

ADAPTEST: Supporting Designers in Authoring and Evaluating Environmentally Responsive User Interfaces for Extended Reality

JUNGWOO RHEE, KAIST, Republic of Korea

GUHN LEE, KAIST, Republic of Korea

YUMIN JUNG, KAIST, Republic of Korea

MUSTAFA DOGA DOGAN, Adobe Research, Switzerland

JUSTIN MOON[†], National University of Singapore, Singapore

ANDREA BIANCHI[†], KAIST, Republic of Korea

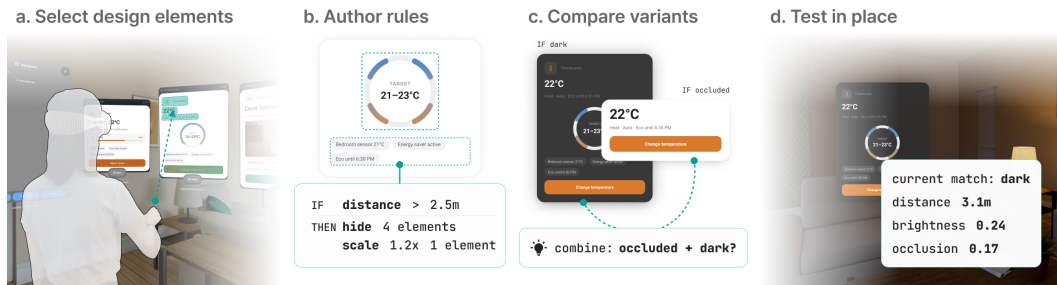


Fig. 1. ADAPTEST lets designers make 2D interfaces respond to the physical viewing context in an XR environment. A designer **(a)** selects an element layer from UI design, **(b)** authors an IF/THEN rule, **(c)** previews condition-specific variants attached around the default design, and **(d)** validates the matched variant in situ as the viewpoint changes.

Extended Reality (XR) headsets increasingly host familiar 2D user interfaces as windows floating in physical environments. However, their usability depends on environmental conditions that designers cannot fully control. Current design workflows often separate how a variant is designed from how it actually looks in the target environment. We present ADAPTEST, an in-situ authoring and testing tool that lets designers select interface elements, define when and how they change, compare the resulting variations, and test their activation without coding. Based on prior research and a formative study with 12 interface designers, the system targets three key environmental variables: distance, lighting, and occlusion. We evaluated ADAPTEST against a desktop design and in-headset preview workflow in a within-subjects study with 12 designers. Compared with the baseline, designers accounted for a wider range of environmental factors, addressed more overlapping

[†] Co-corresponding authors.

Authors' Contact Information: Jungwoo Rhee, KAIST, Department of Industrial Design, Daejeon, Republic of Korea, jwoorhee@kaist.ac.kr; Guhn Lee, KAIST, Department of Industrial Design, Daejeon, Republic of Korea, leeguhn@kaist.ac.kr; Yumin Jung, KAIST, School of Computing, Daejeon, Republic of Korea, jungym887@gmail.com; Mustafa Doga Dogan, Adobe Research, Basel, Switzerland, doga@adobe.com; Justin Moon, National University of Singapore, Department of Design, Singapore, Singapore, justin.moon@nus.edu.sg; Andrea Bianchi, KAIST, Department of Industrial Design, Daejeon, Republic of Korea, andrea@kaist.ac.kr.



This work is licensed under a Creative Commons Attribution 4.0 International License.

© 2026 Copyright held by the owner/author(s).

ACM 2573-0142/2026/11-ARTISS013

<https://doi.org/10.1145/3859325>

conditions, and repeatedly tested and revised their designs in place; there were no significant differences in workload or usability despite longer sessions. Our work contributes an integrated, designer-centered approach to adapting, comparing, and validating environmentally responsive XR user interfaces.

CCS Concepts: • **Human-centered computing** → **Human computer interaction (HCI)**; *Mixed / augmented reality*; *Graphical user interfaces*; User studies.

Additional Key Words and Phrases: XR, Responsive UI, Environmental Adaptation, Interface Design Tools

ACM Reference Format:

Jungwoo Rhee, Guhn Lee, Yumin Jung, Mustafa Doga Dogan, Justin Moon, and Andrea Bianchi. 2026. ADAPTEST: Supporting Designers in Authoring and Evaluating Environmentally Responsive User Interfaces for Extended Reality. *Proc. ACM Hum.-Comput. Interact.* 10, 8, Article ISS013 (November 2026), 29 pages. <https://doi.org/10.1145/3859325>

1 Introduction

Graphical user interfaces are moving off the screen and into physical space. Extended Reality (XR) headsets now render familiar 2D interfaces—dashboards, control panels, exhibit guides—as windows carried into immersive surroundings [14, 66], with deployments spanning museums, homes, and workplaces [30, 49]. Unlike a phone or desktop screen, such an interface is at the mercy of its environment: the viewer walks away and body text becomes illegible [22, 60]; the room dims and low-contrast styling disappears against passthrough [15, 31]; furniture or passersby occlude the panel [3, 40]; cluttered backdrops erode the readability of overlaid content [37, 54]. Most interface design is fixed at authoring time, so designers have to anticipate these conditions as separate cases.

Current tools give visual designers partial control over how their interfaces adapt to the XR environment and limited means to test that behavior in place. Automated systems decide the adaptation at runtime—repositioning or anchoring windows to spatial context [11, 16, 40], matching layouts to the surrounding scene [10], varying information density [42], projecting onto nearby surfaces [21, 24], switching input techniques [34], or recoloring panels against their background [53]. Because these decisions live in code and optimization objectives, designers cannot see or shape what the interface becomes under each condition, and often cannot picture it in advance [8, 33]. Design practice keeps that control in the designer’s hands but loses the tie to condition: designers hand-build each case as a duplicated static artboard (e.g., a dark-room, far-away, or occluded version [1, 32, 33]) and preview the copies in headset prototyping tools such as ShapesXR [57]. These copies let a designer inspect a mockup, yet they do not connect a condition to the change it should trigger, and confirming that one behaves correctly in place would mean coding, rebuilding, and redeploying to the headset—development work outside design practice [1, 48]. Designers are therefore left either to accept automatic adaptation or approximate it by hand, without a way to author their own condition-to-change mappings for the environmental conditions of a 2D interface in XR and test them in the room.

To address this gap, we present ADAPTEST, an authoring tool that lets designers adapt the GUIs they already create to the environments those interfaces will inhabit. After opening a design in the headset, a designer places its screens around the room, examines them where they will be used, and decides when the interface should change as spatial conditions shift. To author that change, the designer selects an element, assigns an alternative style such as larger type, higher contrast, or fewer details, and defines when that style should apply: when the viewer is far, the room is dark, or the panel is blocked. Each rule’s outcome appears in a preview beside the default design, so the designer compares how the interface should adapt across conditions. The designer then tests the rules against the room’s actual conditions, observes each transition as it fires, and refines the thresholds and styling accordingly. In a within-subjects study, twelve designers created

adaptive versions of multi-screen interfaces with ADAPTEST and with a current-practice baseline pairing a 2D design tool with in-headset preview. With ADAPTEST, all twelve participants authored explicit conditional rules, created design variants covering multiple environmental factors, and interleaved authoring with repeated Test Mode runs. Workload and usability remained comparable to the existing workflow. Participants noted that current tools represent adaptive behavior only as text beside the design, whereas ADAPTEST allowed them to author and experience that behavior as a working rule and reuse it across screens.

In summary, this paper contributes (1) a design characterization of environmental rule authoring for GUIs in XR—which elements designers target, which conditions trigger change, which style and layout outputs they reach for, and how far their scenarios cover under live spatial conditions; (2) ADAPTEST, an authoring tool that unifies element styling, environmental rules, condition previews, and situated testing in a single workflow, so that the 2D UI adaptation stays under designer control, indexed to sensed environment, and testable in place; and (3) a user study comparing designers' condition coverage, target selection, combination reasoning, and environmental debugging between ADAPTEST and a Figma-based current-practice workflow.

2 Related Works

ADAPTEST builds on three research areas: how adaptive behavior for XR interfaces is authored, how designers compare design variants, and how XR designs are tested in place.

2.1 Authoring Adaptive Behavior for XR Interfaces

Research on XR adaptivity has made strong progress on automated adaptation, where the system decides at runtime how the interface changes. AUIT [16] lets XR creators attach objectives, such as visibility and reachability, to UI elements in Unity, and a solver adjusts the positions, rotations, and scales at runtime. SemanticAdapt [10] retargets a world-anchored layout to a new room by matching virtual to semantically related physical objects, and SituationAdapt [40] re-replaces elements after a vision-language model rates how socially appropriate each surrounding area is. Further systems select placement strategies by context [11], compute layouts against the physical scene [19], project interfaces onto surrounding geometry [21, 24], and switch selection techniques as targets and posture change [34]. Automated adaptation has also reached visual properties: correcting color and contrast against the background [12, 45], repositioning and recoloring panels from sensed scene context [53], adjusting level of detail with the viewer's visual angle [62], and adapting rendering to the sensed environment in web-based AR [17]. Purpose-built interfaces further demonstrate the value of environmental conditions, such as a museum guide that swaps content based on visitor distance [9]. Even when content creators pre-design the variants (e.g., [42]), where applications carry designed level-of-detail versions and an optimizer selects among them by cognitive load, task, and environment, the runtime still owns the mapping from context to variant. Optimization itself has grown more open to users, who can choose among Pareto-optimal proposals [28] or steer the objectives by spoken preference [41]. Across these systems, the condition–appearance mapping lives in cost functions, objective priorities, and code, which makes it difficult for visual designers to inspect what will happen under a given condition or to adjust it without programming [8, 33].

Rule-based authoring, in contrast, puts the mapping in the author's hands, and XR tools have primarily applied it to interaction events. Trigger-action programming has long enabled non-programmers to state IF/THEN behavior [26]. In XR, XRSpotlight [18] presents the interactions of a Unity scene as natural-language rules that novice developers can inspect and copy across scenes; ProInterAR [65] lets creators stack blocks that react to grasps, gestures, and collisions between AR contents; ProGesAR [64] lets designers place proxemic and gestural triggers that fire when a person crosses a distance threshold or performs a gesture; and RealityCanvas [63] binds sketched

animations to tracked physical motion. FrameKit [61] brings authored adaptation to 2D interfaces by placing keyframes along parameters such as viewer distance, with in-between states interpolated and rendered from simulated sensor values. These works establish that authors can express rich behavior through rules, with triggers as discrete events (e.g., input actions, collisions, threshold crossings) and actions on scene objects or sketches. Rules that respond to sensed environmental conditions and attach to the GUI elements in XR remain an open opportunity.

ADAPTEST extends rule-based authoring by: designers select granular UI elements and state how those elements should change under continuous environmental conditions, keeping the condition-variant mapping explicit, deterministic, and editable without code.

2.2 Comparing Design Variants

Seeing design alternatives side by side is an established aid to design judgment. Creating and comparing prototypes in parallel yields better outcomes and more divergent exploration than refining one at a time [13], and tools have operationalized this insight: Juxtapose [23] runs alternative interface implementations side by side with runtime tuning, subjunctive interfaces let users set up and view multiple scenarios in parallel [43], Scout [56] generates layout variants from high-level constraints for rapid comparison, and DesignTrace [50] tracks alternatives as designers iterate with a generative model. These tools compare alternatives that the designer freely explores; the variants answer "which design do I prefer?" Adaptive XR interfaces pose a different comparison question—"does my design hold up under every condition it will meet?"—which current practice can answer only partially. Designers encode environmental cases as duplicated artboards or component variants [32, 33], but the mapping from condition to variant remains only in file names and memory. Responsive web design offers a designer-facing analogy: media queries bind layout variants to viewport conditions, and design tools preview breakpoints side by side. The same side-by-side preview for the environmental conditions of an XR space is the open question.

To answer this, ADAPTEST produces a condition-specific preview that wraps around the default design, so a designer can inspect coverage across dark-to-bright, far-to-near, occluded-to-visible, and combined conditions at a glance.

2.3 Testing XR Designs in Place

Validating an XR design still requires being in the deployment environment, which slows iteration. Studies of XR practitioners report that creators cannot judge legibility, scale, or lighting from the desktop [1, 33]. Site-specific work demands "frequent on-site visits and additional iterations" [48], developers spent over a third of their time testing in situ and identified lighting as one condition that desktop views cannot reproduce. In-situ authoring tools shorten this loop by moving creation into the target environment: DART [44] moved AR authoring into the designer's workflow, PintAR [20] sketches interfaces directly in AR, Rapido [39] turns video captures into interactive AR prototypes, ImagineAR [38] assembles outdoor scenes in place with AI assistance, and commercial tools such as ShapesXR [57] preview screen designs spatially in the headset; CAVE-AR [7] approaches the same challenge from the opposite direction, simulating and debugging AR experiences from within VR. Fast-Forward Reality [51] generates test cases for context-aware policies and plays as immersive visualizations to verify in XR before deployment. In these tools—as in the live try-out loops of the rule-based authoring tools above—testing is limited to the conditions the room offers at that moment; a darkened room or a twenty-meter viewing distance is difficult to produce on demand, and the tester has limited means to tell which authored state is currently active.

ADAPTEST's builds on these testing practices: the designer dials environmental conditions with manual controls or drives them from live headset sensing, and the system activates the currently matched rule—turning situated testing from re-enacting circumstances into querying the rule set.

3 Formative Study

We conducted a formative study to observe how screen-rooted designers adapt their interfaces when the designs enter a real space. Twelve designers prototyped interfaces for a multi-zone makerspace and reviewed them as virtual screens on Apple Vision Pro from three viewing distances, iterating in place. Their behavior yielded three findings—an element-level adaptation vocabulary, a struggle to cover compounding conditions, and a dependence on being in place to judge thresholds and transitions—from which we derive design goals DG1–DG3.

3.1 Method

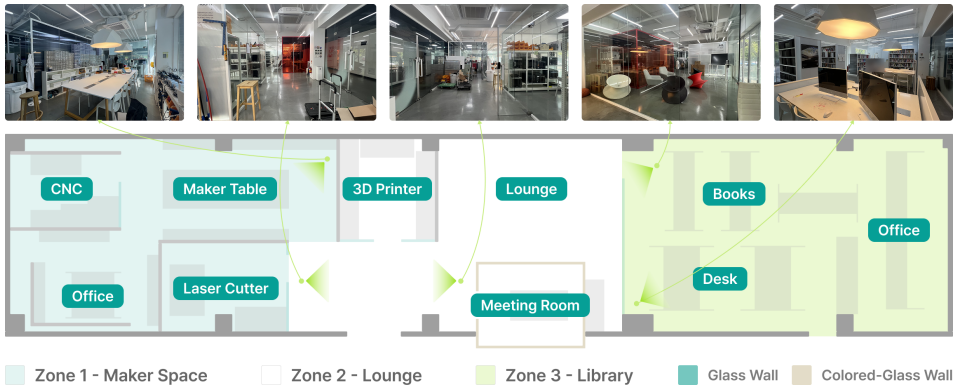


Fig. 2. Floor plan and zone divisions of the formative study environment.

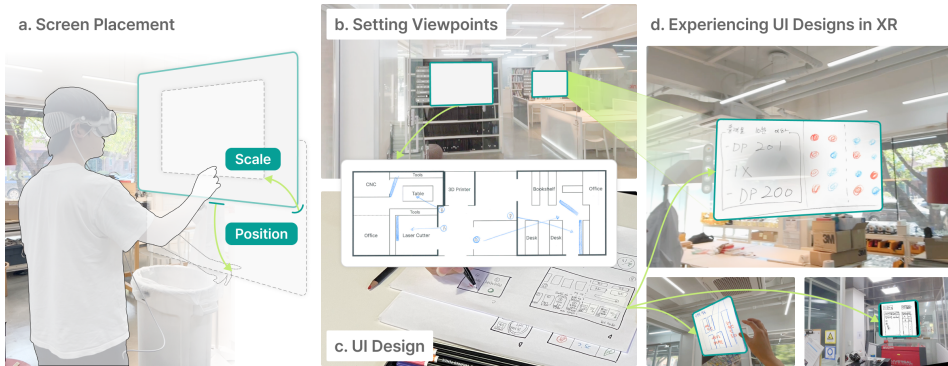


Fig. 3. Formative study task process: **a)** Participants use HMD and browser to choose the position, scale, and angles of the screens, **b)** Set the viewpoints for each screen and document them on the worksheet, **c)** Design the UI on paper using pens and pencils, **d)** Review the UI designs in the XR environment and iterate.

We ran the study in a maker's lounge measuring $33\text{m} \times 7.2\text{m}$ with three functionally distinct zones—fabrication equipment, a communal lounge, and a library area (Figure 2)—following established behavioral-mapping protocols for spatial design [55] informed by proxemic interaction research [2]. The zone separation set three viewing distances: within-zone (average 2.8m), adjacent-zone (approximately 11m), and non-adjacent-zone (approximately 22m). Each two-hour session opened with consent, a short demographic interview, and a five-minute briefing on the study aim,

followed by a headset tutorial on placing, scaling, and rotating a window. Participants then spent ten minutes placing at least two virtual screens in each of two zones, and every window had to remain visible from at least one point in each zone. Three design rounds of about twenty minutes followed, one from the central lounge and one from inside each of the two other zones, so every screen was designed from within its zone, from the adjacent zone, and from the non-adjacent zone. In each round, participants sketched interfaces on paper with real text content, we photographed and displayed the sketches as virtual screens on Vision Pro, and participants reviewed them in place, adjusted window size or angle where needed, recorded their viewpoint and screen positions on a floor-plan worksheet, and iterated before the next round (Figure 3). After the third round, participants placed their designs on the screens and stated the distances at which they wanted each design to switch.

We recruited twelve designers (7 female, 5 male; $M = 2.5$ years of UI design experience, $SD = 1.4$) with design-related academic backgrounds, HMD familiarity, and prior XR use (Appendix A). Only two had authored XR interfaces before, so the sample reflects the screen-interface practice ADAPTTEST targets. Each session closed with a 25-minute semi-structured interview on design choices, spatial influences, and adaptation challenges (Appendix B); participants received approximately 40 USD in local currency. We collected audio, interface designs, headset camera views, and floor-plan markings of viewpoints and screen positions, and we analyzed 148 interface designs across 48 screen placements together with the transcripts through the six-phase thematic analysis [5]. Three authors coded transcripts in English alongside the designs and floor plans, first in one joint pass and then in rotation, so that each author layered codes over every session; the three code sets were merged by discussion into the themes reported below. Each design was coded as one unit and tagged with the adaptation strategies visible in it, and these tags yield the proportions (Figure 5).

3.2 Finding 1: Adapting UIs on Element Level

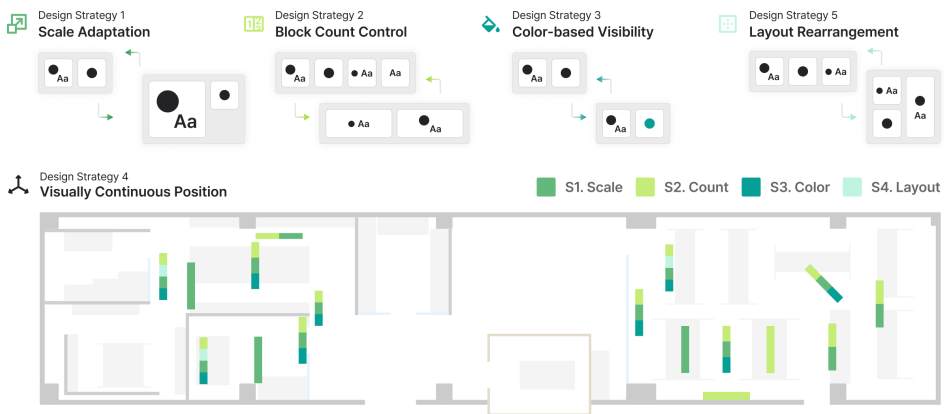


Fig. 4. Five key strategies for UI design, along with a floor plan indicating where each strategy is applied.

Adaptation happened at the level of individual elements, through five recurring strategies (Figure 4; distributions in Figure 5). **Scale adaptation** appeared on 56% of screens (83% of creators): participants enlarged text as distance grew, and 42% of creators swapped text for shapes and icons once even large text stopped reading—P12: *"Once you are far enough away that even large text does not convey meaning, it is better to use shapes to communicate the information."* **Block count control** appeared on 62% of screens (83% of creators): P6 reduced one design from four blocks at 2.5m to two

blocks at 14.7m, and P4 explained, "When far away, the maximum visible blocks should be limited to two or three. At a distance, I kept only one key component set centered." **Color-based visibility** served binary status at range (30% of screens, 50% of creators)—P3: "I intended green glowing to indicate presence, making it more noticeable." **Visually continuous positioning** governed placement: 75% of creators avoided occluding physical objects, and 79% of screens sat against walls or shelves for a flat backdrop. **Layout rearrangement** stayed rare (10% of screens), because participants feared it would read as a different screen—P8: "If component positions change dramatically or components appear and disappear, users perceive it as a completely different screen, requiring additional mental energy to relearn."

These strategies form a compact vocabulary of element-level outputs—resize, reduce, recolor, reposition, reorder—applied per condition, yet every adapted state existed only as another static copy whose triggering condition lived in the designer's memory, as in the duplicated artboards of current practice [32, 33]. This finding motivates **DG1: element-targeted rule authoring**—let designers select the concrete elements they already think in and state the condition–output mapping as an explicit, editable rule.

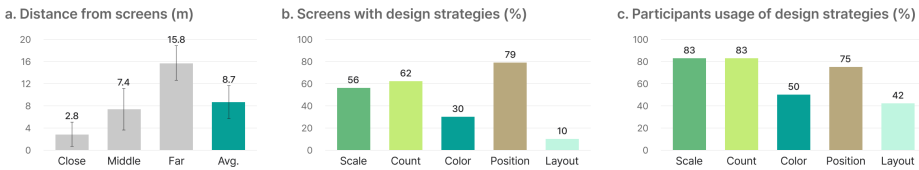


Fig. 5. UI design strategy statistics: **a)** Distribution of distance by zone division, **b)** Distribution of design strategies by screen, **c)** Distribution of design strategies by participant.

3.3 Finding 2: Deciding Which Versions to Focus

Designers found it hard to decide which conditions deserved a distinct version and to treat the same condition consistently across versions. Participants authored progressive series—P12 re-sized text at 15%, 26%, then 41% of the window across three distances—and prioritization strained as versions multiplied; P8: "It becomes difficult to decide which components to focus on first when there are too many UI windows. Users must judge what to focus on, which takes mental energy." The same condition also drew inconsistent treatments: P12 color-coded seat availability at 5.0m and 11.0m but dropped color at 15.8m, while P6 set the threshold elsewhere: "When I'm far away, I just want to know which ones are in use. I don't think it necessarily needs to be color-coded." Participants were further unsure which situations deserved a version at all; P10: "Should I know information about that space when I'm not entering it?"

The task fixed how many versions each screen received, but which conditions each version served, and whether sibling versions agreed, was the designers' own work, and nothing in the designers' materials showed which conditions were handled and which were missed. This finding motivates **DG2: multi-condition preview comparison**—render each rule's condition-specific state beside the default so coverage gaps and mismatched siblings become visible at a glance.

3.4 Finding 3: Judging Transitions in Place

The decisions participants found hardest—when an adaptation should fire and how it should change—were the ones they could not settle away from the space. Several preferred discrete, zone-based switching; P12: "From the user's perspective, thinking there's something physical there, and when they get closer it gets bigger—this feels more naturally acceptable." Others feared jarring changes near

boundaries—P7: “When I get close, the screen suddenly changes, that would be very confusing”—and P11 resisted continuous change: “If the size changes every time I move, it feels chaotic.” Occlusion problems, likewise, surfaced only through physically moving in the space; P11: “But when this wall blocks like this, this icon gets hidden, so I was wondering what to do about that,” and P8 was “forced to place it here” by a bookshelf that blocked or blurred the view.

Participants could author adapted states on paper, but validating a trigger point or a transition required standing where the interface would live. This finding motivates **DG3: situated testing of the active rule**—let designers dial or sense the environmental conditions in place and watch the one currently matched rule drive the design. Of ADAPTEST’s three sensed conditions, distance and occlusion emerged directly from this study, and lighting was added as a condition from prior findings on legibility in dim and bright environments [15, 31].

4 ADAPTEST

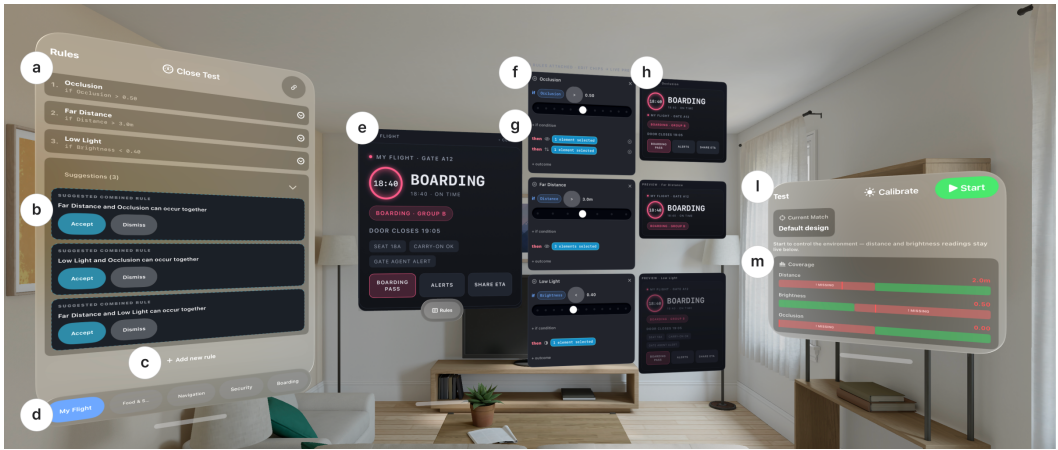


Fig. 6. ADAPTEST overview: Users can **a)** View all created rules and reorder to determine priority. **b)** Get suggestions for combining rules for different scenarios. **c)** Add new rules. **d)** Show or hide for each imported UI. **e)** Review the original UI and select elements for rule application. **f)** Set environmental triggers (IF statements) with specific values. **g)** Style output for the THEN actions. **h)** Preview output of created rules. **i)** Test to see how adaptive UIs respond to manual or **m)** Live sensor values like distance, brightness, and occlusion.

ADAPTEST is an authoring tool for designing adaptive interfaces in the environments where they will be deployed (Figure 6). It lets designers author adaptive behavior as rules over interface elements, compare the condition-specific versions side by side, and test which becomes active directly in the intended physical environment. The resulting design is a set of adaptive interface variants together with the rules that define when each adaptation should occur.

The tool was structured in response to the findings of the literature review and formative study, and designed with three goals: DG1) let designers author adaptation rules directly on the elements of their imported design, stating **what should change and under which environmental condition**; DG2) let designers **compare each condition-specific variant** beside the default design, so overlapping and uncovered conditions stay visible; and DG3) let designers **test which variant becomes active in the real environment**, under sensed or manually controlled values, and refine the rules in place. A typical session runs as follows. A designer imports a multi-screen design, places its screens around the room, and taps an element to make it a rule’s target (DG1). The designer sets

a condition, such as *distance > 3 m*, adds an outcome, such as a larger size, and watches the preview next to the rule update as they edit (DG1). Expanding the screen shows a preview of every rule next to the default; a suggestion card notes that two of their rules can fire together, and the coverage strip shows that brightness has no rule yet, so they add one (DG2). They then start Test Mode, walk across the room, and watch the far variant appear, drag the distance slider past the room's far wall to check a condition the space cannot reach, stop, nudge the threshold, and test again (DG3). Once the rules hold, they copy a rule onto the other screens and export the project for handoff.

4.1 DG1: Element-Targeted Rule Authoring

Designers author each adaptation directly on the rendered design. Importing an interface opens a spatial window per screen (Figure 6e). The designer places these windows around the room, scales one with a two-hand pinch, and toggles each open or closed from the workspace bar (Figure 6d); reopening a saved project restores its rules, variants, and test records. To set *what* changes, the designer taps an element (e.g., the BOARDING GROUP B chip). To set *when*, the designer picks a measure, a comparison, and drags a slider to a threshold such as *distance > 3 m* (Figure 7a). Pressing *+ outcome* then gives the selected elements a visual change: hiding them or bringing them to the front by selection, stepping size or opacity with a slider, or setting a text or background color with a picker (Figure 7b). Each edit plays back in the preview attached to the rule (Figure 6h).

Each rule is an IF-THEN pair over the chosen elements, and *+ if condition* adds conditions so one rule can describe a compound situation, such as a far viewer in a dim room or a 2 to 4 meter distance band. Distance and occlusion follow our formative study; brightness follows legibility findings on dark-mode interfaces in optical see-through head-mounted displays [15, 31]. A THEN output applies to the selected elements or to the whole screen, and one rule may carry several outputs, so different elements respond differently under the same condition, and the output palette (hide, bring-to-front, resize, opacity, text and background color) follows the adaptation strategies designers used in the formative study. When a rule activates, its elements ease into the new state over a short transition, so a hidden element fades and a resized one grows into place. The Rules panel lists the rule set for renaming, enabling, disabling, duplicating, and deleting (Figure 6a); when several rules match at once, the highest in the list takes effect, and designers reorder rules with the arrows on each card. On the web editor, designers can also drag preset condition cards onto a screen or element to seed a rule.

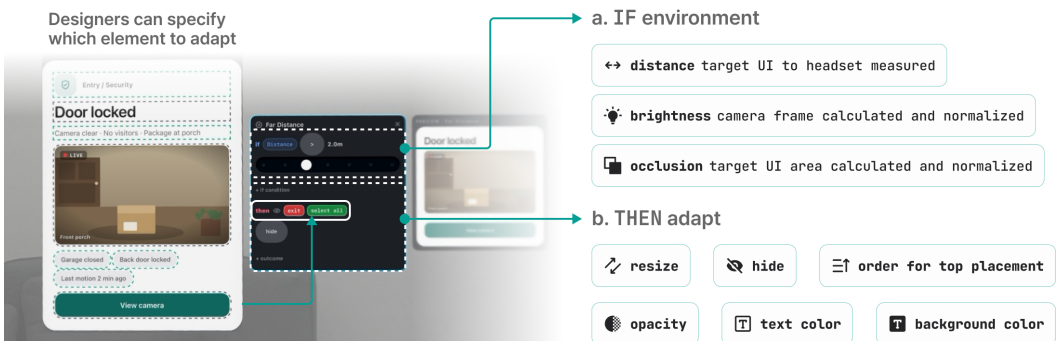


Fig. 7. ADAPTEST supports **a)** IF condition authoring with environmental measures, and **b)** THEN output authoring effective on the whole screen or selected element(s).

Designers can carry a finished rule onto another screen. Pressing *apply* to lists the project's other screens and copies the rule onto the chosen one (Figure 6a); where the original target is missing on that screen, the designer picks which elements the copy should affect.

4.2 DG2: Multi-Condition Preview Comparison

As rules accumulate, designers need to see how the variants relate: where conditions overlap, clash, or leave a situation unaddressed. Expanding a design window lays each rule's preview beside the unchanged default, so several variants read side by side (Figure 6h). When two rules could fire together, a suggestion card appears ("*Far Distance and Occlusion can occur together*"), and the designer accepts it to merge the rules or dismisses it (Figure 6b). When two rules set the same property to different values, a conflict card appears, and the designer combines the rules, chooses which wins, or dismisses it.

The coverage strips which ranges rules already cover and which remain open, with a marker at the live value (Figure 6l, m). Tapping an open range closes it, and the designer chooses whether the design eases gradually into the covered state or snaps to it; a measure with no rule at all prompts a suggestion to add one. In the web editor, a designer can also select two variants and view a per-layer list of all properties that differ between them. Every preview updates as its rules' conditions, targets, or outputs change, so overlaps, conflicts, and gaps stay visible on one authoring surface.

4.3 DG3: Situated Testing and Refinement

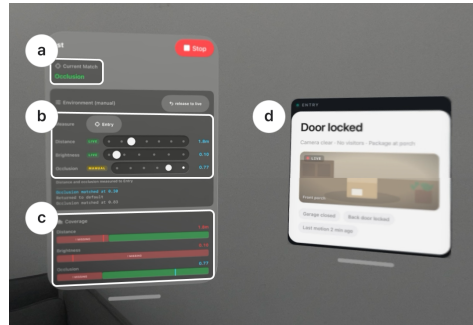


Fig. 8. ADAPT Test Mode: **a**) currently activated design version, **b**) current environmental values from live sensing or manual control, **c**) coverage strip summarizing addressed and under-addressed environmental factors, and **d**) interfaces responding to active sensor values.

Judging whether an adaptation is right, such as what counts as “far” or whether text stays legible in the available light, requires being in the space. Test Mode therefore evaluates authored rules in the physical setting where the interface will appear. Pressing *Start* on the Test panel folds the editing controls away, and the screens begin reacting to the room: stepping closer, moving into shadow, or blocking a screen with a hand switches it to the matched variant (Figure 8). The panel names the active rule, shows the live distance, brightness, and occlusion values, and keeps a trace of recent switches (Figure 8a, b). A *Calibrate* control re-baselines brightness to the current room, and a *Measure* menu sets which screen the readings target.

When the room cannot reach a value, such as a viewing distance longer than the space, the designer drags that measure's slider to set it by hand while the others keep reading from the room; releasing the slider returns it to live sensing (Figure 8b). Screens follow the highest-priority matching rule and return to the default when none matches, and a new match takes effect only

after it holds briefly, so a reading hovering near a threshold does not flicker. Leaving Test Mode returns the designer to authoring, where changing a threshold, target, output, or priority, and then pressing *Start* again, repeats the test.

Once satisfied, the designer exports the project for continued design work, developer handoff, or sharing. The export is HTML/CSS with the rule and style specifications, so a developer can implement the behavior after designers have iterated it in context, and the saved project keeps the default interface, variants, specifications, and test records.

4.4 System Implementation

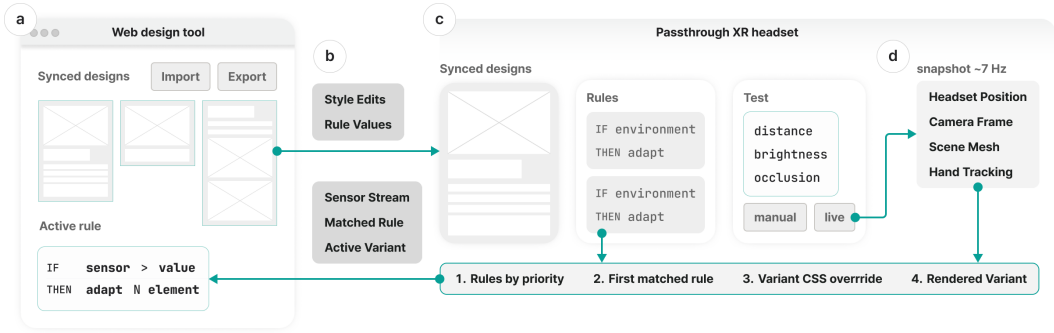


Fig. 9. ADAPTEST architecture: **a)** The React web editor **b)** exchanges a shared project through a typed message bridge with **c)** a SwiftUI visionOS application. Both surfaces render the same HTML/CSS source, while **d)** the active rule engine applies variant-specific CSS during testing.

ADAPTEST combines a React and TypeScript web client with a native SwiftUI application for visionOS (Figure 9). The web client handles import, project management, and detailed rule editing; the native application presents the rules panel, Test panel, and imported interfaces as spatial windows. Both share one project model of screens and layers, variants, rules, window placements, and test sessions.

On import, the web client extracts an interface's layers as CSS selector paths and creates a default variant per screen; each adaptive variant stores only the CSS properties that differ from the default, and its rule records conditions, target variant, enabled state, and priority. Both applications render the same HTML/CSS through WKWebView with the variant's CSS injected over it. A typed JavaScript-to-Swift bridge synchronizes project updates, rule edits, placements, sensor readings, and active-rule changes as JSON messages; updates carry a source and timestamp so the latest write wins, and edits made while disconnected replay on reconnection.

Rule evaluation runs in both TypeScript (authoring previews) and Swift (Test Mode) with the same priority behavior. On the headset, a sensor loop runs every 150 ms, evaluates each open screen's enabled rules against its current distance, brightness, and occlusion, and shows the variant of the lowest-numbered matching rule, restoring the default when none matches. A newly matched variant commits only after it stays the best match for 300 ms, and variant changes patch the variant's CSS into the live document with a 400 ms eased transition. Suggestions and coverage strips derive from the same rules: each rule's conditions reduce to a value range per measure; compatible overlaps become combined suggestions that merge both rules' CSS, conflicting properties are surfaced for the designer, and the strips mark the covered and open bands of each measure. Cross-screen reuse maps source elements by semantic role (primary content, secondary content, status, action button) before the designer confirms targets and a screen-specific copy is created.

Live values come from ARKit. Distance is the world-tracked distance from the headset to each window, with a geometry-based fallback when tracking drops. Brightness comes from environment light estimation, normalized between the lowest and highest values seen since launch or the last *Calibrate*; on devices with main-camera access, the camera frame's mean luminance is used directly. Occlusion samples a 3×3 grid on each window and scores each point by how close a tracked hand comes to the line from the headset to that point (fully occluded within 0.10 m, clear beyond 0.30 m), and ray tests against the classified scene mesh add furniture such as tables and seats as occluders. Manual overrides enter the same evaluator. Export writes each variant as static HTML with its CSS overrides, a JSON package of the project and test sessions, and rule and style specifications for handoff; variant HTML can return to Figma through an external HTML-to-design plugin.

Table 1. Observable end-to-end adaptation latency from video annotation. Each condition uses the first 100 valid environmental-change/UI-response pairs.

Condition	Median (s)	Mean (s)	IQR (s)	Range (s)
Distance	0.77	0.81	0.67–0.93	0.37–1.57
Brightness	1.70	1.81	1.07–2.43	0.23–4.50
Occlusion	0.37	0.46	0.30–0.57	0.10–0.97

We measured end-to-end adaptation latency for each factor from headset-view video. We repeatedly changed one condition, annotated the first 100 valid pairs of a visible environmental change and the first visible UI response, and used the time between them as latency. This covers sensing, rule evaluation, synchronization, and rendering as the designer experiences them (Table 1). Brightness was slower and more variable because room lighting changes more slowly and the light estimate settles gradually.

5 User Study

We ran a within-subjects study comparing how designers author adaptive XR interfaces with ADAPTTEST against their current 2D practice. ADAPTTEST offers three capabilities—element-targeted rules, rule-attached previews with combination suggestions, and in-context Test Mode—and we ask how designer behavior on three fronts differs between ADAPTTEST and a Figma-based current-practice baseline: **RQ1**. How explicitly do designers specify adaptive behavior—which element changes and under which condition—and where does that specification live, in a working rule or in text or speech? **RQ2**. How many environmental conditions, including compound ones, do designers cover? **RQ3**. How do designers test their adaptations in place, and how do tests feed back into thresholds and styling?

5.1 Participants

We recruited twelve designers (7 female, 5 male; ages 20–31, $M = 25.0$, $SD = 2.8$) via university community channels. Participants included UI/UX, industrial design, and HCI researchers and graduate students in related domains. They had an average of 3.2 years of design experience ($SD = 1.7$; range: less than one year to more than five years). Eligibility criteria included prior UI design practice, at least one experience with an HMD or AR/VR, normal or corrected-to-normal vision, and intact color vision. Seven participants also reported XR UI design experience through coursework or research projects (Table 2). All twelve used Figma as a primary design tool and reported high Figma fluency ($M = 6.0$ of 7, $SD = 1.0$); thus, the baseline condition used a tool with which they were already proficient. Participants received approximately 35 USD in local currency for the two-hour session; the protocol was approved by our institutional review board.

Table 2. Self-reported user study participant demographics

ID	Gender	Age	Background	Design Exp.	UI Design Tools	Figma-like Tool Fluency (1–7)	XR Familiarity (1–7)	XR UI Design Exp.
P1	M	24	Computer Science	<1 yr	Figma, code	5	3	–
P2	F	25	Industrial Design	4–5 yrs	Figma	7	2	–
P3	F	24	Industrial Design	4–5 yrs	Figma, code	6	4	AR application
P4	M	20	Industrial Design	1–2 yrs	Figma	7	2	AR goggle UI
P5	M	31	Computer Science	4–5 yrs	Figma, code	5	4	–
P6	F	26	Industrial Design	3–4 yrs	Figma	7	5	–
P7	F	26	Industrial Design	5+ yrs	Figma	6	5	VR navigation UI
P8	M	25	HCI (XR)	1–2 yrs	Figma, code, Unity	4	6	XR gesture-UI research
P9	F	25	Industrial Design	1–2 yrs	Figma, Framer	6	6	VR simulation UI
P10	F	21	Industrial Design	2–3 yrs	Figma, code	6	4	VR concept UI (Quest)
P11	M	27	UX Research/UI Design	4–5 yrs	Figma, code	7	5	AR application (Quest)
P12	F	26	Industrial Design	4–5 yrs	Figma	6	5	–

5.2 Study Procedure

Two authors attended each session: one moderated, while the other managed logging, observation, and assistance. Upon providing consent, participants received an explanation of the research background and completed an intake survey covering demographics and 7-point Likert-scale self-ratings of UI and context-aware design experience, Figma-like tool fluency, and XR usage familiarity. We then calibrated the Vision Pro’s eye and hand tracking for each participant, mirrored the in-headset view to a Mac mini for observation and logging, and provided brief training on moving, resizing, and deleting windows in visionOS.

Each participant then completed the two design tasks in counterbalanced order (six participants per order). After each task, participants briefly walked us through their artifacts, explaining what they adapted, when each state should appear, and what they changed after previewing or testing. They then completed the post-task survey; administering the debrief and survey after each task reduced recency bias in the end-of-session ratings. The baseline required no training because participants were already proficient with the tool. Before the ADAPTEST task, the authors provided a ten-minute spoken walkthrough of each capability while observing the mirrored view. This familiarized participants with the system’s basic operation before their task behavior was measured. The session concluded with a 15-minute semi-structured interview. Having used both tools, participants compared them, reflected on their own practices, and considered possible uses of ADAPTEST (Appendix C). Surveys were administered in English, with verbal clarification in the participant’s language upon request.

5.3 Conditions and Tasks

The two tasks used matched but distinct domains so that no participant authored the same artifact twice. **Task 1 (ADAPTEST)** ran on a smart-home control dashboard (living-room overview, thermostat, entry security) with the full system and suggestions enabled. **Task 2 (baseline)** used workplace information panels (team board, meeting-room status, wayfinding) and represented credible current practice [33]: participants edited on a desktop in Figma, inspected static PNG previews floating on Vision Pro through a minimal view-only image viewer, and returned to Figma between inspections to make revisions. Each domain comprised three screens built on the same scaffold—a header, a primary reading, a subtitle, one rich element (slider, dial, or image), three chips, and a primary button. The conditions therefore provided structurally equivalent authoring surfaces. The viewer eliminated the file-transfer friction of everyday practice while holding window chrome, passthrough blending, and logging constant. The baseline was consequently more convenient than the workflow it represented. The baseline stands for current Figma-based practice, not for XR

authoring tools with conditional-behavior support; it cannot express or execute environmental rules, so rule counts, Test Mode runs, and live triggering are capabilities of one condition only, and we read them as workflow differences. The tasks had intentionally different allocated time budgets. Because all participants were fluent Figma users, the familiar baseline received 15 minutes, whereas the unfamiliar ADAPTEST task received 30 minutes. Participants could exceed either budget upon request, so no one had to rush. We report behavioral outcomes as time-normalized rates alongside raw totals to account for this difference, and we revisit the condition-to-domain pairing in Limitations. We address this tradeoff in three ways: by using the matched scaffold described above; by analyzing reasoning-sensitive measures from the first block only, comparing the six ADAPTEST-first participants with the six baseline-first participants; and by reporting the results as ADAPTEST-on-home versus baseline-on-office.

Both tasks used the same brief: *“This panel will be used in [setting]. People will use it in different situations as the day and the space change. Design the three screens so they stay usable and clear for whoever is using them, however the situation changes.”* The brief did not identify environmental factors because RQ2 and RQ3 examine whether designers notice such conditions independently. Participants worked from provided seed screens, focusing the task on authoring adaptations to an existing design. We staged the factors without naming them: participants stepped back across the room (distance), the room lights were dimmed (brightness), and participants blocked the design with a hand (occlusion). Every participant thus met the same three-staged moments, and whether they addressed each one is a measured outcome. A task was considered complete when every screen included at least one working adaptive state and the participant had previewed or tested the design at least once. The ADAPTEST build used in the study sensed occlusion from tracked hands only, with scene-mesh occlusion disabled so that staging stayed predictable; distance was the head-to-window distance minus a fixed 1.8 m offset, floored at 0.2 m, so authored thresholds sit on that working scale; and brightness was ARKit light estimation normalized to the range seen since launch, so the app was launched in the study room before each block.

5.4 Data Collection and Analysis

We logged designers’ actions as structured events in both conditions: rules authored and their selected targets (RQ1); coverage of the three staged factors, combined rules, and suggestion accept/edit/dismiss decisions (RQ2); Test Mode runs, threshold and style edits with their timing relative to tests, test–refine cycles, and rule reuse across screens (RQ3). The Figma artifacts of all twelve baseline sessions were retained and coded for variants and their intended conditions. One author inventoried each file from its pages and layer names together with the walkthrough recording, listing every on-page state as a variant and recording its trigger from a canvas annotation, a layer name, or the participant’s spoken explanation; a second author reviewed all twelve inventories, and uncertain cases were settled by discussion. A staged factor counts as covered when at least one variant’s trigger names it, and a variant whose trigger names two factors counts as a compound condition. With ADAPTEST, the same counts come from the authored rules, so the baseline criterion is the looser of the two.

After each task, participants rated raw NASA-TLX (six subscales, 1–10) and SUS; each block additionally carried tool-specific items on 7 point Likert scale—two workflow items in the baseline block (*“I could clearly communicate when the interface should change”*; *“I could directly check whether the adaptation worked in XR”*) and five capability items in the ADAPTEST block (element selection, rules, previews, Test Mode, suggestions). Qualitative data comprise the audio-recorded per-task answers and the final semi-structured interview.

Within-subjects contrasts use Wilcoxon signed-rank tests with matched-pairs effect size r , alongside descriptive $M \pm SD$ for both conditions. Reasoning-sensitive measures are read from

the first block only (six versus six; Mann–Whitney U , Cliff’s δ), reported descriptively because twelve participants cannot power a between-subjects hypothesis test. Spearman correlations relate background ratings to perceived and behavioral outcomes. Interview and per-task answers were transcribed, translated, and deductively grouped by research question, first by one author, checked by another author.

Table 3. Session overview per condition: mean (SD), with the Wilcoxon signed-rank p where the measure is comparable.

	ADAPTEST	Baseline	p
Adaptive versions (rules / static variants)	4.6 (2.6)	3.2 (1.5)	—
Session length (min)	27.6 (6.3)	16.8 (2.7)	—
Adaptive versions per minute	0.16 (0.07)	0.20 (0.10)	.57
Raw NASA-TLX overall (0–10)	4.7	4.7	1.0
SUS (0–100)	67.9 (16.6)	69.8 (21.7)	.45

6 Results and Findings

In the smart-home domain, designers authored more adaptive behavior with ADAPTEST than they did in the office domain with their familiar workflow, at comparable workload and usability. With ADAPTEST, all 12 participants captured how the interface should adapt as 55 rules (Table 3), and every rule produced a working adaptive version of the design—a live, condition-specific version, not just a note describing one. In the baseline, participants produced 38 adaptive versions as static variants, with the conditions that should trigger them only written beside the design or spoken. Because the two tasks ran on different time budgets by design, we read this comparison as an authoring rate (Figure 10a): the two rates did not differ reliably¹. The volume difference thus reflects sustained authoring over a longer session at a comparable rate. Self-reported workload was comparable across conditions: raw NASA-TLX showed no reliable difference in overall workload. Performance was the only subscale approaching a reliable difference, with participants reporting marginally greater task success with ADAPTEST ($M = 7.2$ vs. 5.8 of 10, $p = .059$, $r = .39$; all other subscales $p \geq .23$). SUS scores also did not differ reliably between ADAPTEST and the baseline. We report behavioral findings and quantitative results for each research question, followed by participants’ preferences between the two workflows.

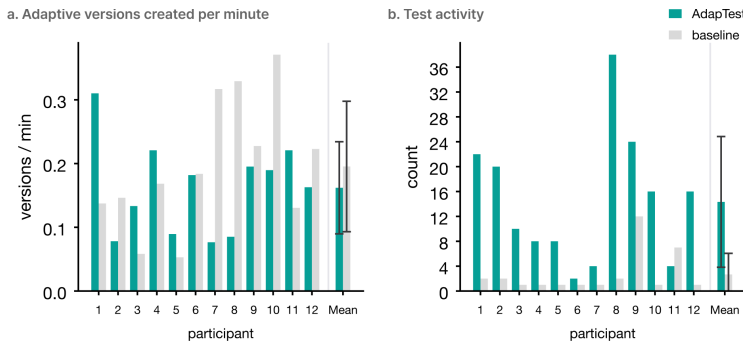


Fig. 10. Cross-condition comparison per participant: (a) adaptive versions authored per wall-clock minute, ADAPTEST beside baseline; (b) test activity—ADAPTEST Test Mode runs beside baseline headset-on inspections.

¹Throughout, we call a difference *reliable* when a statistical test indicates it is unlikely to be due to chance (here, $p < .05$).

6.1 Writing Adaptive Behavior on Specific Elements

With ADAPTTEST, participants recorded how the interface should change as working rules attached to specific elements. In the baseline, the same intent stayed in notes and speech. Of the 55 authored rules, 44 bound specific elements from the original design (2 to 9 elements per rule, $M = 5.6$, $SD = 1.8$) and 11 applied to a whole screen. Participants rated both mechanisms as helpful for specification: selecting target elements helped them state *what* should change ($M = 6.4$ of 7, $SD = 0.8$), while writing IF/THEN rules helped them state *when* it should change ($M = 6.2$, $SD = 1.1$). P1 contrasted this with their usual practice: *“Setting the IF/THEN was much easier; in Figma you could really only write down in text what should happen in which situation, so specifying it here was far easier.”*

In the baseline artifacts, activation logic was written as text or verbally explained. We coded the twelve baseline Figma files alongside the session recordings: seven participants recorded triggering conditions as canvas annotations or layer names, while the other five explained them only during the walkthrough—P8’s two-stage proximity fade, for example, existed as four overlapped static states whose triggering logic was described in speech alone. Thresholds stayed symbolic where ADAPTTEST participants set numbers: P12 annotated the baseline variants with literal placeholders (*“distance > N m”*), explaining that only testing could determine the value. Baseline participants also reached beyond the staged factors: six of twelve built adaptive versions triggered by time, touch, eye gaze, hand-reach, or the user’s own movement—trigger types their static tool could neither encode nor fire. Consistent with this gap between intent and artifact, baseline participants rated *“I could clearly communicate when the interface should change using this workflow”* at $M = 4.4$ of 7 ($SD = 1.9$), compared with 6.2 for the corresponding ADAPTTEST rule item.

Rules could also be reused instead of remade as separate copies. Six of twelve participants carried at least one rule condition onto a second or third screen (13 replicated conditions in the final projects, and P9 applied one distance rule across all three screens). P1, whose five replicated conditions span two screens, described the mechanism: *“the other designs follow the same principle—reduce detail when far—so after setting it once I could just paste the rules over to them.”* Authoring inside the rules still had limits. P11 found detailed content work harder in the headset (*“With Figma, authoring the window content itself was much easier—adding, editing, changing text”*), but distinguished it from adaptive authoring, describing Figma output as *“making a pretty poster and sticking it up”* whereas ADAPTTEST *“makes you consider ... designing the interface around actions and spatial relations.”*

6.2 Noticing and Covering Overlapping Conditions

Participants covered more of the staged environmental factors when authoring with rules. With ADAPTTEST, a factor counts as covered when an authored rule senses it; in the baseline, a factor counts even when its condition exists only in the participant’s spoken explanation, a looser standard, so the comparison is descriptive. Even under this looser baseline criterion, participants addressed more staged factors with ADAPTTEST ($M = 2.4$, $SD = 0.7$; six of twelve covered all three) than in the baseline ($M = 1.4$, $SD = 1.2$; three of twelve covered all three). This within-subjects difference was reliable (Wilcoxon $W = 3.5$, $p = .027$, $r = .64$). The comparison also differs in task domain and time budget, and coverage could carry across blocks, so we read it as a workflow difference and interpret it alongside the more conservative first-block comparison below. The authored rules addressed all three sensed conditions (Figure 11): distance in 36 rules ($N = 12$), brightness in 16 ($N = 9$), and occlusion in 14 ($N = 8$). Combined rules were counted under both constituent factors; consequently, these counts overlap and sum to more than 55. Even with ADAPTTEST, coverage was not complete: every participant covered distance, but three still skipped brightness and four skipped occlusion, and P7 even covered occlusion in the baseline yet not with ADAPTTEST. Rule authoring

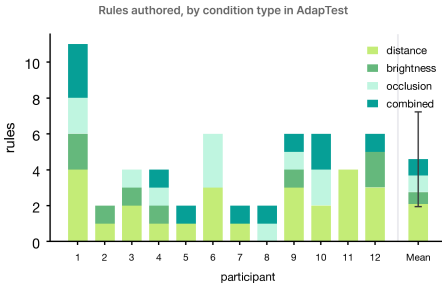


Fig. 11. Rules per participant by condition type. The right bar is the mean with ± 1 SD.

Condition	Rules	Participants	Median threshold
distance > x	16	8	2.9 m
distance < x	8	4	2.55 m
brightness < x	7	5	0.36
brightness > x	1	1	0.80
occlusion > x	6	4	0.66
occlusion < x	5	3	0.00
brightness + distance	8	5	
distance + occlusion	3	3	
distance range	1	1	1–2 m
Total	55	12	
Styling output	Rules	Participants	
hide	39	12	
opacity	24	11	
color	15	8	
size	14	8	
reorder	6	5	

Table 4. Authored rules by condition, and with styling outputs. A rule may use several outputs, and six rules used none.

increased coverage but did not ensure that every factor was addressed. Table 4 summarizes the patterns across the 55 final rules. Hiding elements at a far distance was the modal rule, in 15 of the 16 single distance-greater-than rules, and compound rules mostly paired low brightness with a distance condition. Six rules carried a condition but no styling change.

Combined conditions were particularly more common with ADAPTEST. Nine of twelve participants authored at least one combined rule with ADAPTEST (twelve such rules in all, eleven surviving in the final projects), against two of twelve who built a combined variant in the baseline. P4 named the difference: “In Figma you can’t have two conditions at once, so you keep two separate variants and make a whole new variant for when both apply.” The same direction appears in each participant’s first task, before either tool’s practice could carry into the other block: those who started with ADAPTEST covered $M = 2.5$ factors, and those who started with the baseline covered $M = 1.0$ ($U = 30.0$, $p = .054$, Cliff’s $\delta = 0.67$). Both groups received the same brief, which named no environmental factors. The observed difference is therefore consistent with ADAPTEST making coverage gaps visible during authoring, whereas the baseline provided no equivalent cue. With six participants per group, we read it as a supporting pattern.

The rule-attached previews—live renderings of how each rule changes the design, generated and updated by the tool—were the capability participants endorsed most strongly, and they helped designers spot problems as they worked. The preview item scored highest of the five ($M = 6.9$ of 7, $SD = 0.3$), and participants described editing from the previews directly—P4: “More than the test, I mostly looked at the preview next to the rule and made changes based on that,” and P7: “because hides showed up immediately, I tried hiding one more element than planned, saw it looked fine, and kept it.” Baseline use further highlighted the value of side-by-side comparison. In the baseline block, P7 “regretted not bringing the other version along to compare” when checking far and near placements.

Suggestions received the lowest rating among the five capabilities ($M = 5.3$ of 7, $SD = 0.8$), but still contributed to the authored rule sets. Four participants accepted seven suggestions in total, and the five accepted combination suggestions account for five of the twelve combined rules above. P12 was shown seven suggestions and accepted none, yet covered all three staged factors with self-authored rules, and still projected the value forward: “the suggestion feature could catch cases I was missing—when things get complicated, it could really help.” P11 recalled a suggestion catching a combined case: “the house could be dark ... when the rule suggestion popped up during the pre-test earlier, it felt really convenient.”



Fig. 12. Workflow timelines of all twelve ADAPTTEST sessions: activity lanes (Style, Rule, Arrange, Preview) over elapsed time, with Test Mode spans as full-height bands and idle pauses hatched.

6.3 Testing Rules in the Real Environment

Participants ran repeated tests during the session (Figure 12). They ran Test Mode $M = 14.3$ times per session ($SD = 10.5$, range 2–38), completed $M = 10.2$ test-refine cycles ($SD = 4.6$), and edited rule conditions and thresholds $M = 46.1$ times ($SD = 27.3$). Testing occupied about a quarter of active session time ($M = 26\%$, $SD = 9\%$). Test Mode hides the editing panels while it runs, so a participant who spots a problem during a test must stop the test before making the fix. We therefore counted an edit as test-prompted when it occurred during a test or within 2 minutes after a test ended: 430 of the 1,218 style and condition edits (35%) were test-prompted. Based on build-test alternation, five participants interleaved testing throughout the session, each making at least 15 build-test switches, while seven concentrated testing in later verification passes.

The baseline afforded far less of this in-context checking. Participants could run ADAPTTEST Test Mode immediately during authoring ($M = 14.3$ runs). In the baseline, testing required donning the headset to inspect exported images and occurred much less frequently ($M = 2.7$ inspections). This difference in the frequency of in-context checking is descriptive (Figure 10b), because the two measures represent different activities: running a live Test Mode session and donning the headset to inspect a static image, and the baseline could not run a live test by construction. The two workflows also placed testing at different times (Figure 13): with ADAPTTEST, testing wove through authoring, whereas the baseline separated a long stretch of off-headset Figma design from a few late inspections.

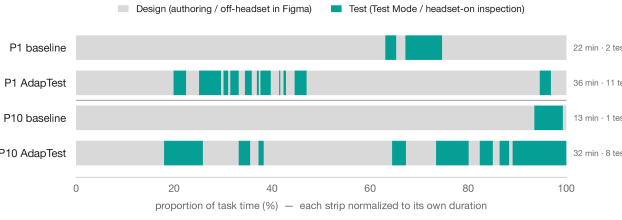


Fig. 13. When testing occurred for P1 and P10 in each condition (strip normalized to own session length).

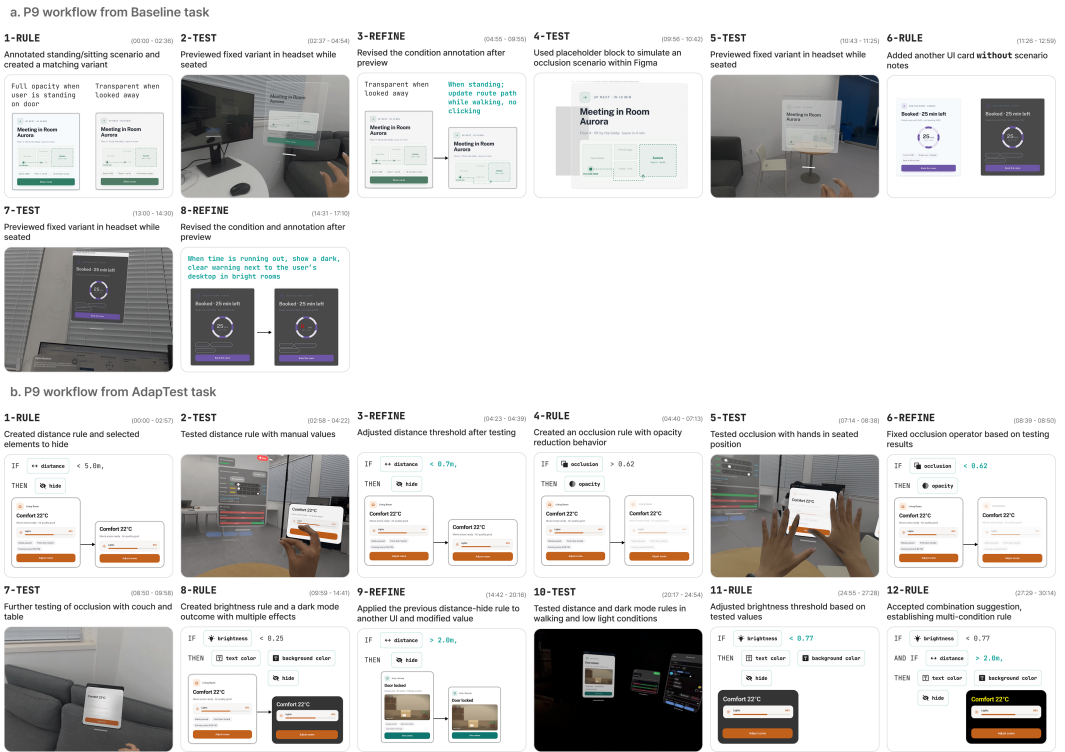


Fig. 14. P9's workflow (completed the baseline task first), split into numbered steps and labeled by phase: **a)** baseline condition, **b)** ADAPTEST condition.

Participants rated Test Mode highly for checking whether adaptations worked in XR ($M = 6.2$ of 7, $SD = 1.2$), compared with $M = 4.3$ ($SD = 2.0$) for the corresponding baseline item. P11 mapped the loop onto a familiar practice: “It was very similar to working in Unity—fix a bit, hit play, test it, then revise again.” Participants described setting thresholds and styling from guesses into observed values during tests. P9 calibrated a distance rule against the live reading (Figure 14): “Watching the live reading, I measured how far the threshold should be—it was about 2m to that door,” and P3 described the same conversion: “only by actually trying did I learn at what closeness or distance it applies.” Styling changed on observation too—P12: “when I actually tested it I thought ‘this is brighter than I expected, I should reduce it’ and redid it,” and P8 dropped the “Living Room” label from a reduced state after testing showed it unnecessary. P1 contrasted this with the baseline’s

stand-ins—“for ‘dark’ you’d just fill a black rectangle behind it, so having it actually happen was much better”—and P10 located the difference: “with Figma the work happens only in my imagination.”

The physical room bounded what Test Mode could enact, and participants split on what should stay physical. P1 wanted a farther condition than the room allowed—“I wanted to see how it looks when it’s tiny from the far end, but the test couldn’t show me that far”—and P2 wanted more extreme conditions than the session offered. P8 asked for a virtual distance control for this reason, yet kept one condition physical: “Occlusion should be tested by physically doing it.” P11 explained why embodiment mattered for numbers at all: while authoring, “I’m sitting still and can’t gauge what 1m or 2m means anyway”—the values became judgeable only in motion.

6.4 Preferences Between Workflows

Participants rated all five ADAPTTEST capabilities above the midpoint (previews, element selection, rules, Test Mode, and suggestions, in order) and the interviews clarified why they preferred one workflow over the other for a given kind of work. Four reasons recurred: immediacy, embodied judgment of values, precision of content edits, and learning cost.

Participants placed the workflow inside practices they already knew. P5 recognized their own habits: “Using the Vision Pro actually feels much more natural to me, because I usually design by quickly roughing something out, placing it, and looking at it,” and P9 saw a round trip disappear: “Designing VR content on a computer means you still have to re-verify it in VR anyway, so with a good system doing it in VR is much better.” Learning cost and content precision favored the baseline. Participants rated ADAPTTEST higher on “I needed to learn a lot of things before I could get going” ($M = 3.0$ vs. 2.1 of 5 , $p = .016$, $r = .49$), and precise content edits such as changing text stayed easier on the desktop. Looking beyond the study task, P7 identified where coverage work becomes particularly important in practice: “For shipping products, companies imagine many very specific edge cases, so this environmental control would be even more useful there,” and P5 noted the enabling condition: “normally we never considered [distance and brightness] precisely because everyday sensors can’t give exact distance; with accurate data the UI experience could get much better.”

7 Discussion

Our formative study surfaced three findings: designers already adapt concrete interface elements with a small visual vocabulary, they lose track of the growing set of environmental cases, and they can judge thresholds and transitions only in the space itself. ADAPTTEST translated each finding into a design goal: element-targeted rule authoring (DG1), multi-condition preview comparison (DG2), and situated testing of the active rule (DG3). The comparative study then showed designers taking up all three capabilities while keeping their own workflows: all twelve externalized adaptive intent as working rules, they covered more of the staged factors, and testing wove through authoring in two distinct rhythms, interleaved throughout or concentrated in later verification passes. We discuss what these results mean for each design goal and close with limitations and future work.

7.1 Authoring Adaptivity as Designer-Owned Rules

Element-targeted rules kept the condition-appearance mapping in the designer’s hands, and designers used that ownership. All twelve participants stated when and how their interfaces should change as working rules, where the same participants’ baseline artifacts held that intent in annotations and speech. They also rated their ability to communicate change conditions higher with rules (6.2 vs. 4.4 of 7). Prior work has advanced the complementary side of this division: automated systems decide adaptations at runtime through solvers and optimizers [10, 16, 40], increasingly with user preferences steering the objectives [28, 41], and even when designers pre-author the variants, the runtime owns the mapping from context to variant [42]. Trigger-action tools established that

non-programmers can author rich behavior over discrete events [18, 26, 64, 65]. Our study adds that visual designers, given concrete elements and continuous environmental triggers, author this mapping fluently and treat it as part of the design itself—P11 described the shift as designing “around actions and spatial relations” where static tools produce posters.

The triggers designers reached for on their own show where rule authoring should extend next. Six of twelve baseline participants authored states for triggers the study did not stage—time, touch, eye gaze [29], hand-reach, and the user’s own movement—even though the task brief and staging pointed only at environmental factors. These triggers fall into four families beyond the environmental one ADAPTEST senses: interaction (touch, gaze, hand-reach), time of day, the user’s own movement and proximity, and, in the fixture ideas below, the presence of other people. Each family maps to a different use case, from a museum panel that reacts to a visitor’s approach to a home panel that goes quiet at night. Several of these families have long precedents: proxemic interaction research established distance, orientation, and movement as interaction dimensions [35, 47, 59], and gradual engagement described the staged reveal-on-approach behavior that P8 rebuilt by hand from static states [46]. The sensing capabilities these triggers require (e.g., eye tracking, hand tracking, and position tracking) are already available on the headset, and adaptation drivers beyond the environment, such as privacy, are emerging in parallel [52]. Extending element-targeted rules to these trigger families would let a single IF/THEN construct carry both interaction- and context-driven adaptivity, keeping the rule set as a single inspectable record of both.

Designers also imagined interfaces that take on a role in the room, not only a position within it. In both studies, participants pictured panels as interior fixtures (e.g., a glanceable clock blended into a desk, or a panel by the door that becomes a video-doorphone as someone approaches) so the interface’s anchor and the activity around it, alongside ambient conditions, shaped how it should behave. This extends the reveal-on-approach of gradual engagement [46] to authored 2D interfaces: treating an interface’s placement and the presence of people near it as conditions, and its role—calm and glanceable at rest, interactive on approach—as an output, would let ADAPTEST author interfaces that behave as ambient objects anchored in a space.

7.2 Seeing Coverage Before Deployment

With condition-specific previews, coverage became an inspection task, and in the baseline it stayed a memory task. Participants addressed more of the staged factors with the tool (2.4 vs. 1.4 of 3), treated combined conditions as routine (nine of twelve authored combined rules), and rated previews highest of the five capabilities (6.9 of 7)—editing directly from them. Comparing interface alternatives side by side is an established aid to design judgment [13, 23, 56, 58]. Those comparisons weigh alternatives a designer freely explores, and condition-indexed previews adapt the same aid to a different question: whether one design stays legible and usable under each situation it will meet—a concern readability research in AR documents across lighting, background, and distance [6]. Coverage still did not become complete—brightness and occlusion trailed distance, and one participant covered occlusion only in the baseline—so visible previews help designers inspect coverage without ensuring they act on every gap. The shift from remembering conditions to inspecting them is not specific to XR: any design that must hold across a space of conditions—responsive layouts across screen sizes, light and dark themes, or accessibility variants—meets the same risk of uncovered cases, and condition-indexed previews with coverage strips could carry the same inspection aid to those settings, as keyframe authoring already spans screen size, ability, and distance for 2D interfaces [61].

The suggestion results sketch how assistance might close those remaining gaps. Accepted combination suggestions seeded five of the twelve combined rules, so acceptance did direct work. At the same time, P12 saw seven suggestions, accepted none, and covered all three factors with

self-authored rules, while rating the feature as able to “catch cases I was missing.” Suggestions may therefore help twice: as rules a designer adopts, and as prompts that provoke the designer to author a response of their own. Assistance for coverage could lean into the second mode—surfacing an uncovered condition as a visible gap beside the previews and leaving the authored answer to the designer.

7.3 Testing as Part of Authoring

Situated testing became a design activity woven through authoring in the ADAPTEST condition, where XR practice studies had described it as a costly separate step [1, 48]. Testing filled about a quarter of active session time; 35% of edits followed a test within 2 minutes; and participants described setting thresholds from observed values—P9 set a distance threshold by walking to a door and reading 2m. Authoring rate did not differ reliably from the baseline while sessions ran longer, so the added time went into testing and refining. The rhythms differed and the tool supported each: seven participants concentrated testing in later verification passes, while five interleaved tests from the start—at cadences from rapid edit–test alternation (P8 ran 38 tests around two rules) to slower build–test cycles. In-situ authoring tools established creation in the target environment [20, 27, 39, 44, 57], where testing reflects the conditions the room offers at that moment. The active-rule Test Mode adds the ability to dial a condition on demand and know which authored state is showing.

Seen across workflow types, the trade-off runs along where checking happens. A fully 2D workflow, as in the baseline, keeps content edits precise and reuses the design system, and it lets designers name any trigger they can imagine. Each condition, though, stays as text or speech, and every check needs a headset round trip that P5 and P9 described as a separate step (Section 6.4). A fully XR workflow, as in ShapesXR [57] or the Unity play loop P11 compared Test Mode to, gives immediacy and spatial judgment from the start. It gives up the fine content editing and asset reuse that participants missed in the headset. We did not evaluate such a tool, so this reading rests on prior work and on participants’ comparisons. ADAPTEST sits between the two. The 2D design stays the source, rules execute and are tested in place, and coverage is visible in previews, at the entry cost participants reported. Which arrangement suits a project depends on whether its conditions can be judged from the desk, how much fine editing remains, and how often the design must be checked in the room. Participants’ requests draw a concrete design space for simulating conditions the room cannot produce. P1 wanted a viewing distance beyond the room’s far wall, P2 wanted more extreme conditions than the session offered, and P8 asked for a virtual distance control—yet P8 insisted occlusion “should be tested by physically doing it,” and P11 could judge what 1m meant only while moving. Simulation and embodiment thus split by condition: distance and lighting invite virtual extension—simulating AR conditions from within VR has precedent [7], and headset immersive environments could present a twenty-meter hall or a dark room inside a small lab—while occlusion and threshold judgment draw their value from the body. Future situated-testing tools could offer simulated ranges for the former while keeping the latter physical, widening testable conditions without leaving the authoring session.

The same executable, in-place rules suggest a role in collaboration that our single-user sessions did not test. Because an authored rule set runs and can be entered in Test Mode, a teammate or developer could step into the space and experience the intended adaptive behavior directly, where a mockup or written specification leaves that behavior to the imagination—a shared, situated reference between designers and the developers who implement the result [4, 25, 48]. Shared rule sets and semantic roles could likewise keep a team’s adaptive behavior consistent across screens and products, and experiencing the behavior together might shorten the design–development iterations that adaptive intent otherwise invites.

7.4 Limitations and Future Work

The main limitation of our study is that each tool was tested on its own fixed task domain, so the comparison is not a pure tool contrast. We fixed this condition-to-domain pairing to keep the physical staging comparable, and the results therefore compare ADAPTEST-on-home with baseline-on-office under matched task structure. The shared screen scaffold and the first-block contrast mitigate this, and a crossed replication would isolate the tool effect. The twelve single-session participants likewise capture only first-contact use, representing one point in the broader space of evaluation strategies [36], and the staged conditions were bounded by what one study room could produce. Longer-term use—on designers’ own projects, with the learning cost amortized—remains to be studied. The evaluation therefore supports claims about the authoring and testing process and about participants’ perceptions in staged tasks. It does not show that the resulting interfaces are more usable for end users, that deployed adaptations behave better, or that developer handoff improves, since none of these was measured.

The tool itself has limitations that point to future work. Participants rated it higher in terms of needing to learn before starting, and fine content edits remained easier on the desktop. Its three sensing factors cover part of designers’ adaptive intent, and the simulation directions above, together with the additional trigger families in Section 7.1, would broaden both the authorable and testable ranges. The sensing itself is uncharacterized. Brightness is normalized to the range seen since launch, so a threshold is relative to that session and to camera exposure. Occlusion in the study came from tracked hands only, so physical obstacles and other virtual windows did not register. None of the three readings was validated against an instrument, so thresholds are comparable within a session and not across rooms. Environmental sensing also carries ethical weight: the tool senses the surroundings only while the designer is testing, and the study stored structured interaction events without camera frames, room reconstructions, or bystander recordings, but tools that ship authored rules into everyday spaces will inherit the privacy questions of continuous sensing [52]. Other people in the room would be sensed without having agreed to it. A version that reads camera frames would also have to say what it keeps and for how long. Finally, the exported rules and variants now travel back to desktop tools, and evaluating that handoff with the developers and collaborating design teams who receive it—where XR development practice [4] and spatially distributed collaboration [25] set their own requirements—would be the next step.

8 Conclusion

We presented ADAPTEST, an authoring tool that lets interface designers make their 2D designs adaptive to the environments those designs will inhabit, through element-targeted rules, attached condition previews, and situated testing of the active rule. Designers pick the elements to change by tapping them on the rendered design, state when and how they should change as IF/THEN rules over distance, lighting, and occlusion, compare each rule’s outcome in previews beside the default design, and validate the behavior in place under manual controls or live sensing. The tool accommodates designers’ own working rhythms, supporting testing woven through authoring as well as verification concentrated at the end. ADAPTEST was designed from a formative study with twelve designers and evaluated with twelve screen-rooted designers against their familiar desktop-and-preview workflow. Participants took up all three capabilities, externalizing adaptive intent as working rules, covering more of the staged environmental conditions, and refining thresholds and styling from what they observed in tests, at workload and usability comparable to the tools they already knew. Future work will extend rules to the trigger families designers already reach for, simulate conditions a single room cannot stage, and follow authored rules into longer-term practice and handoff.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT)(RS-2024-00337803).

References

- [1] 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. doi:10.1145/3313831.3376722
- [2] Till Ballendat, Nicolai Marquardt, and Saul Greenberg. 2010. Proxemic interaction: designing for a proximity and orientation-aware environment. In *ACM International Conference on Interactive Tabletops and Surfaces* (Saarbrücken, Germany) (ITS '10). Association for Computing Machinery, New York, NY, USA, 121–130. doi:10.1145/1936652.1936676
- [3] Blaine Bell, Steven Feiner, and Tobias Höllerer. 2001. View management for virtual and augmented reality. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology* (Orlando, Florida) (UIST '01). Association for Computing Machinery, New York, NY, USA, 101–110. doi:10.1145/502348.502363
- [4] Ingo Börsting, Markus Heikamp, Marc Hesenius, Wilhelm Koop, and Volker Gruhn. 2022. Software Engineering for Augmented Reality - A Research Agenda. *Proc. ACM Hum.-Comput. Interact.* 6, EICS, Article 155 (June 2022), 34 pages. doi:10.1145/3532205
- [5] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (2006), 77–101. arXiv:<https://www.tandfonline.com/doi/pdf/10.1191/1478088706qp0630a> doi:10.1191/1478088706qp0630a
- [6] Maxime Cauz, Antoine Clarinval, and Bruno Dumas. 2024. Text readability in augmented reality: a multivocal literature review. *Virtual Reality* 28, 1 (04 Mar 2024), 59. doi:10.1007/s10055-024-00949-6
- [7] Marco Cavallo and Angus G. Forbes. 2019. CAVE-AR: A VR Authoring System to Interactively Design, Simulate, and Debug Multi-user AR Experiences. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 872–873. doi:10.1109/VR.2019.8798148
- [8] Jonathan Chen and Dongwook Yoon. 2024. Exploring the Diminishing Allure of Paper and Low-Fidelity Prototyping Among Designers in the Software Industry: Impacts of Hybrid Work, Digital Tools, and Corporate Culture. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI '24). Association for Computing Machinery, New York, NY, USA, Article 1024, 14 pages. doi:10.1145/3613904.3642774
- [9] Wenqian Chen, Yifei Shan, Yue Wu, Zihan Yan, and Xiangdong Li. 2021. Design and Evaluation of a Distance-Driven User Interface for Asynchronous Collaborative Exhibit Browsing in an Augmented Reality Museum. *IEEE Access* 9 (2021), 73948–73962. doi:10.1109/ACCESS.2021.3080286
- [10] Yifei Cheng, Yukang Yan, Xin Yi, Yuanchun Shi, and David Lindlbauer. 2021. SemanticAdapt: Optimization-based Adaptation of Mixed Reality Layouts Leveraging Virtual-Physical Semantic Connections. 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, 282–297. doi:10.1145/3472749.3474750
- [11] Shakiba Davari and Doug A. Bowman. 2024. Towards Context-Aware Adaptation in Extended Reality: A Design Space for XR Interfaces and an Adaptive Placement Strategy. arXiv:2411.02607 [cs.HC] <https://arxiv.org/abs/2411.02607>
- [12] Juan David Hincapié-Ramos, Levko Ivanchuk, Srikanth K. Sridharan, and Pourang P. Irani. 2015. SmartColor: Real-Time Color and Contrast Correction for Optical See-Through Head-Mounted Displays. *IEEE Transactions on Visualization and Computer Graphics* 21, 12 (2015), 1336–1348. doi:10.1109/TVCG.2015.2450745
- [13] Steven P. Dow, Alana Glassco, Jonathan Kass, Melissa Schwarz, Daniel L. Schwartz, and Scott R. Klemmer. 2011. Parallel prototyping leads to better design results, more divergence, and increased self-efficacy. *ACM Trans. Comput.-Hum. Interact.* 17, 4, Article 18 (Dec. 2011), 24 pages. doi:10.1145/1879831.1879836
- [14] Barrett Ens, Juan David Hincapié-Ramos, and Pourang Irani. 2014. Ethereal planes: a design framework for 2D information space in 3D mixed reality environments. In *Proceedings of the 2nd ACM Symposium on Spatial User Interaction* (Honolulu, Hawaii, USA) (SUI '14). Association for Computing Machinery, New York, NY, USA, 2–12. doi:10.1145/2659766.2659769
- [15] Austin Erickson, Kangsoo Kim, Alexis Lambert, Gerd Bruder, Michael P. Browne, and Gregory F. Welch. 2021. An Extended Analysis on the Benefits of Dark Mode User Interfaces in Optical See-Through Head-Mounted Displays. *ACM Trans. Appl. Percept.* 18, 3, Article 12 (May 2021), 22 pages. doi:10.1145/3456874
- [16] João Marcelo Evangelista Belo, Mathias N. Lystbæk, Anna Maria Feit, Ken Pfeuffer, Peter Kán, Antti Oulasvirta, and Kaj Grönbæk. 2022. AUIT – the Adaptive User Interfaces Toolkit for Designing XR Applications. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (Bend, OR, USA) (UIST '22). Association for Computing Machinery, New York, NY, USA, Article 48, 16 pages. doi:10.1145/3526113.3545651

- [17] José Ferrão, Paulo Dias, Beatriz Sousa Santos, and Miguel Oliveira. 2023. Environment-Aware Rendering and Interaction in Web-Based Augmented Reality. *Journal of Imaging* 9, 3 (2023). doi:10.3390/jimaging9030063
- [18] Vittoria Frau, Lucio Davide Spano, Valentino Artizzu, and Michael Nebeling. 2023. XRSpotlight: Example-based Programming of XR Interactions using a Rule-based Approach. *Proc. ACM Hum.-Comput. Interact.* 7, EICS, Article 185 (June 2023), 28 pages. doi:10.1145/3593237
- [19] Ran Gal, Lior Shapira, Eyal Ofek, and Pushmeet Kohli. 2014. FLARE: Fast layout for augmented reality applications. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 207–212. doi:10.1109/ISMAR.2014.6948429
- [20] Danilo Gasques, Janet G. Johnson, Tommy Sharkey, and Nadir Weibel. 2019. PintAR: Sketching Spatial Experiences in Augmented Reality. In *Companion Publication of the 2019 on Designing Interactive Systems Conference 2019 Companion (San Diego, CA, USA) (DIS '19 Companion)*. Association for Computing Machinery, New York, NY, USA, 17–20. doi:10.1145/3301019.3325158
- [21] Violet Yinuo Han, Hyunung Cho, Kiyosu Maeda, Alexandra Ion, and David Lindlbauer. 2023. BlendMR: A Computational Method to Create Ambient Mixed Reality Interfaces. *Proc. ACM Hum.-Comput. Interact.* 7, ISS, Article 436 (Nov. 2023), 25 pages. doi:10.1145/3626472
- [22] Chris Harrison and Anind K. Dey. 2008. Lean and zoom: proximity-aware user interface and content magnification. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (Florence, Italy) (CHI '08)*. Association for Computing Machinery, New York, NY, USA, 507–510. doi:10.1145/1357054.1357135
- [23] Björn Hartmann, Loren Yu, Abel Allison, Yeonsoo Yang, and Scott R. Klemmer. 2008. Design as exploration: creating interface alternatives through parallel authoring and runtime tuning. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology (Monterey, CA, USA) (UIST '08)*. Association for Computing Machinery, New York, NY, USA, 91–100. doi:10.1145/1449715.1449732
- [24] Fengming He, Xiyun Hu, Xun Qian, Zhengzhe Zhu, and Karthik Ramani. 2024. AdaptTUI: Adaptation of Geometric-Feature-Based Tangible User Interfaces in Augmented Reality. *Proc. ACM Hum.-Comput. Interact.* 8, ISS, Article 527 (Oct. 2024), 26 pages. doi:10.1145/3698127
- [25] Jaylin Herskovitz, Yi Fei Cheng, Anhong Guo, Alanson P. Sample, and Michael Nebeling. 2022. XSpace: An Augmented Reality Toolkit for Enabling Spatially-Aware Distributed Collaboration. *Proc. ACM Hum.-Comput. Interact.* 6, ISS, Article 568 (Nov. 2022), 26 pages. doi:10.1145/3567721
- [26] IFTTT. [n. d.]. IFTTT - Automate business & home. <https://ifttt.com/> Accessed: 2025-04-17.
- [27] Ramya Iyer, Mustafa Doga Dogan, Maria Larsson, and Takeo Igarashi. 2025. XR-penter: Material-Aware and In Situ Design of Scrap Wood Assemblies. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 689, 16 pages. doi:10.1145/3706598.3713331
- [28] Christoph Albert Johns, João Marcelo Evangelista Belo, Anna Maria Feit, Clemens Nylandsted Klokmose, and Ken Pfeuffer. 2023. Towards Flexible and Robust User Interface Adaptations With Multiple Objectives. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (San Francisco, CA, USA) (UIST '23)*. Association for Computing Machinery, New York, NY, USA, Article 108, 17 pages. doi:10.1145/3586183.3606799
- [29] Anam Ahmad Khan, Florian Weidner, Jungwoo Rhee, Yasmeen Abdrabou, Andrea Bianchi, Eduardo Velloso, Hans Gellersen, and Joshua Newn. 2026. Gaze and Speech in Multimodal Human-Computer Interaction: A Scoping Review. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26)*. Association for Computing Machinery, New York, NY, USA, Article 672, 24 pages. doi:10.1145/3772318.3791662
- [30] Talha Khan and David Lindlbauer. 2024. Don't Block My Stuff: Fostering Personal Object Awareness in Multi-user Mixed Reality Environments. *Proc. ACM Hum.-Comput. Interact.* 8, ISS, Article 526 (Oct. 2024), 24 pages. doi:10.1145/3698126
- [31] Kangsoo Kim, Austin Erickson, Alexis Lambert, Gerd Bruder, and Greg Welch. 2019. Effects of Dark Mode on Visual Fatigue and Acuity in Optical See-Through Head-Mounted Displays. In *Symposium on Spatial User Interaction (New Orleans, LA, USA) (SUI '19)*. Association for Computing Machinery, New York, NY, USA, Article 9, 9 pages. doi:10.1145/3357251.3357584
- [32] Veronika Krauß, Florian Jasche, Sheree May Saßmannshausen, Thomas Ludwig, and Alexander Boden. 2021. Research and Practice Recommendations for Mixed Reality Design – Different Perspectives from the Community. In *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology (Osaka, Japan) (VRST '21)*. Association for Computing Machinery, New York, NY, USA, Article 24, 13 pages. doi:10.1145/3489849.3489876
- [33] Veronika Krauß, Michael Nebeling, Florian Jasche, and Alexander Boden. 2022. Elements of XR Prototyping: Characterizing the Role and Use of Prototypes in Augmented and Virtual Reality Design. 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 310, 18 pages. doi:10.1145/3491102.3517714
- [34] Chao-Jung Lai, Mauricio Sousa, Tianyu Zhang, Ludwig Sidenmark, and Tovi Grossman. 2025. Adaptique: Multi-objective and Context-aware Online Adaptation of Selection Techniques in Virtual Reality. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology (UIST '25)*. Association for Computing Machinery, New York, NY, USA, Article 50, 16 pages. doi:10.1145/3746059.3747790

- [35] David Ledo, Saul Greenberg, Nicolai Marquardt, and Sebastian Boring. 2015. Proxemic-Aware Controls: Designing Remote Controls for Ubiquitous Computing Ecologies. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services* (Copenhagen, Denmark) (*MobileHCI '15*). Association for Computing Machinery, New York, NY, USA, 187–198. doi:10.1145/2785830.2785871
- [36] 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
- [37] Benjamin Lee, Michael Sedlmair, and Dieter Schmalstieg. 2024. Design Patterns for Situated Visualization in Augmented Reality. *IEEE Transactions on Visualization and Computer Graphics* 30, 1 (Jan. 2024), 1324–1335. doi:10.1109/TVCG.2023.3327398
- [38] Jaewook Lee, Filippo Aleotti, Diego Mazala, Guillermo Garcia-Hernando, Sara Vicente, Oliver James Johnston, Isabel Kraus-Liang, Jakub Powierza, Donghoon Shin, Jon E. Froehlich, Gabriel Brostow, and Jessica Van Brummelen. 2025. ImagineAR: AI-Assisted In-Situ Authoring in Augmented Reality. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology* (UIST '25). Association for Computing Machinery, New York, NY, USA, Article 52, 21 pages. doi:10.1145/3746059.3747635
- [39] Germán Leiva, Jens Emil Grønæk, 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. doi:10.1145/3472749.3474774
- [40] Zhipeng Li, Christoph Gebhardt, Yves Inglin, Nicolas Steck, Paul Strel, and Christian Holz. 2024. SituationAdapt: Contextual UI Optimization in Mixed Reality with Situation Awareness via LLM Reasoning. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology* (Pittsburgh, PA, USA) (UIST '24). Association for Computing Machinery, New York, NY, USA, Article 43, 13 pages. doi:10.1145/3654777.3676470
- [41] Zhipeng Li, Christoph Gebhardt, Yi-Chi Liao, and Christian Holz. 2026. Automating UI Optimization through Multi-Agent Reasoning. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems* (CHI '26). Association for Computing Machinery, New York, NY, USA, Article 1139, 20 pages. doi:10.1145/3772318.3791444
- [42] David Lindlbauer, Anna Maria Feit, and Otmar Hilliges. 2019. Context-Aware Online Adaptation of Mixed Reality Interfaces. 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, 147–160. doi:10.1145/3332165.3347945
- [43] Aran Lunzer and Kasper Hornbæk. 2008. Subjunctive interfaces: Extending applications to support parallel setup, viewing and control of alternative scenarios. *ACM Trans. Comput.-Hum. Interact.* 14, 4, Article 17 (Jan. 2008), 44 pages. doi:10.1145/1314683.1314685
- [44] Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. 2004. DART: a toolkit for rapid design exploration of augmented reality experiences. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology* (Santa Fe, NM, USA) (UIST '04). Association for Computing Machinery, New York, NY, USA, 197–206. doi:10.1145/1029632.1029669
- [45] Emanuele Marino, Fabio Bruno, Loris Barbieri, Maurizio Muzzupappa, and Fotis Liarokapis. 2021. Background-Aware Colorization Technique for Augmented Reality Applications. *IEEE Access* 9 (2021), 161761–161772. doi:10.1109/ACCESS.2021.3129076
- [46] Nicolai Marquardt, Till Ballendat, Sebastian Boring, Saul Greenberg, and Ken Hinckley. 2012. Gradual engagement: facilitating information exchange between digital devices as a function of proximity. In *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces* (Cambridge, Massachusetts, USA) (ITS '12). Association for Computing Machinery, New York, NY, USA, 31–40. doi:10.1145/2396636.2396642
- [47] Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring, and Saul Greenberg. 2011. The proximity toolkit: prototyping proxemic interactions in ubiquitous computing ecologies. 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, 315–326. doi:10.1145/2047196.2047238
- [48] Nels Numan, Gabriel Brostow, Suhyun Park, Simon Julier, Anthony Steed, and Jessica Van Brummelen. 2025. CoCreatAR: Enhancing Authoring of Outdoor Augmented Reality Experiences Through Asymmetric Collaboration. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (CHI '25). Association for Computing Machinery, New York, NY, USA, Article 1232, 22 pages. doi:10.1145/3706598.3714274
- [49] Siyou Pei, David Kim, Alex Olwal, Yang Zhang, and Ruofei Du. 2024. UI Mobility Control in XR: Switching UI Positionings between Static, Dynamic, and Self Entities. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI '24). Association for Computing Machinery, New York, NY, USA, Article 611, 12 pages. doi:10.1145/3613904.3642220

- [50] Xiaohan Peng, Debanjana Halder, Wendy E. Mackay, and Janet Koch. 2026. DesignTrace: Exploring, Iterating and Tracking Design Alternatives with GenAI. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26)*. Association for Computing Machinery, New York, NY, USA, Article 434, 22 pages. doi:10.1145/3772318.3791036
- [51] Xun Qian, Tianyi Wang, Xuhai Xu, Tanya R. Jonker, and Kashyap Todi. 2024. Fast-Forward Reality: Authoring Error-Free Context-Aware Policies with Real-Time Unit Tests in Extended Reality. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI '24). Association for Computing Machinery, New York, NY, USA, Article 951, 17 pages. doi:10.1145/3613904.3642158
- [52] Shwetha Rajaram, Macarena Peralta, Janet G. Johnson, and Michael Nebeling. 2025. Exploring the Design Space of Privacy-Driven Adaptation Techniques for Future Augmented Reality Interfaces. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, Article 1226, 19 pages. doi:10.1145/3706598.3713320
- [53] André Corrêa Santos, Pedro Bittar Barão, Rafael Melhado Araujo Lima, and Luciano Pereira Soares. 2025. AI-Driven Responsive GUI for XR: Context-Aware Visual Adaptation. In *2025 27th Symposium on Virtual and Augmented Reality (SVR)*. 202–211. doi:10.1109/SVR67689.2025.00037
- [54] Marc Satkowski and Raimund Dachsel. 2021. Investigating the Impact of Real-World Environments on the Perception of 2D Visualizations in Augmented 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 522, 15 pages. doi:10.1145/3411764.3445330
- [55] Barbara Sommer. 2001. *A Practical Guide to Behavioral Research: Tools and Techniques* (5th ed.). Oxford University Press, Inc., USA.
- [56] Amanda Swearngin, Chenglong Wang, Alannah Oleson, James Fogarty, and Amy J. Ko. 2020. Scout: Rapid Exploration of Interface Layout Alternatives through High-Level Design Constraints. 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. doi:10.1145/3313831.3376593
- [57] Tvor, Inc. 2026. ShapesXR: Spatial design and prototyping platform. <https://www.shapesxr.com/>. Accessed: 2026-07-14.
- [58] Jean Vanderdonckt, Mathieu Zen, and Radu-Daniel Vatavu. 2019. AB4Web: An On-Line A/B Tester for Comparing User Interface Design Alternatives. *Proc. ACM Hum.-Comput. Interact.* 3, EICS, Article 18 (June 2019), 28 pages. doi:10.1145/3331160
- [59] Jo Vermeulen, Kris Luyten, Karin Coninx, Nicolai Marquardt, and Jon Bird. 2015. Proxemic Flow: Dynamic Peripheral Floor Visualizations for Revealing and Mediating Large Surface Interactions. In *Human-Computer Interaction – INTERACT 2015*, Julio Abascal, Simone Barbosa, Mirko Fetter, Tom Gross, Philippe Palanque, and Marco Winckler (Eds.). Springer International Publishing, Cham, 264–281.
- [60] Daniel Vogel and Ravin Balakrishnan. 2004. Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology* (Santa Fe, NM, USA) (UIST '04). Association for Computing Machinery, New York, NY, USA, 137–146. doi:10.1145/1029632.1029656
- [61] Jason Wu, Kashyap Todi, Joannes Chan, Brad A Myers, and Ben Lafreniere. 2024. FrameKit: A Tool for Authoring Adaptive UIs Using Keyframes. In *Proceedings of the 29th International Conference on Intelligent User Interfaces* (Greenville, SC, USA) (IUI '24). Association for Computing Machinery, New York, NY, USA, 660–674. doi:10.1145/3640543.3645176
- [62] Abby Wysopal, Vivian Ross, Joyce Passananti, Kangyou Yu, Brandon Huynh, and Tobias Höllerer. 2023. Level-of-Detail AR: Dynamically Adjusting Augmented Reality Level of Detail Based on Visual Angle. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. 63–71. doi:10.1109/VR55154.2023.00022
- [63] Zhijie Xia, Kyzyl Monteiro, Kevin Van, and Ryo Suzuki. 2023. RealityCanvas: Augmented Reality Sketching for Embedded and Responsive Scribble Animation Effects. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (San Francisco, CA, USA) (UIST '23). Association for Computing Machinery, New York, NY, USA, Article 115, 14 pages. doi:10.1145/3586183.3606716
- [64] Hui Ye and Hongbo Fu. 2022. ProGesAR: Mobile AR Prototyping for Proxemic and Gestural Interactions with Real-world IoT Enhanced Spaces. 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 130, 14 pages. doi:10.1145/3491102.3517689
- [65] Hui Ye, Jiaye Leng, Pengfei Xu, Karan Singh, and Hongbo Fu. 2024. ProInterAR: A Visual Programming Platform for Creating Immersive AR Interactions. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) (CHI '24). Association for Computing Machinery, New York, NY, USA, Article 610, 15 pages. doi:10.1145/3613904.3642527
- [66] Chenyang Zhang, Tianjian Wei, Haoyang Yang, Mar Gonzalez-Franco, Yalong Yang, and Eric J Gonzalez. 2026. Break the Window: Exploring Spatial Decomposition of Webpages in XR. In *Proceedings of the Extended Abstracts of the 2026*

CHI Conference on Human Factors in Computing Systems (CHI EA '26). Association for Computing Machinery, New York, NY, USA, Article 138, 6 pages. doi:10.1145/3772363.3798474

A Formative Study: Demographics

ID	Age	Gender	Background	XR familiarity (1 - 7)	First Time of use	Type of devices	XR Contents Experienced	UI Design Experience	XR Design Experience
P1	25	F	HCI/UX	5	2020	Quest	Space-related Research	4 ~ 5 years	XR research prototypes (1 year)
P2	24	F	HCI	2	2022	Quest	Zapworks & Friend's VR Project	3 ~ 4 years	None (HCI research only)
P3	24	F	Design	2	2017	Quest 2	Games & Entertainment	1 ~ 2 years	None (usage only)
P4	23	F	HCI & SW Development	2	2014	Vision Pro & Unknown iPhone & Quest	Game	1 ~ 2 years	None (usage only)
P5	24	M	UX/UI Design	3	2022	Quest	Game (Pokemon Go)	3 ~ 4 years	Mobile AR interfaces (2 years)
P6	24	F	Industrial Design	3	2023	Unknown	Game	2 ~ 3 years	None (usage only)
P7	24	F	HCI	2	2023	Quest	Experiment	2 ~ 3 years	None (research participant only)
P8	29	M	Industrial design	2	2022	Quest 2	Game	< 1 year	None (usage only)
P9	24	F	Industrial Design	4	2023	Quest	Sports Game	3 ~ 4 years	None (usage only)
P10	26	M	HCI & Industrial Design	2	2018	Quest	Experiment	1 ~ 2 years	VR research prototypes (1 year)
P11	26	M	Industrial Design & Experience Design	5	2023	Quest Pro	Game	4 ~ 5 years	None (usage only)
P12	25	M	Robot Interaction	4	2018	Google Cardboard	Game	< 1 year	None (usage only)

Table 5. Demographic and background information of all formative study participants.

B Formative Study: Interview

The following questions guided the semi-structured interviews:

- (1) Why did you choose to set the number of interfaces to be many/few or complex/simple?
- (2) Did spatial structures or contexts influence your UI design decisions regarding environmental factors?
 - (a) If the space were larger, more complex, or structured differently, would you have designed the interfaces differently?
- (3) What were your reasons for setting the interface size and composition at each viewing location?
 - (a) What was the most important design factor you considered?
- (4) You placed n interfaces in this space—if you were to redesign, would you change the number of interfaces?
- (5) Did you consider there to be an optimal window size, either in absolute physical dimensions or relative to the user's field of view?
- (6) How did you design the UI components (videos, images, text, buttons) to change across different viewing distances?
 - (a) What were your reasons for these component variations?
 - (b) Did you find certain component types to be particularly effective at specific distances?
- (7) For designs targeting the same zone from different viewing positions, why did you make the interfaces similar or different?
 - (a) : How would your interface design have changed if the viewing conditions differed (e.g., distances had been much closer or farther)?
- (8) In what aspects did the environment between user and interface most significantly influence your UI design decisions (size, position, content display method, component types, etc.)?
 - (a) If distance did not influence your design, what were the reasons?

- (9) What factors determined where interfaces would transition between different design states (distance changes, spatial context changes, etc.)?
- (10) What aspects of designing adaptive XR interfaces are most challenging?
- (11) For what types of users or contexts do you think adaptive interface design would be most beneficial?
- (12) Do you have any final thoughts about the design process or spatial interface design in general?

C User Study: Interview Protocol

We include here the full set of questions used in the user study:

Post-task walkthrough (baseline block)

- (1) Please briefly walk us through what you created in this workflow, including anything you expressed only as text notes.
- (2) Which situation is this version designed to respond to?
- (3) Looking at the artifact alone, how would one know when this version should appear?
- (4) What could you check directly in this workflow, and what did you still have to imagine or explain?

Post-task walkthrough (ADAPTEST block)

- (1) Can you walk us through your final design as a whole?
- (2) Choose one rule you consider important and explain it: why this target element, this environmental condition, and this change?
- (3) Did any part of the design change after you used the previews or Test Mode? What did you observe that led to the revision?
- (4) Did a suggested combined rule influence any design decision? If you accepted, edited, rejected, or ignored one, why?
- (5) Were there situations you noticed but chose not to address?

Semi-structured interview

- (1) You used both workflows today. What felt most different between working in Figma and working in ADAPTEST?
- (2) Which parts made it easier or harder to specify what should change and when it should change?
- (3) Compared with your usual design practice, what felt familiar and what felt unfamiliar in ADAPTEST?
- (4) Was there anything you wanted to express in ADAPTEST but found difficult to express?
- (5) In what projects or scenarios could you imagine using a tool like this?
- (6) Conversely, in what situations would you prefer not to use it?
- (7) If you could change one thing about the tool before real use, what would you change? (Probe: at which point of the design–development process would it fit?)

Received 25 July 2025; revised 24 September 2026; accepted 7 October 2026