VRGit: A Version Control System for Collaborative Content Creation in Virtual Reality

Lei Zhang
University of Michigan
Ann Arbor, MI, USA
raynez@umich.edu

Ashutosh Agrawal
University of Michigan
Ann Arbor, MI, USA
agarashu@umich.edu

Steve Oney
University of Michigan
Ann Arbor, MI, USA
soney@umich.edu

Anhong Guo
University of Michigan
Ann Arbor, MI, USA
anhong@umich.edu

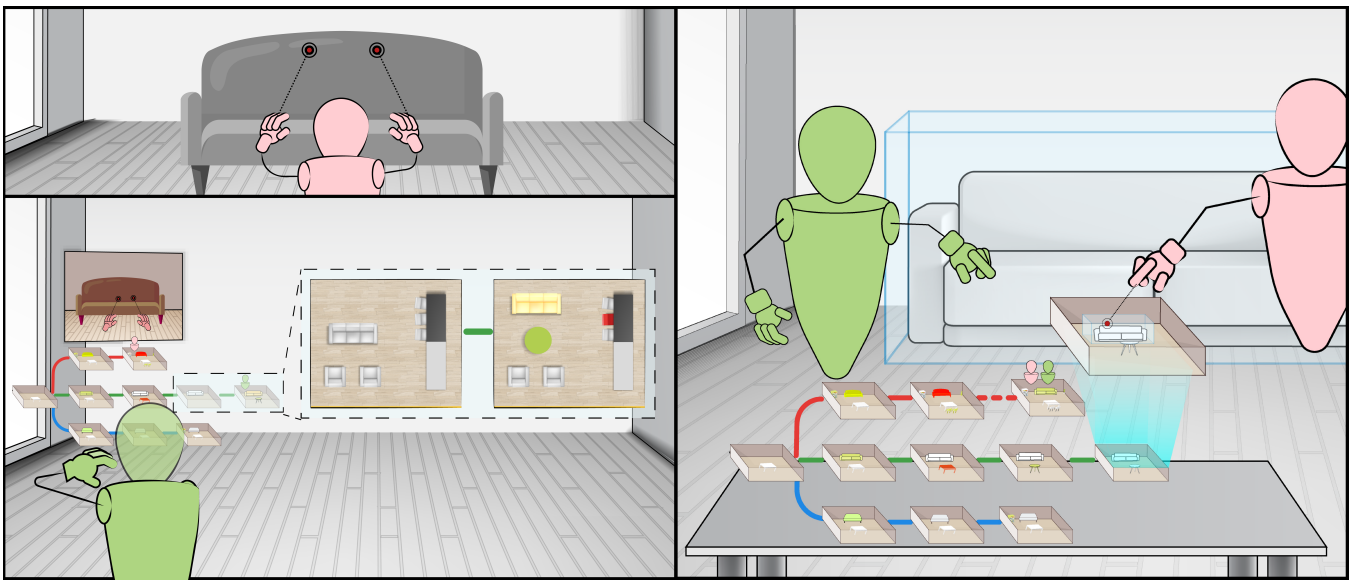


Figure 1: An illustration of VRGit. A History Graph (HG) that represents non-linear version history is anchored on the user's left arm, where each node is a 3D miniature of that version. Inside each miniature, objects are highlighted using color coding if they are changed compared to the previous version. Mini avatars are anchored in the HG to represent which version users are in. Users can also create portals to monitor other users' first-person views. A shared history visualization facilitates group discussion by anchoring the HG on a surface and allowing users to preview a version and reuse objects collaboratively.

ABSTRACT

Immersive authoring tools allow users to intuitively create and manipulate 3D scenes while immersed in Virtual Reality (VR). Collaboratively designing these scenes is a creative process that involves numerous edits, explorations of design alternatives, and frequent communication with collaborators. Version Control Systems (VCSs) help users achieve this by keeping track of the version history and creating a shared hub for communication. However, most VCSs are unsuitable for managing the version history of VR content because their underlying line differencing mechanism is designed for text and lacks the semantic information of 3D content; and the widely adopted commit model is designed for asynchronous collaboration rather than real-time awareness and communication in VR. We introduce VRGit, a new collaborative VCS that visualizes version history as a directed graph composed of 3D miniatures, and enables users to easily navigate versions, create branches, as well as preview and reuse versions directly in VR. Beyond individual uses, VRGit also facilitates synchronous collaboration in VR by providing awareness of users' activities and version history through portals and shared history visualizations. In a lab study with 14 participants (seven groups), we demonstrate that VRGit enables users to easily manage version history both individually and collaboratively in VR.

CHI '23, April 23–28, 2023, Hamburg, Germany

ACM ISBN 978-1-4503-9421-5/23/04…$15.00
https://doi.org/10.1145/3544548.3581136

CCS CONCEPTS

• **Human-centered computing** → **Interactive systems and tools**; **Collaborative and social computing systems and tools**.

KEYWORDS

Virtual Reality, Collaboration, Version Control System

ACM Reference Format:
Lei Zhang, Ashutosh Agrawal, Steve Oney, and Anhong Guo. 2023. VRGit: A Version Control System for Collaborative Content Creation in Virtual Reality. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3544548.3581136

1 INTRODUCTION

Virtual Reality (VR) can enable intuitive and compelling experiences for users to explore immersive, three-dimensional (3D) content. For instance, immersive authoring tools offer a "What You See Is What You Get" experience by allowing users to create, edit, and evaluate 3D content directly while immersed in VR [1, 25, 30, 47, 79, 80]. Research has also shown that VR provides an effective tool for users to evaluate ideas for 3D spatial content in multiple creative domains such as game design [75], architecture [27, 55], urban planning [68], and interior design [36]. Furthermore, as modern workforces in these domains are becoming diverse in terms of their skill sets and backgrounds, better collaborative content creation support is needed for coordination among various roles including designers, developers, and customers/end-users [5, 40].

Collaborative content creation is an iterative process in which users may perform numerous editing operations, explore various design alternatives, communicate with collaborators, and shift between individual and shared activities frequently [32, 33, 72]. Keeping track of version history in this process is important for providing the ability to revert to previous states if necessary. In addition, providing rich history-keeping can help users explore different design alternatives in the task of creative content production [64]. In collaborative settings, keeping track of version history is even more challenging since users may also need to maintain awareness of collaborators' activities. For example, imagine you are collaborating on designing a VR scene, and you would like to explore a design variant of the current scene without interfering with your collaborators' design. Moreover, when you and your collaborators are working on different design variants, you would like to know which versions your collaborators are working on and communicate ideas with them. If all the versions of the scene, including different branches that collaborators are working on, are preserved and visualized in VR, you could easily "travel" between versions and communicate with your collaborators across versions or branches. While existing systems enable compelling experiences for creating and manipulating 3D content in VR, most only enable basic history-keeping (e.g., a linear timeline) for single users in VR.

Version Control Systems (VCSs) have been used widely for keeping track of version history of digital content among collaborators. Most current VCSs, however, are designed for text rather than spatial data such as 3D scenes. Research in VCS for 3D modeling has explored some effective mechanisms for one or two features of a VCS such as comparing and merging scenes or models on 2D displays [9, 10, 18, 20, 63], but still lacks knowledge regarding how to enable multiple users to track version history without breaking the fluidity of immersive authoring. On the other hand, although collaboration systems in VR have long been an exploration in the area of HCI and CSCW [35, 58, 59, 78], most of them lack version control capabilities and keep only one version of 3D scenes at a time for all users. In this work, we aim to explore providing visualization and interactions of version history that are appropriate for collaborative immersive environments. We provide a complete, standalone design and implementation for version control in VR in order to avoid breaking the immersion and the workflow of collaborative content creation, to leverage intuitive interactions that VR affords, and to harness people's spatial skills for understanding and navigating 3D environments. We also take a different approach by enabling collaborators to stay in different versions of 3D scenes and explore supporting awareness and communication among collaborators across different versions.

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

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

2 RELATED WORK

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

2.1 Version Control Systems for 3D Scenes

A Version Control System, also known as a Revision Control System, enables users to keep historical versions of digital content. VCSs such as Git [29] and Subversion [3] have been popular in the domain of software engineering to help developers keep track of the history of source code by committing changes along with text messages that describe the changes. Most VCSs also support asynchronous collaboration among multiple developers at remote locations by allowing them to manually create and merge branches. However, most existing VCSs are unsuitable for tracking and understanding changes of 3D scenes because the underlying line differencing mechanism is designed for tracking changes of text files and thus lacks high-level semantic information of spatial data such as two-dimensional (2D) images and 3D scenes. Recent work, primarily from the Computer Graphics community, has thus explored techniques for tracking changes in media files such as 2D images [13] and 3D scenes [9, 10, 18, 20, 21, 63], which can be categorized into two approaches: state-based and operation-based.

Table 1: Summary of prior VCSs and VRGit. VRGit contributes to a full VCS in collaborative immersive environments.

System	Content Type	Version Control Features					State/Operation	Real-time Collaboration	Immersive
		Navigation$^+$	Commit	Branch	Diff	Merge			
Git [29]	Text	✓	✓	✓	✓	✓	State-based		
Chen et al. [13]	Image	✓	✓	✓	✓	✓	Operation-based		
Dobõs et al. [21]	3D Scene	✓	✓	✓	✓	✓	State-based		
MeshGit [18]	3D Scene				✓	✓	State-based		
SceneGit [10]	3D Scene				✓	✓	State-based		
MeshHisto [63]	3D Scene				✓	✓	Operation-based	✓	
CSculpt [9]	3D Scene				✓	✓	Operation-based	✓	
Spacetime [78]	3D Scene				✓*		State-based	✓	✓
Lilija et al. [45]	Spatial Recordings	✓					State-based		✓
VRGit (this work)	3D Scene	✓	✓	✓	✓*	✓**	Operation-based	✓	✓

Navigation$^+$: History visualization and navigation.
✓*: The system allows users to compare different versions instead of calculating the differences.
✓**: The system allows users to reuse content from different versions instead of merging all changes.

State-based approaches aim to build effective mechanisms that can automatically derive changes by comparing two states, e.g. a version and its successor, after the changes occur. Prior work focusing on state-based approaches has strived to derive changes at different levels of granularity [10, 18, 20, 21]. For example, Doboš and Steed version 3D assets at a coarse granularity of individual nodes of a scene graph such as individual meshes [20, 21]. SceneGit can derive changes at a finer granularity of vertices and faces [10]. In the domain of drawing, techniques such as object-oriented drawing can also preserve states of individual attributes and allow users to revert to previous states of an attribute without interfering with other attributes [77]. The other approach is operation-based, which records changes while they occur. This approach typically records editing operations that users make, and then applies the operations to a state to transform it to the successor state [9, 13, 63]. For example, MeshHisto stores and transmits mesh difference by encoding them as sequences of primitive editing operations [63]. In our work, VRGit uses operation-based change tracking since it is more precise and efficient in determining the difference between two states and provides functionality such as replay and undo. It has also been applied to prior content creation tools such as 3D modelling [63] and sculpting [9].

Our work contributes to existing literature in VCSs for 3D scenes by introducing a full VCS in *collaborative*, and *immersive* environments (Table 1). While prior work has been focused on one or two features of a VCS such as diffing and merging and has targeted VCSs on 2D screens, VRGit aims to explore novel visualization and interactions of a VCS and provide real-time workspace awareness in collaborative content creation in VR.

2.2 Graphical History Visualization

Enabling intuitive graphical visualization and interactions for version or operation history has long been an area of exploration in HCI. Prior work has explored visualization of history using representations such as layers of operations [49], snapshots of before-and-after states [43], and timeline views of history [62]. Later work has then built on these representations and explored techniques that enable users to better understand and interact with operation history. For example, Klemmer et al. built upon snapshots of states of collaborative web editing sessions and embedded non-linear branches in the timeline view [38]. Nakamura and Igarashi captured the Graphical User Interface (GUI) input and output history of graphical documents and visualized the snapshots with annotations of detailed operations [50]. Chronicle instead captures the video history of graphical documents and provides users with a set of probes to filter the revision history [31]. More recently, Chen et al. explored using a Directed Acyclic Graph (DAG) for versioning image editing operations [13]. However, all the above systems for visualizing and interacting with version history are designed for text or 2D content such as paintings and images.

Most graphical history representations for 3D scenes today have primarily focused on viewing and interacting through 2D displays. For example, commercial Computer-Aided Design tools such as Autodesk Maya or Vistrails are able to record modelling history and provide a list of operation history in the editor. Another line of research in this space is focused on interactive summary of long sequences of editing operations. For example, MeshFlow [17] and 3DFlow [19] are proposed to summarize the history of mesh editing by clustering editing operations. Closer to our work that visualizes history in VR, Lilija et al. introduced techniques of visualizing objects' trajectory in 3D scenes and allowing users to view the history of spatial recordings in VR [45], though not in the context of a VCS. Our work expands on prior work to explore graphical representation and interactions of non-linear history (i.e. branching) in a VCS for immersive VR authoring.

2.3 Collaborative Virtual Environments

Researchers have acknowledged the importance of designing real-time collaborative systems that support workspace awareness, i.e. understanding of other collaborators' interaction with the shared workspace [32]. Prior work has explored various techniques for supporting awareness of other users in collaborative virtual environments such as the use of gaze [59], gestures [52, 58, 76], and pointers [24]. Recent advances of the underlying sensing technologies have also allowed for capturing and rendering full bodies of users via 2D projection (e.g. Room2Room [56]) or 3D hologram

(e.g. Holoportation [53]). Beyond the awareness of other users, workspace awareness also involves understanding of collaborators' workspace context. To achieve that, sharing views of the workspace has shown to be an effective technique [26, 28, 51, 73]. For instance, Fraser et al. proposed using peripheral views to support peripheral awareness of other users in collaborative virtual environments [26].

A common limitation of the above techniques is that they are designed for providing awareness when there is only one version of 3D scenes and all users are virtually co-located and therefore visible to each other in that version. Highly relevant to our work, Spacetime proposes the concept of parallel objects that allow users to create parallel versions of the same object similar to branching [78]. However, their technique still requires users to land on one final version and renders all parallel designs using different levels of transparency in the same version. In this work, we take a different approach that allows users to manually branch into different variants of the 3D scenes and make changes in those branches while still maintaining workspace awareness when they are located in different versions or design variants (i.e., branches) of 3D scenes.

2.4 VR Visualization and Interaction Techniques

Our work also builds on prior visualization and interaction techniques for object manipulation and navigation in VR. Prior work in immersive authoring tools has proposed intuitive interaction techniques that allow users to build 3D scenes through direct manipulation and to leverage their spatial reasoning skills [1, 30, 47, 48, 60, 80]. Other research has proposed several techniques for interaction and navigation in 3D scenes of large distances [8, 42, 46, 57, 61, 67]. For instance, Mackinlay et al. propose the using teleportation to navigate large virtual workspaces [46]. Kunert et al. use photos of 3D scenes as portals that allow users to navigate in space and time [42]. To interact with objects at a large distance, "go-go" interaction uses the metaphor of interactively growing the user's arm to interact with distant objects in a virtual environment [61]. Stoakley et al. introduced the concept of World in Miniature (WIM), which enables both navigation and interaction in a large VR scene. A WIM represents the virtual environment and allows users to manipulate objects offered by the miniature, or rapidly teleport in the virtual environment by selecting locations directly in the miniature [67]. It also has the benefit of allowing users to see a preview of the immersive virtual environment without having to travel back and forth between different views.

In this work, we contribute to the literature by using the techniques of WIM and portals in the context of version control. More specifically, we extend the concept of World in Miniature by using them as nodes in the history graph for previewing snapshots and changes of different versions and reusing objects in different versions. We also extend the concept of portals to communicating and sharing views between collaborators in the VCS.

3 VRGIT

VRGit is a VCS for VR that enables users to keep track of multiple versions of 3D scenes, to create and navigate different branches, and to preview and reuse content from different versions. Beyond supporting individual uses, VRGit also supports real-time workspace awareness of users' activities and version history by integrating synchronous communication and enabling a shared history visualization. To instantiate the visualization, interaction, and collaboration design of VRGit, we first build an immersive authoring system that enables users to create and manipulate 3D scenes directly in VR. We choose an example of designing the floor plan of an apartment for evaluation with end-users, because it has been a key VR application since it requires users' spatial capabilities and has been used to evaluate prior collaborative VR systems [36, 54]. We also believe the design concepts behind VRGit are generalizable to other application domains (e.g., game scenes design) as well, because the immersive authoring operations (e.g., manipulation of 3D objects), version control mechanism (e.g., commit and branching), and collaborative support (e.g., portals and shared visualization) are independent of the task context.

Figure 2: The immersive authoring environment. A menu is always attached to the left controller. Users can select pre-made furniture models of different categories and place them in the VR scene.

3.1 Immersive Authoring Environment

We build an immersive authoring environment that allows users to design the spatial layout of an apartment, as a foundation for instantiating the concepts of VRGit. Users can place and manipulate pre-made furniture models in an empty apartment, as seen in Fig. 2. Our system is scoped to authoring VR scenes at a fixed scale (e.g., city scales in urban planning or room scales in interior design) to better focus on the challenges of visualizing and interacting with version histories in VR, and keeping workspace awareness among collaborators across different versions.

3.2 Version Control System

VRGit supports full version control of the 3D scenes during the editing process. We use an operation-based history model that records users' edit operations and then applies the operations to a state to transform it to the successor state. We use this model because *(i)* it has been utilized in prior editing tools such as desktop-based 3D modeling [63] and 2D images [13], and *(ii)* it can be more precise and efficient in determining the difference between two states [63] and provides functionalities such as replay and undo [13]. Our current system supports common immersive authoring operations including creation, transformation, and deletion. We use a Directed Acyclic Graph (DAG) as the underlying data structure where each

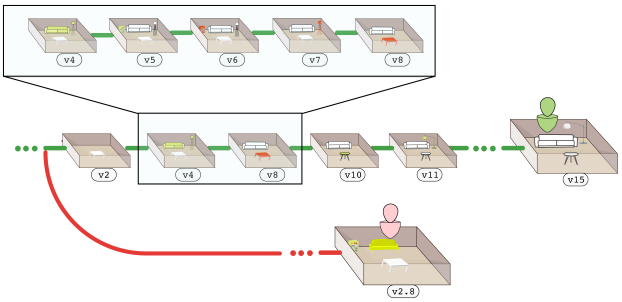
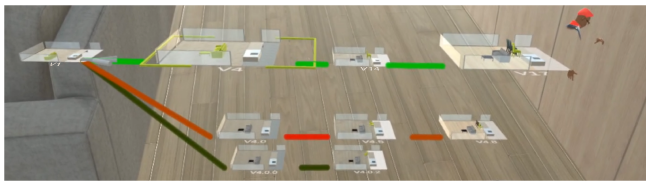


Figure 3: The summarized version history. The upper area refers to part of the DAG that generate a node whenever there is a new operation. The middle area shows part of the summarized HG where *V4-8* are clustered based on operation dependencies. The bottom area shows the constraint of workspace awareness, where a version that another user is working in will always be shown.

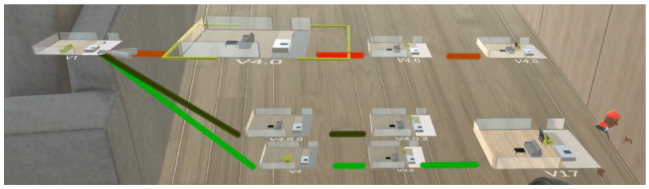
node represents an editing operation with its relevant parameters and each edge along with its direction represents the temporal order between two operations.

3.2.1 History Visualization and Navigation. We visualize a directed graph on the users' left arm to represent version history as a History Graph (HG). Each node is visualized using a miniature of the version state and each edge represents the temporal order between two versions. We also show the version number in text in each node. Inside each miniature, we highlight the difference with its prior version by changing the material of the furniture with color coding: we use green to represent an added object, yellow to represent transformation of an existing object, and red to represent a deleted object. Because of the potential complexity and large size of the underlying DAG, the HG only shows users the summarized version history, as shown in Fig. 3. Our summarization techniques take into account three parameters: *(i)* operation dependencies, *(ii)* spatial constraints, and *(iii)* workspace awareness. We use a simple timeout mechanism to determine operation dependencies, which generates a new node in the history graph after the user has been idle for a specified amount of time (10 seconds when working alone, 15 seconds when working collaboratively—defaults that we found to work anecdotally). Spatial constraints allow us to determine the number of miniatures and branches to display given the amount of available space in a visualization anchor (e.g. users' arms or a tabletop). VRGit also always shows which versions users' collaborators are working in, as a way to improve workspace awareness. Users can select the previous or next version in the history graph by pushing the thumbstick on the left controller horizontally. A cursor of a yellow square is then shown around the miniature when a version is selected. To enter a version, users can select the version and press a button on the controller. The layout of the environment will then change to the state of that version. When there are multiple branches at a node (shown as parallel siblings in the HG), users can also switch branches by pushing the thumbstick vertically.

3.2.2 Commits and Branching. In VRGit, commits (checkpoints in the repository) are made automatically by the system whenever the



(a) Visualization of three branches.



(b) Visualization after switching to the next branch.

Figure 4: Screenshots of the History Graph (HG) that visualizes multiple branches. The user stays in the version of *V17*. In 4a, the user is selecting *V4* which belongs to the branch highlighted in light green. Users can switch to other branches. After switching branches (4b), the user is selecting *V4.0* which belongs to the branch highlighted in dark red.

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

VRGit enables visualization of multiple branches and intuitive interactions for creating, updating, and navigating different branches in the visualization, as shown in Fig. 4. Creative tasks such as content creation usually involve numerous trial-and-error experiments and design variants (branches) [33, 72] and the ability to keep multiple branches has shown to be an important building block of creativity support tools [64]. In our system, users can easily create a branch based on a historical version by first entering that version and then pressing a button on the controller. The system will then create a copy of the node that represents the historical version, and append the copy to its parent node in the underlying DAG. The HG will then be updated by laying out the branches in a circular path that takes advantage of depth afforded by VR displays. The user will be automatically switched to the new branch once it is created and can update the new branch modifying the 3D scene. Users can use the thumbstick to navigate different existing branches as mentioned above.

3.2.3 Previewing and Reusing. A preview allows users to easily examine the state of a historical version without actively entering that version, typically known as "snapshots" or "thumbnails" in 2D graphical editing tools [43]. In our system, as the miniatures in the HG can be small for inspection depending on the anchor of the HG, we enable users to preview of a version in the HG by showing an expanded miniature of the version. Users can open a preview of a version by using the raycast to aim at the version and

(a) Preview. (b) Reuse.

Figure 5: Screenshots of previewing and reusing in VRGit. In 5a, the user is in *V14* and opening a preview of *V5.11*. In 5b, the user reuses the chair in the preview by aiming at it using the raycast shooting from the controller and pressing a button on the controller. The chair then appears in the environment and the system automatically creates a commit that brings the user to *V15*.

pressing a button on the controller, as shown in Fig. 5a. They can also resize the preview to better inspect changes in the version. Multiple previews can exist at the same time and users can directly manipulate the preview for the convenience of inspection.

Another novel functionality that previews afford is reusing spatial design from one or more previews. In conventional VCSs, reusing or combining data between versions is done via *merging*, which applies all changes of one branch to another. Along this line, prior work has also proposed mechanisms for merging changes and resolving conflicts of two 3D scenes [10, 18, 20]. However, merging is typically limited to fusing all changes between only two versions at a time. Content creation, on the other hand, is often an open-ended design process that is subject to unexpected changes of directions or goals [15]. It is therefore common for users conducting creative tasks to explore different design variants by selectively mixing-and-matching changes from multiple sources [11, 12, 44, 69]. For instance, imagine an architect who wants to mix-and-match designs of the balcony from one version, the furniture from a second version, and decoration from a third version. It would be useful if they can open previews of the three versions and directly reuse these specific parts of the 3D scene. Therefore, instead of fusing all changes between two versions at a time, our system allows users to reuse objects from multiple versions. In VRGit, users can reuse spatial design of different versions and selecting the target item in the previews, as shown in Fig. 5b. The selected furniture will then appear with the same transformations (i.e., position, rotation, and scale) in the user's current version. To incorporate designs from multiple versions, users can create multiple previews and reuse objects in those previews accordingly.

3.3 Collaboration in VRGit

Collaboration is an important component in most VCSs. For instance, VCSs such as Git and Subversion are designed for multiple developers at remote locations to collaborate with each other, by allowing them to sync changes (e.g. commits or branches) to the repositories through a central server. The collaboration in such VCSs is asynchronous and users typically work in separate editing environments. Numerous synchronous 3D scene editing tools exist (e.g. [9, 63]), but they do not support active branching or navigating history in their VCSs. In VRGit, users can collaboratively author the 3D scene when they are in the same version, analogous to a synchronous editing tool. They are able to see each other's edits and avatars, point to objects with raycasts, and talk to each other via audio communication. VRGit also allows users to create branches and navigate to different versions, in order to explore different design variants without interfering with each other. In this scenario, they can work on their own branches but are not able to see each other's avatars in the environment, analogous to an asynchronous editing tool. In all, collaborators can easily separate or reconcile by navigating and branching into the same or different versions.

In this work, we address the unique challenges of supporting communication and workspace awareness in immersive authoring through a VCS. We consider two primary scenarios: *(i)* when users are working in different versions and *(ii)* when users are working in the same version. In most existing VCSs, being in different versions means users are working in separate workspaces and there is little support or need for real-time workspace awareness since there is no presence of users or activities in the workspace [71]. However, as in VRGit, users can easily switch and work in different versions or branches, so there should be a consistent presence of collaborators and their activities as well as a convenient way to communicate in order to maintain workspace awareness. We incorporate mini-avatars in the HG that indicate in which version (*where*) collaborators are located. We also integrate the concept of portals and shared history visualizations in our VCS that help users understand *what* collaborators are doing and communicate with each other when they are working in different and the same version respectively. We detail the design of these two features below.

3.3.1 Portals. In collaborative authoring, it is common for collaborators to adopt a 'multi-synchronous' collaboration styles in which they work simultaneously in isolation and subsequently integrate their contribution [22]. VRGit enables this by allowing users to create and work in different branches, and combine their work together by reusing designs from multiple branches. An important aspect in this process is how to enable communication between collaborators and awareness of their activities in order to ensure common ground [14] and to avoid redundant work [37]. VRGit addresses this by integrating portals into the HG that allow users to easily monitor and communicate with each other when they work in different branches. Portals are 2D video streams from collaborators' first-person view, which have been shown to be useful in understanding collaborator's activities [2, 39]. Users can create a portal of another user by using the raycast from the controller to aim at the mini-avatar appearing in the HG and pressing a button on the controller. A 2D plane showing the target user's first-person view will be created next to the mini-avatar, as shown in Fig. 1 (left). Users can directly manipulate and place the portal.

3.3.2 Shared History Visualization. In VRGit, shared history visualization is designed to facilitate discussions about multiple versions (e.g., to compare features) or about the edit history itself, as shown in Fig. 6. A unique challenge for the VCS under this setting is how

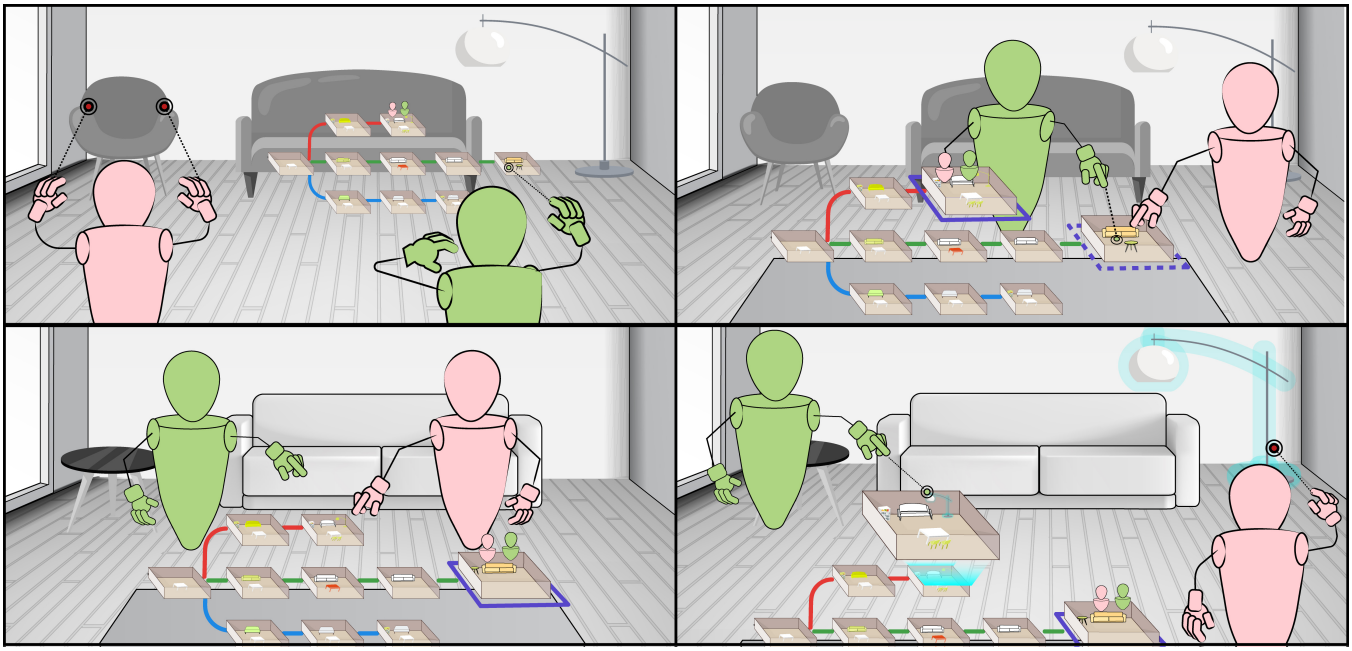


Figure 6: The workflow of shared history visualizations. When two users are working in the same version, one user can start a shared history by pointing at the HG on the left arm and pressing a button (top left). The shared history visualization will then be anchored on the table and the sharer can select a version (highlighted in dashed line) for all users (top right). They can then enter that version collaboratively (bottom left). Finally, all users can collaboratively preview and reuse the lamp from another version (bottom right).

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

3.4 System Implementation

VRGit is implemented using Unity 2020.2.7 and runs on Oculus Quest or Rift headsets. The overall architecture of our system is shown in Fig. 7. We use the Photon Voice[1] plugin in Unity to enable voice chat among users. The rest of our functionalities are mainly synced through a Firebase server.[2] More specifically, the application encodes the animation of avatars that is tracked in Oculus in binary and updates the document linked to the user ID on Firebase. In VRGit, operations of each user are synced through the operation document on Firebase and used to update the local copies of HGs in each application. We currently support five operation types: creation, transformation, deletion, entering, and branching. The first three are immersive authoring operations and the rest are manual operations in the HG that need to be updated. When users start a portal to share their first-person views and communicate with each other, we encode the video streams in binary using WebRTC[3] and sync the streams through Firebase. Finally, while all of the above functionalities are synced in both directions, the shared history visualization is in one direction. When one user becomes the sharer by starting a shared history visualization, all the operations from the sharer such as navigating history, switching branches, and previewing and reusing are sent to the server. All the sharees then pull those shared history operations and update their shared history visualization based on the operations.

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

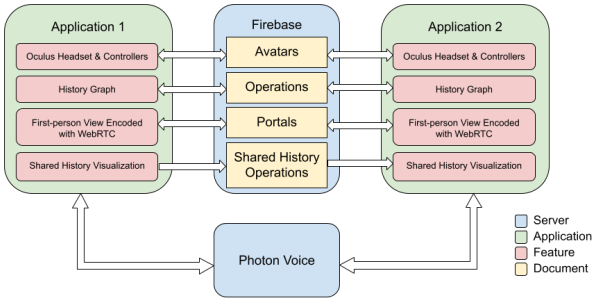


Figure 7: The system architecture of two applications running VRGit. VRGit uses two servers: Photon Voice and Google Firebase. All applications write and read audio data from the Photon Voice server to sync voice communication. Applications sync the movement of users' avatars by writing and reading data of Oculus headset and controllers from the *Avatars* document on Firebase. Similarly, applications sync their version control operations through the *Operations* document and their video data through the *Portals* document. Finally, for shared history visualization, only the sharer (i.e., App 1) writes data to the *Shared History Operations* document and only the receiver (i.e., App 2) reads data from the document.

4 EVALUATION

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

4.1 Participants

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

4.2 Procedure

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

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

At the end of the collaboration task, participants filled out another 5-minute survey that focuses on our collaboration features by examining users' workspace awareness and their communication experience. Finally, we conducted a semi-structured interview with the participants individually that asked about the benefits and challenges of using VRGit. The interview lasted about 15 minutes. We aggregated all the survey data, transcribed and coded the recordings of interview.

4.3 Task

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

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

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

5 RESULTS

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

5.1 Usability and Utility of VRGit

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

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

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

5.1.3 Previewing and reusing were useful for comparison and efficiency. Participants generally found it easy (avg=5.2, sd=1.7) and useful (avg=6.3, sd=0.6) to see the preview of a version. Previews are suitable for closer inspection than the nodes, allowing users to control the viewing angle by direct manipulation and to get a holistic understanding of the floor plan. This has been mentioned by participants in comparison to portals, in which users did not have control over the views and could only see part of the floor plan. *"I feel like for me it was a lot more useful if I pulled up a preview of their version rather than looking through the portal because I have the bird-eye view and I can move it [the preview] around."* (P7a) Opening previews of multiple versions were also useful for comparison. *"I don't know if it is a feature but if we were able to make previews of multiple versions at one time and put them next to each other. I think it would be pretty useful to compare and get the bird-eye view of all of them next to each other."* (P2a) Participants also thought it was easy (avg=5.6, sd=1.4) and useful (avg=6.5, sd=0.7) to reuse objects in the preview. Reusing objects can save time because accurate object manipulation is time-consuming in VR [34] and reusing objects can put those objects at the same location and rotation. One participant also reported to prefer reusing to merging in conventional VCSs such as Git because it allowed them to utilize part of the changes.

"I like it better than, say, like Git or Google Doc because you could pull it [the preview] up and then take in just the pieces that you wanted, versus like Git you have to kind of merge all the updates." (P2a)

5.2 Communication and Awareness in Collaboration Tasks

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

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

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

6 DISCUSSION AND FUTURE WORK

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

6.1 Lowering the Effort for Using VCS in VR

As they are originated from managing source code among developers [74], existing VCSs could require advanced knowledge and skills of both using the system (e.g., shell commands) and interpreting the visualization (e.g., differences between two versions and structures of non-linear version history). Content creation in VR, however, is targeted on a diverse population including non-technical users such as designers and customers/end-users [5, 40]. In our research, we aimed to reduce the barrier of entry to VCSs by designing intuitive visualization and interaction of version history in VR, and found that even people without prior programming experience were able to use VRGit to manage the versions history of their work. This demonstrates the benefits of building graphical representations in VR that end-users are familiar with for tracking version history. This also aligns with the spirit of What-You-See-Is-What-You-Get in immersive authoring research, where many tools and techniques were explored to lower the floor for novices to create content directly in VR [47, 80]. Future work may also draw inspiration from past research that has investigated intuitive visualization and interactions to lower the barrier for using VCS, such as by leveraging people's spatial abilities using the virtual body resizing technique [41] to enable intuitive navigation of version histories. In addition, VRGit only uses controllers as input, which has limited capabilities, making it effortful for users to recall which buttons to press. Future work might also draw inspiration from past research that has investigated richer input modalities in VR such as gestures [4] and cross-device interactions [23, 70] for more precise control and lower mental or physical effort.

6.2 Providing Efficient Comparison and Fusion of Design Variants

We found that VRGit enabled users to easily compare different versions through a combination of visualizing variants using branches, highlighting changes in colors, and previewing versions in miniatures. This extends prior work on VCS for 3D scenes by seamlessly providing the comparison in the content creation process without explicit commands such as *diff*. Our study also suggests that the design of reusing content in VRGit was helpful in quickly copying objects from different versions and fusing ideas of multiple collaborators. This aligns with prior work on creativity support where enabling easy exploration of possible creative solutions such as mix-and-match is a key component [64]. This is different from

prior work on VCS that uses the *merge* command to combine the whole scenes of two branches instead of part of the scene [18]. Both approaches could become inefficient when users have to manually deal with a large number of content that they do or do not want to fuse (e.g., manually resolving conflicts after merging). Therefore, future work could explore more efficient ways of fusing multiple design variants, such as providing suggestions based on other design variants during the creation process, or more efficient group selection techniques to facilitate fusing.

6.3 Designing for Larger-scale Version History

VRGit introduces summarization techniques that can cluster operations and reduce the visual cluttering by clipping the HG based on the constraints. This was sufficient for lab study tasks such as designing the spatial layout of an apartment. However, the HG can become complex and hard to navigate when creating more complex 3D scenes. Future work could investigate techniques that allow visualization and interaction with the HG at larger scales. For instance, there are techniques that allow users to cluster operations manually [50]. One can also consider employing more sophisticated ways of automatic clustering operations or interactions, e.g., by leveraging techniques in interactive summarization of videos [6, 7] and 3D models [17, 19]. While the former is based on image analysis and the latter is based on geometry analysis of single objects, future work could extend this line of work by exploring interactive summarization techniques based on content creation operations clustering. To navigate a large scale HG, one possible solution is to incorporate the metaphor of WIM again where the user can anchor a coarser resolution of HG on the non-dominant hand and a higher resolution of HG in the world space. The user can then navigate the HG at large steps by manipulating the HG on the hand and navigate at smaller steps by manipulating the HG in the world space.

6.4 Leveraging System Support for Richer Workspace Awareness

Our results showed that various channels of workspace awareness were helpful for a VCS to support fluid collaboration experiences. While portals allowed users to understand their collaborators' activities and communicate ideas, when they are working in different versions, participants still sometimes preferred using previews or going directly to the collaborator's version for communication. This is partially because portals were only 2D video streams and did not offer the spatial context of the layout that the other collaborators were working on. In addition, users did not have control over the viewing angle through the portal, which introduced friction when users were looking at their collaborator's environment. Future work could investigate ways to fuse different views of the collaborators' environment in order to support better workspace awareness (e.g. [39]).

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

6.5 Integrating Version Control in the VR Content Creation Process

Shneiderman suggests that rich history-keeping is an essential feature of creativity support tools [64], and more recently, Sterman et al. [66] also suggest that version control for creative domains should be designed based on the needs of creative processes as creative practitioners in these domains could prioritize different values over efficiency and fidelity as those in the software engineering domain. In line with prior research, our study suggests that our design of history visualization and interaction that is appropriate for VR authoring can help users revert to prior versions, compare and explore design variants, and communicate with collaborators. Besides designing VCSs that can suit the needs of creative tasks, our results also suggest that the design of VRGit can help shape collaboration in content creation such as understanding collaborators' activities without leaving their workspace and sparking design suggestions during discussion based on shared visualization. Future researchers and designers should thus consider users' creative and collaborative needs that are specific to the workflow of collaborative VR content creation when designing VCSs. For example, future research could further explore a hybrid VCS that can bridge different devices and platforms as many researchers have explored cross-device and cross-platform development of VR and Augmented Reality (AR) experiences [40, 65]. More specifically, numerous aspects of VR content creation might take place outside of the immersive environments, from 3D modeling to software engineering, and debugging. Future research could thus explore how to integrate the design of VRGit into current version control practices of the above aspects.

6.6 Limitations

Though our findings showed that our VCS is useful for collaborative content creation tasks in VR, our work has several limitations. First, we ran an exploratory lab study of the system with teams of three users for no more than two hours. The way that participants performed the tasks in the lab setting could be different from that in the real-world setting. It is also unclear how well it works for larger collaboration teams, which might reveal additional challenges for collaboration awareness and scalability. In addition, although we did not observe any specific novelty effect during the study, it is possible that it played a role, and a longitudinal study would help us understand how people would use VRGit in practice. A participant response bias could also exist when the participants thought the system was developed by the researchers [16].

Second, we mainly evaluated the usability and utility of VRGit in the context of interior design. Future work could further evaluate other parameters such as correctness, consistency, and scalability and potentially compare VRGit with a control condition where users are provided with the same tasks without using VRGit. It is also worth investigating how different task contexts and populations

could affect the usage of VRGit. For example, future work could evaluate the usage of VRGit in contexts such as game design and architecture, and investigate how people's relationships (e.g. friends, colleagues) and skills (e.g. high VCS literacy but low VR literacy) could affect their usage of VRGit.

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

7 CONCLUSION

In this paper, we presented a new Version Control System (VRGit) for collaborative content creation in VR. VRGit enables novel visualization and interactions of History Graphs in VR that allows users to easily view and navigate version history, create branches, preview and reusing objects from multiple versions. Our system also facilitates communication and awareness between collaborators and the version history, whether users are working in the same version or different versions. Through an exploratory lab study, we found that our system enabled users to easily manage non-linear version histories, communicate with collaborators, and maintain workspace awareness in VR. We provide insight for future research and design around building VCSs for collaboration in VR, including techniques for scaling up to long version history and providing more channels of workspace awareness.

ACKNOWLEDGMENTS

We would like to thank Yan Chen for providing early suggestions that helped shape our research direction. We are also grateful to Zach Behrman, Yufei Quan, and Yifei Wang for their help in system implementation, and Shwetha Rajaram for her feedback on our paper submission. Finally, we would like to thank our participants and our reviewers for their time and effort.

REFERENCES

[1] Adobe. 2021. *Adobe Medium.* https://www.adobe.com/products/medium.html
[2] Judith Amores, Xavier Benavides, and Pattie Maes. 2015. Showme: A remote collaboration system that supports immersive gestural communication. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. 1343–1348.
[3] Apache. 2022. *Apache Subversion.* https://subversion.apache.org/
[4] 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*. 463–477.
[5] 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*. 1–13.
[6] Jackie Assa, Yaron Caspi, and Daniel Cohen-Or. 2005. Action synopsis: pose selection and illustration. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 667–676.
[7] Connelly Barnes, Dan B Goldman, Eli Shechtman, and Adam Finkelstein. 2010. Video tapestries with continuous temporal zoom. In *ACM SIGGRAPH 2010 papers*. 1–9.
[8] Evren Bozgeyikli, Andrew Raij, Srinivas Katkoori, and Rajiv Dubey. 2016. Point & teleport locomotion technique for virtual reality. In *Proceedings of the 2016 annual symposium on computer-human interaction in play*. 205–216.
[9] Claudio Calabrese, Gabriele Salvati, Marco Tarini, and Fabio Pellacini. 2016. cSculpt: a system for collaborative sculpting. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–8.
[10] Edoardo Carra and Fabio Pellacini. 2019. SceneGit: a practical system for diffing and merging 3D environments. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–15.
[11] Siddhartha Chaudhuri, Evangelos Kalogerakis, Leonidas Guibas, and Vladlen Koltun. 2011. Probabilistic reasoning for assembly-based 3D modeling. In *ACM SIGGRAPH 2011 papers*. 1–10.
[12] Siddhartha Chaudhuri and Vladlen Koltun. 2010. Data-driven suggestions for creativity support in 3D modeling. In *ACM SIGGRAPH Asia 2010 papers*. 1–10.
[13] Hsiang-Ting Chen, Li-Yi Wei, and Chun-Fa Chang. 2011. Nonlinear revision control for images. *ACM Transactions on Graphics (TOG)* 30, 4 (2011), 1–10.
[14] Herbert H Clark and Susan E Brennan. 1991. Grounding in communication. (1991).
[15] Robin George Collingwood. 1958. *The principles of art*. Vol. 11. Oxford University Press.
[16] Nicola Dell, Vidya Vaidyanathan, Indrani Medhi, Edward Cutrell, and William Thies. 2012. " Yours is better!" participant response bias in HCI. In *Proceedings of the sigchi conference on human factors in computing systems*. 1321–1330.
[17] Jonathan D Denning, William B Kerr, and Fabio Pellacini. 2011. Meshflow: interactive visualization of mesh construction sequences. In *ACM SIGGRAPH 2011 papers*. 1–8.
[18] Jonathan D Denning and Fabio Pellacini. 2013. Meshgit: Diffing and merging meshes for polygonal modeling. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–10.
[19] Jonathan D Denning, Valentina Tibaldo, and Fabio Pellacini. 2015. 3dflow: Continuous summarization of mesh editing workflows. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–9.
[20] Jozef Doboš and Anthony Steed. 2012. 3D Diff: an interactive approach to mesh differencing and conflict resolution. In *SIGGRAPH Asia 2012 Technical Briefs*. 1–4.
[21] Jozef Doboš and Anthony Steed. 2012. 3D revision control framework. In *Proceedings of the 17th International Conference on 3D Web Technology*. 121–129.
[22] Paul Dourish. 1995. The parting of the ways: Divergence, data management and collaborative work. In *Proceedings of the Fourth European Conference on Computer-Supported Cooperative Work ECSCW'95*. Springer, 215–230.
[23] Tobias Drey, Jan Gugenheimer, Julian Karlbauer, Maximilian Milo, and Enrico Rukzio. 2020. Vrsketchin: Exploring the design space of pen and tablet interaction for 3d sketching in virtual reality. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–14.
[24] Thierry Duval, Thi Thuong Huyen Nguyen, Cédric Fleury, Alain Chauffaut, Georges Dumont, and Valérie Gouranton. 2014. Improving awareness for 3D virtual collaboration by embedding the features of users' physical environments and by augmenting interaction tools with cognitive feedback cues. *Journal on Multimodal User Interfaces* 8, 2 (2014), 187–197.
[25] 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 *Proceedings of the 43rd Graphics Interface Conference*. 156–162.
[26] Mike Fraser, Steve Benford, Jon Hindmarsh, and Christian Heath. 1999. Supporting awareness and interaction through collaborative virtual interfaces. In *Proceedings of the 12th annual ACM symposium on User interface software and technology*. 27–36.
[27] Peter Frost and Peter Warren. 2000. Virtual reality used in a collaborative architectural design process. In *2000 IEEE Conference on Information Visualization. An International Conference on Computer Visualization and Graphics*. IEEE, 568–573.
[28] William W Gaver, Abigail Sellen, Christian Heath, and Paul Luff. 1993. One is not enough: Multiple views in a media space. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*. 335–341.
[29] Git. 2022. *Git.* https://git-scm.com/
[30] Google. 2016. *Google Tilt Brush.* https://www.tiltbrush.com/
[31] Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2010. Chronicle: capture, exploration, and playback of document workflow histories. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. 143–152.
[32] Carl Gutwin and Saul Greenberg. 2002. A descriptive framework of workspace awareness for real-time groupware. *Computer Supported Cooperative Work (CSCW)* 11, 3 (2002), 411–446.
[33] Björn Hartmann, Loren Yu, Abel Allison, Yeonsoo Yang, and Scott R Klemmer. 2008. Design as exploration: creating interface alternatives through parallel authoring and runtime tuning. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*. 91–100.
[34] Devamardeep Hayatpur, Seongkook Heo, Haijun Xia, Wolfgang Stuerzlinger, and Daniel Wigdor. 2019. Plane, ray, and point: Enabling precise spatial manipulations with shape constraints. In *Proceedings of the 32nd annual ACM symposium on user interface software and technology*. 1185–1195.

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

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

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

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

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

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

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

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

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

[44] Jingyi Li, Wilmot Li, Sean Follmer, and Maneesh Agrawala. 2021. Automated Accessory Rigs for Layered 2D Character Illustrations. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 1100–1108.

[45] Klemen Lilija, Henning Pohl, and Kasper Hornbæk. 2020. Who Put That There? Temporal Navigation of Spatial Recordings by Direct Manipulation. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–11.

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

[47] Mark Mine. 1995. ISAAC: A virtual environment tool for the interactive construction of virtual worlds. (1995).

[48] Mark Mine, Arun Yoganandan, and Dane Coffey. 2014. 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*. 80–89.

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

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

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

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

[53] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, et al. 2016. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th annual symposium on user interface software and technology*. 741–754.

[54] Yun Suen Pai, Benjamin I Outram, Benjamin Tag, Megumi Isogai, Daisuke Ochi, Hideaki Kimata, and Kai Kunze. 2017. CleaVR: collaborative layout evaluation and assessment in virtual reality. In *ACM SIGGRAPH 2017 Posters*. 1–2.

[55] Navinchandra K Patel, Simon P Campion, and Terrence Fernando. 2002. Evaluating the use of virtual reality as a tool for briefing clients in architecture. In *Proceedings Sixth International Conference on Information Visualisation*. IEEE, 657–663.

[56] Tomislav Pejsa, Julian Kantor, Hrvoje Benko, Eyal Ofek, and Andrew Wilson. 2016. Room2room: Enabling life-size telepresence in a projected augmented reality environment. In *Proceedings of the 19th ACM conference on computer-supported cooperative work & social computing*. 1716–1725.

[57] Jeffrey S Pierce, Brian C Stearns, and Randy Pausch. 1999. Voodoo dolls: seamless interaction at multiple scales in virtual environments. In *Proceedings of the 1999 symposium on Interactive 3D graphics*. 141–145.

[58] Thammathip Piumsomboon, Gun A Lee, Jonathon D Hart, Barrett Ens, Robert W Lindeman, Bruce H Thomas, and Mark Billinghurst. 2018. Mini-me: An adaptive avatar for mixed reality remote collaboration. In *Proceedings of the 2018 CHI conference on human factors in computing systems*. 1–13.

[59] Thammathip Piumsomboon, Youngho Lee, Gun Lee, and Mark Billinghurst. 2017. CoVAR: a collaborative virtual and augmented reality system for remote collaboration. In *SIGGRAPH Asia 2017 Emerging Technologies*. 1–2.

[60] Kevin Ponto, Ross Tredinnick, Aaron Bartholomew, Carrie Roy, Dan Szafir, Daniel Greenheck, and Joe Kohlmann. 2013. SculptUp: A rapid, immersive 3D modeling environment. In *2013 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE, 199–200.

[61] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. 1996. The go-go interaction technique: non-linear mapping for direct manipulation in VR. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*. 79–80.

[62] Jun Rekimoto. 1999. Time-machine computing: a time-centric approach for the information environment. In *Proceedings of the 12th annual ACM symposium on User interface software and technology*. 45–54.

[63] Gabriele Salvati, Christian Santoni, Valentina Tibaldo, and Fabio Pellacini. 2015. Meshhisto: Collaborative modeling by sharing and retargeting editing histories. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–10.

[64] Ben Shneiderman. 2007. Creativity support tools: Accelerating discovery and innovation. *Commun. ACM* 50, 12 (2007), 20–32.

[65] Maximilian Speicher, Brian D Hall, Ao Yu, Bowen Zhang, Haihua Zhang, Janet Nebeling, and Michael Nebeling. 2018. XD-AR: challenges and opportunities in cross-device augmented reality application development. *Proceedings of the ACM on Human-Computer Interaction* 2, EICS (2018), 1–24.

[66] Sarah Sterman, Molly Jane Nicholas, and Eric Paulos. 2022. Towards Creative Version Control. In *CM Conference on Computer-Supported Cooperative Work and Social Computing*.

[67] 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*. 265–272.

[68] Kaj Sunesson, Carl Martin Allwood, Dan Paulin, Ilona Heldal, Mattias Roupé, Mikael Johansson, and Börje Westerdahl. 2008. Virtual reality as a new tool in the city planning process. *Tsinghua Science & Technology* 13 (2008), 255–260.

[69] Minhyuk Sung, Hao Su, Vladimir G Kim, Siddhartha Chaudhuri, and Leonidas Guibas. 2017. ComplementMe: Weakly-supervised component suggestions for 3D modeling. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–12.

[70] Hemant Bhaskar Surale, Aakar Gupta, Mark Hancock, and Daniel Vogel. 2019. Tabletinvr: Exploring the design space for using a multi-touch tablet in virtual reality. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.

[71] James Tam and Saul Greenberg. 2004. A framework for asynchronous change awareness in collaboratively-constructed documents. In *International Conference on Collaboration and Technology*. Springer, 67–83.

[72] Michael Terry and Elizabeth D Mynatt. 2002. Recognizing creative needs in user interface design. In *Proceedings of the 4th Conference on Creativity & Cognition*. 38–44.

[73] Balasaravanan Thoravi Kumaravel, Fraser Anderson, George Fitzmaurice, Bjoern Hartmann, and Tovi Grossman. 2019. Loki: Facilitating remote instruction of physical tasks using bi-directional mixed-reality telepresence. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 161–174.

[74] Walter F Tichy. 1985. RCS—A system for version control. *Software: Practice and Experience* 15, 7 (1985), 637–654.

[75] Unreal. 2022. *Unreal Editor VR Mode*. https://docs.unrealengine.com/5.0/en-US/vr-mode-in-unreal-editor/

[76] Nelson Wong and Carl Gutwin. 2010. Where are you pointing? The accuracy of deictic pointing in CVEs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1029–1038.

[77] Haijun Xia, Bruno Araujo, Tovi Grossman, and Daniel Wigdor. 2016. Object-oriented drawing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 4610–4621.

[78] Haijun Xia, Sebastian Herscher, Ken Perlin, and Daniel Wigdor. 2018. Spacetime: Enabling fluid individual and collaborative editing in virtual reality. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 853–866.

[79] Enes Yigitbas, Jonas Klauke, Sebastian Gottschalk, and Gregor 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, 1–10.

[80] Lei Zhang and Steve Oney. 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*. 342–353.

A STUDY SURVEYS

A.1 Pre-task Survey

(1) What is your self-rated experience level in using VR? Select only one option below.
- Beginner
- Intermediate
- Expert
- Other:_____

(2) What have you used VR for in the past? Check all that apply.
- ☐ Gaming and Entertainment
- ☐ Art and Design
- ☐ Architecture
- ☐ Education
- ☐ VR Development
- ☐ VR Product Design
- ☐ VR Research
- ☐ Healthcare
- ☐ Other:_____

(3) Have you used any of the following tools/software to track multiple versions of your work? Check all that apply.
- ☐ Version Control System (e.g., Git, SVN)
- ☐ Adobe Creative Suite (e.g., Photoshop, Illustrator)
- ☐ Figma
- ☐ Google Doc
- ☐ Other:_____

A.2 Survey Part I

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

Usability and Utility of Features

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

System suitability

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

A.3 Survey Part II

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

Communication and Awareness across Different Versions (using Portals)

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

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

(1) I was aware of the presence of my collaborators when we are in the same version.
(2) It was easy to understand my collaborator's actions when they are in the same version as me.
(3) It was easy to collaboratively go to a different version using the shared visualization.
(4) During the group discussion, it was easy for me to refer to a specific version in the shared visualization.
(5) During the group discussion, it was easy for me to refer to a specific object in the scene.
(6) During the group discussion, it was easy to understand which version my collaborator was referring to in the shared visualization.
(7) During the group discussion, it was easy to understand which object my collaborator was referring to in the scene.
(8) It was easy to merge objects from another version using the shared visualization.
(9) During the group discussion, I think the shared visualization would be useful.

General Experience

(1) My overall communication with my collaborators felt natural.
(2) To what extent did you experience that you and your partner collaborated?
(3) Think of some previous time (before today) when you enjoyed collaborating with someone. To what extent did you enjoy collaborating with your partner in today's task?
(4) To what extent would you, on another occasion, like to carry out a similar task with your partner?