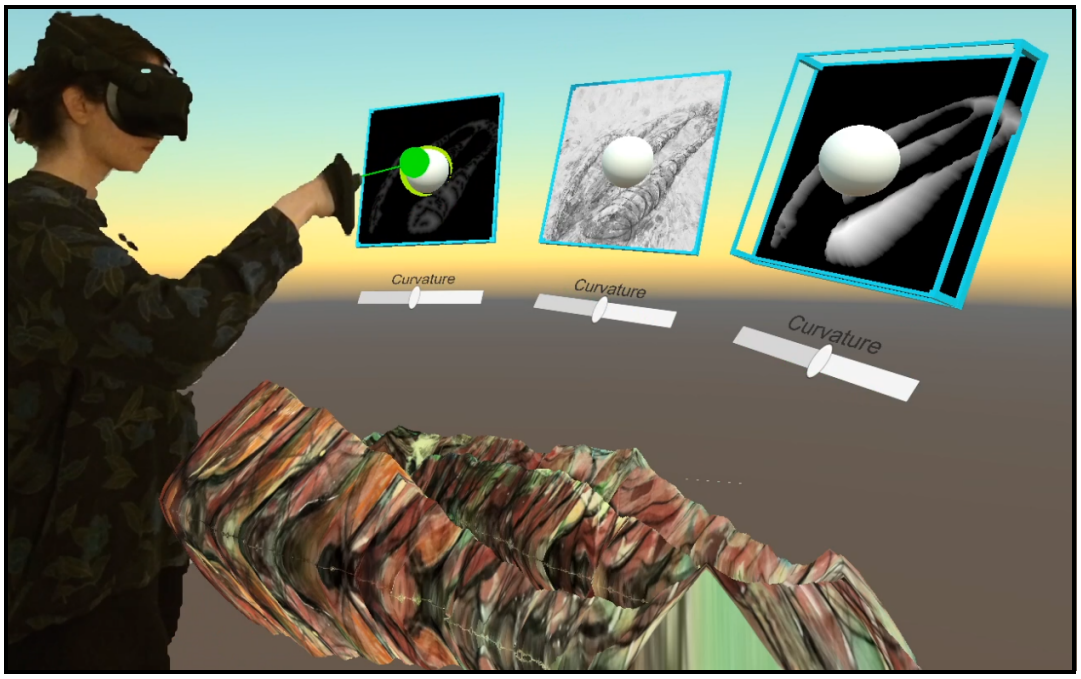



**Figure 2.4:** The user pulls the *drawbar* handle of a generated height map to perform an extrusion on the segmented element.

### 2.2.3 Analog Painting and Sculpting for Rilievo

In our hybrid analog and virtual workflow, graphical elements that are drawn on canvas serve as the starting point for the virtual art piece. Painters can make use of any traditional painting technique that they are accustomed to, like drawing with pencils

or chalk, blending oil paint and watercolor and working arbitrary materials into the painting to achieve the desired appearance. In a second process step, a height field is sculpted underneath a projected image of the painting using clay, spackling paste or other formable materials (see Figure 2.5). This enables the artist to work with familiar painting techniques and afterward augment the piece with a height profile. To achieve this, we take a high-resolution photograph of the painting, rectify and project it onto a ground plane. Finally, a structured light scanner captures a 3D model of the sculpted relief. It is aligned with the photograph and rendered as a depth buffer image to produce a height map that conforms to the original painting. The resulting height map and the photograph of the painting then serve as input for the virtual composition process. Figure 2.5 illustrates the analog creation process.

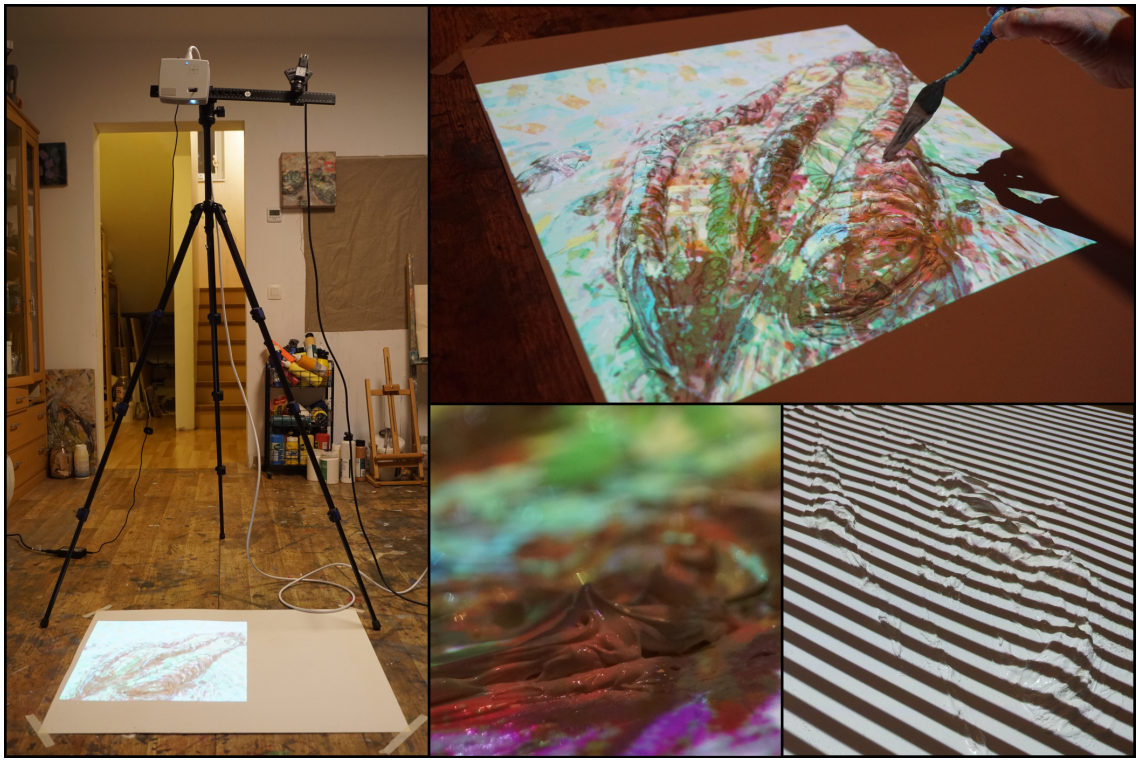


**Figure 2.5:** Analog relief sculpting underneath the projected painting.

### 2.2.4 User Feedback and Discussion

The informal feedback that we received from users of our system was quite promising. Students and colleagues from our institute, some with experience in VR and some without, found the tool simple to use and engaging. Playful experimentation with different input textures and height profiles resulted in many fascinating visual results. After a

short demonstration of the available controls, people used the system without further assistance.

Our collaborating painter Jana Rusch, who has no technical background or prior experience with mesh processing or modeling tools, provided valuable insights. She emphasized that authoring inside VR offered an entirely new artistic perspective, particularly in terms of spatial arrangement, which directly relates to her body proportions and movement. She successfully used the tool to bring her 2D paintings into VR and created further 3D artwork. Based on her feedback, we further developed a visitor mode that allows her to present her work publicly, enabling audiences to experience her creations in an immersive setting. She presents her work titled "(Non)Utopia" to the public [Doepgen, 2022].

We believe that creating artwork within an immersive environment, where it will ultimately be experienced, holds significant artistic potential. In our judgment, the tool effectively supports the envisioned tasks and target audience, demonstrating its potential as an accessible and intuitive VR authoring method.

### 2.2.5 Conclusion and Future Work

In this section, we presented a content authoring technique for artistic creation in VR. It enables the effortless conversion of 2D images into volumetric 3D objects. To achieve this, we proposed a workflow where artistic elements from the input material are extracted using a convenient VR-based segmentation widget. Relief sculpting is then performed interactively by mixing different height maps with our proposed novel direct manipulation technique, *drawbar*. To demonstrate the effectiveness of our approach, we gathered informal feedback from students and colleagues from our institute, as well as from a traditional painter with whom we closely collaborated throughout development. The feedback was highly positive, and the painter actively uses *Rilievo* to create immersive 3D artwork, successfully showcasing her work to the public.

While our approach enables users to bring their 2D paintings and create 3D elements in VR effectively, future studies can explore conducting a user study with a more widespread sample of our audience, particularly including children without any technical background. Additionally, numerous improvements could be explored in future work. Such as, enabling the placement of light sources and improving visual quality via physically based rendering (PBR) with normal maps generated from the height fields. Furthermore, allowing direct drawing of albedo color and PBR material maps, such as roughness and reflectivity, onto the element surface could enrich artistic expressions. To overcome the static nature of the compositions and integrate the possibility of modulating the visual appearance over time or based on external control inputs, we explore this in Chapter 3. Specifically we refer readers to our *modulation mapping* technique in

Section 3.3.1.1, which enables quick and direct mapping of input sources to modulation targets in the scene.

## 2.3 Fluid-Like 3D Content Creation using Natural Interactions

*The contents of this section are based on – and taken in part from – work previously published in [Eroglu et al., 2018].*

**Figure 2.6:** The artist creates 3D fluid-like content while immersed.

In the previous section, we explored how artists can interactively generate 3D objects from 2D images in an immersive environment without needing expertise in geometric modeling. To further expand immersive authoring methods, we now investigate procedural sketching that uses user input to drive rule-based content generation. While previous approaches [Liu et al., 2021; Wu et al., 2024; Hu et al., 2024] have focused on immersive procedural sketching for environmental content creation, our work targets artistic creation. Motivated by the complexity and beauty of natural phenomena, such as the interaction between fluids of different viscosities or swirling smoke, we explore how artists can create 3D fluid-like forms in an immersive environment. Our goal is to enable artists to capture the beauty of these phenomena with greater control and possibilities than the real world allows. Thus, we investigate *how authoring techniques should be designed to enable intuitive creation and manipulation of fluid-like 3D artworks?*

Fluid-based artistic expressions have long fascinated creators but present significant challenges in terms of control. Traditional fluid art techniques rely on natural diffusion

and flow, which make them difficult to predict and manipulate. For example, paper marbling shapes inks on water surfaces using styluses, or combs before transferring the result to paper [Maurer-Mathison, 1999]. However, the ongoing diffusion process demands high levels of expertise and training, since it is impossible to undo any step of the creation process. Similarly, fluid photography, such as capturing swirling smoke or ink dispersing in water, requires complex setups, environmental control, and precise timing [Academy, 2014]. Even under ideal conditions, artists do not have exact control of the smoke or ink due to their nature.

To overcome these challenges, we introduce Fluid Sketching, an immersive sketching system that enables users to create 3D structures that look and behave like fluids. Users interact directly with these structures using hand gestures and a blowing technique. Additionally, users have the ability to configure various attributes of the fluid and its diffusion process. Considering the computational cost of fluid dynamics simulations, we aim to overcome the expensive computation for simulating fluids in real-time. To this end, we use a procedural method because of its low computational cost and the high degree of animator control.

In summary, the main contributions of this work are as follows: first, we introduce a novel medium for the creation of 3D fluid artwork. Second, we present a novel blowing-based interaction metaphor for the manipulation of such artwork. Third, we demonstrate the effectiveness of our approach by means of a qualitative user study with artists and VR experts.

In the following, we first describe the realization of 3D fluid-like creations. Next, we outline the technical aspects of our Fluid Sketching system, before moving on to our interaction concept. Afterward, we present the qualitative user study and its results before concluding this section.

### 2.3.1 Fluid Simulation

In computer graphics, physically-based fluid simulation typically relies on solving the Navier-Stokes equations, which describe the motion of fluids in terms of pressure, velocity, and time [Stam, 1999]. These equations can be solved using grid-based (Eulerian) or particle-based (Lagrangian) methods. Grid-based approaches, such as Stam's stable fluids method [Stam, 2003] and its GPU variants [Crane et al., 2008] using the MacCormack scheme, offer real-time performance with improved accuracy. Particle-based methods, such as Smoothed Particle Hydrodynamics (SPH) [Gingold and Monaghan, 1977; Lucy, 1977], offer flexibility in surface representation and have been widely applied in liquid and soft-body simulations. However, SPH still faces challenges such as consistency, stability, boundary handling, and adaptivity [Vacondio et al., 2021]. These issues can limit its robustness and make it less suitable for responsive, interactive systems.

To evaluate the applicability of these approaches for our interactive system, we implemented both grid-based and SPH methods and tested their ability to offer parameter control for varying viscosities. However, neither approach delivered the necessary performance along with the high degree of parameter control required for our interactive sketching system. Therefore, we rely on a curl-noise based procedural method to realize our Fluid Sketching system. This way, we can handle large enough particle populations in real-time to create detailed immersive fluid artworks. While the resulting fluid simulation is not physically correct, it is still plausible and appealing. It also offers a higher degree of artistic control than strictly physical models.

Specifically, our fluid simulation system is based on the procedural curl-noise approach of Bridson et al. [2007]. It creates plausible fluid advection fields from the curl of a Perlin noise field [Perlin, 1985], while consuming only little computing power. The resulting velocity field is divergence-free, which is an important attribute of incompressible fluids.

To create realistic fluid behavior, the velocity field changes over time, such as, e.g., in FlowNoise [Perlin and Neyret, 2001], by using a time-varying noise function $\vec{\psi}(\vec{x},t) = \vec{N}(\vec{x}/L,t)$. If the variable $\vec{x}$ of the noise function is scaled by $1/L$, the partial derivatives of this function vary over a length scale $L$ and determine the diameter of vortices. We use four-dimensional Simplex noise [Perlin, 2001] instead of the original Perlin noise [Perlin, 1985], with the time variable in the fourth dimension. It allows for higher-dimensional noise fields with less computational cost, has easily computable analytic derivatives, and allows for a more efficient implementation in hardware compared to Perlin noise.

Bridson et al. [2007] suggest summing up several octaves of the noise function at different scales to produce turbulent structures. Lewis Fry Richardson describes their self-similar appearance as follows: “Big whirls have little whirls that feed on their velocity, and little whirls have lesser whirls and so on to viscosity” [Richardson, 1922]. To describe this kind of self-similar structure, fractional Brownian motion (fBm) was first introduced by Kolmogorov [1940] and later studied by Mandelbrot and Van Ness [1968]. It is the summation of several evaluations of a noise function at different frequencies (scales) that are varying in amplitude. Each of these components is called an octave. For each octave, the frequency increases by a lacunarity factor and the amplitude decreases by a gain factor, which is also called persistence.

We use the noise function of Bridson et al. [2007] for the fBm. To compute the curl of the potential field $\nabla \times \vec{\psi}$, we require the derivative of the potential field $\nabla\vec{\psi}$, which is generated by summing the noise derivatives at time $t$ at different scales:

$$\nabla\vec{\psi}(\vec{x},t) = \sum_{i=0}^{O-1} p^i \nabla\vec{N}(\vec{x}\cdot l^i/L,t) \tag{2.1}$$

Here, $L$ is the vorticity scale, $p$ is the persistence, $l$ is lacunarity, and $O$ is the number of octaves. $L$ and $p$ are user-configurable factors that enable interactive control of the fluid behavior. The scale of vortices can be varied from large to small by the scale factor $L$, and more to less turbulent behavior can be achieved by changing the persistence factor $p$. With a higher value of $O$ more physically plausible turbulence behavior can be simulated at the cost of decreasing simulation speed. We use $O = 3$ and $l = 2$, which we found to produce plausible results while maintaining interactive frame rates. In the implementation, we add a constant offset to each dimension of the noise function to make the potential field uncorrelated.

#### 2.3.1.1 Particle System

Particles are used for the visual fluid representation and the positions are advected in the velocity field using first-order Euler integration. Computation of the velocity field and particle advection are implemented on the GPU. Several additional properties are associated with each particle: the velocity from the previous iteration, particle color, a damping coefficient, a freeze state, and particle age. All of these are used to realize certain behaviors, which we illustrate in the following.

To reduce the number of particles to simulate and render, particle positions are constrained to the interaction volume of the target virtual environment. Particles leaving the interaction volume are temporarily marked inactive and will not be considered for computation and rendering. This yields an increase in application performance, while only disregarding particles the user would not be able to interact with anyways.

Driven by the velocity field, all particles move independently of each other. To increase the plausibility of their behavior, we model interaction between particles on top of this. Since modeling all pair-wise interactions would be prohibitive, we implement an approximation based on a low-resolution grid. Each grid cell stores the average velocity of all contained particles and each particle is affected by its current and neighboring cells' average velocities.

#### 2.3.1.2 External Influences

To enable direct interaction with the fluid, we can apply external velocities to the particle system. The overall external velocity is the sum of a spherical velocity term $\vec{v}_{sphere}$ and a conical velocity term $\vec{v}_{cone}$.

Particles inside a spherical region are affected by the movement velocity of that region, weighted with a linear fall-off based on the distance to the sphere center and a

configurable strength factor $f_{strength}$:

$$\vec{v}_{sphere} = \frac{\Delta\vec{p}_{sphere}}{\Delta t}\left(1 - \frac{\|\vec{p}_{sphere} - \vec{p}\|}{r_{sphere}}\right) f_{strength} \tag{2.2}$$

Here, $\vec{p}_{sphere}$ denotes the sphere's position, $\Delta\vec{p}_{sphere}$ is the change in position since the last frame, $\Delta t$ is the time between the current and the last frame, $\vec{p}$ is the particle position, and $r_{sphere}$ is the sphere radius. By using a linear fall-off weight, continuity between neighboring particles in the interaction range is achieved. Particles that are closer to the sphere center are influenced more strongly than distant ones, which we found to produce a more plausible behavior.

The influence on particles that lie within a conical region is given by $\vec{v}_{cone}$ in Equation 2.3. It depends on the direction $\vec{d}_{particle-apex}$ of the ray connecting the particle position $\vec{p}$ and the cone apex $\vec{p}_{apex}$, two linear fall-off weights and a strength factor $f_{strength}$. The angle $\theta$ is computed between $\vec{d}_{particle-apex}$ and the cone axis.

$$\vec{v}_{cone} = \vec{d}_{particle-apex}\left(1 - \frac{\|\vec{p}_{apex} - \vec{p}\|}{h_{cone}}\right)\left(1 - \frac{\theta}{\alpha}\right) f_{strength} \tag{2.3}$$

The first linear fall-off weight in Equation 2.3 is based on the distance between the particle and the cone apex. It leads to particles near the apex of the cone being influenced stronger than particles which are closer to the base, where $h_{cone}$ denotes the height of the cone. The other fall-off weight is calculated depending on the angle $\theta$. It is computed as $(1 - \frac{\theta}{\alpha})$, where $\alpha$ represents the half opening angle of the cone. Therefore, particles that are closer to the center axis of the cone are affected more strongly than the ones which lie at the sides.

The combined velocity $\vec{v}_{sum}$ of each particle is determined as the weighted sum of all influences described above. First, the velocity from the previous advection iteration is used to achieve a smoothing effect and thus temporal consistency of the particle movement. Next, the average velocities of the grid cell containing the particle and its neighboring cells are added to simulate particle–particle interaction. The overall external velocity is added, which is the sum of all external spherical and conical influences. Last, the velocity of the noise field is added for the fluid-like turbulence.

Finally, we account for liquids with different viscosities. Since viscosity has a resisting effect on the movement of particles, we simulate it as dampening, which has been used before to mimic viscous materials or air resistance [Latta, 2004]. The final particle velocity $\vec{v}$ is calculated by scaling $\vec{v}_{sum}$ with a normalized damping factor $k_d$:

$$\vec{v} = (1 - k_d)\vec{v}_{sum} \tag{2.4}$$

The new particle position is then computed as $x(t + \Delta t) = x(t) + \vec{v}\Delta t$.

### 2.3.2 Fluid Sketching

This section covers the user interface of the Fluid Sketching system. We first outline the technical setup of our test implementation. Next, we detail our interaction concept, before briefly explaining our rendering technique.

#### 2.3.2.1 Technical Setup

We implemented our prototype for the five-sided aixCAVE (four walls and a floor). Its footprint of 5.25 m × 5.25 m and the height of 3.30 m allows for drawing impressive life-sized fluid sketches without the need of navigation methods apart from physical walking. Head tracking is provided via an ART opto-electronic tracking system, while stereopsis is realized via 120 Hz active stereo. The system has 25 nodes, each equipped with two Intel Xeon Westmere CPUs (6 Cores at 2.7GHz), 24 GB RAM, and two NVIDIA Quadro 6000 GPUs.

Users are equipped with three different interaction devices. An ART Flystick2 provides three types of input: its 6DOF transformation, buttons, and an analog joystick. The trigger button at the front is used for creating brush strokes. The outer left of the four buttons on the top opens the system control interface, which is realized as Extended Pie Menus [Gebhardt et al., 2013]. In the menus, user-configurable parameters can be set, which are described later on. The other buttons on the Flystick2 give users quick access to the most frequently used features: the inner left button activates freeze mode, the inner right button toggles the eraser and the outer right button toggles pause mode.

Users can manipulate existing sketches with two different interaction metaphors: direct hand interaction and blowing into the sketch. For performing direct hand interaction, the non-dominant hand is equipped with an ART hand tracking target. The blowing metaphor is realized via a Sennheiser EW G2 transmitter in combination with a Sennheiser ME3 wireless head-mounted microphone.

#### 2.3.2.2 Brush

The primary drawing tool of Fluid Sketching is the 3D brush, which emits particles from the tip of the Flystick2. To allow for a high degree of flexibility and creativity, various attributes of the brush are user-configurable via respective menus: brush size and color, viscosity, density, and initial speed of the emitted particles.

Emitted particles are uniformly distributed in the volume of a sphere, whose diameter corresponds to the brush size. The limited frame rate of the tracking system (60 Hz)

results in gaps between measured emitter positions for fast movements. To mitigate this, the center of the emitting sphere is shifted randomly between the previous and current position of the brush for each particle. To obtain smooth, continuous brush strokes, cubic Hermite spline interpolation is applied.

Users can define the viscosity for individual brush strokes to vary between low-viscous fluids like water and high-viscous ones like honey. This assigns the normalized damping factor $k_d$ in Equation 2.4.

The density parameter of the brush defines the number of emitted particles relative to the brush size. The number of emitted particles $n$ is calculated as $n = {f_{density}}^3 \|\vec{p}_1 - \vec{p}_0\|$, where $f_{density}$ is the density and $\|\vec{p}_1 - \vec{p}_0\|$ is the length of the current frame's brush stroke segment.

Particles can be emitted with a user-configurable initial speed. If an initial speed is set, the direction of the initial particle velocity corresponds to the pointing direction of the Flystick2. This creates the effect of an aerosol spray can.

All of the parameters presented above are per-particle attributes, which are permanently assigned upon particle emission. Thus, users are given a high amount of control when drawing brush strokes, e.g., to combine fluids with different viscosities and densities.

#### 2.3.2.3 Fluid Configuration

Users can change three parameters of the particle simulation: turbulence, vorticity, and diffusion speed. The properties have a global effect on the fluid behavior, so users immediately perceive the effects while changing them. The *turbulence* corresponds to the persistence $p$ of the fluid advection (see Equation 2.1). To grant users a more intuitive understanding, we chose to present the less technical parameter name. The *vorticity* parameter corresponds to $L$ and allows users to obtain vortices with varying sizes as illustrated in Figure 2.7. The *diffusion speed* parameter controls the speed of the particle simulation. It gives users a high amount of control on the diffusion process, since small speeds allow for very precise working in the flow field, while high speeds produce time-lapse like effects.

To give more control of the overall amount of diffusion, we add an age-based settling effect for particles, which can be enabled or disabled according to users' needs. It gradually diminishes the diffusion for each individual particle over a user-specified diffusion time. If the effect is enabled, the diffusion process will decelerate and finally settle after the diffusion time has passed. Settled particles can be reactivated by directly interacting with them, either by hand or blowing interaction.

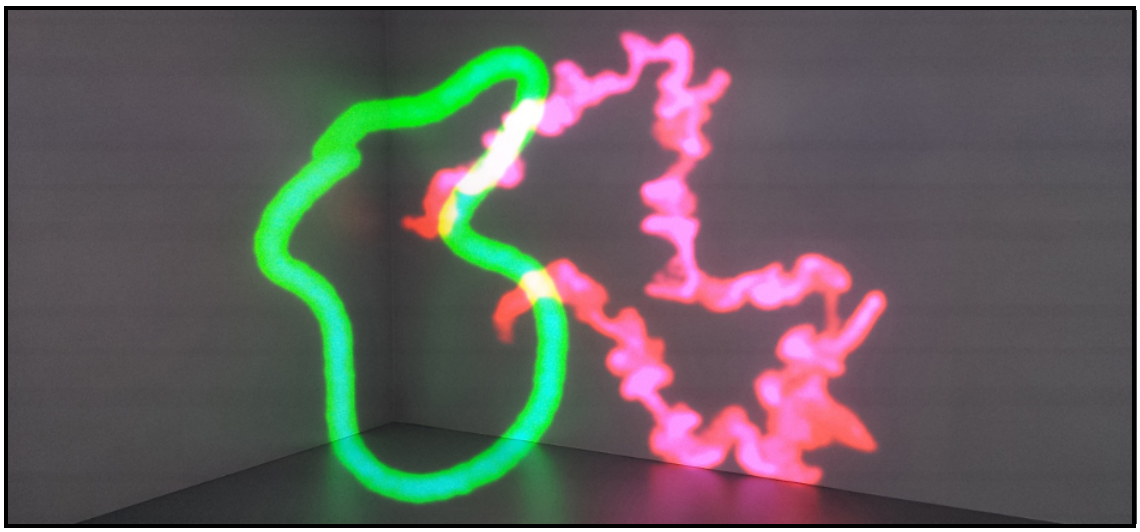


**Figure 2.7:** Strokes with different vorticities (green: low, pink: high).

#### 2.3.2.4 Source–Target Modulation

The brush and fluid parameters presented in the previous sections can be changed between brush strokes, giving rise to a variety of stylistic choices for the drawing tool. However, it might be desired by users to change parameters *while* performing a stroke, just as, e.g, a painter is able to modulate the tilt angle and applied pressure of a brush. To account for this, we enable users to dynamically change predefined parameters while performing a brush stroke: *source–target modulation*. The brush size, color, and density can be manipulated over the course of a single stroke.

Each of these parameters—the *targets*—can be controlled by different *sources*: the X/Y direction of the Flystick2's analog joystick and the brush stroke velocity. They can be bound to the target parameters with a weighting factor ranging from zero to one. Brush configuration as described in Section 2.3.2.2 sets the base values for the brush size, color and density attributes. When picked as targets for the modulation, brush size and density are added as offsets.

A single target can be modulated by multiple sources. When a modulation is active, the offset value is added with its respective weight. This leads to the computation of the final attribute value $A$ as the sum of $A_{base}$ and the per-modulation offset values $A_m$ weighted by the source attribute value $s_m$ and an individual weighting coefficient $w_m$:

$$A = A_{base} + \sum_{m \in M} s_m w_m A_m \tag{2.5}$$

Since subtracting color values from a base color can be non-intuitive depending on the color space, the color in the modulation settings is a target value instead of an offset. Thus, the brush color varies between the base color and one or more target colors. The interpolation is computed as a convex combination between the base color and the weighted target colors. A convex combination is a weighted linear combination of values $v = \sum_i w_i v_i$ with $w_i > 0$ and $\sum_i w_i = 1$.

The weights $\alpha_m$ for the convex interpolation are the product of the source modulation magnitude $s$ and the modulation weight $w_m$, normalized by the number of active modulations $N$:

$$\alpha_m = \frac{|s| \cdot w_m}{N} \tag{2.6}$$

For the convex interpolation weights to sum up to one, the base color weight $\alpha_b$ is calculated as one minus the sum of the target colors' interpolation weights. The sum of the target weights is in the interval $[0, 1]$ due to normalization and the fact that all $w_m$ and $|s|$ also lie in that interval. This leads to the interpolation formula of the resulting particle color $C$ as a convex combination of the base color $C_b$ and the per-modulation target colors $C_m$:

$$C = \alpha_b C_b + \sum_{m \in M} \alpha_m C_m \tag{2.7}$$

By using source–target modulation, artists have a tool to increase the expressiveness of their brush strokes. To give an example: the joystick X direction is bound to the brush size and the velocity of the brush stroke is mapped to a target color. Then both parameters can be varied simultaneously during a brush stroke by moving the joystick while drawing at different velocities. An exemplary outcome of a source–target modulation is depicted in Figure 2.8.

#### 2.3.2.5 Direct Particle Interaction

So far, we discussed interaction techniques related to the emission and behavior of particles. However, an important aspect is to manipulate the existing artwork and directly interact with the fluid. To this end, we add three different ways of direct fluid interaction, which we discuss in the following: hand manipulation, blowing manipulation, and erasing.

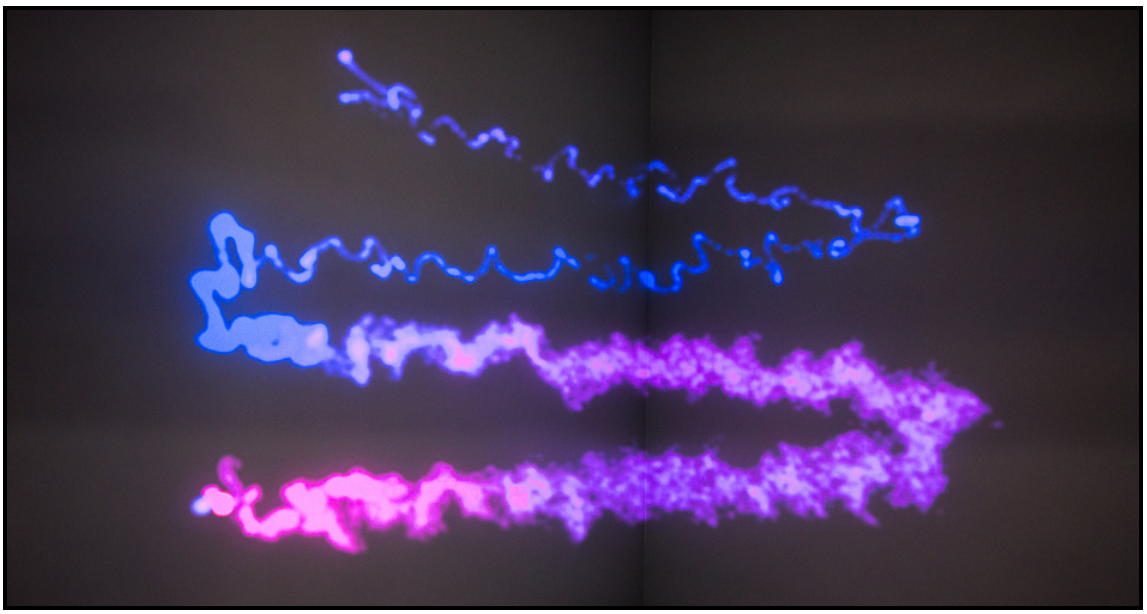


**Figure 2.8:** Variation of color and size via source–target modulation.

#### Fluid Manipulation by Hand

Artists can manipulate the drawn fluid sketch by moving their non-dominant hand through the particle population, as illustrated in Figure 2.9. This creates a spherical external influence (see Section 2.3.1.2), centered at the user's non-dominant hand.

Hand manipulation can be enabled and disabled via a menu. When enabled, users can push and drag the drawn fluid sketch in a natural and intuitive manner. The size of the hand interaction region is user-adjustable via a menu. Optional visual feedback of the interaction region is provided by a white solid sphere that matches the interaction region's position and extents.

A user-configurable strength factor is available in the menu. It determines the value of $f_{strength}$ in Equation 2.2. This allows the user to control the sensitivity, such that the hand interaction feels natural for the given viscosity of the fluid.

#### Fluid Manipulation by Blowing

Surveying techniques for real-world interaction with liquid colors, we found that blowing is regularly used for watercolor paintings to smear ink on the paper surface. Furthermore, turbulence in smoke, e.g., for smoke photography, can easily be created by blowing. Inspired by such techniques, we add a blowing interaction metaphor to Fluid Sketching,

**Figure 2.9:** The user manipulates fluid sketch with her non-dominant hand.

which is, to the best of our knowledge, a completely novel type of interaction for artworks in IVEs. An example of a blowing interaction is illustrated in Figure 2.10.

We realize this interaction metaphor by applying a conical influence (see Equation 2.3.1.2), originating from the user's mouth position. The force strength is determined by the amplitude of recorded sound from the wireless microphone the user is wearing. For the realization, we use an adapted version of BlowClick [Zielasko et al., 2015]. The

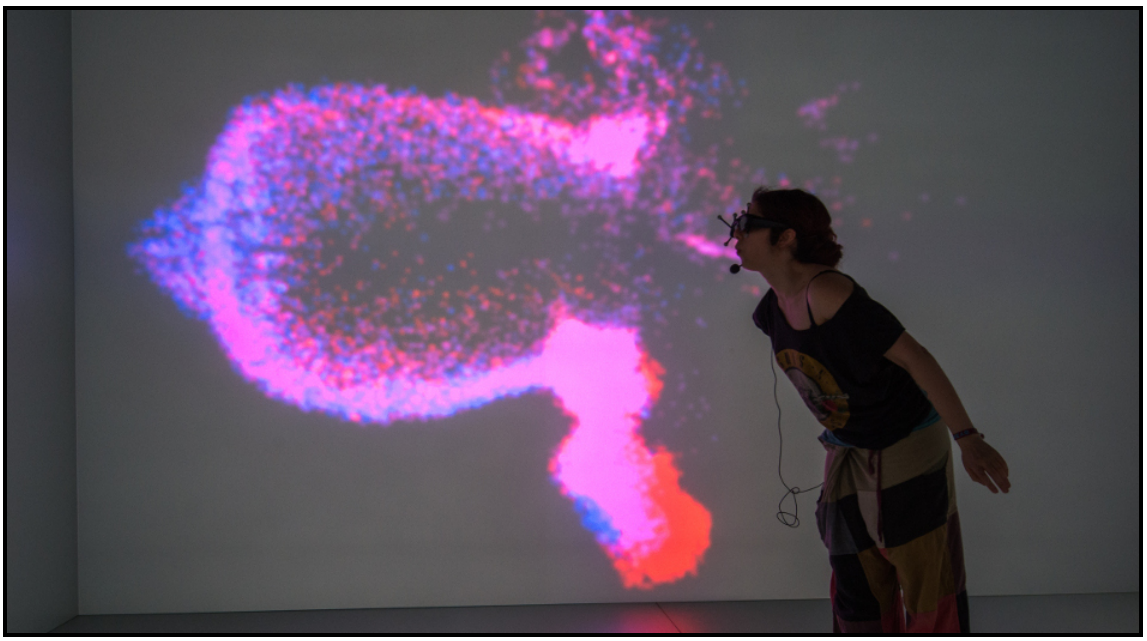


**Figure 2.10:** The user blows into the 3D fluid-like content using the microphone.

audio signature of blow events is recognized and compressed to a single strength value, which is exponentially smoothed over time. It is additionally multiplied with a user-configurable strength factor that allows for more user control. The result is fed to the fluid simulation as the effective strength $f_{strength}$ of the conical force. The opening angle of the conical force is 60°, which showed to be a reasonable width in prior informal testing. The height of the conical interaction region $h_{cone}$—i.e., the blowing interaction range—is user-configurable via a menu to grant users control over the distance at which particles are affected.

### Erasing Particles

When drawing on paper, an eraser is a vital tool to correct for mistakes. The same also accounts for our sketching system, which is why we add an eraser tool. The erase mode can be activated via the menu or a dedicated button on the Flystick2. The eraser is visualized as sphere centered at the non-dominant hand. Since the non-dominant hand is also used for shaping, the eraser sphere is colored in red for disambiguation. An example is illustrated in Figure 2.11.

The system offers the user to change the eraser size via the menu. By moving the hand through space, the part of the sketch that lies within the sphere volume of the eraser is deleted. This is realized by marking the respective particles inactive.

**Figure 2.11:** The user erases parts of the fluid sketch.

#### 2.3.2.6 Overall Scene

A challenge in creating real-world fluid artworks is that artists only have limited control over the dynamics of the diffusion process. We address this issue by providing a pause mode. When enabled, the diffusion process of the fluid gets temporarily disabled. Consequently, any particle movement stops and direct interaction with the sketch is deactivated. However, users can still draw or erase brush strokes, so even complex structures can be created without being distorted by diffusion over time. By deactivating the pause mode, the diffusion process continues for all particles. Pause mode is indicated to the users by changing the background color from light gray to dark gray.

While the pause mode grants users more control for working on details, it lacks the possibility of manipulating the existing sketch. Therefore, we additionally add the freeze mode. It also stops the diffusion process, but in contrast to the pause mode, sets the velocity of all existing particles to zero and marks them as frozen, which excludes them from the fluid simulation. Freezing only affects the fluid sketch that has already been drawn, but not newly emitted particles. To allow for local manipulations, frozen parts of the sketch can be woken up by interacting with them via hand or blowing. Additionally, particles that are affected by neighboring particles are woken up, too, which allows for the propagation of local modifications.

To save the drawn sketch for further processing, we add an export option in the OBJ file format. Thus, fluid sketches can be further edited in a 3D modeling application of the users' choice.

As illustrated above, Fluid Sketching offers a multitude of configuration capabilities. To allow users to easily restore already-used settings, we add the possibility to save and restore configurations as presets. Thus, artists are able to create their own toolset, which can grow over time and can be used to easily create more refined sketches in future drawing sessions.

#### 2.3.2.7 Rendering

The particle visualization in the Fluid Sketching system is realized as billboard rendering with additive blending. We chose this rendering style, because it offers a reasonable trade-off between efficiency and beauty. It enables us to handle large enough particle populations for creating plausible fluid sketches while providing an appealing view.

### 2.3.3 User Study

To get first feedback on the overall concept of Fluid Sketching, we conducted a qualitative user study. It serves as a guideline for further refinement of the interface concept, improvement of the usability, and the implementation of additional features. To this end, we recruited two representative user groups. Professional digital artists, being the primary target audience of the system, formed one group, in order to get feedback from a practitioner's perspective. VR experts from the fields of VR research and development formed the other user group, to get feedback on interaction and implementation aspects. In the following, we first outline the procedure of the study, before successively presenting and discussing the results.

#### 2.3.3.1 Procedure

We used the technical setup as described in Section 2.3.2.1. Participants were guided through the study by a supervisor, while an observer transcribed arising comments. The study consisted of five phases. These comprise filling out a pre-study questionnaire, getting an introduction, performing pre-defined tasks, free exploration, and filling out a post-study questionnaire.

The pre-study questionnaire captured demographic data and the participants' experience with VR technology. Furthermore, they were asked if any vision impairments were present. Additionally, a consent form was signed by the participants and they were informed about their right to quit the study at any time without justification.

Afterwards, the participants were equipped with the interaction devices, before entering the aixCAVE with the supervisor. Here, in a short instruction phase, the overall functionality and the interface of the system were explained. Additionally, the participants were instructed to think-aloud and comment freely on the Fluid Sketching system so that the observer could transcribe comments from outside the aixCAVE via microphones installed in the system.

In the next phase, the participants had to perform four sets of guided tasks. These tasks were related to the brush parameters (see Section 2.3.2.2), fluid parameters (see Section 2.3.2.3), source–target modulation (see Section 2.3.2.4), and direct particle interaction (see Section 2.3.2.5). Each task started with a systematic introduction to all parameters.

Afterwards, the free exploration phase started, during which the participants could try out the functionality of the system freely. In this phase, no further instructions were given, however the participants could ask questions and further comment on the experience.

Finally, the participants had to fill out a post-study questionnaire, comprising questions about feedback on the tasks and the opportunity to mention ideas for additional features. Moreover, the short version of the AttrakDiff 2 questionnaire [**?**] was handed out to obtain information on the user experience.

## 2.3.4 Results and Discussion

Overall, 10 subjects participated in the study, 4 of which were artists and 6 were VR experts. Among the participants, 2 were female and 8 were male, and all had normal or corrected-to-normal vision. Five participants were aged 25–34 years, four 35–44, and one 45–54. All artists stated that they had experience with VR systems before. Based on the transcripts, statements given by the participants were extracted. Similar statements were grouped to identify common feedback given by multiple participants.

### 2.3.4.1 Feedback on Existing Features

Regarding the configuration of the fluid parameters, three of the participants appreciated to directly see the effect of changing the fluid parameters. However, five participants stated that it is difficult to understand the parameters. One of them stated that “There are so many parameters, I can barely keep them in my mind”. While we expect that users would be able to get known to all parameters when using the application for a longer time, at least on first contact, the high number of different parameters might

actually be a considerable limitation. This could be addressed this in different ways. First, a number of presets for various fluid types such as honey or smoke might remove pressure from the participants to directly consider all parameters from the beginning. In addition, tooltips in the parameter menus, giving a textual explanation of and a visual example for each parameter could assist users in understanding and learning the parameters.

Two participants stated that it is difficult to see how far particles are since there are no lighting or depth cues. However, one participant perceived no disturbing artifacts due to the rendering style and further mentioned that she likes the over-saturation and that it encourages to move around in the tracked space. In future design iterations, this should be accounted for this by providing different rendering styles for particles.

Concerning direct particle interaction, the following statements arose. When the viscosity of the brush was high, three participants stated that manipulating the sketch with the hand feels like sculpting. A VR expert pointed out that it's possible to "smear" the sketch and that this allows for a lot of control for manipulation. The fact that artists compared the interaction technique to sculpting shows that it provides a direct and intuitive way of interaction. Furthermore, it was rated by a VR expert as giving a high degree of control. Overall we can conclude that it is a well-suited interaction type for Fluid Sketching.

Five participants stated that they do not need or like a sphere representation of the interaction range for hand interaction and that they had better depth perception without it. However, one artist found it comfortable to use hand interaction with the sphere visualization, stating that "It gives the illusion of having a precise tool". This indicates that making the sphere visibility configurable was the right choice, as it allows to account for different user preferences.

Regarding interaction with the blowing metaphor, nine participants stated that they liked it and two of them expressed that it "feels natural to use". Five participants mentioned that the interaction via blowing is fun. Based on these given statements, we can conclude that the feature is a valuable addition to the artists' toolset enabling a natural and intuitive type of interaction. However, three VR experts noted that the blowing interaction was hard to control and one of them stated that "It often leads to destructive outcomes". This issue could be addressed by adding the blowing strength as a modulation target for the source–target modulation. Thus, finer control over the blowing strength while performing the interaction would be granted. Furthermore, six participants reported that speaking also triggers the blowing interaction. This issue can be resolved by utilizing a more advanced technique, such as BlowClick 2.0 [Zielasko et al., 2017], which uses a neural network to differentiate between blowing and speaking.

For the eraser feature, three participants stated that the eraser sphere is occlusive, just as during hand interaction. However, one participant explicitly mentioned that the eraser

sphere was not disturbing compared to the hand interaction mode. Two participants suggested to draw only the sphere's outline or to make it transparent in order to mitigate the occlusion, which we also deem to be a good way of improving the visualization of the eraser and sphere interaction ranges.

Regarding source–target modulation, a total of eight participants explicitly mentioned that they liked the feature. Based on these statements, we observed that the second-most appreciated feature is the source–target modulation, complementing the intuitive blowing metaphor by a means of performing precisely controlled expressive drawing gestures. However, five of the participants reported that it is complicated and they could not remember their own modulations. To address this issue, we designed a custom interface to quickly and comprehensibly define the modulations, which is presented in Section 3.3 Moreover, two participants stated that the source–target modulation is hard to control, although conversely, one participant commented "It gives me more control and I can really mix the effects, great!". We assume that this difference might be due to the degree of training since some of the users know the interaction devices that were used for the study.

Regarding the scene related features, the following statements emerged for freezing and pausing. Two participants positively mentioned unfreezing by directly interacting with the sketch via hand or blowing. Furthermore, two VR experts stated that they found pausing is a valuable feature that offers more control. This indicates that these features were valued and deemed useful for having precise control over the amount of diffusion.

Concerning the overall system performance, the following statements arose. One participant stated that she was confused when the system got slower and lost interest in using it. Another participant pointed out to notice the "burden on the graphics card". So far, we did not perform in-depth profiling of our application, but this can be investigated in future iterations. However, it should also be noted that the study was carried out on several-year old hardware, and we expect it to perform better on modern hardware.

Considering all given comments, we can conclude that the interaction and scene control features that were developed for the Fluid Sketching system serve their intended purpose well. The goal of providing an artist's interface that allows for natural and direct, but also precise and expressive manipulation of fluid sketches was successfully achieved.

#### 2.3.4.2 Ideas for Additional Features

A goal of our user study apart from evaluating the current state was to gather ideas for improvements and additional features. Concerning overall scene related features the following suggestions were made. One artist requested a feature to import models to paint on. Two participants stated that an undo feature could be helpful for precise

drawings. Two participants would like to have a preview mode to inspect emitted particles without altering the piece of art. Two participants reported that it would be nice to have different rendering options. One participant asked for a heads-up display with status information about the scene. One participant requested a feature to save individual parts of the sketch.

Regarding the brush, the following statements emerged. One participant reported that occlusion problems with the Flystick were disturbing for small structures. Moreover, he asked for more control on where the particles are emitted and suggested to include a precise indicator for the particle emission location. One participant suggested to offer different particle shapes besides the default billboard.

Several suggestions were made for additional color selection and modification features. One participant asked for a feature to change the colors of the drawn fluid sketch. She suggested two different ways of achieving this. First, to add a paint brush that changes the color of existing particles. Second, to offer hue and saturation controls to globally shift hue and change saturation to see the sketch in a different color range. One participant stated that there should be an easier way to quickly change emission colors, and suggested to add a button that switches to a new color.

Regarding the hand interaction, three participants mentioned that they would like to use both hands for shaping the sketch.

These suggestions should be carefully considered for inclusion in future design iterations. While most of the mentioned features can be realized with reasonable effort, adding an undo feature imposes several difficulties due to the dynamic nature of our fluid simulation system.

#### 2.3.4.3 User Experience Questionnaire

The AttrakDiff questionnaire was used on a 7-point scale of semantic differentials with scores ranging from -3 to 3. Fluid Sketching is presented as self-oriented in the AttrakDiff portfolio with a mean of 0.43 for *pragmatic quality* and 1.50 for *hedonic quality*. The mean rating of *attractiveness* is positive with 1.95.

Based on these results, in terms of hedonic quality, Fluid Sketching is clearly above average which means it awakens curiosity, motivates, and stimulates the users. Nevertheless, the value of pragmatic quality is just above the average. Overall, the result indicates that there is room for improvement in terms of usability.

### 2.3.5 Conclusion

In this section, we presented a novel medium for creating 3D fluid artwork in IVEs. It enables artists to draw fluid-like sketches using a 3D brushing tool and offers natural interaction methods for shaping the drawn sketches. Procedural curl-noise was chosen to animate the fluid, which gives a high degree of animator control while having low requirements on memory and computation time. The fluid advection is performed by a GPU-based particle system, into which spherical and conical external velocities were integrated to realize direct hand and blowing interaction. To enable artists to perform expressive brush strokes, the source–target modulation was included.

By performing a qualitative user study with VR experts and artists, we were able to demonstrate the usefulness of our approach. Overall very positive feedback was obtained, indicating that the goal of providing a novel authoring technique for the creation of 3D fluid artwork was successfully achieved.

## 2.4 Fast and Precise Scene Authoring for Automated Vehicle Testing

*The contents of this section are based on – and taken in part from – work previously published in [Eroglu et al., 2021].*

In the previous section, we explored immersive authoring techniques that enable users to create and manipulate 3D fluid-like structures through procedural sketching. This demonstrated how procedural methods can support creative workflows by combining intuitive interaction with rule-based generation. In this section, we shift our focus to high-level scene authoring, where users concentrate on the intent and structure of a scene rather than on low-level details. This approach facilitates more efficient content generation, particularly in complex domains that require extensive scenario design. One such application is automated vehicle testing, where engineers must design and execute controlled traffic scenarios.

Real-life testing and validation of automated and autonomous vehicles is a challenging task regarding safety concerns and efficiency. Engineers have to consider unexpected traffic, road, and weather conditions when preparing testing scenarios. Designing and executing these scenarios may involve potential risks to the system itself, or the environment. Additional to these risks, the reproducibility of the scenarios is limited, and slow development and testing cycles become expensive for automotive companies [Wachenfeld and Winner, 2016]. To address these limitations, original equipment manufacturers

employ simulators of virtual traffic scenarios. This enables much faster development and testing cycles at reduced cost and without safety concerns. Recently, Nvidia introduced the commercial Nvidia Drive Sim and Nvidia Drive Constellation platforms to simulate testing of autonomous vehicles [NVIDIA, 2018]. They collaborate with rFpro which provides photo-realistic rendering of traffic scenes. Furthermore, Siemens recently announced their in-house autonomous driving performance validation and verification solution [Siemens, 2019].

VR offers a number of potential benefits for examination of the simulated scenarios and to perform system validation and verification [Mujber et al., 2004]. Immersive visualization enables improved spatial perception, e.g. for the realistic judgment of driving speeds and distances [Kemeny and Panerai, 2003]. Furthermore, efficient navigation in the 3D environment is possible and the scenario can easily be observed from varying perspectives. For instance, users can follow traffic movement from inside a vehicle and seamlessly switch to a pedestrian's perspective standing at a crossing.

An important building block of such virtual traffic simulations is the design of specific testing scenarios, which includes the creation of road networks and behavioral properties such as speed limits and traffic directions. These scenarios can be generated from real public data sources such as geographic information systems (GIS) [Krajzewicz et al., 2012], or by procedural generation [Galin et al., 2010]. In addition, user-controlled interactive design of road networks and traffic scenarios can be performed to test vehicles in custom environments [IPG-Automotive, 1999]. User-defined scenarios can be adapted to specific requirements and provide increased flexibility in their design.

However, to create such scenarios, engineers are limited to 2D desktop-based domain-specific tools for the design of road networks or common 3D modeling environments, which require adequate knowledge of these expert tools. Even with expertise on such tools, it is inefficient to perform a full design iteration including an evaluation of the scenario in VR. If an intermediate modification of the road geometry is necessary while being immersed, the user has to leave the VR environment and perform the modifications with a desktop-based tool. The model then has to be re-imported and the user has to re-enter the validation environment. This breaks the immersion and, more importantly, slows down the design and evaluation cycle, which costs time and effort.

To overcome these limitations, this work investigates *how an authoring technique can be designed to enable control engineers to effectively create and modify road networks for automated vehicle testing in VR?*

To answer this question, we developed a VR-based authoring environment that enables an iterative workflow of road network authoring and automated driving simulation inside a single IVE. Users can create and modify road networks at run-time via free-hand gestural interactions. Free-hand gestures are employed to support natural and intuitive interactions. Design iterations with direct feedback become possible, which enables a

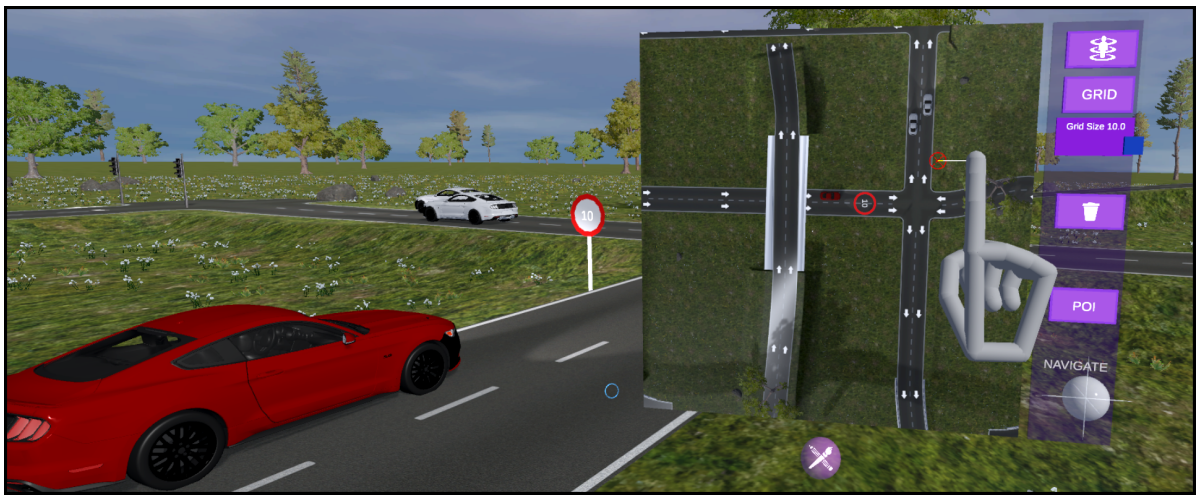



**Figure 2.12:** The user authors a virtual traffic scenario via 3D interactions on a 2D panel by using free-hand gestural inputs.

human-in-the-loop rapid evaluation and comparison of traffic scenarios.

Road network creation is an inherently 2D interaction task. To define a road network, the start and end points of each road segment need to be defined. The height of the terrain, however, is irrelevant for the resulting network topology. Therefore, we designed a control widget that enables this 2D interaction task in a 3D environment by orthographic rendering of the terrain. It is used for road network manipulation as well as an efficient means to navigate the scene. The 2D scene control panel is embedded into the 3D environment and can be placed freely relative to the user. We devised a novel 3D interaction technique for indirect placement and selection of objects on the 2D terrain surface. It is thus possible to perform manipulations of the road network, while immediately observing the result in the 3D surrounding.

To show the effectiveness of our approach, we evaluated possible design alternatives and performed a comparative user study. Existing work on free-hand road network creation in VR proposes a direct manipulation interface that transforms the scene via two-handed gestural interactions [Côté and Beaulieu, 2019]. Objects are placed directly onto a scaled-down version of the terrain, which is then scaled up again to perceive the surrounding in real-world size. In contrast to their approach, we enable users to do simultaneous manipulation and perception of the virtual environment via indirect interaction techniques by utilizing a 2D scene control panel.

Besides quantitative data on precision and task completion time, we furthermore gathered qualitative feedback on our overall system design from a heterogeneous group of participants that included VR experts and control engineers. The insights we gathered from the qualitative evaluation will serve as a basis for further improvement of the

system.

In the following Section 2.4.1, we provide a detailed description of the proposed authoring technique and its system. The user study is described in Section 2.4.2. The results of this study are presented in Section 2.4.3, followed by a discussion in Section 2.4.4. Lastly, the limitations of the proposed approach are discussed in Section 2.4.5.

### 2.4.1 System Description

To enable users to author road networks and driving scenarios, we developed an interactive VR-based authoring environment. The system will be used by novice users who might lack considerable experience with 3D interfaces. We, therefore, designed a natural user interface based on free-hand gestures to make the application as approachable and intuitive as possible. See Section 2.4.2.2 for the technical setup of the system.

For the interactive creation of road networks, various operations need to be implemented. Specific requirements were iteratively developed in close collaboration with control engineering experts of Ford Motor Company. First, the system needs to enable users to create, delete, and cut road segments, add crossings and bridges, and modify the properties of these elements, such as the curvature or slope of a road segment. The second requirement is to define parameters for the later simulation. Thus, our system enables users to change the traffic direction and add static objects, vehicles, and speed limits while creating the virtual environment as well as during the testing phase. The third requirement is that users need to efficiently navigate through the environment during scene authoring. Therefore, users can navigate via flying, teleportation, or by using a minimap technique. Further, our system enables saving and loading of the created scenes, to continue the design at a later time or export the scenario for offline editing. Moreover, to assist users in employing the system, video tutorials for each feature are provided that can be watched inside the virtual environment. In the following, our system is explained in detail.

#### System Menu and Control Interface

To enable novice users to interact with our system, a natural user interface based on free-hand interactions is implemented. Gestures executed by the user serve as the system input. Although gestures can be defined by the whole body of the user, we focus on using hand, finger, and palm. We find that a sitting position is the right setup for interacting with our system since authoring road networks in VR can be tiresome if the user has to stand for hours [Bellgardt et al., 2017; Zielasko and Riecke, 2020]. In the following, we describe the interaction with the UI elements and the scene control panel.

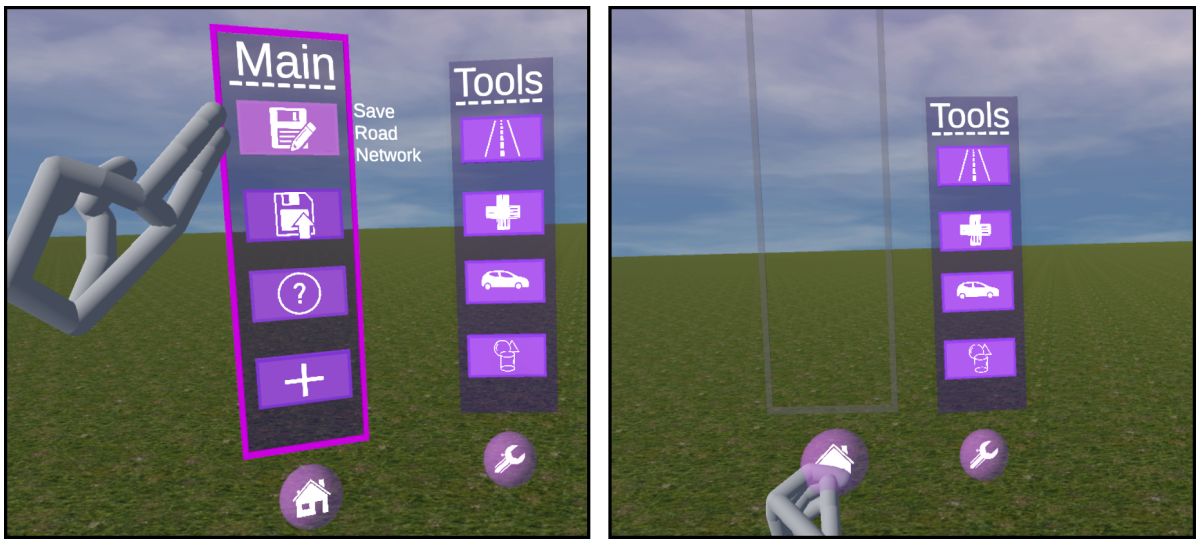



**Figure 2.13:** Adapted 2D menus. Left: Hovering over a menu button. Right: Grabbing the menu handle.

**UI Elements** Because we aim to address a large number of features, adapted 2D menus are implemented, as suggested by Laviola et al. [LaViola Jr et al., 2017]. One of the concerns about floating menus in VR is the occlusion with the environment. However, we addressed this issue by enabling the placement of floating menus freely relative to the user by dragging a sphere handle. Upon grabbing the handle, the attached floating menu is minimized to provide an unobstructed view of the scene, and the semi-transparent frame will be visible to guide the arrangement of the menus in the environment (see Figure 2.13). Furthermore, the floating menus can be anchored to the user's left hand.

Upon rotating the left palm towards the eyes, two menu items become visible. These items can be detached by pulling them away from their anchor. To enable users to interact in a fast manner, these handles snap to their anchor location when they are in a defined range. This is visualized by interpolating the color of the handle from blue to magenta. Furthermore, multi-modal cues, visual and auditory, are provided to substitute for the lack of haptic feedback during interaction with the menu buttons [Bowman et al., 2001].

**Scene Control Panel** The planning of real-world road networks by civil engineers typically starts from a 2D blueprint. To match this common workflow, we designed a 2D interface that enables users to create road networks in VR (see Figure 2.14). The *scene control panel* consists of a minimap, a navigation panel, and a tools menu that can be placed freely in the virtual environment after detaching from its anchor. Via the navigation panel, the user can activate teleportation, zoom in/out, and panning on the minimap. The minimap, a 2D World in Miniature [Stoakley et al., 1995], shows an orthographic projection of the scene, and users can add, select, and delete objects by

interacting with it. Furthermore, altering the properties of objects is possible via the property menus that can be enabled by a left pinch gesture. The property menu spawns in front of the minimap and is drawn semi-transparently to enable users to observe the changes during interaction.

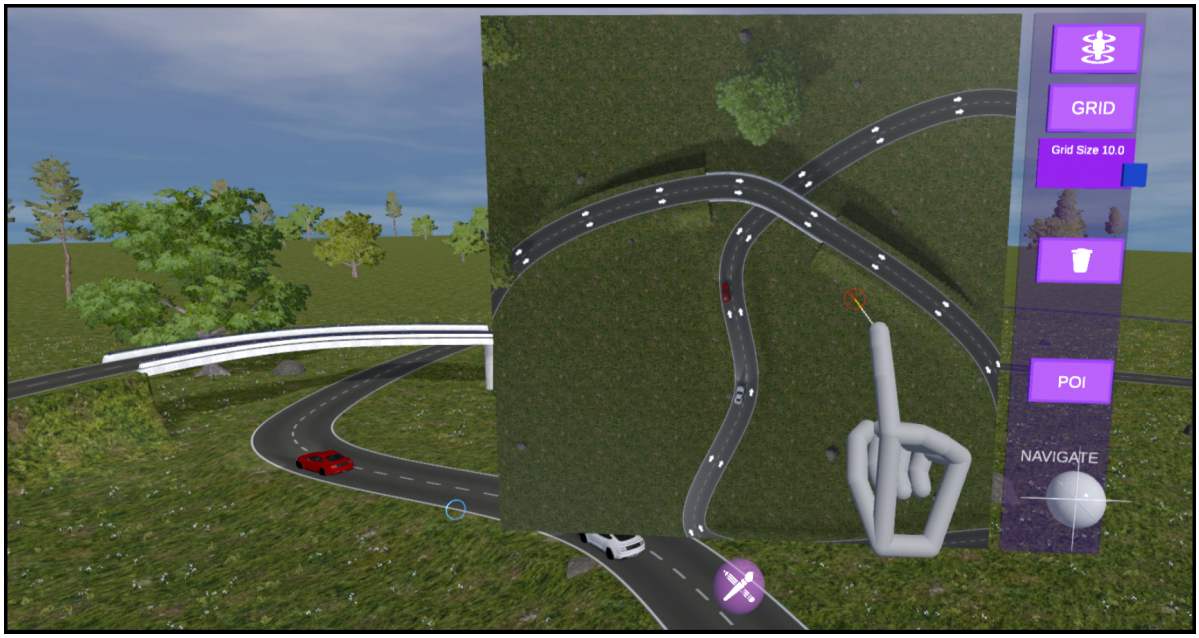



**Figure 2.14:** Interaction with the 2D scene control panel.

### Travel

To address our requirement of efficiently navigating the scene, we implemented three travel techniques that are in the category of steering-based and selection-based techniques [LaViola Jr et al., 2017].

Flying [Robinett and Holloway, 1992] is realized via a multi-modal interaction technique to improve the user experience [LaViola Jr et al., 2017]. It is enabled and disabled by giving "fly" and "stop" voice commands, respectively. After enabling the technique, the position of the user is continuously translated into the pointing direction of the stretched right hand. Furthermore, flying speed can be manipulated by changing the distance between the knuckle of the index finger and the thumb of the right hand.

Teleportation [Bozgeyikli et al., 2016] is realized by instantly changing the location of the user to a position that is selected on the minimap. Furthermore, the view of the minimap can be changed via the *navigation widget* that is located on the bottom right of the scene control panel (see Figure 2.14). By dragging the sphere handle from its resting position, zooming in/out and sideways panning can be performed.

### Object Selection and Manipulation

To address our first requirement, a user interface for the selection and manipulation of objects in VR has to be provided. Therefore, we designed an interaction technique for the selection and manipulation of objects via the minimap on the scene control panel. It provides an efficient tool for 2D selection and placement of objects on the terrain. The interaction is based on free-hand gestural input and illustrated in Figure 2.15. For a better spatial perception of the manipulated objects, we furthermore enable users to perform interactions in 3D space.


**Figure 2.15:** Left: The ray, which is segmented into 3 regions (red, yellow and white represent add, select and idle modes respectively), is drawn perpendicular to the 2D panel. Right: Upon selection of an object.

The selection and placement of objects is performed based on motion of the right index finger towards the minimap. The specific action that is triggered depends on the distance of the fingertip to the 2D panel. A line that represents the pointing ray is drawn perpendicular to the panel, and indicates the active range for the different actions to the user. It is drawn as 3 colored segments and has a cursor, drawn as a red cross (see Figure 2.15 Left). The colored segments represent the different actions that are triggered when the tip of the index finger meets with the respective segment. White, yellow, and red-colored regions on the line represent the *idle*, *select*, and *add* modes, respectively.

An object is added to the virtual environment when the tip of the index finger reaches the red segment of the line and touches the cursor. After an add operation, the indicator line becomes invisible, and the user needs to pull back his index finger to the distance at which the idle mode is defined to perform further interactions. This *push-pull gesture* is used to avoid accidental add operations.

To select an object, the user moves the index finger into the yellow segment of the indicator line (see Figure 2.15 Right). Selected objects are highlighted in a red color. Different actions can then be performed on the selected objects via the control panel, such as deleting, splitting or merging of road segments.

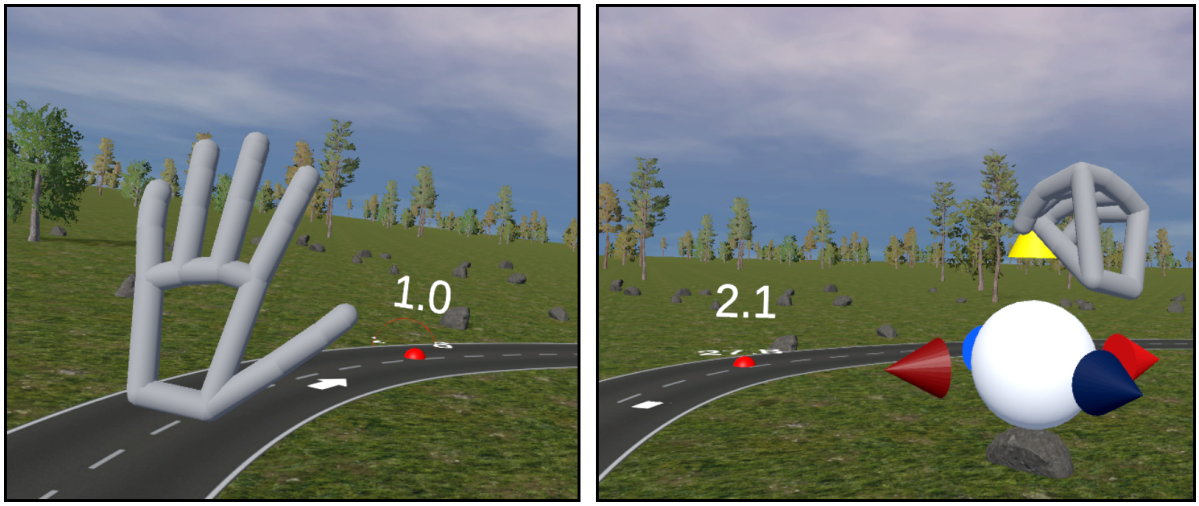



**Figure 2.16:** The flashlight pointing selection (Right) and manipulation of control points via the widget (Left) in the system.

Apart from the indirect interactions via the scene control panel, all objects in the virtual environment can furthermore be transformed via 3D selection and manipulation. For 3D object selection, we use the flashlight pointing technique, which uses a conic selection volume to select small and distant objects [LIANG and GREEN, 1994]. In our system, the apex of the cone is located at the palm of the left hand, such that objects can be selected by pointing the left palm towards an object.

To perform accurate manipulation of an object's position, we employ a transformation widget instead of a direct mapping to the user's hand motion (see Figure 2.16). By dragging the widget via the axis arrows, the position of the selected object can be manipulated.

### Road Network Authoring

Our system enables users to author road networks and traffic scenarios via placement and manipulation of objects in the virtual environment. This addresses our primary requirement for interactive road network creation. Objects can be roads, crossings, bridges, speed limit signs, static scene objects and vehicles.

Regarding road geometry, the default shape of a road segment is a Catmull-Rom spline,

which is defined by adding control points onto the terrain. The user can add a continuous sequence of control points to create connected road segments. By adding the last control point close to the starting point of the road, closed-loop road tracks can be created. The creation of a new, separate road segment can be activated via the tools menu. To enable users to create road segments with zero curvature, an underlying grid is implemented onto which the added control points will snap. The size of the grid can be altered and its visibility can be toggled via the scene control panel.

Our system enables users to interact with the road network control points by means of translating, deleting, adding onto the road segment, or changing the type of control point via the properties menu. Altering the control type has an effect on the shape of the road segment which is generated between the selected markers and the next one. Besides splines, arc segments with constant curvature, as well as straight-line segments can be generated.

To define a road network, the creation of different types of crossings is needed. In our system, the user can add crossings and bridges onto the terrain or onto a road segment. When added onto a road segment, the road will divide into two separate segments that are connected to this crossing. If the added crossing object is close to the beginning or end of an existing road segment, it will snap to the control point and get connected automatically. Road segments that originate from the crossings can be generated. To achieve this, the user first selects the crossing and then adds the road control points. The default crossing shape is an "X" crossing to which 4 separate roads can be connected. It can be switched to a "T" crossing via the crossing properties menu.

For further editing of the created road, the system enables users to cut the road into two separate segments via the minimap. Moreover, separated tracks can be connected by first selecting one of the tracks and then adding a control point close to another road segment. Possible connections are indicated by a floating "Connect" text and by highlighting the target control point. Thereby, the user will have an understanding of his actions. Connections can be created between two separate road tracks, two crossings, or between a crossing or bridge and a road track.

The user furthermore can add speed limit signs at a desired world position via the minimap. Upon adding a speed sign, a small widget will spawn to set the limit (see Figure 2.17). Via this widget, the user can increase or decrease the limit in steps of ten. After confirming the limit, a speed sign with the chosen limit becomes visible on both sides of the road. Undesired speed signs can be selected and deleted via the control panel.

Our system enables users to change the traffic direction of a generated road segment while authoring the environment, or even during the testing phase, which is not possible with existing simulation tools [IPG-Automotive, 1999; Intuition, 1990]. The traffic direction is visualized as an arrow on each lane of the road and can be altered between right-hand,

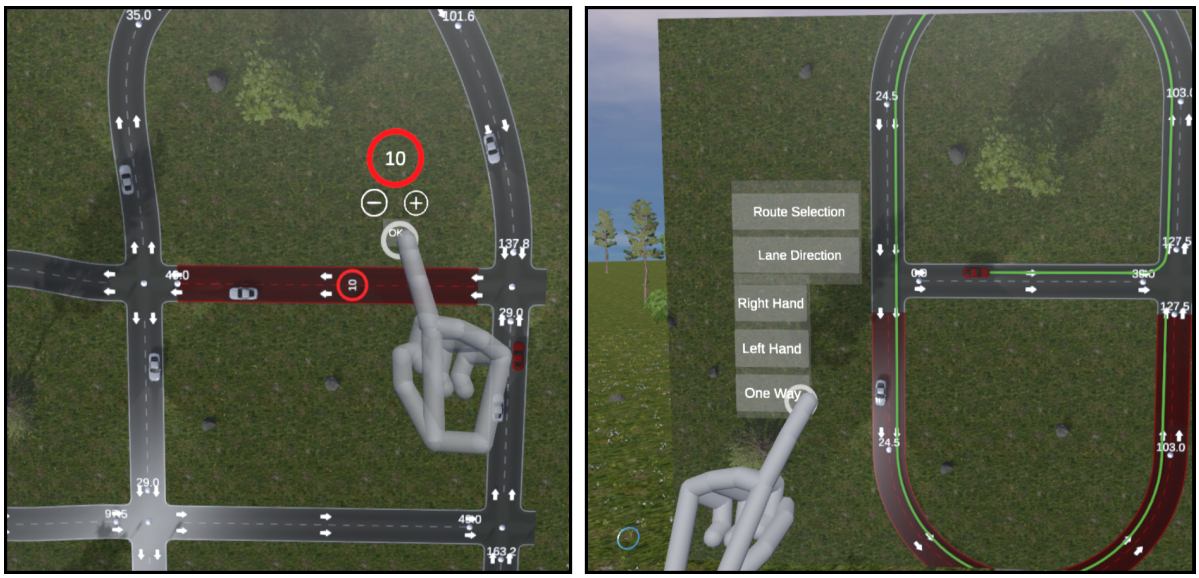



**Figure 2.17:** Left: The speed limit user interface. Right: Traffic direction setting via the property menu.

left-hand, and one-way traffic. Currently, the user can create roads with two lanes, and the direction of these can be changed via the road properties menu (see Figure 2.17).

In the simulation, the system needs to identify dead ends and grant a right or left turn at intersections. Therefore, the underlying road network topology is represented as a directed multi-graph [Bollobás, 2013]. Crossings are defined as vertices and each lane of the road is expressed as a directed edge. By using this underlying multi-graph representation, an automated vehicle can perform route planning during the simulation.

Vehicles can be placed onto a lane via the minimap. The orientation of the vehicle is adapted based on the traffic direction. Furthermore, the user can add static scene objects such as rocks onto the scene. This enables testing scenarios in which the vehicle encounters an unexpected object on the road while the simulation is running.

### 2.4.2 User Study

Since our goal is to provide a tool that can be used by domain experts in a professional environment, we seek to provide an interface for road network authoring that is fast and efficient. Furthermore, the precise placement of objects and road network control points is required for the creation of testing environments for automated driving. At the same time, the system needs to be approachable by novice users and provide a good user experience. Therefore, we consider *speed*, *accuracy*, and *usability* as the three major

relevant aspects. To this end, we compared our indirect interaction technique to an existing approach based on direct free-hand interaction. In addition, we are interested in general feedback on the usability of our system. Therefore, we conducted an empirical evaluation in three parts.

In the first two parts, we focused on the comparison of the two techniques, where the direct method is highly inspired by the VR tool described by Côté et. al [Côté and Beaulieu, 2019]. This approach enables users to create roads by adding control points into the virtual environment by directly touching the terrain. The comparative study aims to investigate whether our indirect approach improves user performance.

Therefore, we investigate the following hypotheses:

**H1**: Regarding the task completion time, indirect interaction will be faster than direct interaction.
**H2**: Regarding the precision, indirect target selection will be more accurate than direct selection.
**H3**: Users will have a higher preference to use the 2D panel for road network creation in VR.
**H3.1**: A top-down view alongside the 3D scene rendering improves the spatial understanding of the scenario.

Considering previous findings in the original work and based on our preliminary discussion, we assume that the direct method is more suitable for adding objects to the environment. Thus, we further hypothesize:

**H4**: Direct interaction will be more preferable by the users to add objects into the VR environment.

In the third part of the study, we gathered qualitative feedback on the overall functionality of our system. This part serves as a guideline for further improvements of the interface concepts, refinement of the usability, and the implementation of additional features.

#### 2.4.2.1 Study Design and Tasks

To test **H1** to **H4**, we created a one-factorial within-subject design where our two conditions (**Indirect** and **Direct**) were implemented. For both conditions, two scenarios were designed. In the first scenario, we measure task completion, selection, and navigation time, while the second scenario measures precision only.

The experimental conditions were carefully designed to be as comparable as possible.

The order of the **Indirect** and **Direct** conditions was counterbalanced across the participants, while the specific target sequence was kept identical. Regarding navigation, we applied the same scaling factors to match the effective distances the user's hands need to travel. Thereby, the measured effects, which are described below, are due to the interaction technique, and the influence of other variables is minimized.

Regarding task completion time (see **H1**), the task is to add a sequence of control points onto predefined target locations, while being as fast as possible. To achieve this, both selection and navigation have to be performed. To make the user employ the travel technique for zooming in on a target, we disabled target placement as long as the view is zoomed out too far. Participants first have to navigate to the target location that is indicated with 2 circles with different colors and size (see Figure 2.18). While the size of the outer circle is animated in a repeated manner, the inner circle has a static size. The participants could only add a control point to the target location as soon as the inner circle became visible based on the zoom or scale level. Task completion time is counted after the first control point has been added to the target location.

Regarding the precision tasks (see **H2**), participants were instructed to be as precise as possible while adding the sequence of control points without time restriction. We disabled the zoom-in/out features so that the target sizes and distances are identical for both conditions, and the precision values are not affected by any other factors. The target location is rendered with an animated circle and an "X" icon. The participants were instructed to add a control point to the center of the "X" icon.


**Figure 2.18:** Task completion time (Left) and precision tasks (Right) for the **Indirect** condition.

In the condition **Indirect**, the participants employed the minimap on the 2D scene control panel to add control points, and the navigation widget to navigate on the minimap (see Section 2.4.1). In the training phase, we instructed the participants to position the navigation widget wherever it feels comfortable by performing a pinch gesture at the desired location. This location then stays fixed throughout the task.

In the condition **Direct**, adding objects is achieved by touching the indicator, attached

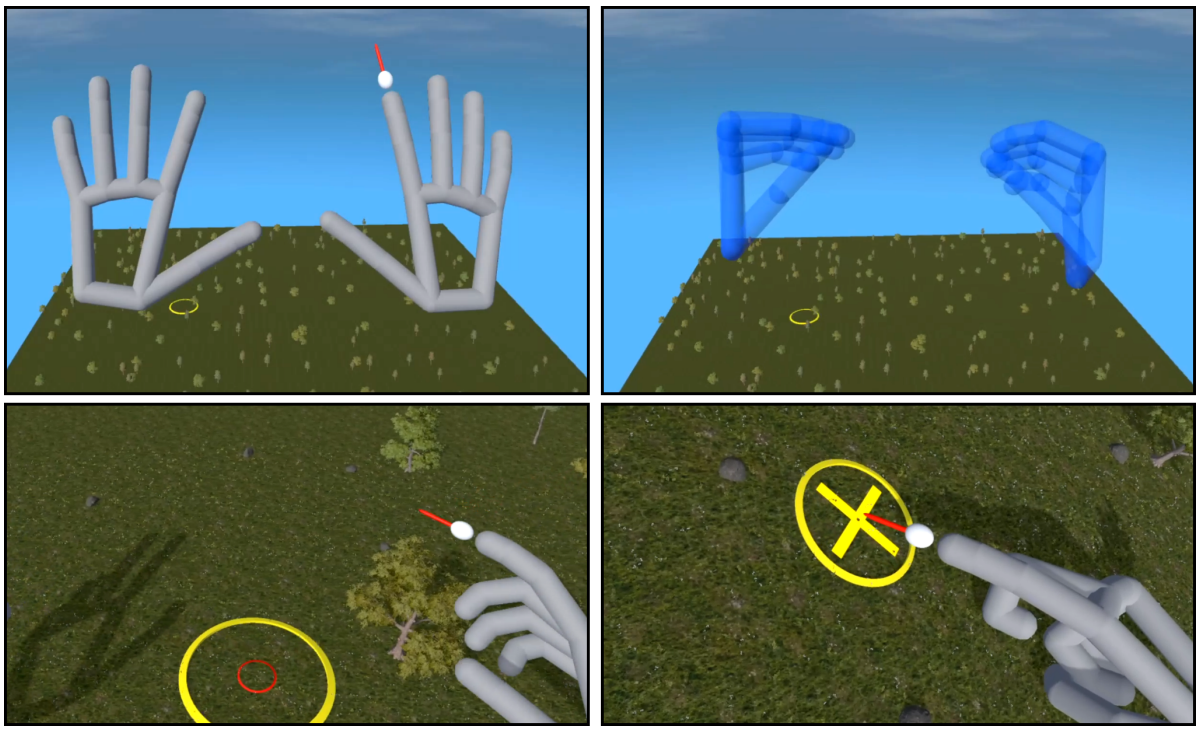


**Figure 2.19: Direct** interaction hands for selection (Top-Left) and navigation (Top-Right).Task completion time (Bottom-Left) and precision tasks (Bottom-Right) for the **Direct** condition.

to the right index finger tip, onto the terrain (see Figure 2.19). Navigation is performed by using two-handed gestural input. It is enabled when the user performs a grab gesture with both hands. This is indicated by changing the color of the hands to blue. Panning in the virtual environment is realized by dragging the virtual world into the desired direction. Moving the grabbed hands apart executes zoom-in and moving towards each other zooms out.

Regarding **H3** and **H4**, a subjective questionnaire was designed to be ranked by the participants after each of the **Indirect** and **Direct** conditions was completed.

Finally, for the qualitative study, we designed a set of guided tasks in which the participants had to create and manipulate a road network, along with performing dynamic changes in the scene while the simulation is running. To validate our hypothesis **H3.1**, another subjective questionnaire (see Table 2.1) was prepared to be ranked after the qualitative study.

#### 2.4.2.2 Apparatus

The study took place in our lab and was conducted in a seated position. The HTC Vive and HTC Vive Pro HMDs were employed, and both of them tracked with two tripod-mounted Lighthouse 1.0 base stations. To track the user's hand, we employed the Leap Motion Controller (LMC) using the 4.0.0 version of the Leap Motion SDK. The LMC was mounted onto the VR headset. The experimental platform was developed in Unity 2018.2.14f1, running Windows 10 on a 3.50GHz Intel Xeon E5-1650 with NVIDIA GeForce GTX 1080 GPU.

#### 2.4.2.3 Procedure

We conducted our study during the Covid-19 pandemic and prepared the study with health considerations by following the guideline presented by Steed et al. [Steed et al., 2020]. Since varying tracking stations and frame rates would cause inconsistency in our measured data, we used a uniform tracking station and a desktop PC. Furthermore, since the disinfection of headsets using wipes is not a safe solution, each participant used a headset that is not shared by anyone else on the same day. Therefore, we waited 72 hours, as suggested by the current study [Van Doremalen et al., 2020], to use the respective headset again for the next participant besides cleaning it with disinfectant wipes.

The rest of the study procedure is described in detail in the following. After the participants signed a consent form, they had to answer pre-study questionnaires regarding demographics, prior experience with 3D User Interfaces (3DUIs), video games, VR, and free-hand interaction in AR/VR, as well as a Simulator Sickness Questionnaire (SSQ) [Kennedy et al., 1993].

At the beginning of both **Indirect** and **Direct** conditions, instruction videos were shown that explain the steps and how to interact with the environment. Afterwards, a training phase was performed. This phase was unrestricted in time and trials and lasted until the participant felt comfortable with the respective technique. Then, participants executed 4 runs of task completion time and 4 runs of precision tasks. For each run, 5 target locations are defined for the completion time task, and 4 for the precision task, respectively.

To avoid fatigue, each condition was designed to last under 10 minutes and took about 8 minutes on average. After each condition, the participants took off the HMD and answered a System Usability Scale (SUS) questionnaire [Brooke et al., 1996] and a 7-point Likert scale questionnaire to give subjective feedback on the used technique. Afterwards, the participants performed a set of guided tasks to create a road network and evaluate the

**Table 2.1:** Subjective questionnaire for the overall system.

| | |
|---|---|
| **Q1** | I would use this system to create a road network in VR. |
| **Q2** | The 2D scene control panel with free-hand interaction allows me to create a road network in VR fast and efficiently. |
| **Q3** | The 2D scene control panel with free-hand interaction allows me to create a road network in VR with high precision. |
| **Q4** | The 2D scene control panel with free-hand interaction allows me to create a road network in VR with low effort. |
| **Q5** | I find panning and zooming the 2D scene control panel using the navigation widget efficient. |
| **Q6** | I find panning and zooming the 2D scene control panel using the navigation widget intuitive. |
| **Q7** | Having a top-down view alongside the 3D scene improves my understanding of the scenario. |
| **Q8** | Dynamic changes to the traffic scenario (tempo limits, lane directions) while the simulation is running enables me to evaluate different scenarios more efficiently. |
| **Q9** | I find changing the slope of the road using the manipulation widget convenient. |

functionality of our system. These tasks consisted of adding and connecting crossings, bridges, and road segments, adding a vehicle onto a road, running the simulation, and adding speed limits while the simulation is running.

After the guided tasks, the participants could try out the functionality of the system freely. In the guided and free exploration phase, the participants were instructed to think-aloud and comment freely on the system so that the observer could transcribe the comments. Lastly, the participants were instructed to answer a 7-point Likert scale questionnaire (see Table 2.1) related to the overall system and a second SSQ. Overall, the study took about 60 min.

#### 2.4.2.4 Participants

16 subjects (1 female and 15 male, mean age = 31.3 years old, SD=6.45) voluntarily participated in the study. Among the participants, 4 were professional control engineers, 8 were VR experts and 4 were students. All participants were right-handed and reported normal or corrected-to-normal vision. Regarding their prior experience with 3D user interfaces as well as VR, 10 participants reported regular usage and 6 used it at least once before. For the video gaming experience, this distribution was 12 to 4. Furthermore, concerning the experience on the free-handed interaction in VR/AR, 3

| | Task Completion Time [s] | Navigation Time [s] | Selection Time [s] | Precision [m] | SUS |
|---|---|---|---|---|---|
| **Indirect** | 30.49 (± 7.43) | 26.37 (± 6.01) | 4.12 (± 1.87) | 0.53 (± 0.31) | 80.31 (± 11.72) |
| **Direct** | 47.90 (± 19.18) | 42.99 (± 18.53) | 4.91 (± 1.96) | 0.93 (± 0.45) | 57.96 (± 27.78) |

**Table 2.2:** Mean and standard deviation of task completion time, navigation time, selection time, precision and the SUS score are presented for **Indirect** and **Direct** conditions.

participants reported regular usage, 9 used it at least once and 4 had no experience at all. The SSQ reported an average score of 11.2 (SD=14.5) after the experiment and 1.1 (SD=10.3) as a difference score between before and after the experiment.

### 2.4.3 Results

We analyzed the scaled measures using a paired samples t-test since our study has a repeated-measures design. To ensure the effectiveness of the presented result of the t-test, we also reported the effect size by calculating Cohen's $d$ [Cohen, 2013]. Regarding the effect size $d$, 0.2, 0.5 and 0.8 are considered small, medium and large respectively. The subjective questionnaires for both conditions (**Indirect** and **Direct**) were analyzed using a Wilcoxon Signed-Rank test. Due to a wrong assumption for evaluating error selections, our results do not include error rates on the selection tasks. For all tests, we assume a significance level of .05.

#### 2.4.3.1 Time and Precision

For the condition **Indirect** and **Direct**, we recorded task completion time, navigation time, selection time and precision per participants. The results are shown in Table 2.2. A paired t-test shows that **Indirect** significantly outperformed **Direct** regarding Task Completion Time ($T(15) = 4.519, d = 1.13, p<.001$), Navigation Time ($T(15) = 4.330, d = 1.08, p<.001$), and Precision ($T(15) = 4.065, d = 1.01, p = .001$). However, regarding Selection Time, results show that there is no significant difference between the techniques ($T(15) = 1.109, d = 0.27, p = .285$).

#### 2.4.3.2 Subjective Feedback

For the **Indirect** and **Direct** conditions, the participants were asked to answer a SUS and a subjective questionnaire with 7-point Likert scale items. The SUS score for each

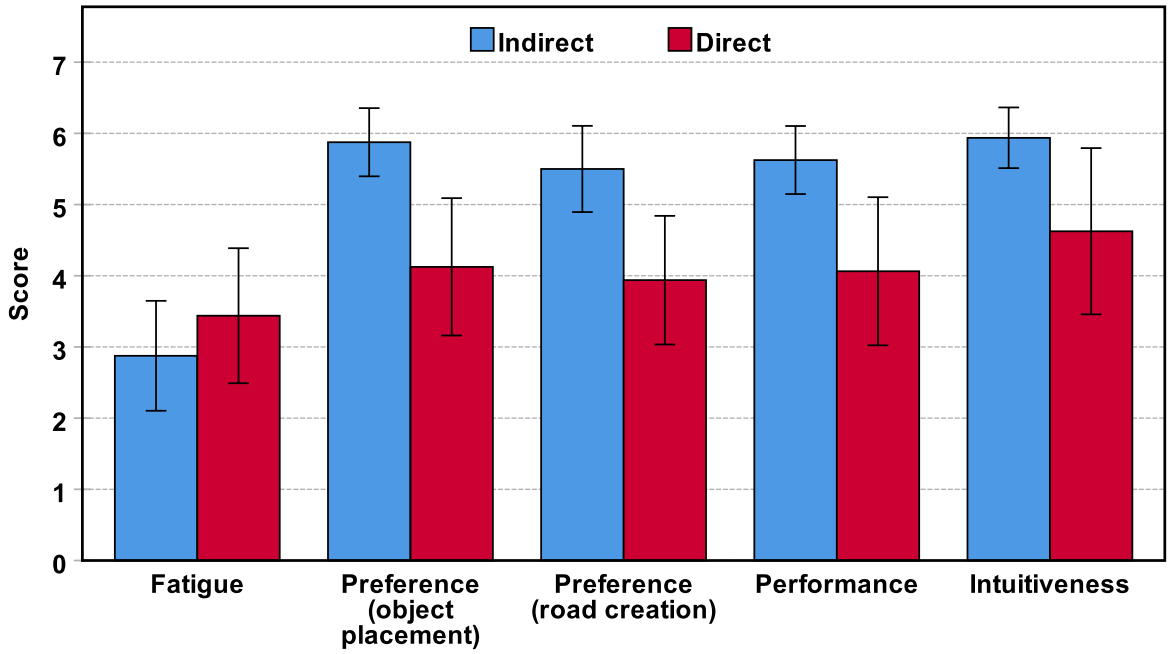



**Figure 2.20:** Subjective feedback that is ranked for **Indirect** and **Direct** conditions. Error bars represent the standard error of the mean.

condition is reported in Table 2.2. The 7-point Likert scale items are ranged from very strongly disagree (1) to very strongly agree (7). The results are shown in Figure 2.20. A Wilcoxon Signed-Rank test was conducted and shows a significant effect of the technique on Preference (object placement) ($Z = -2.523, p = .012$), Preference (road creation) ($Z = -2.760, p = .006$), Performance ($Z = -2.452, p = .014$), and Intuitiveness ($Z = -1.965, p = .049$). However, no significant effect was found on Fatigue ($Z = -0.988, p = .323$).

Concerning the guided task-based condition and free-exploration phase in the main application, the participants were asked to score another subjective questionnaire with 7-point Likert scale items (see Table 2.1). The results can be seen in Figure 2.21.

#### 2.4.3.3 Qualitative Feedback

Overall, most of the participants liked the system. One of the participants stated, "It is nice and fun" and one control engineer stated, "I think I can spend hours here, it is really interesting". Furthermore, three control engineers stated that configuration of the virtual environment is very fast and easy compared to *CarMaker* [IPGAutomotive, 1999], a commercial software that is commonly used by automotive manufacturers. However, they also stated that *CarMaker* provides considerably more overall functionality

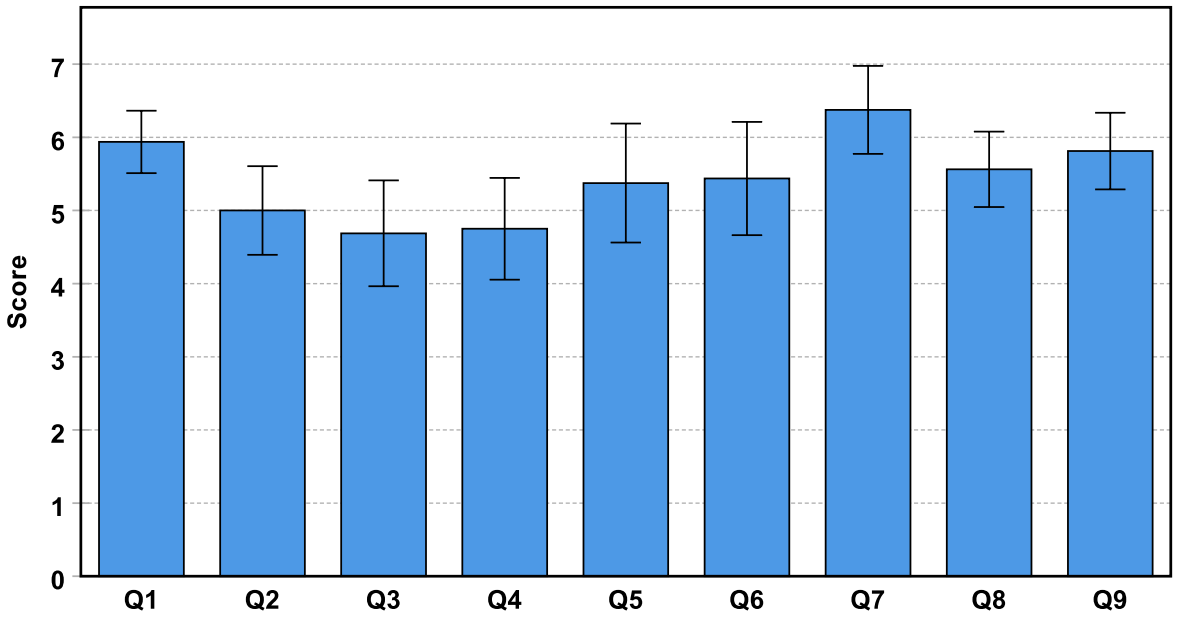



**Figure 2.21:** Subjective feedback that is ranked for the system. Error bars represent the standard error of the mean.

compared to our system. Furthermore, being able to save the scene in XML format, which can be loaded into the system again, was appreciated by the control engineers, since offline editing is possible.

Regarding the interaction with the 2D scene control panel, many participants appreciated to be able to immediately see the creation in the virtual environment. One of them stated, "Being able to manipulate in 2D and seeing it directly in the 3D environment gives me good feedback, and you have a better understanding of the 3D environment". Furthermore, some of the participants found the interaction with the 2D panel "intuitive".

Concerning the 2D scene control panel, four participants liked that the panel spawns upright and facing towards the user. Furthermore, one control engineer appreciated that he can position the menu items and the 2D panel freely in the environment. However, being able to tilt the panel was requested by two participants.

Regarding menu button interactions, six participants stated that toggle functionality should be employed for objects that are expected to be added multiple times into the virtual environment.

Connecting crossings functionality was grasped easily and was intuitively performed by most participants. Although our system provides textual feedback about possible

connections, three VR experts reported that the snapping range was not evident and proposed a visual indicator. However, after enabling the visibility of the underlying grid, they reported that this gives more adequate feedback and helps to comprehend the functionality easily. Furthermore, one VR expert suggested visualizing this zone with a radius indicator.

While the translation of objects via the manipulation widget was liked by the participants, six of them reported that grabbing and moving the manipulation ball instead of dragging the axis arrows was expected to interact with the widget.

Besides the comments given for the existing features, participants also suggested additional features concerning the overall system. One control engineer requested a feature to enter digits for the curvature of the selected road since it would be important to be precise on the curvature values for testing vehicles. Another control engineer stated that it would be nice to see the current velocity of the vehicle and the coordinate information of the objects in the scene. One participant asked for a feature to create tunnels and sidewalks. Furthermore, another participant stated that it would be nice to have an interactive walkthrough to guide novice users in the application.

### 2.4.4 Discussion

As stated in our first hypothesis **H1**, we expected that the participants would complete the tasks faster by employing the indirect interaction techniques, compared to the direct techniques. Based on the results, we found a significant difference in the task completion time that shows **Indirect** outperformed **Direct** in terms of speed, thus we can accept **H1**. To complete the tasks, participants had to utilize selection as well as travel techniques. Therefore, we recorded the time that they spent on selection and navigation individually. The results show no significant difference between the selection techniques, while the travel technique in the **Indirect** condition significantly outperforms the technique in the **Direct** condition. This result indicates that the overall task completion time is affected by the navigation rather than the selection technique.

One possible reason why the navigation was significantly less efficient in the **Direct** condition is the larger travel distance of the two hands compared to the single-handed **Indirect** navigation. Another reason could be the varying degree of experience with 3D spatial transformations of our participants. During the study, we observed that some of the participants had difficulties to perform the tasks in the **Direct** condition while others grasped the technique quickly.

To investigate this further, we analyzed the relation between the expertise (usage of free-hand interaction techniques in VR/AR) and the task completion time in the **Direct** condition with a Pearson test. The result shows a strong correlation between the

expertise and task completion time ($r(16) = -.606, p = .013$), and the participants who employ free-hand interaction techniques frequently completed the tasks faster than the others. As stated earlier, this could be due to a better spatial awareness inside the virtual environment, which enables these participants to grasp a newly introduced interaction and employ it with ease. Regarding the **Indirect** condition, we did not find a significant correlation ($r(16) = -.395, p = .130$). An explanation might be the widespread experience with 2D interfaces that are commonly used in everyday life. This indicates that our technique can be employed comfortably by users with a varied range of expertise on free-hand interaction techniques in VR/AR.

Orthographic projection is commonly used by engineers to create accurate drawings of models. Thus, as stated by our hypothesis **H2**, we expected that the participants would be more precise on the selection tasks using our indirect interaction on the 2D panel, compared to the direct selection. The result of our study supports this hypothesis by showing that **Indirect** significantly outperformed **Direct** in the precision tasks. One possible reason for this could be the posture of the arm. For the direct selection, the arm has to be extended further away from the body, while the indirect selection allows to rest the elbow comfortably close to the body. Another reason could be related to the DOFs of the interaction techniques. The indirect selection uses fewer degrees of freedom compared to the direct technique, for which previous work has shown higher performance in a Fitts' law task [Arsenault and Ware, 2004; Teather and Stuerzlinger, 2013; Luo et al., 2020].

Since road network design is a 2D interaction task by nature, our hypothesis **H3** stated that the participants would prefer to employ the 2D panel over direct interactions. The results show a significant difference between the techniques, and **Indirect** is preferred for road creation which confirms our hypothesis. Regarding **H3.1**, we were able to support our hypothesis based on the score of the subjective question (Q7), which is ranked quite high by the participants (see Section 2.4.3.2). During the guided task phase, one participant stated that he had a better mental model of the scene while employing the 2D panel view alongside the 3D rendering of the scene.

Regarding **H4**, we expected that the participants would prefer to employ direct interactions to add objects into the virtual environment. However, our results show that the participants preferred the **Indirect** technique rather than **Direct** (see Section 2.4.3.2). Thus, we can not confirm **H4**. The reason behind this might be the navigation aspect of the direct interaction. Some users found traveling the scene via zooming in and out with both hands to be less ergonomic than traveling via the minimap. One participant stated that applying the technique "feels like doing a workout". In the next iteration, we plan to do further evaluation by omitting the travel techniques and evaluate this hypothesis based on the selection techniques only.

Based on the comparative and qualitative results, our indirect free-hand selection and manipulation technique has proven to be well-suited for precise and fast placement of

objects onto a terrain surface. We believe that our interaction technique performed particularly well due to its design that addresses the inherently 2D nature of the road network problem. Thus, our system provides an adequate tool for the free-hand authoring of road networks and traffic scenarios that was preferred by the participants over existing scene authoring techniques.

### 2.4.5 Limitations and Future Work

Our system is currently limited to create roads with two lanes. In future work, this can be extended to enable users to create more complex road structures, such as roads with varying number of lanes, widths, and the possibility to merge roads with a different number of lanes.

Regarding our study, one limitation could be using different headsets. However, we tried to minimize the potential confound by using the same tracking system (see Section 2.4.2.2). Another possible limitation is that the results might be not generalizable to a larger population since most participants were male. These limitations should be considered in future iterations.

For further improvements on the system, the numerous valuable feedback we gathered from our participants can be addressed in future work. Furthermore, regarding our selection technique, future studies can explore the performance of our ray-based selection technique when combined with two-handed interactions and other input modalities in a Fitts' law task.

### 2.4.6 Conclusion

In this section, we presented a system for VR-based authoring of virtual environments for the testing and validation of automated vehicles. Our system enables users to create road networks and traffic scenarios, and modify them at run-time based on free-hand gestures. It provides an efficient way to perform design, testing, and validation cycles since all steps are executed inside the application without a break of immersion.

Further, we devised a novel interaction technique that enables the user to add and select objects on the terrain surface by using a 2D control panel that can be placed freely in the 3D environment. 3D objects in the scene get projected onto the 2D panel, and with our interaction technique, the user can employ multiple actions, e.g. selection and addition of objects, based on the distance of the index finger to the control panel.

To evaluate the performance of our interaction technique, we conducted a user study

in which we compared against an existing free-hand interaction method. Furthermore, we gathered feedback for our system by means of a qualitative user study with domain experts. The results show that our indirect technique outperforms direct interaction methods in terms of speed and precision, as well as usability and intuitiveness.

CHAPTER 3

# AUTHORING DYNAMIC AND INTERACTIVE CONTENT BEHAVIOR

In the previous chapter, we explored content generation and manipulation techniques that enable users to construct virtual environments while being immersed. However, building interactive virtual environments requires more than just creating static objects; scene elements must respond to user actions and other in-world events to provide engagement and interactivity [Steuer et al., 1995; Norman, 2002; Zhang et al., 2019]. Defining content behavior is, therefore, a crucial aspect of virtual environment authoring, as it enables users to actively participate in the scenario rather than remain passive observers.

As mentioned in Chapter 1, the traditional way to create object behavior is achieved through programming, in which developers write scripts using programming languages such as C# or C++ to define interactions. However, this approach breaks immersion as it requires users to leave the VR environment to modify the code using external development tools, and relaunch the application to test changes. To enable immersive authoring, research has explored coding directly within VR [Elliott et al., 2015; Fischer, 2016; Castillo et al., 2021]. They provided VR text editors to enable users to write code using either a physical keyboard or an in-VR keyboard. However, despite these efforts, text-based programming in VR remains impractical due to ergonomic challenges, inefficient text entry, and hardware limitations [Dudley et al., 2019; Knierim et al., 2018]. In addition to these constraints, programming is still inaccessible to non-technical users, which limits the ability of designers and artists to author interactive behaviors.

More recently, AI-driven approaches that use Large Language Models (LLMs) have been explored to generate behavior logic. Approaches such as DreamCodeVR [Giunchi et al.,

2024] and LLMR [De La Torre et al., 2024] have demonstrated how AI can assist users in defining interactive object behaviors through natural language prompts. While these approaches offer an exciting alternative to traditional scripting, they also introduce several challenges that limit their effectiveness in content behavior authoring. LLM-generated code is not always reliable [Giunchi et al., 2024]. This leads to users often need to manually debug or refine the output for correctness, which introduce an additional layer of complexity. Another challenge is that LLMs may not fully grasp the intended context or structure of a scenario [De La Torre et al., 2024]. This can lead to unpredictable behavior generation that is inconsistent with user expectations. Additionally, these methods provide limited control over fine-grained behavior logic, which makes them less approachable for projects that require precise, structured interactions [De La Torre et al., 2024]. While AI-driven solutions show potential, they are not yet feasible for end-to-end behavior authoring and still require manual intervention.

Another way to create content behavior that offers users more control over the creation process, compared to AI-driven methods, while remaining accessible to non-programmers, is Visual Programming Languages (VPLs). In the following sections, we provide detailed insights into VPLs and investigate how visual programming paradigms can be adapted within virtual environments to create content behavior while immersed.

## 3.1 Visual Programming Languages

Visual programming languages are an alternative to traditional text-based programming. They provide graphical representation of programming elements and functions and enable users to construct a program using these visual representations. They are approachable by novices because they are found to be easier to read, compose, and recognize due to their respective shape and color-coded functionalities, which reduce the need to memorize complex syntax and thus can support various cognitive aspects of the programming activities [Weintrop and Wilensky, 2015].

However, VPLs can still be challenging to use for those who are not familiar with programming concepts. Furthermore, they have been primarily designed for desktop-based interfaces, where users interact with visual elements using a mouse and keyboard. Integrating VPLs into VR presents unique challenges. We have to rethink how to design programming interfaces, their positioning, and how users will interact with them while immersed.

To tackle these challenges, we explore two primary visual programming paradigms—block-based and dataflow-based—for object behavior authoring. Our selection of these paradigms is based on the distinct advantages each offers for particular tasks. In the following sections, we present their strengths and challenges, review existing adaptations for VR,

examine prior studies on spatial interface positioning, and outline our contributions to this field.

### 3.1.1 Block-Based

Block-based VPLs represent the code structure using graphical blocks that are stacked horizontally and vertically [Resnick et al., 2009]. Each block refers to programming elements such as variables, loops, conditions, or function calls. These blocks are pre-defined and usually available in a palette, enabling users to select and assemble them logically to create a program. The discrete, sequential arrangement of programming blocks in this paradigm is particularly well-suited for linear scenario creation, which enables users to construct step-by-step logic in an organized manner. Instead of memorizing syntax or commands, users can recognize their function based on visual attributes.

This approach has been shown to reduce cognitive load, minimize errors, accelerate the learning process, and make programming more accessible to novices [Weintrop and Wilensky, 2015; Armoni et al., 2015]. Hence, it has been widely used in various domains, with well known desktop-based tools like Alice [Cooper et al., 2000], Scratch [Resnick et al., 2009], Blockly [Google, 2012], MIT App Inventor [Wolber et al., 2011], and Snap! [Harvey and Mönig, 2010].

The adoption of block-based VPLs into VR has been also explored in contexts such as teaching programming and creating interactive experiences. Several approaches, including *Cubely* [Vincur et al., 2017], *VWorld* [Jin et al., 2020] and *VR-ocks* [Segura et al., 2020] are primarily designed for educational purposes to introduce novice programmers to basic programming concepts. A common feature among these approaches is the use of gamification and the representation of programming elements as 3D cubes. However, they differ in interaction techniques and spatial design.

In *Cubely* [Vincur et al., 2017], users engage in a Minecraft-themed virtual environment where a game character is guided through various challenges by executing a correctly structured programming elements. Users view the environment from a third-person perspective and interact with programming elements through direct manipulation. These elements can be placed anywhere in the scene. Similarly, Segura et al. [2020] propose a VR game in which students are tasked with guiding a character through a maze with obstacles by constructing programming blocks. In contrast to *Cubely*, users interact with the environment at a real-life scale. They use raycasting to select floating programming elements and position them in a designated execution area. On the other hand, Jin et al. [2020] with *VWorld*, enables children to interact with blocks using direct hand interactions. It further provides WIM for object selection and navigation. In this approach, the blocks, including motion and audio actions, are located in a programming panel and assigned to the control panel to construct a code sequence. While *VWorld*

offers predefined action blocks (e.g. move forward, turn left, and turn right) similar to *Cubely* and *VR-ocks*, it lacks functionalities for creating logical structures like if-else statements or while conditions.

Some approaches incorporate a 2D editor into VR for block-based programming. For instance, Sun et al. [2021] introduce a Scratch-like 2D code editor within a VR environment, where users add code blocks from a hand-oriented pie menu and numeric input are provided using a virtual keyboard below the editor, and interactions are performed using a ray. Similarly, Horizon Worlds [Meta, 2021] by Meta features a block-based scripting within a 2D editor in VR. Users interact with blocks using a ray, but compared to Sun et al. approach the system offers a more fundamental set of programming elements for constructing complex behavior.

Other approaches build on Blockly's interface, supporting horizontal and vertical block nesting while representing blocks as 3D objects. For example, *LearnIoTVR*, proposed by Zhu et al. [2023], teaches Internet-of-Things (IoT) concepts through block-based visual programming in VR. It categorizes blocks into two types: universal blocks (e.g., loops, conditionals), organized in a floating toolbox, and IoT-specific blocks (e.g., SetPin()), placed above IoT components. Users interact with these blocks using a ray. Similarly, Hedlund et al. [2023] propose *BlocklyVR*, which investigates the constraints and potential of using block-based programming within a large interactive space compared to the desktop-based 2D block editor. The authors provide bigger blocks and enable users to use controllers with short ray for selection, which requires them to move to the respective blocks physically. They found that the performance was not affected compared to the 2D Blockly. However, some participants reported challenges with spatial distribution, block size, and navigation in *BlocklyVR*. They mentioned losing objects often behind them, finding blocks too large for clear code interpretation, and experiencing difficulties in fetching new blocks.

Building on these insights, we investigated the use of the block-based VPL interface for creating dynamic driving scenarios to test automated vehicles. We chose this type of interface since the sequential arrangement of the blocks aligns well with the linear nature of driving scenarios. To this end, we introduce *VRScenarioBuilder*, which enables users to create dynamic driving scenarios using free-hand interactions in VR. Similar to previous works, we represent blocks as 3D objects. However, we simplify the programming process by offering predefined condition and action blocks instead of basic building blocks. Our approach features a natural user interface that enables users to create scenarios quickly and intuitively using a drag-and-drop mechanism. We further provide a WIM interface for scene navigation, entity selection (e.g., vehicles, pedestrians), and observer positioning during scenario playback. Interfaces are positioned in front of the user and within arm's reach to enable comfortable interaction with the objects while seated. While the blocks are stacked vertically, additional parameter adjustments are made using intuitive inputs like button presses, numeric sliders, and demonstration via direct interaction (e.g., defining an entity's trajectory).

Further details on our approach, study findings, and future research directions, including spatial arrangement and managing lengthy blocks, can be found in Section 3.2.

### 3.1.2 Dataflow-Based

Dataflow-based VPL represents the code structure as a directed graph, containing nodes connected by directed edges [Johnston et al., 2004]. Each node, along with its input and/or output ports, may represent a fundamental programming element or a function. The visual representation of a directed graph aligns closely with how individuals naturally conceptualize processes [Tversky, 2013]. Thereby, this can enhance both understanding and manipulation of programs in certain scenarios. For example, Altendeitering and Schimmler [2022] compared the effectiveness of block-based and dataflow VPLs for creating smart spaces scenarios. Their study revealed that users perceived dataflow programming as more visually appealing, easier to understand, and felt more confident when completing tasks.

However, it is important to note that dataflow VPLs can become structurally intricate, particularly in complex scenarios where numerous tangled connections between nodes may arise. Such complexity can pose challenges to program interpretation [Sousa, 2012]. Nevertheless, several well-known 2D desktop-based tools, such as Unreal Engine's Blueprints [Epic Games, 2014], Unity Visual Scripting [Unity Technologies, 2021], LabVIEW [National Instruments, 1986], and TouchDesigner [Derivative, 2008], demonstrate the effectiveness of using a dataflow-based interface in handling complex tasks across diverse domains.

Expanding on desktop-based tools, several immersive VPL approaches [Steed and Slater, 1996; Lee et al., 2004; Ens et al., 2017; Zhang and Oney, 2019; Kao et al., 2020; Zhang and Oney, 2020] have adopted the dataflow paradigm. A pioneering example by Steed and Slater [1996] introduced a system with a dataflow-based interface consisting of three node types: source, filter, and receptor. The system enables users to define object behavior by connecting these nodes within a virtual environment, where source nodes generate input streams, filter nodes process them, and receptor nodes translate the output into visible actions, such as turning on a light.

Another notable approach is *Ivy* by [Ens et al., 2017], an immersive authoring environment based on the dataflow programming model. *Ivy* enables users to design IoT programs by creating logical connections between virtual representations of real-world sensors and actuators within a spatially situated interface. Their study revealed that the process of connecting spatially situated programming elements was found to be intuitive and useful for debugging. However, users also reported challenges with raycasting when selecting distant objects and issues with visual clutter. NeosVR [Solirax, 2018b], a social VR platform, also employs a dataflow-based VPL approach through its *LogiX*

VPL [Solirax, 2018a]. This system allows nodes to be freely positioned within the virtual environment and attached directly to programming objects. While *LogiX* offers comprehensive range of content creation options in a collaborative setting, it faces challenges related to visual clutter and usability [Genz et al., 2021]. *FlowMatic* [Zhang and Oney, 2020] aims to mitigate these challenges by simplifying complexity through the use of abstractions. This is achieved by introducing a set of domain-specific operators and classifying input data into two types: signals, which represent continuous, time-varying values, and event streams, which represent discrete events. The operators process these inputs to either transform the dataflow or trigger actions, such as the creation or destruction of objects. In their system, instead of reaching out for objects in the scene, users can bring them into the programming environment, where programming elements can be arranged on a 2D panel. While this approach may help reduce clutter compared to the nodes positioned freely in the scene, their resulting graph on the panel might still become cluttered, even in relatively simple examples, as shown in Figure 5 of their paper. Another approach proposed by Murray [2022], *RealityFlow* presents a multi-user dataflow VPL that is compatible with desktop, mobile, and VR platforms. Unlike previous approaches, programming nodes are not 3D objects but rather 2D widgets positioned on a 2D panel. All creation activities occur solely within this panel. However, as the project is still a work in progress, an evaluation of the prototype has not yet been published.

Building on these approaches, we present *modulation mapping* which is a simplified visual programming technique that is inspired by dataflow VPL approach. In contrast to previous work, our goal is to further simplify the authoring of reactive behavior for a broader range of users. To achieve this, we propose preparing complex behavior offline by programmers and exposing it to the authoring environment as high-level object properties, following a similar approach proposed by Artizzu et al. [2022]. However, instead of modeling behavior as discrete events and actions defined by logical conditions, we model the immediate effects of continuously varying inputs on the target properties. By introducing a special type of mapping, we enable the modeling of hybrid continuous-discrete signal flows within a unified interface design concept (see Section 3.3.1.2)

As discussed in this and the previous section, previous work has explored different ways to position VPL interface components within the scene; however, none have examined the importance of positioning these elements when authoring in VR. Since our method enables users to define reactive behavior for scene objects while immersed, there is an inherent spatial relation between the programming elements and the objects. Furthermore, the findings of our previous study (see Section 3.2.4) emphasizes the importance of how these elements are spatially arranged. This highlights that the spatial reference frame in which the programming elements are positioned is a crucial factor for the applicability of our approach. To investigate this, we developed two interface layouts: *surround-referenced*, where the programming elements are positioned relative to the physical tracking space, and *object-referenced*, where they are positioned relative to virtual scene objects. An empirical user study was conducted to compare these lay-

outs.

In the following Section 3.1.3, we provide a detailed explanation of the terminology associated with different reference frames for user interface layouts, along with a review of previous studies that explored their impact in immersive environments. Further details regarding our proposed technique, its evaluation with these interface layouts, and the results can be found in Section 3.3.

### 3.1.3 Reference Frames for 3D UI

The placement of 3D user interface components within the virtual environments is crucial because it affects how accessible they are to users and the levels of occlusion present [Bowman et al., 2004]. These spatial arrangements in the VE can be classified according to reference frames to which they are positioned [Ha et al., 2006; Ens et al., 2014b]. In the following, an overview of how reference frames have been classified and evaluated in previous research are provided.

Feiner et al. [1993] pioneered the exploration of 2D window placement in augmented reality (AR), introducing the concepts of world-fixed, display-fixed, and *surround-fixed* windows. In this context, world-fixed refers to windows anchored to specific locations or objects in the 3D scene, while display-fixed windows are displayed at a fixed position relative to the head-worn display, regardless of the user's head orientation. Surround-fixed windows, on the other hand, are located at a fixed position in the surroundings of the user. Later, Bowman et al. [2004] adapted and extended the terminology for graphical menu placement, introducing the categories of world-referenced, object-referenced, head-referenced, body-referenced, and device-referenced. In their adaptation, Bowman et al. [2004] differentiated Feiner et al. [1993]'s world-fixed windows into world-referenced and object-referenced types, and referred to display-fixed windows as head-referenced menus. Additionally, they introduced the term of body-referenced menus, which are anchored to parts of the body (torso, hand, or arm) and device-referenced menus, which refer to a physical display surface like workbenches or handheld AR systems.

Based on these definition, we adopt the *surround-* terminology originally introduced by Feiner et al. [1993] to describe UIs positioned in the user's surroundings, i.e., the physical tracking space. However, in line with Bowman et al. [2004], we replace the term *-fixed* with *-referenced* to indicate that the interface can be repositioned freely within this reference frame. In this regard, a *surround-referenced* interface appears static as the user walks in the tracked area, similar to a world-referenced layout. However, if a travel technique such as steering [Bowman et al., 2004] or teleportation [Bozgeyikli et al., 2016] is employed, the interface moves along with the user. Consistent with Bowman et al. [2004], we use the term *object-referenced* for UIs that are attached to scene objects.

The influence of different reference frames for interface layouts has been previously studied in various contexts. For instance, Billinghurst et al. [1998] conducted an empirical study to compare user task performance using head-fixed and body-fixed information displays on a wearable computer. The study result revealed that body-fixed positioning enabled for faster information retrieval and was perceived as easier to use and more intuitive. Polys et al. [2005] further investigated the role of information layout (object-referenced and head-referenced) with varying screen size, and field-of-view on user performance in VEs. They found that the consistent visibility of the head-referenced layout significantly outperformed object-referenced layout in terms of user performance (accuracy and time) and satisfaction. Similarly, Bernatchez and Robert [2007] found that 3D floating menus are more effective when they are designed to follow the user's position and face towards them, compared to menus that remain static in the scene, follow the user's position only, or attached to the non-dominant hand.

Das and Borst [2010] further evaluated the performance of contextual (object-referenced) versus world-referenced menus for projection-based VR systems. In their study, users were tasked to select a highlighted object first, then select a target item from the menu that appeared. The results showed that contextual menus enabled faster selections. Ens et al. [2014b] compared world-referenced, body-referenced and referenced-referenced virtual displays for selection tasks on head-worn AR displays. Their study revealed that target selection was more accurate with the world-referenced display and the body-referenced display resulted in higher selection errors due to reaching motion. Lediaeva and LaViola [2020] later conducted a comprensive study to evaluate body-referenced (arm, hand, and waist) and world-referenced menus with varying shapes and selection techniques in VR. They found that participants preferred the world-referenced menus the most, and found them to be less mentally and physically demanding. In contrast, the arm-referenced menus were the least favored, being perceived as the most challenging and physically demanding to use.

Compared to spatial virtual displays or menus, immersive authoring of reactive behavior using dataflow VPL involves a graphically represented connecting lines between nodes. This introduces additional information and interaction in 3D space that the user must process and manage when authoring in VR. However, the literature has not yet explored a comparison of reference frames for immersive dataflow VPL interfaces. This comparative analysis is essential to understand the role of using different reference frames within the dataflow VPL paradigm, and for designing effective authoring approaches based on these insights. Building on previous work, we choose a *surround-referenced* layout that travels with the users to ensure consistent visibility, which may enhance accessibility and performance [Polys et al., 2005; Bernatchez and Robert, 2007]. We compare it to an *object-referenced* layout that anchors interface elements to virtual objects in the scene, which may support contextual and spatial awareness [Das and Borst, 2010] and improve user experience [Lediaeva and LaViola, 2020; Ens et al., 2017].

In the context of immersive reactive behavior authoring, our research offers valuable

insights into the design and optimization of immersive dataflow VPL interfaces by systematically evaluating the efficiency and overall user experience of these layouts. Further details regarding the design of the interfaces and the evaluation can be found in Section 3.3.

## 3.2 A Block-Based Approach for Linear Scenario Authoring

*The contents of this section are based on – and taken in part from – work previously published in [Eroglu et al., 2024b].*

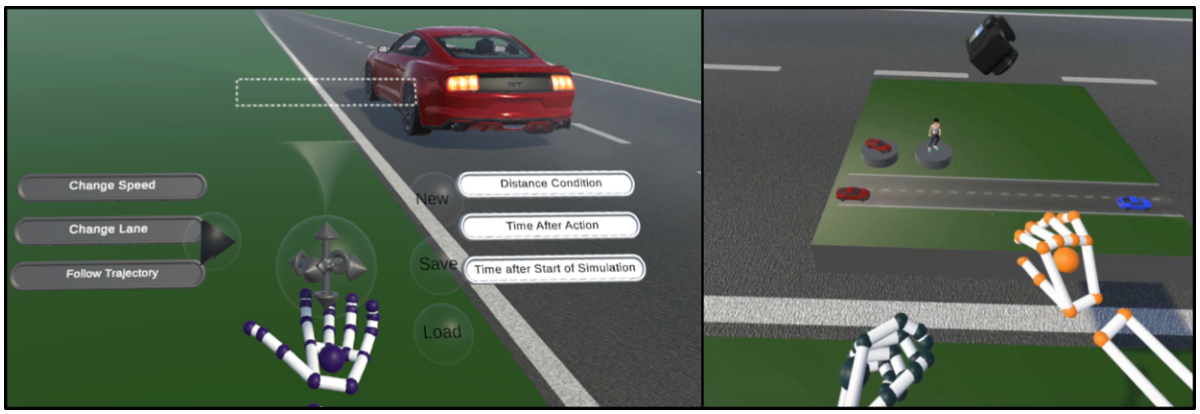



**Figure 3.1:** VRScenarioBuilder to author scenarios for testing automated driving features. Left: Block Editor for block-based assembly of dynamic traffic scenarios. Right: World-in-Miniature interface to navigate the scene, add and select entities (e.g. vehicle or pedestrian), and define the observer position during scenario playback.

Virtual Reality (VR) has gained increasing interest from the automotive industry, primarily due to its ability to provide simulated environments where engineers can actively participate and observe realistic driving scenarios. This enables engineers to test and validate automated vehicles in varying traffic scenarios without endangering themselves, vehicles, or the surrounding environment. By simulating environments and replicating real-world driving conditions, provides a controlled space for quick adjustments and improvements to vehicle systems. This accelerates development and testing cycles, leading to more efficient processes and reduced costs in the development of automated vehicle technologies.

To create testing scenarios for automated driving systems, road networks and traffic scenarios need to be defined. Real public data sources [Cai et al., 2020; Lopez et al., 2018; Azevedo et al., 2017], procedural generation [Paranjape et al., 2020; Tan et al., 2021], learning-based [Bergamini et al., 2021; Sun et al., 2022; Feng et al., 2023] or

user-driven [OpenMSL, 2023; OpenDILab, 2021; Zhou et al., 2023] methods are used to address these needs. User-driven methods offer a customizable approach that can be adjusted to meet specific needs, allowing for flexibility in the design process. Once a desired driving scenario has been identified, the user manually specifies parameters, including coordinates, trajectories, velocities, and orientations using a domain-specific language such as OpenSCENARIO [ASAM, 2024]. This is an open XML-based standard describing dynamic content for virtual test drive simulations, containing the maneuvers of traffic participants. These scenarios, utilized as input for simulators in testing automated driving functions, support reusability across tools and development stages. Thus, engineers commonly use simulators like CARLA [Dosovitskiy et al., 2017], SVL [Rong et al., 2020], CarMaker [IPG-Automotive, 1999], and DYNA4 [Vector, 2023].

However, these simulators do not enable users to modify the traffic scenarios interactively while being immersed. Although previous works [Eroglu et al., 2021; Côté and Beaulieu, 2019] enable users to create and modify roads in an immersive environment, they do not provide the ability to author dynamic traffic scenarios.

To overcome these shortcomings, this work investigates *how an immersive scenario authoring environment be designed to enable engineers to author dynamic traffic scenarios?*

To address these shortcomings, we present VRScenarioBuilder, an immersive test environment that enables engineers to efficiently test and validate automated vehicles in VR. To make interactions with the system as intuitive and approachable as possible, we have designed a natural user interface. Our approach enables users to create and modify dynamic traffic scenarios using free-hand gestures. To this end, we designed an easy-to-use drag-and-drop interface that is inspired by block-based visual programming languages. To evaluate our interface design decisions, we performed a user study with $n = 9$ VR experts. We gathered qualitative feedback to investigate potential difficulties and improve the proposed approach in future iterations. Furthermore, we identify future research directions based on the feedback and our observations.

### 3.2.1 VRScenarioBuilder

We developed our system based on the following requirements that are defined by control engineering experts of an automotive company.

**R1** The system should enable users to create, modify, and fine-tune dynamic traffic scenarios that include driver actions such as changing a lane, overtaking or following a defined trajectory.

**R2** The system should enable users to observe and analyze scenarios from different

perspectives for comprehensive testing.

**R3** The system should facilitate offline editing and compatibility with widely used standards like OpenSCENARIO.

To address each requirement, we designed an immersive authoring system with two main user interface components: the *World-in-Miniature* (Section 3.2.1.1) and the *Block Editor* (Section 3.2.1.2). In the following, we present our system in more detail.

#### 3.2.1.1 World-in-Miniature

To enable users to efficiently interact with virtual objects within arm's reach, we utilize a World-in-Miniature (WIM) interface [Stoakley et al., 1995], a miniaturized replica of the virtual environment. Upon opening the right hand with the palm facing upward, the WIM interface becomes visible in front of the user. To enable users to interact effectively with the virtual environment from various perspectives, the WIM can be freely repositioned with a single hand and rotated around the vertical axis with both hands. To change the scene inside of the WIM, we enable users to employ a swiping gesture on the floor, similar to scrolling on a smartphone.

The first step in designing a traffic scenario (**R1**) is to add entities to the virtual environment. Currently, our system provides two different entities: *Vehicle* and *Pedestrian*. The miniature copies of these entities are located in their respective container on the WIM interface (See Figure 3.1 - Right). These objects can be added to the scene by grabbing and placing them in the WIM. After removing a copy from its container, a new copy of that entity appears in its place.

The position and orientation of scene objects can be manipulated by moving them inside the WIM. To reduce the authoring time, the position and the orientation of vehicles are automatically adjusted to fit the nearest lane on release. Pedestrians, on the other hand, can be positioned and oriented arbitrarily.

To address **R2**, the WIM interface incorporates a camera that enables users to define their initial position and viewing direction for testing the scenario. During authoring phase, the camera can be moved by grabbing it and rotated by manipulating the lens. Via the preview screen of the camera, users can determine their perspective within the scenario. The camera can also be used to follow a specific entity within the simulation. This can be accomplished by dragging the camera onto the entity of interest and releasing it. The camera will then "snap" to the entity and become smaller, indicating to the user that it is now following that entity during the simulation.

#### 3.2.1.2 Block Editor

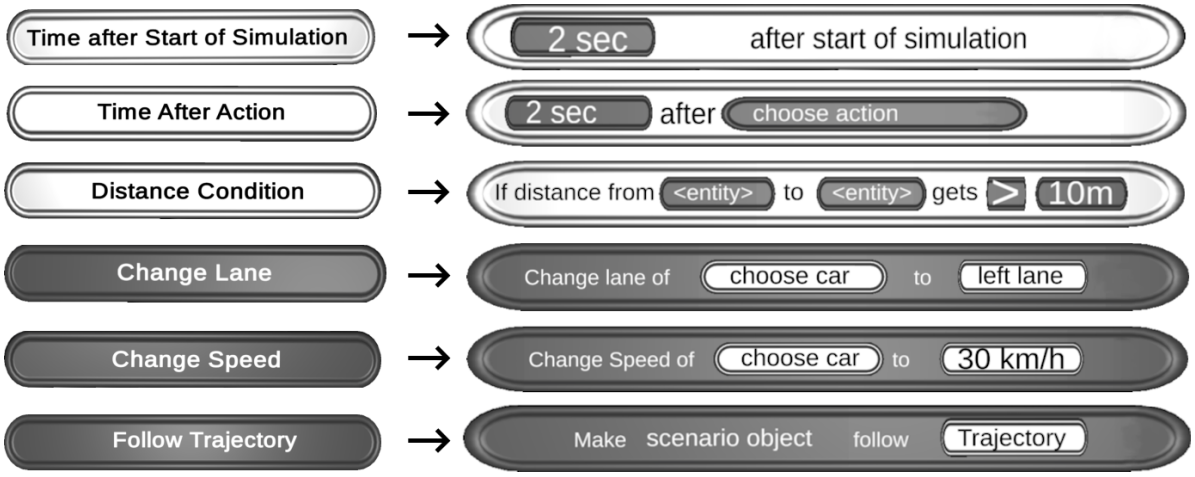



**Figure 3.2:** This figure shows how each block changes in appearance once it has been added to the programming area. This change in appearance causes the blocks to have different parameters to specify the condition/action. These parameters are highlighted black for the white *Condition Blocks* and white for the black *Action Blocks*.

In order to design an easy-to-use interface for authoring dynamic traffic scenarios, we needed to simplify the complex hierarchy of elements in OpenSCENARIO, which can require 258 lines of code to define a simple lane-cutting scenario [Lou et al., 2022]. To achieve this simplicity, we drew inspiration from the block-based visual programming paradigm. In this approach, code snippets are represented by blocks that can be dragged together to create programs. This provides a simple way for program creation that doesn't demand extensive programming skills or knowledge. To this end, our *Block Editor* user interface component enables users to author dynamic traffic scenarios by dragging and dropping building blocks, namely condition and action blocks (**R1**).

The Block Editor can be brought into the virtual environment by turning the left hand with the palm facing upwards. To enable users to comfortably interact with the interface, it can further be repositioned freely by grabbing its handle, drawn as a sphere containing 3D arrows. Additionally, while positioning, the interface rotates in a way that directly faces the user, to ensure optimal visibility and avoid additional interactions that could potentially cause arm fatigue [Hansberger et al., 2017].

#### Assembling Blocks

The main UI elements of the Block Editor component are *Condition Blocks* and *Action Blocks*. Condition Blocks are similar to "if" statements in programming languages, and

Action Blocks serving as function calls within these "if" statements. Once a condition is met, the corresponding actions are executed. Currently, Condition Blocks can only exist on the first level and Action Blocks on the second. This setup allows for multiple dependent action blocks per condition block. However, each condition block is evaluated independently of others on the first level during playback.

The Block Editor currently offers three types of Condition Blocks and three types of Action Blocks. The functionalities of these are designed to create dynamic scenarios — such as "Cut-In", "Double Lane Changer", and "Overtaker" — as described in User Guide [OpenSCENARIO, 2020], meeting the needs of system control engineers (**R1**). To define these scenarios, we have designed the following blocks:

**Condition Blocks:**

- *Distance* is triggered when the distance between two selected entities becomes greater or smaller than a defined threshold in meters.

- *Time After Action* is triggered when the defined time has passed after the execution of a specified action.

- *Time After Start* is triggered when the defined time has passed after the simulation started.

**Action Blocks:**

- *Change Speed* sets the desired speed of a selected vehicle to a specified one.

- *Change Lane* causes a selected vehicle to change its lane to a selected (right or left) lane.

- *Follow Trajectory* make a chosen entity, a vehicle or a pedestrian, to follow a defined trajectory.

To create a scenario, the blocks need to be positioned in input area, denoted by a white dashed rectangle in the upper-middle part of the interface (See Figure 3.1 - Left). Upon adding a block, all exiting blocks move upwards to maintain space in the input area for the next condition or action block. Simultaneously, its appearance expands to display adjustable parameters required for further specification (See Figure 3.2). To remove a block from the scenario, the block can simply be grabbed and thrown away. The block will fall and be deleted.

### Parameter Specification

To enable users to define parameters, our system offers 3D buttons and sliders. When a button on the expanded block is pressed, the system performs a corresponding action based on the parameter type. For numerical parameters, users can define the value by interacting with a slider. For a precise definition of the values, +/- buttons can be employed (See Figure 3.3). Additionally, we provide toggles for binary parameters that are used in the *Distance* condition (“>” or “<”) and the *Change Lane* action blocks (“left lane” or “right lane”).

The *Change Lane* and the *Chance Speed* blocks require the user to specify which vehicle should execute the action. To perform this selection of a vehicle, the WIM interface is employed. After pressing the “choose car” button, the Block Editor disappears and the WIM appears in front of the user, and the vehicle can be selected by touching it directly with the index finger. The *Distance Condition* and *Follow Trajectory* blocks have a similar parameter, “entity”, allowing the selection of a pedestrian or a vehicle. The *Time After Action* block incorporates a parameter defining which action triggers this condition. To set this parameter, the user needs to press the button and then select the desired action.

Lastly, as a parameter of *Follow Trajectory*, a trajectory can be defined. For this reason, we enable users to draw a path by directly grabbing the object (a vehicle or a pedestrian) and moving it in the WIM (See Figure 3.4). The resulting path is projected onto the road and is visualized in white to provide visual feedback.

### Playing Scenarios

After a scenario has been created by assembling condition and action blocks, simulation can be initiated by interacting with the play button on the Block Editor interface. Upon play, the position of the user will be set based on the miniature camera. The position of the user will be updated to follow an entity if the camera is attached to that entity. During the simulation mode, the Scenario Builder is adapted to offer new input capabilities to pause, resume, and stop the scenario. For clear observation of the simulation steps, annotations are provided to indicate ongoing actions (See Figure 3.5). Their transformation dynamically updates based on the user’s position and they disappear after some seconds.


**Figure 3.3:** Depiction of VRScenarioBuilder's *Slider*. The user clicked on the "5 sec" button of the *Time After Start* condition block. The parameter can be set by moving the slider with the index finger. The two buttons with "-" and "+" can be used for fine adjustment.

### Dynamic Editing of Scenarios

Our system further enables users to dynamically modify scenarios while the simulation is running (**R1**). It's a crucial feature for evaluating how individual entity behaviors affect scenario outcomes at specific points in the simulation. To achieve this, the simulation first needs to be paused and then desired action blocks can be added to the input area of the Block Editor. Once the modifications have been made, the user can then resume the simulation and observe the effects of the changes.

### Saving and Loading of Scenarios

Another crucial feature requested by engineers is the ability to modify the scenarios offline and share them for use in other testing environments (**R3**). For this reason, we enable users to save scenarios in the XML format of OpenSCENARIO. It is commonly employed in the automotive industry and compatible with various testing environments such as CarMaker [IPG-Automotive, 1999] or CARLA [Dosovitskiy et al., 2017]. A created scenario can be saved by selecting the save button on the Block Editor (see Figure 3.1 - Left). The scenario file will be stored under the project folder. It can be modified offline and, e.g., loaded into a desktop-based testing environment. The modified file can then be loaded into VRScenarioBuilder to continue the workflow in VR.

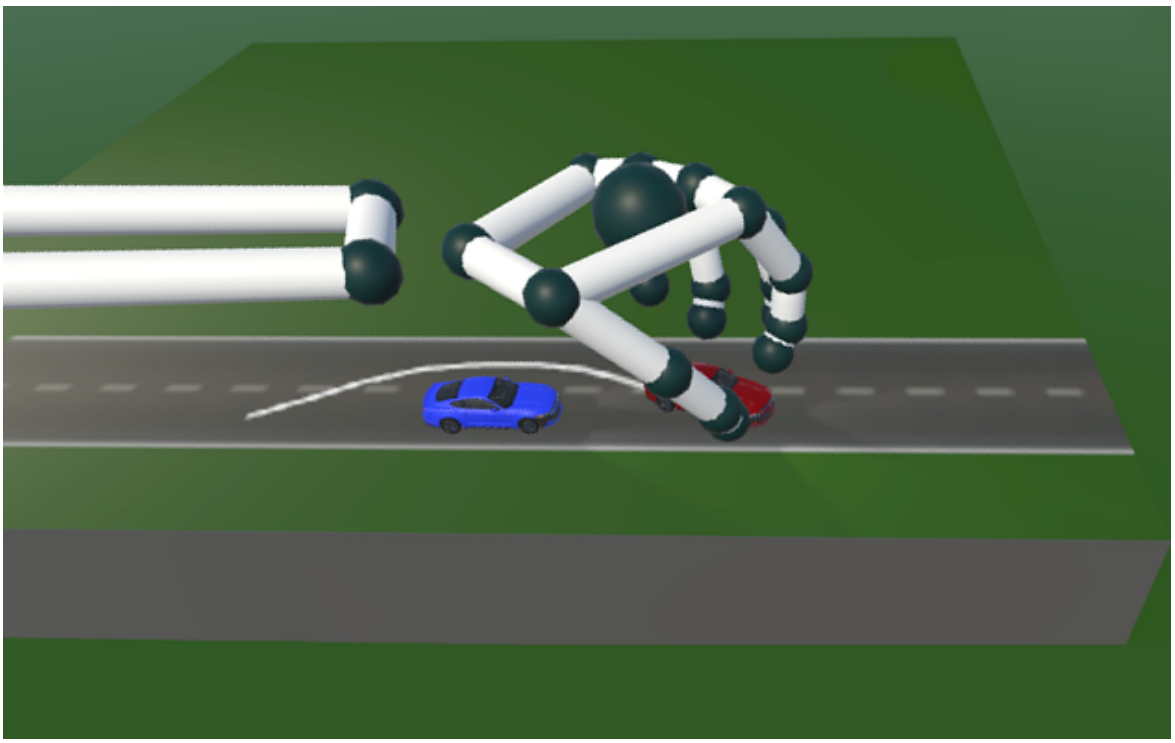


**Figure 3.4:** Creating a trajectory in the WIM: The user is moving the red car, creating a trajectory to overtake the blue car.

### 3.2.2 User Study

To gather initial feedback on the interface components and free-hand interactions of VRScenarioBuilder, we conducted a user study with VR experts. This study serves as an initial guide for refining the interface concept, enhancing usability, and implementing additional features.

#### 3.2.2.1 Apparatus

The study was conducted in a seated position at our lab. We used the HTC Vive HMD with two Lighthouse 1.0 base stations. In order to track the position and orientation of the user‘s fingers and hands, we employed the Leap Motion Controller, mounted onto the VR headset. The experimental platform was developed in Unity 2018.3.14f1 and ran on a 3.50GHz Intel Xeon E5-1650 with NVIDIA GeForce GTX 1080 GPU, operating Windows 10.

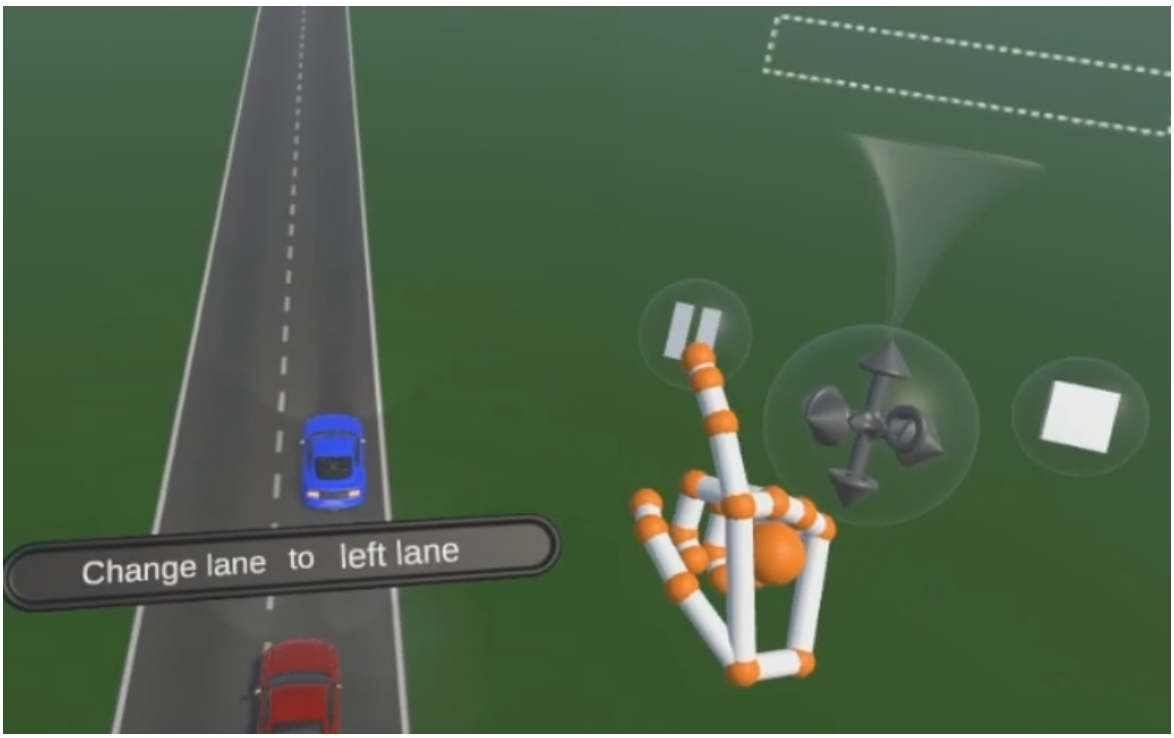



**Figure 3.5:** Playback of a scenario: The Block Editor displays two buttons to pause or stop the simulation. Over the simulated vehicle, an annotation indicates the red car is changing to the left lane.

#### 3.2.2.2 Procedure

The study procedure is composed of several sequential steps. First, participants signed a consent form and filled out a pre-study demographic questionnaire. Then, an example OpenSCENARIO file "Overtaker.xml" and the VRScenarioBuilder workflow were shown and briefly explained.

Once equipped with the HMD, participants were instructed to think aloud and freely comment on the system. After getting familiar with the system, the participants were instructed to load the example file using the Block Editor and play this scenario.

In the next step, the participants had to create a new scenario based on the image that is shown in VR (See Figure 3.6 - Top). Several features were utilized in this task, including block assembly, parameter configuration, as well as creating a trajectory for the pedestrian. Upon successful creation (See Figure 3.6 - Bottom), the participants were instructed to play this scenario.

Once the pedestrian had crossed the street, the participants were instructed to use

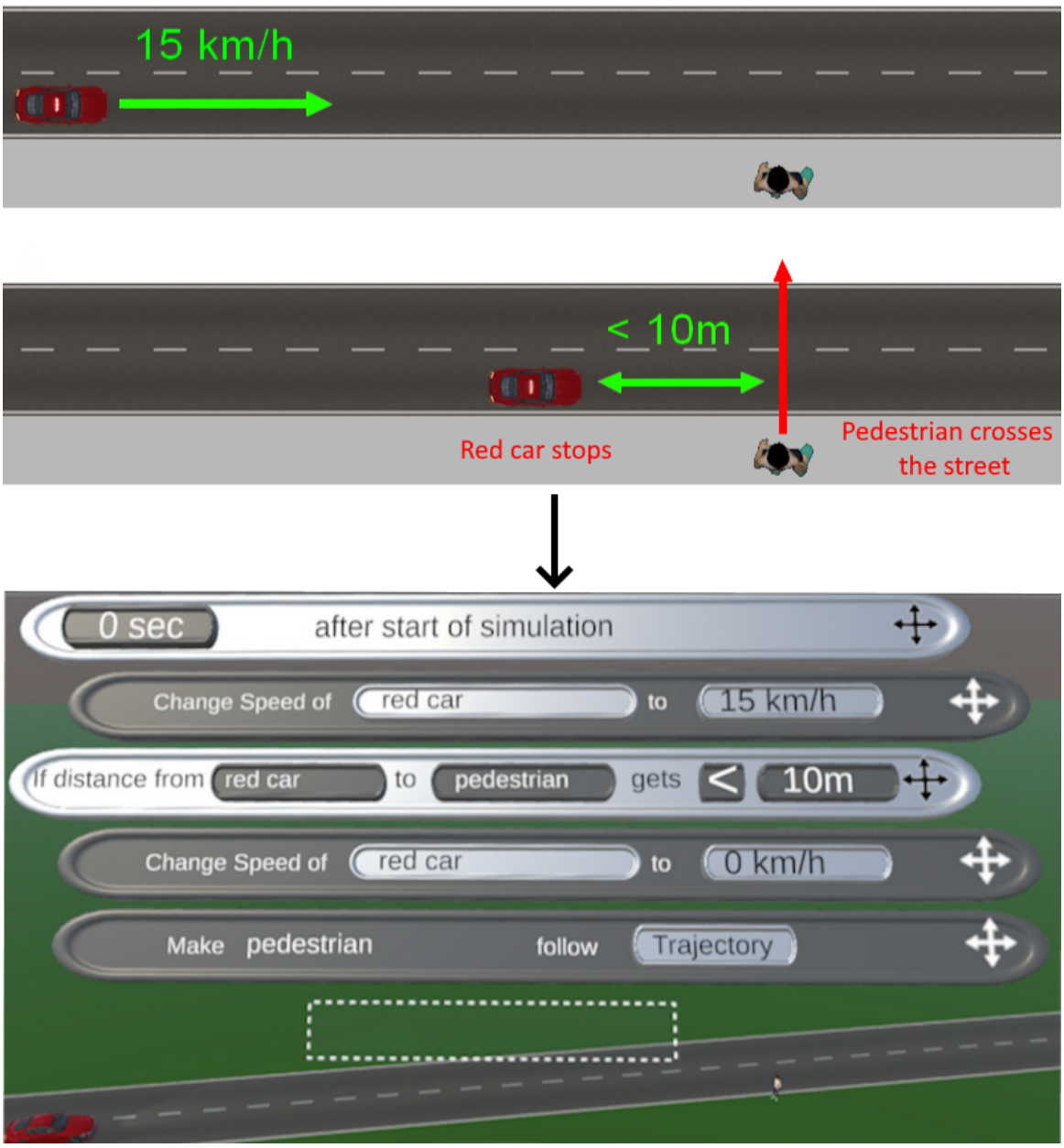



**Figure 3.6:** Top: The scenario depiction shown to users to recreate in the study: The red car begins to drive at 15km/h, if the distance to the pedestrian is less than 10m, the car stops and the pedestrian crosses the street. Bottom: The depiction of correctly assembled blocks and set parameters for the scenario described above.

the dynamic editing feature to make the car continue driving. This involved pausing the scenario, adding a *Change Speed* action, setting the "choose car" parameter, and resuming playback.

After completing the tasks, participants put the HMD off and completed the System

Usability Scale (SUS) [Brooke et al., 1996], a custom 5-point Likert scale questionnaire (See Figure 3.7). Furthermore, semi-structured interviews were conducted with each participant to gather feedback on system difficulties and suggestions for improvement. The entire procedure took approximately 45 min.

#### 3.2.2.3 Participants

9 persons participated in the study. They were between the ages of 22 and 33 (7 male and 2 female). All participants had normal or corrected vision. None of them suffered from color blindness, and all were right-handed. Additionally, all participants reported having experience with VR and 3D user interfaces regularly, with 5 reporting previous experience with free-hand interaction in VR. All participants also reported having regular programming experience.

### 3.2.3 Results and Discussion

Overall, we mainly received positive feedback from VR experts. They were able to solve the tasks without major problems and rated the system's usability as good, giving it an average SUS score of 75. Furthermore, they found the interaction with the system using free-hand gestures was *not physically exhausting* **Q1** and *functioned well* **Q10**. The further results of the custom 5-point Likert scale questionnaire can be seen in Figure 3.7. The experts further provided valuable feedback on existing and suggested additional features, which we present and discuss in the following.

#### 3.2.3.1 Feedback on Existing Features

Concerning loading of scenarios, several participants commented positively on the scenario representation using blocks as being *easy to understand*. Upon playing the loaded scenario, one participant stated that the annotations *could sometimes cover important elements* of the scenario. To address this, we plan to implement an automated annotation positioning approach to prevent interference with important elements of the scenario, similar to previous work [Pick et al., 2010; Tatzgern et al., 2014].

Regarding scenario creation, users expressed confidence in using the system and appreciated the block assembly concept for its *intuitiveness*. These positive responses are further supported by the ratings of **Q2** and **Q7** in the custom questionnaire (See Figure 3.7). Two participants, however, experienced issues when attempting to insert a block into the programming area. The block would sometimes *fall down accidentally*

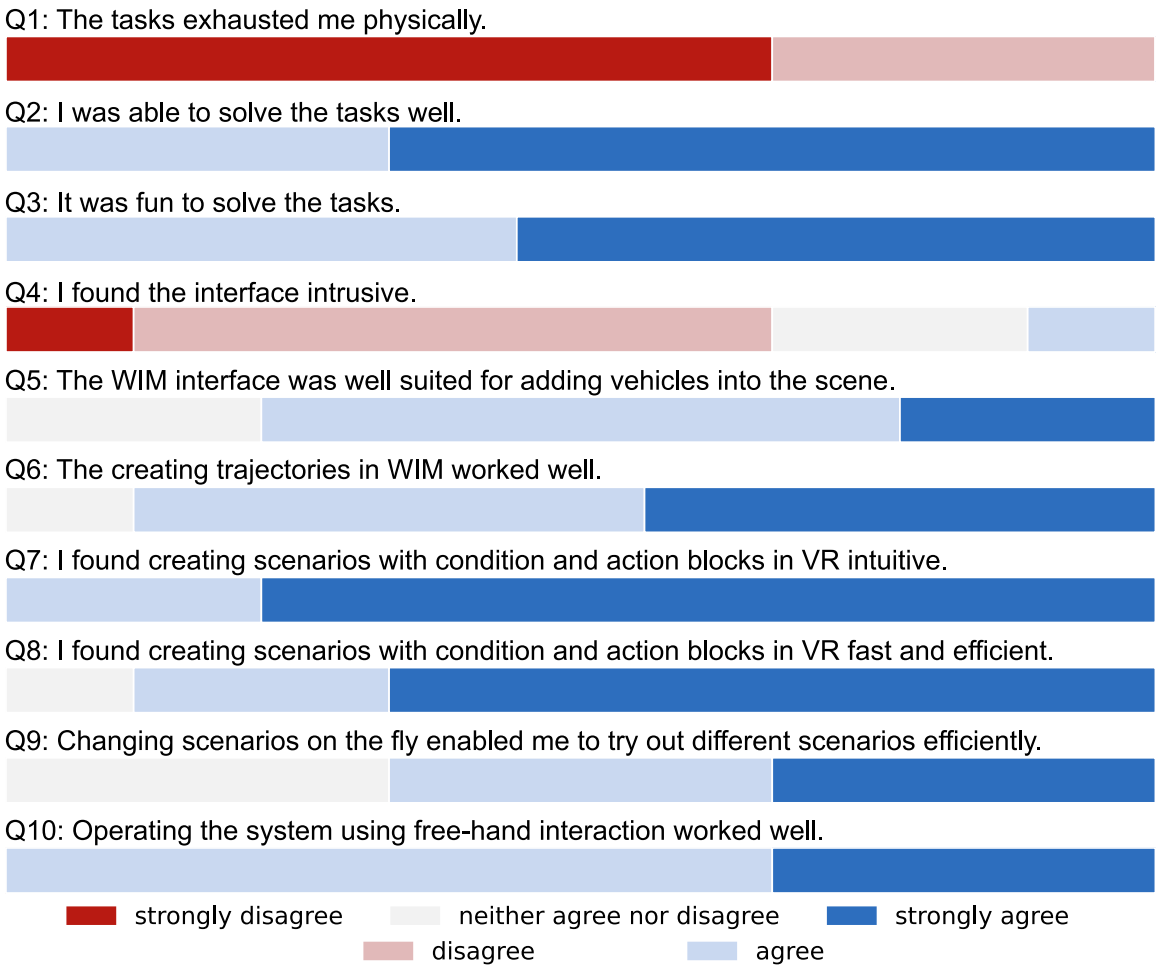



**Figure 3.7:** Histogram of answers to a custom 5-point Likert scale questionnaire.

when released too early. To solve this issue, we plan to provide improved visual feedback to indicate when it is safe to release the block. In addition, we also plan to implement an undo/redo feature to enable users to correct accidental actions.

In terms of parameter specification, several participants gave positive feedback on the *ease of using* the UI elements of the Block Editor to set numerical parameters. However, one participant stated that the transition from the Block Editor and the WIM was sometimes confusing when selecting a "entity" parameter. To improve the user experience, we plan to evaluate different approaches for transitioning between the interfaces.

#### 3.2.3.2 Ideas for Additional Features

We received valuable insights for improving the functionality of VRScenarioBuilder. One notable suggestion is to enable the *combination of multiple conditions* using a logical "and" by nesting consecutive Condition Blocks directly below one another. Another suggestion from two participants was to have a *list of scenario entities* that can be selected if the WIM becomes crowded. Furthermore, regarding the playback of scenarios, one participant suggested a feature to *"scroll in time"* during playback to enable users to revisit specific moments in a scenario. As part of future design iterations, we will carefully consider these suggestions.

### 3.2.4 Lessons Learned and Future Research Directions

Several key lessons have emerged from our observations and the feedback that we gathered.

- **Extension to Multimodal Cues:** We have observed that our system is lacking some important visual cues within the user interface such as indicating where to safely add blocks. In addition to visual cues, carefully selected auditory cues can also be used to further enhance the user experience [Schlünsen et al., 2019]. In order to improve user understanding, reduce confusion, and simplify interaction with the system, the incorporation of auditory and more prominent visual cues should be considered in future iterations.

- **Investigating the Layout of the Interfaces:** When specifying entities as parameters, the WIM interface appears and the Block Editor disappears until an entity has been selected. This design decision was made to avoid potential clutter in the user's view. However, based on feedback we have received, the transition between the Block Editor and the WIM interface can be confusing. This can potentially lead to accidental selection of the wrong entity, especially when the WIM is crowded. Simultaneously displaying the Block Editor and the WIM interfaces may introduce challenges related to visual clutter and optimal positioning within the interaction range of the user. To tackle this issue, we investigate the following research question:

  **RQ1:** How can the spatial positioning and the visibility of user interface elements be designed to minimize visual clutter and improve the user experience during scenario creation in VR?

  To address this question, we designed two interface layouts (see Section 3.3) and evaluated them regarding several aspects, including visual clutter, cognitive load, task completion time, and overall usability. The results are presented in Section 3.3.3.

To further investigate this, future research could incorporate additional quantitative measures besides task completion time and cognitive load. For instance, applying the dual-task paradigm [Wickens, 1981] would enable us to measure cognitive load during the use of the user interface. This will help us gain a deeper understanding of which elements of the user interface affect cognitive load in particular. In this way, it could be possible to examine in more depth how spatial design affects the user experience while creating immersive scenarios.

- **Lengthy Scenario Code Blocks:** We observed that as the number of blocks increases in the Block Editor interface, there could be a possible challenge in terms of navigation, readability, and efficient management. To handle this complexity, we propose the following solutions:

  Implementing a *hierarchical structure that organises blocks into categories* could simplify the navigation process. This way users can collapse or expand sections, allowing for a more focused and efficient exploration of scenario blocks. Additionally, *search* and *filter* functionalities can be added to the interface to enable users to quickly locate specific blocks. This feature may allow users to manage extensive lists more easily by providing targeted access based on naming or functional attributes. Offering users the ability to create *custom groupings* within the interface could be another solution. This would allow users to organise scenario blocks based on their preferences and workflow, which could result in an experience that is intuitive and easy to use. Building upon these proposed solutions, we have developed the following research question:

  **RQ2:** How can an interface be designed to manage and navigate an extensive list of scenario code blocks efficiently in VR?

### 3.2.5 Conclusion

In this section, we presented *VRScenarioBuilder* as a novel solution to address the limitations imposed by 2D desktop-based tools in designing and testing dynamic driving scenarios for automated vehicles in VR. Our approach enables users to create dynamic traffic scenarios and modify them at run-time based on free-hand gestures. By performing design and evaluation steps within the application without a break of immersion, it offers an efficient workflow for testing automated driving features. We designed an intuitive drag-and-drop scenario builder interface, drawing inspiration from block-based visual programming languages for user-friendly interaction.

To evaluate our interface design choices, we conducted a user study with VR experts. Furthermore, we gathered qualitative feedback to find potential challenges and refine our approach in future iterations. The results show the intuitiveness of design decisions for creating dynamic scenarios with free-hand interactions, with the participants rating

the system's usability as good, thereby validating the effectiveness of our approach. In addition, we identified future research directions based on feedback and our observations. These aim to enhance the interface and further contribute to the improvement of immersive scenario authoring for testing automated driving features in VR.

## 3.3 A Dataflow-Based Approach for Behavior Authoring Using Surround and Object-Referenced Spatial Frames

*The contents of this section are based on – and taken in part from – work previously published in [Eroglu et al., 2024a].*

In the previous section, we explored the adaptation of the block-based visual programming paradigm to create dynamic driving scenarios in VR. While the linear and hierarchical arrangement of programming blocks in this paradigm is well fitting for the creation of the driving scenarios, a dataflow-based interface can be more suitable for creating other types of scenarios [Altendeitering and Schimmler, 2022]. The integration of dataflow-based VPLs for immersive authoring have been previously investigated and found to be intuitive [Ens et al., 2017; Zhang and Oney, 2020]. However, as discussed in Section 3.1.1, they may still pose usability challenges due to wide range of functions they offer [Solirax, 2018a] and the potential for visual clutter [Ens et al., 2017; Zhang and Oney, 2020].

To overcome these issues, we further investigate *how to adapt the dataflow-based paradigm into VE to author object behavior*. To this end, we present *modulation mapping*, an approach inspired by dataflow VPLs that simplifies the process of creating reactive behavior while immersed. Our method defines a workflow, in which a developer prepares application-specific degrees of freedom to control object behavior, exposed as object properties. In the VE, the user interactively maps input sources to these predefined properties by drawing connections. Input sources can be user input, the progression of time, properties of other objects, or spatial relations between objects. To make the technique approachable by novices, we only allow direct connections from sources to target properties. To retain expressiveness, behaviors can be interactively customized using visually represented mapping functions chosen from a set of templates.

Possible use cases for *modulation mapping* include teaching scenarios, where instructors may define cause-and-effect relations between objects to illustrate processes in, e.g., mechanics or chemistry. In scientific visualization, domain experts may explore data by mapping user interactions, such as joystick movements to visualization parameters. In game design and virtual storytelling, creators may enrich the scene by adding interactive elements. Specific examples are presented later in this article (see Table 3.1).

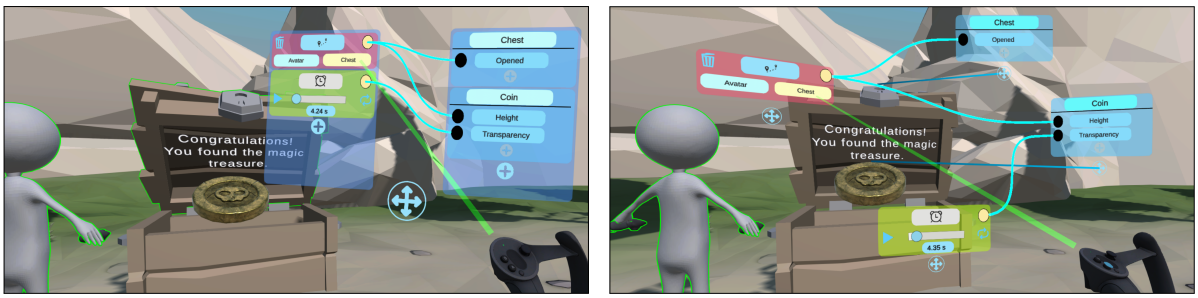



**Figure 3.8:** A user creates reactive behavior by *modulation mapping*. *Surround-referenced* layout (left): menu elements are located on a panel that travels with the user. *Object-referenced* layout (right): menu elements are attached directly to scene objects.

Because our method enables users to define reactive behavior for objects that are located within the virtual scene, there is an inherent spatial relation between the programming elements and the objects. Therefore, an important factor for our approach's applicability is the spatial reference frame in which the programming elements are positioned. While the role of reference frames in information visualization [Feiner et al., 1993; Billinghurst et al., 1998; Ens et al., 2014a] and menu interactions [Lediaeva and LaViola, 2020; Das and Borst, 2010] has been explored, the impact on the immersive authoring of reactive behavior through visual programming remains uninvestigated.

To support the design of immersive authoring techniques, our research further investigates *the role of reference frames on the applicability of modulation mapping for creating reactive behavior*. Therefore, we designed a *surround-referenced* and an *object-referenced* interface layout, previously used by other immersive authoring environments [Ens et al., 2017; Solirax, 2018b; Chauvergne et al., 2023] (see Section 3.1.3 for more detailed terminology). In a study ($n = 34$), we compared these layouts, revealing a significant effect of reference frame choice on the technique's applicability. From our findings, we derive initial interface design recommendations to guide future research on interface layouts for immersive creation of reactive behaviors.

In the following, first, we present the design considerations on which our technique is based. Then, we outline the details of the technique and the user interface layouts (surround-referenced and object-referenced). Afterward, in Section 3.3.2, we describe our empirical study in which we evaluated our technique and the role of reference frames. The results are presented in Section 3.3.3, followed by a discussion and design considerations in Section 3.3.4.

### 3.3.1 Technique Design

Our goal is to design an immersive authoring technique for creating reactive behavior that is easily accessible to a wide range of users. To achieve this, we drew inspiration from dataflow VPL, which represent programs as networks of nodes and connections that naturally depict data and control flow. This visual representation aligns well with how humans conceptualize processes [Tversky, 2013], making it easier to understand the transformation of data, and is thus considered intuitive and end-user friendly [Zhang and Oney, 2019]. To devise our technique, we developed the following design considerations:

**D1 Structural Simplicity**: Dataflow VPL programs can become structurally complex with numerous tangled connections between nodes, making interpretation difficult [Sousa, 2012]. Therefore, our technique should enable users to define reactive behavior without the structural complexity of VPL.

**D2 Effortless Customization**: Users can have difficulty to understand low-level primitives and compose them into high-level behavior [Pane and Myers, 1996]. To make the technique approachable, customizing the influence on a target property should be possible without using explicit mathematical operators.

**D3 Unified Mechanism**: VPL that offer multiple concepts to choose from can negatively impact the ability to solve problems easily [Bell et al., 1991]. Thus, for conceptual simplicity, the same mechanism used to customize mappings should enable modeling of conditional and stateful behaviors.

**D4 Spatial Reference Frame**: As the nodes are closely related to scene objects, it is crucial to investigate their placement relative to the objects or the user's position [Polys et al., 2005; Das and Borst, 2010]. The positioning of the interface elements should provide a good user experience in terms of efficiency, cognitive load, and user-friendliness when creating reactive behavior while being immersed.

Based on these considerations, the following sections explain our conceptual decisions and UI designs.

#### 3.3.1.1 Modulation Mapping

To achieve a low structural complexity that is more approachable to novice users (**D1**), we propose to restrict the mapping between *source* and *target* properties to direct point-to-point connections. To customize the influence (**D2**), a mapping function can be applied to each connection to create non-linear effects, discontinuous transitions, or to

model conditional and stateful behavior (**D3**) (see Section 3.3.1.2). In our workflow, non-trivial behavioral logic is prepared by a programmer and its parameters are exposed to the content author as object properties. The method thus shifts the level of abstraction towards the authoring of scenarios based on ready-to-use template functionality, in contrast to programming by the composition of basic building blocks. Novice users are not required to learn a modeling language but benefit from the high level of abstraction of the provided templates.

We propose three types of *source* properties: *Object Property*, *Time*, and *Relation*. To achieve conceptual simplicity while being easily extensible, we choose to model most inputs as object properties. To create temporal and context-aware behaviors, two additional types of sources are necessary: *Time* sources expose additional playback controls, and *Relation* sources can reference multiple objects.

- ***Time***: Progression of time as a continuous input enables modeling of temporal dynamics, e.g., changing object properties over time or controlling previously recorded animations.

- ***Relation***: Spatial relations between scene objects, such as the distance or angle, can express contextually responsive behaviors. A virtual agent might, e.g., change its mood (anxious, angry, or scared) whenever the user looks directly at it.

- ***Object Property***: Any other scalar-valued input that can influence the scenario. This can, e.g., include universal properties of objects in the scene (scale, movement speed), attributes specific to a class of object (rotation speed of a fan, battery charging status), scene properties (music loudness, sunlight intensity), or external input sources (user interaction, network input).

As an example of external input sources, we enable user input by introducing an avatar as a proxy object, carrying the left- and right-hand controllers. It can be placed into the scene by dragging a body-referenced miniature avatar attached to the user's chest. This way, the user's position, controller positions, and additional controller inputs (buttons, joysticks) can be used as input sources like any other object in the scene. It further enables to observe effects from a third-person perspective, e.g., by moving the avatar closer or farther away from an affected object (see Figure 3.13b).

*Target* properties can similarly be defined as any scalar-valued parameter that affects an individual object's behavior (virtual agent mood, crowd simulation parameter) or visual appearance (scale, transparency), or global scene properties (sky color tint, rain intensity). Each target property is influenced by a single source property. To combine multiple inputs, an operator (add, multiply, average) could have been utilized. This would, however, introduce another layer of structural complexity. To keep the method easily approachable by novices, we chose to offer only a single input per target.

### User Interface Designs

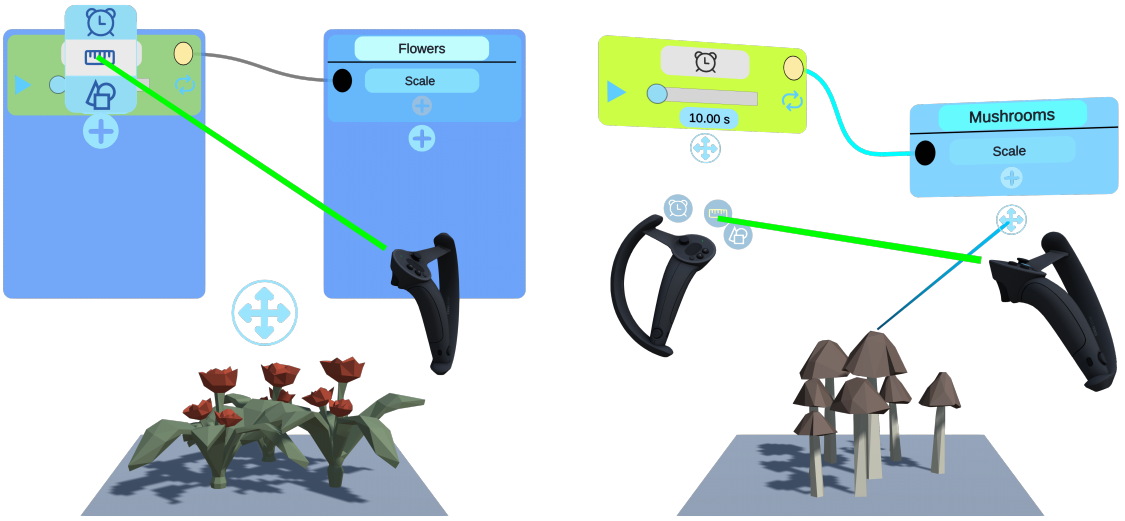
To choose a suitable spatial layout for our user interface elements (**D4**), we reviewed previous work and found the impact of reference frames on immersive authoring through visual programming not addressed in the literature. Therefore, we developed two interface designs for creating modulation mappings using a *surround-referenced* and an *object-referenced* layout that we evaluate in a comparative user study (see Section 3.3.2).

Both UIs utilize the same menu elements to represent source and target properties. We chose adapted 2D menus as a familiar interface to improve the overall user experience [Bowman et al., 2004]. The *Time* element contains a draggable input field for duration, a play button, and a looping toggle. The *Relation* element has two input fields for selecting objects (see Figure 3.8). The *Object Property* element contains an input field for object selection and a button (+) to add properties. Interaction is realized by a ray-based selection technique. Object selection can be performed by pointing (long range), grabbing (close range), or via a drop-down menu. The latter enables to select objects without graphical representation (e.g. scene lighting or background music). To create mappings, the user drags connecting lines between the ports of the respective properties.

**Surround-Referenced Layout:** Our motivation for the *surround-referenced* layout was to ensure that users can interact with minimal effort [Polys et al., 2005] and no travel is necessary to draw connections. We designed an interface on a panel comprising two distinct columns: the left for source properties and the right for target properties (see Figure 3.9a). The objective of the two-column view was to facilitate point-to-point connections that can be traced with minimal cognitive effort. Since all modulation mappings are co-located on a single panel, the interface needs to be kept within the user's reach. Therefore, the interface is anchored relative to the user's surrounding (i.e. the tracking space), such that it follows the user when traveling.

To create modulation mappings, the desired source and target properties have to be added to the respective column. Pressing + on the source panel opens a drop-down list in which *Time*, *Relation*, or *Object Property* can be selected. After selection, a corresponding menu element is created on the panel. Pressing + on the target panel creates a menu element and initiates object selection.

**Object-Referenced Layout:** Our motivation for the *object-referenced* layout, used in previous works [Ens et al., 2017; Solirax, 2018b; Chauvergne et al., 2023], was to maintain the spatial relationship between the programming elements and the scene objects. Interacting spatially by reaching into the scene to draw connections was found intuitive in similar use cases and could improve the overall user experience [Ens et al., 2017].




(a) *Surround-referenced*: Object properties are located on a 2D panel with a two-column interface to create connections. The panel is positioned relative to the tracking space of the user and follows along during travel.

(b) *Object-referenced*: Object properties are attached directly to the scene objects and connections are created between them. Properties without a visual representation are instantiated via buttons on the left controller.

**Figure 3.9:** Our *surround-referenced* and *object-referenced* interface designs to define modulation mappings. All interface elements can be repositioned freely using a handle (blue crossed arrows) relative to their respective frame of reference.

Each object has its own *Object Property* menu element that can be opened by selecting the object. The element appears above the object and can be repositioned by dragging its handle. Pressing + opens a drop-down list for selecting source and target properties, visually distinguished by source properties having a port on the right, and target properties on the left. For *Time*, *Relation*, and objects without graphical representation, we designed a hand-held menu that contains 3D buttons with corresponding icons. Selecting an icon instantiates the corresponding menu element (see Figure 3.9b).

#### 3.3.1.2 Mapping Functions

A mapping function can be applied to customize how the source property affects the target property. To facilitate this, we designed a 2D graph interface (see Figure 3.10-left). Hovering over a connecting line of a mapping displays a graph icon. Selecting this icon opens the graph interface in front of the mapping panel for the *surround-referenced* or the target object's menu for the *object-referenced* case.

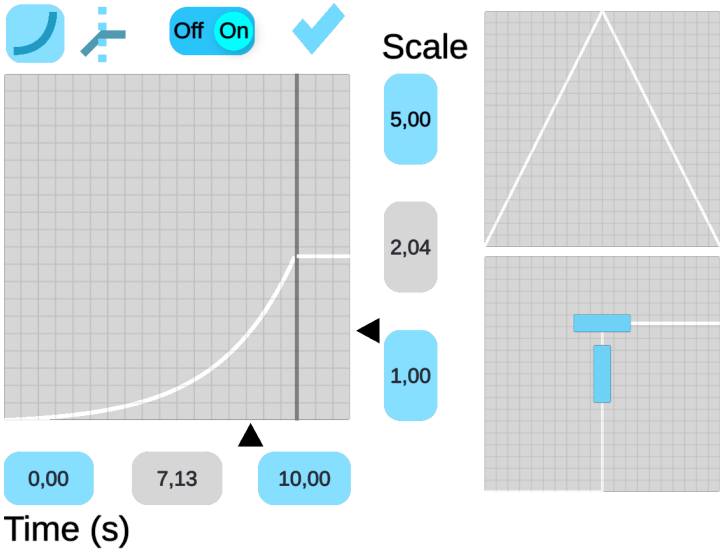



**Figure 3.10:** The mapping function interface (left) can be used to choose a template function, adjust the source and target ranges, enable or disable the mapping, and optionally toggle the *One-shot* option. Examples of template functions (right) are *Triangle* and *Step*.

We provide template functions with visual representations for users to customize modulation mappings without overwhelming them, especially those with limited mathematical knowledge (**D2**). Our templates include *Linear Increase* and *Decrease* for proportional changes, *Exponential Increase* and *Decrease* to create non-linear dynamic behavior, and *Triangle* and *Inverted Triangle* for cyclical patterns like oscillations or repeating effects.

To realize discrete conditions and events (**D3**), we provide a *Step* and its inverted function. The *Step* can be used to model conditional behavior, where the output value switches depending on whether a threshold is exceeded. We offer custom draggable handles for direct control of the threshold and target value (see Figure 3.10-right).

All functions can be toggled to act as a *One-shot* function, disabling the mapping after the threshold is reached to create stateful behavior. For example, a group of plants could grow when the water level rises due to rain. Using a *One-shot* mapping, the plants will retain their size after the rain stops. The user can design and test the effects of the *One-shot* function while the graph view is open; once closed, the mapping triggers once and deactivates when the threshold is crossed, until reactivated by the on/off switch.

The interface includes a graph view of the mapping function and draggable input fields for the source value range $[s_{min}, s_{max}]$ and target value range $[t_{min}, t_{max}]$, depicted along the X- and Y-axis, respectively. To compute the target value $t$, the source property $s$ is normalized to the interval $[0, 1]$, transformed by the mapping function $f : [0, 1] \rightarrow [0, 1]$,

and mapped to the target value range.

$$t(s) = \begin{cases} t_{min} & \text{if s} \leq s_{min} \\ t_{max} & \text{if s} \geq s_{max} \\ t_{min} + f(\frac{s - s_{min}}{s_{max} - s_{min}}) * (t_{max} - t_{min}) & \text{else} \end{cases} \tag{3.1}$$

The user can observe the current value via black arrowheads at both axes, as well as numerical feedback in gray text boxes. These are continuously updated to reflect a changing source value. This direct feedback enables users to understand how the source value is translated to the target value. Thereby, the user can test the effects of the mapping function on the scene while designing it in VR.

### 3.3.2 Empirical Evaluation

To provide an immersive authoring technique by visual programming that offers a good user experience, it is important to use a suitable spatial layout for its programming elements (**D4**). Since previous work did not investigate the role of reference frames in this context, our main research investigates the benefits and limitations of two interface layouts regarding the applicability of modulation mapping. Therefore, we designed an empirical user study to evaluate the *surround-referenced* (**SR**) and *object-referenced* (**OR**) layouts in the context of immersive authoring of reactive behavior. We determine the applicability of our technique by the efficiency, cognitive load, and user-friendliness. Our study design was guided by the following hypotheses:

Regarding efficiency:

- The mean task completion time will be lower for **SR** (**H1a**).
- The differences in task completion times between **SR** and **OR** depend on the task complexity (**H1b**).

Regarding cognitive load:

- The mean score for task load will be lower for **SR** (**H2**).

We operationalize user-friendlines based on the measures stated in the following hypotheses:

- We expect the mean reported clutter to be higher for **OR** (**H3**), because user interface elements attached to scene objects can obstruct the user's view [LaViola Jr

et al., 2017].

- To gain insight into how users perceive the layouts, we formulate the following undirected hypotheses. The mean scores for the six User Experience Questionnaire (UEQ) scales (**H4**), ease of use (**H5**), ease of learning (**H6**), and preference (**H7**) will be different based on the interface layout.

#### 3.3.2.1 Study Design and Tasks

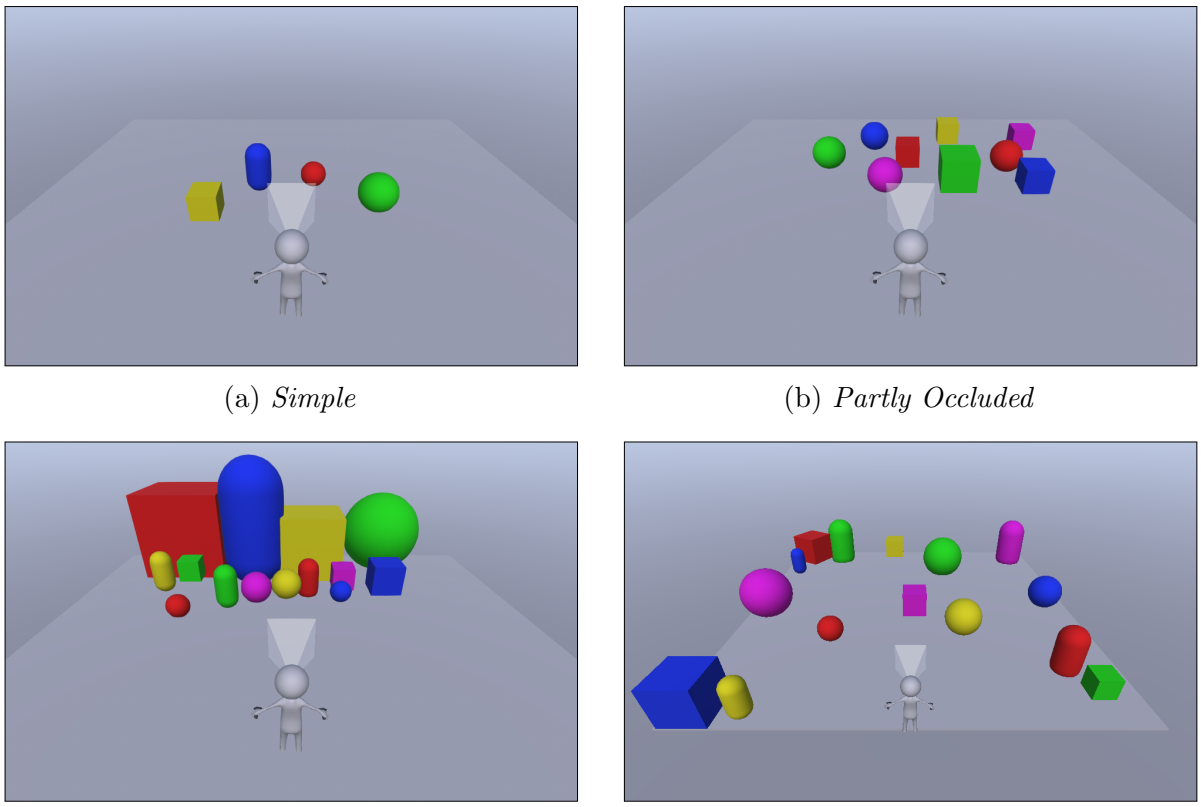
We devised a within-subject design with two subsequent parts to investigate our hypotheses regarding the *object-referenced* and *surround-referenced* layouts as two levels of one factor. In the first part of the study (see Section 3.3.2.1), participants were asked to create modulation mappings in abstract scenarios that contained only primitive objects with simple colors. The object properties were denoted with letters from A to F (see Figure 3.12). Our motivation was to evaluate task completion times (**H1**), task load (**H2**), and perceived clutter (**H3**) in a controlled setting without additional biases induced by the semantics and visual complexity of real-world scenarios. We aimed to get a clearer view of the interface design's performance without too much influence by the conducted task.

In the second part, participants were asked to create modulation mappings within a realistic scenario (Section 3.3.2.1) without time constraints. This part of the study aimed to resemble real-world use cases more closely, so that we could gain insight into both interfaces' practical applicability and gather empirical data with a higher level of external validity compared to the first part of the study. Without time pressure, participants can thoroughly evaluate the interfaces to provide feedback on their experience (**H4**), ease of use (**H5**), ease of learning (**H6**), and overall preference (**H7**).

##### Abstract Scenarios

To test **H1**, we created a second factor that varies the complexity of the task. When evaluating and assessing different testing scenarios, we found that the object arrangement has a major influence on the task complexity. Therefore, while we use abstract objects with a minimal appearance to avoid distractions, we choose their positioning to resemble typical scene arrangements that can be encountered in immersive authoring.

Individual objects may, for example, be laid out next to each other and easily reachable, be partially obstructed, or have different sizes. Objects may also be located at a distance from each other, such that users have to change their field of view to understand their spatial relation. We considered previous work [Steed, 2006] and went through an iterative



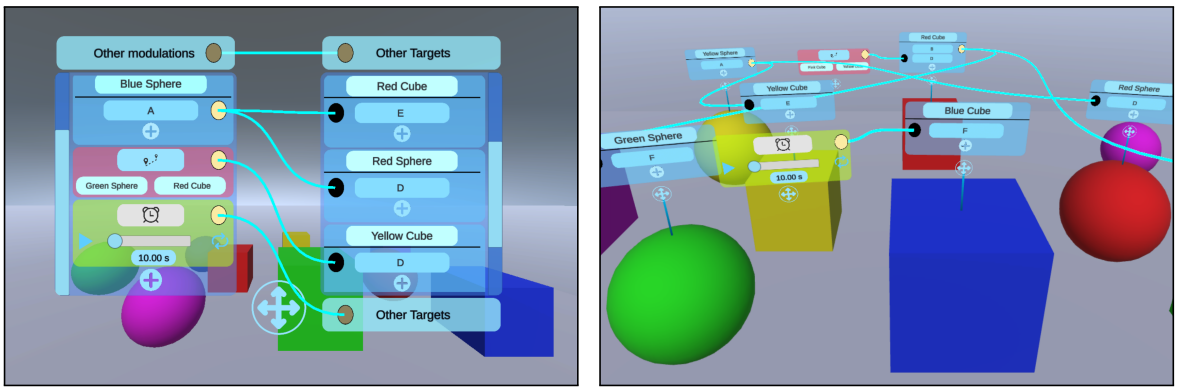
(a) *Simple* (b) *Partly Occluded*

(c) *Varying Scale* (d) *Distant*

**Figure 3.11:** Overview of part one of the study, showing the abstract scenes.

process in which we included VR experts from our lab with the goal of producing a set of scenes with representative object arrangements. Eventually, we derived four scenes that we denote as *Simple*, *Partly Occluded*, *Varying Scale*, and *Distant* (see Figure 3.11).

The specific objects involved in each task and the corresponding number of menu elements were chosen to reflect the quality of each scene. In the *Simple* scene, 4 objects are laid out next to each other. To complete the task, 6 menu elements and 4 connections have to be created. For the *Partly Occluded* scene, we arranged 9 objects such that the menu elements partially occlude each other, and users have to create 9 menu elements and 6 connections. The *Varying Scale* scene contains 15 objects, where small ones are located in front of bigger ones to avoid the selection of fully occluded objects. The task involves creating 9 menu elements and 5 connections between objects of different sizes. Regarding *Distant*, there are 15 objects and participants have to create 8 menu elements and 10 connections. The mappings were chosen such that connections need to be made between far-apart objects, requiring participants to travel and repeatedly change their field of view.




(a) *Partly Occluded, Surround-referenced* (b) *Partly Occluded, Object-referenced*

**Figure 3.12:** Exemplary view of the abstract scenarios during the first part of the user study.

| | |
|---|---|
| **Task 1** | Modulate the scale of the two groups of flowers by the background music loudness. |
| **Task 2** | Lower the bridge based on the distance of the avatar to the gate. |
| **Task 3** | Change the mood of the sky and the rain intensity based on the distance of the avatar to the totem. |
| **Task 4** | Open the chest as the avatar approaches with the right controller. |
| **Task 5** | The treasure inside of the chest should fade in and out continuously. |

**Table 3.1:** Task descriptions in the realistic scenario.

We used hand-based steering rather than teleportation to enable users to draw connections while traveling. In all scenes, objects are placed within an area of 10 by 10 meters. Instructions are listed on a panel attached to the left controller, which might read: $GreenCube(A) \rightarrow RedCube(D)$. The specific instructions that were used in the study and a Unity package that contains the abstract scenarios can be found at https://zenodo.org/records/10627368.

In this first part of the study, the mapping function interface (see Fig 3.10) was disabled. Since the interface is identical for both layouts, it would introduce unnecessary variance to the measured effects we investigate by **H1** to **H3**. Regarding **H2** and **H3**, we look at the main effect caused by the layout, resulting in a one-factorial (layout) within-subject design. Participants ranked questionnaires regarding task load and perceived visual clutter after completing all four tasks for each respective interface layout.


(a) *Task 1, Object-referenced* (b) *Task 2, Surround-referenced*

**Figure 3.13:** Exemplary view of the realistic scenarios during the second part of the user study.

#### Realistic Scenarios

We created a virtual scenario for participants to perform five tasks (see Table 3.1) that consisted of adding reactive behaviors to different scene objects. The tasks were designed to cover the key functionalities of our presented technique. This includes the utilization of object properties as sources and targets, distance relations between objects, time-modulated behaviors, and custom mapping functions. All tasks were performed in the same order to construct a coherent narrative. To validate our hypotheses **H4** to **H7**, we chose a one-factorial within-subject design, where participants ranked subjective questionnaires after completing all five tasks in each condition.

### 3.3.2.2 Apparatus

The study took place in our lab, with participants seated on a rotatable chair. We used a Valve Index HMD with its controllers, tracked by four Lighthouse 2.0 base stations. The experimental platform was developed in Unity 2021.3.5f1, running on a Windows 10 with a 3.7GHz Intel Core i9-10900X CPU and an NVIDIA RTX 3090 GPU.

### 3.3.2.3 Procedure

Upon arrival, participants signed a written informed consent form and completed a pre-study questionnaire about demographics and experience in VR, programming, and

visual programming.

In part one of the study, participants went through a familiarization phase before executing the tasks with each layout. They received instructions from a panel and watched demonstration videos in VR. They practiced creating modulation mappings for up to 15 minutes. Participants then performed the tasks in the four scenes (see Section 3.3.2.1). Once the first menu element had been instantiated, completion times were recorded and the participant could start traveling. The order of the layouts and scenes was counterbalanced using a Latin Square. After completing all four scenes for the given layout, participants answered NASA-TLX [Hart, 2006] and a 7-point Likert scale custom questionnaire to rate visual clutter.

In part two of the study, participants first entered another familiarization phase to try out the mapping function graph and learn to place the avatar into the scene. Afterward, they were asked to create reactive behaviors in a realistic scenario (see Section 3.3.2.1). After completing all tasks for a given layout, participants answered UEQ [Laugwitz et al., 2008], a 7-point Likert scale questionnaire on preference, ease of learning and use, and freely commented on the recently used layout. The layouts were assigned in the same order as in part one.

Lastly, they provided free comments on the modulation mapping technique and answered a 7-point Likert scale questionnaire regarding its effectiveness and ease of learning the mapping function graph interface. The entire procedure took approximately 90 min.

#### 3.3.2.4 Participants

We recruited 36 participants on the university campus through various communication channels, but we had to exclude two due to technical issues. The remaining 34 participants (7 female, 26 male, 1 non-binary) were aged between 20 and 41 ($M = 27$ years old, $SD = 4.85$), all right-handed with normal or corrected-to-normal vision. Their experience levels in VR applications ranged from 2 novices, 6 beginners, 9 advanced, to 17 experts, in programming from 1, 1, 7, to 25, and in visual programming from 8, 14, 9, to 3.

### 3.3.3 Results

To determine statistically significant differences between the layouts for varying task complexity, we analyzed the measurements using 2x4 factorial repeated-measures ANOVA. We further computed a two-sided paired-sample t-test, concerning variables that were

measured only once per condition without a second factor. When sphericity assumptions were not met, we carried out Mauchly's tests and applied Greenhouse-Geisser correction. Additionally, when we observed a statistically significant overall effect, we conducted Bonferroni-corrected post-hoc tests for pairwise comparisons.

Effect sizes for ANOVA were assessed with $\eta_p^2$ (thresholds: $.01, .06$, and $.14$ for small, medium, and large effects), while t-test effect sizes were computed using Cohen's d (thresholds: $.2, .5$, and $.8$ for small, medium, and large effects) [Cohen, 2013]. For all statistical tests, we assume a significance level of $p < .05$, marked as (*) in the figures. Box plots show the median (-), mean (x) and interquartile range ($IQR$), with whiskers indicating $IQR * 1.5$.

#### 3.3.3.1 Objective Measures

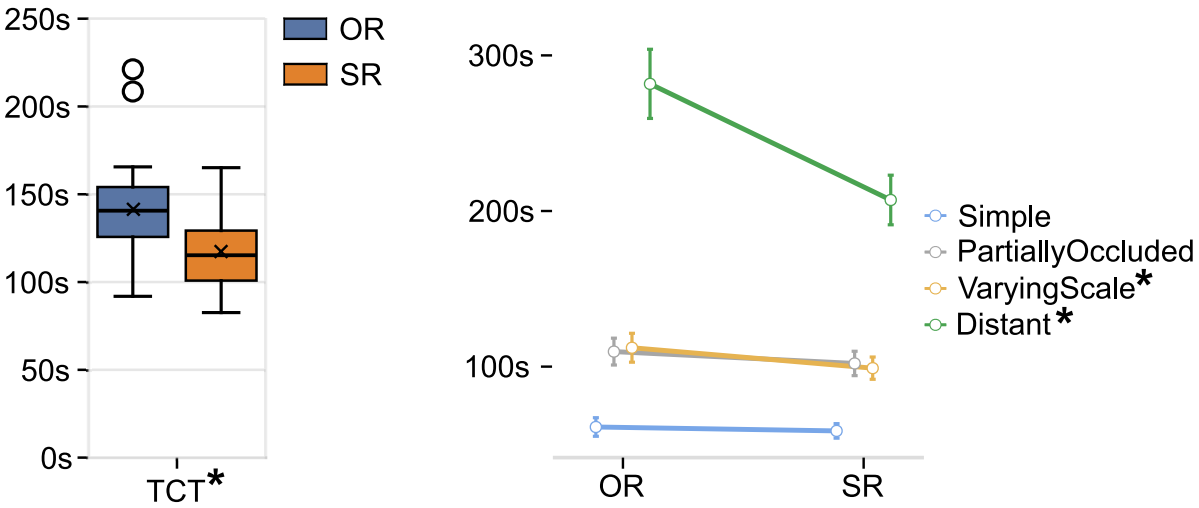



**Figure 3.14:** Mean task completion time (TCT): Box plots of the main effect (left). Marginal means per condition show the interaction between layout and task (right), whiskers indicate the 95% confidence interval.

Regarding **task completion time**, we observed a significant main effect of layout ($F(1, 33) = 56, p < .001, \eta_p^2 = .629$), supporting **H1a**. We further observed a significant interaction effect between layout and scene ($F(1.47, 48.44) = 40.1, p < .001, \eta_p^2 = .549$). Post-hoc tests revealed two significant differences between the layouts in *Varying Scale* ($p = .039, d = .598$) and *Distant* ($p < .001, d = 1.296$), supporting **H1b**. To understand the underlying factors, we recorded the time spent interacting with menus, dragging connecting lines, traveling, and the amount of head rotation per task (See Figure 3.14). Descriptive statistics are in Table 3.2.

Menu **interaction time** includes pressing buttons, scrolling, and repositioning menus.

| | | Task Time [$s$] | Travel Time [$s$] | Menu Time [$s$] | Drag Time [$s$] | Head Movement [°] |
|---|---|---|---|---|---|---|
| S | O | 61.2 (17.11) | 2.01 (3.73) | 3.17 (1.23) | **6.03** (2.22) | 1192.26 (442.95) |
| | S | 58.7 (13.39) | 0.17 (0.76) | 3.51 (1.59) | **2.93** (1.16) | 1013.34 (324.42) |
| PO | O | 109.66 (24.74) | **9.69** (8.38) | 5.77 (3.4) | **10.67** (3.96) | 1891.61 (590.3) |
| | S | 102.04 (22.45) | **0.62** (1.82) | 5.63 (1.79) | **4.34** (1.48) | 1842.16 (627.45) |
| VS | O | **112.13** (26.61) | **8.76** (9.52) | 6.15 (4.45) | **11.88** (5.49) | 2031.79 (692.79) |
| | S | **99.01** (20.51) | **0.26** (0.96) | 5.74 (1.05) | **4.07** (1.81) | 1820.79 (661.57) |
| D | O | **281.71** (63.69) | **50.9** (33.24) | 15.62 (8.33) | **38.38** (17.64) | **7413.98** (1911.4) |
| | S | **207.08** (45.66) | **1.24** (4.99) | 12.47 (5.78) | **8.38** (2.94) | **4299.45** (1359.3) |

**Table 3.2:** Mean (standard deviation) for *Simple* (S), *Partly Occluded* (PO), *Varying Scale* (VS) and *Distant* (D) scenes using the *object-referenced* (O) and *surround-referenced* (S) interface layouts. Significant differences are indicated in bold.

We observed no significant main effect of layout ($F(1, 33) = 1.88, p = .180, \eta_p^2 = .054$).

Concerning **drag time**, we observed a significant main effect of layout ($F(1, 33) = 149.8, p < .001, \eta_p^2 = .819$), and a significant interaction effect between layout and scene ($F(1.27, 41.93) = 79.2, p < .001, \eta_p^2 = .706$). Post-hoc tests revealed significant differences between layouts in all scenes ($p_{all} < .001, d_{all} \geq 1.26$).

For **travel time**, we observed a significant main effect of layout ($F(1, 33) = 84.6, p < .001, \eta_p^2 = .719$), and a significant interaction effect between layout and scene ($F(1.20, 39.59) = 68.2, p < .001, \eta_p^2 = .674$). Post-hoc tests revealed significant differences between layouts in all scenes except *Simple* ($p_{all \setminus Simple} < .001, d_{all \setminus Simple} \geq .903$).

Regarding **head rotation**, we found a significant main effect of layout ($F(1, 33) = 60.9, p < .001, \eta_p^2 = .648$), and a significant interaction effect between layout and scene ($F(1.46, 48.07) = 104.8, p < .001, \eta_p^2 = .760$). Post-hoc tests showed significant differences between layouts only in *Distant* ($p_{Distant} < .001, d_{Distant} = 1.83$).

#### 3.3.3.2 Subjective Measures

We conducted a two-sided paired-sample t-test for the following variables since they were measured only once per layout.

The mean **task load** is significantly lower for SR than OR ($T(33) = 5.19, p < .001, d = .891$), supporting **H2**. To investigate the factors determining task load, we examine the NASA TLX sub-scales (see Figure 3.15) and except Temporal Demand ($T(33) = .96, p =$

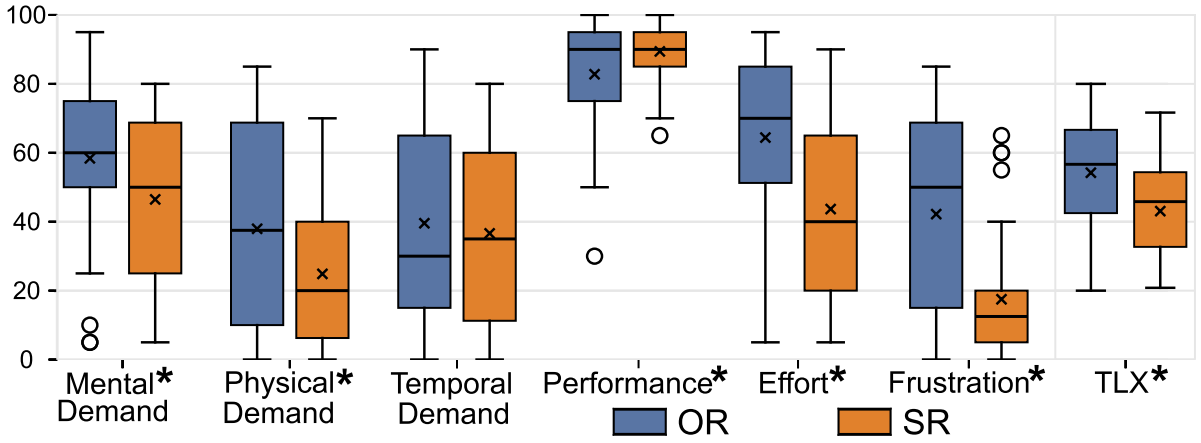



**Figure 3.15:** Box plots of NASA-TLX sub-scores and total scores.

.343), we observed significant differences. SR is significantly lower in Mental Demand ($T(33) = 3.94, p < .001, d = .68$), Physical Demand ($T(33) = 2.51, p = .017, d = .43$), Effort ($T(33) = 4.39, p < .001, d = .75$) and Frustration ($T(33) = 5.52, p < .001, d = .947$) and higher in Performance ($T(33) = 3.17, p = .003, d = .54$) compared to OR.

Regarding **user experience**, significant differences were observed in five out of six scales of the UEQ, except Stimulation, which partially supports **H4**. SR significantly outperformed OR in Attractiveness, Perspicuity, Efficiency, and Dependability, while OR significantly outperformed SR in Novelty. See Figure 3.16 for the distribution, statistical test results, and descriptive statistics.

Concerning the **subjective questionnaires**, significant differences were found in **Visual Clutter** ($T(33) = -6.80, p < .001, d = -1.165$), **Ease of Learning** ($T(33) = 2.04, p = .049, d = .35$), **Ease of Use** ($T(33) = 2.94, p = .006, d = .504$), and **Preference** ($T(33) = 2.32, p = .027, d = .398$), supporting **H3, H5, H6, H7**. Participants also rated the **Effectiveness** of the modulation mapping technique ($M = 6.38, SD = 1.10$) and the **Ease of Learning** of the mapping function ($M = 6.32, SD = .97$). See Figure 3.17 for the results.

#### 3.3.3.3 Qualitative Results

We analyzed the comments on layouts and modulation mapping using thematic analysis [Braun and Clarke, 2012] and present the findings in this section.

Regarding *object-referenced*, 12 participants found it **intuitive** and **engaging** to interact directly with objects in the scene to define behavior. P9 stated, *"It is really intuitive and feels immersive since I was able to place the mapping behavior directly to the ob-*

| | Statistics | | | Mean (SD) | |
|---|---|---|---|---|---|
| **UEQ Scales** | *t* | *p* | *d* | **OR** | **SR** |
| Attractiveness | 2.50 | 0.017 | 0.43 | 1.56 (1.08) | 1.99 (0.56) |
| Perspicuity | 2.58 | 0.014 | 0.44 | 1.74 (0.82) | 2.14 (0.71) |
| Efficiency | 4.73 | $<0.001$ | 0.81 | 0.87 (1.30) | 2.00 (0.68) |
| Dependability | 4.15 | $<0.001$ | 0.71 | 1.41 (0.85) | 1.99 (0.68) |
| Stimulation | 1.05 | 0.302 | 0.18 | 1.68 (1.02) | 1.85 (0.67) |
| Novelty | -2.66 | 0.012 | -0.46 | 1.46 (0.83) | 0.94 (1.06) |

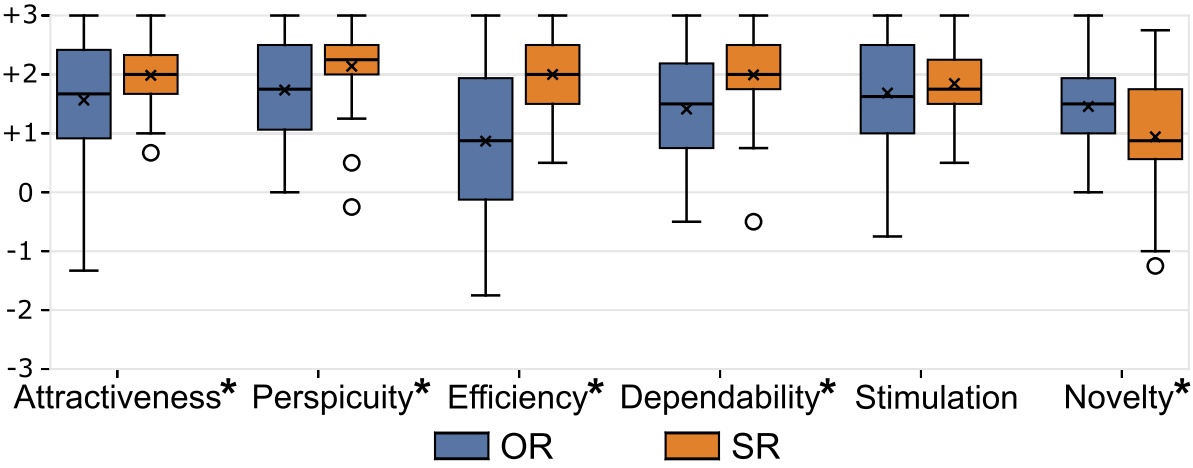



**Figure 3.16:** UEQ sub-scales: Paired sample t-test statistics, mean (standard deviation), and distributions for each layout.

*jects"*. Five participants (P12, P20, P25, P33, P34) stated that selecting properties and connecting ports from a **distance can be challenging**, especially with objects far apart. The layout was considered valuable for tasks like **visual debugging** (P9, P12, P15, P26), however, some participants also noted potential issues with **clutter** (P15, P24, P26, P33, P35) when dealing with a large number of connections. Moreover, P7 suggested automated UI scale adjustment and layouting.

Regarding *surround-referenced*, eight participants (P2, P9, P15, P17, P20, P26, P31, P33) appreciated the **efficiency** and **ability to access** the mappings **without requiring travel**. However, seven participants (P1, P20, P21, P24, P26, P30, P33) stated that with numerous mappings, the interface can become **cluttered**, making it harder to navigate and **maintain an overview**. P31 stated, *"The only downside is having to scroll more for objects at the top and I can imagine that in more complex scenarios the overview can get lost when you can't see the mappings at once"*. P1 and P19 suggested **grouping** options for better list organization.

Regarding the *modulation mapping technique*, 15 participants found authoring reactive behaviors by creating direct connections **easy to use** and **intuitive**. Some participants

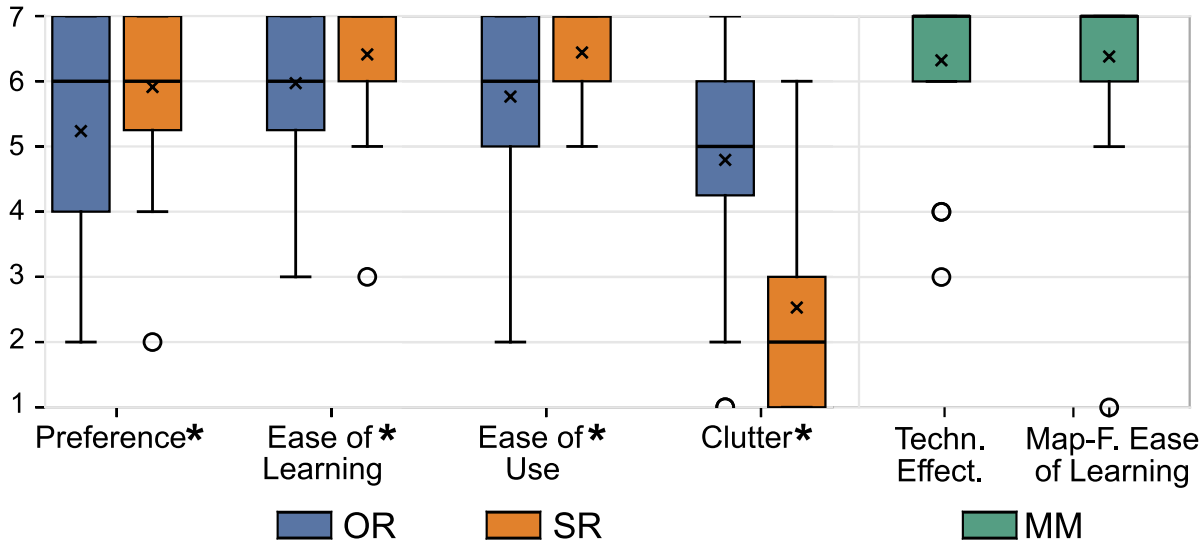



**Figure 3.17:** 7-point Likert questionnaire, with items ranging from very strongly disagree(1) to very strongly agree(7). Comparison of the layouts (left), and evaluation of the modulation mapping (MM) technique (right) regarding its effectiveness and the ease of learning of the mapping function interface.

appreciated the real-time feedback and customization of behaviors via predefined mapping functions. P11 and P35 found the technique beneficial to expert programmers: VR expert P11 stated, *"not only novices but also experienced programmers can benefit from the technique for quick testing of the behaviors without diving into the code or taking the HMD off"*.

Four participants (P19, P25, P32, P27) proposed **combining elements of both layouts**. P26 stated, *"I liked using surround-referenced, however, I was missing the connection to the virtual environment from the mappings"*. Some participants suggested drawing indicator lines from the SR panel to the respective scene objects upon hover. In addition, two VR experts (P27, P31) suggested to enable instantiating multiple mapping panels that can be placed next to scene objects and summoned when needed.

### 3.3.4 Discussion

We found **SR** to outperform **OR** in terms of faster task completion. The results further revealed an interaction effect between layout and scene. While we observed significant differences in task completion time for *Varying Scale* and *Distant* scenes, there were no significant differences for *Simple* and *Partly Occluded*. Based on our secondary results, a possible explanation could be that the **OR** layout requires additional interactions (travel, head movement) or poses precision challenges when drawing connecting lines (drag time),

which is more evident when objects are at a distance or have varying scales.

Regarding task load, participants found creating modulation mappings using **OR** significantly more mentally and physically demanding, and frustrating compared to using **SR**. We speculate that the increased head movement and travel for drawing connections between distant objects led to split attention, increasing the mental load compared to the **SR** condition, where interactions could be performed co-located on the SR panel [Sweller et al., 1998]. Participants expressed frustration when ray-casting precision issues caused failed connections, requiring more physical effort. With **SR**, the interface elements are often reachable in close proximity, thereby the mapping creation may be achieved with lower effort.

Placement of menus can clutter a VE and occlude the user's view [Bowman et al., 2004]. Since in the **OR** layout, menus are placed in the environment, we hypothesized that **OR** clutters the scene more than **SR**. The results support our hypothesis, showing that **OR** significantly cluttered the view more. This suggests that **SR** provided a more streamlined interaction, contributing to better user experience and task performance. These results are in line with previous work [Rosenholtz et al., 2007; Bacim et al., 2013; Ragan et al., 2015] and our own findings regarding **H1a** and **H4**.

Concerning user experience, **SR** was rated higher than **OR** for attractiveness, perspicuity, efficiency, and dependability. Furthermore, **SR** was found easier to learn, easier to use, and preferable over **OR**. We believe that **SR**, by placing interface elements within arm's reach, provided better overview and easier access, thereby reducing mental and physical load. This, in turn, improved the overall user experience. Our findings are in line with previous work [Hu et al., 1999], showing that interfaces with lower cognitive load lead to higher user satisfaction. When users find an interface easy to learn (**H5**), easy to use (**H6**), and not mentally demanding (**H2**), they tend to have a more positive overall perception of the interface (**H4**, **H7**).

Given the high ratings for our technique's effectiveness, ease of learning the mapping function, and overall positive feedback, modulation mapping shows potential for enabling users to easily and effectively create reactive behavior in immersive environments. Our technique appears to be a promising and valuable addition to existing immersive authoring tools.

#### 3.3.4.1 Design Implications

Based on our empirical results, we propose initial design implications regarding the choice of reference frame for user interfaces in immersive authoring by visual programming.

We suggest to use *surround-referenced* interface layouts where efficiency and streamlined

workflows are a priority, especially for scenes with a large number of objects. However, consider to organize the property lists clearly by providing tools to navigate and maintain an overview of connections. This type of interface would be most suitable for behaviors that do not strongly rely on spatial relationships between objects. Additional visual feedback, such as drawing indicator lines between mappings and the respective scene objects upon hover, may further support the authoring process.

The *object-referenced* interface layout was found intuitive, engaging, and valuable for visual debugging. However, it can become cluttered with numerous objects. Therefore, use it for scenes with fewer objects and behaviors that strongly rely on their spatial configuration. Consider (semi-)automatic layouting [Satriadi et al., 2020], grouping, and hiding of objects to minimize clutter [Ens et al., 2017]. These could be guided, e.g., by estimating the cognitive load during task execution [Lindlbauer et al., 2019]. Consider selection techniques that address the ray-casting precision issues [Krüger et al., 2024] to select properties and connect nodes at a distance.

Generally, we believe that a *hybrid* approach that combines elements of both *surround-* and *object-referenced* layouts may leverage their respective strengths and mitigate their weaknesses. We see interesting opportunities for future research in immersive reactive behavior authoring that explores how both approaches can be linked in a synergistic way.

### 3.3.5 Limitations and Future Work

Our study design considers task complexity as a factor influencing efficiency. To vary task complexity, we altered the spatial arrangement of the objects alongside the number of required interactions. We found both related factors relevant to achieve a representative and diverse set of tasks, however, they can be investigated separately in future trials. Further, we only explored static horizontal arrangements of objects. Future research could examine authoring reactive behavior in scenes with moving, or vertically arranged objects. In this initial design, we used ray-casting for selection and hand-directed steering for travel. Since interaction techniques impact task performance and user experience, our future research will evaluate different selection and travel techniques in this context.

While our participants completed all tasks successfully and provided overall positive feedback on the modulation mapping technique, future research may conduct further user studies regarding its effectiveness in different authoring scenarios and application domains.

### 3.3.6 Conclusion

In this section, we presented *modulation mapping*, a simplified approach for immersive authoring of reactive behavior by dataflow visual programming. In a comparative user study, we compared a *surround-* to an *object-referenced* interface layout regarding task efficiency and user experience. From the results, we derived an initial set of interface design implications for future research and development of similar immersive authoring techniques. Participants found modulation mapping an intuitive and effective way of authoring reactive behavior while immersed. At the same time, experts in VR and programming found the technique a valuable addition to a developer's workflow for authoring immersive environments.

CHAPTER 4

# CONCLUSION

In this thesis, we investigated how to enable users to build interactive virtual environments within VR. To address this, we devised and empirically evaluated user-friendly immersive authoring techniques and interfaces that simplify the creation of both content and interactivity. To this end, we divided our work into two parts.

The first part of the thesis focused on content authoring techniques for creating and arranging 3D scene elements, which are essential for building virtual environments. In this regard, we presented three novel authoring techniques at increasing levels of abstraction, each tailored to different use cases and interaction modalities to support a range of authoring needs. First, we introduced an artistic workflow that enables users to transform 2D paintings into expressive 3D artworks with minimal effort. This was achieved via a novel 3D sculpting technique that supports intuitive shape manipulation. With our approach, creative professionals can author 3D geometric content without the need for prior knowledge in mesh processing or 3D modeling. Second, we developed and evaluated a sketching-based procedural generation method for creating fluid-like 3D artworks. These can be further manipulated via a combination of mid-air gestures and a novel blowing-based interaction. This approach supports expressive creation and introduces an alternative input modality for shaping fluid-like content. Third, we developed a high-level scene authoring technique focused on efficient creation of simulation scenarios. Specifically, we designed a VR environment for authoring road networks for automated vehicle testing. We presented novel indirect free-hand interactions using a 2D WIM for object placement and travel. A comparative user study showed that our technique improved precision, reduced task completion time, and was found to be intuitive and preferred compared to direct free-hand interaction techniques. This chapter demonstrates that effective and intuitive 3D content authoring begins with understanding user needs. When creativity is prioritized, the techniques and workflows should enable users

to expand creative boundaries; when efficiency is the focus, techniques should enable fast creation, minimizing time and physical fatigue in VR.

In the second part, we focused on the immersive authoring of content behavior. Specifically, we investigated how to enable users to define dynamic behavior and interactivity, which is an important factor in improving both realism and user engagement in VR. In this context, we introduced two approaches based on visual programming paradigms and investigated how to position programming interfaces within the scene to improve the user experience. The first approach provided a simplified block-based interface that enabled users to define interactive scenarios by arranging condition and action blocks in a sequential structure via drag-and-drop free-hand interactions. This method supported scenario creation in automated driving simulations and its effectiveness was demonstrated through a user study. The findings also highlighted the importance of interface positioning in improving user understanding and task execution. Following this, we introduced another simplified approach for authoring object behavior inspired by dataflow visual programming, allowing users to map input sources to predefined properties by drawing connections. To examine the impact of programming interface positioning, we developed two configurations: *object-referenced* (anchored to objects) and *surround-referenced* (anchored to surrounding of the user). We evaluated them in abstract tasks to get a clearer view of the performance, as well as in realistic scenarios in terms of user experience and preference to gain insight into practical applicability. We found that the surround-referenced interface was faster, less cognitively demanding, easier to learn, and preferred by users. In contrast, the object-referenced layout was seen as more intuitive and helpful for visual debugging. Based on the findings, we outlined design implications for behavior authoring via visual programming and demonstrate that our simplified approach supports intuitive and effective reactive behavior creation in VR.

Through these contributions, we made important progress toward addressing our main research objective by developing and evaluating novel approaches that simplify content and interactivity authoring. This work contributes valuable insights into the design of immersive authoring techniques and VR interfaces that support a broad range of creators in building interactive virtual environments. While our contributions demonstrate the effectiveness of the proposed approaches, we believe that immersive authoring still holds considerable potential for further advancement. While several future directions were outlined previously, we further identify the following possible research avenues. One promising direction is the integration of generative AI into immersive authoring workflows. Future work could investigate AI-driven approaches that support efficient content creation while also providing users more control over the final outcome. Another direction is the development of intelligent, adaptive, context-aware user interfaces that dynamically adjust their design or positioning based on user behavior, specific task requirements, or changing scene environment. These may reduce the cognitive load and the overall user experience in immersive content authoring. Lastly, content authoring in a collaborative immersive environment is an important aspect that enables users to co-create content, share ideas, and provide instant feedback. Future research should

explore efficient ways to collaborate on the same task, including intelligent strategies for managing conflicts that arise from simultaneous edits on objects or scenario creations.

Further advancing these research directions may contribute to a future in which users can more easily become the creators of virtual environments. By reducing technical complexity and improving support for creativity and task-oriented efficiency, immersive authoring systems can enable a broad range of users to construct interactive virtual experiences that reflect their individual goals, ideas, and imagination, ultimately positioning users not only as consumers but as active creators within virtual spaces.

# Bibliography


Abich IV, J., Parker, J., Murphy, J. S., and Eudy, M. (2021). A Review of the Evidence for Training Effectiveness with Virtual Reality Technology. *Virtual Reality*, 25(4):919–933.

Abras, C., Maloney-Krichmar, D., Preece, J., et al. (2004). User-Centered Design. *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications*, 37(4):445–456.

Academy, N. Y. F. (2014). How To Photograph Smoke. Available at: https://www.nyfa.edu/student-resources/how-to-photograph-smoke, last-visited: 2017-11-18.

Adami, P., Rodrigues, P. B., Woods, P. J., Becerik-Gerber, B., Soibelman, L., Copur-Gencturk, Y., and Lucas, G. (2021). Effectiveness of VR-based Training on Improving Construction Workers' Knowledge, Skills, and Safety Behavior in Robotic Teleoperation. *Advanced Engineering Informatics*, 50:101431.

Adobe (2020). Top 3D Sculpting Tools for Virtual Reality Authoring. Available at: https://www.adobe.com/products/medium.html, last-visited: 2025-05-15.

Altendeitering, M. and Schimmler, S. (2022). End-User Development for Smart Spaces: A Comparison of Block and Data-flow Programming. In *SMARTGREENS*, pages 15–22.

Armoni, M., Meerbaum-Salant, O., and Ben-Ari, M. (2015). From Scratch to "Real" Programming. *ACM Transactions on Computing Education (TOCE)*, 14(4):1–15.

Arora, R. and Singh, K. (2021). Mid-air Drawing of Curves on 3D Surfaces in Virtual Reality. *ACM Transactions on Graphics (TOG)*, 40(3):1–17.

Arqueros, N., Prieto, P., Zúñiga, M., et al. (2012). A New Tool for Immersive 3d Free-form Surface Modelling and Refining. In *DS 70: Proceedings of DESIGN 2012, the 12th International Design Conference, Dubrovnik, Croatia*, pages 365–372.

Arsenault, R. and Ware, C. (2004). The Importance of Stereo and Eye-coupled Perspective for Eye-hand Coordination in Fish Tank VR. *Presence: Teleoperators & Virtual Environments*, 13(5):549–559.

Artizzu, V., Cherchi, G., Fara, D., Frau, V., Macis, R., Pitzalis, L., Tola, A., Blecic, I., and Spano, L. D. (2022). Defining Configurable Virtual Reality Templates for End Users. *Proceedings of the ACM on Human-Computer Interaction*, 6(EICS):1–35.

ASAM (2024). OpenSCENARIO DSL. Available at: https://www.asam.net/standards/detail/openscenario-dsl, last-visited: 2025-05-21.

Azevedo, C. L., Deshmukh, N. M., Marimuthu, B., Oh, S., Marczuk, K., Soh, H., Basak, K., Toledo, T., Peh, L.-S., and Ben-Akiva, M. E. (2017). Simmobility Short-term: An Integrated Microscopic Mobility Simulator. *Transportation Research Record*, 2622(1):13–23.

Bacim, F., Ragan, E. D., Scerbo, S., Polys, N. F., Setareh, M., and Jones, B. D. (2013). The Effects of Display Fidelity, Visual Complexity, and Task Scope on Spatial Understanding of 3D Graphs. In *Graphics Interface*, volume 2, pages 25–32.

Barot, C., Carpentier, K., Collet, M., Cuella-Martin, A., Lanquepin, V., Muller, M., Pasquier, E., Picavet, L., Van Ceulen, A., and Wagrez, K. (2013). The Wonderland Builder: Using Storytelling to Guide Dream-like Interaction. In *2013 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 201–202. IEEE.

Bell, B., Rieman, J., and Lewis, C. (1991). Usability Testing of a Graphical Programming System: Things We Missed in a Programming Walkthrough. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 7–12.

Bellgardt, M., Pick, S., Zielasko, D., Vierjahn, T., Weyers, B., and Kuhlen, T. W. (2017). Utilizing Immersive Virtual Reality in Everyday Work. In *2017 IEEE 3rd Workshop on Everyday Virtual Reality (WEVR)*, pages 1–4. IEEE.

Benes, B., Zhou, X., Chang, P., and Cani, M.-P. R. (2021). Urban Brush: Intuitive and Controllable Urban Layout Editing. In *The 34th annual acm symposium on user interface software and technology*, pages 796–814.

Bergamini, L., Ye, Y., Scheel, O., Chen, L., Hu, C., Del Pero, L., Osiński, B., Grimmett, H., and Ondruska, P. (2021). SimNet: Learning Reactive Self-driving Simulations From Real-world Observations. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5119–5125. IEEE.

Bernatchez, M. and Robert, J.-M. (2007). A Study on the Impact of Spatial Frames of Reference on Human Performance in Virtual Reality User Interfaces. In *2007 IEEE International Conference on Systems, Man and Cybernetics*, pages 2600–2605. IEEE.

Billinghurst, M., Bowskill, J., Dyer, N., and Morphett, J. (1998). An Evaluation of Wearable Information Spaces. In *Proceedings. IEEE 1998 Virtual Reality Annual International Symposium (Cat. No. 98CB36180)*, pages 20–27. IEEE.

Bollobás, B. (2013). *Modern Graph Theory*, volume 184. Springer Science & Business Media.

Bowman, D. A., Kruijff, E., LaViola, J. J., and Poupyrev, I. (2004). *3D User Interfaces: Theory and Practice*. Addison Wesley Longman Publishing Co., Inc.

Bowman, D. A., Kruijff, E., LaViola Jr, J. J., and Poupyrev, I. (2001). An Introduction to 3-D User Interface Design. *Presence: Teleoperators & Virtual Environments*, 10(1):96–108.

Bozgeyikli, E., Raij, A., Katkoori, S., and Dubey, R. (2016). Point & Teleport Locomotion Technique for Virtual Reality. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*, pages 205–216. ACM.

Braun, V. and Clarke, V. (2012). *Thematic analysis.* American Psychological Association.

Bridson, R., Houriham, J., and Nordenstam, M. (2007). Curl-Noise for Procedural Fluid Flow. *ACM TOG*, 26(3):46.

Brooke, J. et al. (1996). SUS-A Quick and Dirty Usability Scale. *Usability evaluation in industry*, 189(194):4–7.

Cai, P., Lee, Y., Luo, Y., and Hsu, D. (2020). SUMMIT: A Simulator for Urban Driving in Massive Mixed Traffic. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4023–4029. IEEE.

Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 8(06):679–698.

Castillo, V. S. S., Merino, L., Hecht, G., and Bergel, A. (2021). VR-based User Interactions to Exploit Infinite Space in Programming Activities. In *2021 40th International Conference of the Chilean Computer Science Society (SCCC)*, pages 1–5. IEEE.

Chauvergne, E., Hachet, M., and Prouzeau, A. (2023). Authoring Interactive and Immersive Experiences Using Programming by Demonstration. In *Proceedings of the 34th Conference on l'Interaction Humain-Machine*, pages 1–13.

Chen, X., Neubert, B., Xu, Y.-Q., Deussen, O., and Kang, S. B. (2008). Sketch-based Tree Modeling Using Markov Random Field. In *ACM SIGGRAPH Asia 2008 papers*, pages 1–9.

Cohen, J. (2013). *Statistical Power Analysis for the Behavioral Sciences*. Routledge.

Cooper, S., Dann, W., and Pausch, R. (2000). Alice: a 3-D Tool for Introductory Programming Concepts. *Journal of Computing Sciences in Colleges*, 15(5):107–116.

Côté, S. and Beaulieu, O. (2019). VR Road and Construction Site Safety Conceptual Modeling Based on Hand Gestures. *Frontiers in Robotics and AI*, 6:15.

Crane, K., Llamas, I., and Tariq, S. (2008). Real-Time Simulation and Rendering of 3D Fluids. In Nguyen, H., editor, *GPU Gems 3*, pages 633–675. Addison-Wesley.

Das, K. and Borst, C. W. (2010). An Evaluation of Menu Properties and Pointing Techniques in a Projection-based VR Environment. In *2010 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 47–50. IEEE.

Dashti, S., Prakash, E., Navarro-Newball, A. A., Hussain, F., and Carroll, F. (2022). PotteryVR: Virtual Reality Pottery. *The Visual Computer*, 38(12):4035–4055.

De La Torre, F., Fang, C. M., Huang, H., Banburski-Fahey, A., Amores Fernandez, J., and Lanier, J. (2024). LLMR: Real-time Prompting of Interactive Worlds using Large Language Models. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–22.

De Leon, J. D. O., Tavas, R. P., Aranzanso, R. A., and Atienza, R. O. (2016). Genesys: A Virtual Reality Scene Builder. In *2016 IEEE Region 10 Conference (TENCON)*, pages 3708–3711. IEEE.

Deering, M. F. (1995). HoloSketch: A Virtual Reality Sketching/Animation Tool. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 2(3):220–238.

Delgado, J. M. D., Oyedele, L., Demian, P., and Beach, T. (2020). A Research Agenda for Augmented and Virtual Reality in Architecture, Engineering and Construction. *Advanced Engineering Informatics*, 45:101122.

Derivative (2008). TouchDesigner. Available at: https://derivative.ca, last-visited: 2025-05-21.

Doepgen, S. (2022). Durch Kunstwerke fliegen: Möglich durch virtuelle Realität. Available at: https://brf.be/kultur/kunst/1649949, last-visited: 2025-05-21.

Dosovitskiy, A., Ros, G., Codevilla, F., López, A. M., and Koltun, V. (2017). CARLA: An Open Urban Driving Simulator. In *Proceedings of the Conference on Robotic Learning (CoRL)*.

Dudley, J., Benko, H., Wigdor, D., and Kristensson, P. O. (2019). Performance Envelopes of Virtual Keyboard Text Input Strategies in Virtual Reality. In *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 289–300. IEEE.

Elliott, A., Peiris, B., and Parnin, C. (2015). Virtual Reality in Software Engineering: Affordances, Applications, and Challenges. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 2, pages 547–550. IEEE.

Emilien, A., Vimont, U., Cani, M.-P., Poulin, P., and Benes, B. (2015). Worldbrush: Interactive Example-based Synthesis of Procedural Virtual Worlds. *ACM Trans. Graph.*, 34(4):106–1.

Ens, B., Anderson, F., Grossman, T., Annett, M., Irani, P., and Fitzmaurice, G. (2017). Ivy: Exploring Spatially Situated Visual Programming for Authoring and Understanding Intelligent Environments. In *Proceedings - Graphics Interface*.

Ens, B., Hincapié-Ramos, J. D., and Irani, P. (2014a). Ethereal Planes: a Design Framework for 2D Information Space in 3D Mixed Reality Environments. In *Proceedings of the 2nd ACM symposium on Spatial user interaction*, pages 2–12.

Ens, B. M., Finnegan, R., and Irani, P. P. (2014b). The Personal Cockpit: A Spatial Interface for Effective Task Switching on Head-Worn Displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3171–3180.

Epic Games (2014). Blueprints Visual Scripting. Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/blueprints-visual-scripting-in-unreal-engine, last-visited: 2025-05-21.

Eroglu, S., Gebhardt, S., Schmitz, P., Rausch, D., and Kuhlen, T. W. (2018). Fluid Sketching—Immersive Sketching Based on Fluid Flow. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 475–482. IEEE.

Eroglu, S., Schmitz, P., Martinez, C. A., Rusch, J., Kobbelt, L., and Kuhlen, T. W. (2020). Rilievo: Artistic Scene Authoring via Interactive Height Map Extrusion in VR. *Leonardo*, 53(4):438–441.

Eroglu, S., Schmitz, P., Sinke, K., Anders, D., Kuhlen, T. W., and Weyers, B. (2024a). Choose Your Reference Frame Right: An Immersive Authoring Technique for Creating Reactive Behavior. In *30th ACM Symposium on Virtual Reality Software and Technology*, pages 1–11.

Eroglu, S., Stefan, F., Chevalier, A., Roettger, D., Zielasko, D., Kuhlen, T. W., and Weyers, B. (2021). Design and Evaluation of a Free-Hand VR-based Authoring Environment for Automated Vehicle Testing. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*, pages 1–10. IEEE.

Eroglu, S., Voigt, A., Weyers, B., and Kuhlen, T. W. (2024b). VRScenarioBuilder: Free-Hand Immersive Authoring Tool for Scenario-based Testing of Automated Vehicles. In *2024 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 196–202. IEEE.

Feiner, S., MacIntyre, B., Haupt, M., and Solomon, E. (1993). Windows on the World: 2D Windows for 3D Augmented Reality. In *Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology*, pages 145–155.

Feng, L., Li, Q., Peng, Z., Tan, S., and Zhou, B. (2023). TrafficGen: Learning to Generate Diverse and Realistic Traffic Scenarios. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3567–3575. IEEE.

Fischer, M. H. (2016). Inception: a creative coding environment for virtual reality, in virtual reality. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*, pages 339–340.

Freeman, I. and Salmon, J. L. (2017). A Review of the Capabilities of Current Low-Cost Virtual Reality Technology and Its Potential to Enhance the Design Process. *Journal of Computing and Information Science in Engineering*, 17:031013–1.

Fu, Z., Xu, R., Xin, S., Chen, S., Tu, C., Yang, C., and Lu, L. (2022). EasyVRModeling: Easily Create 3D Models by an Immersive VR System. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 5(1):1–14.

Galin, E., Peytavie, A., Maréchal, N., and Guérin, E. (2010). Procedural Generation of Roads. In *Computer Graphics Forum*, volume 29, pages 429–438. Wiley Online Library.

Galyean, T. A. and Hughes, J. F. (1991). Sculpting: An Interactive Volumetric Modeling Technique. *SIGGRAPH Comput. Graph.*, 25(4):267–274.

Gambi, A., Mueller, M., and Fraser, G. (2019). Automatically Testing Self-driving Cars with Search-based Procedural Content Generation. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 318–328.

Gasch, C., Chover, M., Remolar, I., and Rebollo, C. (2020). Procedural Modelling of Terrains with Constraints. *Multimedia Tools and Applications*, 79:31125–31146.

Gebhardt, S., Pick, S., Leithold, F., Hentschel, B., and Kuhlen, T. (2013). Extended Pie Menus for Immersive Virtual Environments. *IEEE transactions on visualization and computer graphics*, 19(4):644–651.

Genz, F., Fuchs, N., Kolb, D., Müller, S., and Kranzlmüller, D. (2021). Evaluation of Proprietary Social VR Platforms for Use in Distance Learning. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.

Gingold, R. A. and Monaghan, J. J. (1977). Smoothed Particle Hydrodynamics: Theory and Application to Non-spherical Stars. *Monthly Notices of the Royal Astronomical Society*, 181(3):375–389.

Giunchi, D., Numan, N., Gatti, E., and Steed, A. (2024). DreamCodeVR: Towards Democratizing Behavior Design in Virtual Reality with Speech-Driven Programming. In *2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*, pages 579–589. IEEE.

Google (2012). Blockly. Available at: https://developers.google.com/blockly, last-visited: 2025-05-20.

Ha, V., Wallace, J., Ziola, R., and Inkpen, K. (2006). My MDE: Configuring Virtual Workspaces in Multi-display Environments. In *CHI'06 Extended Abstracts on Human Factors in Computing Systems*, pages 1481–1486.

Hansberger, J. T., Peng, C., Mathis, S. L., Areyur Shanthakumar, V., Meacham, S. C., Cao, L., and Blakely, V. R. (2017). Dispelling the Gorilla Arm Syndrome: The Viability of Prolonged Gesture Interactions. In *Virtual, Augmented and Mixed Reality: 9th International Conference, VAMR 2017, Held as Part of HCI International 2017, Vancouver, BC, Canada, July 9-14, 2017, Proceedings 9*, pages 505–520. Springer.

Hart, S. G. (2006). Nasa-Task Load Index (NASA-TLX); 20 Years Later. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. SAGE Publicationssage CA: Los Angeles, CA.

Harvey, B. and Mönig, J. (2010). Bringing "No Ceiling" to Scratch: Can One Language Serve Kids and Computer Scientists? *Proc. Constructionism*, pages 1–10.

Hedlund, M., Jonsson, A., Bogdan, C., Meixner, G., Ekblom Bak, E., and Matviienko, A. (2023). BlocklyVR: Exploring Block-based Programming in Virtual Reality. In *Proceedings of the 22nd International Conference on Mobile and Ubiquitous Multimedia*, pages 257–269.

Hendrikx, M., Meijer, S., Van Der Velden, J., and Iosup, A. (2013). Procedural Content Generation for Games: A Survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(1):1–22.

Hsu, T.-W., Tsai, M.-H., Babu, S. V., Hsu, P.-H., Chang, H.-M., Lin, W.-C., and Chuang, J.-H. (2020). Design and Initial Evaluation of a VR Based Immersive and Interactive Architectural Design Discussion System. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 363–371. IEEE.

Hu, P. J.-H., Ma, P.-C., and Chau, P. Y. (1999). Evaluation of User Interface Designs for Information Retrieval Systems: A Computer-based Experiment. *Decision Support Systems*, 27(1-2):125–143.

Hu, Y., Wang, K., Shao, Y., Plass, J., Wang, Z., and Perlin, K. (2024). Generative Terrain Authoring with Mid-air Hand Sketching in Virtual Reality. In *Proceedings of the 30th ACM Symposium on Virtual Reality Software and Technology*, pages 1–10.

Ichikawa, S., Takashima, K., Tang, A., and Kitamura, Y. (2018). VR Safari Park: A Concept-based World Building Interface using Blocks and World Tree. In *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*, pages 1–5.

Ijiri, T., Owada, S., and Igarashi, T. (2006). Seamless Integration of Initial Sketching and Subsequent Detail Editing in Flower Modeling. In *Computer Graphics Forum*, volume 25, pages 617–624. Wiley Online Library.

Intuition, A. (1990). Mechanical Simulation. Available at: https://www.carsim.com, last-visited: 2024-09-09.

IPG-Automotive (1999). CarMaker. Available at: https://ipg-automotive.com/en/products-solutions/software/carmaker, last-visited: 2025-05-20.

IPGAutomotive (1999). Solutions for Virtual Test Driving. Available at: https://ipg-automotive.com, last-visited: 2024-09-09.

Jackson, B. and Keefe, D. F. (2016). 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.

Jerald, J., Mlyniec, P., Yoganandan, A., Rubin, A., Paullus, D., and Solotko, S. (2013). MakeVR: A 3D World-Building Interface. In *2013 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 197–198. IEEE.

Jin, Q., Liu, Y., Yuan, Y., Yarosh, L., and Rosenberg, E. S. (2020). VWorld: An Immersive VR System for Learning Programming. In *Proceedings of the 2020 ACM Interaction Design and Children Conference: Extended Abstracts*, pages 235–240.

Johnston, W. M., Hanna, J. P., and Millar, R. J. (2004). Advances in Dataflow Programming Languages. *ACM Computing Surveys (CSUR)*, 36(1):1–34.

Kao, D., Mousas, C., Magana, A. J., Harrell, D. F., Ratan, R., Melcer, E. F., Sherrick, B., Parsons, P., and Gusev, D. A. (2020). Hack.VR: A Programming Game in Virtual Reality.

Keefe, D., Zeleznik, R., and Laidlaw, D. (2007). Drawing on Air: Input Techniques for Controlled 3D Line Illustration. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):1067–1081.

Keefe, D. F., Feliz, D. A., Moscovich, T., Laidlaw, D. H., and LaViola Jr, J. J. (2001). CavePainting: A Fully immersive 3D Artistic Medium and Interactive Experience. In *Proc. of the 2001 Symp. on Interactive 3D Graphics*, pages 85–93. ACM.

Kemeny, A. and Panerai, F. (2003). Evaluating Perception in Driving Simulation Experiments. *Trends in Cognitive Sciences*, 7(1):31–37.

Kennedy, R. S., Lane, N. E., Berbaum, K. S., and Lilienthal, M. G. (1993). Simulator Sickness Questionnaire: An Enhanced Method for Quantifying Simulator Sickness. *The International Journal of Aviation Psychology*, 3(3):203–220.

Knierim, P., Schwind, V., Feit, A. M., Nieuwenhuizen, F., and Henze, N. (2018). Physical Keyboards in Virtual Reality: Analysis of Typing Performance and Effects of Avatar Hands. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–9.

Kolmogorov, A. N. (1940). Wienersche Spiralen Und Einige Andere Interessante Kurven Im Hilbertschen Raum. *Acad. Sci. URSS*, 26(2):115–118.

Kottlowski, L. (2019). From 2D to 3D to VR in Less than a Year. Presented at SIGGRAPH 2019. Available at: https://www.youtube.com/watch?v=ZgsdHOc9a4s, last-visited: 2025-05-21.

Krajzewicz, D., Erdmann, J., Behrisch, M., and Bieker, L. (2012). Recent Development and Applications of SUMO-Simulation of Urban MObility. *International Journal on Advances in Systems and Measurements*, 5(3&4).

Krüger, M., Gerrits, T., Römer, T., Kuhlen, T., and Weissker, T. (2024). IntenSelect+: Enhancing Score-Based Selection in Virtual Reality. *IEEE Transactions on Visualization and Computer Graphics*.

Latta, L. (2004). Building a Million Particle System. In *Game Developers Conference*.

Laugwitz, B., Held, T., and Schrepp, M. (2008). Construction and Evaluation of a User Experience Questionnaire. In *HCI and Usability for Education and Work: 4th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society, USAB 2008, Graz, Austria, November 20-21, 2008. Proceedings 4*, pages 63–76. Springer.

LaViola Jr, J. J., Kruijff, E., McMahan, R. P., Bowman, D., and Poupyrev, I. P. (2017). *3D User Interfaces: Theory and Practice*. Addison-Wesley Professional.

Lediaeva, I. and LaViola, J. (2020). Evaluation of Body-referenced Graphical Menus in Virtual Environments. In *Graphics Interface 2020*.

Lee, G. A. (2009). *Immersive Authoring of Virtual Worlds*. PhD Thesis, Pohang University of Science and Technology. Available at: https://oasis.postech.ac.kr/handle/2014.oak/8835, last-visited: 2025-05-21.

Lee, G. A., Kim, G. J., and Billinghurst, M. (2005). Immersive Authoring: What You Experience Is What You Get (WYXIWYG). *Communications of the ACM*, 48(7):76–81.

Lee, G. A., Nelles, C., Billinghurst, M., and Kim, G. J. (2004). Immersive Authoring of Tangible Augmented Reality Applications. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 172–181. IEEE.

Li, A., Chen, S., Sun, L., Zheng, N., Tomizuka, M., and Zhan, W. (2022). SceGene: Bio-Inspired Traffic Scenario Generation for Autonomous Driving Testing. *IEEE Transactions on Intelligent Transportation Systems*, 23(9).

LIANG, J. and GREEN, M. (1994). JDCAD: A Highly Interactive 3D Modeling System. *Computers & graphics*, 18(4):499–506.

Lindlbauer, D., Feit, A. M., and Hilliges, O. (2019). Context-Aware Online Adaptation of Mixed Reality Interfaces. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, pages 147–160. Association for Computing Machinery.

Liu, Z., Zhang, F., and Cheng, Z. (2021). BuildingSketch: Freehand Mid-air Sketching for Building Modeling. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 329–338. IEEE.

Longay, S., Runions, A., Boudon, F., and Prusinkiewicz, P. (2012). TreeSketch: Interactive Procedural Modeling of Trees on a Tablet. In *SBIM@ Expressive*, pages 107–120. Citeseer.

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and WieBner, E. (2018). Microscopic Traffic Simulation using SUMO. In *2018 IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582. IEEE.

Lou, G., Deng, Y., Zheng, X., Zhang, M., and Zhang, T. (2022). Testing of Autonomous Driving Systems: Where Are We and Where Should We Go? In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 31–43.

Lucy, L. B. (1977). A Numerical Approach to the Testing of the Fission Hypothesis. *The Astronomical Journal*, 82:1013–1024.

Luo, S., Teather, R. J., and McArthur, V. (2020). Camera-Based Selection with Cardboard Head-Mounted Displays. In *International Conference on Human-Computer Interaction*, pages 383–402. Springer.

Machuca, M. D. B., Asente, P., Stuerzlinger, W., Lu, J., and Kim, B. (2018). Multiplanes: Assisted Freehand VR Sketching. In *Proceedings of the 2018 ACM Symposium on Spatial User Interaction*, pages 36–47.

Machuca, M. D. B., Israel, J. H., Keefe, D. F., and Stuerzlinger, W. (2023). Toward More Comprehensive Evaluations of 3D Immersive Sketching, Drawing, and Painting. *IEEE Transactions on Visualization and Computer Graphics*.

Mandelbrot, B. B. and Van Ness, J. W. (1968). Fractional Brownian Motions, Fractional Noises and Applications. *SIAM Review*, 10(4):422–437.

Mao, R. Q., Lan, L., Kay, J., Lohre, R., Ayeni, O. R., Goel, D. P., et al. (2021). Immersive Virtual Reality for Surgical Training: A Systematic Review. *Journal of Surgical Research*, 268:40–58.

Mapes, D. P. and Moshell, J. M. (1995). A Two-handed Interface for Object Manipulation in Virtual Environments. *Presence: Teleoperators & Virtual Environments*, 4(4):403–416.

Maurer-Mathison, D. V. (1999). *The Ultimate Marbling Handbook: A Guide to Basic and Advanced Techniques for Marbling Paper and Fabric*. Watson-Guptill.

Mendes, D., Medeiros, D., Sousa, M., Ferreira, R., Raposo, A., Ferreira, A., and Jorge, J. (2017). Mid-air Modeling with Boolean Operations in VR. In *2017 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 154–157. IEEE.

Meta (2021). Horizon Worlds. Available at: https://horizon.meta.com, last-visited: 2025-05-21.

Mine, M. (1995). ISAAC: A Virtual Environment Tool for the Interactive Construction of Virtual Worlds. *UNC Chapel Hill Computer Science Technical Report TR95-020*.

Mine, M., Yoganandan, A., and Coffey, D. (2014a). Making VR work: Building a real-world immersive modeling application in the virtual world. In *SUI 2014 - Proceedings of the 2nd ACM Symposium on Spatial User Interaction*.

Mine, M., Yoganandan, A., and Coffey, D. (2014b). 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.

Mujber, T. S., Szecsi, T., and Hashmi, M. S. (2004). Virtual Reality Applications in Manufacturing Process Simulation. *Journal of Materials Processing Technology*, 155:1834–1838.

Müller, P., Wonka, P., Haegler, S., Ulmer, A., and Van Gool, L. (2006). Procedural Modeling of Buildings. In *ACM SIGGRAPH 2006 Papers*, pages 614–623. ACM.

Murray, J. T. (2022). RealityFlow: Open-Source Multi-User Immersive Authoring. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 65–68. IEEE.

National Instruments (1986). LabVIEW. Available at: https://www.ni.com/labview, last-visited: 2025-05-21.

Nebeling, M., Lewis, K., Chang, Y. C., Zhu, L., Chung, M., Wang, P., and Nebeling, J. (2020). XRDirector: A Role-Based Collaborative Immersive Authoring System. In *Conference on Human Factors in Computing Systems - Proceedings*.

Norman, D. A. (2002). *The Design of Everyday Things*. Basic Books, Inc., USA.

NVIDIA (2018). NVIDIA Introduces DRIVE Constellation Simulation System to Safely Drive Autonomous Vehicles Billions of Miles in Virtual Reality. Available at: https://nvidianews.nvidia.com, last-visited: 2025-05-15.

Okabe, M., Owada, S., and Igarashi, T. (2006). Interactive Design of Botanical Trees Using Freehand Sketches and Example-based Editing. In *ACM SIGGRAPH 2006 Courses*, pages 18–es. ACM.

OpenDILab (2021). DI-drive: OpenDILab Decision Intelligence platform for Autonomous Driving simulation. Available at: https://github.com/opendilab/DI-drive, last-visited: 2025-05-19.

OpenMSL (2023). ALKS Scenario Interpretation in OpenSCENARIO. Available at: https://github.com/openMSL/sl-3-1-osc-alks-scenarios, last-visited: 2025-05-19.

OpenSCENARIO, A. (2020). ASAM OpenSCENARIO 1.0.0 User Guide. Available at: https://releases.asam.net/OpenSCENARIO/1.0.0/ASAM_OpenSCENARIO_BS-1-2_User-Guide_V1-0-0.html, last-visited: 2025-05-20.

Paes, D., Arantes, E., and Irizarry, J. (2017). Immersive Environment for Improving the Understanding of Architectural 3D Models: Comparing User Spatial Perception Between Immersive and Traditional Virtual Reality Systems. *Automation in Construction*, 84:292–303.

Pane, J. F. and Myers, B. A. (1996). *Usability Issues in the Design of Novice Programming Systems*. Carnegie-Mellon University. Department of Computer Science.

Paranjape, I., Jawad, A., Xu, Y., Song, A., and Whitehead, J. (2020). A Modular Architecture for Procedural Generation of Towns, Intersections and Scenarios for Testing Autonomous Vehicles. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 162–168. IEEE.

Pedram, S., Palmisano, S., Skarbez, R., Perez, P., and Farrelly, M. (2020). Investigating the Process of Mine Rescuers' Safety Training with Immersive Virtual Reality: A Structural Equation Modelling Approach. *Computers & Education*, 153:103891.

Perlin, K. (1985). An Image Synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296.

Perlin, K. (2001). Noise Hardware. *Real-Time Shading SIGGRAPH Course Notes*.

Perlin, K. and Neyret, F. (2001). Flow Noise. In *28th Int. Conf. on Computer Graphics and Interactive Techniques*, page 187. SIGGRAPH.

Pick, S., Hentschel, B., Tedjo-Palczynski, I., Wolter, M., and Kuhlen, T. W. (2010). Automated Positioning of Annotations in Immersive Virtual Environments. In *Proceedings of the 16th Eurographics Conference on Virtual Environments & Second Joint Virtual Reality*, pages 1–8.

Polys, N. F., Kim, S., and Bowman, D. A. (2005). Effects of Information Layout, Screen Size, and Field of View on User Performance in Information-rich Virtual Environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pages 46–55.

Ponto, K., Tredinnick, R., Bartholomew, A., Roy, C., Szafir, D., Greenheck, D., and Kohlmann, J. (2013). SculptUp: A Rapid, Immersive 3D Modeling Environment. In *2013 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 199–200.

Ragan, E. D., Bowman, D. A., Kopper, R., Stinson, C., Scerbo, S., and McMahan, R. P. (2015). Effects of Field of View and Visual Complexity on Virtual Reality Training Effectiveness for a Visual Scanning Task. *IEEE Transactions on Visualization and Computer Graphics*, 21(7).

Rausch, D. and Assenmacher, I. (2008). A Sketch-Based Interface for Architectural Modification in Virtual Environments. In *5. Workshop der GI-Fachgruppe VR/AR*.

Rausch, D., Assenmacher, I., and Kuhlen, T. (2010). 3D Sketch Recognition for Interaction in Virtual Environments. In Erleben, K., Bender, J., and Teschner, M., editors, *VRIPHYS*. The Eurographics Association.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., et al. (2009). Scratch: Programming for All. *Communications of the ACM*, 52(11):60–67.

Richardson, L. F. (1922). Weather Prediction By Numerical Process Cambridge University Press. *Cambridge Richardson Weather Prediction by Numerical Process 1922*.

Riegler, A., Riener, A., and Holzmann, C. (2021). A systematic review of virtual reality applications for automated driving: 2009–2020. *Frontiers in Human Dynamics*, 3:689856.

Robinett, W. and Holloway, R. (1992). Implementation of Flying, Scaling and Grabbing in Virtual Worlds. In *Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 189–192.

Rocha, J. B. and Prada, R. (2025). Procedural Content Generation for Cooperative Games-A Systematic Review. *IEEE Transactions on Games*.

Rong, G., Shin, B. H., Tabatabaee, H., Lu, Q., Lemke, S., Možeiko, M., Boise, E., Uhm, G., Gerow, M., Mehta, S., et al. (2020). LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving. *arXiv preprint arXiv:2005.03778*.

Rosales, E., Araújo, C., Rodriguez, J., Vining, N., Yoon, D., and Sheffer, A. (2021). AdaptiBrush: Adaptive General and Predictable VR Ribbon Brush. *ACM Trans. Graph.*, 40(6):247–1.

Rosales, E., Rodriguez, J., and Sheffer, A. (2019). Surfacebrush: From Virtual Reality Drawings to Manifold Surfaces. *ACM Transactions on Graphics*.

Rosenholtz, R., Li, Y., and Nakano, L. (2007). Measuring Visual Clutter. *Journal of Vision*, 7(2):17–17.

Rother, C., Kolmogorov, V., and Blake, A. (2004). "GrabCut" Interactive Foreground Extraction using Iterated Graph Cuts. *ACM Transactions on Graphics (TOG)*, 23(3):309–314.

Sachs, E., Roberts, A., and Stoops, D. (1991). 3-Draw: A Tool for Designing 3D Shapes. *IEEE Comput. Graph. Appl.*, 11(6):18–26.

Satriadi, K. A., Ens, B., Cordeil, M., Czauderna, T., and Jenny, B. (2020). Maps Around Me: 3D Multiview Layouts in Immersive Spaces. *Proceedings of the ACM on Human-Computer Interaction*, 4(ISS):1–20.

Sayyad, E., Sen, P., and Höllerer, T. (2017). PanoTrace: Interactive 3D Modeling of Surround-view Panoramic Images in Virtual Reality. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*, pages 1–10.

Schlünsen, R., Ariza, O., and Steinicke, F. (2019). A VR Study on Freehand vs. Widgets for 3D Manipulation Tasks. In *Proceedings of Mensch Und Computer 2019*, pages 223–233. ACM.

Schwarz, M. and Müller, P. (2015). Advanced Procedural Modeling of Architecture. *ACM Transactions on Graphics (TOG)*, 34(4):1–12.

Segura, R. J., del Pino, F. J., Ogáyar, C. J., and Rueda, A. J. (2020). VR-OCKS: A Virtual Reality Game for Learning the Basic Concepts of Programming. *Computer Applications in Engineering Education*.

Siemens (2019). Siemens Introduces Revolutionary New Validation Program to Accelerate Autonomous Vehicle Development. Available at: https://newsroom.sw.siemens.com/en-US/pave360-media-alert, last-visited: 2025-05-20.

Smelik, R. M., Tutenel, T., de Kraker, K. J., and Bidarra, R. (2011). A Declarative Approach to Procedural Modeling of Virtual Worlds. *Computers & Graphics*, 35(2):352–363.

Solirax (2018a). Logix. Available at: https://wiki.neosvr.com/LogiX, last-visited: 2025-05-21.

Solirax (2018b). Neos Wiki. Available at: https://wiki.neosvr.com, last-visited: 2025-05-21.

Sousa, T. B. (2012). Dataflow Programming Concept, Languages and Applications. In *Doctoral Symposium on Informatics Engineering*, volume 130.

Stam, J. (1999). Stable Fluids. In *Proc. of the 26th Annual Conf. on Computer Graphics and Interactive Techniques*, pages 121–128. ACM Press/Addison-Wesley Publishing Co.

Stam, J. (2003). Real-time Fluid Dynamics for Games. In *Proc. of the Game Developer Conference*, volume 18, page 25.

Steed, A. (2006). Towards a General Model for Selection in Virtual Environments. In *3D User Interfaces (3DUI'06)*, pages 103–110. IEEE.

Steed, A., Ortega, F. R., Williams, A. S., Kruijff, E., Stuerzlinger, W., Batmaz, A. U., Won, A. S., Rosenberg, E. S., Simeone, A. L., and Hayes, A. (2020). Evaluating Immersive Experiences During Covid-19 and Beyond. *Interactions*, 27(4):62–67.

Steed, A. and Slater, M. (1996). Dataflow Representation for Defining Behaviours Within Virtual Environments. In *Proceedings - Virtual Reality Annual International Symposium*.

Steuer, J., Biocca, F., Levy, M. R., et al. (1995). Defining Virtual Reality: Dimensions Determining Telepresence. *Communication in the Age of Virtual Reality*, 33:37–39.

Stoakley, R., Conway, M. J., and Pausch, R. (1995). Virtual Reality on a WIM: Interactive Worlds in Miniature. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 265–272.

Sun, N., Feng, A., Patton, R., Gingold, Y., and Lages, W. (2021). Programmable Virtual Reality Environments. In *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 619–620. IEEE.

Sun, Q., Huang, X., Williams, B. C., and Zhao, H. (2022). InterSim: Interactive Traffic Simulation via Explicit Relation Modeling. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11416–11423. IEEE.

Sweller, J., Van Merrienboer, J. J., and Paas, F. G. (1998). Cognitive Architecture and Instructional Design. *Educational Psychology Review*, 10:251–296.

Takala, T. M., Mäkäräinen, M., and Hämäläinen, P. (2013). Immersive 3D modeling with Blender and off-the-shelf hardware. In *2013 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 191–192.

Tan, S., Wong, K., Wang, S., Manivasagam, S., Ren, M., and Urtasun, R. (2021). SceneGen: Learning to Generate Realistic Traffic Scenes. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 892–901. IEEE.

Tatzgern, M., Kalkofen, D., Grasset, R., and Schmalstieg, D. (2014). Hedgehog Labeling: View Management Techniques for External Labels in 3D Space. In *2014 IEEE Virtual Reality (VR)*, pages 27–32. IEEE.

Teather, R. J. and Stuerzlinger, W. (2013). Pointing at 3D Target Projections with One-Eyed and Stereo Cursors. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 159–168.

TiltBrush (2016). Google Tilt Brush. Discontinued in 2021; source code archived on GitHub: https://github.com/googlevr/tilt-brush, last-visited: 2025-05-15.

Trueba, R., Andujar, C., and Argelaguet, F. (2009). Multi-scale Manipulation in Indoor Scenes with the World in Miniature Metaphor. In *Proceedings of the 15th Joint Virtual Reality Eurographics Conference on Virtual Environments*, pages 93–100.

Tversky, B. (2013). Visualizing Thought. In *Handbook of Human Centric Visualization*, pages 3–40. Springer.

Unity Technologies (2021). Unity Visual Scripting. Available at: https://unity.com/features/unity-visual-scripting, last-visited: 2025-05-21.

Vacondio, R., Altomare, C., De Leffe, M., Hu, X., Le Touzé, D., Lind, S., Marongiu, J.-C., Marrone, S., Rogers, B. D., and Souto-Iglesias, A. (2021). Grand challenges for Smoothed Particle Hydrodynamics numerical schemes. *Computational Particle Mechanics*, 8(3):575–588.

Van Doremalen, N., Bushmaker, T., Morris, D. H., Holbrook, M. G., Gamble, A., Williamson, B. N., Tamin, A., Harcourt, J. L., Thornburg, N. J., Gerber, S. I., et al. (2020). Aerosol and Surface Stability of SARS-CoV-2 as Compared with SARS-CoV-1. *New England Journal of Medicine*, 382(16):1564–1567.

Vector (2023). ADAS Testing with Virtual Test Drives. Available at: https://www.vector.com/dyna4, last-visited: 2025-05-20.

Vincur, J., Konopka, M., Tvarozek, J., Hoang, M., and Navrat, P. (2017). Cubely: Virtual Reality Block-based Programming Environment. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST*.

Wachenfeld, W. and Winner, H. (2016). The Release of Autonomous Vehicles. In *Autonomous driving*, pages 425–449. Springer.

Wang, J., Leach, O., and Lindeman, R. W. (2013). DIY World Builder: An Immersive Level-Editing System. In *2013 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 195–196. IEEE.

Weintrop, D. and Wilensky, U. (2015). To Block Or Not to Block, That Is the Question: Students' Perceptions of Blocks-based Programming. In *Proceedings of the 14th International Conference on Interaction Design and Children*, pages 199–208.

Wesche, G. and Seidel, H.-P. (2001). FreeDrawer: A Free-form Sketching System on the Responsive Workbench. In *Proc. of the ACM VRST*, pages 167–174. ACM.

Wickens, C. D. (1981). Processing Resources in Attention, Dual Task Performance, and Workload Assessment.

Wolber, D., Abelson, H., Spertus, E., and Looney, L. (2011). *App Inventor*. O'Reilly Media, Inc.

Wu, D., Yang, M., Liu, Z., Tu, F., Liu, F., and Cheng, Z. (2024). VRTree: Example-Based 3D Interactive Tree Modeling in Virtual Reality. In *Computer Graphics Forum*, page e15254. Wiley Online Library.

Yu, E., Arora, R., Stanko, T., Bærentzen, J. A., Singh, K., and Bousseau, A. (2021a). Cassie: Curve and Surface Sketching in Immersive Environments. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–14.

Yu, X., DiVerdi, S., Sharma, A., and Gingold, Y. (2021b). ScaffoldSketch: Accurate Industrial Design Drawing in VR. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 372–384.

Zhang, F., Liu, Z., Cheng, Z., Deussen, O., Chen, B., and Wang, Y. (2021). Mid-air Finger Sketching for Tree Modeling. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*, pages 826–834. IEEE.

Zhang, L., Bowman, D. A., and Jones, C. N. (2019). Exploring Effects of Interactivity on Learning with Interactive Storytelling in Immersive Virtual Reality. In *2019 11th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*, pages 1–8. IEEE.

Zhang, L. and Oney, S. (2019). 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.

Zhang, L. and Oney, S. (2020). 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.

Zhang, L., Pan, J., Gettig, J., Oney, S., and Guo, A. (2024). VRCopilot: Authoring 3D Layouts with Generative AI Models in VR. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–13.

Zhou, Y., Sun, Y., Tang, Y., Chen, Y., Sun, J., Poskitt, C. M., Liu, Y., and Yang, Z. (2023). Specification-based Autonomous Driving System Testing. *IEEE Transactions on Software Engineering*.

Zhu, X. and Yang, Y. (2024). Interactive Mesh Sculpting with Arbitrary Topologies in Head-Mounted VR Environments. *Mathematics*, 12(15):2428.

Zhu, Z., Liu, Z., Zhang, Y., Zhu, L., Huang, J., Villanueva, A. M., Qian, X., Peppler, K., and Ramani, K. (2023). 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.

Zielasko, D., Freitag, S., Rausch, D., Law, Y. C., Weyers, B., and Kuhlen, T. W. (2015). BlowClick: A Non-Verbal Vocal Input Metaphor for Clicking. In *Proceedings of the 3rd ACM Symposium on Spatial User Interaction*, pages 20–23. ACM.

Zielasko, D., Neha, N., Weyers, B., and Kuhlen, T. W. (2017). BlowClick 2.0: A Trigger Based on Non-verbal Vocal Input. In *2017 IEEE Virtual Reality (VR)*, pages 319–320. IEEE.

Zielasko, D. and Riecke, B. E. (2020). Sitting vs. Standing in VR: Towards a Systematic Classification of Challenges and (Dis) Advantages. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 297–298. IEEE Computer Society.