ProtoAR: Rapid Physical-Digital Prototyping of Mobile Augmented Reality Applications

Michael Nebeling[1], **Janet Nebeling**[1], **Ao Yu**[1,2], **Rob Rumble**[1]
[1] University of Michigan School of Information [2] Tsinghua University Department of Automation
nebeling@umich.edu — http://protoar.com

ABSTRACT
The latest generations of smartphones with built-in AR capabilities enable a new class of mobile apps that merge digital and real-world content depending on a user's task, context, and preference. But even experienced mobile app designers face significant challenges: creating 2D/3D AR content remains difficult and time-consuming, and current mobile prototyping tools do not support AR views. There are separate tools for this; however, they require significant technical skill. This paper presents *ProtoAR* which supplements rapid physical prototyping using paper and Play-Doh with new mobile *cross-device multi-layer authoring* and *interactive capture tools* to generate mobile screens and AR overlays from paper sketches, and quasi-3D content from 360° captures of clay models. We describe how ProtoAR evolved over four design jams with students to enable interactive prototypes of mobile AR apps in less than 90 minutes, and discuss the advantages and insights ProtoAR can give designers.

Author Keywords
mobile augmented reality; physical-digital prototyping; quasi-3D 360° captures.

ACM Classification Keywords
H.5.2. Information interfaces and presentation: Input devices and strategies, Interaction styles.

INTRODUCTION
This work is driven by the vision of a future in which designers, not only experienced developers, will be able to create augmented reality (AR) interfaces. Previously, special hardware and extensive instrumentation of user and environment were required, but the latest technologies like Apple ARKit and Google ARCore enable AR on existing smartphones. This facilitates a new generation of mobile interfaces that are no longer limited to apps living in a purely digital form on the device. Instead, the device becomes a lens into the virtual world, making use of a user's physical environment to augment their view with digital content. For example, IKEA's Place app gives users previews of how new furniture fits into their homes before committing to the purchase and assembly.

CHI 2018, April 21–26, 2018, Montréal, QC, Canada.

ACM ISBN 978-1-4503-5620-6/18/04...$15.00.
https://doi.org/10.1145/3173574.3173927

This research considers the significance of these developments and the new need for more AR designers. The main problem is that interaction designers, who in today's world play a key role in user experience design, are not equipped with the toolbox to design tomorrow's AR interfaces. A significant part of the problem is that existing research has mostly focused on technical aspects of AR as a new technology [2, 3, 25, 40]. AR as a new medium for interaction designers has received much less attention [24]. These designers are guided by principles such as affordances, mappings, and constraints [30]. But in this new design world that mixes the physical and the digital, there are no established design rules and only a limited understanding of the kinds of interactions that users would find intuitive and natural [32]. We are not the first to recognize this need for AR tools to empower interaction designers, but other than DART [24] there are not many good examples. Even advanced commercial tools—InVision, Sketch, and Adobe XD—do not provide access to phone cameras as a basic requirement for mobile AR.

Existing research has focused on new digital tools like DART to create AR interfaces without programming. However, creating 2D/3D digital content for AR remains difficult and time-consuming [10]. When designing mobile apps, paper prototypes [33, 35] are typically the starting point, but this seems too limiting for AR [13]. To address this, we explore how working with modeling compounds like Play-Doh could complement paper prototyping to create props that could later be substituted with higher quality digital 3D content.

This paper presents the *ProtoAR* tool designed to investigate lightweight and creative ways to quickly transition from physical to digital prototyping of mobile AR apps. There are two key innovations in ProtoAR: *(i) cross-device multi-layer authoring tools* for live editing of mobile AR apps on phones; *(ii) interactive capture tools* to generate mobile screens or AR overlays from paper sketches, and 3D models from Play-Doh. We offer three main contributions with ProtoAR:

- the design and study of ProtoAR as a mobile AR live prototyping tool that extends cross-device authoring techniques [6, 11, 12, 27, 28, 29] to AR interface design;
- simple and fast techniques for digitizing paper mockups and Play-Doh models without the need for special hardware or expensive algorithms for 3D scanning [14];
- initial explorations of the design space of mobile AR apps that can be developed using a tool like ProtoAR and how it could expand to more advanced prototypes in the future.

BACKGROUND

There is already an extensive number of prototyping tools available to interaction designers, in particular, for mobile apps. However, across all tools—both physical and digital—existing support for AR design is limited and scattered.

It is common for interaction designers to start the process on paper, producing initial sketches, storyboards, and wireframes of potential interface screens [7, 26, 34]. It is easy for designers and users to explore and iterate on design ideas together because, unlike with digital tools, paper is guaranteed to be familiar to both [5]. Paper is also considered a useful tool for creating interactive prototypes that can be tested [33]. The designer can sketch the interface in terms of screens on different sheets of paper, swap them as the user moves to a different screen, or add overlays in the form of post-it notes or removable tape to them to simulate drop down menus and other changes in the screen [35].

Rather than replace paper with digital tools, studies with web designers, for example, have shown that a better solution is to provide integrated support for both [18, 22]. To address the paper-digital divide, researchers have explored how to make paper more interactive without affecting its qualities as a lightweight and flexible medium [1, 17, 16, 20, 21]. Lift-Off [15] is particularly motivating for ProtoAR, as it transforms 2D paper sketches into 3D models, albeit in VR, not AR.

Recently, Hunsucker et al. [13] conducted a preliminary study in which they experimented with non-digital ways for interaction design students to prototype AR interfaces. In their study, they found paper prototyping to be too limiting. Instead, they instrumented a classroom to simulate an interactive museum experience using physical posters and 3D printouts as stand-ins for superimposed digital 2D/3D objects. Interactive behavior was provided by means of Wizard of Oz [8, 9] where the designers moved objects around the room or added posters to the walls to facilitate user interactions. As an experience prototyping approach [4] it can provide rich data for designers, but significant preparation, training, and coordination of the study team are required [19] before it can become an effective method for prototyping AR interfaces.

For interaction designers of digital desktop, mobile, and web interfaces, there is a wide variety of digital prototyping tools. InVision, Sketch, and Adobe XD are just some of the popular choices today. These tools typically support designing for devices such as smartphones and tablets, have a concept of multiple screens, support defining active regions in these screens, and implement basic mouse and touch interactions to transition between screens. While they offer good support for prototyping mobile interfaces, they do little to help transition from paper to digital. ProtoAR bridges physical and digital prototyping, and demonstrates the delta over tools like InVision to support AR, specifically camera-based interactions.

At this stage, none of the existing prototyping tools interface with AR displays and marker tracking/environment sensing technologies such as ARToolKit, Tango, ARKit, and ARCore. Most AR devices, however, support integration with the Unity 3D game and application development platform. Unity provides an editor in which 3D scenes can be visually authored and assets such as 3D models, textures, and sounds can be managed. With Three.js, A-Frame, AR.js, and argon.js, there is growing support for creating AR/VR interfaces using web technologies; however, the vast majority are programming libraries targeted at developers rather than visual tools for designers. Moreover, with any of these solutions, new virtual 2D/3D content needs to be created using *external* tools such as TinkerCAD, SketchUp, and Blocks targeted at beginners, or more advanced tools such as Blender, 3ds Max, and Maya targeted at professionals. ProtoAR integrates support for creating AR views *and* 2D/3D content.

Significant programming and computer vision knowledge are usually required to specify interactive behavior. Platforms like A-Frame come with simple record-replay tools for designers to capture user interactions, especially with VR controllers, but the focus is on developer support for testing and debugging of interactive applications. In most cases, experimentation with alternative AR interaction designs would have to be done at the coding level, which will be very challenging and time-consuming for most interaction designers.

The closest tool to ProtoAR is DART [24]. It shares our goal of allowing designers to visually specify, rather than program, interfaces that blend the physical and virtual worlds. DART is a timeline-oriented AR scene editor that integrates with AR display and tracking technologies. Some of the support in DART goes beyond ProtoAR. For example, it provides integrated support for behaviors (e.g., translate, scale, rotate 2D/3D objects in 3D space) and cues (e.g., gestures and speech commands, camera movements in 3D space). DART also supports 3D animatic actors (informal, sketch-based content) and users can capture and replay synchronized video and sensor data. This allows designers to complete authoring tasks in other locations than the target environment. As there is initial support in ProtoAR, we plan to incorporate such features in the future. For now, however, ProtoAR adds the ability to generate 2D/3D digital content from physical prototypes, and live author the AR interface on connected devices. In recent years, a number of design tools have been proposed to support cross-device authoring of interfaces. For example, XDStudio [28] supports interactive cross-device design by using one device for authoring that simulates all target devices or by using on-device authoring on the target devices themselves. Weave [6], WatchConnect [12], and XDBrowser [27, 28] refined the techniques to support cross-device authoring on various mobile devices. However, none of these cross-device authoring tools have the support provided by ProtoAR for mobile AR apps. In particular, ProtoAR's support for composing multi-layered interfaces and live streaming them between mobile devices is novel. It may be known from running Google Instant Apps, but we extend it to live authoring.

INITIAL AR DESIGN JAMS

To better understand the requirements for ProtoAR to become an effective tool, we conducted a series of four design jams with interaction design masters' students at a university with one of the largest HCI programs in the nation. All design jams had the same goal of designing a mobile AR furniture

placement app similar to the one planned by IKEA. While we acknowledge that this is only one type of AR interface, experience taught us that having multiple student teams tackle the same design problem often leads to richer discussion and feedback among teams and organizers. Rather than trying to explore many different kinds of AR interfaces, our main idea was to vary fidelity and prototyping tools for the same kind of interface and so be able to study at different stages of design. This section describes the first two design jams in which students exclusively worked with non-digital tools. The remaining two will be described later as they were conducted once we had most of ProtoAR's digital tool support in place.

Method

For each round, we recruited eight students and formulated two teams each time (N=16). We started both design jams with a 30-minute demo including Q&A around an AR furniture placement app similar to IKEA's, which we had previously developed for Tango phones and HoloLens headsets. This way we wanted to make sure that all participants have had some exposure to AR interfaces. This also helped them to get a good sense of requirements as we introduced participants to two scenarios. In the first, they were asked to consider they had just moved from overseas to a new empty place and needed to buy all new furniture. In the second, they wanted to move in with friends into an already furnished place and needed to make sure they could move their old furniture. As all participants felt familiar with both scenarios, they understood that the two scenarios differed in requirements for measuring available space and previewing furniture placement in the room, as well as for capturing existing furniture digitally before physically moving it to the new place.

The main design activity was structured into three challenges:

1. **Sketching** mobile UI wireframes and user flows on paper;
2. **Modeling** increasingly complex furniture with Play-Doh;
3. **Walkthrough** of their demo making use of both media.

Each challenge was introduced and became the main design activity for about 30 minutes each. Finally, we had a demo and feedback session after which participants were asked to comment on the prototyping experience.

Results

Despite the short time, all teams successfully completed all three challenges and produced low-fi interactive AR prototypes using paper to construct phone stencils and screen mockups as well as Play-Doh to substitute for existing real furniture or virtual 3D models of new furniture (Figure 1).

Participants spent the vast majority of the time figuring out the main screens, user flow, and key interactions. They started with this challenge, but it continued to be the main challenge until the very end of the design activity. Identifying the main design components is a general challenge for mobile apps and not specific to AR. There are many tools for storyboarding and wireframing in existing prototyping software that we decided against attempting to replicate and match with ProtoAR, but that could be adopted later.

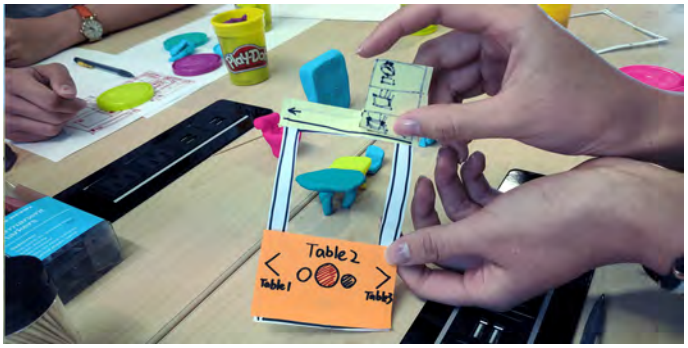


Figure 1. Low-fi prototype of furniture preview screen in mobile AR app

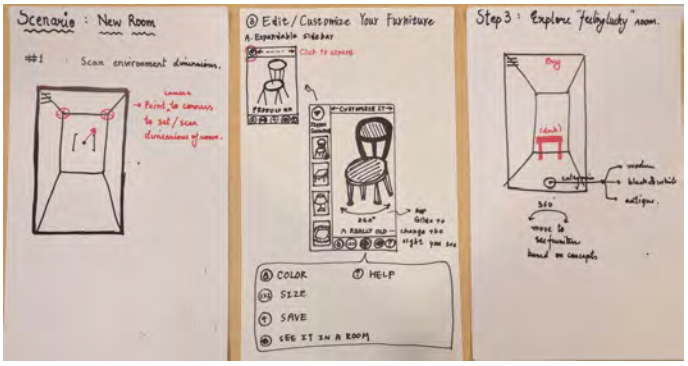


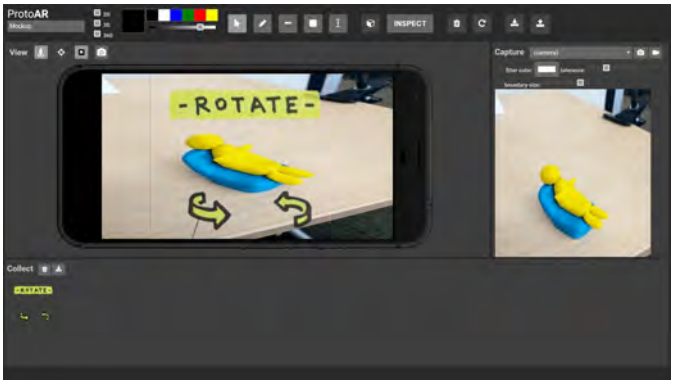
Figure 2. Three common screen mockups: 1) screen with 2D and 3D overlays for room measurement task, 2) screen with 2D menu and 360 preview of furniture, 3) screen with AR view of virtual furniture.

Figure 3. 20 selected Play-Doh models of furniture created by students

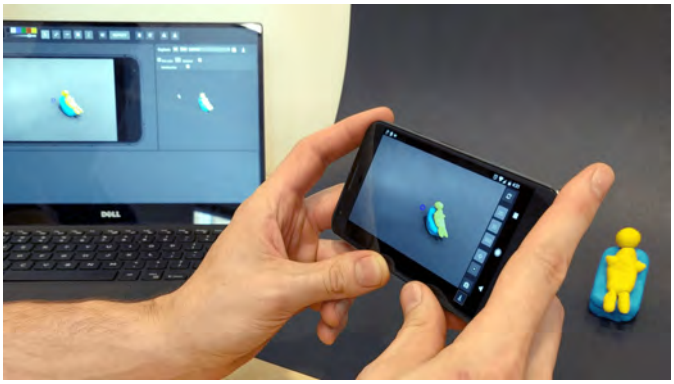
A key observation from participants' use of paper was that they paid little to no attention to detail of the room seen through the camera. Instead, they left areas where the room would be visible blank and only sketched the virtual content that would later become real-world overlays or objects anchored in the 3D room representation (Figure 2). this sometimes led to confusion of what virtual and real-world elements when looking at the mockups alone and required more explanation. Individual students made use of color to mark virtual 3D content, but there was no clear pattern that emerged.

Finally, it was interesting to observe how the second challenge of creating 3D models with Play-Doh quickly turned into a side activity while students continued to revise layout and design of traditional mobile interface elements. Over the two design jams, students produced an impressive amount of more than 50 Play-Doh furniture models (Figure 3). Despite

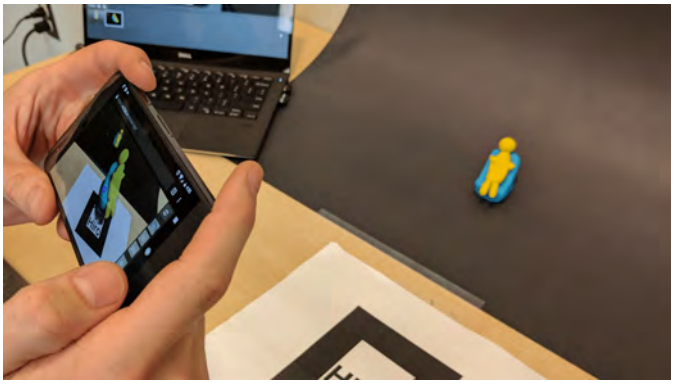
(a) Designer takes photo of UI mockup sketched on paper



(b) He creates a digital 2D overlay from captured paper sketch



(c) He creates a 360° capture of Play-Doh object



(d) He views 360° captured object in AR from similar angle

Figure 4. Example of a designer using ProtoAR's interactive capture tools to for physical-digital prototyping of a mobile AR furniture placement app.

some of them being highly complex, students often took less than 10 minutes to create them and became more creative with additional media, such as toothpicks and Post-it notes, to provide structural support and hold individual pieces together. We would believe that digitally modeling furniture would require a lot more attention.

In the feedback session, students commented highly positively on the AR prototyping experience with physical materials. In particular, flexibility and speed of prototyping with paper and Play-Doh were mentioned frequently. While not every student was equally proficient with clay modeling, everyone was able to produce at least one 3D model using Play-Doh, and most between three and five, whereas they had never used any 3D modeling tool before. Despite some limitations, we felt that we could get a good grasp of students' AR designs even from just their physical prototypes used in the demos.

REQUIREMENTS

From these early-stage AR student design jams, we deduced three main requirements for tools like ProtoAR:

- **Need for flexible non-digital tools and materials:** Paper and Play-Doh prove very flexible and sufficiently powerful materials to compose rather complex interfaces and demonstrate challenging interactions with AR content. They seemed indispensable for the early AR design stages.
- **Need for quick transition between physical and digital prototyping:** Since students demonstrated a lot of promise with paper and Play-Doh as physical AR prototyping tools, it was clear that we wanted to supplement, rather than replace, them with a digital prototyping tool like ProtoAR. The key challenge is to support quick transition between physical and digital tools to mix and match them.
- **Need to support implicit interaction with real-world objects:** As also found in prior work [24], a lot of the interaction with real-world objects happened to be implicit. Students moved the device around and positioned it relative to physical objects to trigger digital content. This requires image or object tracking where techniques have recently made a lot of progress. Explicit interaction involved touch and gesture on or around the phone. There are established techniques [23, 36, 37, 39] out of scope of this paper.

PROTOAR

Our vision for ProtoAR is to make it as flexible a digital prototyping tool for AR interfaces as paper and Play-Doh proved to be for physical prototyping. Through experimentation and student design jams, we have identified a small set of features that can be combined in flexible and powerful ways for physical-digital AR prototyping. Using a typical usage scenario illustrated in Figure 4, this section walks through the main features and explains how designers can use them.

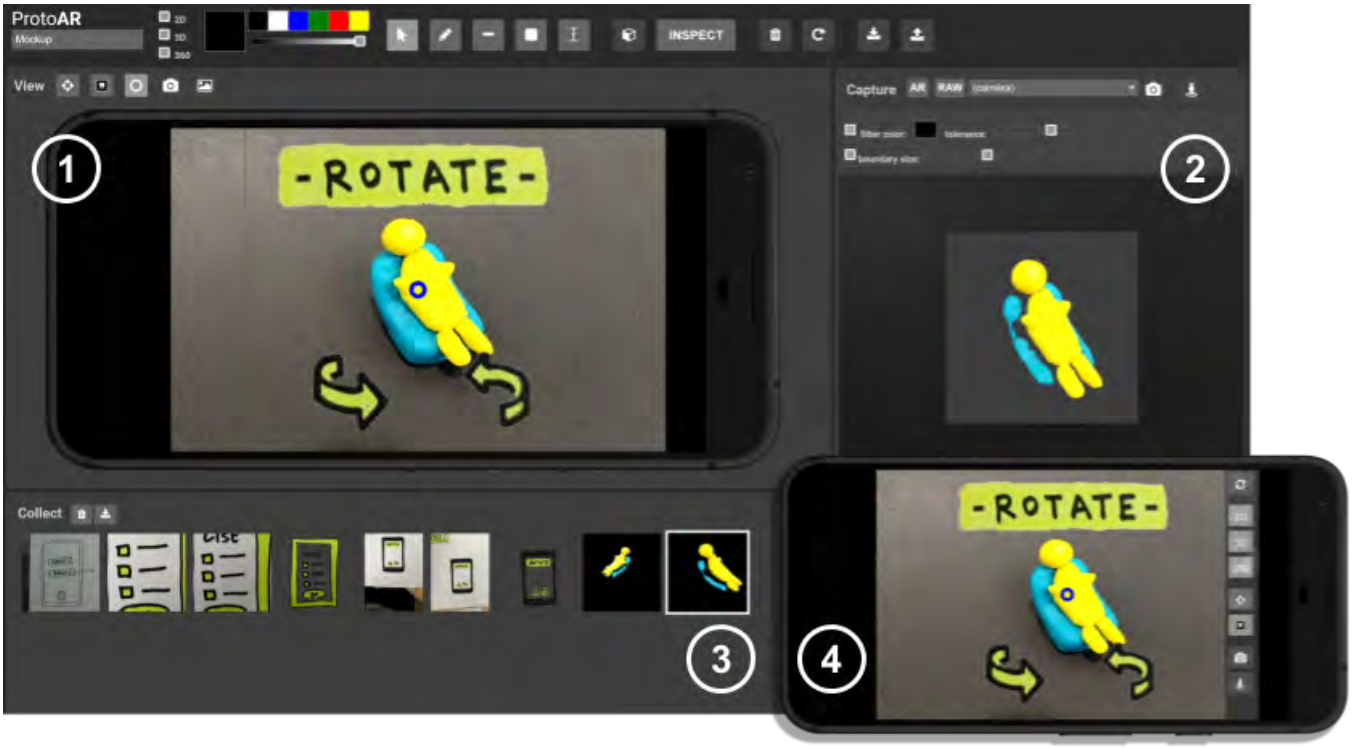


Figure 5. ProtoAR consists of two main interfaces: the *ProtoAR Editor* for laptop and the *ProtoAR App* for phone. The editor consists of three main components: *(1)* the *view* pane showing a live preview of the AR interface also shown on the phone, *(2)* the *capture* pane providing interactive capture tools to filter colors and crop the image, *(3)* the *collect* pane showing a collection of digitized 2D/3D objects that can be inserted into the view. *(4)* The app shows the AR interface streamed from the laptop and provides access to some editor functions.

Prototyping a Furniture Placement App in ProtoAR

Consider a designer who wants to prototype a mobile AR furniture placement app. Assume the designer has gone through physical prototypes similar to our initial design jams, and now wants to use ProtoAR for digital prototyping (Figure 4).

The designer marks areas of a paper screen mockup that he wants to capture with a highlighter. He then takes a photo of the mockup using ProtoAR on his phone (Figure 4(a)). In an editor running on his laptop, he can filter out the white background of the paper, leaving the highlighted areas as an overlay image. He then places the image on the rendered view of the AR app in the editor and resizes it to fit the view. As the laptop streams any updates to the phone, the designer can already get a preliminary sense of how the AR interface would look on his phone. For example, he can place a physical object in front of the phone's camera to view the object with the mockup superimposing the real-world view (Figure 4(b)).

Prototyping AR content to be registered in 3D may be accomplished through the use of AR markers. Using ProtoAR, the designer can create a 360° capture of the Play-Doh object against a monochromatic background (Figure 4(c)). The video is cut up into frames that correspond to different angles of the Play-Doh object. He can filter out the background as before and that way create a virtual quasi-3D object from the Play-Doh. He can place the object into the editor's 3D scene and position it relative to an AR marker. With marker tracking enabled, the app then superimposes the quasi-3D object on the marker, showing the video frame that corresponds to the camera's perspective on the marker. As the designer moves around the marker, he gets a sense of how the virtual object would look in the app from different angles (Figure 4(d)).

ProtoAR App

We refer to the ProtoAR interface running on the phone as "the app" (Figure 5.4). The app serves two main purposes. First, it shows a live view of the AR interface as it is authored in the editor. To obtain the augmented camera view, the app streams the phone's raw camera to the editor, the editor renders layers of 2D/3D digital content on top of that live video, and streams the augmented view back to the app. As we will explain in the implementation, significant engineering and fine-tuning were required to get this to work in real time on commodity smartphones. Second, it provides remote control of functions in the editor's view and capture panes. Inspired from cross-device authoring tools like XDStudio [29], this makes it possible to control the rendering of the AR interface and capture the phone's camera on either device without having to go back and forth between laptop and phone.

ProtoAR Editor

The heart of ProtoAR is the editor (Figure 5.1-5.3). It is a complex tool that consists of three main components to view, capture, and collect AR content. Here, we introduce each component and go into more detail in the following sections.

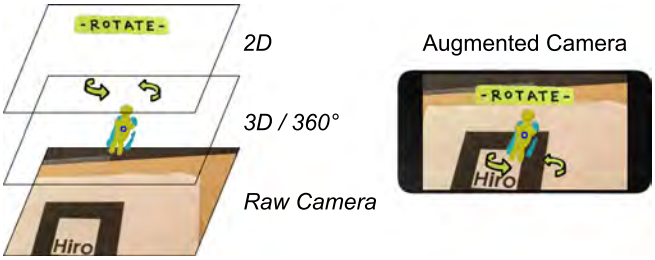


Figure 6. The ProtoAR editor renders three layers on top of the phone's raw camera stream and streams the augmented view back to the phone.

The *view* pane (Figure 5.1) shows a preview of the AR interface that is composed of three layers (Figure 6):

- a **2D layer** used for screen UI widgets that are opaque or for semi-transparent 2D overlays fixed on the screen;
- a **3D layer** for virtual 2D/3D objects that are anchored in a 3D scene and react to camera movement;
- a **360° layer** for virtual quasi-3D objects with perspective renderings of 360° captures from the camera direction.

These layers can be toggled depending on the designer's view, edit, and capture needs. The *capture* pane (Figure 5.2) provides tools for digitizing physical content such as paper sketches and Play-Doh models in view of a selected camera. The input is not limited to the phone's rear camera as in the example above—it can also be the laptop's user or environment-facing cameras, external cameras connected from additional mobile devices, or the augmented camera view as it is rendered itself. ProtoAR manages all connected cameras and collected content in a session and shares both with connected mobile devices via a unique channel name. Finally, the *collect* pane (Figure 5.3) is used to create a collection of captured 2D/3D content that can be inserted into the view via drag-and-drop. It also supports drag-and-drop of external files to import existing images, videos, and 3D objects into the editor. New captures can be created of the augmented camera view rendered in the view pane and of the content shown in the capture pane. The collect pane can also be used to load captures and existing captures may be re-captured to apply different settings. This gives a lot of flexibility and allows complex captures to be broken down into smaller tasks.

Interactive Capture Tools

A lot of the power and flexibility in ProtoAR come from its interactive capture tools. ProtoAR supports three types of captures: *photo snapshots*, *video sequences*, and *360° captures*. The first two can both be used to create 2D image overlays or 3D object textures with the difference that the former produces a still image while the latter provides an animated image. For example, in the first step in the example above, the designer created photo snapshots from paper sketches, changed capture settings to filter the paper, and added the 2D overlays to his content collection in ProtoAR (Figure 5.3).

360° captures are created from camera streams similar to video sequences, but they use a special capture mode that gives users up to 10 seconds to capture a physical object from various angles. Users can create them either by rotating the physical object in front of the camera (Figure 7(a)) or by moving the camera around the object in an orbit (Figure 7(b)).

(a) Designer rotates the Play-Doh in front of the camera

(b) Designer moves the camera around the Play-Doh in an orbit

Figure 7. Alternative 360° capture methods

Internally, 360° captures are sprites holding key frames captured at a certain interval over the duration of the capture (Figure 8). When inserted into the view, only the frame that matches the camera angle from which it was recorded is rendered. While they are internally captured in 2D, moving the camera renders a different perspective, which is why we refer to them as virtual quasi-3D objects. In the example above, the designer used this feature to capture the Play-Doh from a narrower angle given limited movement around the marker.

Figure 8. Example of 360° capture of a fairly complex Play-Doh model

Any capture can be edited to filter out colors, such as monochromatic backgrounds, and adjust the filter's tolerance to filter a smaller or wider range of color values (Figure 5.2). Additionally, the size of the capture can be adjusted with a boundary size slider. This can be done any time during live capture or post-hoc and also when re-capturing by replaying a video recording with capture enabled. In the example above, the designer first created a 360° capture using his phone and then actually re-captured in the editor on the laptop. He did this to tune the start and end view angles and filter out a slightly larger color range to remove artifacts from the shadow he cast while moving the camera around the object during capture.

Multi-Layer Authoring Tools

Motivated by some of the shortcomings of DART [10], ProtoAR comes with a set of editing tools for layering digital 2D/3D content over the live AR view (Figure 9).

The 2D layer can be composed of free-hand drawings, lines, boxes, and text, in addition to images and videos inserted from the capture and collect panes or imported from files. For each object, color, transparency, rotation, and z-index can be controlled. All 2D objects can be captured at once and added

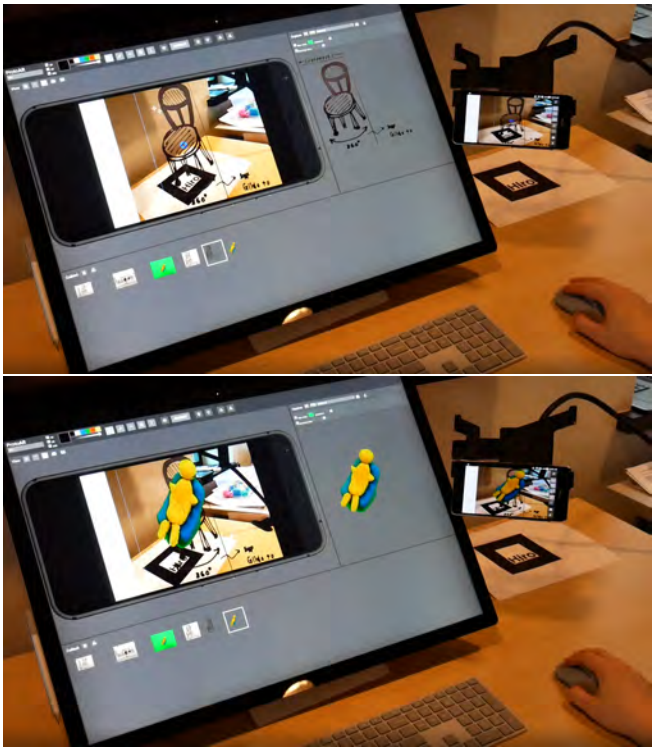


Figure 9. Live editing of the 2D and 3D layers in ProtoAR

to the collection. This is useful to compose arbitrarily complex 2D objects from multiple individual images.

The 3D layer can be composed of camera-facing planes or 3D boxes anchored in a virtual 3D scene. The user can rotate the 3D virtual camera using the mouse or touch to view the scene from different angles and can also move along 6DOF in the scene using WASD keys. The editor shows a blue ring that functions as a cursor to select 3D objects for editing. In addition to changing color and transparency, users can drop images and videos on the view to set the selected 3D object's texture. We believe that this way of authoring 3D scenes lends itself well to novice users with little experience with 3D software but with some first-person video gaming experience. For more experienced users, we integrated a visual inspector that provides a birds-eye view of the 3D scene with property editors to manipulate the 3D objects in the scene.

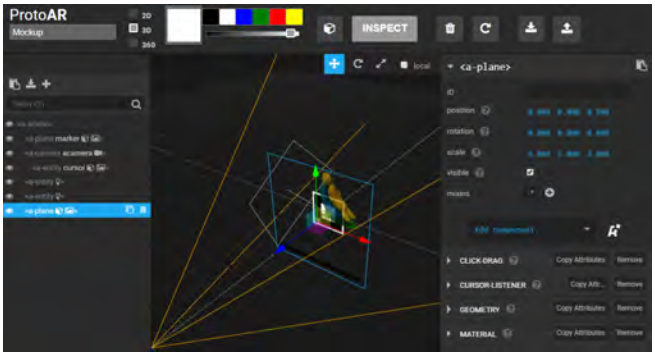


Figure 10. Birds-eye view of the 3D scene using the A-Frame inspector

The 360° layer is composed of virtual quasi-3D objects that react to camera movement with corresponding perspective renderings. Users can either manually change the camera perspective by manipulating the virtual camera of the 3D scene as described above, or activate marker tracking. As a first step, we added support for one ARToolKit-based marker, but multiple markers could be trained and supported. In the example above, the designer placed the marker on the table to view a virtual quasi-3D object that he had just captured from the Play-Doh model in the background (Figure 4(d)). Because there is no distinction between design-time and runtime in ProtoAR and marker tracking can be toggled at any time, markers can even be used when editing the scene to move virtual objects under the cursor by moving the marker in physical space or the phone's camera relative to the marker.

IMPLEMENTATION

ProtoAR is implemented using Fabric.js for the 2D layer, A-Frame and Three.js based on WebGL for the 3D/360° layers, HTML5 video and canvas with WebRTC for video streaming, as well as AR.js for marker tracking with WebRTC data channels for keeping the app and editor in sync (Figure 11).

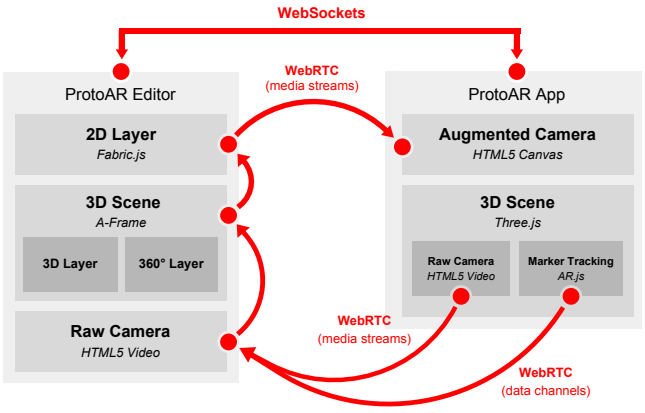


Figure 11. ProtoAR's architectural components, network communication, and AR rendering pipeline

The AR interface is rendered as follows. The app running on the phone streams the rear camera video without augmentations to the editor running on the desktop using WebRTC media streams. The editor takes this raw camera input and first renders all content contained in the 3D and 360 layers on the camera image before rendering the 2D content on top. This augmented view is streamed back to the app where it is shown instead of the raw camera. This happens in real time.

Note that simply streaming the AR content to the app and rendering it on the phone is not possible because HTML5 video and WebRTC media streams do not support the alpha channel required for transparency. Also, since HTML5 video is very resource heavy, most phones will not be able to play two video streams in real time. The solution we found was to hide the raw camera video once the streaming is active and render the incoming augmented camera stream on an HTML5 canvas rather than showing the receiving video element directly. That way both video streams can be processed in the background and the canvas can be updated in real time.

(a) Augmented view of furniture app in ProtoAR

(b) Number of steps required to create the AR interface above

Figure 12. Sample solution for the final design jam tasks

Both the editor and the app use a 3D scene, but the editor uses the A-Frame wrapper around Three.js and the app uses Three.js directly with AR.js to track markers. There is an A-Frame AR.js implementation, but it was too slow for our purposes. Using A-Frame in the editor is nice because it comes with a visual inspector giving a fullscreen birds-eye view of the 3D scene. If marker tracking is enabled, the app also sends updates of the marker position via WebRTC data channels. The editor takes the marker position and updates the virtual camera on the laptop. This triggers changes to how all content on the 3D/360° layer is rendered. The animate loop updates the perspective renderings of all 360° captures.

By default, 360° captures are sampled at 3 FPS. This achieved good results on Nexus 6P phones and laptops with i5 processor, 8 GB RAM, and onboard graphics card. More powerful devices can use higher sampling rates and smoothen the transition between perspective renderings.

FOLLOW-UP DESIGN JAMS WITH PROTOAR

We conducted two follow-up design jams with another group of interaction design students (N=16). We used the same design problem as in the initial design jams, but focused on digital prototyping in ProtoAR.

In the first of these design jams, we gave a demo of ProtoAR and conducted a systematic walkthrough with six students to learn how they would use the tools for digitizing some of the physical prototypes from earlier design jams. We chose this less hands-on format for two reasons. First, we wanted to get a sense of the amount of instruction required for students to understand the tools and keep the learning curve flat. Second, we wanted to see what kinds of strategies students might employ for digitizing physical prototypes and how they may choose to adapt the physical prototypes to make digitizing them in ProtoAR easier. We noticed two main challenges for students. First, as we also noted earlier, students had trouble discerning virtual from physical objects in the sketches.

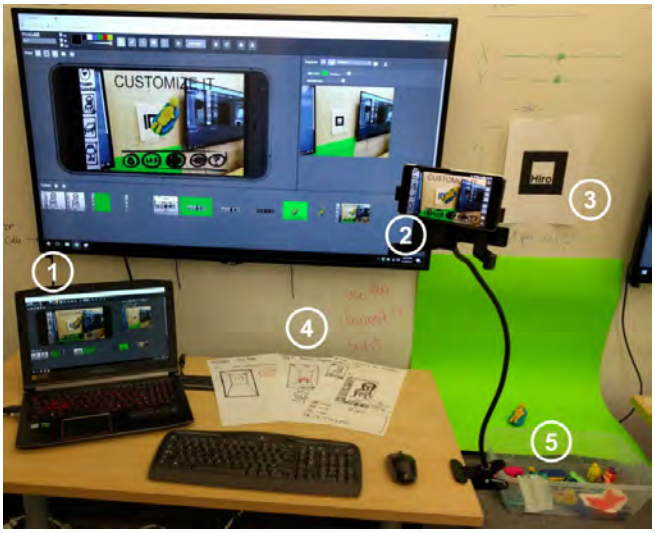


Figure 13. ProtoAR setup for final design jam: (1) laptop with the editor (mirrored on large screen), (2) phone with the app (mounted on a stand), (3) ARToolKit markers (on the wall), (4) selected screen mockups (from earlier design jams), (5) Play-Doh models (with green capture screen).

Second, they were not always sure whether the rear camera view would be visible or whether a screen was supposed to be opaque. We provided clarification based on our observations from the earlier design jams. Students then chose to color digital and physical objects differently and mark areas to be filled with the raw camera view using a highlighter because it could be easily filtered with ProtoAR's capture tools.

Using this input from students, we developed a sample solution for the final design jam (Figure 12). To create this solution, we first captured the menu on the left from the original paper mockup, and then chopped up the capture into smaller pieces. To remove captured areas we did not want, we added green boxes on the 2D layer, and re-captured the layer with a green filter to make them transparent. To capture the chair and keep important white areas opaque while making the rest transparent, we first highlighted those areas on paper, then captured the mockup, cropped the capture to the chair, and re-captured it with a white filter.

Method

For the final design jam, we recruited ten students and assembled two teams of five. We set up two desks with laptop and phone running ProtoAR, prepared a round standing table with a monochromatic poster board for 360° captures, and placed several identical markers in the room (cf. Figure 13).

We used the first 20 minutes to give a brief demo of ProtoAR's main features and introduce students to the AR furniture placement app design problem and the materials created in earlier design jams. This time we did not systematically structure the main design activity into challenges. Rather, we gave students 60 minutes to try out ProtoAR's tools to digitize and design 2D screens from paper sketches (cf. Figure 2) and Play-Doh models (cf. Figure 3) from earlier design jams. Students were free to choose physical 2D/3D objects they wanted to capture and were asked to show the results when ready or ask for help if they got stuck. Before wrapping

up with a 20-minute open discussion, we asked students to fill in a feedback form asking them to rate ProtoAR's interactive capture tools and live AR previews including 360° captures (Table 1), and name three strengths and three weaknesses.

Results

The majority of our students had a limited amount of experience designing mobile and augmented reality interfaces. Except for one student who had some experience using Rhino, none of the others had previously used 3D design tools. Still, using ProtoAR for less than an hour, they were able to create digital AR interfaces from earlier design jams' paper sketches and Play-Doh models similar to our sample solution. While students typically required more steps than in our sample solution and several capture attempts, they were able to complete most tasks themselves. The few times they asked for help, it was usually sufficient to provide additional instruction without taking over the ProtoAR editor or the app. Below we report the students' written feedback and comment on their use of ProtoAR's features and where they struggled the most, according to our observations and their notes.

Statements	Mean	Range
Enjoyed digitizing Play-Doh models	6.13	[5, 7]
Enjoyed digitizing paper mockups	6.11	[5, 7]
Easy to digitize paper mockups	6.11	[5, 7]
Got a good sense of final AR UI	5.56	[5, 7]
Fast to digitize paper mockups	5.44	[4, 7]
Fast to digitize Play-Doh models	4.38	[1, 6]
Easy to digitize Play-Doh models	4.25	[3, 5]

Table 1. 7-point Likert-scale ratings of 10 students after using ProtoAR for one hour to create digital AR interfaces from paper and Play-Doh

Overall, the students saw ProtoAR as a powerful, intuitive tool that allowed them to prototype interfaces quickly (Table 1). Students positively mentioned that ProtoAR only requires "a short orientation session." They found "a lot of options for customization in both 2D & 3D" and "got a good sense of how things [could] potentially fit in [the] environment." Speaking to its ease of use, they felt that ProtoAR allowed them to "digitalize both paper and 3D mockups in an intuitive way", and to "capture fairly complex objects and edit the files very quickly." Students considered it a "fast prototyping tool" for AR interfaces that appeared "very quick for rendering objects in perspective."

The students' experiences with ProtoAR were not without their pitfalls, however. Problems identified in the design jams fall into one of three categories:

- **Multi-layered authoring has a steep learning curve:** While students quickly understood and learned to master the 2D editing and capture tools, we definitely noticed a learning curve with multi-layered authoring. At first, it was difficult for students to understand the functionality of the different layers. While editing the 2D layer felt very familiar, most students were not used to editing in 3D. In particular, having to toggle layers to target 3D objects below the 2D layer when both were activated was often confusing. Also the fact that object insertion, selection, and movement works differently on the 3D/360° layers added to the learning curve. Our idea of using the cursor for choosing objects seemed to find more agreement over time, but initially it felt unfamiliar to them (at least for content authoring tasks). Whether our approach is better than using the A-Frame 3D scene inspector requires investigation.
- **Cross-device authoring is unfamiliar but powerful:** Students also seemed unfamiliar with interfaces that support cross-device interaction. Initially, they seemed inclined to just rely on one device (usually the laptop running the editor) and made only limited use of the phone app for capture. The fact that they could also control some of the editor's functions remotely seemed easy to forget and not anything students knew from other applications and therefore would expect. However, after reminding students, they started to appreciate the support and made increasing use of it, in particular, to quickly do 360° captures of Play-Doh objects on the phone, insert them into the 3D scene using the laptop, and preview the AR interface on the phone.
- **360° capture is simple in theory but hard in practice:** Students saw good potential in ProtoAR's support for 3D capture, but also that "the tools were not sufficient to create [actual 3D] objects." One negatively commented that it "requires a very particular method for capture and a steady hand" and another that it felt "a bit time consuming and requires precision to work well." We observed that students particularly struggled with smoothly moving the camera around Play-Doh objects for 360° capture while keeping the image centered. They seemed to achieve better results putting the Play-Doh on a toothpick and spinning it in front of the camera. However, this was only an option for smaller and less fragile Play-Doh objects. For some of the more complex objects they achieved the best results when using the green screen as a bottom layer, putting the object on it, and working with a partner who would rotate the pad. In the discussion, they also asked whether image stabilization techniques could be added in the future.

DISCUSSION

This section presents a discussion of related techniques, limitations, and extensions of ProtoAR.

3D Reconstruction/3D Scanning/Photogrammetry

We developed quasi-3D 360° capture because of the limitations of 3D reconstruction, 3D scanning, and photogrammetry. First, the resolution of 3D reconstruction techniques like MobileFusion [31] is not high enough to capture mesh and texture of tiny Play-Doh props. Second, 3D scanning is more precise (down to 1-2mm), but requires expensive (usually $1,000+) external hardware. Often multiple scanning rounds and technical knowledge are required, e.g., users need to *pre*-specify scan area and mesh density. Photogrammetry is considered superior because good texture often covers up bad geometry, but has a number of conditions. It works best with daylight and strong visual features—it does not work well on one-colored, smooth, featureless Play-Doh surfaces. It also needs consistent camera settings, while phones typically auto-correct focus and light. Further, the photogrammetry workflow is complex (sparse cloud → dense cloud → mesh → texture), and each step takes considerable time.

The three techniques lead to increasingly higher fidelity models, but with trade-offs. We would rank our technique before photogrammetry as it enables 360° capture of small props in a single pass in real-time. Even if other techniques are improving, ours with quasi-3D remains useful for rapid prototyping.

Limitations

Inherent to its design, there are two main limitations of our quasi-3D technique. First, 3D rendering is limited to those perspectives at which an object was originally captured. At this stage, we do not extrapolate from the images extracted from the video capture to increase the number of supported viewing angles. Second, image resolution is limited even though common 8-12MP smartphone cameras can obtain high-quality images of both paper and Play-Doh props. At the moment, the bottleneck is that HTML5 Media Capture and WebRTC Streaming are limited to Full HD 1080p (2.1MP) on mobile, though Chrome has started to support 4K (8.3MP) on desktop. Also, WebRTC Peer Connections support high-res low-compression media, but negotiate quality. On less powerful devices or with poor connectivity, streams may therefore downscale resolution and use more image compression.

While image scaling to reflect changes of the z-position is possible, the extreme case of walking up to room-scale Play-Doh captures can make tiny features like surface finish (toothpick engraved green sofa in Figure 3) appear more pronounced. HTML5 Canvas supports image smoothing in Chrome, which helps. Also, switching perspective renderings is more salient when upscaled, but higher FPS capture reduces jumps. Chrome limits canvas size to $32k^2$ px, but this is good enough for 30-sec 1080p video at about 18 FPS.

Future Extensions to ProtoAR

ProtoAR covers early-stage digital AR prototyping. Increasing fidelity is feasible given what is already in place. First, ProtoAR supports common 3D obj files so that quasi-3D models can be substituted with higher quality 3D content, either generated using the above techniques or modeled manually. Second, support for explicit interactions can easily be added to the 2D layer based on hotspots similar to InVision. For the 3D scene, click and gaze can be added using existing A-Frame components. As stated earlier, support for gestures is important, but is subject of our ongoing research.

ProtoAR is not only a versatile prototyping tool, but is also one that can be extended at various levels. To illustrate this, we describe three extensions to core components of ProtoAR that we have implemented: the first adapts the output by adapting the rendering pipeline, the second extends the input sources with minimal changes to the app, and the third leverages ProtoAR's interactive capture tools for live updates of 2D/3D objects from additional cameras.

- **Rendering Pipeline:** We experimented with two extensions that adapt the rendering pipeline of ProtoAR's editor. First, we added support for holographic displays such as the Spectre Hologram Projector[1] based on the classic illusion technique, Pepper's Ghost. This required rendering ProtoAR's layers four times with step-wise 90° rotation and mirroring of the left and right images before streaming it to the app. Second, we also added support for a hand-made display from plexiglass tilted at 45° against another screen. We process the augmented view and make all pixels transparent that do not change between two frames. This essentially removes static background and can be used to create holographic displays from live video.
- **User Interactions:** We also added support for recognizing users' mid-air gestures and tracking the device's absolute coordinates within the room using Kinect. This prototype overrides the marker tracking in the app to update the position when the marker cannot be visually tracked. When the "marker lost" event occurs, we update the position based on the user's spine coordinates in Kinect, and switch back to marker tracking as soon as a marker is detected. Using Kinect, we can extend the input vocabulary to gestures around the phone. This extends the idea of public ambient interactive displays [38] to AR interfaces.
- **Live Streaming:** We also added an experimental feature to ProtoAR's editor that makes it possible to link the live preview of the capture pane directly to digital 2D/3D objects inserted in the AR view. Using this feature, additional cameras can be used to stream live content updates to 2D/3D objects as they are rendered on the phone. This is an advanced feature that we developed that we believe could prove useful for prototyping adaptive interfaces as well as simulating interactive behaviors provided via Wizard of Oz [8, 9], but this requires further investigation.
- **HMD Support:** Finally, ProtoAR already works with modified Cardboards to view the phone's rear camera and Aryzon/HoloKit mobile phone wrappers for AR. Prototyping for dedicated AR displays like HoloLens requires more research. At the moment, HoloLens only works with Edge browser with poor support for WebRTC, but Microsoft is actively working on increasing the level of support.

[1] http://www.spectrehologram.com

CONCLUSION

This paper presents a new augmented reality (AR) tool, *ProtoAR*, specifically designed with interaction designers in mind. ProtoAR comes with a visual editor that composes AR interfaces from multiple layers with 2D/3D content and an app that provides a live preview of the AR interface in real-time. Over a series of four AR design jams with interaction design students, we learned about the strengths and weaknesses of both physical prototyping with paper and Play-Doh and digital prototyping based on these materials.

Future work should be dedicated to adding support for explicit user interaction using touch and gesture. As in previous work [24], we argued that a lot of the primary interaction in AR is implicit in that content needs to react to changes in the camera view. ProtoAR's live AR previews of all layers already make it possible to simulate interaction in a Wizard of Oz manner [8, 9], by manually moving 2D/3D objects around in the editor in response to the camera feed. This can be done in real-time with marker tracking enabled or disabled, and provides a useful basis for future research.

SUPPLEMENTARY MATERIAL

We have started to use ProtoAR in teaching and have developed instructional materials and example resources to guide students, which we are happy to share with the community. A working prototype of ProtoAR, example applications, and additional resources are available at **http://protoar.com**.

REFERENCES

1. Michelle Annett, Tovi Grossman, Daniel J. Wigdor, and George W. Fitzmaurice. 2015. MoveableMaker: Facilitating the Design, Generation, and Assembly of Moveable Papercraft. In *Proc. UIST*. 565–574. DOI: **http://dx.doi.org/10.1145/2807442.2807483**
2. Ronald Azuma. 1997. A Survey of Augmented Reality. *Presence* 6, 4 (1997), 355–385.
3. Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre. 2001. Recent Advances in Augmented Reality. *IEEE Computer Graphics and Applications* 21, 6 (2001), 34–47. DOI:**http://dx.doi.org/10.1109/38.963459**
4. Marion Buchenau and Jane Fulton Suri. 2000. Experience Prototyping. In *Proc. DIS*. 424–433. DOI: **http://dx.doi.org/10.1145/347642.347802**
5. Bill Buxton. 2010. *Sketching user experiences: getting the design right and the right design*. Morgan Kaufmann.
6. Pei-Yu (Peggy) Chi and Yang Li. 2015. Weave: Scripting Cross-Device Wearable Interaction. In *Proc. CHI*.
7. Alan Cooper, Robert Reimann, David Cronin, and Christopher Noessel. 2014. *About face: the essentials of interaction design*. John Wiley & Sons.
8. Steven Dow, Jaemin Lee, Christopher Oezbek, Blair MacIntyre, Jay David Bolter, and Maribeth Gandy. 2005a. Wizard of Oz interfaces for mixed reality applications. In *Proc. CHI Extended Abstracts*. 1339–1342. DOI: **http://dx.doi.org/10.1145/1056808.1056911**
9. Steven Dow, Blair MacIntyre, Jaemin Lee, Christopher Oezbek, Jay David Bolter, and Maribeth Gandy. 2005b. Wizard of Oz support throughout an iterative design process. *IEEE Pervasive Computing* 4, 4 (2005), 18–26. DOI:**http://dx.doi.org/10.1109/MPRV.2005.93**
10. Maribeth Gandy and Blair MacIntyre. 2014. Designer's augmented reality toolkit, ten years later: implications for new media authoring tools. In *Proc. UIST*.
11. Peter Hamilton and Daniel J. Wigdor. 2014. Conductor: Enabling and Understanding Cross-Device Interaction. In *Proc. CHI*.
12. Steven Houben and Nicolai Marquardt. 2015. WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications. In *Proc. CHI*. DOI: **http://dx.doi.org/10.1145/2702123.2702215**
13. Andrew J. Hunsucker, Kelly McClinton, Jennifer Wang, and Erik Stolterman. 2017. Augmented Reality Prototyping For Interaction Design Students. In *Proc. CHI Extended Abstracts*. 1018–1023. DOI: **http://dx.doi.org/10.1145/3027063.3053684**
14. Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard A. Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew J. Davison, and Andrew W. Fitzgibbon. 2011. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proc. UIST*. 559–568. DOI: **http://dx.doi.org/10.1145/2047196.2047270**
15. Bret Jackson and Daniel F. Keefe. 2016. Lift-Off: Using Reference Imagery and Freehand Sketching to Create 3D Models in VR. *TVCG* 22, 4 (2016).
16. Konstantin Klamka and Raimund Dachselt. 2017. IllumiPaper: Illuminated Interactive Paper. In *Proc. CHI*. 5605–5618. DOI: **http://dx.doi.org/10.1145/3025453.3025525**
17. Scott R. Klemmer, Jack Li, James Lin, and James A. Landay. 2004. Papier-Mache: toolkit support for tangible input. In *Proc. CHI*. 399–406. DOI: **http://dx.doi.org/10.1145/985692.985743**
18. Scott R. Klemmer, Mark W. Newman, Ryan Farrell, Mark Bilezikjian, and James A. Landay. 2001. The designers' outpost: a tangible interface for collaborative web site. In *Proc. UIST*. 1–10. DOI: **http://dx.doi.org/10.1145/502348.502350**
19. Boriana Koleva, Ian Taylor, Steve Benford, Mike Fraser, Chris Greenhalgh, Holger Schnädelbach, Dirk vom Lehn, Christian Heath, Ju Row-Farr, and Matt Adams. 2001. Orchestrating a mixed reality performance. In *Proc. CHI*. 38–45. DOI: **http://dx.doi.org/10.1145/365024.365033**
20. James A. Landay and Brad A. Myers. 1995. Interactive Sketching for the Early Stages of User Interface Design. In *Proc. CHI*. 43–50. DOI: **http://dx.doi.org/10.1145/223904.223910**
21. Chunyuan Liao, François Guimbretière, Ken Hinckley, and James D. Hollan. 2008. Papiercraft: A gesture-based command system for interactive paper. *ACM Trans. Comput.-Hum. Interact.* 14, 4 (2008), 18:1–18:27. DOI: **http://dx.doi.org/10.1145/1314683.1314686**
22. James Lin, Mark W. Newman, Jason I. Hong, and James A. Landay. 2000. DENIM: finding a tighter fit between tools and practice for Web site design. In *Proc. CHI*. 510–517. DOI: **http://dx.doi.org/10.1145/332040.332486**
23. Hao Lü and Yang Li. 2012. Gesture Coder: A Tool for Programming Multi-touch Gestures By Demonstration. In *Proc. CHI*.

24. Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. 2004. DART: a toolkit for rapid design exploration of augmented reality experiences. In *Proc. UIST*. DOI: **http://dx.doi.org/10.1145/1029632.1029669**
25. Paul Milgram and Fumio Kishino. 1994. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems* 77, 12 (1994), 1321–1329.
26. Bill Moggridge and Bill Atkinson. 2007. *Designing interactions*. Vol. 17. MIT press Cambridge, MA.
27. Michael Nebeling. 2017. XDBrowser 2.0: Semi-Automatic Generation of Cross-Device Interfaces. In *Proc. CHI*.
28. Michael Nebeling and Anind K. Dey. 2016. XDBrowser: User-Defined Cross-Device Web Page Designs. In *Proc. CHI*.
29. Michael Nebeling, Theano Mintsi, Maria Husmann, and Moira C. Norrie. 2014. Interactive Development of Cross-Device User Interfaces. In *Proc. CHI*.
30. Don Norman. 2013. *The design of everyday things: Revised and expanded edition*. Basic Books (AZ).
31. Peter Ondruska, Pushmeet Kohli, and Shahram Izadi. 2015. MobileFusion: Real-Time Volumetric Surface Reconstruction and Dense Tracking on Mobile Phones. *TVCG* 21, 11 (2015).
32. Thammathip Piumsomboon, Adrian J. Clark, Mark Billinghurst, and Andy Cockburn. 2013. User-Defined Gestures for Augmented Reality. In *Proc. INTERACT*. 282–299. DOI: **http://dx.doi.org/10.1007/978-3-642-40480-1_18**
33. Marc Rettig. 1994. Prototyping for Tiny Fingers. *Commun. ACM* 37, 4 (1994), 21–27. DOI: **http://dx.doi.org/10.1145/175276.175288**
34. Yvonne Rogers, Helen Sharp, and Jenny Preece. 2011. *Interaction design: beyond human-computer interaction*. John Wiley & Sons.
35. Carolyn Snyder. 2003. *Paper prototyping: The fast and easy way to design and refine user interfaces*. Morgan Kaufmann.
36. Jie Song, Gábor Sörös, Fabrizio Pece, Sean Ryan Fanello, Shahram Izadi, Cem Keskin, and Otmar Hilliges. 2014. In-air gestures around unmodified mobile devices. In *Proc. UIST*. 319–329. DOI: **http://dx.doi.org/10.1145/2642918.2647373**
37. Eugene M. Taranta II, Amirreza Samiei, Mehran Maghoumi, Pooya Khaloo, Corey R. Pittman, and Joseph J. LaViola Jr. 2017. Jackknife: A Reliable Recognizer with Few Samples and Many Modalities. In *Proc. CHI*. 5850–5861. DOI: **http://dx.doi.org/10.1145/3025453.3026002**
38. Daniel Vogel and Ravin Balakrishnan. 2004. Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users. In *Proc. UIST*.
39. Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. 2007. Gestures without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes. In *Proc. UIST*.
40. Feng Zhou, Henry Been-Lirn Duh, and Mark Billinghurst. 2008. Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR. In *Proc. ISMAR*. 193–202. DOI: **http://dx.doi.org/10.1109/ISMAR.2008.4637362**