Multiverse Architecture for Desired Realities

Andreas Fender
VISUS
University of Stuttgart
Stuttgart, Germany
andreas.fender@visus.uni-stuttgart.de

Dieter Schmalstieg
VISUS
University of Stuttgart
Stuttgart, Germany
Graz University of Technology
Graz, Austria
dieter.schmalstieg@visus.uni-stuttgart.de

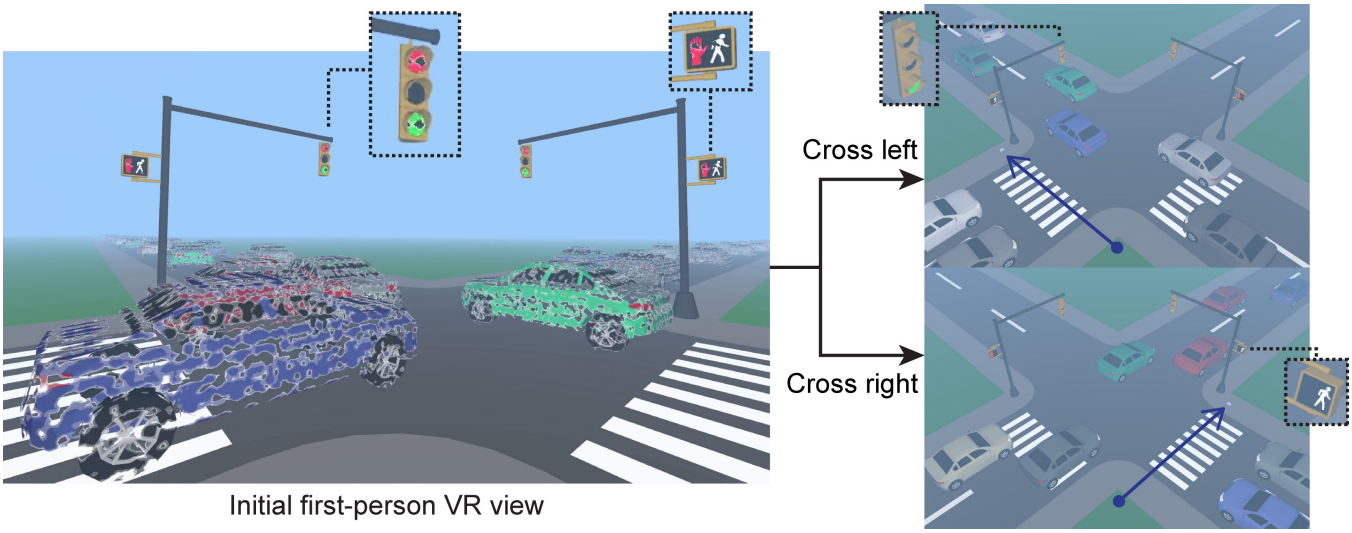
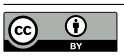


Figure 1: A minimal example of a Desired Reality. Conventionally, whether one can cross the desired direction when arriving at a crossroad at a random instance in time depends on an *observed* state of reality. In contrast, in a Desired Reality, observation does not lead to a collapse of possibilities into a single reality (left). Hence, green and red are equally possible for pedestrians and cars in all directions. Depending on the side the user crosses, the two possibilities collapse into a single reality (right).

Abstract

Many previous works that deal with uncertainty, such as touch or text input, resolve imprecise or ambiguous input via probabilistic models, aiming to hide ambiguities or letting users explicitly select from a number of suggestions. In this paper, we introduce a multiverse architecture for Desired Realities, a new type of reality that, instead of hiding ambiguities and uncertainty, makes them a central part of immersive experiences and interactions. Our multiverse architecture is a generalization of conventional universe systems and enables applications to seamlessly split into a multiverse at runtime whenever needed, to later collapse into a universe again. Undesired universes are removed implicitly—users simply interact with their desired universe without resorting to explicit disambiguation.

This tech-concept paper forms the foundation for technical as well as user-centered multiverse research, while inviting readers to rethink conventional (universe) application logic.



UIST '26, Detroit, MI, USA

ACM ISBN 979-8-4007-2856-3/2026/11
https://doi.org/10.1145/3830398.3830478

CCS Concepts

• **Human-centered computing** → **Virtual reality**; *Mixed / augmented reality*.

Keywords

Input uncertainty, Multiverse, Virtual Reality

ACM Reference Format:
Andreas Fender and Dieter Schmalstieg. 2026. Multiverse Architecture for Desired Realities. In *The 39th Annual ACM Symposium on User Interface Software and Technology (UIST '26), November 02–05, 2026, Detroit, MI, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3830398.3830478

1 Introduction

Immersive environments, no matter whether they simulate real environments or are fully fictional, conventionally obey many of the laws of physical reality. However, designers of virtual environments may choose to ignore some of the laws of physics or provide unrealistic capabilities to the user (e.g., "superpowers" [8, 13, 33, 34]). Deliberately 'ignoring' certain limits of physics, while still retaining a sense of a cohesive reality, can have different goals, such as enhancing intuitive human collaboration across space [30, 32, 47, 50], across time [6, 11], or across different virtual realities [48]. An often

overlooked, but even more fundamental law than space-time is the *effect of observing reality itself*. In physical reality, humans are aware of the many possible outcomes of real events—before or even without experiencing them. Similarly, humans can imagine the possible states of entities before having seen them. However, once *observed*, we are used to experiencing a *single* reality. This understanding is also grounded in common interpretations within quantum mechanics, which (depending on the specific interpretation or theory) hold that observing physical entities leads to a collapse of all possibilities into a single experienced reality, while other possibilities that *could have been* are deferred to a parallel universe [3, 28, 38], or simply cease to exist. This fundamental aspect of reality is mostly unquestioned even in fully fictional artificial realities. However, in such artificial realities, there is in fact no reason why possibilities must collapse into a single reality once observed.

In this work, we challenge the assumption that observation (by users) leads to a collapse into a single reality and present an architecture designed around this premise. As an alternative trigger for collapsing into a single reality, we use *desire*—implicitly expressed through the actions a user performs when presented with multiple possibilities. Figure 1 shows a minimal example of what we call a *Desired Reality*. The example argues that, before arriving at a crossroad (i.e., before observing it) and without prior knowledge, any direction could have a green light for cars or pedestrians. In conventional realities[1], a single direction has green light when observing the crossroad upon arrival, so one of the possibilities becomes reality. However, with a Desired Reality, observation is not enough. Only when the user expresses a desire by crossing in one direction, the corresponding traffic light signals become real, as the user *must* have green. Generalizing from this simple example, we present and expand upon the Desired Reality concept [12]. Specifically, beyond the concept, this paper presents a *multiverse architecture* for immersive Desired Reality experiences. At the core of our architecture are *possibility trees*, which streamline the process of handling multiple co-existing or competing possibilities. Using our architecture, we implemented several illustrative examples that showcase the concept through specific instances of possibility trees. Taken together, we contribute:

- The concept of Desired Realities as a new paradigm for resolving uncertainty through post-hoc interaction, thereby embracing ambiguity as a part of the user experience.
- An architecture and reference implementation for building Desired Realities.
- A set of implemented examples with specific possibility trees to showcase different experiences enabled by our architecture.

2 Related work

Our work builds upon two avenues of research: Uncertain or ambiguous input as well as multiverse simulations.

2.1 Uncertain or Ambiguous Input

Conventionally, the goal of systems that deal with uncertain input is to resolve ambiguities as quickly as possible, i.e., during or immediately after the ambiguous action [16, 41]. This means that ambiguity is either hidden from users or users can explicitly select different options, such as word suggestions when using text input [45]. For example, Mankoff et al. [27] focus on explicitly resolving ambiguities after the uncertain input (pen input in their case) and investigate different ways of displaying and explicitly selecting the correct input afterward. Zhu et al. [54] hide ambiguity and resolve it internally using Bayesian methods. Researchers also developed frameworks that provide support for resolving uncertain input, in particular for 2D user interfaces and 2D gestures [5, 24, 25]. Most notably, the motivation of Schwarz et al. [40] is similar to ours. They aim to resolve uncertainties not as quickly as possible but during the direct follow-up interaction (in their case within a 2D UI context). Later work by Schwarz et al. [42] introduced an architecture to handle and visualize uncertainty in 2D UIs.

Overall, previous toolkits for uncertain input were specifically designed for 2D user interfaces, meaning that they focus on typical UI widgets and components. Our concept generalizes from uncertain menu input and allows to propagate uncertainty to and from other aspects of applications, such as physics and 3D interaction.

Multimodality. Input based on different modalities, such as gaze and speech, can be useful as a natural, complementary input. Bolt's 'Put-that-there' approach [4] utilizes multiple modalities at once. In isolation, different modalities can be error prone [18, 26], which is why they are typically combined for the purpose of disambiguation [1, 46, 51, 53]. For instance, Kaiser et al. [20] combine 3D input with speech and gaze input for disambiguation in AR/VR.

To resolve an ambiguous action, our aim is to use interactions that are separated from the ambiguous action, which is conceptually similar to multimodal approaches. However, we use *follow-up actions* instead of simultaneous actions, meaning that those actions are *temporally* separated and not necessarily by their type of input. Hence, the ambiguous action can be of the same modality as the follow-up action or their modalities can differ.

2.2 Multiverse Simulations

In the context of our work, we interpret anything that runs in multiple instances with a similar start condition as a 'multiverse' (or 'parallel universes'). For instance, network systems that aim to reduce the experienced latency (e.g., *rollback netcode* [43]) run multiple game instances and therefore qualify as multiverses in our context. Given this terminology, several previous works have explored the use of multiverses to simulate variations of the starting conditions, such as tools for running multiple instances of statistical tests to improve robustness, reproducibility, and transparency [17, 23]. For example, the 'multiverse analysis' by Dragicevic et al. [9] allows the reader to switch between multiple possible statistical tests within the output document (i.e., changing the contents about the used tests and their results). Sarma et al. [36] run multiple instances of data analysis tasks in *R* while reducing redundant data and processing. Similar concepts about incorporating uncertainty can be applied to other types of analysis or simulation [15, 37]. Schindler et al. [39] let analysts control multiple data flow scenarios in a multiverse for a comparative analysis of time-dependent data. Waser et al. [49] use similar concepts to deal with uncertainty in flood simulations. Multiverses can even represent real-world entities, such as physical entities in multiversal digital twins [7, 35].

[1]With "conventional realities", we refer to both, physical reality as well as artificial realities that partially mimic physical reality (e.g., virtual reality with single universe).

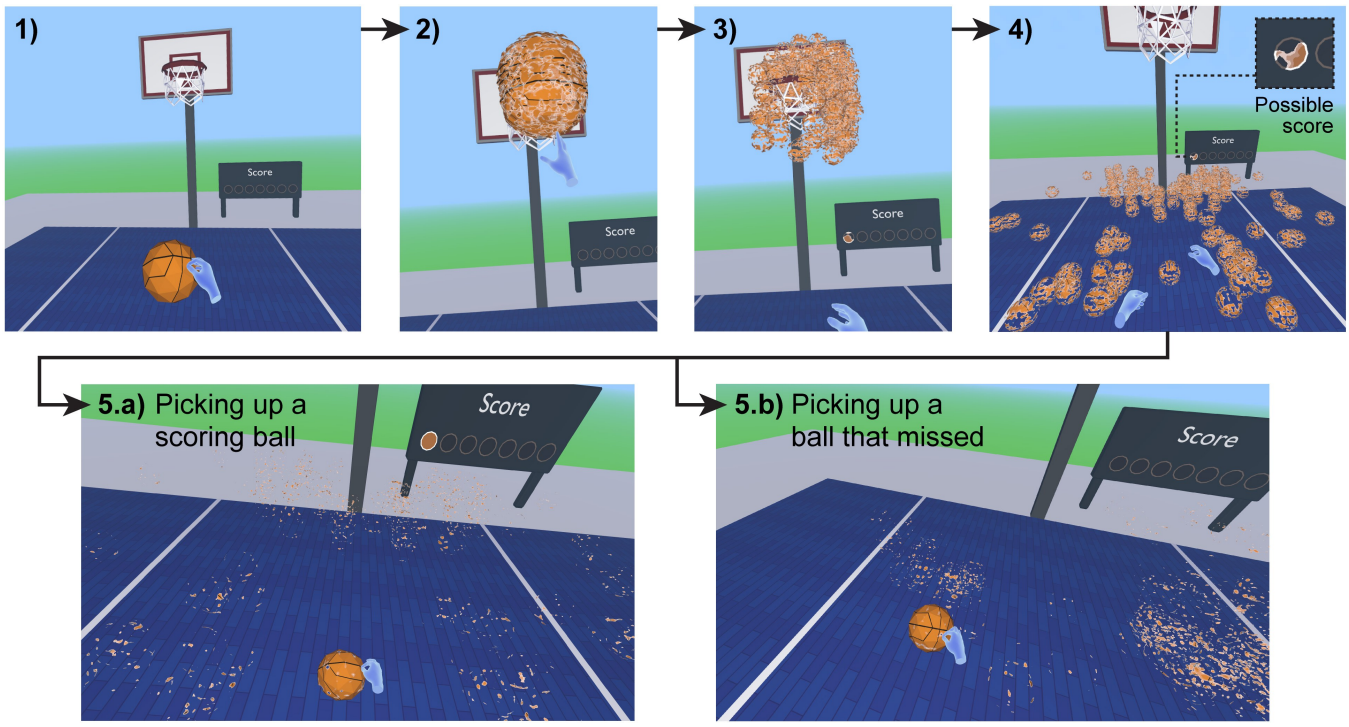


Figure 2: Simple example of a physics-based Desired Reality. 1: The user stands on a virtual basketball field holding a ball. 2: When thrown towards the basket, the ball splits into multiple possible balls, each with slightly different movement parameters, leading to different trajectories. 3: Some balls pass through the basket. 4: Scoring and not scoring are both possible (semi-visible point on score display). 5: Whether or not the user scored depends on whether they pick up a ball whose trajectory previously passed through the basket (see score displays in 5.a and 5.b). In both cases, all other balls are impossible and hence vanish.

Overall, multiverses have been used in many different contexts and in different forms, but have rarely been used as an integral part of the user experience in interactive realtime applications, i.e., visualized inside a 3D environment and for the purpose of resolving ambiguities through post-hoc interaction.

3 Illustrative Examples

To illustrate our concept, we implemented several examples [12], each focusing on one typical aspect of realtime VR applications and games, namely, *logic*, *physics*, *menus*, and *gestures*. This section briefly describes our examples, while subsequent sections use them for discussion and for exemplifying the architecture.

3.1 Crossroad Example

(demonstrates: *application logic*)

The aforementioned *Crossroad* example (Figure 1), represents a minimal use case of Desired Reality. The user stands at the corner of a virtual crossroad. Depending on which side the user chooses to cross (thereby expressing a desire), the respective possibility becomes crisp[2]. This minimal example only features two concurrent universes without involving a split into possibilities based on user actions. We demonstrate the latter in the subsequent examples.

[2]In this paper, "crisp" refers to fully existing now or in the past without having alternative versions within the multiverse.

3.2 Basketball Throw Example

(demonstrates: *physics, retrocausality*)

Our second example is about a throwing action inside a VR environment [14]. It involves splitting a crisp object into many similar possible objects (Figure 2). Specifically, the user picks up a basketball (Figure 2.1) and throws it toward a basket (Figure 2.2). When throwing, multiple possible basketballs spawn (150 in this example), each with slightly different movement parameters, leading to different trajectories. Some of those balls pass through the basket (Figure 2.3), which means that it is *possible* but not certain that the user scored (indicated by a semi-filled point on the scoreboard in Figure 2.4). When the user picks up a basketball, this ball becomes crisp, and all other balls vanish (Figure 2.5). If the user picks up a ball that has passed through the basket, the score becomes crisp (Figure 2.5a). Otherwise, it vanishes (Figure 2.5b). Hence, the *earlier* scoring event is affected by a *later* action (picking up a ball), which can be seen as a *retrocausal interaction*.

3.3 Settings Menu Example

(demonstrates: *uncertain menu input*)

We implemented a minimal 2D user interface in which two buttons right next to each other summon two different pop-up menus in mid-air (one with buttons, one with sliders, Figure 3). If there is no uncertainty (i.e., the user clearly presses one of the buttons), then

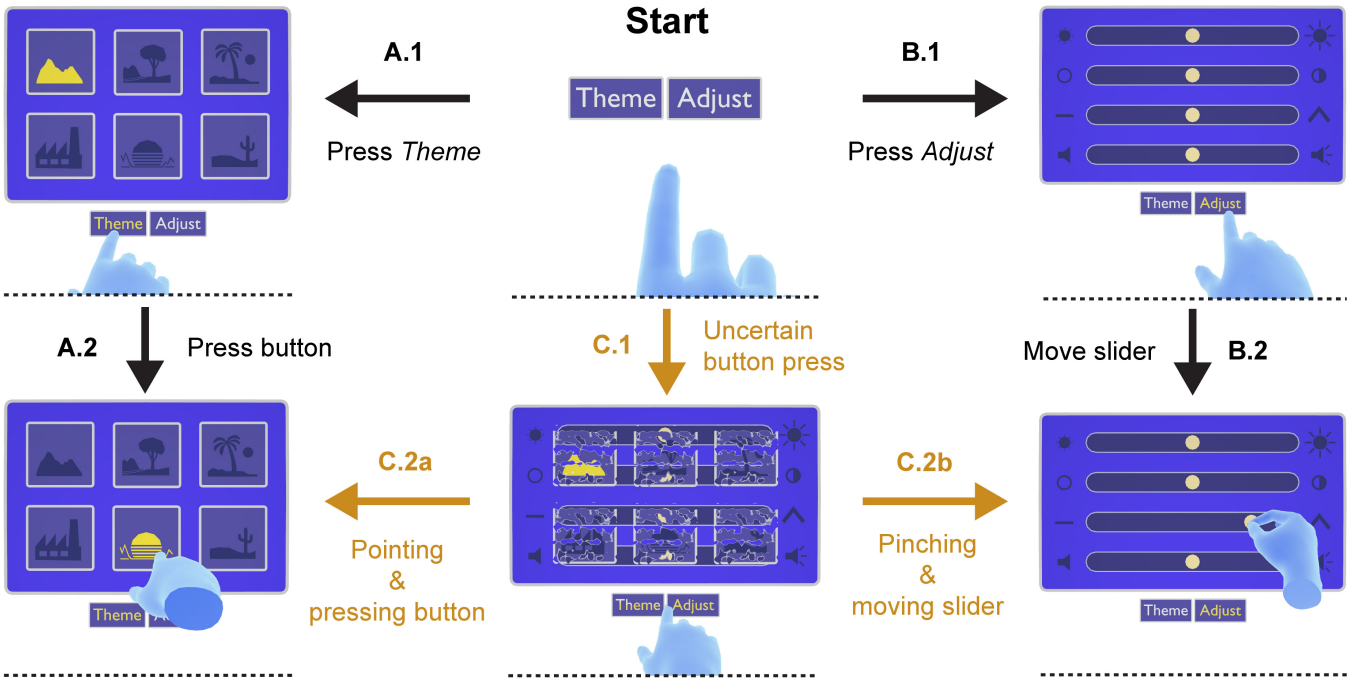


Figure 3: Minimal example of crisp (A & B paths) versus uncertain (C paths) menu interaction. Start: Two buttons located next to each other summon two different menus, namely, 'Theme' and 'Adjust'. If the button presses are clear, the respective menu is opened and used in a crisp way (paths A & B). Path A: Pressing 'Theme' opens a menu with buttons, and the user presses one of them. Path B: Pressing 'Adjust' opens a menu with sliders, and the user pinches one of them. Paths C: Due to inaccuracies (either by tracking inaccuracies or the user not aiming precisely) it is not certain which menu the user wanted to summon. Hence, both menus are possible (C.1). Depending on the user's follow-up action—pointing gesture to press a button (C.2a) versus pinching a slider (C.2b)—our system resolves post-hoc which menu the user wanted to open in the beginning. Hence, even with the uncertainty, the amount of actions is the same as with crisp interaction, since no explicit error recovery is required.

an ordinary menu interaction is used (black arrows in Figure 3). If it is uncertain which of the two buttons was pressed, e.g., due to tracking inaccuracies or imprecise movements by the user, our system superimposes both possible menus. The user continues to interact with the intended menu while ignoring the other one, i.e., the user either presses buttons or moves sliders. The type of interaction, i.e., pointing gesture for a button press (Figure 3.C.2a) versus pinch gesture for a slider (Figure 3.C.2b), can be used to determine post-hoc which menu the user intended to open. Hence, the desired menu becomes crisp, while the other menu vanishes.

3.4 Sorcery Example

(demonstrates: *uncertain gestures, sequences of possible actions*)

Every example so far featured a single ambiguity, which was resolved through a single direct follow-up action. In the following, we demonstrate a slightly more complex *Sorcery* example, which revolves around gestural interaction sequences with nested uncertainties (Figure 4). In this example, the user is standing at a sorcerer's desk (Figure 4.1) with four objects: two potions (red and blue), a green magic wand for summoning plants, and a red magic wand for summoning creatures. The user wants to summon a specific magical entity. First, the user tries to pick up a wand with the right hand. However, the blue potion and the two wands are so close to each other that the system cannot reliably determine which of them the user meant to pick up. Hence, all three are superimposed in the user's hand (Figure 4.2). The user stretches the right arm forward (Figure 4.3). This gesture is typical for using a magic wand, which implies that the user wanted to pick up one of the wands and *not* the potion. Hence, the potion is now crisp in its original location on the table, as it was never actually picked up (Figure 4.3 bottom). The two wands summon different magical entities, but they do so using the same gesture, namely, a circular motion (Figure 4.4). The creature wand has an additional function: Pointing towards a surface for a short time summons a small creature attached to that surface. Since the user sketched the imaginary circle somewhat slowly, the system does not yet know whether it was a circular gesture or a pointing gesture. Therefore, three possible entities appear in total: a plant, a creature in mid-air (Figure 4.4 center), and a creature attached to the wall (Figure 4.4 top). The user simply ignores the unintended creature on the wall and proceeds by pinching (with the left hand) within the region of the plant and the flying creature. This pinch action implies that the creature on the wall was not intended, and consequently it vanishes. Still, both magic wands and the remaining summoned entity of each are possible, because all movements so far are equally supported by both wands. Hence, the wands' and entities' existence depend on one last action. If the user

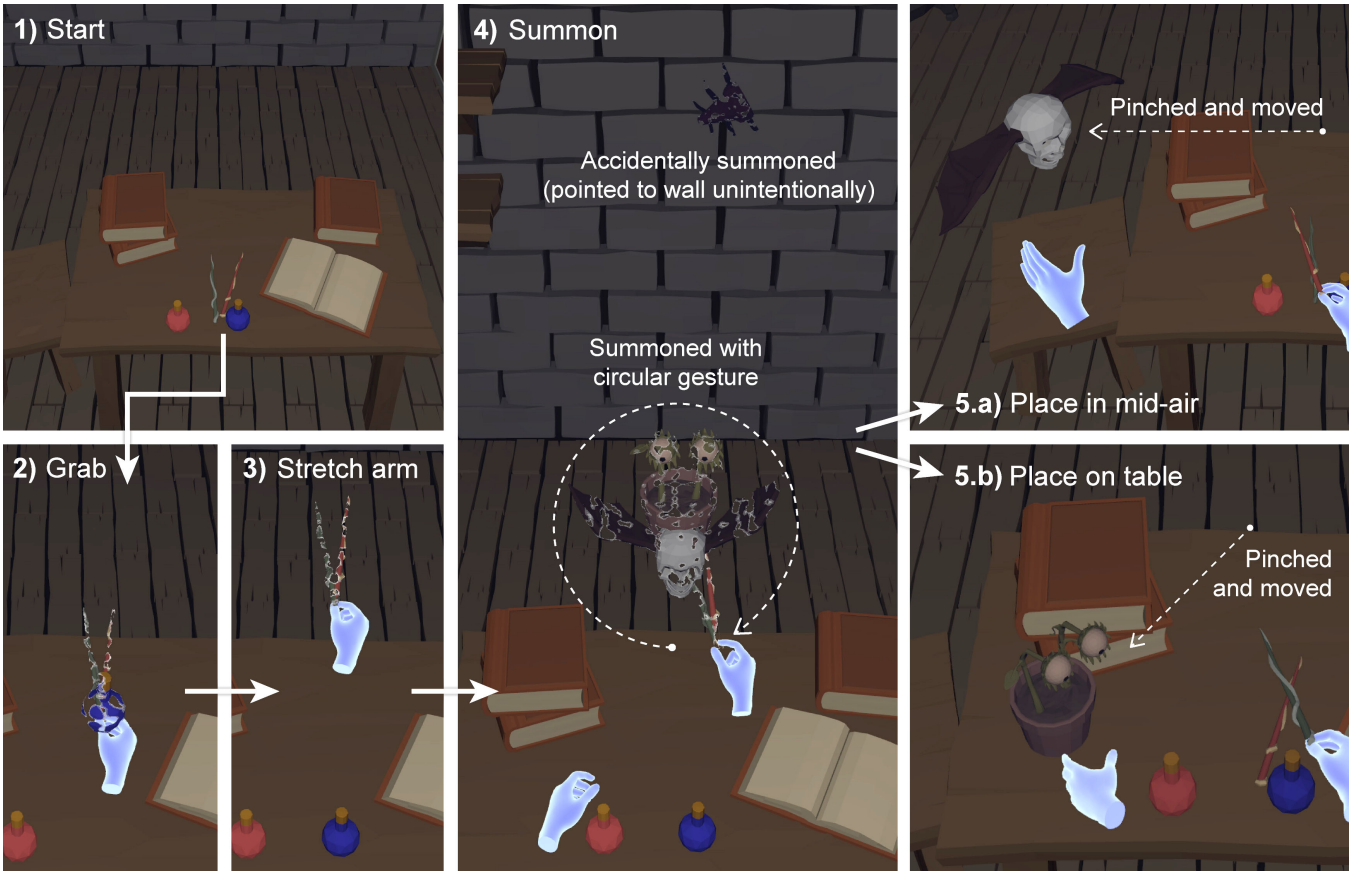


Figure 4: Example of a sequence of different types of gestures and actions. The user wants to summon a magical entity. 1: The user is standing in front of a table with two magic wands and two potions. The red wand is for summoning creatures and the green wand for plants. Both wands support a circular motion gesture for summoning a magical entity. The red wand can additionally summon an entity by pointing at surfaces for a few seconds. 2: The user wants to grab a magic wand, but because of input uncertainty, the other wand and one potion are grabbed as well. 3 & 4: The user simply continues and does a magic wand gesture for summoning an entity. Besides the intended entity, two additional unintended entities are summoned due to a misclassified additional gesture and the possible alternative wand. 5: The user simply ignores the unintended entities and interacts with the intended entity by placing it in the air (5.a) or on the table (5.b).

releases the pinch in the air—an action that does not make sense for the plant—the flying creature becomes crisp and only the red wand is held (Figure 4.5.a). In contrast, when the pinch is released close to a surface, the plant turns crisp and only the green wand is held (Figure 4.5.b). After this final step, no more ambiguities remain, i.e., all objects and the magical entity are now crisp.

3.5 Summary and Implications of Examples

The examples were about interacting with multiple possibilities, as opposed to hiding ambiguities. Due to the post-hoc interpretation, a single universe emerges simply by acting as if only this one universe existed in the first place. One implication of such *acting in the desired universe* (while ignoring undesired ones) is that false positives do not necessarily require error recovery by the user—as they simply continue interacting inside the 'true positive' universe. Hence, users can navigate into their desired reality *implicitly* instead of choosing options *explicitly*. A central goal of this principle is that users retain their agency when implicitly resolving uncertainty without significantly changing how they would act in a universe.

The examples showcase various types of ambiguity. Some uncertainties could arise due to inaccurate tracking or imprecise gestures of the user (*Settings Menu* example and *Sorcery* example), especially within potentially cluttered environments and with hard-to-reach targets. For the sake of clarity, the targets in the *Settings Menu* example and in the *Sorcery* example are directly in front of the user. However, in practice, the targets to press or pick up could also be in the periphery or even outside the field of view [44], which means that movements can be even more challenging to track or that users fully rely on proprioception. Other forms of imprecision and ambiguity can be imprecise motions or tracking inaccuracies during throwing movements (*Basketball Throw* example), or can be a deliberate part of the experience (*Crossroad* example).

4 Architecture

Our architecture is heavily inspired by theories about quantum mechanics. While we are not using such theories as underlying models in our system, we aim to simulate a reality in which rules of quantum mechanics are altered—in particular, removing the effect of observation and allowing persistence of multiple possibilities.

A trivial solution for implementing the Desired Reality concept is to run every single universe as an independent instance, while replicating user input in each. However, this 'brute force' solution does not scale well. The amount of computational resources would grow quickly, because many unnecessary duplicate objects and processes would be created. At the same time, an architecture should enable an implementation paradigm, in which logical entities do *not* require 'multiverse knowledge'. Hence, for the sake of encapsulation, a virtual entity's internal logic should not need to take other universes into account when reading user input and accessing resources. We therefore propose an architecture in which *not* multiple universes but *possibilities* comprise the primary data structure (see Figure 5). Universes are instead implicit and can be recovered from the relationships between possibilities. The remainder of this section describes the main parts of our architecture as well as specific examples to demonstrate its capabilities [22].

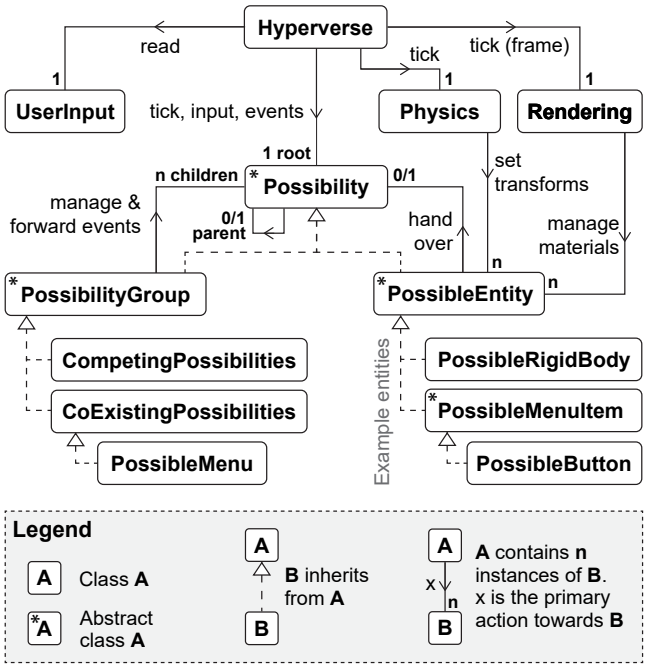


Figure 5: Overview of our Desired Reality system architecture. The core part of our system is a tree of possibilities. The Hyperverse acts as the main controller and handles high-level tasks, such as managing the possibility tree and synchronizing it with the external physics engine.

4.1 The Hyperverse

The main controller of our architecture is called Hyperverse (Figure 5 top), a singleton that interfaces with UserInput, Physics, Rendering, and the root of a tree of Possibility instances, i.e., the *possibility tree*. The primary task of the Hyperverse is event dispatching. It provides a multiversal clock and handles the multiversal dataflow. The dataflow ensures that possibilities receive the right callbacks and resources at the right time. A Desired Reality application needs to provide the possibilities and their hierarchical grouping to the Hyperverse. The possibilities can be a mix of built-in and custom subclasses. We provide more details about possibilities and the tree structure in the next subsection.

4.2 Possibilities

Application-specific event loops (logic, entity behaviors, and so on) are encapsulated in *possibilities*, which are the essential components of our architecture. A possibility can be an object, an event, or can be a group of sub-possibilities. They can either be a result of the interplay between the user's action and the application logic, or they can exist a priori (e.g., *Crossroad* example). There are two central categories of possibilities, namely, *possible entities* and *possibility groups*, both of which we describe in the following.

Possible Entities. Instances of PossibleEntity subclasses comprise the core of the application by providing behaviors, visual appearances, etc. They can contain object instances to be rendered. Our reference implementation includes built-in entities, which can be instantiated directly, such as menu elements or wrappers for rigid bodies. The application can also provide custom subclasses of PossibleEntity to implement application-specific behaviors (e.g., Crossroad cars or the magic wands in the *Sorcery* example).

A PossibleEntity can be turned inactive. This means that its resources are released and its event loop stops. Importantly, turning a PossibleEntity inactive is not the same as *deleting* it. An inactive possibility remains in the possibility tree, as it *existed or might have existed in the past*, but a deleted possibility (i.e., any sub type of Possibility) retrospectively *never existed*. This distinction is important when recovering the single crisp universe that emerges over time from the possibility tree, including its past chain of events.

Instances of PossibleEntity can *hand over* to another single Possibility (see Figure 5 right), meaning that all callbacks are forwarded to this possibility. For instance, the single basketball in the *Basketball Throw* example turns inactive and hands over to a new CompetingPossibilities instance when splitting into multiple possible basketballs (Figure 6.1–2).

Possibility Groups. Subclasses of PossibilityGroup have an arbitrary number of child-possibilities. This means that PossibilityGroup instances are what leads to the tree-like structure of possibility relationships. The Hyperverse stores a reference to the root possibility, from where all possibilities can be reached. Two classes inherit from PossibilityGroup, namely, CoExistingPossibilities and CompetingPossibilities. All children of a CoExistingPossibilities instance can interact with each other. For instance, two separate possible balls can collide if they are co-existing. In contrast, children of a CompetingPossibilities instance do not interact at all, such as the competing possible balls that pass through each other in the *Basketball Throw* example.

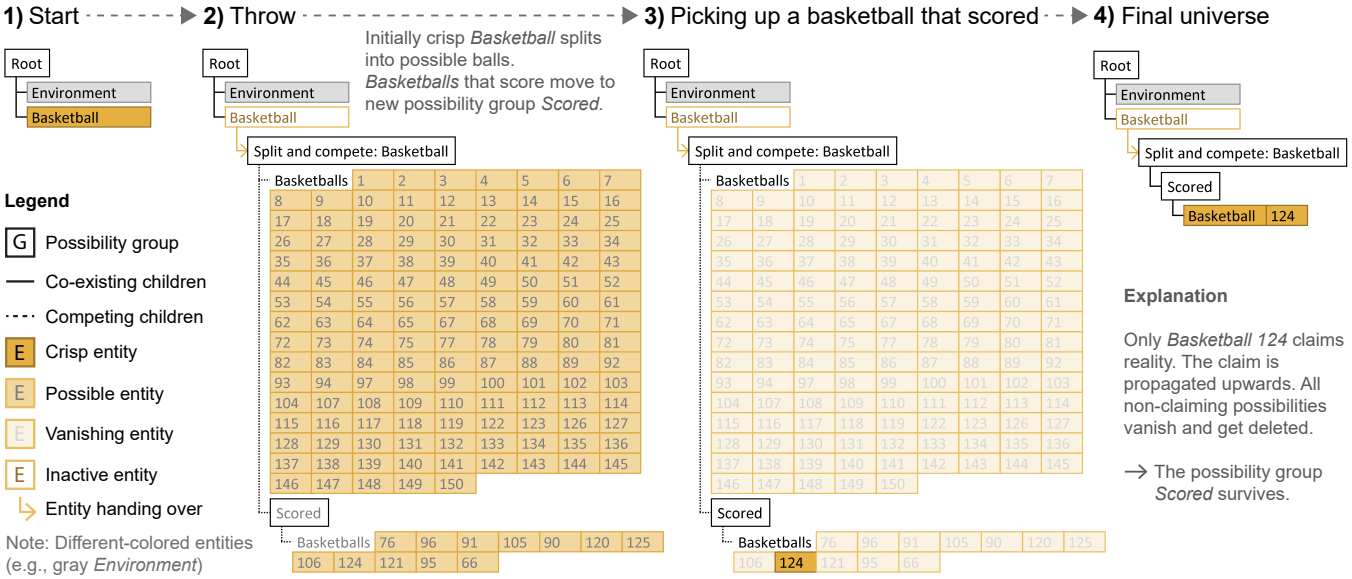


Figure 6: Sequence of possibility trees of the *Basketball Throw* example (scoring version, also see sequence of actions in Figure 2). 1: The basketball is initially crisp. 2: Throwing the basketball creates a CompetingPossibilities group with many PossibleEntity instances representing different trajectories. Basketballs that pass through the basket move to the 'Scored' group. 3: When picking up a ball, the possibilities of all other balls vanish. 4: The picked-up basketball (number 124) becomes crisp and with it the 'Scored' possibility, as the basketball is a child of that group. Note that the visualizations are generated by our system for inspection and illustration purposes, but they are not meant to be visible to end users.

4.3 Example Tree: Basketball Throw

Figure 6 shows the sequence of possibility trees of the *Basketball Throw* example. Initially, the basketball is crisp (Figure 6.1). After throwing the ball, it turns inactive and creates a group of competing possibilities containing replicas of the basketball (Figure 6.2). This group contains basketballs that only slightly differ in their trajectories. Further, the group contains an initially empty group of competing possibilities called 'Scored'. Each basketball that passes through the basket becomes a child of the 'Scored' group. Since all basketballs are competing, they cannot collide with each other. When picking up a basketball (Figure 6.3), it becomes crisp and all other basketballs vanish. If the ball is a child of the 'Scored' group, then this group becomes crisp as well (Figure 6.4).

4.4 Mightiness

Instead of using probabilities, we define *mightiness*, which denotes the partial or full membership of a possibility in reality (conceptually closer to fuzzy set theory than to probability). A mightiness going towards 0 means that the possibility is *vanishing*, i.e., it becomes invisible, but is not deleted yet (it might even re-materialize). A mightiness of 1 means that the possibility is crisp. This range is typically not exceeded. As opposed to probabilities, the mightiness values of all possibilities do not need to add up to 1.

The code in the application layer generally does not control mightiness directly. Instead, the application code instantiates possibilities and defines, how they should be grouped, while the possibilities are *claiming reality* as part of their logic. A possibility can claim reality whenever it 'senses' from the user input that the user is actively engaging with it. The inner logic of a possibility does not require multiverse knowledge for claiming reality—it simply suggests to the Hyperverse that the user's interactions indicate that the possibility is real. The mightiness values are then dictated by the tree hierarchy and reality claims with the following rules.

(1) The tree's root possibility has a mightiness of 1.
(2) If multiple possibilities are in a CompetingPossibilities group, each has a mightiness of half the group's mightiness.
(3) A single Possibility (regardless of whether or not it claims reality) in a CompetingPossibilities group has the mightiness of the group. This implies that if the group is crisp and has one child, then the single possibility in that group is also crisp.
(4) A Possibility in a CompetingPossibilities group is vanishing (visibly fading over time and eventually deleted), if it is not claiming reality, while one or more other possibilities in the same group are claiming reality.
(5) Reality claims are propagated upwards, i.e., the parent of a reality-claiming possibility, its parent's parent, and so on, implicitly claim reality as well. Propagation stops at already-crisp parents (where claiming reality does nothing).

Crisp possibilities with a mightiness of 1 cannot become non-crisp again, because crisp possibilities strictly exist or have existed in the past (i.e., even if they are not present anymore as an interactive object in the application). However, they can split and hand over to competing possibilities (e.g., Figure 6 transition from 1 to 2).

A step-by-step explanation of an example for applying the mightiness rules above can be found in the **supplemental materials** (file: *explanation_of_sorcery_possibility_tree_sequence.pdf*).

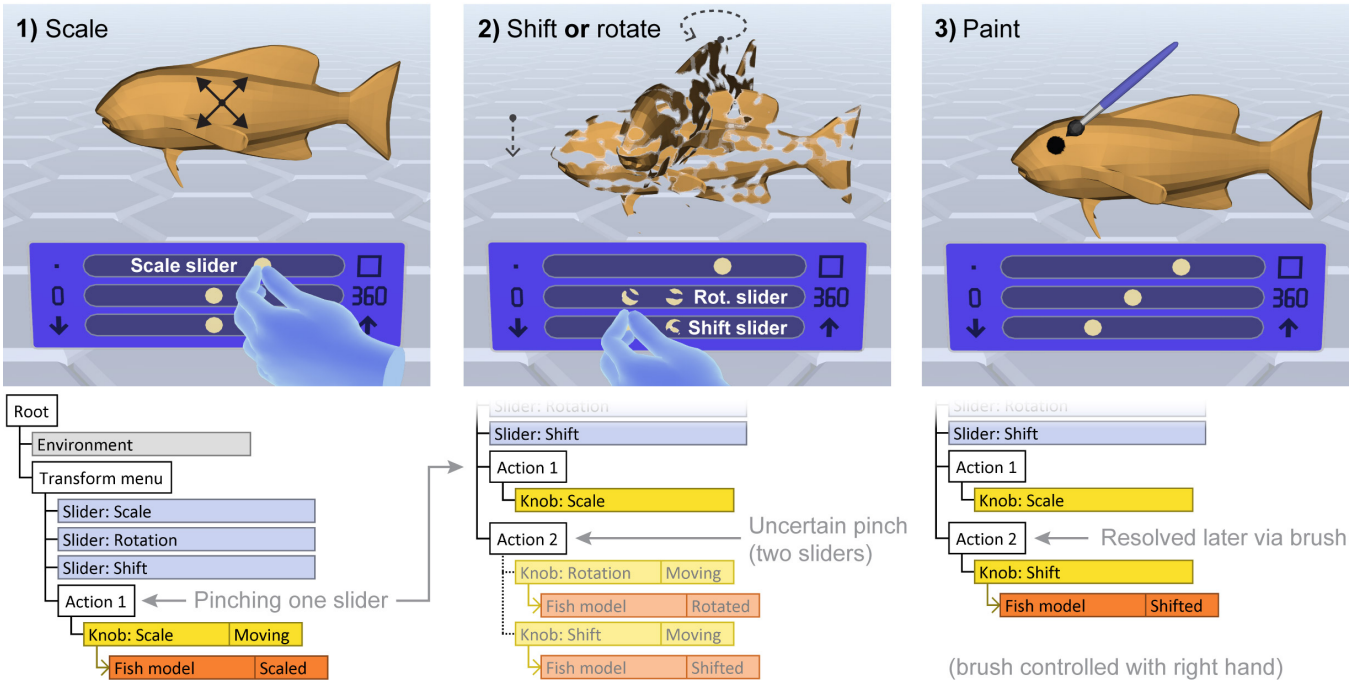


Figure 7: Combination of possibilities across contexts. 1: The user clearly uses the slider for scaling the object and the action is added to the menu accordingly. 2: It is unclear, whether the rotation slider or the shift slider is pinched. Hence, the two slider movements and their effect on the object's transformation are possible. 3: After releasing the sliders and painting on the desired object, the earlier slider movement is resolved (here, shifting was intended).

4.5 Menus and Cross-Context Disambiguation

As mentioned before, our architecture allows for disambiguation across application contexts. Figure 7 shows an example sequence of menu interaction followed by interaction outside the menu context. We will utilize this figure in the following to explain the tree structure of menus as well as cross-context disambiguation.

Menus use the same underlying tree elements as the other examples, but they follow a specific pattern. A possible menu is a group of co-existing possibilities. It contains an arbitrary number of co-existing possible menu elements, such as buttons and sliders (e.g., the sliders in Figure 7). In addition, for each action, an instance of a CompetingPossibilities group is created, which in itself is co-existing with the menu elements. For instance, in Figure 7.1 the user moves a slider that scales a virtual fish. Accordingly, 'Action 1' is added to the 'Transform menu' group, the moved slider becomes its only (and hence crisp) child, and the scaled fish becomes the child of that slider (as a *handover* Possibility). In contrast, in Figure 7.2, it is not clear, which slider is being moved[3], so both slider movements are possible and they are hence competing inside 'Action 2'. Both moved sliders independently request to become the new parent of the fish. In general, if multiple possibilities request re-parenting a possible entity to them at the same time, the Hyperverse automatically clones the possible entity accordingly without affecting the logic of the possibilities. Hence, each slider individually manipulates their possible version of the fish, and the sliders unknowingly compete, while they are both constantly claiming reality as long as the pinch is inside their bounding box. If the user already notices during slider manipulation that the outcome is uncertain, pivoting the pinching fingers in one direction makes the respective widget crisp (because the other one stops claiming reality), without requiring a post-hoc action. This capability—which is reminiscent of the widget behavior in the framework of Schwarz et al. [40, pg. 8]—emerges from our possibility tree structure.

Alternatively, the same ambiguity can be resolved post-hoc outside the menu context without requiring application code adjustments. This is possible because all aspects of the application are implemented as possibilities. In this case, if the sliders are still uncertain after releasing them (they both stop claiming reality at the same time), then both possible instances of the fish model persist in the virtual world (one rotated, one shifted). If the user now simply interacts with the intended object, e.g., by painting on the object (Figure 7.3), then that object is claiming reality, which in turn makes its parent slider claim reality as well, so they become crisp. Consequently, depending on the situation, it might not be necessary to manually disambiguate the slider movement, if the user sees the opportunity to simply interact with the desired instance of the 3D model. This type of cross-context disambiguation does not require specific implementation efforts at the application layer, preserving encapsulation of different parts of the application code. For instance, sliders and object instances do not require multiverse knowledge and they do not need knowledge about the context in which the uncertainty originated.

[3]Note that the bounding boxes of the sliders for pinching extend beyond their visual boundaries and hence can overlap (the same is true for the touch boundaries of buttons).

The video file *menu_painting_combination* in the **supplemental materials** shows both versions of this example (pivoting towards one slider versus resolving uncertain slider input post-hoc through interaction with a virtual object).

4.6 Possibilities versus Universes

The tree structure with competing and co-existing possibilities inherently allows to share resources between universes, compared to storing all possible universes directly. For instance, competing possibilities still share the data and states of co-existing possibilities in parent and sibling nodes of their CompetingPossibilities group, similar to a scene graph structure. A universe[4] can be constructed by traversing the tree as follows (where Current is a Possibility initially set to the root).

(1) Add Current to the list.
(2) If Current is a CoExistingPossibilities group or a PossibleEntity, then iterate over all of its children. For each child: set as Current and go to step 1.
(3) If Current is a CompetingPossibilities group, choose one child, set it as Current, and go to step 1.

The multiverse is the set of all universes resulting from choosing different combinations in step 3. Within the scope of possibility trees, 'reality' is a fully crisp universe.

5 Universe Playback and Streaming

There is always a universe path to the desired state, meaning that a single crisp universe can be played back afterwards or can even be live-streamed when introducing a fixed delay. We describe those functionalities in the following.

Universe Playback. By recording the multiverse user input (head and hand motions) and storing the final crisp tree (with only the possibilities that survived at the end), we can easily play back a sequence of crisp possibilities in a single universe. Specifically, we can play back the user input from the beginning and run the unaltered application logic, while the Hyperverse only allows those possibilities to spawn for which it is known from the tree that they will survive at the end. These possibilities are crisp at all times.

Universe Live Streaming. Our architecture also supports *delayed universe live streaming* (Figure 8). Specifically, when a local user (the streamer) interacts in a multiverse, our system can send a reduced set of universes or even a single universe to remote viewers by deliberately adding latency (see d and skewed action arrows in Figure 8). This latency provides a time window for the streamer to implicitly interact in their desired outcome, while the streaming system automatically discards vanished possibilities in the delayed stream (see straight red arrows from vanishing timelines in Figure 8).

The *universe_live_stream* video in the **supplemental materials** shows a concrete example with a multiverse user juggling in VR, which we describe in the following. Whenever the multiverse user throws a skittle with one hand, it splits into multiple possible skittles. Within around 2 seconds, one of the possible skittles (with a suitable position and orientation at that point in time) is caught by the other hand. Shortly after, the delayed universe is at the point of

[4]Note that we define universes solely for the sake of completion. All calculations are done on possibilities, so it is never actually required to create explicit universes.

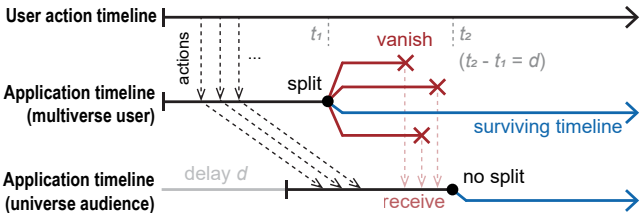


Figure 8: Universe live streaming. All *actions* of the local multiverse user are sent to the remote audience with a delay d. In this example, the timeline of the multiverse user splits at t_1, so the multiverse user experiences four timelines. Three of those vanish, and a single timeline survives. The remote universe does not need to split when its timeline arrives at t_2, because, within the delay, it received the identifiers of the timelines that will vanish.

throwing the skittle. Knowing which possible skittle will be caught based on the upcoming movement of the catching hand, there is no need to split into a multiverse. The skittle simply hands over to an immediately crisp skittle that has the linear and angular velocity of the multiverse's surviving skittle.

This concept allows streamers to implicitly choose their desired outcome for the audience without requiring to inform the system a priori about the streamer's intentions or requests of the remote audience. For instance, a live VR presenter could spawn objects, demonstrate concepts, and respond to audience chat messages without resorting to error recovery when using gestures for presentations or when conducting virtual physics experiments. This concept induces a trade-off: More delay means longer turnaround time for the chat, but a higher likelihood that a single universe is seen by the audience. Universe live streaming is inherently supported by our architecture without adjustments inside the application logic.

6 Implications and Future Work

By challenging a previously unquestioned law of any type of reality—namely, the effects of *observation* on reality—the main purpose of this technical-conceptual paper is not to provide final solutions but to open up new opportunities and challenges in research and practice within HCI, computer graphics, physics simulations, and games. We discuss some of those future opportunities in the following.

6.1 Use Cases of Desired Realities

Our multiverse framework enables new types of applications, for which best practices have yet to emerge. We support two types of use cases: (1) Extending existing interactions and applications with multiverse functionalities, and (2) enabling new use cases with multiverses as core part of the experience.

6.1.1 Extending Existing Interactions. While our examples were focusing on a multiverse state for illustrative purposes, we envision that applications can remain crisp most of the time, while still being able to seamlessly split into multiverses whenever needed and later collapse into a single universe again. Hence, developers can choose to use multiverses for specific aspects of their otherwise mostly universe-based applications, meaning that the main use

cases do not fundamentally differ from conventional virtual reality use cases (immersive modeling, training, etc.). Desired Realities can also be useful when physical movement is limited, e.g., due to confined space [21], where the lack of space can be made up for via continuously disambiguating follow-up actions.

Relatedly, we implemented our concept in virtual reality, because VR input and output fit particularly well to our examples. However, our concept is not limited to VR. Our architecture can also be compatible with conventional applications when replacing the UserInput with, e.g., ordinary gamepad or mouse input.

6.1.2 New Use Cases. Our concept and architecture enables a new set of use cases that make the multiverse an essential part of the experience. Examples include multiverse storytelling or puzzle games that make players think about how their actions affect previous outcomes, or shooters where players implicitly choose whether they hit the enemy or were hit depending on how they move post-hoc, as a form of player-driven narrative. Self-contained future works that build upon our conceptual and technical foundation will be able to explore the potential of this direction with the involvement of digital artists and (game) designers.

6.2 Multiverse Engine

In our current prototype implementation, competing possibilities are processed and rendered individually. To better scale multiverses and retain realtime performance, better handling of potentially many possibilities is required. Therefore—beyond our core architecture—a dedicated *multiverse engine* will be needed to efficiently handle processing and rendering across possibilities. For instance, similar possibilities could automatically be merged into a single possibility, whenever computational limits are reached. Relatedly, all possibilities are currently rendered with the same noise-based shader. Future multiverse rendering approaches can use different alternatives that could not only be more efficient but also visually more clear. This includes rendering similar possibilities as convex hulls or utilizing stochastic models for rendering semi-transparent possibilities [10]. Furthermore, besides overall clarity, different rendering approaches likely work best for different types of virtual entities in terms of perception. While our noise-based shader works well for 3D objects, superimposing 2D menus (containing icons and text) would likely work better with other approaches. Specifically, alternative approaches such as alpha blending, view-dependent visibility, or rendering one menu per eye (also see Figure 9) could be supported options in a future multiverse engine—also to investigate the perceptual implications.

Another important aspect that a multiverse engine should support is dealing with potential fail cases, e.g., avoiding or recovering from situations in which competing possibilities become too chaotic. Merging similar possibilities as described above will likely not only be beneficial for performance but will also help keep competing possibilities manageable for users. Furthermore, since a possibility tree also stores previously existing possibilities (see remarks about inactive possibilities described in subsection 4.2), it could be extended to enable going back in time as a last resort.

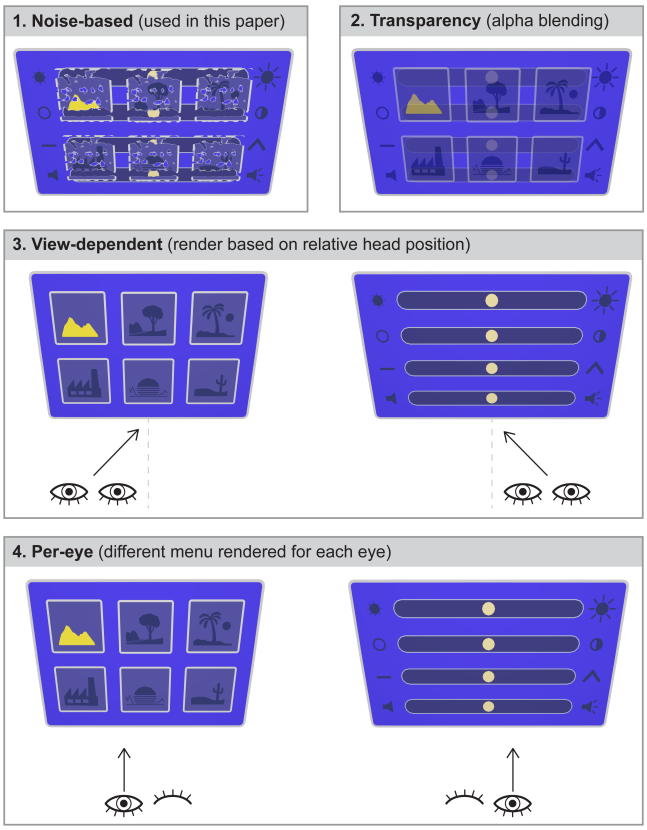


Figure 9: Potential alternative superimposition approaches for two competing menus. Instead of our noise-based rendering (1), alternative menu superimposition approaches could use alpha blending (2), decide which menu to render based on the head position (3), or render one menu per eye (4).

6.3 Human Factors

Our architecture is a basis for exploring the implications of Desired Realities. From a human factors perspective, we anticipate that parameters such as the number of feasible competing possibilities vary significantly among use cases. While many possibilities can be spawned for similar universes (e.g., basketballs), the manageable number of competing possibilities is likely much lower when they are semantically different (for instance, spawning and overlaying different menus). This also points to many questions with regards to cognitive load. At this point, we do not know in which situations the cognitive demand of multiverse interaction is higher than conventional error recovery and by how much, or when it becomes overwhelming. Hence, future studies could compare multiverse menu interaction (including uncertain button presses) with crisp menu navigation (navigating back to recover from errors).

Furthermore, we anticipate that there will be different preferences when comparing novice versus expert usage. For instance, a novice using an application for the first time may prefer seeing functionalities in a crisp way (even at the cost of error recovery) before they learn to navigate through the application using only visual hints that represent possibilities. This would allow expert

users to navigate applications faster, where a loss of accuracy due to faster movements is then compensated for with lenience for false positives through post-hoc disambiguation.

Furthermore, retaining user agency (without resorting to explicit selection) is one of the goals of our concept and architecture, but it is currently a design hypothesis, i.e., it remains to be tested how users perceive their agency depending on the use case or situation. Similarly, not only pragmatic metrics but also hedonic metrics (e.g., how pleasant and intriguing it is to inhabit a multiverse) will require dedicated testing. Hence, future studies could not only measure performance but also subjective aspects surrounding agency and hedonic qualities.

6.4 Implications for Menu Design

When designing multiverse menus, not only the menu layout but also the overlap of menu elements must be taken into account. Resolving button presses post-hoc may enable designers to use smaller buttons, but then closely spaced buttons should spawn menu elements suited for superposition (as in our *Settings Menu* example with buttons and sliders). This requirement may be addressed by adding multiverse-related objective function terms to automatic UI optimization methods [2, 31]. Primarily, objective functions for multiverse menu optimization will need to take the 'overlay compatibility' of sub menus into account.

Such multiverse UI considerations can also be combined with the interactions in the virtual environment (outside of the UI context). For instance, a multiverse painting application could have a toolbar in which closely spaced tools are arranged by the dissimilarity of follow-up interactions, so that uncertain menu selection can be resolved easily post-hoc. Hence, a *brush* tool button would be positioned next to a *flood fill* tool button, as their follow-up interaction would be easy to distinguish in case the selection is uncertain.

6.5 Multi-User Support

A straightforward multi-user extension would be to share the same multiverse (with the same set of possibilities) across all users, while merging the users' VR input into a single indicator for a shared desire. However, interesting constraints emerge when considering that users can have different desires. For instance, the *Crossroad* example could evolve into a multi-user traffic simulation and optimization system. Users would navigate the virtual worlds to get from A to B, while experiencing how the traffic adjusts to meet their desire. However, constraints imposed by different users (be it pedestrians or drivers of virtual cars) mean that some streets cannot be crossed instantly, as there would be contradicting desires (similar to related challenges in concurrent editing [52]). Alternatively, instead of imposing constraints from other users, large shared multiverses could continuously split into sub-multiverses in a way that users' desires within sub-multiverses are not contradicting each other. For instance, all users crossing the same street as oneself would share the same sub-multiverse, while all others vanish into a different online session. A related use case would be a situation in a frantic online battle royale game in which two players seemingly survive when shooting each other and the whole game session splits in two, with new players (who waited in a lobby until then) instantly taking over the other characters in the copied universe.

6.6 Interplay of Causality and Retrocausality

Future work could investigate the juxtaposition of causality and retrocausality further. For instance, in physical reality as well as in artificial realities, objects generally signify their affordances through visible features or shapes [29], such as the handle of a cup to drink from. Similar to defining physical properties of virtual objects by acting out their desired physical behavior [19], a Desired Reality could invert the concept of affordance. For example, given a set of 'possible objects to drink from', users could signal the desired affordance (e.g., grabbing it at a handle like a cup, at a stem like a wine glass, or with both hands like a bowl). Based on the expressed affordance, the object's shape including its signifiers would then become crisp.

7 Conclusion

We presented Desired Realities, a multiverse concept and architecture in which users implicitly choose their experienced reality through their actions. When presented with multiple possibilities, users simply act as if only their Desired Reality is real, while ignoring undesired possibilities. Our multiverse-based architecture enables the implementation of Desired Reality applications, featuring multiverse event dispatching, encapsulation of self-contained behaviors within possibilities, streamlined resource sharing in the multiverse, universe streaming, and more. To demonstrate different parts of our concept and architecture, we implemented several examples that span across various application aspects, namely, application logic, physics, menus, and gestures. There are many potential future research opportunities to explore different aspects or extensions of Desired Realities, such as multiverse menu optimization, multi-user support, and affordance inversion.

Acknowledgments

This work was supported by the *Alexander von Humboldt Foundation* funded by the *German Federal Ministry of Research, Technology and Space*, and the *German Research Foundation DFG* (grants 563933827, 495135767, and 390831618).

References

[1] Valentino Artizzu, Kris Luyten, Gustavo Rovelo Ruiz, and Lucio Davide Spano. 2024. ViRgilites: Multilevel Feedforward for Multimodal Interaction in VR. *Proc. ACM Hum.-Comput. Interact.* 8, EICS, Article 247 (jun 2024), 24 pages. doi:10.1145/3658645

[2] Gilles Bailly, Antti Oulasvirta, Timo Kötzing, and Sabrina Hoppe. 2013. MenuOptimizer: interactive optimization of menu systems. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology* (St. Andrews, Scotland, United Kingdom) *(UIST '13)*. Association for Computing Machinery, New York, NY, USA, 331–342. doi:10.1145/2501988.2502024

[3] Jeffrey A. Barrett. 2001. *The Quantum Mechanics of Minds and Worlds*. Oxford University Press. doi:10.1093/acprof:oso/9780199247431.001.0001

[4] Richard A. Bolt. 1980. "Put-that-there": Voice and gesture at the graphics interface. *SIGGRAPH Comput. Graph.* 14, 3 (jul 1980), 262–270. doi:10.1145/965105.807503

[5] Daniel Buschek and Florian Alt. 2017. ProbUI: Generalising Touch Target Representations to Enable Declarative Gesture Definition for Probabilistic GUIs. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 4640–4653. doi:10.1145/3025453.3025502

[6] Kevin Chow, Caitlin Coyiuto, Cuong Nguyen, and Dongwook Yoon. 2019. Challenges and Design Considerations for Multimodal Asynchronous Collaboration in VR. *Proc. ACM Hum.-Comput. Interact.* 3, CSCW, Article 40 (nov 2019), 24 pages. doi:10.1145/3359142

[7] Chiara Cimino, Gianni Ferretti, and Alberto Leva. 2021. Harmonising and integrating the Digital Twins multiverse: A paradigm and a toolset proposal. *Computers in Industry* 132 (2021), 103501. doi:10.1016/j.compind.2021.103501

[8] Thomas J. Clarke, Ian Gwilt, Joanne Zucco, Wolfgang Mayer, and Ross T. Smith. 2024. *Superpowers in the Metaverse: Augmented Reality Enabled X-Ray Vision in Immersive Environments.* Springer Nature Switzerland, Cham, 283–309. doi:10.1007/978-3-031-57746-8_15

[9] Pierre Dragicevic, Yvonne Jansen, Abhraneel Sarma, Matthew Kay, and Fanny Chevalier. 2019. Increasing the Transparency of Research Papers with Explorable Multiverse Analyses. 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, 1–15. doi:10.1145/3290605.3300295

[10] Eric Enderton, Erik Sintorn, Peter Shirley, and David Luebke. 2010. Stochastic transparency. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Washington, D.C.) *(I3D '10)*. Association for Computing Machinery, New York, NY, USA, 157–164. doi:10.1145/1730804.1730830

[11] Andreas Rene Fender and Christian Holz. 2022. Causality-preserving Asynchronous 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 634, 15 pages. doi:10.1145/3491102.3501836

[12] Andreas Rene Fender and Dieter Schmalstieg. 2025. Desired Realities: Concept and Examples. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA '25)*. Association for Computing Machinery, New York, NY, USA, Article 188, 8 pages. doi:10.1145/3706599.3719838

[13] Tiare Feuchtner and Jörg Müller. 2017. Extending the Body for Interaction with Reality. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 5145–5157. doi:10.1145/3025453.3025689

[14] Amirpouya Ghasemaghaei, Mykola Maslych, Yahya Hmaiti, Esteban Segarra Martinez, and Joseph J. LaViola. 2024. Towards Better Throwing: A Comparison of Performance and Preferences Across Point of Release Mechanics in Virtual Reality. In *Proceedings of the 50th Graphics Interface Conference* (Halifax, NS, Canada) *(GI '24)*. Association for Computing Machinery, New York, NY, USA, Article 26, 11 pages. doi:10.1145/3670947.3670963

[15] Purvi Goel and Doug L. James. 2022. Unified many-worlds browsing of arbitrary physics-based animations. *ACM Trans. Graph.* 41, 4, Article 156 (July 2022), 15 pages. doi:10.1145/3528223.3530082

[16] Miriam Greis, Hendrik Schuff, Marius Kleiner, Niels Henze, and Albrecht Schmidt. 2017. Input Controls for Entering Uncertain Data: Probability Distribution Sliders. *Proc. ACM Hum.-Comput. Interact.* 1, EICS, Article 3 (jun 2017), 17 pages. doi:10.1145/3095805

[17] Ken Gu, Eunice Jun, and Tim Althoff. 2023. Understanding and Supporting Debugging Workflows in Multiverse Analysis. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 149, 19 pages. doi:10.1145/3544548.3581099

[18] Scott E. Hudson and Gary L. Newell. 1992. Probabilistic state machines: dialog management for inputs with uncertainty. In *Proceedings of the 5th Annual ACM Symposium on User Interface Software and Technology* (Monteray, California, USA) *(UIST '92)*. Association for Computing Machinery, New York, NY, USA, 199–208. doi:10.1145/142621.142650

[19] Søren Qvist Jensen, Andreas Fender, and Jörg Müller. 2018. Inpher: Inferring Physical Properties of Virtual Objects from Mid-Air Interaction. 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, 1–5. doi:10.1145/3173574.3174104

[20] Ed Kaiser, Alex Olwal, David McGee, Hrvoje Benko, Andrea Corradini, Xiaoguang Li, Phil Cohen, and Steven Feiner. 2003. Mutual disambiguation of 3D multimodal interaction in augmented and virtual reality. In *Proceedings of the 5th International Conference on Multimodal Interfaces* (Vancouver, British Columbia, Canada) *(ICMI '03)*. Association for Computing Machinery, New York, NY, USA, 12–19. doi:10.1145/958432.958438

[21] Mohamed Kari and Christian Holz. 2023. HandyCast: Phone-based Bimanual Input for Virtual Reality in Mobile and Space-Constrained Settings via Pose-and-Touch Transfer. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 528, 15 pages. doi:10.1145/3544548.3580677

[22] 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. doi:10.1145/3173574.3173610

[23] Yang Liu, Alex Kale, Tim Althoff, and Jeffrey Heer. 2021. Boba: Authoring and Visualizing Multiverse Analyses. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (Feb 2021), 1753–1763. doi:10.1109/TVCG.2020.3028985

[24] Jennifer Mankoff, Gregory D. Abowd, and Scott E. Hudson. 2000. OOPS: a toolkit supporting mediation techniques for resolving ambiguity in recognition-based interfaces. *Computers & Graphics* 24, 6 (2000), 819–834. Calligraphic Interfaces: towards a new generation of interactive systems. doi:10.1016/S0097-8493(00)00085-6

[25] Jennifer Mankoff, Scott E. Hudson, and Gregory D. Abowd. 2000. Providing integrated toolkit-level support for ambiguity in recognition-based interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (The Hague, The Netherlands) *(CHI '00)*. Association for Computing Machinery, New York, NY, USA, 368–375. doi:10.1145/332040.332459

[26] Jennifer C. Mankoff and Gregory D. Abowd. 1999. Error Correction Techniques for Handwriting, Speech, and Other Ambiguous or Error Prone Systems. (1999). http://hdl.handle.net/1853/3385

[27] Jennifer C. Mankoff, Gregory D. Abowd, and Scott E. Hudson. 1999. Interacting with Multiple Alternatives Generated by Recognition Technologies. (1999). http://hdl.handle.net/1853/3393

[28] Christopher Menzel. 2013. Possible Worlds. Stanford Encyclopedia of Philosophy. https://plato.stanford.edu/entries/possible-worlds/. (Last access: July 28, 2026).

[29] Donald A. Norman. 1999. Affordance, conventions, and design. *Interactions* 6, 3 (may 1999), 38–43. doi:10.1145/301153.301168

[30] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L. Davidson, Sameh Khamis, Mingsong Dou, Vladimir Tankovich, Charles Loop, Qin Cai, Philip A. Chou, Sarah Mennicken, Julien Valentin, Vivek Pradeep, Shenlong Wang, Sing Bing Kang, Pushmeet Kohli, Yuliya Lutchyn, Cem Keskin, and Shahram Izadi. 2016. Holoportation: Virtual 3D Teleportation in Real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 741–754. doi:10.1145/2984511.2984517

[31] Antti Oulasvirta, Niraj Ramesh Dayama, Morteza Shiripour, Maximilian John, and Andreas Karrenbauer. 2020. Combinatorial Optimization of Graphical User Interface Designs. *Proc. IEEE* 108, 3 (2020), 434–464. doi:10.1109/JPROC.2020.2969687

[32] 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* (Bangkok, Thailand) *(SA '17)*. Association for Computing Machinery, New York, NY, USA, Article 3, 2 pages. doi:10.1145/3132818.3132822

[33] Robin S. Rosenberg, Shawnee L. Baughman, and Jeremy N. Bailenson. 2013. Virtual Superheroes: Using Superpowers in Virtual Reality to Encourage Prosocial Behavior. *PLOS ONE* 8, 1 (01 2013), 1–9. doi:10.1371/journal.pone.0055003

[34] Shadan Sadeghian and Marc Hassenzahl. 2021. From Limitations to "Superpowers": A Design Approach to Better Focus on the Possibilities of Virtual Reality to Augment Human Capabilities. In *Proceedings of the 2021 ACM Designing Interactive Systems Conference* (Virtual Event, USA) *(DIS '21)*. Association for Computing Machinery, New York, NY, USA, 180–189. doi:10.1145/3461778.3462111

[35] Batool Salehi, Utku Demir, Debashri Roy, Suyash Pradhan, Jennifer Dy, Stratis Ioannidis, and Kaushik Chowdhury. 2024. Multiverse at the Edge: Interacting Real World and Digital Twins for Wireless Beamforming. *IEEE/ACM Transactions on Networking* 32, 4 (2024), 3092–3110. doi:10.1109/TNET.2024.3377114

[36] Abhraneel Sarma, Alex Kale, Michael Jongho Moon, Nathan Taback, Fanny Chevalier, Jessica Hullman, and Matthew Kay. 2023. multiverse: Multiplexing Alternative Data Analyses in R Notebooks. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 148, 15 pages. doi:10.1145/3544548.3580726

[37] Abhraneel Sarma, Xiaoying Pu, Yuan Cui, Michael Correll, Eli T Brown, and Matthew Kay. 2024. Odds and Insights: Decision Quality in Exploratory Data Analysis Under Uncertainty. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 1034, 14 pages. doi:10.1145/3613904.3641995

[38] Simon Saunders, Jonathan Barrett, Adrian Kent, and David Wallace. 2010. *Many worlds?: Everett, quantum theory, & reality.* Oxford University Press.

[39] Benjamin Schindler, Jürgen Waser, Hrvoje Ribičić, Raphael Fuchs, and Ronald Peikert. 2013. Multiverse Data-Flow Control. *IEEE Transactions on Visualization and Computer Graphics* 19, 6 (June 2013), 1005–1019. doi:10.1109/TVCG.2012.296

[40] Julia Schwarz, Scott Hudson, Jennifer Mankoff, and Andrew D. Wilson. 2010. A framework for robust and flexible handling of inputs with uncertainty. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology* (New York, New York, USA) *(UIST '10)*. Association for Computing Machinery, New York, NY, USA, 47–56. doi:10.1145/1866029.1866039

[41] Julia Schwarz, Jennifer Mankoff, and Scott Hudson. 2011. Monte carlo methods for managing interactive state, action and feedback under uncertainty. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 235–244. doi:10.1145/2047196.2047227

[42] Julia Schwarz, Jennifer Mankoff, and Scott E. Hudson. 2015. An Architecture for Generating Interactive Feedback in Probabilistic User Interfaces. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 2545–2554. doi:10.1145/2702123.2702228

[43] Anthony Steed and Manuel Fradinho Oliveira. 2009. *Networked Graphics: Building Networked Games and Virtual Environments.* Elsevier.

[44] Paul Streli, Rayan Armani, Yi Fei Cheng, and Christian Holz. 2023. HOOV: Hand Out-Of-View Tracking for Proprioceptive Interaction using Inertial Sensing. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 310, 16 pages. doi:10.1145/3544548.3581468

[45] Paul Streli, Jiaxi Jiang, Andreas Rene Fender, Manuel Meier, Hugo Romat, and Christian Holz. 2022. TapType: Ten-finger text entry on everyday surfaces via Bayesian inference. 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 497, 16 pages. doi:10.1145/3491102.3501878

[46] Bernhard Suhm, Brad Myers, and Alex Waibel. 2001. Multimodal error correction for speech user interfaces. *ACM Trans. Comput.-Hum. Interact.* 8, 1 (2001), 60–98. doi:10.1145/371127.371166

[47] 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* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 161–174. doi:10.1145/3332165.3347872

[48] Chiu-Hsuan Wang, Chia-En Tsai, Seraphina Yong, and Liwei Chan. 2020. Slice of Light: Transparent and Integrative Transition Among Realities in a Multi-HMD-User Environment. 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, 805–817. doi:10.1145/3379337.3415868

[49] Jurgen Waser, Hrvoje Ribicic, Raphael Fuchs, Christian Hirsch, Benjamin Schindler, Gunther Bloschl, and Eduard Groller. 2011. Nodes on Ropes: A Comprehensive Data and Control Flow for Steering Ensemble Simulations. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 1872–1881. doi:10.1109/TVCG.2011.225

[50] 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* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 853–866. doi:10.1145/3242587.3242597

[51] Sicheng Yang, Yukai Huang, Weitong Cai, Shitong Sun, You He, Jiankang Deng, Hang Zhang, Jifei Song, and Zhensong Zhang. 2025. Plug-and-Play Clarifier: A Zero-Shot Multimodal Framework for Egocentric Intent Disambiguation. arXiv:2511.08971 [cs.HC] https://arxiv.org/abs/2511.08971

[52] 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* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 36, 14 pages. doi:10.1145/3544548.3581136

[53] Qiaohui Zhang, Atsumi Imamiya, Kentaro Go, and Xiaoyang Mao. 2004. Resolving ambiguities of a gaze and speech interface. In *Proceedings of the 2004 Symposium on Eye Tracking Research & Applications* (San Antonio, Texas) *(ETRA '04)*. Association for Computing Machinery, New York, NY, USA, 85–92. doi:10.1145/968363.968383

[54] Suwen Zhu, Yoonsang Kim, Jingjie Zheng, Jennifer Yi Luo, Ryan Qin, Liuping Wang, Xiangmin Fan, Feng Tian, and Xiaojun Bi. 2020. Using Bayes' Theorem for Command Input: Principle, Models, and Applications. 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, 1–15. doi:10.1145/3313831.3376771