GestureCanvas: A Programming by Demonstration System for Prototyping Compound Freehand Interaction in VR

Anika Sayara
sayanika@cs.ubc.ca
University of British Columbia
Vancouver, British Columbia, Canada

Emily Lynn Chen
echen05@students.cs.ubc.ca
University of British Columbia
Vancouver, British Columbia, Canada

Cuong Nguyen
cunguyen@adobe.com
Adobe Research
San Francisco, California, USA

Robert Xiao
brx@cs.ubc.ca
University of British Columbia
Vancouver, British Columbia, Canada

Dongwook Yoon
yoon@cs.ubc.ca
University of British Columbia
Vancouver, British Columbia, Canada

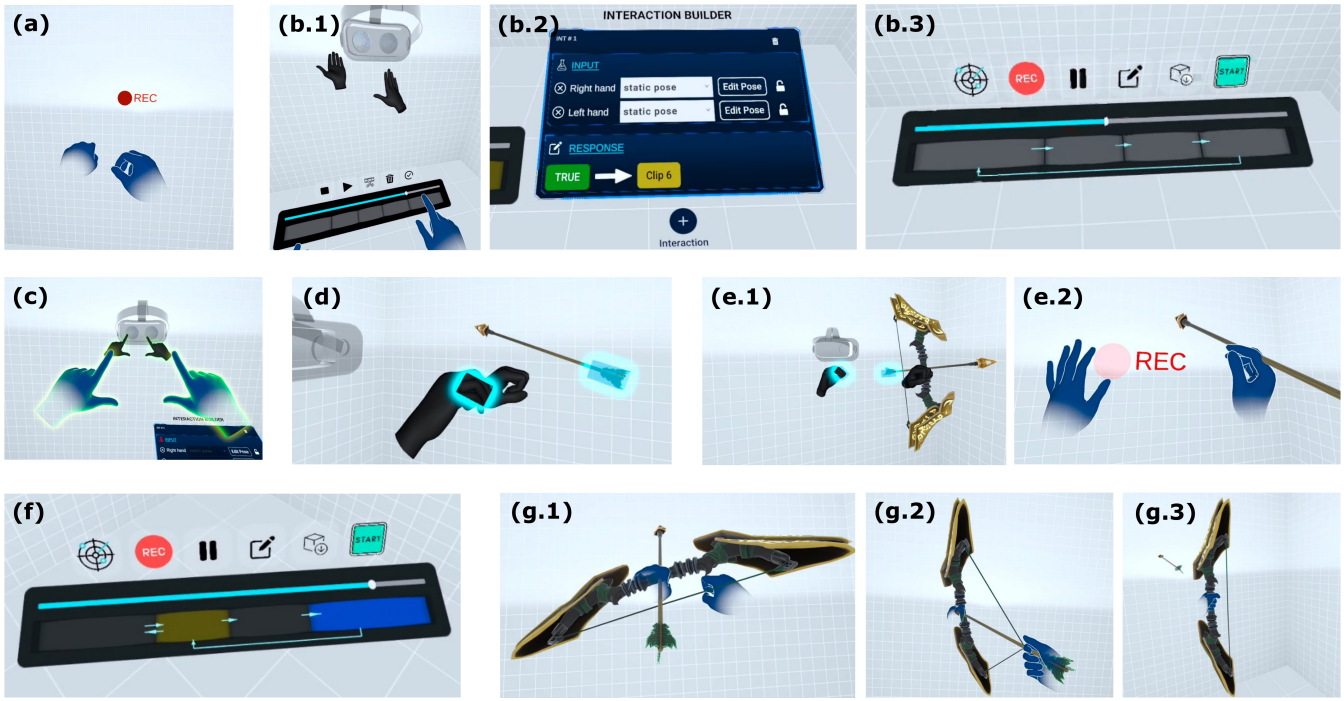

Figure 1: A workflow of prototyping an example compound freehand interaction (CFI) in GestureCanvas. (a) Demonstrate the whole sequence of CFI. (b) Create sub-interaction clips: (b.1) Segment the recorded demonstration into clips. Then, the system (b.2) infers the input trigger logic and (b.3) sets default transition order between clips. (c) Test the recognizer on-the-fly. (d) Refine the trigger logic by creating a Spatial Event Trigger (a Proximity Trigger glowing in blue, in this example) and anchoring it to the target object. (e) Author the system response: (e.1) Import and place props (the bow and arrow) in the environment; (e.2) Record a spatial animation by directly manipulating props (the arrow). (f) Weave the clips with new transition logics for branching and looping. (g) Playtest the full sequence of the authored CFIs. (g.1) Perform a gesture to summon bow and arrow; (g2) Pull back arrow and the bow string; (g3) Release to see the arrow fly away.

UIST '23, October 29–November 01, 2023, San Francisco, CA, USA

ACM ISBN 979-8-4007-0132-0/23/10…$15.00
https://doi.org/10.1145/3586183.3606736

ABSTRACT

As the use of hand gestures becomes increasingly prevalent in virtual reality (VR) applications, prototyping Compound Freehand Interactions (CFIs) effectively and efficiently has become a critical need in the design process. Compound Freehand Interaction (CFI) is a sequence of freehand interactions where each sub-interaction in the sequence conditions the next. Despite the need for interactive prototypes of CFI in the early design stage, creating them is effortful

and remains a challenge for designers since it requires a highly technical workflow that involves programming the recognizers, system responses and conditionals for each sub-interaction. To bridge this gap, we present GestureCanvas, a freehand interaction-based immersive prototyping system that enables a rapid, end-to-end, and code-free workflow for designing, testing, refining, and subsequently deploying CFI by leveraging the three pillars of interaction models: event-driven state machine, trigger-action authoring, and programming by demonstration. The design of GestureCanvas includes three novel design elements — (i) appropriating the multimodal recording of freehand interaction into a CFI authoring workspace called Design Canvas, (ii) semi-automatic identification of the input trigger logic from demonstration to reduce the manual effort of setting up triggers for each sub-interaction, (iii) on the fly testing for independently validating the input conditionals in-situ. We validate the workflow enabled by GestureCanvas through an interview study with professional designers and evaluate its usability through a user study with non-experts. Our work lays the foundation for advancing research on immersive prototyping systems allowing even highly complex gestures to be easily prototyped and tested within VR environments.

KEYWORDS

compound freehand interaction, virtual reality, prototyping, programming by demonstration

ACM Reference Format:
Anika Sayara, Emily Lynn Chen, Cuong Nguyen, Robert Xiao, and Dongwook Yoon. 2023. GestureCanvas: A Programming by Demonstration System for Prototyping Compound Freehand Interaction in VR. In *The 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23), October 29–November 01, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3586183.3606736

1 INTRODUCTION

Hand interactions enhance immersiveness in virtual reality (VR) experiences by converging the way we interact with the contents of the physical and the virtual world [9, 60]. With the recent advancements in freehand tracking technology [20, 21, 62] and the widespread inclusion of controller-free hand tracking capabilities in commercial head-mounted displays (HMD), interaction designers can now incorporate more expressive hand gestures into their designs that go beyond basic touch, grab and hold [5, 9]. This shift toward flexible hand inputs is evident from recent VR game titles where users use numerous hand actions and gestures [4, 56].

These hand interactions consist of high dimensional hand-based input and rich, expressive responses [5]. Many of these high dimensional freehand interactions are also *compound* in nature. A *compound freehand interaction* (CFI) can be defined as a sequence of trigger-action pairs where each sub-interaction conditions the next, similar to a compound gesture in touch interfaces [39]. For example, consider a spell casting interaction in a VR game: the user makes a fist to conjure up a fireball, pulls back the hand to make the fireball bigger, and then throws the hand forward to launch the fireball attack. Notice that this interaction is composed of a sequence of three distinct sub-interactions: 1) making a fist, 2) pulling back the hand, and 3) throwing the hand forward. Here, sub-interactions 2 and 3 are conditioned on 1. Also, note the potential branching behavior of this sequence: after sub-interaction 1, users can take two possible routes: they can either perform only sub-interaction 3 (producing a small fireball), do both 2 and 3 sequentially (producing a larger fireball), or loop back to the initial empty state by releasing the fist pose (canceling the spell). Thus, CFI involves the phrasing [10] of sub-interactions into a higher level interaction unit. The overall interaction is therefore achieved through the seamless coordination and synchronization of different hand movements, postures or actions involved in each sub-interaction. Likewise, CFI can be explained with the following characteristics:

- It is composed of a sequence of sub-interactions defined as trigger-action pairs.
- Each sub-interaction in the sequence conditions another.
- May have branches and/or loops.

Because of its embodied and complex nature, early testing is important for effectively designing a viable CFI. For example, our preliminary need-finding study suggests that physical comfort is an important consideration when designing CFI. This is consistent with prior research findings and hand interaction design guidelines [7, 48, 50]. However, early stage prototyping methods such as paper prototyping or role playing are insufficient for assessing physical comfort of CFI accurately [7, 28, 45] since the limited FoV of the VR headset influences how users approach a task [13, 16, 27, 37]. Thus, compared to the traditional design process, the need for an interactive prototype arises much earlier in the design process for CFI.

Despite the clear need for interactive prototypes of CFI from the early design stage, creating them is effortful and time consuming. This is not ideal for early design stages that require rapid design iterations. In the absence of support for CFI in both late breaking works [5, 19, 34, 43, 44, 58] and commercially available rapid prototyping tools [1, 12, 18, 36], designers lean towards programming based approaches for creating interactive prototypes [7, 28]. This requires them to build gesture recognizers, set up high dimensional input conditionals, author logic for transitioning to different sub-interactions including looping and branching, and create conditional response events for each sub-interaction that maps an input to either an action or a state transition. This highly technical workflow excludes non-technical designers [7, 33, 34] and end-user developers [7] and is effortful to create, even for designers with a good technical background. Need for programming also slows down the design cycle [34], which discourages design exploration often leading to design fixation [25, 34]. Our preliminary study participants echoed this sentiment, expressing frustration when realizing a design will not work given the hardware or software limitations later in the design process, and reluctance in changing the designs given the effort they put in for creating the prototype [28]. Thus, there is a need for a rapid prototyping tool for CFI that can increase the productivity of designers by letting them focus on designing the interaction itself rather than spending hours coding an interaction just so they can try out a candidate design.

An ideal CFI authoring system aimed at designers should satisfy the following requirements: *(1) Code-free VR authoring:* The designer should be able to rapidly create, test and iterate on their

CFI designs in VR without coding [7, 28, 32, 33, 46] by abstracting the complexities of building the gesture recognizers, setting up input conditionals and transition logic for the sub-interactions. *(2) Integrated testing:* Past research [7, 17, 28] and our preliminary study indicates that designers often find testing and debugging difficult due to frequent switching between authoring on desktop and testing in VR. Thus an ideal CFI authoring system should allow designers to test the recognizers as well as the individual components of an input condition within the immersive environment without having to switch between 2D and 3D. *(3) Iterative Refinement:* Designers should be able to make iterative refinements to the design and incrementally improve the performance of the recognizers with added gesture data and/or augment the hand-based conditions with additional event triggers as delimiters for reduced false positives/negatives [23, 29, 51, 57]. *(4) Support early-to-medium stage prototyping in developer tools* Past research [7, 28, 33] indicates that a code-free prototyping tool should make it easy to transition to code-based development platforms, as a way to facilitate early-to middle-stage prototyping in developer tools. This is particularly important to ensure that the prototyped interaction matches the final build and developers don't have to go through the entire process of building recognizers, refining conditions, setting up transitions all over again when moving beyond the prototyping phase.

To address these requirements, we designed GestureCanvas, which extends the use of *programming by demonstration* (PbD) to CFI authoring. PbD is a commonly used approach for enabling the authoring of complex behaviors without code [14, 17, 33, 39, 58]. While GesturAR [58] previously used PbD for freehand interaction authoring, it lacked adequate support for the sequenced nature of CFI.

Our solution GestureCanvas, addresses this gap by leveraging three pillars of interaction models: event-driven state machine, trigger-action authoring, and programming by demonstration for a rapid, end-to-end, and code-free workflow of CFI authoring. Thus our contributions include:

- A system which enables a rapid workflow for designing, training, testing, and refining CFI without coding, and provides a seamless transition to code-based tools such as Unity when moving to later design stages.
- The design of GestureCanvas, the hand-interaction based immersive authoring environment for CFI that enables this rapid workflow and includes the following novel design elements:
  - Turning users' demonstration of the interaction into an authoring interface called Design Canvas
  - Inferring the input trigger logic for each sub-interaction in the sequence from users' demonstration of hand actions
  - On the fly testing to enable the validation of each input trigger individually and subsequent in-situ refinement or revision

2 RELATED WORK

In GestureCanvas, we build upon ideas from AR/VR authoring and programming-by-demonstration domains. Below, we explain how our system is situated in existing literature.

2.1 Authoring Tools for AR/VR

Although consumer AR/VR devices are becoming more accessible and affordable, authoring new AR/VR experiences remains challenging for non-technical creators [7]. In a positional paper [46], Michael Nebeling and Maximilian Speicher categorized the landscape of authoring tools for AR/VR into five classes: (1) tools for mobile and web designers, (2) tools for basic AR/VR behaviors, (3) tools for advanced AR camera-based interactions, (4) tools for 3D modeling, and (5) 3D game engines. A major research theme in recent years is developing immersive prototyping tools that fall partially between classes two and three ([33, 34, 42, 43]). These tools focus on helping designers mock up complex AR/VR behaviors quickly rather than developing and shipping complete applications. However, recent studies [7, 28, 33] identified that limited testing capability and lack of support for early-to-medium stage prototyping in these tools create barriers in the design process by locking them into the tool and making it hard to transition to developer platforms when moving beyond early stage prototyping. Moreover, defining and developing complex interactions based on gesture recognition still requires significant programming and is better suited for tools in class five. GestureCanvas addresses this gap and focuses on enabling designers to prototype and test interactive VR experiences with freehand tracking and makes it easy for designers to move to later design stages by providing support in developer tools.

2.2 Authoring Dynamic Behaviours through Programming by Demonstration (PbD)

Making computer programming more accessible for non-technical users through PbD [26, 35] is an active area of research in HCI with numerous explorations in sensor-based interactions [14, 22, 31], touch gesture [38, 39], and AR [33, 58]. GestureCanvas extends PbD workflow to VR to help users create *compound freehand interaction.*

While PbD has been explored previously for authoring touch based gestures [38, 39] and mid-air gestures [6, 22, 61], these works require 2D interfaces to visualize and edit the sensor outputs, which is not suitable for VR. Additionally, one of the primary focus of these works is the design of the recognizer itself. In contrast, our contributions do not include designing novel recognizers, but rather designing a rapid workflow that fits designers' need for prototyping and testing in an immersive environment.

Closest to our work is GesturAR [58] which explored the use of PbD for flexible freehand interaction authoring. To author CFI using GesturAR [58], one would have to capture each static pose separately, specify the corresponding action, then add the next pose in the sequence. Whereas, with GestureCanvas, one can demonstrate the entire interaction at once, use the timeline interface to segment it into sub-interaction clips, refine the system-inferred triggers as needed and finally add actions or transitions. Thus, GestureCanvasdistinguishes itself uniquely from GestureAR [58] in its state-model based branches, conditionals and sequencing. This allows designers to model the contextual and temporal relationships within a gesture sequence to form the interaction flow, unlike GestureAR which supports only the recognition of single poses or fixed sequences of poses.

Also, note that in GestureCanvas, the designer performs the entire gesture sequence first. Hence, there is no spatial mapping to virtual objects at this time. To allow composability, GestureCanvas represents the demonstration of the interaction as a coupled avatar-timeline space like [11, 59], but extends its use to authoring of responses and event triggers with spatial contextualization and direct manipulations of the CFI sequence.

2.3 Design Tools for Hand-Based Interactions

More VR applications are starting to enable hand tracking as the primary interaction mode. Creating hand-based interactions is traditionally a highly technical task. Developers typically need to write *trigger* scripts to recognize events from continuous tracked joints on the hands and *action* scripts that map the event outputs to 3D effects rendered in VR. Creating easier interfaces to help designers create gesture-based interactions in VR is a major research theme.

Several solutions have looked at the *action* side. In MagicalHands [5], the authors explore different interaction techniques to control an animation in VR using gestures. The focus of the work was mainly on the response side; gestures were triggered by pressing a button on a VR controller. GestureWiz [52] leverages a wizard-of-oz technique to help designers test new interactions with human help. Using the crowd to mock up system responses allows designers to explore a wide range of effects, but it requires extra time and work to coordinate with crowd users. We adopt a similar approach to GesturAR [58] by allowing designers to create animations and use them as responses. But to make it more suitable for VR, we allow designers to specify the physics of the object and spatially associate it to bone joints and player position using the Design Canvas.

Other work focuses on helping users create novel *triggers*. Gesture Knitter [41] allows users to quickly create novel recognizers without having to train a new machine learning model and collect data from scratch. A key idea of Gesture Knitter is that users can combine multiple smaller gesture primitives to form a more complex gesture. Each primitive is authored by automatic decoding from user demonstration or by writing declarative scripts. Gesture Knitter focuses on creating a robust recognizer by providing data synthesis and modeling tools. By contrast, our focus is on assisting the end-to-end design of an interaction. We provide interfaces to help designers experiment with both the recognizers and how their output would map to effects in VR. Our approach is also inspired by Gesture Studio [39], which is a mobile authoring system that combines both programming by demonstration and video timeline interfaces to help users turn a sequence of touch inputs into compound touch gestures. Our interfaces extend this idea to spatial authoring in VR.

3 PRELIMINARY NEED-FINDING STUDY

To understand current practices, designers' needs, and challenges of creating freehand interactions for VR experiences, we conducted two exploratory analysis: one interview study with five professional and five novice VR designers and one video analysis study. We distilled these findings into a set of design requirements for designing hand interactions in VR.

3.1 Interview Study

Participants. We recruited ten (10) participants with varying levels of design experience. Out of the 10 participants, five were professional designers working in the AR/VR domain in either academia or industry in professional capacity. Their job titles included – Senior Design Lead, Creative Director, Business Analyst, Freelance Designer, and Research Scientist. The length of their professional design experience ranged from three to twenty years, and they have worked on projects involving hand interactions. The remaining five were novice designers who were all graduate students with the experience of designing at least one AR/VR application with hand interaction. The professionals were recruited via purposive sampling, where we reviewed their publicly available online profiles and/or portfolios to ensure that the individual met the recruitment criteria before sending out the request to participate. The novice designers were recruited via convenience sampling.

Procedure. The interviews were about one hour long and semi-structured. Our interview questions were focused on capturing the participants' experience of designing hand interactions for AR/VR applications. We encouraged the interviewees to ground their responses in their past projects by screen sharing videos of AR/VR hand interactions they designed in the past, whenever available, and also by presenting representative example systems we sampled from online public videos. This helped us get more detailed accounts of hand interaction design challenges. The interviews were conducted remotely over online video-conferencing (Zoom). The interviews were video recorded, transcribed, and then analyzed using reflexive thematic analysis [8].

3.2 Video Content Analysis

To identify requirements of the prototyping system for freehand interactions, we coded and analyzed the online public videos on freehand interactions in diverse VR applications using inductive content analysis [15]. These applications were chosen based on the list of popular hand tracking apps in the SideQuest and Meta stores. The list also contains some demo projects that implemented experimental hand interactions (e.g. Tiny Castles [49]). The sampling yielded 12 videos that feature a variety of freehand interactions in VR. The full list of the selected VR applications are available in the supplementary materials.

Our video content analysis shows a wide range of hand-based input triggers used in VR. For example, applications like Elixir [40] were dominated by simple freehand interactions with input triggers primarily comprised of touch events and static hand poses. On the hand, The Wizards - Dark Times [53] is an application that employs CFI for casting various magic spells. The responses to these interactions were mostly relative to hands' spatial context in some way. Hand Physics Lab [24] is an application that relies heavily on physics-based responses to hand input. Spatially relative responses include fire or spells coming out from the fingers like in Finger Gun [54] and Tiny Castles [49] or energy spheres that appear on hands in Elixir [40] and Wizards - The Dark Times [53].

3.3 Design Requirements

Triangulating the findings from the interview study and the video content analysis yielded a set of design requirements (R0-5) for

hand interactions in VR. These consist of the overarching requirement (R0) for rapid prototyping of CFI in VR and five supporting requirements R1 - R5 that when fulfilled resolve R0.

R0. Rapid prototyping for CFIs in VR without coding. Despite early testing being important for freehand interaction design, state of the art tools lack support for rapidly prototyping interactions in VR. Most participants feel that the current workflow of creating interactive VR prototypes of hand interaction are effortful and time consuming, which is not ideal for early testing. Since the state of the art rapid prototyping tools for VR do not provide support for hand interaction, designers need to rely on game engines and software development toolkits. But as P6 mentioned, these tools were not "intended to be used for prototyping in the first place". Of note, the primary frustration among the participants about using game engines for prototyping VR hand interaction was not lack of coding skill, but the time and effort it takes to roll out something that works. For example, P4 is very comfortable in coding, but would still appreciate "anything capable of hand tracking without coding" because he thinks that jumping to Unity for coding just after sketching is a huge jump and there should be something in between that would allow him to quickly get the feel of the designed hand interaction in VR.

R1. Authoring high dimensional hand-based input conditions. The high dimensionality of hand-based input in VR results from the high degrees of freedom (DoF) of hands, their dynamic nature, and their spatiotemporal context in relation to the virtual elements in the VR environment. For example, the high DoF contributes to many possible hand poses, and if the hand input is dynamic, then the combination of the different hand poses along with the added dimension of direction, velocity, and sequence of changing hand poses introduce even more possibilities. Additionally, hand-based input can also be conditioned on how and when hands approach virtual objects.

R2. Defining the delimiting conditions for CFIs. State is a special kind of mode with well-defined input and response.

Designers need to meticulously handle the system's interpretation of hand input, which can get particularly tricky for hand interaction due to its open-ended and implicit nature. Modes refer to the different interpretations of the user input by the system, depending on the state which is active. For example, the same CAPS LOCK key on the keyboard is used to turn on and off capital letters when typing. Modes are useful to limit the number of different gestures to be used in a system but can be tricky to handle because "unlike non-gestural systems, mode switching is done implicitly in hand interactions" (P8). According to P10, it is important to meticulously design the delimiting conditions for hand input to avoid "false positives". She explains, "Your hands are always moving … How am I going to make sure that those hands aren't being tracked when I don't want them to be tracked?" Both P8 and P10 pointed out that such situations are likely to occur if (1) mode switches are not addressed properly in the state designs and (2) if there are similar interactions in the gesture set.

R3. Testing the generated recognizers for designing reliable interactions. Testing the performance of recognizers is important for designing reliable interactions, but are effortful to run. Here reliability refers to the consistency of tracking technology. Highlighting the importance of reliable interactions, P6 mentioned, "Not all hand interactions can be reliably detected by existing technology … poorly designed gestures that do not give consistent output will cause frustration among users and they simply won't use it." However, the current workflow of testing the performance of recognizers in VR "comes with a lot of baggage." (P6). According to P8, "the frustrating part when dealing with virtual reality prototyping is - it's hard to get these 'unit tests' done". P8 further explained how he needed to put in extra effort of setting up a testing environment in VR for his project. In addition, the repeated context switching between designing in 2D and testing in 3D, is frustrating and disorienting (P2, P6, P8). As P8 mentioned, "You're stuck to your seat. You don't want to move around too much. So it's out of balance between only interacting with it in certain ways, … so I have to move away from the computer to properly test it, but then looking at all the debug logs or something in real time is hard, because now you have to build a way for your application to stream those debug logs to the headset".

R4. Authoring rich system responses as dynamic spatial animations The system responses to hand-based input are often spatially related to users' hands. For example, flamethrowers in Tiny Castles or fire from Finger Gun are associated with specific finger bones to give users the impression that fire is coming out from that particular finger joint. Moreover, for realistic representation of responses to direct manipulation, the virtual objects are often influenced by the physics system. For example, pushing an apple across the table, or throwing an object into the air require physics-enabled virtual objects that respond to gravity.

R5. Playtesting the designed hand interaction by performing it in VR. Because the feasibility of the designed hand interaction can only be known by performing it in VR, designers feel the need to try them out early in the design process. It is difficult to account for the limitations of the device, the tracking technology, and the associated embodied experience of hand interaction without performing it in VR and hence almost all participants emphasized the need for testing them out early in the design process. Hand interaction design in VR is heavily influenced by the limitations of the device and the tracking technology (P6, P7, P8, P9, P10). For example, P8 explained how gestures that are too broad can be problematic in VR if it starts outside of the Field of View (FoV). He also mentioned how users approach the same task differently with and without putting on the headset, impacting the dynamics of the intended hand interaction. Participant P6 pointed out the importance of early testing for designing "reliable" interactions with consistent tracking of the hand input. He mentioned "Whatever design concepts you come up with, the issues that can come up when in VR can be very unintuitive." Highlighting the mismatch between 2D designs and 3D implementation, P10 mentioned, "Nothing is as it seems when you actually put it into HoloLens and actually play with it". Participants also emphasized on testing to understand the embodied experience of hand interaction (P6, P8, P10). According to P8 "whether the gesture is comfortable enough or can it cause injuries or is tiring if done for long hours ... can only be detected by performing the hand gestures in VR".

As summarized and depicted in Fig 2, these requirements are met by the design of our system detailed in the next section.

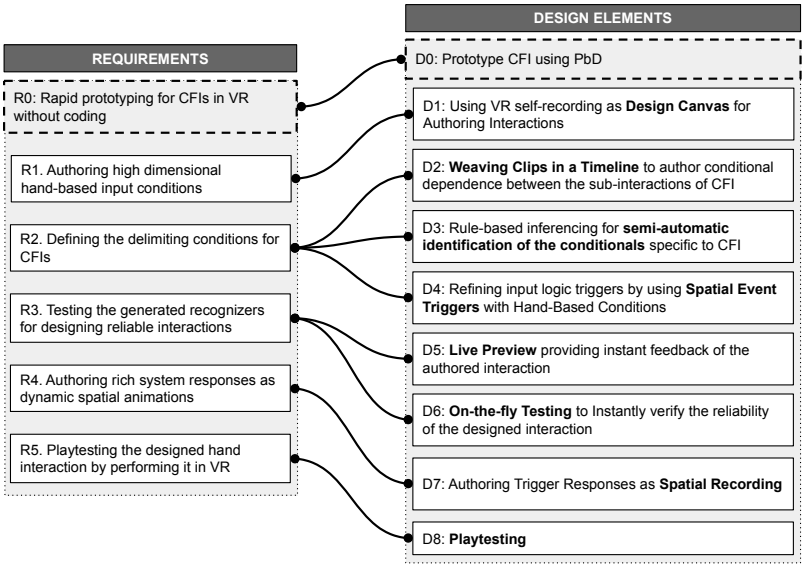

Figure 2: How the design requirements translate to the design elements of GestureCanvas. See Section 3.3 for the full description of the requirements and Section 5.1 for the design elements.

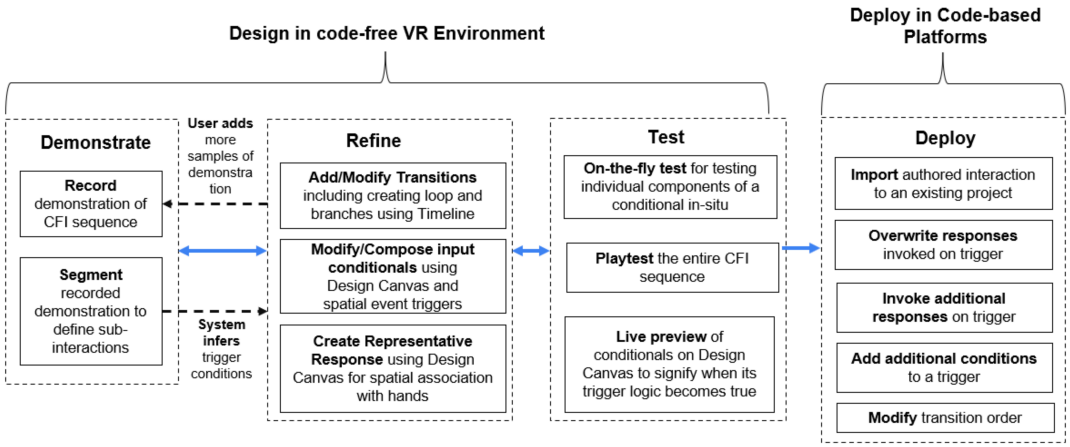

Figure 3: The rapid workflow enabled by GestureCanvas that allows designers to iteratively design, test, refine and subsequently deploy CFI without coding.

4 WORKFLOW ENABLED BY GESTURECANVAS

GestureCanvas enables a rapid iterative workflow for designing, training, testing, and refining CFI in a code-free immersive environment and provides support for subsequent transition to code-based tools when moving to late design stages. This ensures that the prototyped interaction matches the final experience and reduces the effort of creating the CFI all over again when moving to code-based tools. During prototyping in the code-free immersive environment, designers will:

- ***Record*** a demonstration of the entire CFI sequence, which is later represented as a spatially embodied self-recording called Design Canvas (D1).
- ***Create sub-interaction clips*** by segmenting the recorded demonstration using the metaphor of a timeline (D2).

Once the segmentation is done, GestureCanvas automatically infers the input triggers for each sub-interaction and creates a linear transition order by default (D3). At this point it is ready for playtesting and deploying. However, depending the need for refinement, designers can do any of the following optional steps:

- ***Create and modify clip transition logic*** including loops and branches.

- ***Compose / modify system inferred input triggers*** by augmenting it with spatial event triggers (D4) and/or by using Design Canvas to select specific pose / gesture combinations
- ***Test the performance of the recognizers*** in the authored input triggers (D5)
- ***Add samples*** to improve the performance of the recognizers
- ***Create representative responses*** using Design Canvas for spatial contextualization
- ***Playtest*** the full sequence of the authored CFI in VR.

After prototyping in the immersive environment, designers may deploy the authored interaction in code-based platforms such as Unity. At this point they may run the interaction as-is in the Unity project using our toolkit or may make code-level changes for further refinement.

5 DESIGN AND IMPLEMENTATION OF GESTURECANVAS FOR IMMERSIVE CODE-FREE AUTHORING OF CFI

For rapid, end-to-end, and code-free workflow, GestureCanvas leverages three pillars of interaction models: event-driven state machine, trigger-action authoring, and programming by demonstration.

- *Event-driven state machine:* To support the sequenced nature of CFIs, GestureCanvas incorporates a state machine-based backend that operates on event-state relationships. Each step of an interaction sequence is defined as a *clip* (state). The progress of the steps is defined as the transitions between them (events). A step, which we refer to as a sub-interaction henceforth, comprises a freehand gesture and the system's response to it.
- *Trigger-action:* A trigger is an atomic event that drives the state machine, and an action is a response to a given trigger. The system provides two types of actions: within-state actions authored as spatial animations (either synchronous or delayed), and between-state transitions to another clip. In the semantics of the trigger-action model, a trigger is conditioned by a given input gesture and the currently given clip (state).
- *Programming-by-demonstration (PbD):* To reduce the user's workload in authoring complex input conditions (R2), the state machine is designed via PbD. From the provided demonstration of freehand input sequences, the GestureCanvas system can infer the trigger logic. To facilitate manual refinement of the inferred conditionals, the system presents the user's demonstration as an embodied recording that affords spatial manipulation.

5.1 The Design Elements of GestureCanvas

In this section, we present the design elements of GestureCanvas and elaborate design rationales for them.

D0: PbD-based Prototyping. The challenge of addressing R0 is that the authoring process of CFI involves too many steps such as creating trigger-action pairs for each sub-interaction, providing samples and training the required recognizers for hand-based inputs, testing, editing and extending each of those input triggers, and creating conditional dependencies. Hence, letting the user go through each of these steps manually would be too effortful for the user even in a no-programming environment like GestureCanvas.

To simplify the workflow and reduce manual tweaking needed by the user, we chose to employ the programming by demonstration technique where users can author the intended interaction by directly demonstrating it in VR. Note that PbD has not reduced the steps that constitute a target compound freehand interaction, but rather it simplifies the authoring process by reducing the manual tweaking required for going through each of the steps mentioned above.

D1: Using VR self-recording as a Design Canvas. Since authoring CFIs requires the user to handle spatial, dynamic, and high-dimensional data (R1), the unique design challenge is to present and give users access to this rich data set that includes the changing spatial (positional and rotational) data of hand joints and head movement and the hand shape (curl, abduction, flexion, opposition) data of the fingers over a period of time. Defining each sub-interaction in CFI involves specifying trigger-action pairs. This demands the use of hand shape data for input triggers using static hand poses and spatiotemporal hand joint data for input triggers using dynamic hand gestures. Additionally, the experiential nature of CFI demands authoring of responses be done relative to users' spatial context. However, giving users access to this high dimensional dataset in VR for manipulation is a challenging design task.

To make the CFI data streams amenable to spatial contextualization and direct manipulation native to VR, we present the user's recording of self-demonstration as an embodied playback that can be viewed and controlled in the 3rd person perspective (see Fig 5b). In other words, the Design Canvas embodies the user's demonstration of intended interaction and serves as an authoring interface in the later stages of the authoring process. In particular, the Design Canvas shows the possible static hand poses and dynamic hand gestures for a given sub-interaction. Additionally, it provides visualization of the hand-based condition used as input trigger. Moreover, when using Spatial Event Triggers with hand-based conditions or when authoring responses using virtual props or visual effects, users can anchor the Spatial Event Triggers (Fig 5(a)) and virtual props or visual effects (Fig 5 (b)) directly onto the Design Canvas. Hence, the particular benefit of the design comes from "allowing" users to use the avatars to author CFI instead of having them to directly tweak raw data.

To implement the Design Canvas, we map the recorded spatial data of head and hand movements of each frame onto rigged hand models and a VR headset model in chronological order. Then we associate the progress of frames to the progress of slider allowing user to control the playback like video scrubbing.

D2: Weaving clips in a timeline to author conditional dependence between the sub-interactions of CFI. To facilitate the authoring of conditional dependency between the sub-interactions of CFI (R4) including loops and branches without using programming constructs in VR, we use the concept of weaving clips in a timeline, which is a common interaction technique for manipulating temporal sequences [11]. In GestureCanvas, each sub-interaction of CFI is represented as a clip in a timeline (Fig 1 (f)). In the timeline, the arrows connecting the clips define the transition order of the sub-interactions. Hence, to modify the transition order, one will

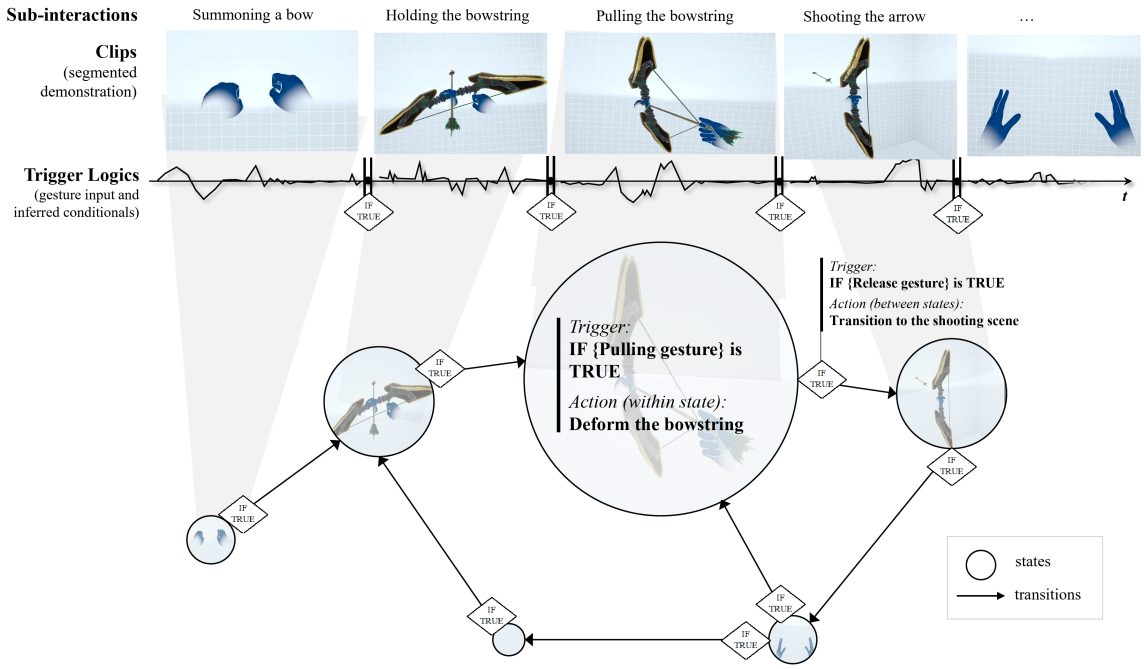

Figure 4: The GestureCanvas interaction model comprises an event-driven state machine, trigger-action authoring, and PbD. The event-driven state machine concretizes the sub-interaction steps of CFI. The trigger-action paradigm models atomic events that drive the state machine. The rapid design of the state machine is made possible by the PbD approach.

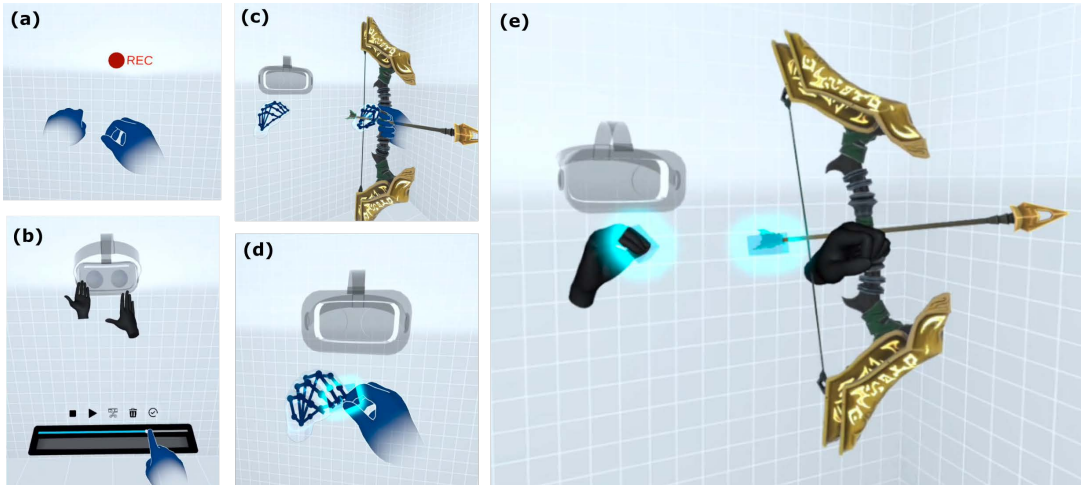

Figure 5: Self-recording as a canvas for PbD design. (a) Recording demonstration of interaction to prototype; (b) Viewing and controlling the recorded CFI demonstration in the 3rd person perspective; (c) anchoring virtual props (bow and arrow) on the Design Canvas. Moving closer to the Design Canvas reveals skeletal structure allowing user to anchor props on bone joints. (d) anchoring proximity (Spatial) Event Trigger on Design Canvas; (e) anchored props and Proximity Event Trigger on the Design Canvas.

need to modify the arrow connectors between the clips and creating loops or branches is as simple as creating a new arrow connector from the origin to the destination clip.

D3: Rule-based inferencing for semi-automatic identification of the conditionals specific to CFI. Even though the PbD approach reduces the effort required for creating and training recognizers for all possible hand-based triggers in each sub-interaction (R4), users still need to create the individual input triggers by specifying the hand-based conditions to use. This still is effortful and may become overwhelming for users to tweak manually.

Hence, to reduce manual effort needed for authoring input logic for CFI, GestureCanvas uses rule-based inferencing for semi-automatic identification of the input trigger logic from users' demonstration of the intended interaction. The data of each clip is analyzed to identify whether user's intent was 1) to create a unimanual or a bimanual input trigger, 2) for unimanual trigger, what is the handedness (right or left) of the input trigger, and 3) whether the user wanted to use a static hand pose or a dynamic hand gesture. The right hand and left hand recorded data of each clip is analyzed according to the rules presented in Fig 6. If hand condition is set for both hands, then the system infers it as a bimanual input trigger, otherwise as a right hand unimanual or left hand unimanual input trigger depending on the hand whose condition was set.

D4: Refining input logic triggers by using Spatial Event Triggers with hand-based conditions. Typically, in a VR experience, the hand-based conditions are used together with other Spatial Event Triggers, either to extend input logic (R1) or as delimiting conditions for hand-based input (R2). Setting up these triggers requires spatial information of the user and their virtual environment. Event triggers of this type include: position (the player arrives at a location), collision (player's hands collide with an object), approach (player approaches an object), ray hit (raycast from player's hands hits an object or a location), gaze (player looks at an object) etc.

Since the recording of the intended hand interaction is the first step for authoring and no virtual objects are set up at this stage, GestureCanvas cannot capture the spatial contextual relevance of hands and virtual objects at the same time. Hence, users have to manually set up the conditions for Spatial Event Triggers. This poses two design challenges: 1) how to represent the Spatial Event Triggers in VR so that they can be configured through direct manipulation with reference to spatial context; and 2) how to provide visual feedback of the condition it embodies?

Since the list of possible Event Triggers that can be used with hand-based conditions is quite long, we implemented only the Proximity Event Trigger as a representative example. The design of the Proximity Event Trigger can be extended to create other relevant triggers. We chose proximity as a representative example because most Event Triggers used in freehand interactions involved one of position, collision, approach or touch events and all of these can be implemented as a single Proximity Event Trigger that measures proximity between two objects or points and fires when a set distance is detected.

We represent the Proximity Event Trigger in VR as a volumetric cube that users can scale, position in the environment, or use on the Design Canvas. This embodiment of the proximity condition as manipulable cubic volume visually depicts the condition it embodies. Once the proximity trigger is placed or anchored in the intended location, it needs to be configured to specify when it should fire. For this, the user first makes a selection of touch (player is within the boundaries of the proximity volume) or distance (player is at a certain distance from the proximity volume). To configure a touch based proximity event, the user starts recording the proximity condition. During the recording time, any part of the player's hands that is within the boundary of the proximity volume is specified as the object to monitor. This is visually shown to the user using the Design Canvas (Fig 5 (d)). To configure distance based proximity events, the user moves a second control to the desired distance from the object, pulls up the hand menu and sets the trigger distance.

We make use of Unity's collision detection system to implement the proximity trigger event logic. The manipulable volumetric proximity trigger monitors the paired object or hands for collision (touch condition) or for set distance (distance condition) and fires when the set condition is met.

D5: On-the-fly Testing to instantly verify the reliability of the designed interaction. The design challenge for fulfilling R3 is to incorporate testing of the performance of the hand-based input triggers into the system such that it does not feel disjointed from the authoring process.

To support this, GestureCanvas allows on-the-fly testing of the input triggers, where users can test each input trigger individually, allowing them to test, refine, or revise in-situ. During on-the-fly testing, GestureCanvas uses users' real time hand data, passes it to the relevant triggers and signifies output of the recognizers by highlighting the user's hands (Fig 8 (b)) whenever the set input trigger becomes true.

The recognizer used for static hand pose is based on the ShapeRecognizer ScriptableObject of the Oculus Interaction Package [2]. We record the state of the features (curl, flexion, abduction, and opposition) of the fingers in real time and then perform template matching with selected hand pose data for detecting the pose. In the same way, we also make use of the TransformRecognizer ScriptableObject of the same package for distinguishing between different wrist orientations. For detecting dynamic hand gestures, we use Jackknife [55], which is a general purpose recognizer. The reason for selecting Jackknife is that it does not require extensive training data to work; in fact, a single training sample is sufficient in most cases. However, with added samples it can be scaled to support users' hand gestures in general.

D6: Live Preview providing instant feedback of the authored interaction. R3 can be partially addressed by enabling users to debug the authored interaction. This introduces the design challenge of providing a live preview of the authored interaction.

To enable this, GestureCanvas passes the recorded demonstration data, shown looping repeatedly in the GestureCanvas, to the configured trigger recognizers and indicates the result of each recognizer by highlighting the hands. That is, the hands of the avatar glows green at points in the demonstration when the set trigger becomes True (Fig 8 (a)). This helps users to debug their logic visually without requiring them to open the interaction builder. It also helps them to visually see the changes they make on the input trigger logic.

D7: Authoring System Responses as Recorded Spatial Animation. To facilitate creating representative response to hand based input, we allow creating spatial association with hand position and orientation using the Design Canvas. To elaborate, designers can anchor virtual assets, visual effects or recorded animation directly on the bone joints of the Design Canvas (Fig 5 (c)). Thus, the Design Canvas serves as a placeholder for player and provides a way for designers to specify the position and orientation of the assets or animation playback relative to that of the player. For example, to record a flying arrow, the user applies physics to the arrow and then while

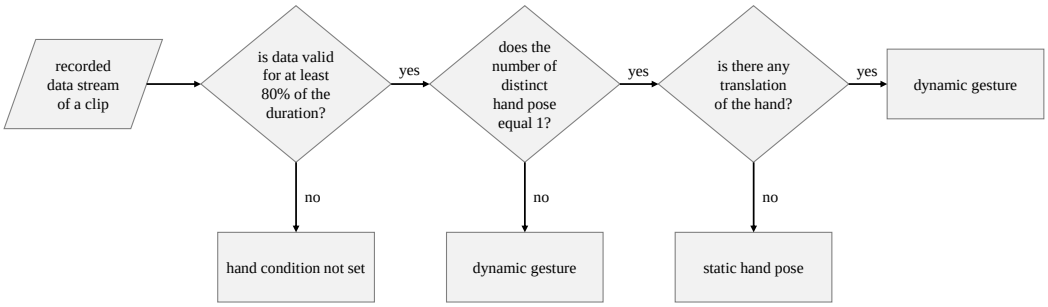

Figure 6: Rules for inferring the input trigger logic from users' demonstration of the interaction: The system logs invalid data code when hands are out of FoV. If at least 80% of recorded data within a clip is valid, the system extracts the unique hand poses found within the clip. If the number of unique hand poses within the clip is more than one, GestureCanvas infers it as dynamic gesture trigger, otherwise it checks whether there is any translation of the hand during the duration of the clip. If translation exists, then it is dynamic gesture trigger, otherwise it is a static pose trigger.

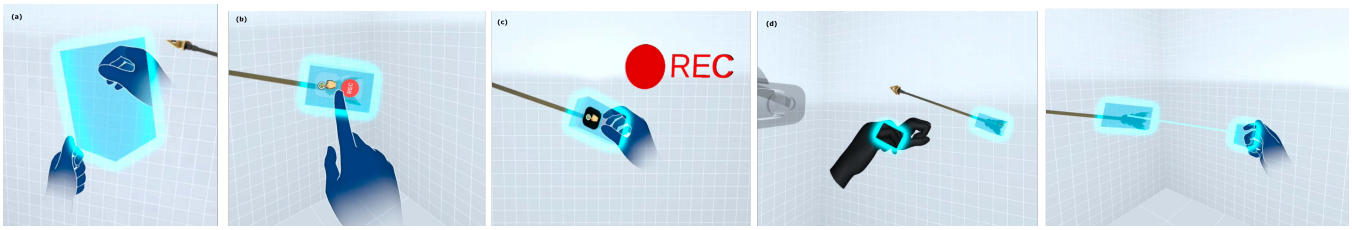

Figure 7: Refining conditionals using a proximity trigger. (a) Proximity trigger represented as a volumetric cube that users can directly manipulate. (b) Option to set up the condition appears when user is not manipulating the trigger. User can toggle between touch and distance. (c) User anchored the Proximity trigger on the tail of the arrow and started to record the touch condition. Since the user wants to detect touch with the tip of the index finger, the user placed the tip of their finger inside the proximity volume. (d) Touch condition between tip of arrow and users' index finger tip is visualized in Design Canvas. (e) Distance mode of operation. When distance is selected, another proximity volume splits out from the first, connected by a line. The user can place the second volume at the desired distance.

recording, simply throws the arrow in the air. Once the trajectory of the arrow's flying path is recorded, users can anchor the arrow onto the right hand of the Design Canvas such that during playtest, the arrow's trajectory will be recalculated from the new position (user's hands) using the original recorded path. To recalculate the recorded path, we subtract the displacement vector from each point in the recorded path to get the corresponding point in the new path.

6 DEPLOYING THE PROTOTYPED INTERACTION TO CODE-BASED PLATFORMS

While the rapid workflow for designing, testing, and refining CFI in VR is useful, a key challenge is that the prototyping is completely isolated from the development. This is not ideal since developers must essentially start from scratch after receiving a prototype from designers, whose mechanics might be completely different from the one designers have prototyped, tested and refined using GestureCanvas. Additionally, developers want flexibility in the development process so that they can incrementally add/modify game logic.

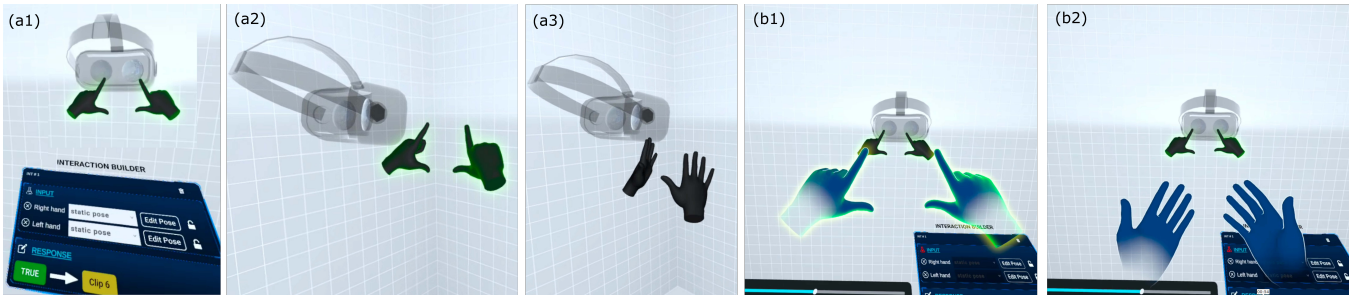

Figure 8: On-the-fly gesture testing and Live Preview for debugging the authored interaction. (a) Live Preview. (a1) The green glow on the hands of the Design Canvas gives a preview of the hand pose for the set trigger, which is seen on the Interaction Builder [Right Hand Pose and Left Hand Pose to be True]. (a2) Whenever the playback would activate the selected trigger, the hands of the avatar glows green to signify the trigger to the user, (a3) the hands of the Design Canvas do not glow green when playback data is not true for set trigger. (b) On-the-fly Testing. (b1) The user can test each individual conditional they set up by starting the test mode, which lets them test the condition in-situ. Here, the user is doing the same hand pose as the trigger set previously, hence, the triggers at this point are true causing the user's hands to glow green. (b2) However, the glow turns off whenever the triggers become false. This lets user to test the recognition reliability of their designed trigger in-situ.

For example, a developer might want to use the bow and arrow CFI prototyped in GestureCanvas in their project. However, they might also want to add additional game logic on top of it where the player can only use 5 arrows every time they pick up a collectible. Such game logic cannot be prototyped in GestureCanvas and must be done programmatically. Thus an ideal interaction prototyping system should *(1) enable a seamless transition to development and (2) afford post-hoc code-level editing.*

Thus, we have implemented a toolkit on top of Unity that enables seamless transition from prototyping to development by allowing developers to import the interactions prototyped in GestureCanvas into an existing application. The imported items include the trained recognizers and CFI data stream consisting of sub-interaction data in a JSON format. The toolkit comes with a preconfigured drag-and-drop asset that can use the imported recognizers and CFI data to run the prototyped CFI in the target application. This ensures that the performance of the recognizers in the target application matches the one used in GestureCanvas.

To allow post-hoc level editing without putting the overhead on developers to understand the code-structure for running the imported CFI, we have included a custom GameEventListener script within the package exported from GestureCanvas. These scripts enable developers to attach custom triggers and responses to their own GameObjects using the Unity Inspector window, and add the capability to define more specific conditional inputs to each response. Additionally, developers can directly manipulate the clip sequence through functions that modify the JSON so that new clips can be added, existing clip data can be modified, or the sequence of clips can be adjusted as per new and evolving application choices and requirements. Thus, the toolkit allows developers to make the following code-level changes to the imported CFI:

- Overwrite responses invoked on trigger
- Invoke additional responses on trigger
- Add additional conditions to a trigger
- Modify transition order

Figure 9: The bow and arrow CFI prototyped in GestureCanvas is deployed in a game project and an additional game logic is added using our Unity toolkit such that every time player shoots an arrow the text showing arrow counter decreases

Using the toolkit, we deployed the bow and arrow interaction prototyped in GestureCanvas (Fig 9) and made post-hoc editing to add the game logic of player running out of arrows in three simple steps - (1) Attach the provided GameEventListener script to the GameObject executing the response (in this case, the bow and arrow count text) (2) Drag and drop the function decreasing the arrow counter under "Custom Responses" in Unity Inspector window and (3) invoke the GameEvent under custom-specified conditions by calling 'GameEvent.Raise()'.

7 EVALUATION

We conducted a user study with 8 non-expert participants with little to no experience designing hand interactions to evaluate the usability of GestureCanvas. Additionally, in order to get insight about the utility of the tool [30] in a designers' typical workflow for authoring CFI, we conducted an expert interview study.

7.1 Evaluating the Usability of GestureCanvas with Non-Experts

For assessing the usability of GestureCanvas, we recruited 8 participants (6 women and 2 men) between the ages 19 and 34 with little to no experience designing hand interaction.

During the study, the participants first completed a tutorial design task where a member of the research team gave step-by-step instructions for creating their first CFI using GestureCanvas. The participants then took part in a think-aloud design session and created interactive prototypes of CFI using GestureCanvas. Finally, participants answered a multiple-choice questionnaire (Appendix B). The questionnaire included a modified SUS questionnaire [3] and questions on specific system features such as the PbD approach, automatic inferencing, and testing. We concluded the session with a short interview where the participants were encouraged to clarify their ratings or comments made during the task or in the survey.

Participants found the PbD approach intuitive, easy and rapid. For many participants, this was either their first VR experience or first time designing something in VR. Despite the limited experience, all participants were able to prototype and test a CFI using GestureCanvas . The average time taken by the participants to complete a CFI prototype using GestureCanvas is 8 minutes. L5 acknowledged the speed of prototyping in GestureCanvas: *"This was my first time designing anything in VR, and first when I heard I will be building hand gestures I was expecting to do a lot of steps, but it was really simple. I just had to do the hand motion, then trim, edit, add elements and play. And really it seemed to be very logical and intuitive."* Similarly, L7 remarked that she *"successfully accomplished what I wanted to create, which is good considering I am someone who never designed anything [in] VR before."*

Participants found the automatic inference of the conditionals to be helpful and time saving. For our question asking how the inference influenced the time spent in creating the conditionals, the majority of the participants responded positively (5 strongly agreed that it saved their time, 2 agreed, and 1 neutral). L2 said, *"Since I know nothing about this, VR or hands or anything, the defaults [inferred by the system] were really good."* This was also mentioned by L5 who said, *"I did not have trouble figuring out what I needed to put in the input because I could tell looking at the defaults. Sometimes it was not correct, but I could just change it if it is wrong, So I actually liked that a lot. I don't know about others, but for someone like me [novice], it was helpful."*

On-the-fly testing helped participants assess the reliability of their designed hand interaction. When L4 was considering three different gestures as competing design options, they recorded all three in a row and then used the on-the-fly testing feature to determine the option that yielded the best recognition performance.

Design Canvas was appreciated for enabling direct manipulation and spatial control in VR. L6 commented, *"I really liked that I could attach things exactly where I wanted. Say for example if I wanted something at the tip of my fingertips, I can just attach it at the tip of my fingertips."* Another participant, L4, also found direct manipulation over Design Canvas useful: *"The controls were nice, but the fact that I could grab things, move it, and put it on hands, it was awesome."*

Despite the promises, the participants also saw room for improvement in the current authoring approach of Gesture- Canvas.

Segmentation was a non-trivial task for novice participants. Almost all participants needed some help segmenting their demonstration into clips. Participants were not sure at what points they should make the cut and expressed that they would have tried to create more complex interactions if they had a clearer understanding of how to determine what should go in each clip. According to L6, "it was a little hard for me initially to understand what would go in each clip, how it would go. But maybe it was just me not being familiar with any of this at all". L3 shared a similar remark, but mentioned that they were able to understand how to segment the clips after the initial confusion.

7.2 Evaluating the Utility of GestureCanvas with Experts

For assessing the utility of GestureCanvas we conducted an interview study with 4 professional designers with experience of designing hand interactions for XR. Their design experience ranged between 3 years and 8 years.

During the study, the participants were given a video walkthrough of the workflow for prototyping the bow and arrow interaction in GestureCanvas, importing it in a game project and then making code level changes. Additionally, they were provided the Android Package Kit (APK) of the game so that they could install and play the interaction using their headset. Later, a one hour Zoom interview was conducted to get insight about the utility of the tool in their design workflow. During this time, the designers reflected on their past and present workflow of creating CFIs and provided insight on how they see GestureCanvas fit in the process.

The expert participants reported various ways of adopting the rapid workflow enabled by GestureCanvas in their current design and development lifecycle. During the design phase, they are likely to adopt this workflow for testing assumptions (T1, T2), presenting ideas to potential investors (T2, T4), developing working systems during design jams (T2), and prototyping interaction in GestureCanvas and handing it to developers for further development instead of trying to explain what they want (T1).

During the development phase the expert participants found utility in rapidly setting up the layout of the interactions in GestureCanvas to build from there (T3) and use it as a data collection environment (T2). According to T3, "*The biggest advantage is that the state management is set up for you in code, which is usually very complicated process to do in code... ...an interaction like that can have many many different states. So taking the code away from that...was the biggest advantage for me"* T2 saw the potential of the system in the process of data collection for gesture recognition - *"I always thought that there needs to be a better approach to how we collect the data... This could be very powerful for that."*

The process of creating sub-interactions by segmenting the recording fits their natural workflow of designing CFIs According to T3, *"From a professional perspective, state management is a very natural part of [an interaction] this"*. This sentiment was echoed by all four participants and they felt "at the end of the day, most interactions we design are state based." This directly contradicts with results of non-expert study, where participants were finding it difficult to understand segmentation process.

Expert participants felt that GestureCanvas will make them more creative than their current practice. T1 mentioned that with a system like GestureCanvas, "*I would be more open to playing with gestures and hand based interactions more than I currently do... it will make me more creative and that would be the biggest impact*". T2 expressed a similar sentiment by saying that GestureCanvas will *"help create like a really highly complex mode of gameplay by very simple building blocks like lego, so it comes down to your creativity when you really don't have to worry about coding stuff."* Thus, expert participants mentioned that they would be willing to explore more than what they do currently because as T3 mentioned, *"don't really like doing something in code, and then realize oh no its not the right thing and then have to redo it re iterate. I guess this multiple iterations gets me annoyed when developing...so as early on you want to be able to get that feedback if something works, whether it makes sense or not...having that quick access prototyping is very very valuable... and I think [GestureCanvas] allows us to accomplish that"*.

8 DISCUSSION

8.1 Scalability

The GestureCanvas approach presents a scalable solution for prototyping complex CFIs in VR. By offering a code-free, end-to-end workflow, the system allows for rapid creation and testing of increasingly intricate CFIs. The automatic inference of input triggers, the ability to create and modify clip transition logic, and the option to add samples for improved recognizer performance all contribute to the scalability of GestureCanvas in handling greater interaction complexity. However, it is important to note that as the number of gestures, clips, and conditionals grows, the manual effort required to refine the prototyping may increase, potentially affecting the overall efficiency of the approach.

To further enhance the scalability of GestureCanvas, several technical approaches can be employed in the future: (1) Improving machine learning algorithms for gesture recognition and trigger inference can reduce manual intervention and increase the system's capacity to handle a larger number of gestures and clips, (2) Adaptive user interfaces that intelligently organizes and presents clips, gestures, and conditionals based on the complexity or frequency of use can streamline the prototyping process for designers, (3) Automation and optimization to generate more efficient transition logic, reduce redundancies, and eliminate conflicts among gestures, clips, and conditionals, (4) Modular design and reusable components through which designers can create, share, and reuse pre-built components, such as gesture templates, transition logic, or response actions, can expedite the prototyping process and improve scalability, and (5) Collaboration and version control to support designers working together efficiently on complex projects, reducing the time and effort needed to handle an increasing number of gestures, clips, and conditionals. By incorporating these approaches, GestureCanvas can become more scalable and better equipped to handle the growing complexity of CFIs in virtual reality environments.

8.2 Flexibility vs Expressiveness

For any rapid prototyping tool, it is important to strike the right balance between flexibility and expressiveness where flexibility refers to the ability of making rapid design changes [47] and expressiveness refers to the ability of accomplishing more by expressing less [47].

While GestureCanvas allows designers to rapidly modify and test triggers and transition logic in a CFI, changing hand actions within a sub-interaction would require starting from the beginning due to its demonstration-first approach. In comparison, tools like GesturAR [58] record and stitch each sub-interaction independently making it easier to modify hand actions within a specific sub-interaction, but results in reduced expressiveness. GestureCanvas, however, has better expressivity since it significantly reduces the effort of expressing CFI interaction through semi-automatic inferencing of the input triggers of each sub-interaction in a demonstration-first approach.

8.3 Beyond CFIs and Games

GestureCanvas presents a versatile and code-free approach that can be generalized to prototyping various types of VR interactions beyond CFIs. Its core features, such as the Design Canvas, enable a user-friendly space for recording, creating, and modifying interaction sequences, making it applicable to a wide range of interaction design scenarios. The semi-automatic identification of input triggers and the on-the-fly testing capabilities can be utilized in different contexts to streamline the creation and validation of interaction chains. The inherent flexibility in GestureCanvas's workflow, which includes options to modify clip transition logic, input triggers, and system responses, allows designers to tailor interactions to specific use cases or application requirements. By incorporating these features, GestureCanvas can facilitate the prototyping and testing of diverse VR interactions, from simple gesture-based controls to complex, multi-step processes, catering to the evolving needs of the immersive design landscape.

While the examples presented in this paper (see appendix A) primarily focus on VR games, it is important to note that GestureCanvas is applicable to a wide range of application domains. In order to demonstrate the versatility of our system, we have developed an array of VR experiences encompassing gaming (Fig 10), workforce training (Fig 12), and educational contexts (Fig 11). These diverse use cases are illustrated in the appendix and showcased in our demo video, further emphasizing the generalizability of GestureCanvas across various application domains.

8.4 Limitations and Future Work

Although our system has demonstrated promising results, it is not without limitations. Firstly, the current implementation of GestureCanvas only supports a limited set of input triggers (Proximity), leaving room for exploring additional trigger types and their potential impact on the overall design process. Secondly, the accuracy of the inferencing algorithm could be improved to further streamline the prototyping workflow and reduce the manual effort required by designers. Lastly, the evaluation conducted in this study, while informative, could benefit from a more extensive investigation involving a larger and more diverse group of participants actively engaged in the domain of VR interaction design.

8.5 Conclusions

This systems research on GestureCanvas establishes a foundation for a code-free, seamless, and rapid prototyping approach to

complex free-hand interactions (CFIs). By designing, implementing, and evaluating GestureCanvas, we demonstrate the effectiveness of three interaction models: event-driven state machines, trigger actions, and programming by demonstration. We faced challenges in integrating these models into a single coherent and effective interface, and addressed these by introducing a suite of innovative design solutions, including self-recording as a Design Canvas, VR-Embodied Triggers, and automatic inference of input triggers. Our two-phase evaluation revealed that GestureCanvas offers efficient, learnable, and usable interaction features. It also supports end-to-end design in VR, allowing for the demonstration, refinement, testing, and deployment of designed CFIs in a continuous loop.

ACKNOWLEDGMENTS

This work was supported by a grant from the Korea Evaluation Institute of Industrial Technology (KEIT), funded by the Korean government (MOTIE, No. 20009940). Additional support came from the NSERC Discovery Grant and CREATE programs, as well as a generous gift from Adobe Research.

REFERENCES

[1] 2016. tvori: Designing VR & AR Simplified. https://tvori.co/. Accessed April 04, 2023.
[2] 2020. Hand pose detection: Oculus developers. Retrieved Accessed Sep 13, 2022 from https://developer.oculus.com/documentation/unity/unity-isdk-hand-pose-detection/
[3] Bill Albert and Tom Tullis. 2013. *Measuring the user experience: collecting, analyzing, and presenting usability metrics*. Newnes.
[4] Aldin. 2019. The Wizards - Dark Times. https://www.oculus.com/experiences/quest/2280285932034855. Accessed April 04, 2023.
[5] 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.
[6] Daniel Ashbrook and Thad Starner. 2010. MAGIC: a motion gesture design tool. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2159–2168.
[7] 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
[8] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.
[9] Gavin Buckingham. 2021. Hand tracking for immersive virtual reality: opportunities and challenges. *Frontiers in Virtual Reality* (2021), 140.
[10] William Buxton. 1995. Chunking and phrasing and the design of human-computer dialogues. In *Readings in Human–Computer Interaction*. Elsevier, 494–499.
[11] 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*. 521–534.
[12] Shapes Corp. 2022. ShapesXR: VR Creation and Collaboration Platform for Remote Teams. https://www.shapesxr.com. Accessed April 04, 2023.
[13] Javier M Covelli, Jannick P Rolland, Michael Proctor, J Peter Kincaid, and PA Hancock. 2010. Field of view effects on pilot performance in flight. *The International Journal of Aviation Psychology* 20, 2 (2010), 197–219.
[14] Anind K. Dey, Raffay Hamid, Chris Beckmann, Ian Li, and Daniel Hsu. 2004. A CAPpella: Programming by Demonstration of Context-Aware Applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria) *(CHI '04)*. Association for Computing Machinery, New York, NY, USA, 33–40. https://doi.org/10.1145/985692.985697
[15] Satu Elo and Helvi Kyngäs. 2008. The qualitative content analysis process. *Journal of advanced nursing* 62, 1 (2008), 107–115.
[16] Barrett Ens, David Ahlström, and Pourang Irani. 2016. Moving ahead with peephole pointing: Modelling object selection with head-worn display field of view limitations. In *Proceedings of the 2016 Symposium on Spatial User Interaction*. 107–110.
[17] Maribeth Gandy and Blair MacIntyre. 2014. Designer's augmented reality toolkit, ten years later: implications for new media authoring tools. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. 627–636.
[18] VRdirect GmbH. 2022. VRdirect: The Enterprise Virtual Reality Solution. https://www.vrdirect.com/. Accessed April 04, 2023.
[19] Uwe Gruenefeld, Jonas Auda, Florian Mathis, Stefan Schneegass, Mohamed Khamis, Jan Gugenheimer, and Sven Mayer. 2022. VRception: Rapid Prototyping of Cross-Reality Systems in Virtual Reality. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 611, 15 pages. https://doi.org/10.1145/3491102.3501821
[20] Shreyas Hampali, Sayan Deb Sarkar, Mahdi Rad, and Vincent Lepetit. 2022. Keypoint transformer: Solving joint identification in challenging hands and object interactions for accurate 3d pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11090–11100.
[21] Shangchen Han, Beibei Liu, Randi Cabezas, Christopher D Twigg, Peizhao Zhang, Jeff Petkau, Tsz-Ho Yu, Chun-Jung Tai, Muzaffer Akbay, Zheng Wang, et al. 2020. MEgATrack: monochrome egocentric articulated hand-tracking for virtual reality. *ACM Transactions on Graphics (ToG)* 39, 4 (2020), 87–1.
[22] 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*. 145–154.
[23] Ken Hinckley, Patrick Baudisch, Gonzalo Ramos, and Francois Guimbretiere. 2005. Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 451–460.
[24] Holonautic. 2021. Hand Physics Lab. https://www.oculus.com/experiences/quest/3392175350802835. Accessed Sep 13, 2022.
[25] David G Jansson and Steven M Smith. 1991. Design fixation. *Design studies* 12, 1 (1991), 3–11.
[26] Jun Kato, Takeo Igarashi, and Masataka Goto. 2016. Programming with examples to develop data-intensive user interfaces. *Computer* 49, 7 (2016), 34–42.
[27] Julian Keil, Dennis Edler, Thomas Schmitt, and Frank Dickmann. 2021. Creating immersive virtual environments based on open geospatial data and game engines. *KN-Journal of Cartography and Geographic Information* 71, 1 (2021), 53–65.
[28] 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.
[29] Ben Lafreniere, Tanya R. Jonker, Stephanie Santosa, Mark Parent, Michael Glueck, Tovi Grossman, Hrvoje Benko, and Daniel Wigdor. 2021. False Positives vs. False Negatives: The effects of recovery time and cognitive costs on input error preference. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 54–68.
[30] David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg, and Saul Greenberg. 2018. Evaluation Strategies for HCI Toolkit Research. 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, 1–17. https://doi.org/10.1145/3173574.3173610
[31] David Ledo, Jo Vermeulen, Sheelagh Carpendale, Saul Greenberg, Lora Oehlberg, and Sebastian Boring. 2019. Astral: Prototyping Mobile and Smart Object Interactive Behaviours Using Familiar Applications. In *Proceedings of the 2019 on Designing Interactive Systems Conference* (San Diego, CA, USA) *(DIS '19)*. Association for Computing Machinery, New York, NY, USA, 711–724. https://doi.org/10.1145/3322276.3322329
[32] Gun A Lee, Claudia Nelles, Mark Billinghurst, and Gerard Jounghyun Kim. 2004. Immersive authoring of tangible augmented reality applications. In *Third IEEE and ACM international symposium on mixed and augmented reality*. IEEE, 172–181.
[33] 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
[34] Germán Leiva, Cuong Nguyen, Rubaiat Habib Kazi, and Paul Asente. 2020. Pronto: Rapid Augmented Reality Video Prototyping Using Sketches and Enaction. 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/10.1145/3313831.3376160
[35] Henry Lieberman, Fabio Paternò, and Volker Wulf. 2006. *End user development*. Vol. 9. Springer.
[36] DraftXR LLC. 2020. DraftXR: Design, Test, & Share VR Interfaces. https://www.draftxr.com/. Accessed April 04, 2023.

[37] Xiaolong Lou, Xiangdong A Li, Preben Hansen, and Peng Du. 2021. Hand-adaptive user interface: improved gestural interaction in virtual reality. *Virtual Reality* 25 (2021), 367–382.
[38] Hao Lü and Yang Li. 2012. Gesture coder: a tool for programming multi-touch gestures by demonstration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2875–2884.
[39] 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*. 257–266.
[40] Magnopus. 2020. Elixir. https://www.oculus.com/experiences/quest/3793077684043441. Accessed Sep 13, 2022.
[41] George B. Mo, John J Dudley, and Per Ola Kristensson. 2021. Gesture Knitter: A Hand Gesture Design Tool for Head-Mounted Mixed Reality Applications. 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 291, 13 pages. https://doi.org/10.1145/3411764.3445766
[42] 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
[43] Michael Nebeling, Katy Lewis, Yu-Cheng Chang, Lihan Zhu, Michelle Chung, Piaoyang Wang, and Janet Nebeling. 2020. XRDirector: A Role-Based Collaborative Immersive Authoring System. 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, 1–12. https://doi.org/10.1145/3313831.3376637
[44] 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
[45] 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
[46] 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, 333–337.
[47] Dan R Olsen Jr. 2007. Evaluating user interface systems research. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*. 251–258.
[48] Thammathip Piumsomboon, Adrian Clark, Mark Billinghurst, and Andy Cockburn. 2013. User-defined gestures for augmented reality. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*. 955–960.
[49] Meta Quest. 2021. Tiny Castles. https://www.oculus.com/experiences/quest/3647163948685453. Accessed Sep 13, 2022.
[50] David Rempel, Matt J Camilleri, and David L Lee. 2014. The design of hand gestures for human–computer interaction: Lessons from sign language interpreters. *International journal of human-computer studies* 72, 10-11 (2014), 728–735.
[51] Jaime Ruiz and Yang Li. 2011. DoubleFlip: a motion gesture delimiter for mobile interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2717–2720.
[52] Maximilian Speicher and Michael Nebeling. 2018. GestureWiz: A Human-Powered Gesture Design Environment for User Interface Prototypes. 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, 1–11. https://doi.org/10.1145/3173574.3173681
[53] Carbon Studio. 2022. The Wizards - Dark Times. https://www.oculus.com/experiences/rift/2599598633394538. Accessed Sep 13, 2022.
[54] Miru Studio. 2022. Finger Gun. https://www.oculus.com/experiences/quest/5870383889703320. Accessed Sep 13, 2022.
[55] Eugene M. Taranta II, Amirreza Samiei, Mehran Maghoumi, Pooya Khaloo, Corey R. Pittman, and Joseph J. LaViola Jr. 2017. Jackknife: A Reliable Recognizer with Few Samples and Many Modalities. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. ACM, 5850–5861. http://doi.acm.org/10.1145/3025453.3026002
[56] void room. 2022. Tea For God - Demo. https://www.oculus.com/experiences/quest/3762343440541585/. Accessed Sep 13, 2022.
[57] Bryan Wang and Tovi Grossman. 2020. BlyncSync: enabling multimodal smartwatch gestures with synchronous touch and blink. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–14.
[58] 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
[59] Tianyi Wang, Xun Qian, Fengming He, Xiyun Hu, Ke Huo, Yuanzhi Cao, and Karthik Ramani. 2020. CAPturAR: An Augmented Reality Tool for Authoring Human-Involved Context-Aware Applications. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 328–341. https://doi.org/10.1145/3379337.3415815
[60] Pavel Zahorik and Rick L Jenison. 1998. Presence as being-in-the-world. *Presence* 7, 1 (1998), 78–89.
[61] Bruno Zamborlin, Frederic Bevilacqua, Marco Gillies, and Mark D'inverno. 2014. Fluid gesture interaction design: Applications of continuous recognition for the design of modern gestural interfaces. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 3, 4 (2014), 1–30.
[62] Baowen Zhang, Yangang Wang, Xiaoming Deng, Yinda Zhang, Ping Tan, Cuixia Ma, and Hongan Wang. 2021. Interacting two-hand 3d pose and shape reconstruction from single color image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 11354–11363.

A APPLICATION SCENARIOS

In the paper, we explained in detail how to use GestureCanvas for prototyping a bow and arrow CFI. In this section, we use GestureCanvas to prototype CFIs for three different domains.Please refer to supplementary materials for video demo of these examples.

A.1 CFI for a Gaming Experience

We prototyped a wizard attack CFI (Fig 10) involving 8 sub-interactions and 11 transitions including 3 branches using GestureCanvas and imported it into a game. The attack interaction requires players to hold their hands with palms facing each other to make a magic spell appear between them (Fig 10 (a)). Detecting this hand action requires transform-constrained hand pose along with proximity trigger. Once the spell appears, the player can grab it to have a magic ball in each hand (Fig 10 (b)). At this point, the user can attack with the right hand first and then the left hand or vice versa. The attack is done by pushing either of the hands forward (Fig 10 (c)) and hence uses dynamic gestures as triggers. Thus, this CFI contains both bimanual and unimanual sub-interactions. The response is authored by anchoring the fireball on the Design Canvas and using the Design Canvas to place the magic spell in relative position of the hands.

A.2 CFI for an American Sign Language (ASL) Learning Experience

We prototyped a bimanual CFI for ASL learning experience (Fig 11) with 4 sub-interactions and 3 linear transitions in GestureCanvas. In this experience the user needs to perform a certain hand action (corresponding to a word or phrase) to advance to the next sub-interaction in the sequence. Since hand location is important in ASL, triggers in this CFI include proximity event trigger for hand location along with the combinations of dynamic hand gesture and static hand pose. For example, the trigger in the first sub-interaction is the AND condition between proximity (relative position of hands to the user), left hand static pose (transform constrained) and right hand dynamic gesture.

A.3 CFI for an Infant Cardiopulmonary resuscitation (CPR) Training Experience

We prototyped a CFI for an Infant CPR Training Experience (Fig 12) with a loop between 2 sub-interactions. Both sub-interactions are

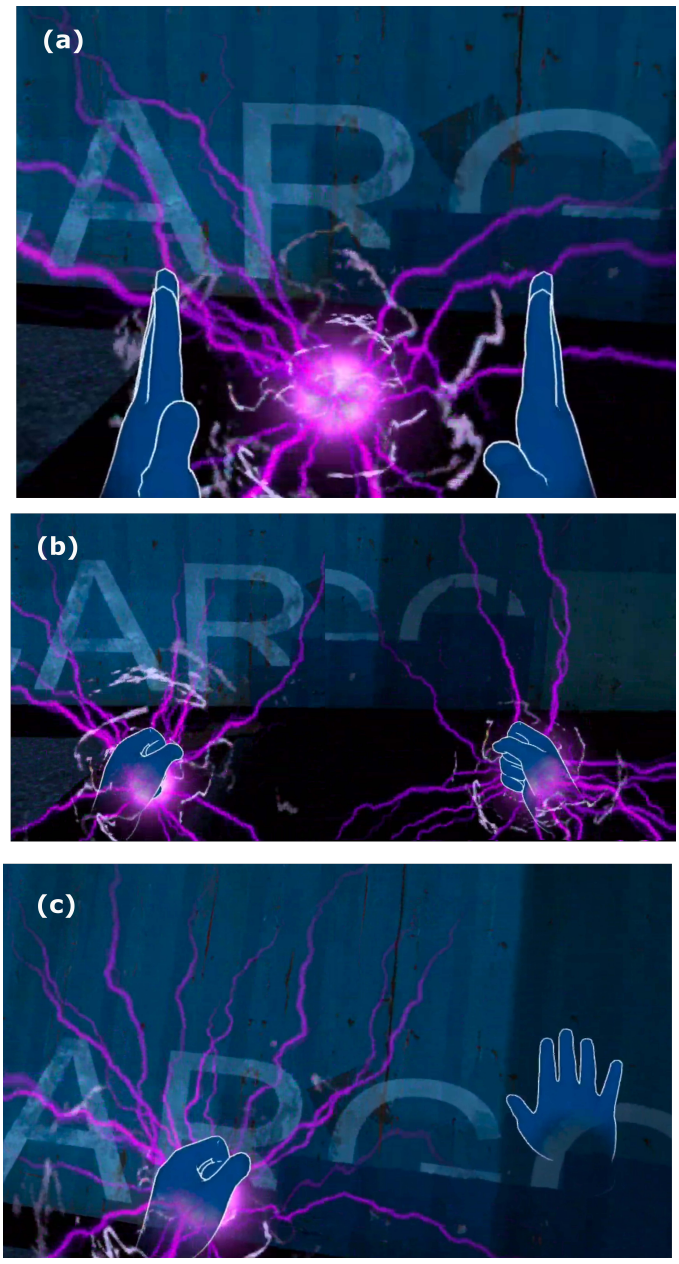

Figure 10: CFI prototyped in GestureCanvas and deployed in a gaming experience: (a) player summoning magic spell between hands where the transform-constrained hand pose with proximity is used as trigger, (b) player grabbing onto the magic ball and (c) player attacking with right hand dynamic gesture but still holding onto the magic ball in left hand

unimanual in nature where the first sub-interaction in the sequence uses AND condition between proximity (touch) and dynamic gesture as trigger to transition to the second sub-interaction, which again loops back to the first when the proximity (touch) is FALSE. The action of giving CPR requires giving 30 compressions. Hence,

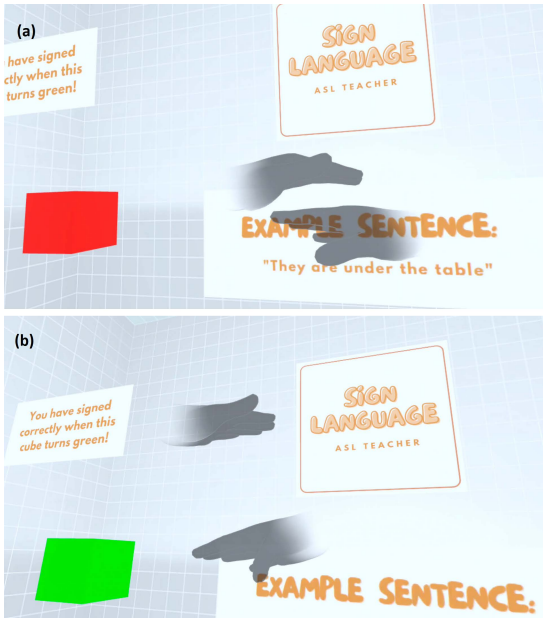

Figure 11: CFI prototyped in GestureCanvas and deployed in an ASL Learning Experience where proximity triggers are used to distinguish between (a) wrong and (b) correct hand location.

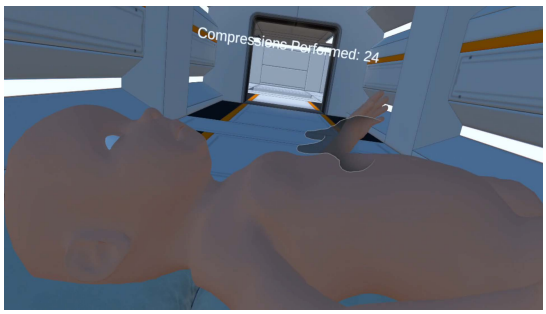

Figure 12: CFI prototyped in GestureCanvas and deployed in an Infant CPR Training Experience. Counter was added post hoc via code level editing using our toolkit

we create the states, the looping transition and the hand based triggers in GestureCanvas and import it in a Unity project, where we use our toolkit to keep count of the number of compressions performed by counting the number of state transitions and stop looping once 30 compressions has been performed.

B QUESTIONNAIRE

The questionnaire items used for evaluating the usability of GestureCanvas with non-experts are listed below. Participants rated each item in the questionnaire on a 5-point Likert scale, ranging from Strongly Disagree to Strongly Agree.

- I found the programming by demonstration approach of prototyping hand interaction easy to use

- I found the system defaults of input triggers a good starting point
- Creating Interactive Prototypes of Hand interaction using GestureCanvas took a lot of effort
- GestureCanvas was useful in understanding the performance of my hand interaction design
- GestureCanvas would be useful for early design stage prototyping
- If I were a VR designer, I think that I would like to use this prototyping tool frequently.
- I found this prototyping tool unnecessarily complex.
- I thought this prototyping tool was easy to use.
- If I were a VR designer, I think that I would need the support of a technical person to be able to use this prototyping tool.
- I found the various functions in this prototyping tool were well integrated.
- I think there was too much inconsistency in this prototyping tool.
- I would imagine that most people would learn to use this prototyping tool very quickly.
- I found this prototyping tool very cumbersome to use.
- I felt very confident using this prototyping tool.
- I needed to learn a lot of things before I could get going with this prototyping tool.