Authoring Interactive and Immersive Experiences Using Programming by Demonstration

Edwige Chauvergne, Martin Hachet, Arnaud Prouzeau

Authoring Interactive and Immersive Experiences Using Programming by Demonstration

Editer des expériences interactives et immersives en utilisant la programmation par démonstration

EDWIGE CHAUVERGNE, Inria, Université de Bordeaux, CNRS, France
MARTIN HACHET, Inria, Université de Bordeaux, CNRS, France
ARNAUD PROUZEAU, Inria, Université de Bordeaux, CNRS, France

Immersive technologies, like virtual and augmented reality, allow engaging the general public in interactive experiences, which are particularly promising in educational and cultural activities (e.g. museums, exhibits). For now, the design of such interactive experiences requires extensive knowledge of expert programming tools, and thus they are not accessible to non-developers. A promising method to ease the prototyping of interactive scenes for non-expert users stands on the concept of programming-by-demonstration. With such an approach, novice users can simply demonstrate an interaction or the expected behavior of a virtual object to build their prototype, without knowing the underlying coding mechanisms. In this paper, we propose an immersive authoring system that bases on that approach. We prototyped two use cases based on this system, as well as a user study. Using our observations and the results of the study, we discuss challenges associated with the design of such systems and provide guidelines for the development of future immersive programming-by-demonstration tools.

CCS Concepts: • **Human-centered computing** → **Virtual reality**; • **Software and its engineering** → **Programming by example**; **Software prototyping**.

Additional Key Words and Phrases: Virtual reality, immersive, authoring interactions, demonstration, direct manipulation, end users, challenges

Les technologies immersives, comme la réalité virtuelle et augmentée, permettent de proposer au grand public des expériences interactives, particulièrement dans les activités éducatives et culturelles. Pour l'instant, la conception de telles expériences nécessite une connaissance approfondie de langages de programmation, difficilement accessibles aux non-développeurs. La programmation par démonstration est une approche qui permet de faciliter le prototypage de scènes interactives pour les utilisateurs non experts. Avec cette approche, les utilisateurs peuvent construire un prototype sans connaître les mécanismes de codages sous-jacents, simplement en effectuant l'interaction ou le comportement attendu d'un objet virtuel. Dans cet article, nous proposons un système de création immersif basé sur cette approche. Nous avons prototypé deux cas d'utilisations, ainsi que réalisé une étude utilisateur. En s'appuyant sur nos observations et les résultats de l'étude, nous discutons des défis et des lignes directrices pour le développement de futurs outils de programmation immersive par démonstration.

Mots-clés additionnels : Réalité Virtuelle, immersif, édition d'interactions, démonstration, manipulation directe, utilisateurs finaux, défis

ACM Reference Format:
Edwige Chauvergne, Martin Hachet, and Arnaud Prouzeau. 2023. Authoring Interactive and Immersive Experiences Using Programming by Demonstration. In *IHM '23: Proceedings of the 34th Conference on l'Interaction Humain-Machine (IHM '23), April 3–7, 2023, TROYES, France.* ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3583961.3583981

IHM '23, April 3–7, 2023, TROYES, France

ACM ISBN 978-1-4503-9824-4/23/04…$15.00
https://doi.org/10.1145/3583961.3583981

1 INTRODUCTION

The use of immersive technologies in educational and cultural contexts is rising [8, 36]. Thanks to virtual reality, users can live experiences that would be too far in space or time, or too abstract to be experienced in reality. Such technology also supports active learning and engagement [12].

Educators and cultural mediators benefit from their expertise in their own domain to design relevant VR experiences. In 2019, the Louvre's curators team and VR developers collaborated to create "Mona Lisa: Beyond the Glass"[1], an immersive and interactive scene in which the Louvre's visitors get the opportunity to closer analyze the painting, explore it through different angles and vision techniques. Strengthen by the cultural and artistic knowledge of curators and the coding skills of developers, they proposed a unique experience by giving life to the painting.

To create such immersive VR experiences, non-technical experts need to rely on developers in the early stages of prototyping. This dependency can hinder the design process and slows the emergence of new projects [2]. Enabling VR authoring to non-developers is therefore an opportunity to support the expansion of VR. Although many tools are available for static immersive prototyping and animation editing [18], authoring tools for interactive experiences are not yet available to end-users.

Interactive prototypes help designers envision the type of interactions that would work in a specific context, and they support the discovery of design flaws through unexpected behaviors from users. However, to author such interactive prototypes without code is still an open research question. Various methods have been investigated such as simplified desktop interfaces [41], visual programming [42], and authoring by demonstration [3, 29, 39].

Immersive authoring by demonstration appears as a promising way of creation as users directly visualize the outcome of their actions and create the interaction with the first-person point of view inside the context. However, it raises several design challenges that have not been studied yet. Such investigation would help in the design of more adapted and more accessible authoring tools.

In this paper, we propose a tool for the interactive creation of immersive experiences to investigate the challenges in this domain. Building upon two interactive experiences authored using our tool, and a user study, we introduce five challenges and future research questions for immersive VR authoring by demonstration. The goal of this paper is not to compare and promote the use of programming by demonstration over other methods such as visual programming, but rather explore its use and identify its challenges in both the conception and use stages. We hope this paper will serve as a basis for future designers in the creation of VR authoring tools by demonstration, as well as a groundwork for future research in the domain.

2 RELATED WORK

In their study, Ashtari et al. [2] interviewed end-user developers to tackle the current practices and challenges of AR/VR authoring. They noticed that many non-developer AR/VR creators use complex authoring tools such as Unity[2] and therefore face many challenges including searching for resources, debugging, and testing. Nebeling et al. [28] studied augmented and virtual reality authoring tools, and they identified a gap in high-fidelity tools requiring a low level of technical skills. Enabling programming to non-developers is a common concern in research and several methods have been explored to make it more graphical and less abstract.

2.1 Authoring Immersive Application

To allow end-users to participate in the authoring of immersive applications throughout the design phase is essential to identify design flaws in an early stage, get feedback for an iterative design process and improve the design before reaching the costly implementation stage. A common basic prototyping technique is paper

[1] https://store.steampowered.com/app/1172310/Mona_Lisa_Beyond_The_Glass/?l=french
[2] https://unity.com/

prototyping. This technique is fast and efficient for 2D interactions. Yet, as AR and VR evolve in a three-dimensional space and are usually dynamic, conveying ideas through paper is more challenging. With 360Proto [26], Nebeling and Madier generate a 360° AR/VR prototype from paper which gives a more realistic preview of the final experience.

To help users get a better understanding of the final product, it is possible to use videos that are more dynamic or technologies closer to the final product. Burns et al. [6] suggest combining informance design and video prototyping. This method allowed them to identify design flaws during the prototype creation and get feedback from a large audience as they shared the video. Researchers also focused on solutions that rely on technologies similar to the final device, and explore the use of real AR and VR for prototyping. DART [23] is a desktop AR prototyping tool that allows 3D dynamic storyboarding. With VREUD [41], Yigitbas et al. proposed a desktop authoring interface with a low entry barrier. Using VREUD users can edit the scene, event-based interactions, and tasks. Such a solution benefits from the accuracy of the desktop, yet, it forces users to code out of context and to go back and forth between the desktop and the HMD while testing and debugging. With ProtoAR [27], Nebeling et al. facilitate the creation of virtual props using tangible Play-Doh props. Similarly, SpatialProto [25] enables immersive spatial prototyping by recording animation using physical props.

2.2 Authoring in Immersive Environments

Oulasvirta et al. [30] discuss the benefits of designing in the original context or an environment similar to it. Such practice helps designers better understand the context and be aware of it during the design phase. As designers do not rely on their memory, their mental model of the context is more reliable. With Reality Editor [15], users can edit the behavior of smart tangible objects in AR. MARVist [10] and DXR [33] allow their users to edit visualization directly in the context it will be experienced which facilitates previewing the final result. Ens et al. implemented Ivy [11] a VR spatially situated visual programming tool for understanding data flows and connections between smart objects. As the original context is not always accessible or available, the authoring can also occur in a reproduction of the original context. It allows in-context remote authoring. Using CAVE-AR [9], users can simulate and debug AR experiences from a VR CAVE. In DistanciAR [40] and Corsican Twin [31] users author the AR experience in a virtual scan of the place. Thus, authoring in context supports the users' awareness of the final environment during the authoring process. Beyond being in context, in this paper we investigate how the user can interact with this context to author interactions.

In immersive environments, users can more naturally interact with virtual objects thanks to direct manipulation. Lee et al. introduce this concept as WYXIWYG (*What You eXperience Is What You Get*) [20] which describes the benefits of immersive authoring. With VR Safari Park [16], Ichikawa et al. illustrate the ease with which users can edit a virtual environment using direct manipulation. Suzuki et al. introduce RealitySketch [35], an authoring tool for AR interactive graphics and visualizations. Interactions can be created by manipulating the virtual objects and directly linking triggers and actions attached to these objects as demonstrated by Vargas González et al. [37]. LevelEd VR [3] is a VR level editor in which users can create and edit virtual objects and visually script the game conditions. VRFromX [17] and Ng et al.'s system [29] enable the edition of virtual objects and making them interactive by creating and connecting triggers and actions.

Most of the examples previously mentioned allow users to author using direct manipulation, however, it only allows, most of the time, to design applications with simple interactions. To prototype more complex interactions, some researchers proposed to use *Visual Scripting*, a programming method allowing its users to code by manipulating and assembling graphical programming elements instead of textual coding. It is used in FlowMatic [42], an immersed visual scripting tool. It allows its users to directly script interaction in the virtual environment and to benefit from authoring in context. Another method, more accessible to non-programmers, is to use demonstration to define an interaction.

2.3 Authoring by demonstration

Programming by demonstration has been developed to allow end-users to easily specify an expected behavior to the machine (robot or computer) by simply demonstrating it. Programming by demonstration differs from other programming methods during the authoring phase rather than during its visualization. While programming by demonstration users directly manipulate objects, they need less fundamental knowledge and do not have to learn and use the textual primitives that are usually used with classical visual programming, which decreases the accessibility level. However, its visualization can either be textual with the demonstration being translated to code, or it can be more visual. In that case, after the authoring phase, the results of programming by demonstration can be considered as visual programming.

Saket et al.'s system [32] analyses the users' actions to extract their intent and deduce the expected behavior. The analysis can either rely on one example of the event like AREDA [4], an authoring tool for AR assembly instructions, or on several examples of the expected behavior. Relying on several demonstrations helps the system generalize a behavior and manage individual variations. Gesture Studio [22] as well as Hartmann et al.'s system [14] rely on multiple examples to generalize an interaction.

The rising availability of VR and AR equipment, as well as the increasing power of such devices, has allowed the exploration of VR/AR authoring by demonstration. Immersive VR demonstration has been explored for intuitive animation editing. In their paper, Arora et al. [1] study the use of mid-air hand gestures for the authoring of complex animations like particle behavior. Tvori[3] is a commercialized tool that allows VR film-making through demonstration. Rapido [21] uses a similar combination of demonstration and interfaces comparable to video editing interfaces to turn video prototypes of AR experience into executable state machines that simulate interactions on tablet.

Demonstration is not only meant to specify the users' behavior but also the elements they interact with. In GhostAR [7], a human-robot interaction authoring tool, users not only demonstrate the users' actions but also the robot's reaction. Lécuyer et al. [19] explore the use of demonstration for authoring a scenario-based training experience. The scenario is demonstrated directly in VR and is then edited on a desktop interface. With AffordIt! [24], Masnadi et al. fully immersed the authoring of affordances. Users directly cut into the virtual meshes and manipulate them to specify the affordances, for instance, users can cut a circle into the mesh of a washing machine to make a door that can be opened. With GesturAR [39], Wang et al. introduce a freehand interaction authoring tool. Users can create either dynamic or static interactions and behaviors that can also be either discrete or continuous. However, the system only allows linking triggers and actions and does not support more complex authoring such as chains or delays.

Through this state of the art, we notice that authoring by demonstration has been explored before, yet the expressivity and power of the tools remain limited. They do not enable the authoring of complex interactions. The studies mentioned above focus on specific issues such as the use of hand gestures for demonstration [1, 39]. As they do not investigate the authoring of complex interactive experiences, they do not discuss the main challenges that need to be studied and overcome to allow immersive authoring by demonstration of complex interactions. Thus, we decided to design, implement and test an authoring tool for interactive experiences in order to better identify the challenges in both the design phase and the tool adoption.

3 SYSTEM OVERVIEW

In order to identify the challenges while designing and using interaction authoring tools, we implemented and tested a VR authoring tool based on demonstration. In this section, we introduce a simple framework for authoring interactions, present our prototype and share the implementation details of our prototype.

[3]https://tvori.co/

3.1 Framework

To illustrate the different concepts exposed here, we will consider a very simple interactive system: a virtual room with a white cube floating in the air at eyes level. There is a table in the middle of the room with a big button on top of it with the label "Change color" on it. Our user, Alice, is at the beginning standing idle in front of the table.

In an interactive system, users can act on the virtual scene and modify some of its entities using interactions. It means that, for our scene to be interactive, Alice should be able to modify the cubes around her: move them, remove them, change their color, etc. According to Yigitbas et al. [41], an interaction can be defined as an event that acts as the trigger and an effect that defines the modification done on the virtual scene. A basic interaction in our scene could be that when Alice presses the big button (**trigger**), the cube becomes blue (**effect**). Conditions can be added to the interactions. For instance, we could add as a condition that the button should be pressed with the left hand and not the right one.

To author an interaction, Alice would have to define an action that would act as a trigger, define an action that would be the effect of the interaction, and link both to show that one provokes the other. There are multiple ways to define these actions to the system, it could be programmed in a script or visually in the VR scene, but in this work, we choose to use *Demonstration*. As explained before, programming by demonstration can be easier to understand for non-programmers and can be done directly in context (i.e. inside the virtual scene). In our scene, if Alice wants to author the interaction "When a user touches the cube, it falls on the ground", she has to demonstrate the first action: *touching the cube* that is the trigger, and then demonstrate the second action: *the cube falls on the ground* (that can be done by having Alice moving the cube to the ground) that is the effect.

In our framework, we define 4 steps to author an interaction:

(1) Recording the interaction: In this step, the user records one or several actions that will be used as trigger and one or several actions that will be used as effect. Table 1 shows all the actions that can be recorded by our prototype as trigger, effect, or both.
(2) Inspecting the interaction: In the previous step, all actions performed either by the user or objects were recorded, however, not all of them are relevant to the interaction the user wants to create. In our example, when Alice records the action *touch the cube*, an action corresponding to the movement of her hand is recorded, then the action of her hand colliding with the cube. She needs to remove the action of the movement, else the interaction will be triggered only when the hand follows the same movement as the one demonstrated and then collides with the cube.
(3) Linking the trigger and the effect: The user needs then to tell the system which set of actions is the trigger of the interaction and which set is the effect.
(4) Testing the interaction: The user can then test the interaction to make sure it has the appropriate behavior.

3.2 Prototype

Based on our framework we implemented a prototype of a system that allows for the authoring of interactive immersive scenes.

3.2.1 Basic functionalities. To allow users to author an interactive experience, it needs to provide first a set of basic functionalities. The prototype allows users to create a static scene with basic 3D geometries, 3D objects, and images. Using an inspector, a panel that is displayed near a selected entity, users can change their size, color, and visibility. The system also allows users to draw trigger areas on the floor or in the air. Wireframe visual representations allow the users to position them in the environment (see Figure 2). Then, during the actual experience, these trigger areas become invisible but have a collider meaning that they can trigger a collision event. They can be used to create interactions in which the trigger is *When the user or the object moves to this*

Table 1. List of the actions that can be recorded by the tool either as a trigger or as an effect.

Action (Dim.)	Description	Trigger & Effect	
Collision (C)	Collision between two entities (e.g. objects, users' hands, users' head)	✓	✗
Move (M)	Specific movement of an entity (e.g. objects, users' hands, users' head)	✓	✓
Grab (G)	Grabbing of an object by users using the hand or a controller (pressing the trigger or grip button)	✓	✗
Release (R)	Dropping of an object by users (after it was grabbed)	✓	✗
Drag (D)	Movement of an object that has been grabbed by users	✓	✗
Look (L)	Gaze of users is focused on an object	✓	✗
Enter (En)	Users enter a trigger area	✓	✗
Exit (Ex)	Users exit a trigger area	✓	✗
Property Change (Co: Color, S/H: Visibility)	Property (e.g. colors, visibility) of an object is changed	✓	✓

specific area or *When the user or the object leaves this specific area.* Finally, users can navigate inside the scene either by physically moving or by using teleportation.

3.2.2 Recording interactions. A recording mode in the system can be activated by users at any moment and starts the saving of the performed actions. To limit the number of actions recorded, users can choose to only record either their own actions (hands/head movement, collision, grab, etc.) or the actions of objects. The user recording mode allows users to create 5 types of actions: movement, collision, grab, drag and drop. In the object recording mode, users can either record object movements or collisions with other objects or property changes such as color or visibility changes.

A puppet is also available to users that can be used to demonstrate, from a third-person perspective, actions also done by the user as it can be moved in space and each hand can be independently manipulated (see Figure 1). It can demonstrate movement and collision with hands and body. The puppet is also used to record the specific action of *Looking at a specific object/area.* As the users' gaze stares at many different objects and moves a lot, recording such an action directly with the tracking of the user's head would lead to too many events or would require to create an interaction, such as pushing a controller's button, to instantiate the creation of the event. Yet, the possibilities for such an interaction are limited since it needs to be triggered without the user looking elsewhere, which would be especially difficult for novices. Therefore we believe that using an intermediary such as a puppet could solve this issue. A yellow sphere, shown in Figure 1, in front of the puppet head represents its gaze, it can be moved by users and when it collides with an object or area it creates a Look action. This way, users have full control over the creation of look actions which limits the creation of unwanted actions.

3.2.3 Inspecting the interaction. After the recording, the saved actions are displayed in the scene using semi-transparent bubbles with a letter indicating the type of the recorded action (see Figure 3 and Table 1 for the meaning of each letter). Actions that were recorded but not wanted by users can be removed. In order to provide more information about the action, when users point at a bubble, a ghost simulation of the action is played. The bubble is positioned near the object it concerns, except when it only concerns the user's movements actions.

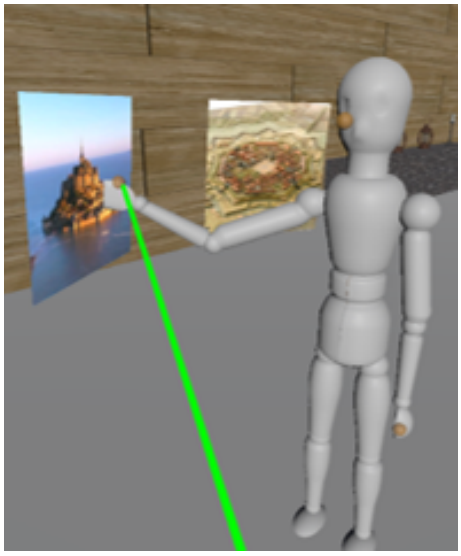

Fig. 1. Users can record interactions using a puppet that represents the user. In this picture the user manipulates the puppet to make it touch the picture and create a collision event between the picture and the user's hand.

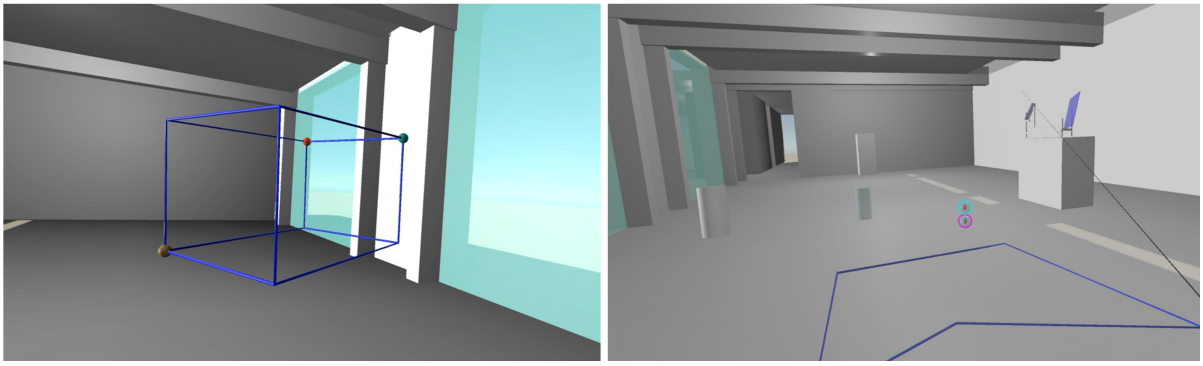

Fig. 2. Users can create trigger areas. They can shape 3D cubes (on the left) or draw an area on the floor (on the right)

By clicking on the bubble of a movement or drag action, users can show the trajectory with all the recorded points[4], and move or delete them (it is not possible to add points). It is also possible to turn the trajectory into an infinite move following the vector $\vec{AB}$ with A the center of the object and B the position of the last point.

Inspecting the action bubbles can be done at any time in the authoring process. It is not a mandatory action from users but it is rather meant to provide information on the action to users and allow them to modify trajectories.

Actions saved during the same recording are linked through a thin purple thread and the first and the last created events are highlighted with different colors. The order of their creation is used to specify in which order they have to be done. For instance, if users want to define a trigger in which a cube, a sphere, and a cylinder should

[4]In order to keep a limited number of points, the trajectory is filtered using the Ramer–Douglas–Peucker algorithm. It allows to only keep the points that define the main shape of the trajectory.

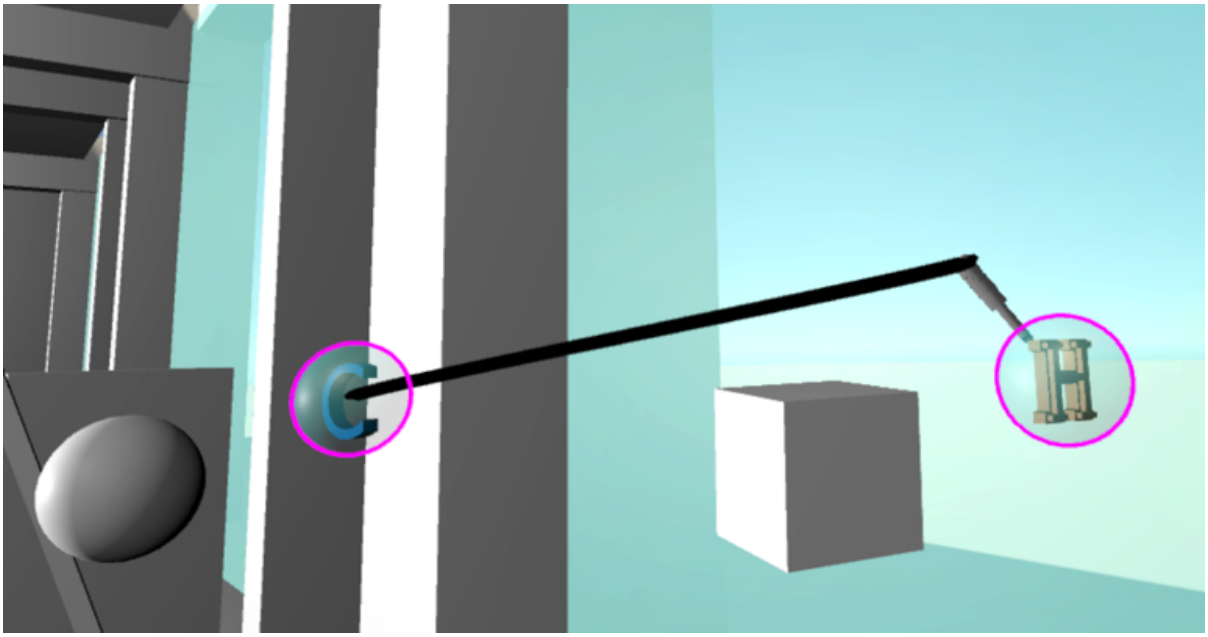

Fig. 3. Events are visualized with a floating bubble. In this simple example the Collision of the sphere with the user's hand, makes the cube become invisible (Hide)

be touched sequentially, they would record their hand touching these three objects in a row. This functionality allows users to integrate some sequences of actions into their interactions.

3.2.4 Linking the trigger and the effect. Up to this step, there are only actions, users now need to define which ones are the trigger and which ones are the effect of the interaction. This relationship is defined by drawing an arrow that goes from the last action of a set of actions (this set will be the trigger) to the first action of another set of actions (the effect). An effect can have several triggers meaning that it can be triggered by any one of them. A trigger can be linked to several effects. Using a thread to represent a link is quite common in visual programming and is used in several immersive authoring tools [11, 29, 39]. The system does not contain manipulable logical operators, however, linking two different actions to the same effect is equivalent to the OR operator.

3.2.5 Testing the interaction. It is important to allow users to quickly and easily test while creating the interactions. Thus, they can verify if the interactions behave as expected and make sure that they are adapted to the final users. For instance, users can make sure that after moving, an object will still be accessible for the next interaction (neither behind a wall, invisible, nor too high). In test mode, users can choose to show or not the graph (bubbles + links) of the recorded interactions.

3.3 Implementation

The prototype was implemented with Unity 2019.4.18f1 using C# and SteamVR. The prototype runs on a HTC Vive and a laptop computer MSI GT63 Titan 10SF with an Intel Processor Core i7-10750H CPU 2.59 GHz, 16Go RAM and a Nvidia GTX 2070 graphic card. The code is publicly available and open source[5].

[5] https://gitlab.inria.fr/egros/authoring_by_demonstration

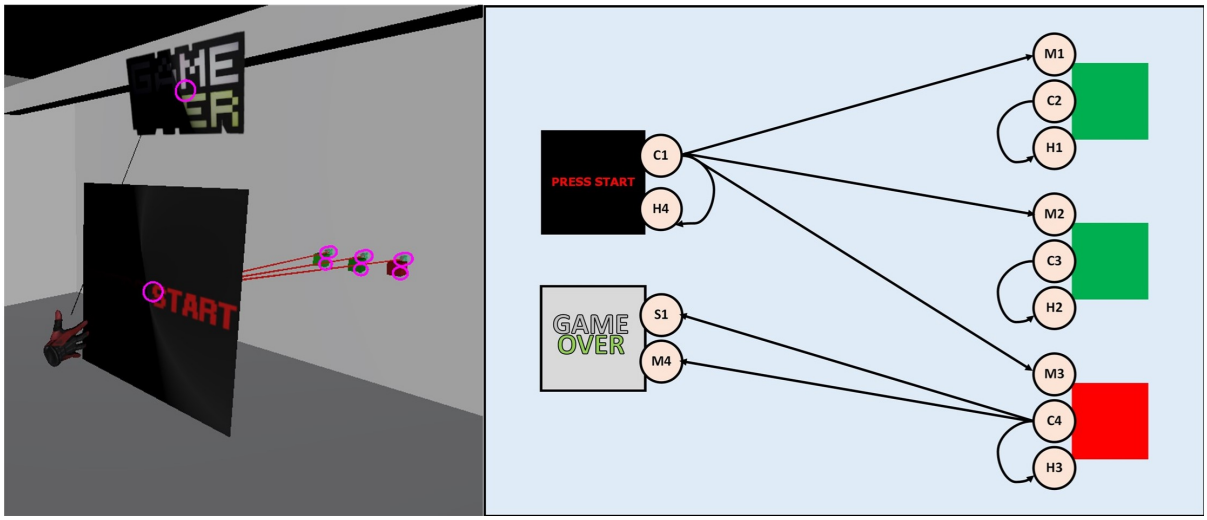

Fig. 4. On the left, a screen capture of the authoring of the Beat Saber scene. The diagram on the right describes the logic behind the Beat Saber interactive scene we authored.

4 USE CASES

In this section, we present and discuss two interactive experiences, a Beat Saber inspired scene and an escape game, authored with our prototype. The figures 4 and 5 are provided in this paper as a graphical representation of the authored interactions to help readers better understand the interactions, however, participants to the study described in section 5 did not see such visualization.

4.1 Beat saber

For this interactive scene, we drew inspiration from the VR game Beat Saber [6].

4.1.1 Scenario. The user starts facing a picture with "Press to start" written on it. When they touch the picture, it disappears and two green cubes and one red cube move toward them. The user needs to touch the green cubes to make them disappear. However, if they touch the red cube with their hand, all cubes disappear and a "game over" picture falls in front of them.

4.1.2 Authored interactions . We describe here the logic behind the Beat Saber scene and provide a diagram (see Figure 4) as a support to the description. To create this interactive scene we first created a collision event (C1) between the user's hand and the "Press to start" picture.

Each cube has three events attached to it, an infinite linear movement event (M1, M2, M3), a collision with the user's hand event (C2, C3, C4), and a hide event (H1, H2, H3). C2, C3 and C4 trigger respectively H1, H2, and H3, which makes the cubes disappear when they collide the user's hand. C1 is linked to M1, M2 and M3 so when the user touches the picture, the cubes start moving toward them. It is also linked to a hide event (H4) to make the picture disappear once the game starts.

The "Game Over" picture has two events attached to it, a show event (S1) to make it appear and a move event (M4) to make the picture fall in front of the user. Finally, C4, which is attached to the red cube, is linked to S1 and M4.

[6] https://store.steampowered.com/app/620980/Beat_Saber/

4.1.3 Discussion. In order to make the game interesting, we would need lots of cubes coming at the player. If we used the copy and paste tool to create them all, the authoring environment would soon be overcrowded. Thus, users need a way to specify that they want multiple identical objects with the same behavior created over time. In order to do so, we would need a *"virtual printer"* that would take a template as input and can be activated or deactivated during the experience.

Since we don't want all cubes to come at once, we would also need the possibility to control when they are produced by the virtual printer. Thus, the trigger sent to activate the printer needs to be conditioned with a time delay. To improve enjoyability of the game, we need to add randomity to this time delay.

Finally, since the interactive experience is a game, we want to count points. For instance, each time the player touches a green cube, a score displayed on the wall in front of them would get incremented. This requires to have a variable that the user can modify and apply arithmetical operations on, such as additions or subtractions.

4.2 Escape Game

4.2.1 Scenario. The user is in a virtual room with no accessible exit since there are three huge blocks blocking the corridor. In order to escape, the user needs to solve three enigmas. All three enigmas are scattered inside the room. The first enigma is a bomb on a block in the middle of the room. The bomb has three colored wires, blue, green, and red. Next to the bomb, there are a wire cutter and a sphere half sunk inside the block. When the user grabs the wire cutter and touches the blue wire with it, nothing happens. The user drops the wire cutter and touches the sphere with their right hand. The sphere turns red, then green, then blue. The user grabs the wire cutter again and touches in order the red, the green, and the blue wires. Each time the wire cutter collides with a wire, the wire disappears. Once all wires disappeared, the bomb disappears as well as one of the blocks inside the corridor. The first enigma is solved.

For the second enigma, the user has to open a chest. When the user touches or tries to grab the chest, nothing happens. The user starts looking for a key and finds one hidden in the room. When they drag the key to the keyhole of the chest, the top of the chest disappears. At the bottom of the chest, there is a hammer. The user grabs it and starts walking to the corridor. In the corridor, one of the two blocks is cracked. The user uses the hammer to hit it and the block disappears.

For the last enigma, there is a maze on the wall with a red sphere in the middle of it. In front of the maze are displayed four buttons, up, down, right, and left. As the maze is a spiral, the user needs to press alternately the down, right, up, and left buttons until the sphere is outside the maze. When the sphere gets outside the maze, the last block in the corridor disappears.

The user then walks through the corridor and reaches the exit. When the user crosses the exit, a trophy and a picture with 'Congratulation' written on it appear.

4.2.2 Authored interactions. We now describe the logic behind the Escape Game interactive scene. We also provide a visual support to this description in Figure 5. Each of the three blocks that seal off the exit has a hide event (H1, H2, H3) attached to it. The first enigma is composed of six objects, a bomb, a wire cutter, three wires (red, green and blue) and a sphere. The wire cutter has four events attached to it, a grab event (G1) to specify this object can be grabbed by the final user, and three collision events (C1, C2 and C3) that are connected with an order link so they have to be triggered in the right order. Each collision event is linked to a hide event attached to a wire (H4, H5, H6). The sphere has four events attached to it, a collision event (C4) with the user's hand that is linked to a red color event (Co1) which is linked to a green color event (Co2), which is finally linked to a blue color event (Co3). Finally, the bomb has a hide event attached to it (H7) which is linked to H1.

The second enigma is composed of four objects, a key, a closed chest, an open chest and a hammer. A grab event (G2) and a drag event (D1) are attached to the key. The drag event's trajectory depicts the key going through the chest key hole. The closed chest has a hide event (H8) and the open chest a show event (S1) attached to it. These

two events are triggered by D1. A grab event (G3) is attached to the hammer as well as a collision event (C5) that is triggered when the hammer collides with the cracked block at the exit. Finally, C5 is linked to H2.

The third enigma has six objects, a spiral maze, a ball and four direction buttons, and it only has two types of events, four collision events (C6, C7, C8, C9) between the direction buttons and the user's hand, and 11 movement events (M1 to M11). The movement events describe the 11 movements the ball needs to do to get out of the maze and are connected with an order link. C6 is linked to all the left movements (M4, M8), C7 to all the up movements (M3, M7, M11), C8 to all the right movements (M2, M6, M10), and C9 to all the down movements (M1, M5, M9). Finally, the last movement M11 triggers H3.

The exit has two objects, a congratulation picture and a cup, as well as an interactive object, a 2D trigger area. The trigger area covers the exit of the room and has an entry event (En1) attached to it that is triggered when the user enters the area. En1 is linked to two show events (S2, S3) attached to the picture and the cup.

4.2.3 Discussion. In order for an event to work with either the right or the left hand, the creator needs to record it twice, one time for each hand. For instance, for the bomb enigma in the escape game, when the user touches the sphere, if we want the sphere to change color no matter which hand touches it, we need to record two collisions, one for each hand. As the system relies on a single demonstration of the interaction, it needs additional information to deduce if the interaction targets one specific hand or both.

In this scene, if the user touches the wires with the wire cutter in the wrong order, for instance by touching first the blue wire instead of the red wire, nothing happens. To create a "Game Over" message that appears when the user cuts the blue wire, we would need to add a condition "The event collision between wire cutter and red wire hasn't been triggered" which requires using the NOT and AND logical operators. Yet, our system does not allow its users to create logical operators.

Finally, to make the sphere move into the spiral maze, we recorded all expected movements in one recording. Thus, as explained in the section 3.2.3, all actions were linked by a time constraint forcing the action n to be triggered before being able to trigger the action n+1. In this case, in order to trigger the third expected movement, the first two movements needed to be executed. However, it is not possible to go backward, nor choose between two different directions. This example would require to have the object itself as a reference frame for the movement as well as being able to control the length of the translation as distances vary in the maze. In our system, movements have the world has a reference frame, which means that the coordinates (0,0,0) refer to the origin of the world coordinates and not the object's position in the world.

5 FEEDBACK SESSION

We performed a feedback session to better understand the challenges that participants could face while authoring interactions. In this section we discuss not only the challenges inherent in interaction authoring, but also those related to design flaws so it could benefit to future designers.

5.1 Study Design

5.1.1 Participants. We had 6 participants (4 female and 2 male) aged 24 to 29, (average:26), all participants had experience with either virtual or augmented reality. All participants had experience in programming in general, however, two of them had no programming experience in neither virtual reality nor augmented reality. Among the participants, two of them were left-handed.

5.1.2 Procedure. The participants were first asked to fill out a demographic questionnaire. To show the participants an example of what can be authored with the tool, we started trials by showing the participants a reduced version of the escape game. We showed the participants the test mode without the logic appearing so they could

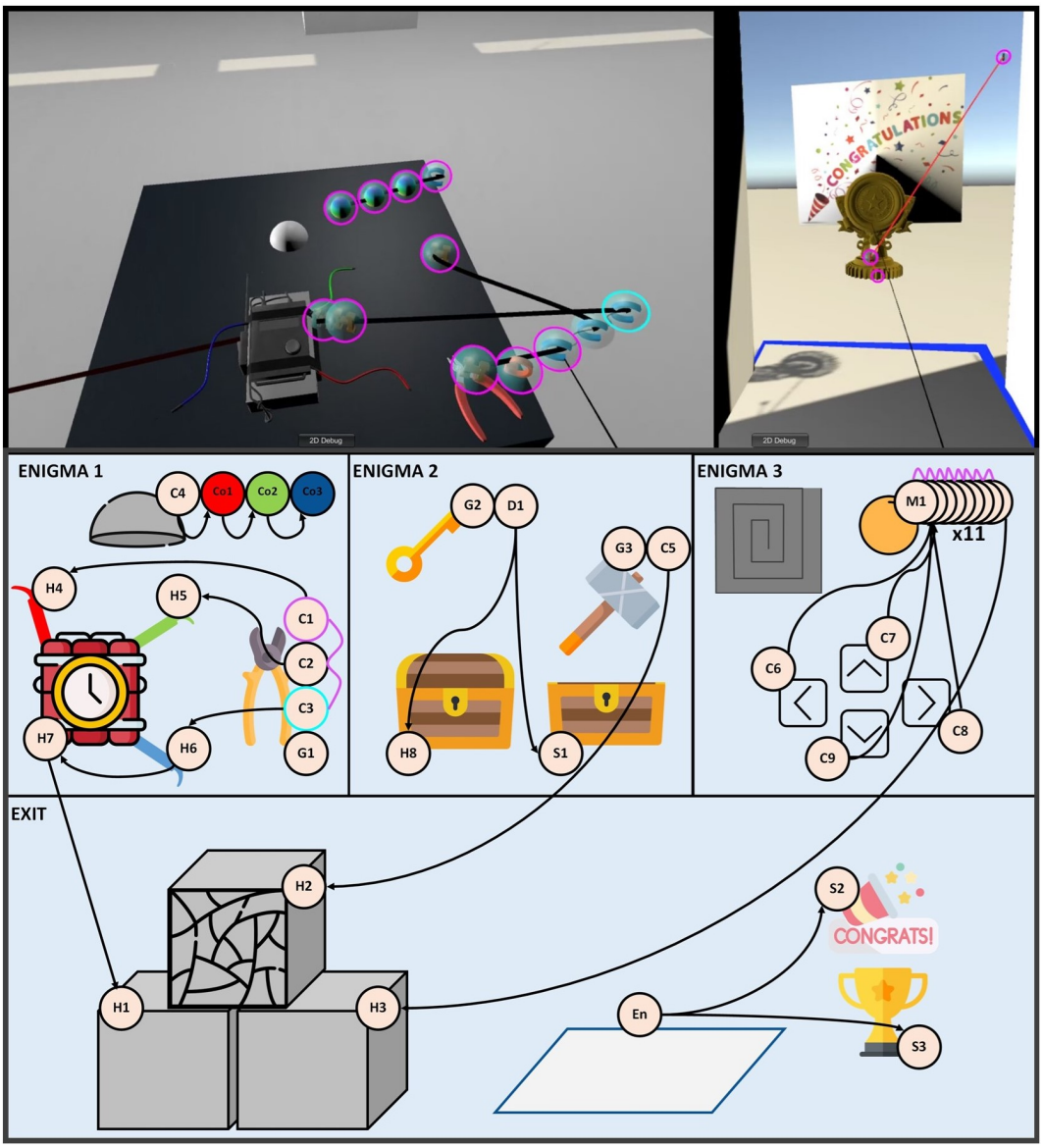

Fig. 5. A screen capture of the first enigma's logic (top left), a screen capture of the logic of the exit animation (top right), and a diagram that describes the logic behind the Escape Game interactive scene we authored (bottom).

not see it first. We then let them try the experience by themselves. Then, we explained all functionalities to the participants through demonstration and answered any questions they could have.

For the next step, the participants were instructed to author two interactive experiences given by the experimenter. The first instruction, "When the user touches an object, the color of the object changes" described a low-complexity scene and was meant to help the participants get familiar with interaction authoring. The second

instruction, "When the user gets out of the room, an object appears behind him. When the user turns over and looks at it, the object starts falling.", depicted a more complex scene and forced the participants to use the puppet recording mode as well as the trigger area functionality.

Once the participants performed both tasks, they did a free authoring session with no time constraint. The only condition was to create at least one interaction. During the constrained and free creation steps, automatic backups were made every 30 seconds. The creation tasks were followed by a semi-structured interview during which participants were asked to share their feelings about their experience. At the end of the study, participants were asked to fill out a raw NASA-TLX questionnaire [13] followed by a semi-structured interview.

5.2 Results

5 of the 6 participants showed a significant improvement during training and one participant had difficulties understanding how the authoring tools for interaction worked and couldn't create successful interactions during the free creation step. In total, the participants spent less than two hours using the system. At the end, the participants mentioned feeling more confident with the system and experimenters noticed that participants progressively made less mistakes and needed less help during the session.

5.2.1 NASA-TLX. Results to the raw NASA-TLX questionnaire are depicted in Figure 6. The participants had a tendency to find the system mentally demanding as only two participants rated the mental demand below 50 and half of them rated the effort they put into their work above 50. Yet, most participants seemed satisfied with the work they achieved, all participants rated their performance equal or below 50, meaning they judge their work as successful. Moreover, the frustration scores were very low as 5 of the 6 participants rated their frustration equal to or below 20.

5.2.2 Interactions Log. We analyzed the final results of the free creation task and seek which interactions were the most created and whether they were correctly used. On Figure 7, we can observe that the events move, collision, show, color, and zone collider are the most used. On the other side, the events drag, drop and look were never correctly used. The most popular events actually are the events that were necessary to complete the two constrained tasks. We can notice that the look event was not used during the free creation task even though it was required in the second constrained task, however, many participants did not instinctively realize the need for the look event and either realized it by themselves during creation or had to be reminded.

5.2.3 Feedback from interviews. One of the main difficulties reported by the participants was the difficulty to assimilate all functionalities. The participants not only had to learn how to use the interaction authoring tools but also the modeling tools, as well as how to navigate and how to manipulate objects. Therefore, the participants had a lot to remember, which hindered their experience and could explain the high mental demand found in the NASA-TLX scores. Moreover, all participants thought that they would get better if they had more time to learn how to use the tool.

P1 suggested adding a history to the palette in order to easily visualize what had already been created and what is left to make. The possibility to add a history or a 2D representation of the authored interactions on the palette was also mentioned by P5. Indeed having a mixed interface, both 2D and 3D would allow the user to have a global view of the program. Once the program is authored, the user could easily rearrange or select blocks of codes.

The participants could easily read and understand the events they created. The letters inside bubbles were a first clue to remember the nature of the event and the participants could easily get more information by pointing at the bubble in order to launch the ghost simulation. They particularly enjoyed the ghost visualization for events, however, they believed it could be improved. For instance, when the collision simulation is launched, a ghost of

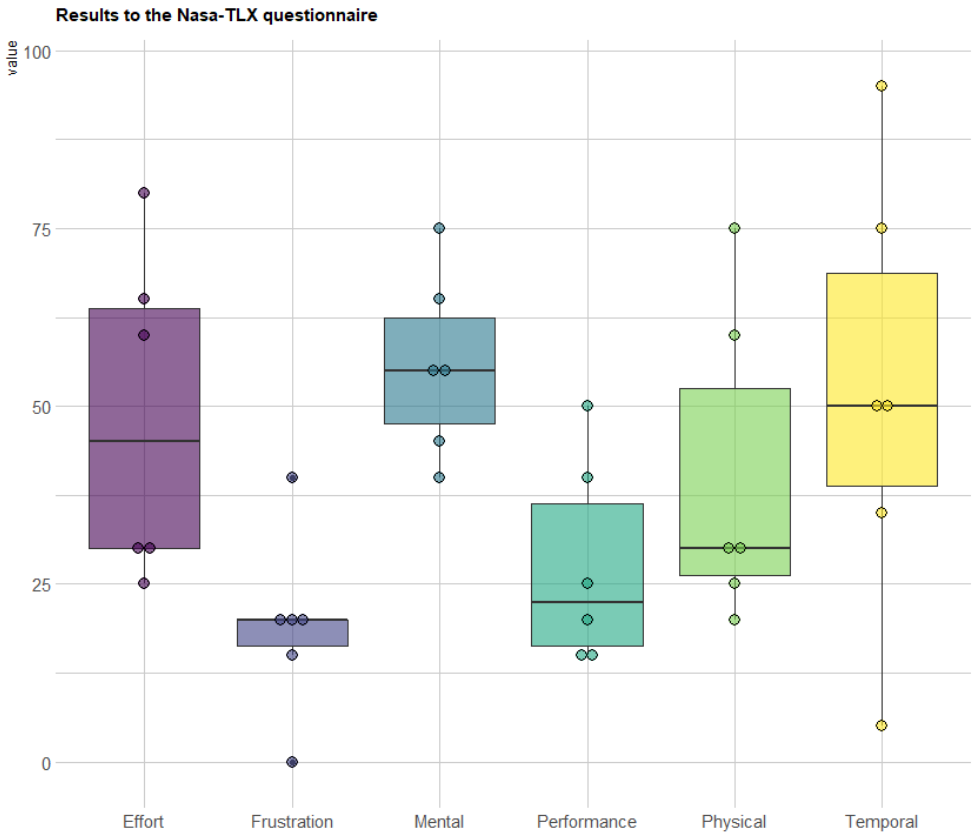

Fig. 6. Results to the NASA-TLX questionnaire

the object appears next to the collided object and hits it. Yet the user doesn't know where the ghost will appear. P5 suggested adding a textual label that could be hidden attached to the event.

5.2.4 Experimenters' observations. Several participants had difficulties understanding how to link triggers to the effect. Two of them linked the effect before the trigger and therefore inverted the relationship between the two events. In a more general manner, most participants reminded themselves out loud the order "trigger then effect" to avoid any mistake. During the first constrained task P4 forgot to link the trigger to its action before testing and P2 launched the test mode before creating the trigger and expected the action to be triggered by default. All these observations happened during the two constrained tasks and did not happen again after our clarification.

During the first constrained task, 3 participants misidentified the events they were expected to create. For the first task "The color of the object changes when the user touches it", P2 and P4 created a grab event instead of a hand collision and P3 kept a hand movement event in addition to the collision.

The participants also had a tendency to get confused about the way attributes work. Some of them started changing the object's color without recording, thinking it would create a color event. Several participants also

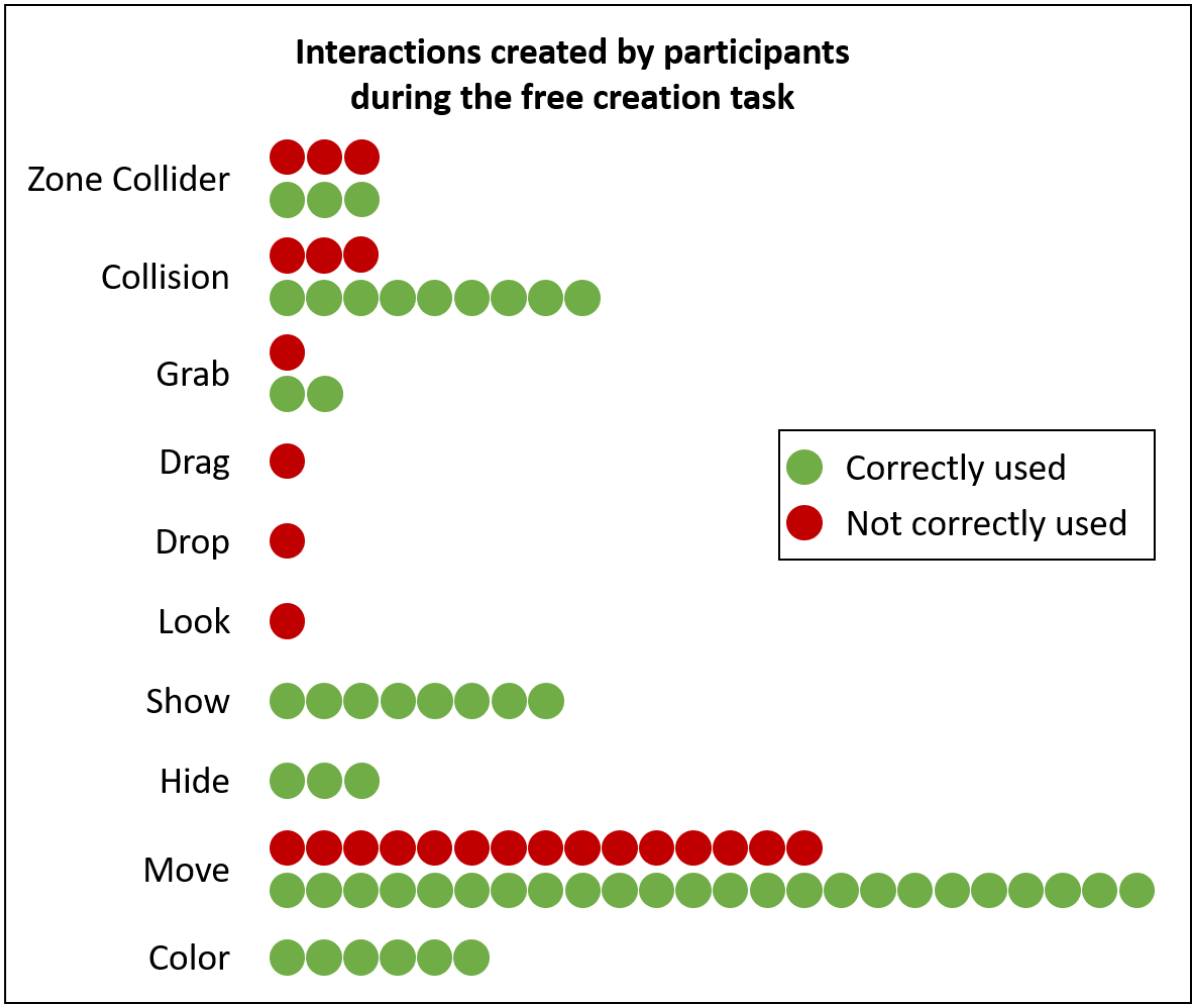

Fig. 7. Interactions created by the participants during the free creation task: the participants mostly used interactions that they had to use during the first two constrained tasks

forgot to set the default visibility before recording and ended up creating the opposite event (show instead of hide or hide instead of show).

During the free creation step, two participants launched the wrong recording modes. P1 recorded the left hand actions instead of the right hand and P3 launched the right hand recording instead of the objects recording mode.

6 DISCUSSION

In this section we discuss the challenges of authoring by demonstration as well as the limitations to our study.

6.1 Impossible manipulation

Authoring by demonstration allows users to specify an event by grabbing an object and directly manipulating it. Yet, some wanted behaviors could be conflicting with the users' physical constraints. For instance, if the user wants an object to spin and turn on 360° or more, their arm won't be able to turn that much, therefore the recorded move will not be smooth nor complete. A similar problem could occur if the user wants to move the object over a long distance or manipulate very large objects. One solution to this problem is to use a World-In-Miniature (WIM) [34], yet visualizing and manipulating small objects in it would be laborious.

6.2 Demonstration of abstract concepts

Looking back at the Beat Saber use case (see Section 4), let's imagine we want to create the following interactive experience, "Every 0.2 to 2 seconds, a red or green cube appears and moves towards the player. If the player touches the green cube, they win a point, but if they touch a red cube, the game stops and a game over message is displayed". These two sentences involve the demonstration of many abstract concepts such as time to specify the delay between two appearances, randomness to make this time vary, multiple instantiations of objects with similar behaviors, generalization of an interaction to all green or red cubes and not just the one the user interacts with during the demonstration, and variables for score management.

Some of these concepts are intangible and therefore are not directly manipulable for the demonstration. In our example, it is the case for time, randomness, multiple instantiations, and digital values. To make these concepts manipulable and allow demonstration, one solution would be to materialize them by creating manipulable metaphors. For instance, in the Beat Saber use case (see Section 4), we mentioned a virtual printer to instantiate multiple times interactive objects. One of the main challenges for this kind of virtual metaphor is to create a visualization that speaks for itself and is easily manipulable. We believe that the manipulation and demonstration of time is an interesting research topic. We encourage future research to explore how users may include time concepts such as delays or order restrictions during the authoring process.

In the same example, we would also need to be able to generalize the collision between the player's hand and the green cubes, to both of the player's hands on one side, and to all green cubes on the other side. To do so the user starts the recording and touches a green cube with their dominant hand. This demonstration has several possible interpretations. The collision can involve either both hands or the dominant hand only. Similarly, the system does not know if the interaction targets only this specific cube, all green cubes, all cubes, or all green objects.

This constitutes a second challenge involved by abstract concepts, generalizing a demonstration to a set of entities sharing one or several similar properties. Such a deduction isn't possible based on a single demonstration as there are too many possible interpretations. Thus, future research should focus on either context specification or the use of multiple demonstrations, a method already explored for authoring by demonstration [32]. Such a method would also enable logic deduction, and differentiate conjunction from disjunction.

6.3 Process ambiguities

One demonstration can lead to several outcomes. As the user performs the demonstration, the recording is polluted by all the actions the user has to do to simulate the event but that are not the target of the demonstration. We illustrate this challenge with a simple demonstration. *"The user **walks** to the sphere, they **reach** for it, as the hand **touches** the sphere they **grab** it, **drag** it on their right and **drop** it, and finally stop the recording."*. This single demonstration has many possible interpretations. Does the user want the sphere to move when they walk toward it, or when the hand touches it? The point of the collision or the hand's trajectory could also matter. Another possible interpretation is the demonstration of a dragging movement.

As we can see, it is unsure which actions are of interest nor which object is the focus of the demonstration (user body, hands, or sphere). To partially resolve this issue we split the recording according to the referent of the action, yet it disables the possibility to demonstrate an interaction at once as well as automatic logic creation.

The issue of ambiguous demonstrations has previously been studied in human-robot interaction. Naive human teachers are likely to perform ambiguous and incomplete demonstrations to robots which need to interpret the teachers' intent. Future VR authoring tools by demonstration could for instance build upon Breazeal et al.'s work [5], they designed a robot which is a socially engaged and cognitive learner.

6.4 Creating and visualizing logic

To improve the Escape Game use case (see Section 4), we may want the bomb to explode if the player does not cut the wires in the right order. To do so, the user needs to specify "If the player cuts the blue wire (Event A) or the green wire (Event B) but the red wire is not cut (! Event C), then the bomb explodes (Event D)." which corresponds to the following logic "If (A || B) && !C, do D". Yet, the link connectors commonly used in immersive systems such as our prototype or the systems mentioned in the related work [3, 29, 39] can only specify the logical operator OR by connecting two trigger events to the same action event.

Enabling all logical connectors would require to complicate this logic creation process whereas participants are already struggling to handle the current logic (see Section 5). Thus we believe the logic should be created by the system and deduced from the recorded demonstration. Yet, it raises the question of how the system can differentiate conjunction from disjunction in a demonstration. Such logic elements will also need to be visualized and understood by non-programmers and avoid crowding too much space. Visual occlusion is one of the challenges mentioned in FlowMatic [42]. There is also a risk of creating an interaction graph that would become complex to read as interaction can be scattered in the virtual space. Some participants of the user study already mentioned getting confused with the current graph and suggested using a 2D visualization or a history.

Moreover, the logic is not limited to the OR, AND and NOT operators. In this paper, we limited our framework to triggers being the result of a binary change of state (done or not done), yet, it could also be continuous based on the state of an object. For instance, the color of a virtual thermometer could vary depending on the size of a virtual mercury bar. Other elements could also be added to this trigger-action relation, such as delays, order constraints and iterations. We could also consider other logic concepts such as Vernier and Nigay's framework [38] or applying the set theory to movements. Thus, creating logic using programming by demonstration is a complex subject which requires to be explored.

6.5 Visual programming and programming by demonstration

The goal of the paper is not to claim the superiority of programming by demonstration over visual programming. We believe that it is first necessary to identify and study the challenges raised by immersive authoring of interactive experiences by demonstration.

During the feedback session, we did not provide the participants with visual representations similar to figures 4 and 5 because we wanted participants to create the interactions by demonstration from spoken language without going through a graphical programming representation first. In future studies, it would be interesting to compare the participants' capacities to create an interactive experience from spoken language using either visual programming or programming by demonstration.

It is most likely that, when compared, visual programming and programming by demonstration happen to have both drawbacks and benefits. Thus, both methods could be complementary. Programming by demonstration is likely to show strengths whenever spatiality is involved (movements, collisions...) and when figuring out the relations between objects. However, visual programming seems more promising to easily get an overview of the

created interactions and to handle abstract concepts. Thus it would be interesting to explore, in the future, the use of an authoring system combining both methods.

6.6 Provide assistance to users

All our participants had at least basis in programming, yet they still struggled to decompose the interactive experiences they had to author into primary events. Creating the logic, despite being reduced to its minimum, also required the users to go through a learning phase. Learning Authoring by demonstration can induce an important cognitive load for users. They need to learn how to navigate and interact with the virtual environment, model objects, author and debug interactions. As the authoring tool gains in power and expressivity, functionalities are likely to multiply and the complexity to rise.

Thus, we believe authoring systems should provide as much assistance as possible to their users. In VRFromX [17] users can benefit from IA assistance during the modeling phase and an affordance recommander during the authoring phase.

6.7 Limitations

Our analysis of the challenges faces the limitations of our prototype and feedback session. All of our participants had at least basic knowledge in coding, thus, we might have missed challenges that non-developers users would have. Yet, we believe that the challenges we identified with our participants can be generalized to non-developer users and that our work provides a foundation for future research in interaction authoring. In order to create interactions, users needed to get familiar with prerequisite functionalities such as navigation, grabbing, and modeling. These functionalities were not the focus of our study yet were necessary to be mastered by users. This resulted in an increased learning curve of the prototype. In addition to that, the learning phase was hindered by the communication and awareness issues inherent in the isolation of VR HMDs. Finally, the participants experience was hindered by the technical limitations of our prototype. We implemented a basic modeling tool for this study, yet we noticed that the limitations of the modeling system has repercussions on the expressivity of the authoring, as well as its usability.

7 CONCLUSION AND PERSPECTIVES

In this paper we investigate the challenges of immersive authoring by demonstration of interactive VR experiences through the design, implementation and evaluation of a prototype. We expect this paper to pave the way of future research on immersive authoring by demonstration, by highlighting design issues that need to be studied.

Beyond authoring prototypes, immersed authoring by demonstration is useful to edit trainings and tutorials directly in the immersive environment. Lécuyer et al. [19] actually enable medical training using authoring by demonstration which could be generalized to other domains of application. Implementing a tutorial for complex VR application is laborious and an authoring tool by demonstration would greatly facilitate the process.

REFERENCES

[1] Rahul Arora, Rubaiat Habib Kazi, Danny M. Kaufman, Wilmot Li, and Karan Singh. 2019. MagicalHands: Mid-Air Hand Gestures for Animating in VR. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 463–477. https://doi.org/10.1145/3332165.3347942

[2] Narges Ashtari, Andrea Bunt, Joanna McGrenere, Michael Nebeling, and Parmit K. Chilana. 2020. Creating Augmented and Virtual Reality Applications: Current Practices, Challenges, and Opportunities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376722

[3] Lee Beever, Serban Pop, and Nigel W. John. 2020. LevelEd VR: A virtual reality level editor and workflow for virtual reality level design. In *2020 IEEE Conference on Games (CoG)*. IEEE, New York, NY, USA, 136–143. https://doi.org/10.1109/CoG47356.2020.9231769

[4] Bhaskar Bhattacharya and Eliot H. Winer. 2019. Augmented reality via expert demonstration authoring (AREDA). *Computers in Industry* 105 (2019), 61–79. https://doi.org/10.1016/j.compind.2018.04.021

[5] Cynthia Breazeal, Matt Berlin, Andrew Brooks, Jesse Gray, and Andrea L. Thomaz. 2006. Using perspective taking to learn from ambiguous demonstrations. *Robotics and Autonomous Systems* 54, 5 (2006), 385–393. https://doi.org/10.1016/j.robot.2006.02.004 The Social Mechanisms of Robot Programming from Demonstration.

[6] Colin Burns, Eric Dishman, William Verplank, and Bud Lassiter. 1994. Actors, Hairdos & Videotape—Informance Design. In *Conference Companion on Human Factors in Computing Systems* (Boston, Massachusetts, USA) *(CHI '94)*. Association for Computing Machinery, New York, NY, USA, 119–120. https://doi.org/10.1145/259963.260102

[7] Yuanzhi Cao, Tianyi Wang, Xun Qian, Pawan S. Rao, Manav Wadhawan, Ke Huo, and Karthik Ramani. 2019. GhostAR: A Time-Space Editor for Embodied Authoring of Human-Robot Collaborative Task with Augmented Reality. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 521–534. https://doi.org/10.1145/3332165.3347902

[8] Daniel W. Carruth. 2017. Virtual reality for education and workforce training. In *2017 15th International Conference on Emerging eLearning Technologies and Applications (ICETA)*. IEEE, New York, NY, USA, 1–6. https://doi.org/10.1109/ICETA.2017.8102472

[9] Marco Cavallo and Angus G. Forbes. 2019. CAVE-AR: A VR Authoring System to Interactively Design, Simulate, and Debug Multi-user AR Experiences. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, New York, NY, USA, 872–873. https://doi.org/10.1109/VR.2019.8798148

[10] Zhutian Chen, Yijia Su, Yifang Wang, Qianwen Wang, Huamin Qu, and Yingcai Wu. 2020. MARVisT: Authoring Glyph-Based Visualization in Mobile Augmented Reality. *IEEE Transactions on Visualization and Computer Graphics* 26, 8 (2020), 2645–2658. https://doi.org/10.1109/TVCG.2019.2892415

[11] Barrett Ens, Fraser Anderson, Tovi Grossman, Michelle Annett, Pourang Irani, and George Fitzmaurice. 2017. Ivy: Exploring Spatially Situated Visual Programming for Authoring and Understanding Intelligent Environments. In *GI '17: Proceedings of the 43rd Graphics Interface Conference*. Canadian Human-Computer Communications Society, Waterloo, Canada.

[12] Laura Freina and Michela Ott. 2015. A literature review on immersive virtual reality in education: state of the art and perspectives. In *The international scientific conference elearning and software for education*, Vol. 1. eLSE, Bucharest, Romania, 10–1007.

[13] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Human Mental Workload*, Peter A. Hancock and Najmedin Meshkati (Eds.). Advances in Psychology, Vol. 52. North-Holland, Amsterdam, Holland, 139–183. https://doi.org/10.1016/S0166-4115(08)62386-9

[14] Björn Hartmann, Leith Abdulla, Manas Mittal, and Scott R. Klemmer. 2007. Authoring Sensor-Based Interactions by Demonstration with Direct Manipulation and Pattern Recognition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '07)*. Association for Computing Machinery, New York, NY, USA, 145–154. https://doi.org/10.1145/1240624.1240646

[15] Valentin Heun, James Hobin, and Pattie Maes. 2013. Reality Editor: Programming Smarter Objects. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication* (Zurich, Switzerland) *(UbiComp '13 Adjunct)*. Association for Computing Machinery, New York, NY, USA, 307–310. https://doi.org/10.1145/2494091.2494185

[16] Shotaro Ichikawa, Kazuki Takashima, Anthony Tang, and Yoshifumi Kitamura. 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* (Tokyo, Japan) *(VRST '18)*. Association for Computing Machinery, New York, NY, USA, Article 6, 5 pages. https://doi.org/10.1145/3281505.3281517

[17] Ananya Ipsita, Hao Li, Runlin Duan, Yuanzhi Cao, Subramanian Chidambaram, Min Liu, and Karthik Ramani. 2021. VRFromX: From Scanned Reality to Interactive Virtual Experience with Human-in-the-Loop. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI EA '21)*. Association for Computing Machinery, New York, NY, USA, Article 289, 7 pages. https://doi.org/10.1145/3411763.3451747

[18] Ben Lang. 2021. 12 Tools for Painting, Sculpting, & Animating in VR. https://www.roadtovr.com/vr-painting-drawing-modeling-animation-art-tools-quest-pc/. Accessed: 2022-10-05.

[19] Flavien Lécuyer, Valérie Gouranton, Adrien Reuzeau, Ronan Gaugne, and Bruno Arnaldi. 2019. Create by Doing – Action Sequencing in VR. In *Advances in Computer Graphics*, Marina Gavrilova, Jian Chang, Nadia Magnenat Thalmann, Eckhard Hitzer, and Hiroshi Ishikawa (Eds.). Springer International Publishing, Cham, 329–335.

[20] Gun A. Lee, Gerard J. Kim, and Mark Billinghurst. 2005. Immersive Authoring: What You EXperience Is What You Get (WYXIWYG). *Commun. ACM* 48, 7 (July 2005), 76–81. https://doi.org/10.1145/1070838.1070840

[21] Germán Leiva, Jens Emil Grønbæk, Clemens Nylandsted Klokmose, Cuong Nguyen, Rubaiat Habib Kazi, and Paul Asente. 2021. Rapido: Prototyping Interactive AR Experiences through Programming by Demonstration. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 626–637. https://doi.org/10.1145/3472749.3474774

[22] Hao Lü and Yang Li. 2013. Gesture Studio: Authoring Multi-Touch Interactions through Demonstration and Declaration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 257–266. https://doi.org/10.1145/2470654.2470690

[23] Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. 2005. DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences. *ACM Trans. Graph.* 24, 3 (July 2005), 932. https://doi.org/10.1145/1073204.1073288

[24] Sina Masnadi, Andrés N. Vargas González, Brian Williamson, and Joseph J. LaViola. 2020. AffordIt!: A Tool for Authoring Object Component Behavior in VR. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. IEEE, New York, NY, USA, 740–741. https://doi.org/10.1109/VRW50115.2020.00221

[25] Leon Müller, Ken Pfeuffer, Jan Gugenheimer, Bastian Pfleging, Sarah Prange, and Florian Alt. 2021. SpatialProto: Exploring Real-World Motion Captures for Rapid Prototyping of Interactive Mixed Reality. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 363, 13 pages. https://doi.org/10.1145/3411764.3445560

[26] Michael Nebeling and Katy Madier. 2019. 360proto: Making Interactive Virtual Reality & Augmented Reality Prototypes from Paper. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300826

[27] Michael Nebeling, Janet Nebeling, Ao Yu, and Rob Rumble. 2018. ProtoAR: Rapid Physical-Digital Prototyping of Mobile Augmented Reality Applications. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173927

[28] Michael Nebeling and Maximilian Speicher. 2018. The Trouble with Augmented Reality/Virtual Reality Authoring Tools. In *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. IEEE, New York, NY, USA, 333–337. https://doi.org/10.1109/ISMAR-Adjunct.2018.00098

[29] Gary Ng, Joon Gi Shin, Alexander Plopski, Christian Sandor, and Daniel Saakes. 2018. Situated Game Level Editing in Augmented Reality. In *Proceedings of the Twelfth International Conference on Tangible, Embedded, and Embodied Interaction* (Stockholm, Sweden) *(TEI '18)*. Association for Computing Machinery, New York, NY, USA, 409–418. https://doi.org/10.1145/3173225.3173230

[30] Antti Oulasvirta, Esko Kurvinen, and Tomi Kankainen. 2003. Understanding Contexts by Being There: Case Studies in Bodystorming. *Personal Ubiquitous Comput.* 7, 2 (jul 2003), 125–134. https://doi.org/10.1007/s00779-003-0238-7

[31] Arnaud Prouzeau, Yuchen Wang, Barrett Ens, Wesley Willett, and Tim Dwyer. 2020. Corsican Twin: Authoring In Situ Augmented Reality Visualisations in Virtual Reality. In *Proceedings of the International Conference on Advanced Visual Interfaces* (Salerno, Italy) *(AVI '20)*. Association for Computing Machinery, New York, NY, USA, Article 11, 9 pages. https://doi.org/10.1145/3399715.3399743

[32] Bahador Saket, Hannah Kim, Eli T. Brown, and Alex Endert. 2017. Visualization by Demonstration: An Interaction Paradigm for Visual Data Exploration. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 331–340. https://doi.org/10.1109/TVCG.2016.2598839

[33] Ronell Sicat, Jiabao Li, Junyoung Choi, Maxime Cordeil, Won-Ki Jeong, Benjamin Bach, and Hanspeter Pfister. 2019. DXR: A Toolkit for Building Immersive Data Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2019), 715–725. https://doi.org/10.1109/TVCG.2018.2865152

[34] Richard Stoakley, Matthew J. Conway, and Randy Pausch. 1995. Virtual Reality on a WIM: Interactive Worlds in Miniature. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '95)*. ACM Press/Addison-Wesley Publishing Co., USA, 265–272. https://doi.org/10.1145/223904.223938

[35] Ryo Suzuki, Rubaiat Habib Kazi, Li-yi Wei, Stephen DiVerdi, Wilmot Li, and Daniel Leithinger. 2020. *RealitySketch: Embedding Responsive Graphics and Visualizations in AR through Dynamic Sketching*. Association for Computing Machinery, New York, NY, USA, 166–181. https://doi.org/10.1145/3379337.3415892

[36] Anastasios Theodoropoulos and Angeliki Antoniou. 2022. VR Games in Cultural Heritage: A Systematic Review of the Emerging Fields of Virtual Reality and Culture Games. *Applied Sciences* 12, 17 (2022), 19. https://doi.org/10.3390/app12178476

[37] Andrés Vargas González, Senglee Koh, Katelynn Kapalo, Robert Sottilare, Patrick Garrity, Mark Billinghurst, and Joseph LaViola. 2019. A Comparison of Desktop and Augmented Reality Scenario Based Training Authoring Tools. In *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, New York, NY, USA, 339–350. https://doi.org/10.1109/ISMAR.2019.00032

[38] F. Vernier and L. Nigay. 2001. A Framework for the Combination and Characterization of Output Modalities. In *Interactive Systems Design, Specification, and Verification*, Philippe Palanque and Fabio Paternò (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 35–50.

[39] Tianyi Wang, Xun Qian, Fengming He, Xiyun Hu, Yuanzhi Cao, and Karthik Ramani. 2021. GesturAR: An Authoring System for Creating Freehand Interactive Augmented Reality Applications. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 552–567. https://doi.org/10.1145/3472749.3474769

[40] Zeyu Wang, Cuong Nguyen, Paul Asente, and Julie Dorsey. 2021. DistanciAR: Authoring Site-Specific Augmented Reality Experiences for Remote Environments. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, Article 411, 12 pages. https://doi.org/10.1145/3411764.3445552

[41] E. Yigitbas, J. Klauke, S. Gottschalk, and G. Engels. 2021. VREUD - An End-User Development Tool to Simplify the Creation of Interactive VR Scenes. In *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE Computer Society, Los Alamitos, CA, USA, 1–10. https://doi.org/10.1109/VL/HCC51201.2021.9576372

[42] Lei Zhang and Steve Oney. 2020. *FlowMatic: An Immersive Authoring Tool for Creating Interactive Scenes in Virtual Reality.* Association for Computing Machinery, New York, NY, USA, 342–353. https://doi.org/10.1145/3379337.3415824