Unity MARS Companion Apps

Matt Schoen, April 23, 2020

Technology

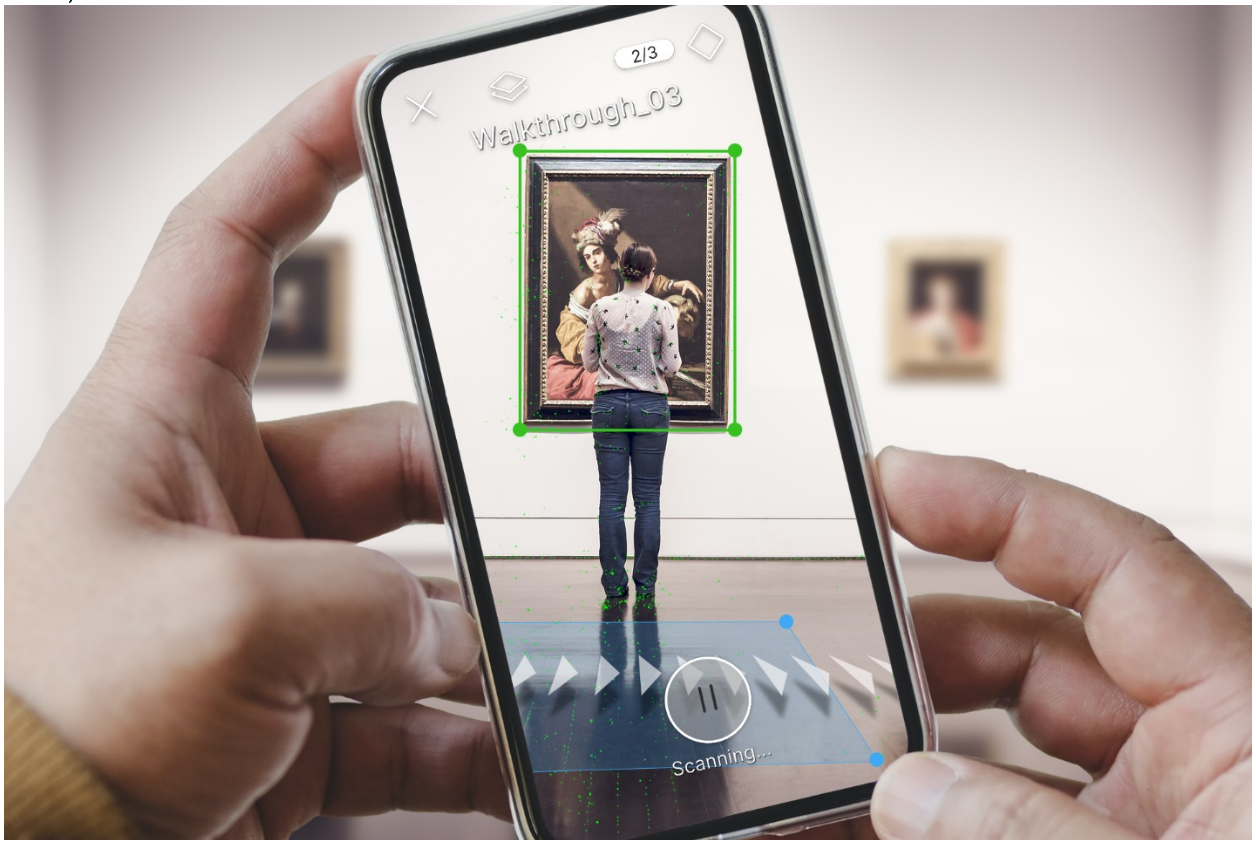
Unity MARS is a suite of authoring tools and runtime systems for creating the next generation of spatial computing applications. Companion apps that allow for authoring and data capture on augmented reality (AR) devices are a key part of this suite. We believe that the combination of PC Editor tools and on-device authoring applications is the most powerful and accessible way to create AR content.

We announced Unity MARS at Unite Berlin 2018 and shared our progress more recently at Unite Copenhagen, where we showed off an early version of the companion app for smartphones. We also published a blog post earlier this year on lessons learned in spatial design while working on the head-mounted display (HMD) version. You can find out more general information on our Unity MARS page. Here, we'll talk about the motivation behind creating the apps in the first place and how they fit within the overall AR authoring workflow of Unity MARS.

Editing in context

Content authoring is at its best when creators can take advantage of any and all tools available. With the Unity MARS companion apps, we have the opportunity to not only take advantage of 3D authoring using tracked devices, but enable users to do this in any location, on any AR device, without needing access to the full project and Unity Editor. If a creative agency is working on a museum tour app, they don't need to physically visit the museum to take pictures of the art or capture scans of the rooms. They can share their project with their client – the museum curator – who can install the

Unity MARS companion app from their smartphone app store, capture the necessary data, and save it to the cloud.



Users in remote locations can edit Unity MARS proxies against real data

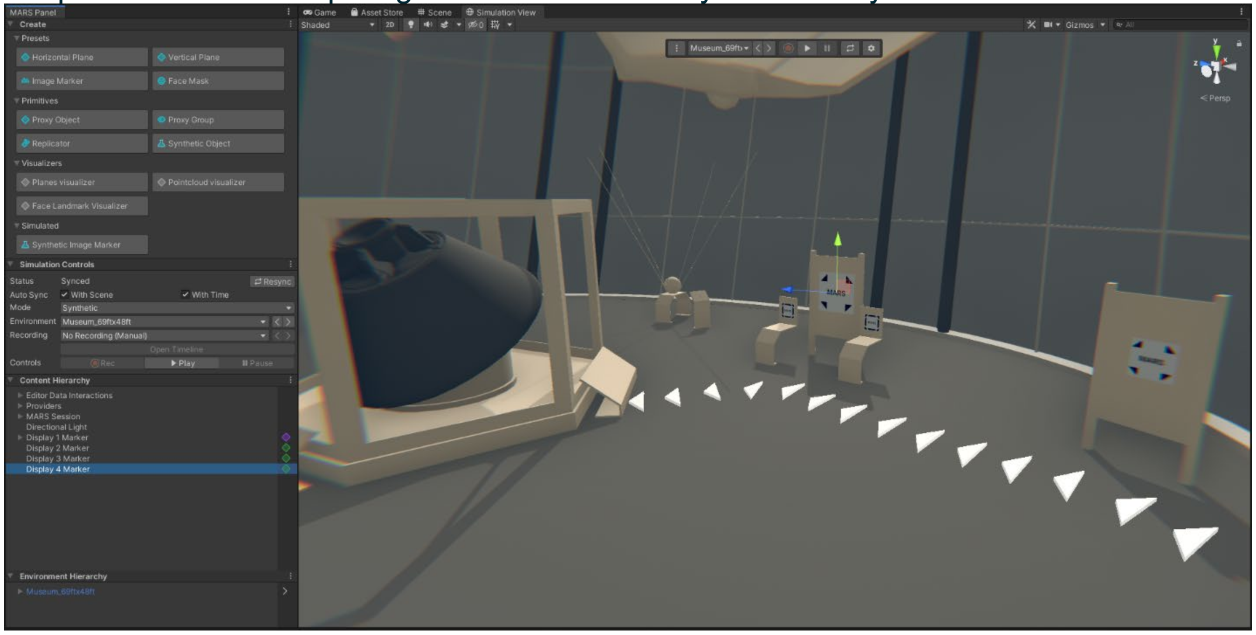
World-aware apps need to make sense of an unpredictable and dynamic environment. To confront these constraints head-on, Unity MARS includes a robust set of simulation features, including synthetic environments, simulated AR data discovery, and recording/playback for AR data. These tools are great for testing your data constraints or testing AR interactions in play mode. But the strongest advantage is that you can start to see your app as it will appear on a device in real-time. AR devices allow users to lay out their scenes in real space – either the dedicated location or any useful room – on the device. And if a bug occurs because of lighting conditions or a particular room setup, it's useful to have the actual device data to reproduce and fix the issue.

We are trying to create workflows that are accessible to nontechnical users but still familiar to Unity developers. You are editing the same scene, with the same assets, using tools that share terminology and, where it makes sense, interaction patterns. Proxies in the companion app have an active state, can have child objects, and show a list of components. You save/load a “scene” and group all of these assets in a “project.”

We don’t expose the full hierarchy and inspector, but these types of tools will eventually make their way into the EditorXR runtime feature set.

Why build companion apps?

One of the biggest pain points of mobile development in Unity is build times. You can learn about the strides we’ve made to improve build times by watching a recent Unite Now talk on the subject. All of these features help developers iterate and test their final app on their target hardware. With the Unity MARS companion apps, by using the target device’s AR capabilities to test and iterate on specific pieces of content, either during development or in real-world scenarios, more team members can be incorporated into the process without requiring them to work directly in the Unity Editor.



Status Synced
Auto Sync ✓ With Scene ✓ With Time
Mode Synthetic
Environment Museum_69ftx48ft
Recording No Recording (Manual)

The Unity MARS Simulation View

This is where the Unity MARS companion app comes in. The person who is laying out the scene, whether it’s a user, a client, or a remote team member, can capture their environment or make a data recording and upload it to the cloud, and any Unity user can download it in the Editor and see what their teammates have made. In the Editor, you can also tweak constraints until the app works in the virtual environment and save those edits to the cloud.

By building the companion apps in Unity, we are better able to serve our customers across all platforms and potential use cases. We are currently building companion apps for Android, iOS, Magic Leap, and Hololens, but in theory any platform supported by

Unity can serve as a platform for our companion apps. Any device that supports Unity and can capture real-world data is a potential platform for spatial authoring. Furthermore, each platform has its own strengths, and the best possible authoring environment involves using each device for its best characteristics. For precise, complex workflows, a keyboard and mouse are hard to beat. For getting a sense of presence and scale, as well as how to deal with a limited world understanding, we've found nothing is better than working on the device directly.

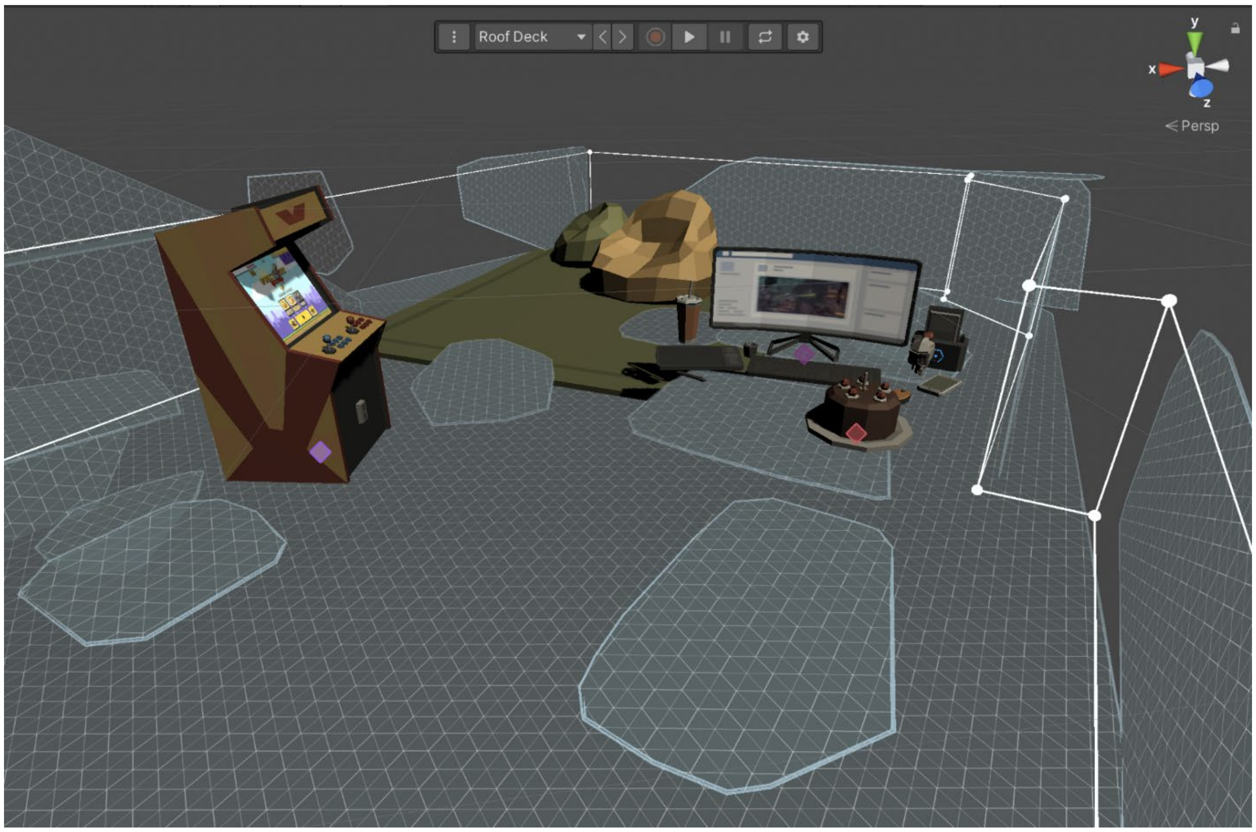
For even more immediate feedback, the XR Platforms team is working on an AR Remoting package that establishes a direct connection between the Editor and the device to stream live AR data. As these features come online, we will be incorporating them into the Unity MARS simulation view and companion apps.

Feature breakdown

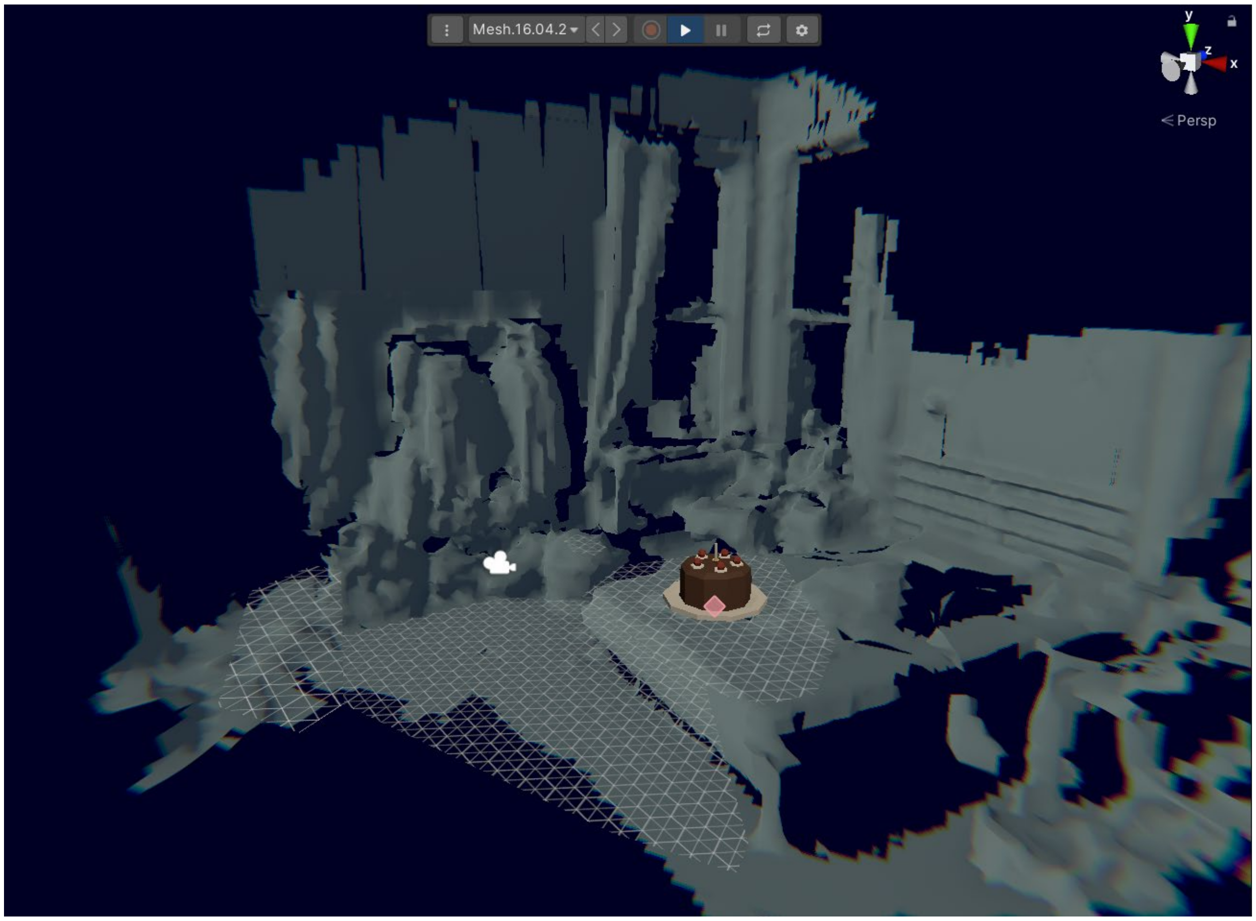
Users of the Unity MARS companion apps can perform two primary tasks: data capture and in-situ authoring. Both of these tasks involve sending and receiving data to and from the Editor, which is done via cloud storage. This means that users all over the world can work on the same project together, and that the data persists between sessions and is shared between users. We use the existing Cloud Services account and project permissions to control which users have access to what data. If a user can access the cloud dashboard for a particular project, they have access to that project's companion app data, both on their AR device and in the Editor. We use QR codes to transfer the project ID, and users will be able to log into the app using either their Unity account or a temporary token shared via QR code. Users can work offline or on an unlinked project and can later sync their data or resolve conflicts between edits made to the same resource based on when that edit was made.

Capturing data

Data capture comprises a few different tasks. Users can use the Environment mode to capture a static environment scan: either a full mesh in the case of a Magic Leap and Hololens, or a set of flat surfaces from ARKit and ARCore. In the future, it will be possible to integrate third-party software for smartphones to produce mesh scans or other AR data, and create custom builds of these apps to record it. For now, we are focusing on data provided by the OS. In the case of ARKit and ARCore, we also allow users to manually trace out the corners of their room to provide walls in the simulation environment. These aren't exposed to the simulation as data, but they help provide visual context for the surfaces that were scanned.

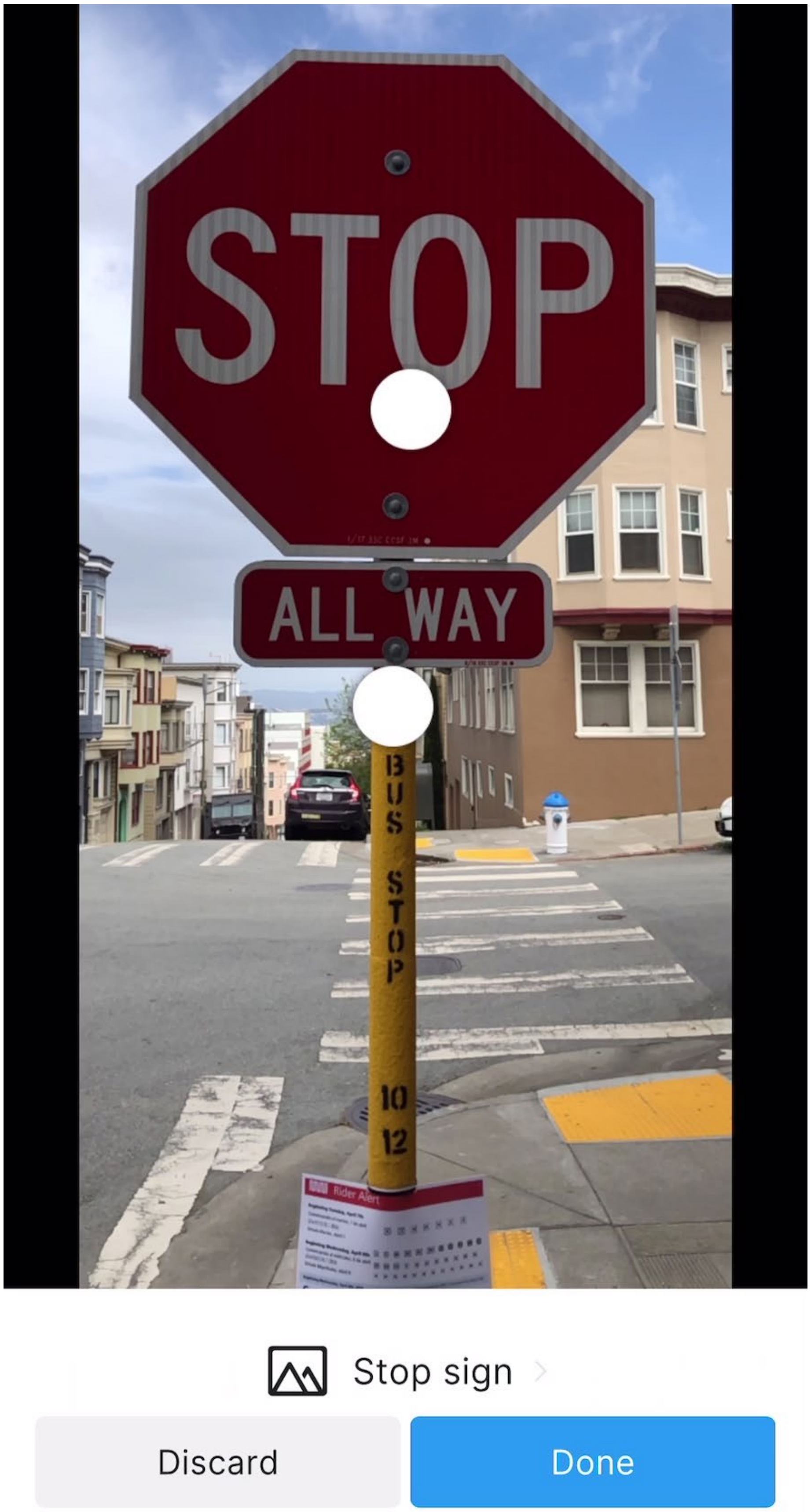


ARKit scan in simulation view



Magic Leap scan in simulation view

If the issue or interaction you wish to test involves the user moving around and scanning their environment, users of the app can use the Data Recording mode to capture video, camera path, and environment data as it changes over time. This can range from a simple "user approaches a surface" recording to a full recording of the scanning process from start to finish. We encourage developers to record basic user interactions for playback in the Editor to do as much refinement as possible on the "noisy" data that comes from how users handle devices in the real world. Of course, video files can be quite large, so brief recordings of specific actions are preferred over long recordings that cover multiple actions. It is also possible to select which data streams should be recorded and uploaded to reduce bandwidth usage.
Users can also capture AR Image markers with their device's camera and annotate specific locations, or "hotspots," on the image for anchoring content.


STOP
ALL WAY
BUS STOP
10
12
Rider Alert
Stop sign
Discard
Done

Capture photos for image markers and annotate with hotspots

Watch this space

From our experience on EditorXR, we have learned that being inside a VR scene while you are making edits to it is hugely beneficial. Further, the degree of control afforded by tracked head and hands is a great way to quickly lay out a scene and have fun while you're doing it. We also learned that things like manipulating text and fine-tuning are best done with a mouse and keyboard – it's best to avoid those tasks in AR/VR. In fact, the Unity MARS companion apps incorporate features of EditorXR that were created to support editing scenes at runtime. We are extracting these features into reusable packages to create a Runtime Authoring Framework. Transform manipulators, inspectors, and scene serialization will come from a shared code-base for doing authoring work outside of the Unity Editor.

Designing our workflows across devices has had some unexpected benefits. The Unity MARS Create and Compare flow, where users in the Editor can drag and drop assets directly into the Simulation View and have Unity MARS infer the necessary constraints, came from thinking about how to author Unity MARS content for an HMD. It makes a lot of sense to grab and place an object in the world where you want it to go when using a Magic Leap. It makes just as much sense to do it in the Editor, but we were only accustomed to dragging prefabs into the Scene View. As we push the capabilities of the companion apps further, we may yet find more ways to incorporate spatial design thinking into our 2D tools.

As we approach the public release of Unity MARS, we will be sharing more information about its features and lessons learned over the course of development. We plan to release builds of the smartphone companion apps on the App Store and Google Play alongside Unity MARS, so creators can dive right into these workflows from day one, with the HMD version to follow later in the year. We believe that Unity MARS is the best way to develop apps for spatial computing, and that creators are best served by using all of the tools available.

This is just the beginning. We are excited to expand the world of Unity authoring outside of the Editor and take our first steps toward a connected, distributed authoring environment. We can't wait to hear from you about what we should build next.

Related posts

Introducing Unity MARS – a first-of-its-kind solution for intelligent AR

Phil Chacko
June 8, 2020•42
Technology

How AR Foundation and Unity MARS work together to enable interactive, multiplatform AR experiences

Braelyn Johnson
May 11, 2020•9
Technology

Taking AR projects from imagination to reality

Ellen Grogan
September 3, 2020•3
Technology

A plain-language approach to authoring AR

Jono Forbes
September 2, 2020•Reply
Technology

A look at how Simulation works in Unity MARS

Amy DiGiovanni
August 14, 2020•3
Technology

Exploring the Unity MARS Starter Templates

Andrew Maneri +1
July 2, 2020•5
Technology

Introducing Unity MARS – a first-of-its-kind solution for intelligent AR

Phil Chacko
June 8, 2020•42
Technology

How AR Foundation and Unity MARS work together to enable interactive, multiplatform AR experiences

Braelyn Johnson
May 11, 2020•9
Technology

Taking AR projects from imagination to reality

Ellen Grogan
September 3, 2020•3
Technology

A plain-language approach to authoring AR

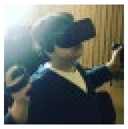
Jono Forbes
September 2, 2020•Reply
Technology

- 1
- 2
- 3
- 4
- 5
- 6

3 replies on “Unity MARS Companion Apps”

Stefansays:

May 27, 2020 at 6:10 pm
Are the companion apps and/or MARS packages going to be open source?



Narek Torosyansays:

June 9, 2020 at 12:35 pm
Most likely no, since it’s a paid feature.

wwindows518says:

May 3, 2020 at 4:09 pm
f