

Rapido: Prototyping Interactive AR Experiences through Programming by Demonstration

Germán Leiva
Jens Emil Grønbæk
Clemens Klokmoose

leiva@cavi.au.dk
jensemil@cs.au.dk
clemens@cs.au.dk

Aarhus University, Denmark

Cuong Nguyen
Rubaiat Habib Kazi

Paul Asente
cunguyen@adobe.com
rhabib@adobe.com
asente@adobe.com
Adobe Research, USA

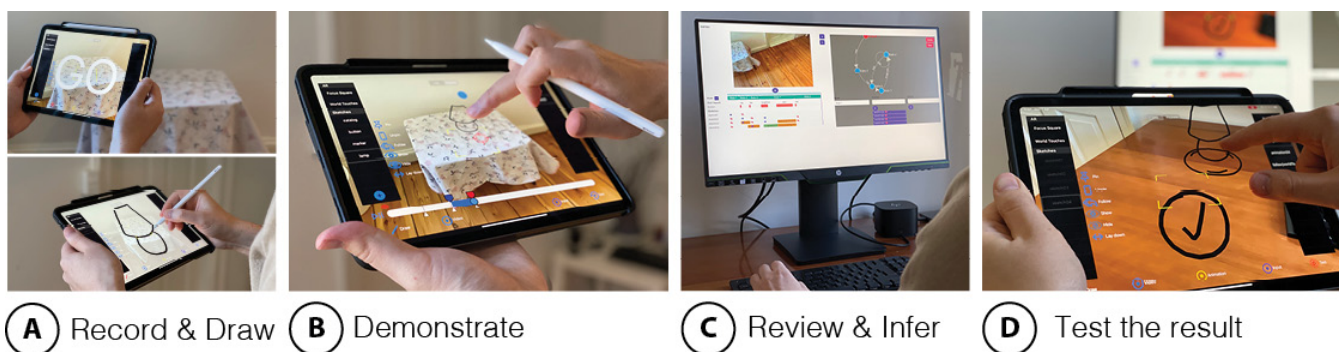


Figure 1: RAPIDO has an *AR interface* with live and playback modes, as well as an *overview interface* with a timeline and a state machine diagram. **A:** In the *AR interface*, designers record a video scenario and sketch content. **B:** They next demonstrate inputs, animations, and rules. **C:** In the *overview interface*, all the demonstrations are saved in the timeline, at their corresponding demonstrated time, to create a video prototype. The designer reviews the timeline and lets RAPIDO infer an initial state machine from it. **D:** The state machine drives an interactive prototype that the designer can test in the *AR interface*.

ABSTRACT

Programming by Demonstration (PbD) is a well-known technique that allows non-programmers to describe interactivity by performing examples of the expected behavior, but it has not been extensively explored for AR. We present RAPIDO, a novel early-stage prototyping tool to create fully interactive mobile AR prototypes from non-interactive video prototypes using PbD. In RAPIDO, designers use a mobile AR device to record a video prototype to capture context, sketch assets, and demonstrate interactions. They can demonstrate touch inputs, animation paths, and rules to, e.g., have a sketch follow the focus area of the device or the user's world-space touches. Simultaneously, a live website visualizes an editable overview of all the demonstrated examples and infers a state machine of the user flow. Our key contribution is a method that enables designers to turn a video prototype into an executable

state machine through PbD. The designer switches between these representations to interactively refine the final interactive prototype. We illustrate the power of RAPIDO's approach by prototyping the main interactions of three popular AR mobile applications.

CCS CONCEPTS

• **Human-centered computing** → **Mixed / augmented reality**; *Interface design prototyping*; *Systems and tools for interaction design*; *User interface programming*.

KEYWORDS

rapid prototyping, design by enactment, programming with examples

ACM Reference Format:

Germán Leiva, Jens Emil Grønbæk, Clemens Klokmoose, 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 (UIST '21)*, October 10–14, 2021, Virtual Event, USA. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3472749.3474774>

1 INTRODUCTION

Augmented Reality (AR) applications are becoming widely available due to the support provided by major mobile operating systems. For example, there were 13 million downloads of AR apps within

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

UIST '21, October 10–14, 2021, Virtual Event, USA

© 2021 Association for Computing Machinery.

ACM ISBN 978-1-4503-8635-7/21/10...\$15.00

<https://doi.org/10.1145/3472749.3474774>

the first six months of the release of the ARKit development platform [43]. Almost half of these downloads were games, but other popular categories included utilities, such as measuring tapes; entertainment, such as camera filters; lifestyle apps, such as furniture placement; and education, such as celestial body visualizers [48]. While apps that exploit new input modalities, such as face, hand, and body tracking, are emerging, the vast majority of mobile AR apps are based on common touch gestures for mobile devices [18].

There is a need for authoring tools that do not rely on code [3, 11, 23, 38] to allow non-programmers to prototype new AR ideas. In a recent study [3], AR/VR creators ranged from hobbyists to domain experts to professional designers. Most professional designers struggled with current tools and felt they required “too much coding” [3]. Ashtari et al. [3] identified eight barriers in authoring AR/VR applications, including *difficulty knowing where to start*, *designing for the physical aspect of immersive experiences*, *planning and simulating motion*, and *designing story-driven immersive experiences*. Among other considerations, they suggest supporting early-to-middle-stage AR prototyping, personalizing AR authoring tools based on expertise, and building integrated debugging and testing facilities [3]. A recent study on professional AR/VR creators [23] presented five implications for design; this work will focus on four of them: *using simple tools developed based on tasks and goals*; *drawing from existing methods, approaches, and workarounds*; *creating well-built artifacts for interdisciplinary communication*; and *making use of all three dimensions*.

Designers’ practices, such as storyboarding [39], sketching [14], animating [50], and video editing [29], contain rich but implicit descriptions of the expected behaviors relevant for the creation of interactive AR prototypes. The designers’ practice of acting out the design with their own bodies in the real world [22], or with the help of props while using the Wizard-of-Oz (WOz) technique [20], can be an alternative medium to the ubiquitous code representations.

Some commercial AR tools offer intuitive direct manipulation interfaces to author simple mobile interactions. For instance, Torch [46] lets non-programmers create mobile AR prototypes with common discrete inputs, such as tapping on the screen. On the other end of the spectrum, the symbolic representations (code, node diagrams) of visual/textual programming tools (e.g., Spark Studio [9], Unity [47], RealityComposer [2]) afford the flexibility and power required for more diverse AR interactions. Yet, these programming tools are complex, indirect, and often challenging for non-technical designers wanting to explore ideas. Furthermore, laborious programming and crafting detailed 3D assets is at odds with rapid prototyping [5, 6]. Krauß et al. [23] reported that some participants “had trouble in getting rid of artifacts which were no longer needed, because they had to put a lot of effort in constructing them”.

In the context of AR prototyping, our goal is to leverage Programming by Demonstration (PbD) [34] to bridge the benefits of these two approaches—the direct manipulation, speed, and contextual nature of video prototyping [25], and an underlying symbolic representation (i.e., state machine) to construct flexible and diverse interactive AR prototypes. Video prototyping, state machines, and PbD are known techniques; however, transitioning from a video prototype to a computational representation requires technical skills. Our target user group are interaction designers familiar with touch gestures and 2D prototyping tools (paper, video editing) that are in

the early exploration of an interactive AR experience—in the design phases before development. We ask, how can PbD enable designers to use their video prototyping efforts to create an interactive prototype without coding? Our key contribution is a new workflow to turn a video prototype into an executable state machine through PbD. We implemented this workflow in a prototyping system called RAPIDO. Its goal is to explore this new authoring workflow for early-stage prototyping of existing and popular AR mobile experiences.

To clarify how RAPIDO’s specific features fit into the context of existing work, we are postponing a detailed discussion of related work to Section 7.

2 MOTIVATION

One of the most popular types of AR lifestyle apps is helping users shop for furniture by letting them place items in their own rooms [43]. We selected prototyping this AR experience as our motivating scenario and ask the question, “How can a designer create an interactive prototype of an app like this without using conventional programming?”

Today, designers use different technologies to create early prototypes. The most popular early-stage design mediums are art supplies like pen, paper and scissors, and software tools like graphic editing software and presentation software [7]. Designers can quickly sketch interaction ideas with art supplies. Software tools are more time consuming but they provide higher visual precision [5] and support dynamic interactions. Graphic editing software can provide a refined static visual representation of an interaction. Presentation software can illustrate certain level of interactivity by using simple animations as system actions and re-purposing slide transitions as user inputs [21]. However, none of these technologies are well-equipped to support AR interactions in a 3D world.

Can we provide a fast AR prototyping process that is as familiar as sketching on paper and that also supports interactivity? Our goal is to provide a workflow for rapid prototyping of AR interactions using techniques and methods already familiar to designers. Video prototyping [32] is “the process of videotaping the use of physical prototyping material (paper, transparencies, Post-It notes) when acting out an interaction idea as part of a design process” [15]. Video prototyping [30, 31, 49] uses non-technical methods like paper prototyping [41] and the WOz technique [13]. While video prototyping primarily focuses on physical materials, we use a broader definition that includes digital materials, such as in Virtual Video Prototyping [15] or in Montage [25]. These tools are well-aligned with our goal of supporting rapid prototyping interactions and have been adapted for AR [25, 27]. However, the output of these tools is a non-interactive video, while our goal is an interactive prototype.

We want to combine a fast workflow with an interactive result by augmenting the video prototyping process with a designer-friendly computational representation of the interaction. Our goal is to allow designers to start by creating a video prototype of their AR experience, and, instead of discarding the effort of creating it, use it to transition to a fully interactive prototype [5]. We want to support a process similar to the WOz technique. However, as the designer demonstrates system behaviors by playing the role of the computer, we capture the demonstrations in a state machine via PbD, and use this state machine to drive a live, interactive prototype.

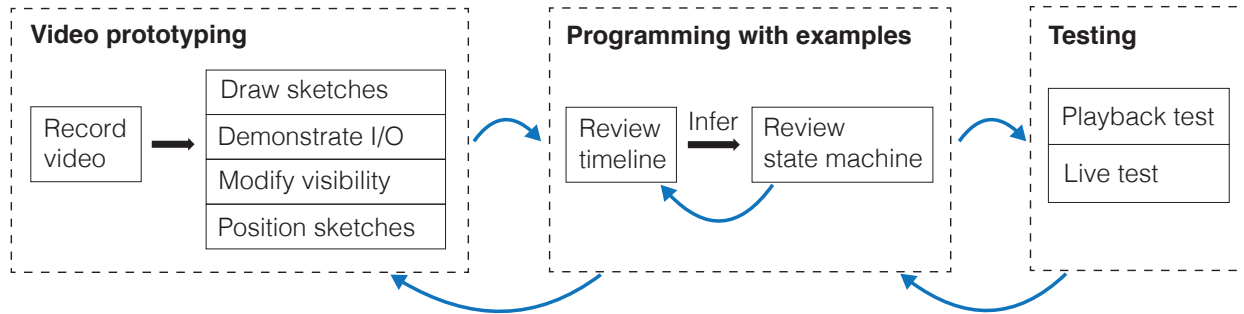


Figure 2: The iterative workflow with RAPIDO. Blue arrows indicate how the designer can iterate and refine the design.

We propose an iterative workflow with three phases (Figure 2): video prototyping, programming with examples [19], and testing.

During the *video prototyping* phase, the designer will:

- **record** a video background along with AR information,
- **draw** sketches to represent content and UI elements, and
- **demonstrate** user inputs and the prototype’s actions.

During the *programming with examples* phase, the designer will:

- **define** the states of the user experience,
- **define** the transitions between those states, and
- **associate** demonstrated actions with states and transitions.

During the *testing* phase, the designer can assess the prototype by interacting with the environment in real time or automatically replaying the demonstrated inputs over the video.

Designers are already used to the concept of states and transitions from their current practices. Tools like Figma [10] and In-Vision [17] let designers create simple interactive click-through prototypes. Designers define meaningful visual states (generally represented by the screen of the website or app), hot-spot areas (generally on buttons or scrolling areas), and triggers associated with user inputs to generate a transition between states (generally a click or a tap).

How does this workflow align with the design implications described by Ashtari et al. [3] and Krauß et al. [23]? Video prototyping is an early-to-middle-stage prototyping technique suitable for designers with low technical expertise to quickly explore diverse design ideas. Designers are familiar with video editing concepts, and creating a single linear video scenario is a concrete task that leverages this familiarity. Inspired by existing paper prototyping practices, designers should be able to draw rough sketches starting from scratch. The *video prototyping* and the *programming with examples* phases need a shared vocabulary to ease the transition from one to the other. Designers can demonstrate examples in the real world, taking advantage of the 3D space when necessary. Finally, the integration among the three phases allows for in situ testing and debugging.

This workflow can accommodate different types of user inputs, such as voice, mid-air gestures, and gaze; actions, such as sound, haptic feedback, and visual changes; as well as multiple computational representations of the interaction supporting states and transitions, such as object-oriented high-level programming code, spreadsheets, and flow diagrams.

3 RAPIDO

RAPIDO is an AR prototyping tool based on the rapid prototyping workflow described in the previous section (Figure 2) and influenced by the design principles presented in Enact [26]: *Multiple viewpoints* to provide representations of the design at different levels of abstraction; *One source of truth* to provide mechanisms to keep the viewpoints in sync; *Reveal the invisible* to reify concepts that are generally unavailable for the designer; and *Design by enactment* to design through embodiment and demonstrations. It supports touch user inputs, actions and interactive behaviors represented in a state machine diagram. Actions apply to sketched assets and include things like appearing, pinning, and following an animation path. RAPIDO has two views, an AR interface and an overview interface (Figure 3). The AR interface let designers capture a video scenario, draw sketches, demonstrate individual interaction examples, and experience the final interactive prototype. The overview interface has a timeline and a state machine diagram. The timeline organizes all the sketches and demonstrated examples based on their starting and ending times, and has all the information necessary to create the video prototype. The state machine diagram organizes actions inside states and inputs inside transitions. The underlying state machine will define the interactive behavior of the prototype.

In RAPIDO, these two interfaces are on two different devices: an AR-capable tablet and a website displayed on a computer screen. The tablet is used to create the video prototype by capturing spatial information and drawing sketches. The website displays the

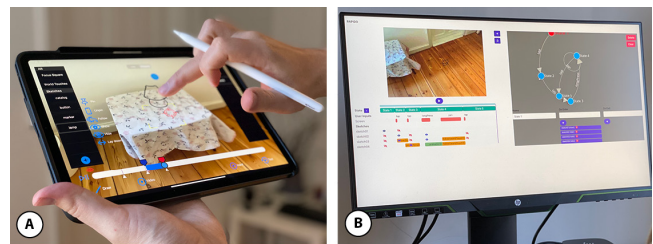


Figure 3: The RAPIDO system. A: The mobile AR interface. B: The overview interface, composed of a timeline (left) and state machine diagram (right).

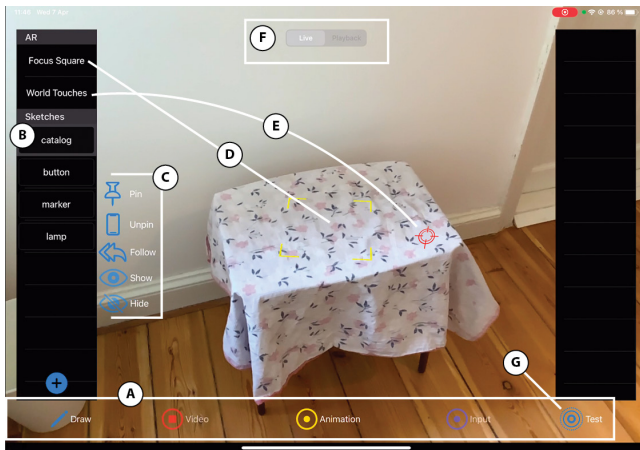


Figure 4: The AR interface in live mode. A: The five buttons. B: All sketches are in the list on the left. C: Available action buttons for the selected sketch. D: A focus square reifies a rectangular area of interest according to the current AR camera pose. E: A world touch reifies a point of interest in world-space according to a user's touch on the screen. F: Live/Playback toggle button at the top. G: Test button.

timeline and state machine diagram enabling the designer to turn the video prototype into an interactive AR prototype.

3.1 The AR interface

The AR interface (Figure 4) lets designers capture video, draw sketches, and demonstrate inputs and actions. At the bottom there are five buttons (Figure 4A) that let the designer draw, capture video, demonstrate animations, demonstrate user inputs and test. The Draw, Video and Test buttons are toggles that designers press to activate and press again when finished. The Animation and Input buttons are active for the duration of the user's next touch gesture and deactivate automatically when that touch finishes.

The AR interface focuses on working on one sketch at a time. All the sketches are in a list on the left (Figure 4B). The designer presses the plus button to create a new sketch and then starts drawing with the stylus. A sketch can be in screen space, representing a 2D U element (the default), or it can represent an object in the 3D world. When a sketch is selected, its name can be changed and potential action buttons appear next to it (Figure 4C): Pin, Unpin, Follow, Show, Hide, and Lay down. The designer can pin a sketch to a fixed location in the scene or make a sketch follow another object, such as a world-space touch or the device's focus. They can control the visibility of a sketch with the "Show" and "Hide" buttons. Finally, a sketch in the 3D world can be either perpendicular (the default) or parallel to its vertical or horizontal plane; the "Lay down" button toggles between them. A perpendicular sketch could represent a chair on the floor or a shelf sticking out from the wall. A parallel sketch could represent a carpet on the floor or a poster on the wall. To demonstrate an action for a selected sketch (see Section 3.1.2 for more details) the designer can tap on the corresponding action button or, in some cases, create a link with a stylus between the action button and an object in the 3D scene.

RAPIDO reifies [4] AR programming concepts typically unavailable for the designer, such as world-space touches and the device’s focus, into visual 3D objects in the scene that designers can directly manipulate [44]. The focus square represents the user’s area of interest in the scene (Figure 4D). Its position is calculated as the intersection of a ray cast orthogonally from the center of the screen and a plane detected by the AR engine; it lies on that plane and is depicted as a yellow square. A world-space touch, from now on simply *world touch*, is a point in world space corresponding to a screen-space touch (Figure 4E). Its position is calculated as the intersection of a ray cast orthogonally from the screen at the location of the touch and a plane detected by the AR engine; it is depicted as a red target on that plane.

3.1.1 Live Mode. This is the starting mode of the AR interface (Figure 4). The live camera feed is shown at the center of the device while the AR engine detects feature points and infers surfaces. Designers record the initial video from this mode and switch to playback mode (Figure 4F) to see the result. However, a designer can return to live mode to

- (1) move the camera freely to author outside the field of view of the camera's position at playback time, or
- (2) test the interactive prototype by pressing the test button (Figure 4G).

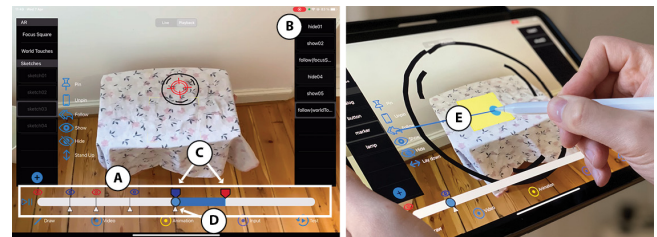


Figure 5: The AR interface in playback mode. A: The playback slider. B: The action list with the last item selected. C: Start time (blue handle) and end time (red handle) of the selected action. D: A white triangle showing when a user input was demonstrated. E: Linking to create a follow rule between the selected sketch (*marker*) and the focus square.

3.1.2 Playback Mode. This mode adds a playback slider (Figure 5A) to the interface to control the video playback. When an already-demonstrated action is selected on the right (Figure 5B), handles appear on the slider to change its beginning and end (Figure 5C). Also, the start time for each demonstrated input is depicted at the bottom of the slider with a triangle (Figure 5D). Adjusting the handles lets a designer synchronize a demonstrated action with a demonstrated input. In this mode, the designer can press the test button to replay the saved inputs over the recorded video. In this way the designer can observe the recorded inputs and assess the interactive behaviors of the prototype over the recorded scenario.

3.2 Actions: Animations and Rules

There are two types of actions: animations and rules. The list on the right shows all the actions that the designer has demonstrated for the selected sketch.

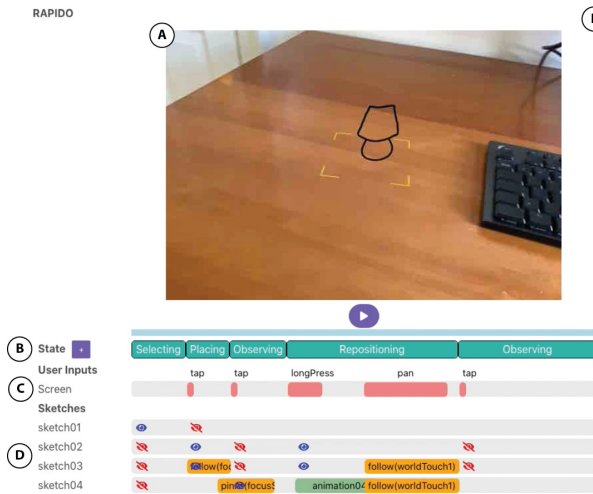


Figure 6: The overview interface: timeline. A: Video preview showing the recorded or the live view from the AR device. B: The different state placeholders of the video scenario. C: A row holding the demonstrated screen user inputs, i.e. touch gestures. D: One row per sketch, holding their corresponding demonstrated outputs. E: Buttons to reveal and to infer the state machine from the timeline information.

3.2.1 Animations. Animations are time-based keyframe changes in the transformation matrix for a sketch. To create one, the designer selects a sketch, presses the demonstrate Animation button, and creates a 2D animation path by dragging a finger across the screen. When the finger is released, the system creates a keyframe animation based on the touched screen positions. The begin time comes from the current position of the playback slider and the duration is inferred from the duration of the pan gesture. RAPIDO generates a default name that can be changed and adds the animation to the action list on the right.

3.2.2 Rules. A rule defines a value for some property of a sketch or how the value will be calculated. For example, it could define that the the visibility property is true or that the position property follows the focus square. To create a rule, the designer selects a sketch and then interacts with an action button from the list next to the sketches. Some actions have implicit values; for example the Show and Hide action buttons set the visibility property to true and false. The designer just needs to tap these. Other actions require more interaction. To make a sketch follow the focus of the user, the designer draws a link connecting the the Follow action button to the focus square (Figure 5E).

RAPIDO generates a name derived from the type of the action and adds it to the list of demonstrated actions on the right. Similar to an animation, the action’s begin time comes from the current position of the playback slider, but its default duration is to last until the next user input.

3.3 The Overview Interface

The overview interface is a live website that provides an overall visualization of the created sketches and demonstrations in a

timeline (Figure 6) and, organizes the interactive behaviors into a state machine diagram (Figure 7).

3.3.1 The Timeline. The timeline has a preview of the final video prototype at the top (Figure 6A), a playback slider in the center, and multiple rows at the bottom. It lets the designer refine the timing of the demonstrated actions in relationship with each other and the demonstrated inputs. The slider controls the current time of the video playback, going from 0 to 100 percent. The **State** row contains placeholders for the potential states of the user experience (Figure 6B). RAPIDO assumes that every demonstrated input creates a transition to a new state, e.g., four inputs separate the time into five placeholder states (Figure 6B). Each following row contains rectangles representing demonstrated inputs and actions. The position and width of each rectangle represent the starting time and duration. When the action is a show or hide action, it is depicted as an eye icon, open or crossed out, instead of as a rectangle. The **User inputs** row shows demonstrated inputs—touch gestures such as tap, pan, and pinch (Figure 6C). Each sketch gets a row in the **Sketches** section, and each row contains rectangles representing demonstrated actions (Figure 6D). For example, Figure 6 shows four sketches. Sketch04 has an animation colored in green and a follow(worldTouch1) rule colored in orange.

3.3.2 The State Machine Diagram. The state machine diagram represents states as circles and transitions as arrows in a directed graph (Figure 7). Designers can create new states with a double click and new transitions with a shift-drag. The designer presses the Infer button (Figure 6E) to make RAPIDO infer the initial state machine from the demonstrated inputs and the placeholder states created in the timeline. For example, Figure 8 shows six potential placeholder states based on five demonstrated inputs. However, if a user input

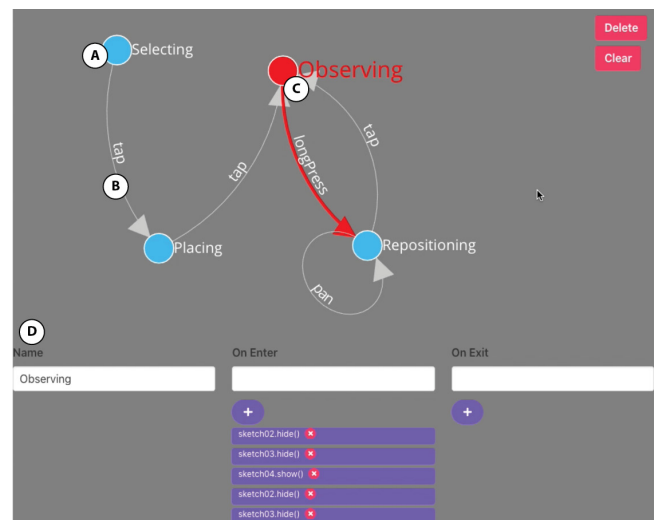


Figure 7: The overview interface: state machine diagram. A: A state inferred from the video prototype. B: A transition inferred from the video prototype. C: The state that is currently manually selected or active during playback. D: Properties of the selected/active state; its name, and animations/rules on enter or exit of the state.



Figure 8: Merging placeholder states in the timeline view. A: The placeholder states before merging. B: The designer drags a link from one placeholder to another. B: The two placeholder states are merged into one.

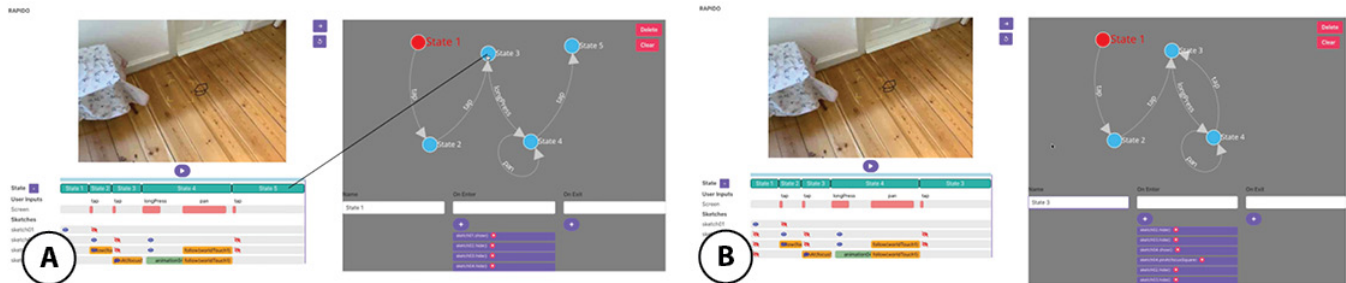


Figure 9: Indicating recurring states in the state machine view. A: The designer drags a link from the placeholder State 5 in the timeline to State 3 in the state machine. Before the link, there are five states. B: Once the link is applied, State 5 is deleted and all the transitions ending in State 5 will end in State 3.

should not create a new transition (see Section 4.4 for an example), designers can improve the inference by merging the placeholder states in the timeline (Figure 8A). Also, if a state is revisited in the timeline, i.e. there is no need to create a new state, the placeholder state can be linked with an existing state in the diagram (Figure 9).

The state machine defines the behavior of an interactive prototype, and the designer can run and test this prototype by pressing the Test button in the AR interface. The designer can test either in playback mode or in live mode. In playback mode, when the recorded video plays, RAPIDO synthetically recreates the demonstrated inputs at their recorded times. However, in this mode, the AR outputs are dynamically controlled by the state machine instead of coming from the video prototype. This lets the designer verify that changes made to the state machine are correct. In live mode, the state machine controls the current scene being captured by the AR device’s camera, and the designer must perform the inputs in this new context. This lets the designer verify that the prototyped experience generalizes correctly, and is a true live, interactive prototype. In either mode, when the prototype is running, the diagram highlights the currently active state or transition (Figure 7D). This help designers debug the prototype and detect the need for changes, e.g. merging placeholder states or modifying the timings of the demonstrated actions in the timeline.

We will explain how to create an interactive prototype with the RAPIDO system using our motivating scenario.

4 MOTIVATING SCENARIO: AN AR FURNITURE APP

A designer is prototyping an AR experience that lets users visualize how a new piece of furniture will look in their home.

First, the designer will capture a video scenario, draw sketches, and demonstrate inputs to create an initial video prototype. Second, the designer will create a state machine based on this video prototype that defines the behavior of the interactive experience. Third, the designer will switch back-and-forth between the video prototype and the state machine to test and refine the prototype.

4.1 Use scenario

In the experience the designer wants to prototype, the user selects an item from a catalog, positions it in the room, and then decides to move it somewhere else. From the point of view of the user, this seemingly simple AR interaction is not trivial and includes many detailed steps:

- (1) **seeing** a 2D view containing a *catalog* of furniture items
- (2) **tapping** an item of interest—a *lamp*—to select it and make the *catalog* disappear
- (3) **seeing** the camera live feed with a *marker* indicating the focused world surface, and a *check button* on the screen
- (4) **tapping** the *check button* once the *marker* is in the desired location
- (5) **seeing** the *lamp* appear at the *marker* location and the *marker* and the *check button* disappear
- (6) **long-pressing** the *lamp* to initiate repositioning
- (7) **seeing** the *check button* reappear, and the *lamp* floating up-and-down with the *marker* appearing underneath it
- (8) **dragging** the *lamp* to reposition it on another surface
- (9) **tapping** the *check button* to confirm the new location
- (10) **seeing** the *check button* and the *marker* disappear and, the *lamp* resting at the final position

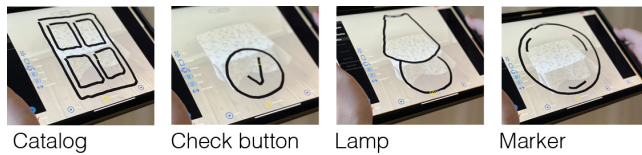


Figure 10: Sketches needed for the furniture app scenario

4.2 Creating a video prototype

The designer uses the AR interface to record a video scenario of the area the user is working in, draw sketches, demonstrate potential user inputs, and animate them.

4.2.1 Capturing a video scenario. With the AR interface in live mode, the designer presses the Video button to start a recording. After a 3-2-1-go countdown, RAPIDO starts capturing video frames along with spatial 3D information of the environment. The designer walks through the room holding the device, imagining the user selecting the desired furniture, focusing on an area to place it, and moving to other areas where the user might want to reposition it. When the designer presses the Video button again, the recording stops and RAPIDO switches to playback mode. In playback mode the designer can play, pause and scrub the recorded video with the playback slider.

4.2.2 Drawing sketches. The video scenario starts out empty, and it needs some content (Figure 10). The designer presses the plus button to create a new sketch. A new sketch appears on the left list with the default name *sketch01*. The designer draws a menu UI element with four options, representing the initial screen of the system (Step 1 above). To change the name of the sketch the designer simply scribbles a new name on the list, in this case *catalog*. Next they sketch two more user interface items: a check button to confirm furniture placement and a custom circular marker that will indicate a potential position for the furniture. Then, the designer sketches a lamp to be used as the selected furniture. Finally, the designer demonstrates a hide action at the beginning of the video for the *check button*, *marker*, and *lamp* sketches, so that only the *catalog* will be shown when the video starts.

4.2.3 Demonstrating user inputs. The designer demonstrates an input by navigating to the desired time in the video and demonstrating the touch gesture that the user is expected to perform at that time. To demonstrate the action of selecting a furniture item from the *catalog* by tapping (Step 2), the designer selects the *catalog* sketch from the left list, advances the video a couple of seconds, presses the Input button, and performs a tap gesture on the *catalog*. The demonstrated input over the *catalog* sketch is saved and depicted in the player’s slider with a white triangle.

4.2.4 Modifying visibility. The designer wants the *catalog* to disappear when the user taps it and wants the *check button* and the positioning *marker* to become visible (Step 3). To make the *button* and the *marker* sketch appear, the designer leaves the time slider at the tap time, selects the sketches and presses “Show” to create two *show* rules in the action list. Similarly, the designer selects the *catalog* and presses “Hide” to create a *hide* rule. The new rules take

place at the current time; *show* rules appear as a blue open eye and *hide* rules appear as a red crossed-out eye above the playback slider.

4.2.5 Positioning sketches in the world. The default position for a sketch is to be attached to the screen in the same position it was drawn, and not to respond to the content of the video. However, the designer wants the *marker* to be located at the position of the user’s focus area in 3D space. RAPIDO reifies the focus area in the form of a yellow flat square called the *focusSquare*. The designer selects the *marker* and drags a link between the “Follow” button and the *focusSquare*, indicating that the *marker* should follow the *focusSquare*. This creates a new *follow* rule in the right list and is depicted in the player’s slider with a blue bar with two handles, blue and red, representing the beginning and end of the time the rule applies. The default orientation of the *marker* is perpendicular to the *focusSquare* instead of being parallel to the floor, so the designer presses the “Lay down” button to fix this.

The prototype now has the *check button* attached to the screen (the default attachment) and the *marker* following the *focusSquare*. The designer next wants to demonstrate having a tap input on the *check button* place the *lamp* on the current focus position (Step 4). After advancing the time several seconds to a frame that shows a plausible location for the lamp, the designer presses the Input button, demonstrates a tap gesture on the *button*, hides the *button* and the *marker*, and shows the *lamp*. However, the *lamp* has its default position, attached to the screen. Instead it should be positioned at the current location of the *focusSquare* location, but unlike the *marker*, it should stay at the current location and not follow the square (Step 5). The designer selects the *lamp* and creates a link between the “Pin” button and the *focusSquare*. This creates a new *pinAt* rule in the right list and is also depicted in the playback slider with a blue bar.

4.2.6 Demonstrating an animation. Finally, the designer wants to prototype the user repositioning the *lamp* (Step 6). The envisioned experience has the lamp float up and down a few times to show that the user can move it. The designer advances the video and demonstrates a long-press input to initiate the action. Then, they create the floating animation by pressing the Animation button and drawing an animation path on the screen. As before, the designer makes the *check button* and the *marker* reappear (Step 7).

4.2.7 Using world touches. The designer wants to have the user drag the *lamp* and the *marker* with a pan gesture (Step 8). All previously demonstrated inputs were stationary, i.e. the camera position was not changing while tapping or long-pressing. However, in this case, the designer wants to demonstrate dragging combined with a change in the scene, as recorded in the video. To record an input while the video is playing instead of paused, the designer can hold the Input button for a second and then demonstrate a drag input while the video plays. After demonstrating the input, the designer needs to make the *lamp* and the *marker* follow the drag in world-space. Every screen-space touch in the demonstrated input has a corresponding *world touch*, represented as a red target over the nearest detected plane. The designer creates “follow” rules between the *lamp* and the *world touch*, and between the *marker* and the *world touch*, via linking.

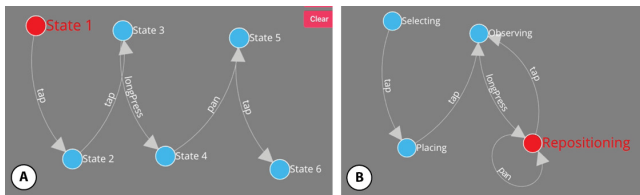


Figure 11: State modifications: the initial and final state machines for the furniture app prototype.

4.3 Finalizing the video prototype

As before the designer demonstrates a tap input (Step 9) and makes the *button* and *marker* disappear and the lamp take its final position (Step 10). The designer can play the video to assess that everything looks as expected.

4.4 Creating a state machine

The designer opens RAPIDO’s overview interface to create an initial state machine from the saved timeline information. The first row of the timeline shows six placeholder states, which is one more than the number of demonstrated inputs. If the designer presses the infer button, RAPIDO creates a naive state machine with six states and five transitions (Figure 11A). However, dragging the *lamp* should not change to another state; the user is still repositioning the furniture. To fix this, the designer merges states 4 and 5 so that the *pan* input returns to the same state. The designer renames the first four states to be *Selecting* the furniture, *Placing* the furniture, *Observing* the placed furniture, and *Repositioning* the furniture, and leaves the last one as *State 6*. Step 4 and Step 9 should transition the prototype to the same state—after *Repositioning*, the system should return to *Observing*. To adjust this, the designer creates a link between the last placeholder state in the timeline to the state named *Observing*. This leaves *State 6* with no incoming transition. Refreshing the inference makes *State 6* disappear, giving the expected result (Figure 11B).

4.5 Testing the prototype

The designer can test the prototype live in the AR interface. RAPIDO is a live environment; pressing the Test button in the AR interface starts the prototype. Typically the designer would first test in playback mode, to verify that the state machine modifications are correct, and then play the role of the end user in live mode in a different area or with different camera positioning.

Testing reveals a problem with the prototype: after repositioning the *lamp* and tapping the *check button*, the *lamp* returns to its initially-placed position instead of staying at the desired location. Inspecting the timeline and observing how the corresponding states and transitions highlight in the state machine help diagnose the problem. The designer discovers that the `pinAt(focusSquare)` rule happens within the state named *Observing*. However, it should happen when leaving the state named *Placing*. A state executes actions either when entering or exiting that state. To fix this, the designer can drag the `pinAt(focusSquare)` rule in the timeline to the left to be under the *Placing* state rather than *Observing*.



Figure 12: The planetary app. A: Drawing the sun. B: Pinning it at the window using the AR device’s current 3D location. C: Viewing the pinned sketch of the sun from further away.

5 ADDITIONAL SCENARIOS

We explained in detail how to prototype an interactive experience with our motivating example. There are many mobile AR applications that can be prototyped with RAPIDO’s building blocks. Other types of applications require additional functionality. RAPIDO’s actions, such as *pin* and *follow*, rely on the capacity of the AR engine to detect planes. We would like to use RAPIDO to prototype objects that are not attached to any plane. Also, the previously presented actions are applied to the entirety of the sketch. We would like to decompose a sketch and execute independent actions over the parts. We present two additional scenarios to illustrate how RAPIDO supports these situations.

5.1 Planetary App

A designer is trying to prototype an AR mobile application that displays information about celestial bodies, such as the sun, the moon, and Mars. The designer envisions the user looking at the sky with a mobile device, tapping on a region of the sky, and then seeing information related to the closest celestial body. Previously, we showed how to position objects by pinning to the `focusSquare` or by following a `worldTouch`. In a planetary app, the content’s location is not related with any detected plane. RAPIDO lets the designer use the position of the device’s camera in live mode to pin a sketch to it. First, the designer moves the device to the desired position. Second, with the desired sketch selected, the designer presses the “Pin” action button instead of dragging a link from the “Pin” button into the scene. RAPIDO pins the sketch at the physical location of the device’s camera. With this interaction, the sketch of the sun can be placed on top of the actual location of the sun within the recorded video. A tap input could reveal extra information, such as the name of the celestial body or the distance to planet Earth. The result can be seen in Figure 12.

5.2 Measuring Tape App

A designer is trying to prototype an AR interaction to measure the distance between two points. The user should tap once to set the initial point to the current focus location, move the device, and tap again to set the final point to the new focus location. While the user moves after the first tap between these locations, the output will be similar to “rubberbanding” in a drawing program, with a line connecting the initial point and the current focus location.

In RAPIDO, every sketch has a start point where the stylus first touched the screen and an end point where the final stroke ended. The designer can reveal these points by swiping right on top of a selected sketch. Once they are revealed, these points are depicted



Figure 13: The measuring app. A: The selected start point is pinned to the current focus. B: The selected end point will follow the focusSquare. C: Viewing the result.

as spheres that the designer can select by tapping. The points can be part of demonstrated actions in the same way as sketches. If the start and end points have been given locations through a *pin* or *follow* action, RAPIDO displays the sketch by stretching it to match the points' locations.

To prototype the measuring tape interaction, the designer sketches a line and needs to pin its start point to a fixed position and its end point to the current focus. To pin the start point the designer selects the start sphere and links the "Pin" action button to the focusSquare. To make the end point follow the focus, the designer selects the end sphere and links the "Follow" action button to the focusSquare. The result can be seen in Figure 13.

6 SYSTEM IMPLEMENTATION

RAPIDO consists of two user interfaces and a server. Its mobile AR interface is implemented on iOS 14 running on an iPad Pro 2 tablet with an Apple Pencil 2 as the stylus. The video recording uses ARKit¹ to save frames and spatial information. We use SceneKit² to render the 3D scene in live and playback mode. RAPIDO's Overview interface is implemented as a Web app using VueJS³ running on a MacBook Pro 2019 laptop computer with a 2.6Ghz processor and 32GB of memory. The tablet and the laptop communicate through a server implemented in NodeJS⁴ also running on the laptop. The components expose services with Bonjour⁵—a zero configuration networking protocol—allowing the creation of socket connections without manually specifying IP addresses. Once connected, the tablet and the laptop share the timeline information, such as the demonstrated examples or the current playback time, using a custom communication protocol build on top of SocketIO⁶. Also, the tablet continuously live streams the AR view to the laptop.

The overview interface uses a simple inference algorithm to create a state machine on demand, based on the timeline information. For each:

- (1) Placeholder State in the timeline, a new State is created.
- (2) Demonstrated input between two Placeholder States, a Transition is created between the involved States.
- (3) Demonstrated input timed within a Placeholder State, a new Transition is created to the same State.

¹<https://developer.apple.com/documentation/arkit/>

²<https://developer.apple.com/documentation/scenekit/>

³<https://vuejs.org/>

⁴<https://nodejs.org/>

⁵<https://developer.apple.com/bonjour/>

⁶<https://socket.io>

- (4) Demonstrated actions within a Placeholder State, an action will be added to the corresponding state's "On Enter" or "On Exit" list.

If a demonstrated action is timed within the first half of the state it will be assigned to "On Enter," otherwise to "On Exit." To help align the demonstrated actions and inputs precisely, RAPIDO snaps the beginning and end times to the transitions between Placeholder States. For example, when the designer drags a demonstrated action, its start time snaps to the start time of the Placeholder State unless the distance between them is greater than a threshold of 100ms.

The source code of RAPIDO is available at <https://github.com/germanleiva/rapido>.

7 RELATED WORK

There is extensive research proposing the use of concrete result-ing examples [28] or demonstrations of a procedure [8] to create computer programs without the need of coding. Myers et al. [34] categorize three types of demonstrational interfaces: no inference, simple rule-based inferencing, and sophisticated AI algorithms. RAPIDO has a demonstrational interface that uses simple rule-based inferencing. However, Kato et al. [19] warns that "PbE [Programming by Examples] systems without the capability of explicit programming are not suitable for user interface design." We build on top of their "programming with examples" approach [19]. RAPIDO provides means to manipulate the examples in the timeline and to have explicit programming on the state machine diagram. However, RAPIDO's approach is closer to the concept of "design by enactment" where the end-goal is designing and not necessarily creating a computer program [25].

While there are many research tools for prototyping interactive systems, we will focus on the most relevant for AR. Video prototyping systems (VPS) like Montage [25] and Pronto [27] support the design of AR interfaces. Montage is a general purpose VPS that overlays 2D sketches on top of a regular video scenario without any 3D information. Pronto was designed specifically for AR and captures 3D information that lets designers navigate a 2D video frame in 3D space, allowing the creation of spatial layers to draw sketches. However, both of these systems have a non-interactive video prototype as an output. RAPIDO builds on top of these approaches but introduces mechanisms to transition from these videos to fully interactive prototypes [5].

DART [45] pioneered many aspects of AR prototyping. It was build as a plugin on top of Director, a timeline-based design tool, but required programming with textual scripts for manipulating interactive behaviors. It provided the ability to replay captured information and sketch early assets. However, creating animations was time consuming and was done via the GUI rather than by demonstration, e.g., "ARTIST expressed a desire to identify physical locations by moving through the space" [11]. RAPIDO acknowledges that time-based representations are a friendly medium for designers but do not require the use of textual programming to create interactive behaviors.

There are AR prototyping tools specifically focused on early-stage design such as 360Proto [36], ProtoAR [37], PintAR [12], and XRDirector [35]. 360Proto relies in laborious paper backgrounds for creating 360 experiences while ProtoAR combines paper with

clay modeling. These representation are designer-friendly but the prototypes require live WOz to support interactivity. RAPIDO provides similar creation capabilities without needing WOz every time that the prototype is tested. PintAR [12] uses spatial sketches and video. However, video is used only as an output and not as a design medium, hindering the ease of iteration. XRDirector supports collaboration among multiple designers and uses demonstrations. However, the demonstrations are used for live performances rather than for capturing interactive behaviors. RAPIDO uses video and replay capabilities to better support iterations without the need to repeat the same demonstrations multiple times during design.

Newer AR tools, such as RealitySketch [45], enable real time bindings between sketches and physical objects for embedded visualizations. However, these bindings are defined by indirect manipulation requiring a structured naming of the potential variables. Saquib et al. [42] presents an interface to map predefined user gestures to graphical elements for live presentation purposes. However, unlike these works, RAPIDO enables prototyping interactive AR experiences with a state machine representation that support multiple actions going beyond single input-output mappings.

Commercial tools such as Reality Composer and Adobe Aero [1] share our target audience. However, they cannot prototype continuous inputs, such as panning, without external symbolic programming; they hide discrete trigger-action behaviors inside virtual objects without an overview; and every test of the prototype requires manual inputs without a playback testing feature. Finally, these commercial tools output only linear scenes (e.g. in glTF⁷ format) while RAPIDO supports non-linearity by revisiting states.

8 DISCUSSION

Olsen [40] presents several dimensions for evaluating user interface systems research. Previous work that follows this approach includes D-Macs [33], WatchConnect [16] and Astral [24]. We will organize the discussion using some of Olsen's evaluation dimensions.

8.1 Situations (S), Tasks (T) and Users (U)

We developed RAPIDO for interaction designers (U) to enable prototyping the interactions of a mobile AR experience without coding (T). It supports early-stage prototyping (S), transitioning from a video to an interactive state machine using a video timeline as an intermediate representation. RAPIDO uses a particular approach to defining states. The state machine organizes actions in states that represent the user flow (selecting/placing/observing/repositioning) rather than individual UI widgets' states such as button-pressed. Designers already work with visual states in the form of screen-shots, for example, on a user flow diagram or a wireframe. We expect RAPIDO's states to resemble these user-level visual states in existing design tools. However, the state machine diagram is a proof-of-concept, understanding to which extent it can be appropriated by designers would require a user evaluation. This could also inform future iterations in the design of the state machine diagram and its integration with the other components of RAPIDO.

⁷<https://www.khronos.org/glTF/>

8.2 Expressive leverage

RAPIDO provides four common actions: pin/unpin, follow, show/hide, and lay/stand. This limited set of actions already exhibits a "power in combination" that covers many AR mobile experiences. For example, when a single follow action is created, it encapsulates a keyframe animation in the video prototype and an instruction to make the target sketch mimic the transformation of the linked object when testing. Achieving the same functionality in Unity requires programming an event handler and expressing locations symbolically in a script.

Currently, RAPIDO does not support proximity or physics simulations but intermediate objects representing distances or force vectors could be added. A new rule called *applyForce* could receive a force vector as a parameter. Similarly to how designers demonstrate animations, a force vector could be extracted from a touch gesture on the screen or from the tablet's motion. For example, to prototype throwing a ball in AR, a force vector could be demonstrated with a touch gesture over the sketch of a ball. Next, RAPIDO would save the *applyForce* action and the designer could add it after a particular user input transition, such as a tap or a swipe on the ball. During playback, the ball will be animated to follow the path that the demonstrated force vector would generate if applied on an object of a default mass and gravity. During testing, a mechanism to dynamically generate the force vector in relation with the properties of the user input should be in place—using the same default mass and gravity. For example, the force vector's direction and magnitude should be calculated in runtime taking into consideration the direction and speed of the swipe gesture.

8.3 Expressive match

Sketches are drawn on top of a real scenario and user inputs are loaded by demonstration. This reduces the cognitive load of imagining where an asset will appear in the scene and navigating hierarchical menus looking for touch gestures' names—or realizing that a continuous gesture is not supported, requiring external coding. In RAPIDO, the designer performs a desired gesture and the system saves it at the current video time for later use. This user input can be reused during testing, not making the designer perform it manually. A pitfall is that creating user inputs becomes repetitive. This can be mitigated in the overview by copy-pasting timeline elements such as an input example.

9 LIMITATIONS AND FUTURE WORK

We presented how RAPIDO supports the prototyping of an interactive mobile AR experience from an initial video prototype. There are many design decisions in RAPIDO that are a consequence of the selected motivating example. We applied a bottom-up approach with multiple scenarios to keep the concepts of RAPIDO general enough to cover multiple mobile AR experiences. RAPIDO cannot currently prototype all types of mobile AR interactions, but its approach can be extended to support other types of prototyping and input modalities.

9.1 Additional inputs and actions

We cannot currently prototype a game where the user throws an object and causes an action to be triggered on a collision (e.g., as

when a poke ball hits a Pokemon in Pokemon Go). A state machine can represent this behavior by having transitions triggered on the detection of a collision. However, the current version of RAPIDO does not use physics engine events for transitions. Nonetheless, the timeline representation is flexible enough to accommodate these events as examples, alongside the touch gestures. Similarly, RAPIDO could be extended to allow transitions based on image or object detection and tracking, e.g., to change state when the face of a person is detected or touched.

9.2 Computational model

Interactivity in RAPIDO is modeled using a finite state machine. This limits the complexity of the generated prototypes because memory (e.g., counting or keeping track of points) has to be encoded as states and, hence, only a constant amount of information can be stored. Currently, it is not possible to prototype a shopping experience where the interface reacts differently to different types of products, or extracts prices of products from a database to accumulate a total price when furnishing a home. Going beyond what is expressible with a state machine will require specifying side effects like incrementing a counter on a transition and guarding transitions based on conditionals. Future work will assess the trade-off between expressivity and complexity that this would entail.

9.3 Reifications

RAPIDO currently reifies the focus area and the world touches. Other actions present in AR experiences can be also supported by reifying other concepts. For example, to support billboardage, i.e. having a sketch always face the camera, the camera could be reified and a “Look At” rule could be added. This new “Look At” rule could work between sketches too, allowing one sketch to always face another. When slight variations to existing rules are necessary, e.g., following an object but with certain offset, reifying the offset could be a way of letting designers still customize the prototype without programming. Finally, RAPIDO could support other input styles, such as face, hand or body tracking, by reifying the detection of body parts and allowing links to these joints.

9.4 Supporting other platforms

RAPIDO is currently constrained to the iOS platform and mobile AR. There was a need for a larger space to support the timeline, requiring the use of multiple devices. However, a similar prototyping system could be implemented on a single AR or VR head-mounted device. This would extend the working space without being constrained to the size of the tablet’s screen. However, a benefit of supporting web technologies is the ease of collaboration. We expect to extend RAPIDO to support multiple users, and a simple URL could be shared with other stakeholders to let them design or test with their own AR devices.

10 CONCLUSION

We presented RAPIDO, an early-stage prototyping tool for authoring interactive mobile AR experiences using Programming by Demonstration (PbD). We addressed the problem that prototyping interactivity for AR currently requires complex tooling and programming skills. With RAPIDO, we show how it is technically possible to create

interactive prototypes starting from video enactments and hand-drawn sketches that are gradually imbued with interactivity. Based on a combination of live direct manipulation on a handheld device and editing a timeline in a web interface, the designer can use RAPIDO to generate a state machine that describes the interactive behaviors of the prototype. The interactive prototype can be tested by running the state machine live in an AR device, such as a tablet. RAPIDO is a proof-of-principle that has been designed to realize examples of mobile AR applications that primarily rely on touch gestures, so the ceiling of expressivity is currently limited to this input modality. However, even with these limitations, we have shown that it is possible to create non-trivial interactive prototypes such as a furniture placement app. RAPIDO uses only a portion of the AR information that is available, and the rest of this information has great potential to enable creating richer and more expressive prototyping experiences that include proximity triggers and physics-based interactions. We hope RAPIDO will inspire other researchers to study the impact of this approach in AR creators’ design workflows, to extend it to other input modalities, and to apply it to other types of AR devices.

ACKNOWLEDGMENTS

This work was funded by Carlsbergfondet and Adobe Research. We thank the reviewers for their insightful comments and suggestions.

REFERENCES

- [1] Adobe. 2021. Aero. Retrieved 2021-04-01 from <https://www.adobe.com/products/aero.html>
- [2] Apple Inc. 2021. AR Tools - Augmented Reality - Apple Developer. Retrieved 2021-04-01 from <https://developer.apple.com/augmented-reality/tools/>
- [3] Naiges 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 (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. <https://doi.org/10.1145/3313831.3376722>
- [4] Michel Beaudouin-Lafon and Wendy E. Mackay. 2000. Reification, Polymorphism and Reuse: Three Principles for Designing Visual Interfaces. *Proceedings of the Working Conference on Advanced Visual Interfaces (2000)*, 102–109. <https://doi.org/10.1145/345513.345267>
- [5] Michel Beaudouin-Lafon and Wendy E. Mackay. 2003. Prototyping Tools and Techniques. In *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*. 1017–1039. <https://doi.org/10.1201/9781410615862>
- [6] Bill Buxton. 2007. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann. 448 pages. <https://doi.org/10.1016/B978-0-12-374037-3.X5043-3>
- [7] Adam S. Carter and Christopher D. Hundhausen. 2010. How is User Interface Prototyping Really Done in Practice? A Survey of User Interface Designers. In *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*. IEEE, 207–211. <https://doi.org/10.1109/VLHCC.2010.36>
- [8] Allen Cypher and Daniel C. Halbert. 1993. *Watch What I Do: Programming by Demonstration*. MIT Press. 652 pages. <http://acypher.com/wwid/WWIDToC.html>
- [9] Facebook Inc. 2021. Spark AR Studio - Create Augmented Reality Experiences. Retrieved 2021-04-01 from <https://sparkar.facebook.com/ar-studio/>
- [10] Figma Inc. 2016. Figma: the Collaborative Interface Design Tool. <https://www.figma.com/>
- [11] Maribeth Gandy and Blair MacIntyre. 2014. Designer’s Augmented Reality Toolkit, Ten Years Later. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology - UIST '14*. ACM Press, New York, New York, USA, 627–636. <https://doi.org/10.1145/2642918.2647369>
- [12] 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 (DIS '19 Companion)*. ACM, New York, NY, USA, 17–20. <https://doi.org/10.1145/3301019.3325158>
- [13] Paul Green and Lisa Wei-Haas. 1985. The Rapid Development of User Interfaces: Experience with the Wizard of OZ Method. *Proceedings of the Human*

- Factors Society Annual Meeting* 29, 5 (1985), 470–474. <https://doi.org/10.1177/154193128502900515>
- [14] Saul Greenberg, Carpendale Sheelagh, Marquardt Nicolai, and Buxton Bill. 2012. *Sketching User Experiences: The Workbook*. Morgan Kaufmann. 272 pages. <https://doi.org/10.1016/C2009-0-61147-8>
- [15] Kim Halskov and Rune Nielsen. 2006. Virtual Video Prototyping. *Human-Computer Interaction* 21, 2 (may 2006), 199–233. https://doi.org/10.1207/s15327051hci2102_2
- [16] Steven Houben and Nicolai Marquardt. 2015. *WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications*. Association for Computing Machinery, New York, NY, USA, 1247–1256. <https://doi.org/10.1145/2702123.2702215>
- [17] InVisionApp Inc. 2011. InVision | Digital Product Design, Workflow and Collaboration. <https://www.invisionapp.com/>
- [18] Jiyoung Jeon, Min Hong, Manhui Yi, Jiyoung Chun, Ji Sim Kim, and Yoo-Joo Choi. 2016. Interactive Authoring Tool for Mobile Augmented Reality Content. *JIPS* 12, 4 (2016), 612–630.
- [19] Jun Kato, Takeo Igarashi, and Masataka Goto. 2016. Programming with Examples to Develop Data-Intensive User Interfaces. *Computer* 49, 7 (jul 2016), 34–42. <https://doi.org/10.1109/MC.2016.217>
- [20] J. F. Kelley. 1983. An Empirical Methodology for Writing User-friendly Natural Language Computer Applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '83)*. ACM, New York, NY, USA, 193–196. <https://doi.org/10.1145/800045.801609>
- [21] Khella Productions Inc. 2013. Keynotopia. <https://keynotopia.com/>
- [22] Scott R Klemmer, Björn Hartmann, and Leila Takayama. 2006. How Bodies Matter: Five Themes for Interaction Design. In *Proceedings of the 6th Conference on Designing Interactive Systems (DIS '06)*. Association for Computing Machinery, New York, NY, USA, 140–149. <https://doi.org/10.1145/1142405.1142429>
- [23] Veronika Krauß, Alexander Boden, Leif Oppermann, René Reiners, Sankt Augustin, and Sankt Augustin. 2021. Current Practices, Challenges, and Design Implications for Collaborative AR/VR Application Development. *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (2021)*. <https://doi.org/10.1145/3411764.3445335>
- [24] David Ledo, Jo Vermeulen, Sheelagh Carpendale, Saul Greenberg, Lora Oehlberg, and Sebastian Boring. 2019. Astral: Prototyping Mobile and Smart Object Interactive Behaviours Using Familiar Applications. In *Proceedings of the 2019 on Designing Interactive Systems Conference (San Diego, CA, USA) (DIS '19)*. Association for Computing Machinery, New York, NY, USA, 711–724. <https://doi.org/10.1145/3322276.3322329>
- [25] Germán Leiva and Michel Beaudouin-Lafon. 2018. Montage: A Video Prototyping System to Reduce Re-Shooting and Increase Re-Usability. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology - UIST '18*. ACM Press, Berlin, Germany. <https://doi.org/10.1145/3242587.3242613>
- [26] Germán Leiva, Nolwenn Maudet, Wendy Mackay, and Michel Beaudouin-Lafon. 2019. Enact: Reducing Designer-Developer Breakdowns When Prototyping Custom Interactions. *ACM Trans. Comput.-Hum. Interact.* 26, 3 (may 2019), 19:1–19:48. <https://doi.org/10.1145/3310276>
- [27] Germán Leiva, Cuong Nguyen, Rubaiat Habib Kazi, and Paul Asente. 2020. Pronto: Rapid Augmented Reality Video Prototyping Using Sketches and Enaction. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. <https://doi.org/10.1145/3313831.3376160>
- [28] Henry Lieberman. 2001. *Your Wish is My Command: Programming by Example*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- [29] Danwei Tran Luciani and Peter Vistisen. 2017. Empowering Non-Designers Through Animation-based Sketching. 7 (2017). http://www.nordes.org/nordes2017/assets/full_papers/nordes17a-sub1006-cam-i26_LUCIANI_v2.pdf
- [30] Wendy E. Mackay. 1988. *Video Prototyping: A Technique for Developing Hypermedia Systems*. Vol. 5. ACM/SIGCHI.
- [31] Wendy E. Mackay and Anne-Laure Fayard. 1999. Video Brainstorming and Prototyping: Techniques for Participatory Design. *CHI'99 Extended Abstracts on Human Factors in ...* May (1999), 118–119. <https://doi.org/10.1145/632716.632790>
- [32] Wendy E. Mackay, Anne V. Ratzner, and Paul Janeczek. 2000. Video Artifacts for Design: Bridging the Gap Between Abstraction and Detail. In *DIS '00*. ACM, New York, New York, USA, 72–82. <https://doi.org/10.1145/347642.347666>
- [33] Jan Meskens, Kris Luyten, and Karin Coninx. 2010. *D-Macs: Building Multi-Device User Interfaces by Demonstrating, Sharing and Replaying Design Actions*. Association for Computing Machinery, New York, NY, USA, 129–138. <https://doi.org/10.1145/1866029.1866051>
- [34] Brad A. Myers, Richard G. McDaniel, and David Wolber. 2000. Programming by Example: Intelligence in Demonstrational Interfaces. *Commun. ACM* 43, 3 (mar 2000), 82–89. <https://doi.org/10.1145/330534.330545>
- [35] Michael Nebeling, Katy Lewis, Yu-Cheng Chang, Lihan Zhu, Michelle Chung, Piaoyang Wang, and Janet Nebeling. 2020. XRDiretor: A Role-Based Collaborative Immersive Authoring System. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–12. <https://doi.org/10.1145/3313831.3376637>
- [36] Michael Nebeling and Katy Madier. 2019. 360Proto: Making Interactive Virtual Reality and Augmented Reality Prototypes from Paper. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, 596:1–596:13. <https://doi.org/10.1145/3290605.3300826>
- [37] Michael Nebeling, Janet Nebeling, Ao Yu, and Rob Rumble. 2018. ProtoAR: Rapid Physical-Digital Prototyping of Mobile Augmented Reality Applications. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, 353:1–353:12. <https://doi.org/10.1145/3173574.3173927>
- [38] Michael Nebeling and Maximilian Speicher. 2018. The Trouble with Augmented Reality/Virtual Reality Authoring Tools. In *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. 333–337. <https://doi.org/10.1109/ISMAR-Adjunct.2018.00098>
- [39] Mark W. Newman and James A. Landay. 2000. Sitemaps, Storyboards, and Specifications: A Sketch of Web Site Design Practice. In *Proceedings of the Conference on Designing Interactive Systems Processes, Practices, Methods, and Techniques - DIS '00*. ACM Press, New York, New York, USA, 263–274. <http://dl.acm.org/citation.cfm?id=347642.347758>
- [40] Dan R. Olsen. 2007. Evaluating User Interface Systems Research. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (Newport, Rhode Island, USA) (UIST '07)*. Association for Computing Machinery, New York, NY, USA, 251–258. <https://doi.org/10.1145/1294211.1294256>
- [41] Marc Rettig. 1994. Prototyping for Tiny Fingers. *Commun. ACM* 37, 4 (apr 1994), 21–27. <https://doi.org/10.1145/175276.175288>
- [42] Nazmus Saquib, Rubaiat Habib Kazi, Li-Yi Wei, and Wilmot Li. 2019. Interactive Body-Driven Graphics for Augmented Video Performance. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, 622:1–622:12. <https://doi.org/10.1145/3290605.3300852>
- [43] SensorTower Inc. 2018. ARKit-only Apps Surpass 13 Million Downloads in First Six Months, Nearly Half from Games. Retrieved 2021-03-30 from <https://sensortower.com/blog/arkit-six-months>
- [44] Ben Shneiderman. 1983. Direct Manipulation: A Step Beyond Programming Languages. *Computer* 16, 8 (aug 1983), 57–69. <https://doi.org/10.1109/MC.1983.1654471>
- [45] Ryo Suzuki, Rubaiat Habib Kazi, Li-yi Wei, Stephen DiVerdi, Wilmot Li, and Daniel Leithinger. 2020. RealitySketch: Embedding Responsive Graphics and Visualizations in AR through Dynamic Sketching. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST '20)*. Association for Computing Machinery, New York, NY, USA, 166–181. <https://doi.org/10.1145/3379337.3415892>
- [46] Torch 3d Inc. 2020. Torch. Retrieved 2020-05-25 from <https://www.torch.app/>
- [47] Unity Technologies. 2021. Unity Real-Time Development Platform | 3D, 2D VR and AR Engine. Retrieved 2021-04-01 from <https://unity.com/>
- [48] Verizon Media. 2018. ARKit-only Apps Top 13 Million Installs, Nearly Half From Games | TechCrunch. Retrieved 2021-03-30 from <https://techcrunch.com/2018/03/28/arkit-only-apps-top-13-million-installs-nearly-half-are-games>
- [49] Laurie Vertelney. 1989. Using Video to Prototype User Interfaces. *ACM SIGCHI Bulletin* 21, 2 (oct 1989), 57–61. <https://doi.org/10.1145/70609.70615>
- [50] Peter Vistisen. 2016. *Sketching with Animation: Using Animation to Portray Fictional Realities—Aimed at Becoming Factual*. Aalborg Universitetsforlag.