A look at how Simulation works in Unity MARS

Amy DiGiovanni, August 14, 2020

Technology

> The Simulation system in Unity MARS reduces testing time for augmented reality (AR) app development. It provides world-understanding capabilities, like plane and image marker detection, right in the Unity Editor — in both Play and Edit Mode. Read on to learn more about this system and how creators can use it to rapidly iterate on context-adaptable AR experiences.

Unity MARS makes it easy to create complex AR experiences that intelligently adapt to the real world. We designed this authoring tool to address the three most common pain points for AR developers: authoring for an enormous possibility space of data, iterating and testing against real-world scenarios, and delivering apps that adapt responsively to their environment.
Simulation solves the problem of iteration time by enabling AR testing within the Unity Editor to preview how an AR device would work with its physical surroundings. Simulation facilitates testing against a variety of environments that imitate or come from the real world. This makes it easier to create apps and experiences that are adaptable and accessible. In combination with the proxy and condition-matching systems of Unity MARS, Simulation helps empower creators to push the boundaries of spatial computing.
As a system, Simulation can be broken down into two core functionalities that are conceptually separate, but work together to drastically reduce iteration time:

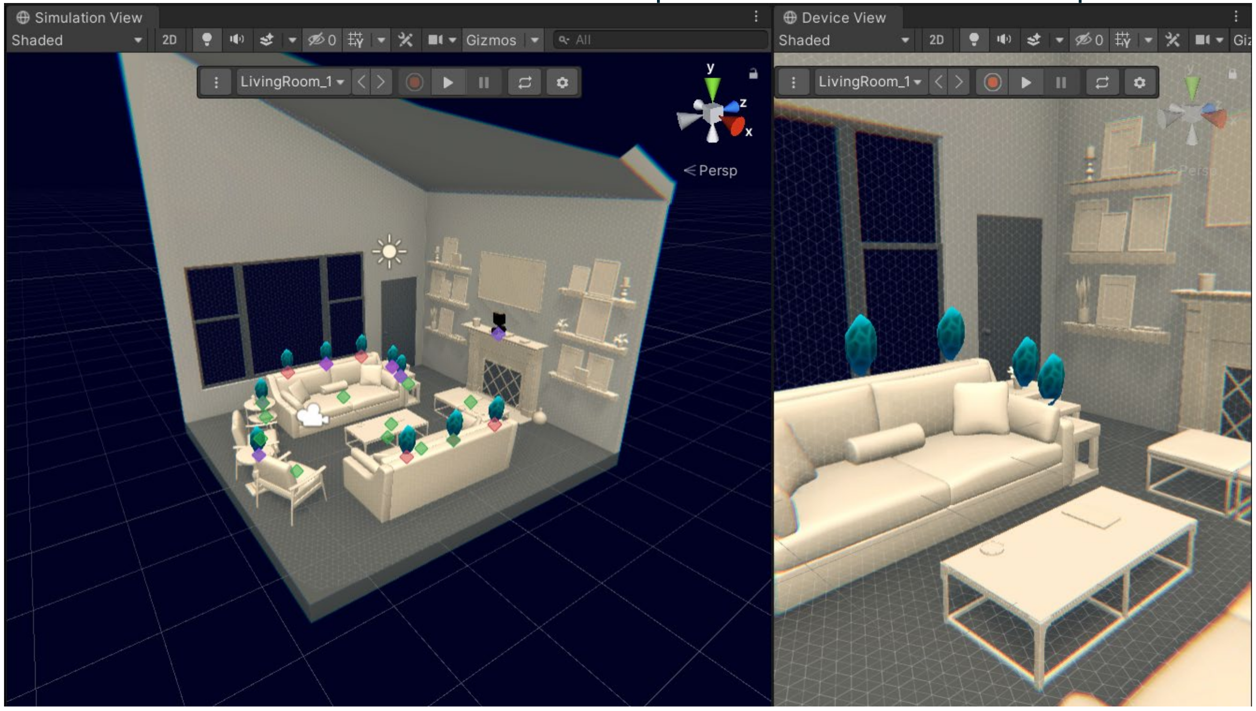
1. Preview the execution of a scene in isolation in Edit Mode.

2. Set up an environment scene in the Editor, and provide AR data based on that environment.

We'll go into detail on how these parts of Simulation work to improve AR development.

How execution previewing works

Simulation lets you preview the execution of a scene in Edit Mode to support rapid iteration and provide an instantaneous understanding of how flexible proxies that represent real-world objects might behave in the real world. In the execution preview process, Simulation copies the GameObjects in the active scene into a preview scene – called the *content scene* – and sets the runInEditMode flag on each MonoBehaviour in the content scene that opts into running. See the Unity MARS documentation for details on how to write MonoBehaviours that are compatible with Simulation scene preview.



With the Simulation view, you can interact with this scene execution preview in isolation from your open Unity scenes. The Simulation view is a custom Scene view that shows the content scene rendered over the environment scene (the next section digs more into the environment scene), from either a third-person perspective or from the camera's perspective.

In Unity MARS, the default behavior of a Simulation preview is to run in *instant mode*. In this mode, the entire execution process happens over a single frame, in which Unity MARS tries to match proxies against all data at once. While this behavior is not realistic, it has the benefit of providing immediate feedback as you define proxies and their conditions.

You can also manually start and stop a *continuous mode* Simulation preview. This mode is more analogous to Play Mode, in that the behaviors in the content scene run frame by frame until the preview is stopped. App interaction and behavior in continuous preview are limited, since some Unity systems, like physics and input, only work in Play Mode. Continuous preview is useful for seeing how your app responds to data that changes over time while still being able to quickly make revisions to your scene.

Video Player
00:00
00:17

The Simulation system keeps track of the status of the content scene in relation to the active scene, keeping it up to date as you make changes. If you make any property modification to an object that has a copy running in the content scene, Simulation picks up on that change and indicates that it is *Out of Sync*, which means the latest changes will not apply to the preview until it is resynced. In the resync process, Simulation destroys the content objects, replaces them with new copies from the active scene, and then starts running the new copies.

Simulation has an option to automatically resync preview, so you can iterate on proxies and instantly see your changes reflected in the Simulation view. This is especially powerful in combination with the visual authoring features of Unity MARS, which work not only in the standard Scene view, but also in the Simulation view. You can change proxy conditions in the context of the environment and instantly see how adjusting condition parameters changes the outcome. For large and complex scenes that would be expensive to copy, we recommend that you disable the auto-sync option and instead manually resync when necessary.

How Simulation provides AR data

The construct of an *environment* is fundamental to Simulation. An environment is a set of objects separate from the app content that is running in Simulation. Practically speaking, the environment is a scene with its own isolated physics and lighting that renders underneath the app content, both in the Simulation view as well as the Game view in Play Mode. This separation exists so that testing in the Editor can behave as closely as possible to testing on an AR device, with the environment acting as the real world in the Editor.

Data providers are the key link between the environment and content. Like other kinds of AR data providers, Simulation providers exist alongside the app content, but they also have access to information from the environment. To work with execution previewing, these providers must work in both Play Mode and Edit Mode. There are a variety of ways Simulation providers can use environment information to provide data, but there are three main categories of data sources that Simulation works with: synthetic, recorded, and live.

Synthetic data

Synthetic data is artificially created, rather than generated from real-world events. This data comes from a prefab that is instantiated in the environment scene. Unity MARS provides a set of default environment prefabs that includes rooms, buildings, and outdoor spaces. You can also create your own simulated environments.

We've launched Simulation with synthetic AR capabilities for motion tracking, geolocation, plane finding, point cloud detection, image marker detection, and light estimation, with more to come. Most of this data is provided instantaneously from synthetic objects within the environment prefab.

Video Player

00:00

00:27

Simulation can also create data for motion tracking, point cloud detection, and plane finding through a process of simulated data discovery. Simulated discovery provides data dynamically based on the contents of the environment prefab (the simulated "real world") and the state of the main camera in the Simulation content scene (the simulated "device"). To emulate data discovery on mobile devices, we only detect data when the camera moves and only use parts of the environment within the camera frustum to add to our simulated understanding of the world. We also create planes that grow in "chunks," as they do on mobile devices, but these planes are limited to facing up, down, left, right, forward, or back.

Recorded data

Simulation can also provide data by playing back a recording of an AR session. These recordings can be created using the upcoming Unity MARS companion app or using Simulation itself. Unity MARS also provides sample recordings of face-tracking data. Recorded session data is stored in a Timeline asset, with one track for each type of data. A Playable Director in the environment plays the session recording Timeline, and provider objects in the app content are bound to data tracks so they can provide the actual data. See the [Unity MARS documentation] for more information on session recording objects.

Because session recordings are represented as Timelines, you can work with them as you would other Timelines. For example, you can modify the looping behavior or tweak the contents of the data tracks.

Video Player

00:00

00:18

In Edit Mode, you can also scrub through a recording in the Timeline window. An important caveat of scrubbing is that **the proxy system and other Unity MARS behaviors cannot run backward in time**, so the Simulation system treats time scrubbing as a change that requires a resync of the execution preview. When execution restarts, the recording begins anew and execution runs at a faster time scale to catch up to the new time. For details on how to write behaviors compatible with time resyncing, refer to the Unity MARS documentation.

Live data

This data comes from real-time, real-world events. We've launched Simulation with the ability to provide a live camera image via a webcam device. The environment in this case includes the video player object and the camera image quad, while the data provider in the app content provides the raw texture. A third-party license is required to get face-tracking data from this texture.

Layers of Unity MARS and AR development

Unity MARS can work with data from AR Foundation, from Simulation, or from custom providers. All of this data goes through an abstraction layer that includes the Unity MARS database – used by the proxy-matching system – as well as a layer for general AR functionality. This way your app content and the systems that use data do not need to know which device the data comes from or whether the data comes from Simulation.

We know Simulation is a powerful general tool for AR development. **We're excited to share that later this year, creators using Unity MARS will be able to use the Simulation system without requiring the proxy system.** With Simulation as its own XR subsystem, creators will be able to take advantage of Edit Mode previewing and AR capabilities in the Editor, adding flexibility and Simulation support for any kind of

XR project you're working on. And, of course, all Unity MARS features, including Simulation, will always work with AR Foundation.

For more information about Simulation and Unity MARS

Visit our webpage for more information about Unity MARS and join us in the Unity Forums for further discussion and questions related to Simulation.

Get started with Unity MARS

Related posts

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

Braelyn Johnson
May 11, 2020•9
Technology

Unity MARS Companion Apps

Matt Schoen
April 23, 2020•3
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

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

Unity MARS Companion Apps

Matt Schoen
April 23, 2020•3
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

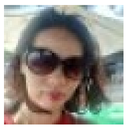
- 1
- 2

- 3
- 4
- 5
- 6

3 replies on “A look at how Simulation works in Unity MARS”

btssays:

August 28, 2020 at 4:17 am • Reply
good job~



tetrissays:

August 17, 2020 at 7:00 am • Reply
Your application content and the system using the data do not need to know what device the data is coming from or whether the data is coming from a Simulation?

Andrew Msays:

September 4, 2020 at 7:12 am • Reply

Exactly – that is why simulation is so powerful