XR Tools and Where They Are Taking Us

Characterizing the evolving research on augmented, virtual, and mixed reality prototyping and development tools

This article reviews the significant growth in XR tools research over the past few years. It first identifies key dimensions to consider when assessing XR tools, then presents trends in XR research along these dimensions. The author concludes with three wishes for future research to foster the design of new XR authoring tools.

By Michael Nebeling

DOI: 10.1145/3558192

Many user experience and interaction designers today are excited about creating augmented, virtual, and mixed reality (XR) applications, but there are still many barriers to entry. Since starting my research group at the University of Michigan in 2016, I have enjoyed building XR toolkits and authoring tools to enable more designers to create XR experiences and help shape the XR tools landscape from an HCI perspective. Despite much of the HCI research, creating XR experiences is still hard for many reasons. For example, getting started with XR is still hard, often a patchwork of tools is required to design and implement XR experiences, there is generally a lack of guidelines and metrics that constitute a "good" design, and it is difficult to find the right examples and tools for novice or less technically inclined XR creators [1]. Tools like Unity and Unreal now provide native support for XR development, but they can be overwhelming and the learning curve quite high.

SO, WHAT XR TOOLS DO WE NEED?

Much of the HCI research on XR authoring tools is focused on enabling high-fidelity prototyping without requiring much technical expertise. Two important criteria in this regard are Myers et al.'s "threshold" and "ceiling" [2]. How easy is it for a new creator to pick up a tool (threshold)? How much can they do with it and what are the limitations (ceiling)? In 2018, my postdoc, Max Speicher, and I classified the XR tools landscape in terms of the fidelity that can be achieved and the required skill [1]. Tools like Unity and Unreal in the highest classes support more complex XR prototypes (higher fidelity), but also require scripting (more technical skill). Additional challenges we identified include maintaining an overview of the design space covered by different tools given each

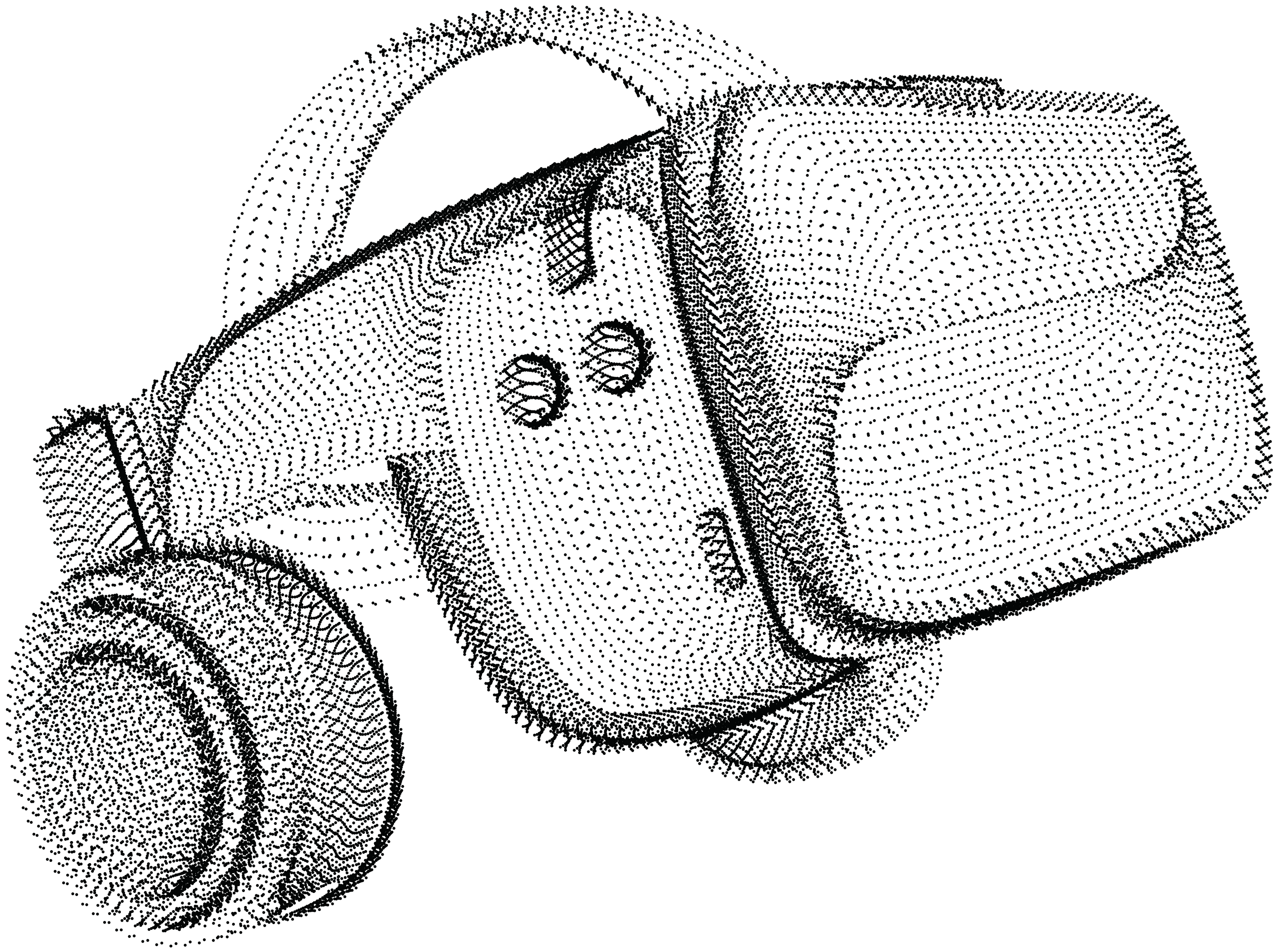


Image by printstocker / Shutterstock.com

tool's varying limitations, as well as transitioning between tools, which is often necessary for design iteration but hard when tools are in different classes. Many new tools have appeared since then, particularly for immersive authoring. While the gaps between the classes may have become smaller, the classes overall have not changed much.

Lim and Stolterman argue focusing on fidelity overemphasizes evaluation rather than support of design exploration [3]. Prototypes play a role as filters, to traverse a design space, and as manifestations, to represent design ideas. Prototyping is about a designer finding the manifestation that filters the qualities they are interested in, allowing the designer to reflect on the knowledge they have gained through the prototyping process. Rettig's "prototyping for tiny fingers" [4] emphasizes the role of paper prototyping as a way to rapidly bring design ideas to life. This form of prototyping does not require programming. Rather, with the help of a human "computer," design ideas can be produced for users and tried out even in the low-fidelity stage. In the XR space, this can be very difficult because XR technology is often "bleeding edge." Too often XR tools stand in the way of designing something quick-and-dirty or impose a particular process on the designer. This is why I am intrigued by XR tools that aim to complement, rather than replace, designers' existing workflows, which has been the focus of much of my own research.

Creating XR experiences is a significant responsibility that novice designers may not always realize. An easy mistake to make is to reduce the XR experience to interacting with 3D content in virtual reality (VR) or augmented reality (AR). When I think of XR authoring, I think of two key concepts, context and story, and how they differ between VR and AR (see Figure 1). In terms of context, a VR designer needs to author not just the virtual content. Rather, they need to design the VR environment to fit the constraints (and affordances) of the user's physical context. Similarly, an AR designer needs to author not just the virtual elements, but the user experience as a whole that results from blending the virtual and physical worlds. In terms of story, the VR designer needs to tell the story at

Figure 1. Two key elements to creating XR experiences are context and story. Most existing XR tools focus on only the story part and fail to address the context requirement.

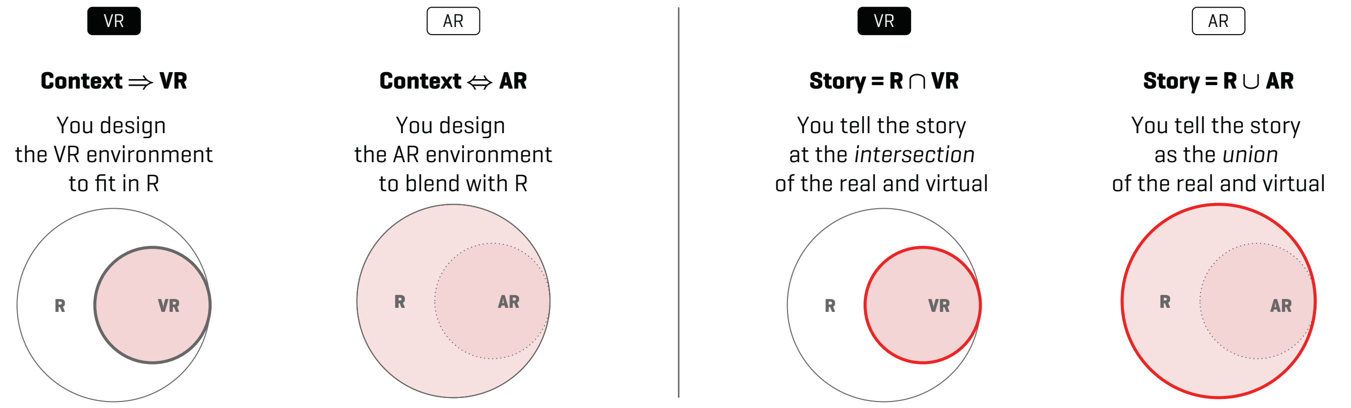

VR
Context ⇒ VR
You design the VR environment to fit in R
R
VR
AR
Context ⇔ AR
You design the AR environment to blend with R
R
AR
VR
Story = R ∩ VR
You tell the story at the *intersection* of the real and virtual
R
VR
AR
Story = R ∪ AR
You tell the story as the *union* of the real and virtual
R
AR

Figure 2. Categorization of XR tools across context, story, interactivity, collaboration, cross-device, and cross-reality. Evaluation is included as an additional category. XR tools between categories indicate hybridization; lines between XR tools indicate citations.

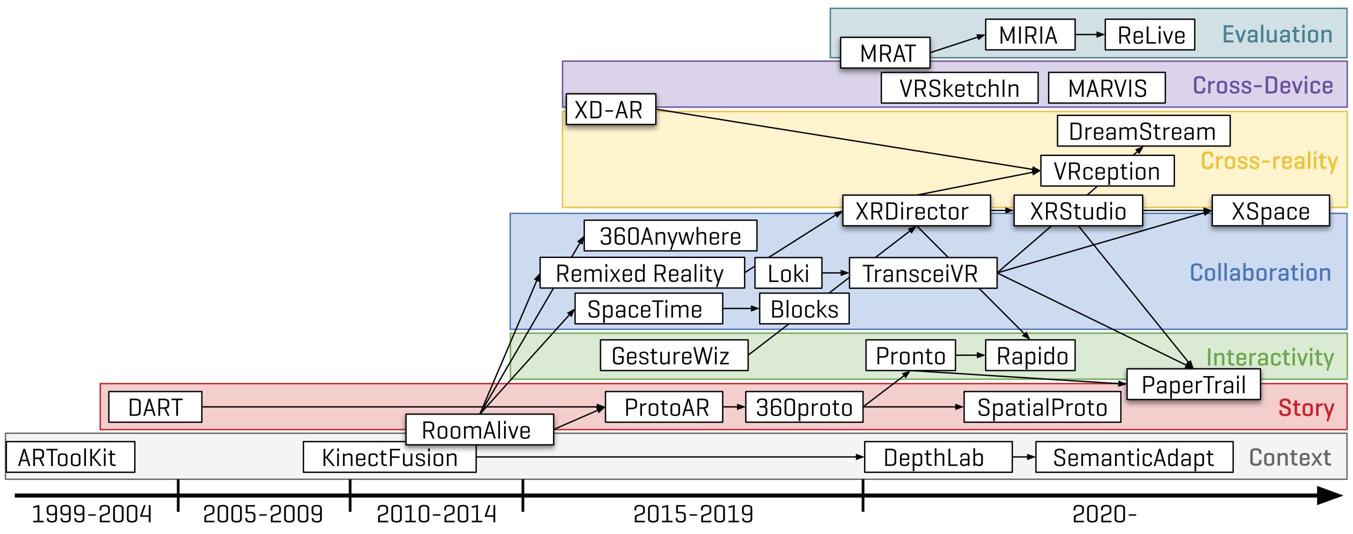
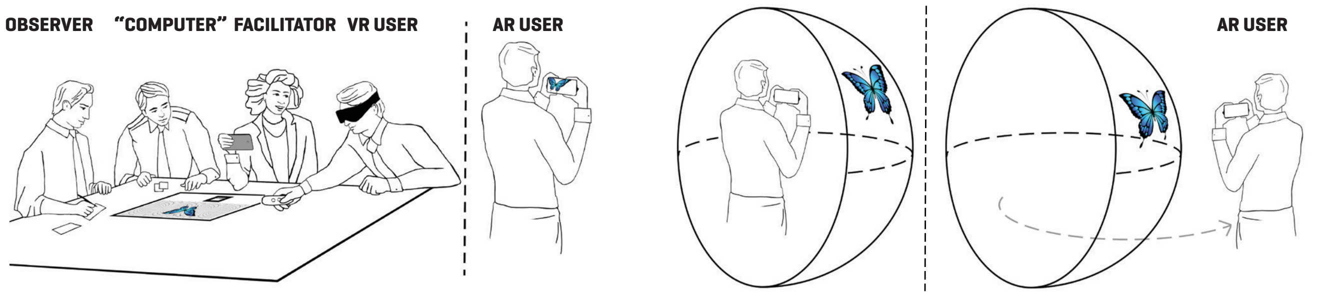


Figure 3. Paper prototyping is simple yet powerful for testing user experiences early. 360proto is an example of a tool that adapts this familiar technique for XR based on 360 paper templates.



the intersection of the real and virtual worlds, not just the virtual. In contrast, the AR designer needs to tell the story as the union of the real and the virtual. Context drives the story in both cases but in different ways. Unfortunately, much of the XR tools today focus on only the story part and fail to address the context requirement. This leads to basic issues such as spawning VR content where the user is likely to collide with physical objects, or making AR characters walk through physical objects, constantly disrupting the user experience and limiting presence and immersion in XR. Therefore XR tools need to strike a balance supporting both concepts and their nuances with respect to VR and AR authoring.

Finally, the HCI literature often emphasizes novelty in XR tools research, identifying particular features of a tool that go beyond what was previously possible. There has been a long discourse in the HCI community on how to evaluate tools research with guiding frameworks by several prominent authors. As UIST 2021 program co-chairs, Ranjitha Kumar and I developed a section in the author guide that argues for holistic evaluation of HCI tools research [5]. So, while reviewers tend to assess novelty and significance of the contribution (the delta), I see tools research more as a function of how it shapes the space. I call this the "forward vector." It is about the direction in which a tool is pointing, not the length. I find it is less important that the direction is new. The direction has to be interesting. A key reason I find XR tools research exciting is that I still see so many interesting directions to explore. In the remainder, I will highlight a few of them.

Most AR/VR devices are still expensive and not in widespread use, which is one of the main issues hindering adoption in many educational settings.

TRENDS IN XR TOOLS RESEARCH

For my analysis, I have focused on XR tools that were presented in the HCI literature (mostly CHI/UIST) in the last decade. I have identified the "forward vector" (or main direction) of each tool and grouped tools that point in the same direction. Figure 2 shows seven vectors with examples from the literature [6]. Apart from those focused on context and story, new directions include supporting more interactivity, enabling collaboration, supporting cross-device interaction, and transitioning AR/VR modalities (cross-reality). Looking at the vectors more closely, I see five key research themes (or trends) that have started to emerge. For each trend, I also consider the developments in industry and opportunities for research to shape practice.

1. Telling the story leveraging non-XR methods and tools already familiar to designers. A lot of research over the last two decades is commonly motivated by the need to enable more people to participate in XR design. A common audience is those who are already trained in 2D design and want to translate their skills to 3D and XR content creation. Early examples include the Designer's Augmented Reality Toolkit (DART), which was designed to enable a rapid transition from storyboards to a working prototype in AR that can be experienced quickly. DART was implemented on top of Macromedia Director, a popular tool among designers at the time, to enable visual authoring of story-based AR experiences through 3D animatic actors as informal, sketch-based representations of the final AR content. My work on ProtoAR and 360proto emphasized paper's key role in user experience and interaction design. I created tool support around new sketching and prototyping techniques using paper cut-outs and Play-Doh models for quickly creating 3D characters and 360-degree paper templates for environmental design and character animation in 3D space. ProtoAR implemented a pipeline to capture 2D content with mobile phone cameras and transform it into 3D representations using 3D planes to show the extracted content captured from different camera perspectives. 360proto built on Rettig's prototype testing workflow to bring paper prototypes to life in VR and AR. This required a facilitator to capture the physical materials in equirectangular format while a "computer" places and moves the content along a 360 grid, to animate the digital captures spatially around the user (see Figure 3). These simple techniques worked without depth cameras and 3D reconstruction algorithms and were "good enough" for rapid prototyping. SpatialProto is the latest tool I know which extends this line of work to include full 3D reconstruction.

Another key aspect to these tools is that they enable testing early and often. In DART, designers could capture and replay synchronized video and sensor data, making it possible to experience and test their prototypes off-site. Our work on the Mixed Reality Analytics Toolkit (MRAT) adapted common usability testing methods for XR, providing support for capturing, replaying, and analyzing XR experiences using a set of familiar metrics such as task completion time and error rate that we tried to automate. We also created in-situ visualizations of the collected data so that designers can analyze user studies in the same context using AR. Some of the latest tools adapt MRAT's principles to increase support for spatio-temporal interaction data analysis and immersive analytics. In Figure 2, I list evaluation as a separate vector since there appears to be significant movement in that research direction.

In practice, Unity and Unreal are not good examples of this trend although they come with powerful visual editors that, to a limited extent, resemble the look of tools like Photoshop, Illustrator, and After Effects. However, interaction designers who often do not have a game design background leverage relatively little that seems familiar. I would not consider immersive authoring tools like Tilt Brush, Quill, or Gravity Sketch. While they enable sketching and storyboarding in VR, it is a big leap from sketching on paper and in 2D. They also try to port all the widgets for content layout and animation known from desktop publishing tools.

However, these tasks are often more efficient on a desktop than trying to do everything in VR. Rather, the best example I can think of is A-Frame, a web framework that leverages familiarity with HTML/CSS and JavaScript, and extends it with the Entity-Component-System (ECS) framework to specify 3D scene graphs and XR behavior in a declarative way, which is familiar to web designers and relatively easy to pick up. It also allows testing on desktop and XR devices, and since it is based on the web, also makes it easy to share XR experiences with others. Another example is Figma, a popular interaction design tool that has been extended with templates for prototyping HoloLens experiences based on MRTK.

2. Dealing with context by providing abstractions and new software interfaces. In the early 2000s, context awareness and finding ways to represent context in toolkits was a popular research topic with the Context Toolkit being a prime example. As explained earlier, for AR, it is important to design the spatial user experience, which is highly context dependent. Earlier efforts in making AR programming easier led to ARToolKit, a marker-based toolkit where fiducials can be used to specify the context and derive spatial relationships between the user and environment. For marker-less AR, the Kinect sensor and Kinect Fusion toolkit provided a foundation for real-time 3D reconstruction and interaction using a moving depth camera. One of my favorite examples along this direction is RoomAlive, a Kinect-based toolkit for creating spatial (or projective) AR experiences that enables AR experiences without holding or wearing an AR device. In RoomAlive, projector-depth camera units are individually auto-calibrating and self-localizing, which automates the projection mapping of the AR content to the user environment requiring only little user intervention. DepthLab extends these principles to mobile AR, making real-time 3D interaction with depth maps accessible to AR content creators. It enables depth-based UX paradigms through data structures and techniques that provide abstractions to simplify geometry-aware rendering of occlusions and shadows, spatially map the environment to enable surface interactions via collision detection and path planning, and visual effects to enhance AR experiences.

Unity and Unreal have come a long way and are the go-to tools for creating XR applications in practice. One of the key issues in industry has been the sheer variety of devices that are part of the XR landscape. All XR devices come with SDKs for at least one of the two. Unity has specifically adopted XR technologies into its core, initially by bundling with AR tools like Vuforia that simplify marker tracking tasks, then creating abstractions like AR Foundation to enable marker-less AR for ARKit/ARCore on iOS and Android. Recently, Unity has adopted standards like OpenXR into its key technology stack, now supporting a rich set of AR and VR devices and platforms via built-in XR plugin management, a unified input system, and developer tools such as XR Interaction Toolkit (XRI), which in turn form the basis of more sophisticated toolkits like MRTK with its universal widget library and interaction handlers.

3. Increasing interactivity without the need for programming. Another trend in many tools, including some I already mentioned, is to provide scripted behaviors or find ways to simulate interactions without the need for programming. DART structured behaviors into those relevant for characters in the scene (or actors), events and cues such as showing a marker, pressing a button, or moving the camera (hence the device) to a position. The DART researchers also explored how Wizard of Oz techniques can be extended to AR, by enabling remote control of AR content to improvise support for interactions that are not yet implemented in code. With GestureWiz, we demonstrated that human wizards can recognize gestures with an accuracy and speed sufficient for prototyping, allowing relatively complex interaction proposals obtained in user-driven elicitation studies to be enacted and experienced with the help of human wizards. In XRDirector, we created a system that allows designers to embody any 3D characters in XR scenes, including lights, sounds, and the camera itself, and manipulate their appearance (both geometry and material), including parameters such as object size, texture, and mesh, light intensity, sound volume, etc. in response to user input, creating fully interactive XR experiences for test users that are simulated by designers acting in different roles. XRDirector also supported post-processing to adjust spatio-temporal parameters and export of animation timelines generated from the demonstrated behaviors. Tools like Pronto and Rapido extended these principles to video and immersive prototyping, supporting programming-by-demonstration and generating executable state machines.

The XR authoring tools discussed here are geared toward novice XR content creators. This makes privacy/security considerations particularly important.

In practice, the Wizard of Oz approach is an accepted technique in the early stages of design, but even the latest immersive authoring tools like Reality Composer, Aero, or ShapesXR do not provide support for remote control and programming by demonstration. However, many of these tools support exporting the assets and basic interactive behaviors to Unity with scripts. As Unity and Unreal have healthy developer communities, one is likely to find scripted behaviors as part of a packaged solution in their online stores.

4. Enabling novel forms of collaboration to accelerate XR content creation. The vast majority of XR tools were designed with a single user in mind, though tools like Unity and Unreal have add-on support for project sharing and collaboration. A more recent trend in research has been enabling different forms of collaboration and studying how they could benefit

XR content creation. At the highest level, we can distinguish between support for synchronous and asynchronous collaboration (same time or different time), and symmetric and asymmetric collaboration (same or different modalities). Tools representative of this trend include SpaceTime, a VR scene editing tool introducing novel interaction concepts to group, transform, and clone 3D characters and other objects in the scene—including user avatar objects—to support simultaneous editing of the same objects and coordination of different users while maintaining individual users' agency. Blocks supported both synchronous and asynchronous collaboration between multiple remote co-located or remote AR users. With Remixed Reality, Lindlbauer and Wilson implemented principles of spatio-temporal manipulations of 3D reconstructed live or video feeds captured with multiple external depth cameras. Loki built on this work to enable asymmetric remote collaboration through mixed-reality telepresence scenarios where a local user working in AR can be instructed by a remote user in AR or VR, with the local user's environment being reconstructed for the remote user. TransceiVR extends this work to asymmetrical interactions with one user in VR and another on a tablet, allowing the remote user to annotate the VR scene at the correct depth by reconstructing depth from the VR stereo-view video feed. In 360Anywhere, we explored mobile adhoc collaboration with users in a meeting room sending a live 360 video feed and using a projector for mixed reality telepresence. In XRDirector, we adopted roles from filmmaking to structure the asymmetric collaboration process between designers working in 3D, VR, and/or AR. In XRStudio, we developed a VR scene annotation tool for instructors using mixed-reality capture and live streaming of lectures conducted in VR that students can attend remotely on a computer. Finally, with XSpace, we present a framework and tool support for configuring virtual collaboration spaces that can be based on real physical environments but mixed and matched from virtual cut-outs to fit the collaborator's needs.

A more recent trend in research has been enabling different forms of collaboration and studying how they could benefit XR content creation.

5. Using multiple devices and transitioning AR/VR modalities through cross-reality support. A final trend in research that is also beginning to emerge in industry with Unity's Mixed and Augmented Reality Studio (MARS) being a precursor is increased support for context awareness and adaptivity, allowing XR applications to better adapt to the user's environment and task. In the latest tools presented in research, this goes beyond the cross-platform support in Unity and MRTK, where the same application code can be deployed to a VR or AR headset, toward adaptable systems that can seamlessly transition the reality-virtuality continuum. Our earlier work on XD-AR is relevant here because we asked what responsive design could mean for XR and developed a conceptual framework and first toolkit to support multi-user applications that can run on different AR displays (hand-held, head-worn, and projective). Our principles were implemented in several XR authoring and evaluation tools, including VRSketchIn, MARVIS, and our own MRAT, to provide pen-and-touch tablet interfaces for 3D sketching and visual data analysis tasks in VR/AR. Collectively, these systems cover a large portion of the mixed-reality continuum, but none of them allow directly transitioning it according to the user's task, preferences, or needs. VRception is a recent prototyping tool for authoring "cross-reality" experiences, allowing a designer to experience their prototype in VR or AR via pass-through video feeds of the physical world. What is lacking is a way of specifying how and when these modality transitions should happen. Recent research on context-aware adaptation of mixed-reality applications enables developers to specify rules and conditions for automatically adapting virtual content to the user's environment. When combined with procedural generation of VR content according to the physical context demonstrated in DreamWalker, future XR applications may allow a degree of flexibility and end-user customization that will balance system and user control so that the XR applications themselves become authoring tools. While this sounds exciting, this is likely to create many interesting new issues worth studying.

THE THINGS I'D LIKE TO SEE...

Overall, we can see alignment between the things explored in research and how industry practice is shaping. There are, of course, many new ideas explored exclusively in research. The papers included in this review do a fairly good job of building on each other, although it is sometimes tricky to tell who pushed the envelope further and by how much. I also have not offered my reflection on the state of the art yet. The 10 years of DART study analyzed the usage of an AR tool that originated in research and how different types of designers appropriated it for their projects and tasks. As I am thinking about the XR tools landscape I have just laid out, the key findings from the DART study, and my own research on needs of novice XR creators [1], I believe we still face many of the same issues. The DART researchers identified debugging AR applications as a key challenge. More research was needed into debugging constructs for designers whose expertise and developmental approach might differ from someone trained in software engineering. They also noted that while they were pleased with the variety of projects enabled by DART, many features they had conceived to support designers were underused. They concluded that, in addition to better documenting those features, it was important to lay out workflows for using them.

If I could make three wishes for future XR tools research, this is what I would like to see happen:

Exploring different programming and debugging paradigms to better

cater to XR. While originally unfamiliar, I have gotten used to Unity's component-based programming model. I also see a lot of value in A-Frame's ECS paradigms. But I also question them. It is so hard to walk my students through how certain XR interactions are implemented in these systems. I have to go over the scene graph, find specific components, and explain what happens at what point in the rendering loop. There is relatively little research in HCI that challenges this. I appreciated Mercury from Steven Feiner's lab, which explored a message-based approach to increase modularization of user interfaces implemented in Unity, and Zhang and Oney's FlowMatic, which combines dataflow and functional reactive programming in an immersive authoring tool. Personally, I have been intrigued by the popularity of the Processing framework among designers, which was designed for teaching non-programmers key principles of programming in a visual context. I am currently developing a p5.js-inspired version of A-Frame. My plan is to gather experience using this version in some of the XR courses I teach to improve the tool support and then work with others teaching XR to try to address their course-specific needs. Finally, I have also been working on immersive debugging approaches that visualize interaction flow and reimagine familiar debugging constructs like breakpoints in XR development.

Incorporating equity, accessibility, and privacy/security considerations into XR tools. In addition to improving the underpinnings of XR authoring tools, also broadening access to XR has been one of my major concerns. Most AR/VR devices are still expensive and not in widespread use, which is one of the main issues hindering adoption in many educational settings. In our recent work on Paper Trail, we explored how instructors might want to enhance paper-based instructional materials with AR content to enrich existing teaching methods. I would also like to see more XR tools like TransceiVR and XRStudio because they enable asymmetrical interaction which can help address part of the issue. I believe research like DreamStream to increase audience interactions during VR streaming is an important direction. I would also like to see more research into accessibility. SeeingVR implements 14 visual and audio augmentations designed to enhance a VR application for people with low vision. What is missing is research into making the XR authoring tools themselves accessible. For our AR-based iGYM project, we are currently exploring how to best extend the platform to allow game developers to directly work with motor impaired children to author new adaptive games for our floor projection system. Finally, many of the XR authoring tools discussed here are geared toward novice XR content creators. This makes privacy/security considerations particularly important. Much of the XR tools research at CHI and UIST tries to push the boundaries, but often fails to consider privacy/security issues in doing so. Since I have also been guilty of this, my PhD student, Shwetha Rajaram, and I have been working together with experts in AR/VR and privacy/security to design new methods and tools that promote safer XR design.

Designers today are excited about creating augmented, virtual, and mixed reality (XR) applications, but there are still many barriers to entry.

Learning key XR concepts and "good" design through the tools. I started this article by introducing my understanding of context and story as key elements to "good" XR design. I have been teaching XR for the past five years and have always tried to balance theory and practice, typically teaching concepts first independent of particular implementations in tools. I still believe that is fundamentally the right teaching philosophy. However, the truth is that many XR creators are self-taught and usually start by picking up a tool like Unity and making their way through a first tutorial and examples they find online. Tools like SteamVR or MRTK come with scenes that are packed with interaction examples that serve little more than the purpose of demonstrating the features of the toolkit but, to a much lesser degree, how interactions are best implemented. First, I would like to see more research into what constitutes good examples in XR authoring that others can learn from, and this is something we have been exploring in a few recent master theses in my lab. Furthermore, most of the XR tools I discussed here are borne out of an idea (which is great) to provide some kind of abstraction to simplify certain XR creation tasks. However, this means that the understanding of XR technology that is shaping as a result of a designer's using a particular tool is directly limited by these abstractions. I find it particularly interesting to explore ways of teaching more advanced tools like Unity by finding ways to transition to Unity from new XR authoring tools, and teaching "good" design directly through the tools.

References

[1] Nebeling, M., and Speicher, M. The trouble with augmented reality/virtual reality authoring tools. in *2018 IEEE International Symposium on Mixed and Augmented Reality, Adjunct (ISMAR-Adjunct)*. IEEE, 2018, 333–337.

[2] Myers, B., Hudson, S. E., and Pausch, R. Past, Present, and future of user interface software tools. *ACM Transactions on Computer-Human Interaction (TOCHI)* 7, 1 (2000), 3–28.

[3] Lim, Y., Stolterman, E., and Tenenberg, J. The anatomy of prototypes: Prototypes as filters, prototypes as manifestations of design ideas. *ACM Transactions on Computer-Human Interaction (TOCHI)* 15, 2 (2008), 1–27.

[4] Rettig, M. Prototyping for tiny fingers. *Communications of the ACM* 37, 4 (1994), 21-27.

[5] See https://uist.acm.org/uist2021/author-guide.html#systems for the UIST 2011 Author Guide.

[6] See https://github.com/mi2lab/xrds2022 for the full list of XR tools and citations.

Biography

Dr. Michael Nebeling is an associate professor at the University of Michigan School of Information where he leads a research group focused on HCI and AR/VR. At Michigan, he is a faculty innovator-in-residence with the Center for Academic Innovation advising them on XR strategies, created the Extended Reality for Everybody MOOC specialization on Coursera, and teaches core XR design and development courses in UMich's XR graduate certificate program. He enjoys playing computer games, most recently VALORANT.