AR Simulation

by needle — tools for unity

> Build AR apps with confidence.
> Iterate fast, right in Editor.
> Non-invasive, drop-in solution.
> Fair pricing.

This package allows you to fly around in the Editor and test your AR app, without having to change any code or structure. Iterate faster, test out more ideas, build better apps.
ARSimulation is a custom XR backend, built on top of the XR plugin architecture.

AR Simulation implements an XR backend, just as ARCore on Android or ARKit on iOS do. Because of that architecture it does work in editor just as it would on device (this holds true for all supported features, please see Known Limitations section). It enables creators to develop AR apps with AR Foundation and reduces iteration time significantly: just hit play in editor for testing.

Note: for the most up-to-date documentation, please see AR Simulation on GitHub. Thank you!

Quick Start ⚡

1. If your scene is empty or not setup for AR just click `Tools/AR Simulation/Convert to Basic AR scene`
2. Click play
3. Press RMB (Right Mouse Button) + Use WASD to move around, LMB (Left Mouse Button) to click • touch • interact with your app

For more examples import the `Getting Started` samples from the Package Manager window.

Usage 📃

AR Simulation comes with a couple of built-in components that let you create and control tracked AR planes, pointclouds or images as well as the camera (your device).

AR Planes

- `SimulatedARPlane` is the most basic way to spawn a `ARPane`. Just add the component to a gameobject and position it in your scene. You can use local scale x and z for changing its size.
- `SimulatedARPlaneGeneration` is a more advanced component. It uses raycasts to sample points in your scene to generate planes dynamically. This is a closer representation of how planes get created on device.

AR Tracked Images

- `SimulatedARTrackedImage` can be used to simulate image tracking. You can track any image that is used in a `XRReferenceImageLibrary` asset. By default tracking automatically uses the camera frustum to update its `Tracking State`.

AR PointClouds

- `SimulatedARPointCloud` is the most basic way to spawn a `ARPointCloud`. By default is generates random points in either a spherical or a planar shape but it's also possible to edit points directly in editor or via code.
- `SimulatedARPointCloudRaycaster` uses raycasts to sample points in your scene.

AR Anchors

- Supported but we don't have a editor component implementation as with the other features right now. You should be able to spawn anchors with AR Foundation and see them being created just as on device.

Background Image (Experimental)

- `SimulatedAREnvironment` can be added to your root environment gameobject (the gameobject that contains objects that you want to render as a camera image). You can enable `IsActive` to assign itself to a `SimulatedAREnvironemntManager` which does the heavy lifting.
- `SimulatedAREnvironmentManager` can be added to a scene for handling camera background rendering. The `Scene Or Prefab` field can be used to reference either a scene asset, a prefab or a gameobject in the current scene to be rendered as a camera image.

Technical Details

Requirements

- Unity 2019.3 and above
- AR Foundation 3 and above
- Support for: Device Simulator
- Support for: New Input System / Legacy Input System / Both
- Support for: URP / Built-in

Known limitations

- Camera background is supported (with custom 3D scenes), but no occlusion support right now
- Environment cubemaps is platform-specific and currently not supported. Issue tracker
- No support for simulating faces, people, or collaboration right now (let us know if you feel this is important to you!)
- Partial support for meshing simulation (some support, but not identical to specific devices)
- Object tracking is not yet supported
- Touch input is single-touch for now, waiting for Unity to support it better (Device Simulator only supports single touch, since Input.SimulateTouch only supports one)
- There's a number of warnings around subsystem usage in Editor. They seem to not matter much but are annoying (and incorrect).

- Device Simulator disables Mouse input completely - we're working around that here but be aware when you try to create Android / iOS apps that also support mouse. Forum Thread
- in 2020.1 and 2020.2, even when you enable "New Input System", the Input System package is not installed in package manager. You have to install it manually. Forum Thread
- switching from a scene with Object Tracking to a scene with Image Tracking on device crashes Android apps (we'll report a bug soon)
- If your scene feels to dark / does not use environment lighting, make sure "Auto Generate" is on in Lighting Window or bake light data. (spherical harmonics simulation will only work if the shaders are aware that they should use it)
- AR Simulation currently has a dependency on XRLegacyInputHelpers that isn't needed in call cases; we will remove that dependency in a future release.

Contact ✍

Forum • Discord

needle — tools for unity • @NeedleTools • @marcel_wiessler • @hybridherbst • Say hi!

Documentation

Features

Zero Config

When you don't configure anything and don't add any objects to your scene, ARSimulation will by default look at the AR Foundation managers in the scene and prepare your scene when entering play mode accordingly. For example if your scene has an AR Plane Manager but neither a `SimulatedARPlane` nor a `SimulatedARPlaneGeneration` component in your scene it will create a `SimplateARPlane` in view of the camera. This behaviour can be disabled in `Project Settings/XR Plug-in Management/AR Simulation`

If you need more control over your scene, you can either set up geometry and use a `SimulatedARPlaneGeneration` component for automatically detecting planes (similar to what happens on device) or just add `SimulatedARPlane` components to your scene as necessary for testing.

AR Planes

- `SimulatedARPlane` is the most basic way to spawn a `ARPane`. Just add the component to a gameobject and position it in your scene. You can use local scale x and z for changing its size.
- `SimulatedARPlaneGeneration` is a more advanced component. It uses raycasts to sample points in your scene to generate planes dynamically. This is a closer representation of how planes get created on device.

Note: they don't have any geometry - they are purely generating XR SDK planes, which in turn generate the proper ARFoundation

AR Tracked Images

- `SimulatedARTrackedImage` can be used to simulate image tracking. You can track any image that is used in a `XRReferenceImageLibrary` asset. By default tracking automatically uses the camera frustum to update its `Tracking State`.

AR PointClouds

- `SimulatedARPointCloud` is the most basic way to spawn a `ARPointCloud`. By default is generates random points in either a spherical or a planar shape but it's also possible to edit points directly in editor or via code.
- `SimulatedARPointCloudRaycaster` uses raycasts to sample points in your scene.

AR Anchors

- Supported but we don't have a editor component implementation as with the other features right now. You should be able to spawn anchors with AR Foundation and see them being created just as on device.

Background Image (Experimental)

- `SimulatedAREnvironment` can be added to your root environment gameobject (the gameobject that contains objects that you want to render as a camera image). You can enable `IsActive` to assign itself to a `SimulatedAREnvironemntManager` which does the heavy lifting.
- `SimulatedAREnvironmentManager` can be added to a scene for handling camera background rendering. The `Scene Or Prefab` field can be used to reference either a scene asset, a prefab or a gameobject in the current scene to be rendered as a camera image.

Light Estimation (Experimental)

- A `SimulatedARCameraFrameDataProvider` component can be used to simulate some camera data and lighting information. You can reference a light component in `Input Light` wich can be used to control the light direction and intensity for spherical harmonics (it is recommended to disable this light and use it only as a control).

Render Pipelines

URP is supported. For camera background rendering we provide a renderer feature that works similarly to the AR Foundation background renderer feature.

Render Pipeline	
Built-in	✓ (Camera Background is Experimental)
URP	✓ (Camera Background is Experimental)
HDRP	✗ (Not tested)

Supported Configurations

Unity Version	**Input System**			**ARFoundation**		**Interaction Mode**	
	Old	Both	New	3.1	4.0	Game View	Device Simulator[1]
@ 2019.3/4	✓	✓	✓	✓	✓	✓	✓
@ 2020.1b	✓	✓	✓	✓	✓	✓	✓
@ 2020.2a	✓	✓	✓	✓	✓	✓	✓

Unity Version	**Render Pipeline**			**Platform**		
	Built-in	URP	HDRP[2]	Editor	iOS/Android Build[3]	Desktop Build[4]
@ 2019.3/4	✓	✓	—	✓	✓	untested
@ 2020.1b	✓	✓	—	✓	✓	untested
@ 2020.2a	✓	✓	—	✓	✓	untested

[1] Recommended. Feels very nice to use, and gives correct sizes for UI etc.

[2] HDRP is not supported by Unity on iOS/Android currently.

[3] "Support" here means: ARSimulation does not affect your builds, it is purely for Editor simulation.

[4] We haven't done as extensive testing as with the others yet. Making Desktop builds with ARSimulation is very useful for testing multiplayer scenarios without the need to deploy to multiple mobile devices.

XR Interaction Toolkit

Works out of the box. (If you have unity editor input that has multitouch, we recommend using LeanTouch)

Comparison between MARS and ARSimulation ⚔️

⚔️	**ARSimulation**	**MARS**
Claim	Non-invasive editor simulation backend for ARFoundation	Framework for simplified, flexible AR Authoring
Functionality	XR SDK plugin for Desktop: positional tracking simulation, touch input simulation, image tracking, ...	Wrapper around ARFoundation with added functionality: custom simulation window, object constraints and forces, editor simulation (including most of what ARSimulation can do), file system watchers, custom Editor handles, codegen, ...
Complexity	• 1 package • no additional files in project, only for XR SDK configuration • < 80 Types	• 6 packages • 5 new top-level folders in your project • > 800 Types and classes • 27 different ScriptableObjects with settings • 18 code-generated scripts with defines etc.
Changes to project	none	
Required changes	none	ARFoundation components need to be replaced with their MARS counterparts

The following table compares ARSimulation and MARS in respect to in-editor simulation for features available in ARFoundation.
Note that MARS has a lot of additional tools and features (functionality injection, proxies, recordings, automatic placement of objects, constraints, ...) not mentioned here that might be relevant to your usecase.
See the MARS docs for additional featuers.

⚔️	**ARSimulation** ***Simulation Features***	**MARS** ***Simulation Features***
Plane Tracking	✔	✔
Touch Input	✔	✖ [1]
Simulated Environments	(✔)[2]	✔
Device Simulator	✔	✖ [3]
Point Clouds	✔	✔
Image Tracking	✔	✔

	ARSimulation *Simulation Features*	**MARS** *Simulation Features*
Light Estimation Spherical Harmonics	✔	✕
Anchors	✔	✕
Meshing	(✔)	✔
Face Tracking	✕	(✔)[4]
Object Tracking	✕	✕
Human Segmentation	✕	✕

Revision History

Date	Reason
2020-06-27	Added forum link, sub documents and prepared PDF
2020-06-22	Updated documentation
2020-06-01	Initial public release