Imagine WebAR World Tracker

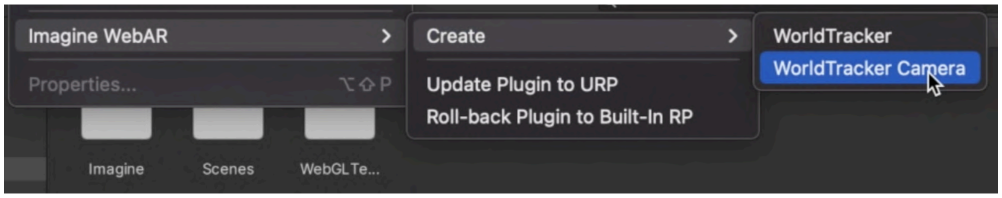
Version:	1.5.0
WebGL Demo:	https://webar.imaginerealities.com.au/wt/demo
Contact Support:	https://imaginerealities.com.au/contact-support
Publisher's Website:	https://imaginerealities.com.au
Unity Forums Thread:	https://forum.unity.com/threads/released-imagine-webar-world-tracker-for-unity-webgl.1451896/

Discord Community:	https://discord.gg/ypNARJJEbB

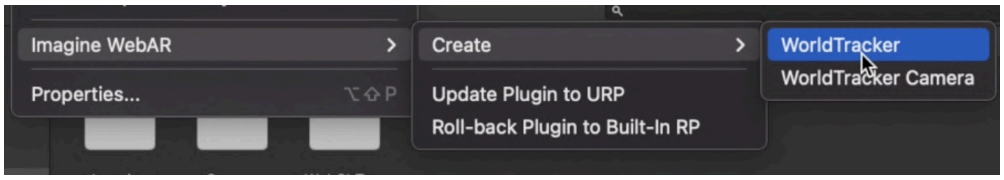
Setting up your AR Scene in Unity

A simple AR scene can be set up in a few minutes.

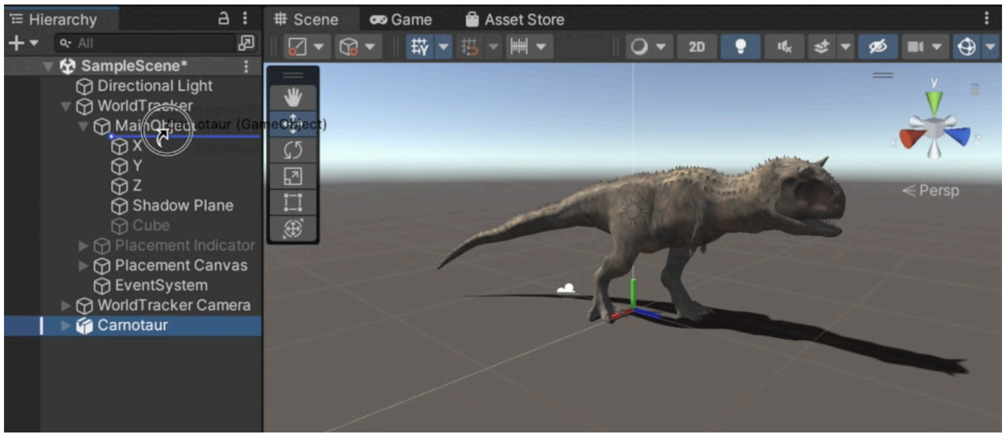
1.) Import the plugin and create a new Scene in Unity.
2.) Create a **World Tracker** from **Assets>Imagine WebAR>Create>World Tracker**



3.) Delete the Main Camera gameobject, and create a **World Tracker Camera** from **Assets>Imagine WebAR>Create>World Tracker Camera.** Drag this object into the Tracker Cam property of the World Tracker

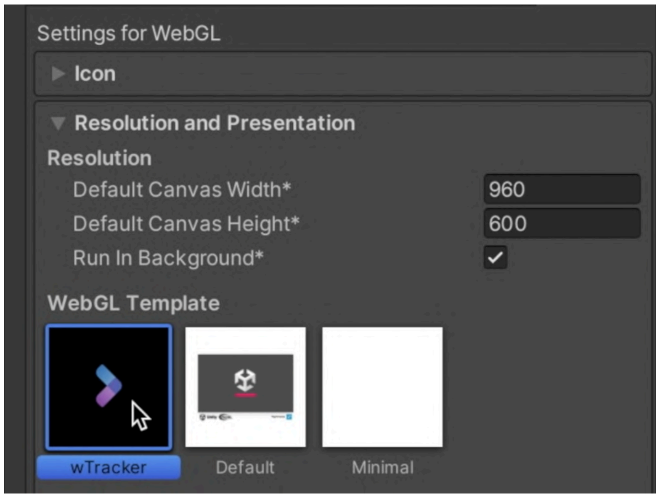


4.) Now we are ready to add our AR Content. Drag your 3D models inside **World Tracker>MainObject.**



Building your AR Scene

1.) Go to **File>Build Settings,** switch to the **WebGL** platform if needed and add your scene in the build. Then select **wTracker** template in **Player Settings>Resolution and Presentation>WebGL Template**



Settings for WebGL

Icon

Resolution and Presentation

Resolution

Default Canvas Width*	960
Default Canvas Height*	600
Run In Background*	✓

WebGL Template

wTracker Default Minimal

2.) Build your project.

3.) Upload your build in your preferred hosting site (eg. Google Firebase or AWS). Make sure to enable https/ssl (otherwise, the webcam will not be started)

4.) Open your URL in your mobile device. Allow access to your device camera and motion sensors. You should now see the game objects anchored in your environment.

Tracker Settings

Mode:

The tracker supports 3 different modes: 6DOF, 3DOF and 3DOF Orbit

Plane Mode

Horizontal

Default mode for tracking ground and horizontal surfaces

Vertical (Experimental)

Experimental mode for tracking walls and vertical surfaces. In this case, the user must need to face the wall squarely before placement. Otherwise, the main object will be skewed.

6DOF Mode

The 6DOF Mode allows camera movement along X-Y-Z (position) and Yaw-Pitch-Roll (rotation).

6DOF > DepthMode

SCALE_AS_DEPTH - this depth-mode emulates the z camera distance by scaling the 3D object. This allows for very accurate representation of depth comparable to SLAM in ideal scenarios. The downside is that trail renderers, particles and physics will be affected by the scale changes.

~~**Z_AS_DEPTH_EXPERIMENTAL** - this depth-mode approximates the actual depth using computer-vision. This is still a work in progress and may only perform well on a few use cases.~~

6DOF > Max Pixels

Using higher values can increase the tracker accuracy at the expense of framerate.

6DOF > Max Points
Using higher values can increase the tracker accuracy at the expense of framerate.

~~6DOF > Target Confidence~~
~~Minimum threshold where 6DOF falls back to 3DOF tracking~~

6DOF > Clamp Pixel Drift
Maximum allowable translation change in one frame. Values beyond this threshold will be treated as invalid. We don’t recommend changing this parameter.

6DOF > Clamp Scale Drift
Maximum allowable scale change in one frame. Values beyond this threshold will be treated as invalid. We don’t recommend changing this parameter.

6DOF > Min Point Quality
Minimum similarity threshold when tracking points. Higher values will result in better drift resistance but will be less robust to high motion. We don’t recommend changing this parameter.

6DOF > Min Good Point Quality
Minimum similarity threshold when tracking points in low confidence scenarios. Lower values will result in better drift resistance in low confidence scenarios, but with less chances to trigger the 3DOF fallback. We don’t recommend changing this parameter.

6DOF > Min Coplanar Factor
Threshold for treating points as part of the major ground/wall plane. Higher values will allow more points and more stable tracking but with less precision (floatiness). Lower values will be more precise but with less stability to high motion.

6DOF > Angle Smooth Factor
Strength of the smoothness filter applied for angle readings.

6DOF > Angle Drift Threshold
When the difference between smoothened angle value and actual angle value exceeds this threshold, then we reuse the actual value instead.

6DOF > Pose Correction Enabled
Enable pose correction when the original placement area has been recognized.

6DOF > Pose Correction Interval
The time interval in between pose correction checks.

3DOF Mode

The 3DOF Mode only allows camera rotations (Yaw-Pitch-Roll). This mode is desired in some experiences such as “Stationary or Look-Around AR”. And unlike the 6DOF Mode, this mode is not affected by drift. Also, select this mode when you are using Geolocation

3DOF > Arm Length:
The Arm Length property is the approximate distance (in meters) of the camera to the vertical axis of rotation. Eg. A person holding a smartphone has an arm length of 0.3-0.5 meters.

3DOF > Use Extra Smoothing:
Enable this to remove jitter/noise from motion sensors.

3DOF > Smoothen Factor:
Higher values will be smoother, but will be slower to converge to the correct position (“lerping-effect”).

3DOF > Angle Smooth Factor
Strength of the smoothness filter applied for angle readings. This filter is done prior to the extra smoothing filter in Unity.

3DOF > Angle Drift Threshold
When the difference between smoothened angle value and actual angle value exceeds this threshold, then we reuse the actual value instead.

3DOF Orbit Mode

Use the 3DOF Orbit Mode as an interactive way to present your 3D models. In this mode, the camera “orbits” around the object based on motion sensor readings.

3DOF Orbit Mode > Orbit Distance:
The distance between the camera and the center of the orbit

3DOF Orbit Mode > Smoothen Factor:
Orbit mode uses extra smoothing by default. Higher values will be smoother, but will be slower to converge to the correct position (“lerping-effect”).

3DOF Orbit Mode > Angle Smooth Factor
Strength of the smoothness filter applied for angle readings. This filter is done prior to the extra smoothing filter in Unity

3DOF Orbit Mode > Angle Drift Threshold
When the difference between smoothened angle value and actual angle value exceeds this threshold, then we reuse the actual value instead.

3DOF Orbit Mode > Center Transform:
The transform where the camera orbits around

3DOF Orbit Mode > Swipe To Rotate:
Enable this flag to allow rotation around the y-axis on user swipe gestures

3DOF Orbit Mode > Swipe Sensitivity:
The rotation speed/sensitivity when user is swiping

3DOF Orbit Mode >Pinch To Scale:
Enable this flag to allow zooming-in/out when user does the pinch gesture

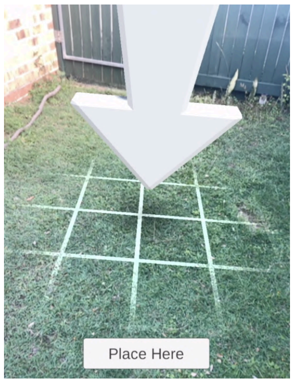
3DOF Orbit Mode >Min Dist/Max Dist:
The minimum and maximum distances allowed when the user zooms in/out of the object

MainObject
The gameObject to be placed in AR

Camera Start Height
The distance (in meters) of the AR Camera to the ground.

Use Compass (Experimental)
Uses the absolute angles with respect to the device's compass north. (when available). Make sure to enable this when using Geolocation.

Use Placement Indicator
This property will allow the user to place the object manually before starting the experience. If this is unchecked, the object will be automatically placed upon scene initialization.



<u>Placement Indicator Settings > MinZ</u>
The nearest distance where your Main Object can be placed.

<u>Placement Indicator Settings > MaxZ</u>
The furthest distance where your Main Object can be placed.

<u>Placement Indicator</u>
The gameObject for the arrow that will indicate where your Main Object will be placed in the ground.

<u>OnPlacementIndicatorShown</u>
Unity event that is called when the device is pointed to a valid spot in the ground (determined by MinZ and MaxZ).

<u>OnPlacementIndicatorHidden</u>
Unity event that is called when the device is pointed to an invalid spot in the ground (determined by MinZ and MaxZ).

<u>OnPlacedOrigin</u>
Unity event that is called when the user places the MainObject to the ground.

<u>OnResetOrigin</u>
Unity event that is called when the user resets the placement of the MainObject.

<u>ShowGameObjectsWhenPlaced</u>
GameObjects in this list are deactivated during placement phase, and will be activated when MainObject is placed.

<u>ShowGameObjectsWhenReset</u>
GameObjects in this list are activated during placement phase, and will be deactivated when MainObject is placed

Geolocation (Experimental)

This feature allows you to create **GPS Pins** to anchor Gameobjects to GPS coordinates which can be explored by users if they are in the vicinity. This feature only works with 3DOF mode, and make sure to enable *UseCompass*. Note that the accuracy of this feature is highly dependent on GPS signal strength, hence it is not recommended to use this on indoor experiences. The accuracy is around 5-20m when outdoors, and 20-50m when indoors (or in places where GPS signal is obstructed).

Geolocation Settings > Activation Radius
The maximum distance where GPS Pins are shown.

Geolocation Settings > Pin Radius
The bounding radius of a GPS Pin.

GPS Pin > Name
A descriptive name for your GPS Pin

GPS Pin > Transform
The transform that will be re-positioned relative to the camera, based on the GPS reading

GPS Pin > Latitude/Longitude/Altitude
The coordinates of the pin in the real world. Since altitude is very inaccurate in most device sensors, it is recommended to set altitude=0.

Geolocation Settings > OnGPSPositionUpdated
UnityEvent that is called when a new GPS position reading is received from the world tracker

Geolocation Settings > OnGPSPinInRange
UnityEvent that is called when a GPS Pin goes inside the activation radius

Geolocation Settings > OnGPSPinOutOfRange
UnityEvent that is called when a GPS Pin goes outside the activation radius

Geolocation Settings > OnAllGPSPinsOutOfRange
UnityEvent that is called when all GPS Pins goes outside the activation radius

Geolocation Settings > OnEnterGPSPin
UnityEvent that is called when the user goes inside a GPS Pin's pin radius

Geolocation Settings > OnExitGPSPin
UnityEvent that is called when the user goes outside a GPS Pin's pin radius

Geolocation Settings > Debug Start Lat/Lon
Use these for testing your GPS Pins from within the editor. Input your preferred coordinates, and you can walk around using the W,A,S,D keys

Setting up your Geolocation Scene

Follow the steps below to set up your geolocation scene, such as a location-based game.

1.) Drag-in a new **WorldTracker - Geolocation** prefab in your scene.
2.) Delete your default scene camera and drag-in a new **ARCamera** prefab in your scene. Then drag it in the **Tracker Camera** property of your WorldTracker.
3.) Create a gameObject for your GPS Pin and parent it inside WorldTracker>MainObject. Then add a **GPS Pin** script.
4.) Give your new GPS Pin a unique id, and set the latitude and longitude (recommended altitude = 0, due to inaccuracy of this reading in most devices).
5.) Add more GPS pins as needed.
6.) Build, upload and test your project.

ARCamera Settings

Video Plane Modes

- **NONE** - does not render a video plane inside Unity. Instead, the unity canvas is transparent and the camera image is rendered in javascript. This is the fastest video plane mode, but as a tradeoff, it can have rendering issues with transparency, and does not support Post Processing and HDR. Capturing screenshots while using this mode results in a black camera feed.
- **TEXTURE_PTR** - renders a video plane inside Unity to support Post Processing, HDR, and screenshot capture. This mode can have a slight performance impact. This mode is not available in the free version
- **DATAURLTEXTURE (DEPRECATED)** - renders a video plane inside Unity to support Post Processing, HDR, and screenshot capture. This mode uses dataURL textures to pass image data from javascript to Unity and can be very slow, especially on older devices.
- **WEBCAMTEXTURE (DEPRECATED)** - renders a video plane inside Unity to support Post Processing, HDR, and screenshot capture. This mode uses Unity's webcamtexture to get the image data. This method is unreliable and may have issues on some devices/browsers/OS versions.

Video Plane Mat

- The material to be used to render your video plane

Video Distance

- The distance between the ARCamera and the video plane

Unpause Pause On Enable Disable:

- If this is enabled, UnpauseCamera() and PauseCamera() are automatically called when ARCamera gameObject is activated or deactivated

Pause On Destroy:

- If this is enabled, PauseCamera() is automatically called when ARCamera gameObject or scene is destroyed

Pause On Application Lose Focus:

- If this is enabled, PauseCamera() is automatically called when the browser tab goes to the background. This is useful to keep the webcam alive when switching tabs/apps while the experience is running.

OnResized:

- Unity Event called when video camera dimensions are resized. This gets called when device orientation changes. Note that resizing the browser window does not necessarily trigger this event.

Resize Delay:

- The delay between resize events, where the ARCamera starts re-computing for the new size. Later versions of iOS (17) has a relatively long resize delay (bug) which hinders resize events in Unity. If so, try using a longer resize delay.

OnCameraImageFlipped:

- This gets called when the flipping/toggling the camera from front to back or vice versa is successful (using ARCamera.FlipCamera function).

OnCameraOrientationChanged:

- This gets called when the device is rotated from portrait to landscape, or vice versa.

SwipeToRotate

Use this script to allow players to rotate your MainObject by swiping across the screen.

PinchToScale
Use this script to allow players to scale your MainObject by doing pinch gestures

TwoFingerPan
Use this script to allow players to move your object by dragging on the screen with 2 fingers. Note that in 6DOF mode, if your MainObject is dragged too far from the origin, the tracking drift may become more prominent.

TapToPlace
Use this script to allow players to move your object by tapping. Note that in 6DOF mode, if your MainObject is placed too far from the origin, the tracking drift may become more prominent.

FAQ and General Questions

Camera does not open when I host in my website

- Make sure you are hosting on your server with https enabled. Otherwise, access to the webcam will be blocked due to security reasons.
- Check your **Player Settings>Resolution and Presentation** and make sure that you have selected the **wTracker** WebGL template.

Unity loading bar is stuck at 90%

- This is usually caused by your WebGL compression. You can set **Player Settings>Publishing Settings>Compression Format** to **Disabled.** You can also compress your build but you have to ensure that gzip(.gz) or brotli(.br) is enabled in your hosting server.

Read more FAQs in our Discord channel

Known Issues:

Visit the #bug-reports channel in our discord

Change Notes:

Version 1.5.0
• Major tracking improvement. We have reworked our 6DOF tracking algorithm which significantly improves tracking quality and resistance to drift! This is a must-have update if you are working on 6DOF experiences
• [ADDED] Pose-recovery when original placement area has been recognized (in 6DOF)

• [ADDED] Tracking confidence values for 6DOF. Determined by the availability and reliability of visible keypoints
• [ADDED] 6DOF fallbacks to 3DOF when tracking confidence falls down to a minimum threshold
• [ADDED] Tap-to-reposition the MainObject during 6DOF tracking (Alternative to Two-Finger-Pan)
• [ADDED] Experimental Vertical/Wall tracking feature. User needs to stand squarely in front of the wall before placement, otherwise, there will be some skewing issues.
• [FIXED] Issue where camera flickers white when switching orientation or resizing the window. This should also fix the case when tracker stops working and needs to be restarted after a resize event
• [ADDED] Debug functionality to reposition the GPS Pins in the Geolocation demo for testing purposes
• [FIXED] GPS Pins still being stuck at the origin during GPS initialization
• [ADDED] Experimental demo experience for 6DOF portal. Still drifting when trying to walk past the origin and/or turning around. But good enough for "peek through" experiences.
• [ADDED] Experimental support for new Input System (For debug and swipe/pan/pinch gesture controls)
• [FIXED] Missing VideoPlane material in ARCamera prefab

Version 1.4.11
• [FIXED] Race-condition bug where GPS Pins are stuck near the camera
• Refactored Geolocation GPS Pins class outside WorldTracker for easier customisation
• [ADDED] OnEnterGPSPin/OnExitGPSPin events and pinRadius property
• [FIXED] Chroma Key shader not showing on iOS 17 devices
• [FIXED] Webcam not displaying properly after autorotation when using ARCamera>TexturePtr on iOS 17 devices
• [ADDED] ARCameraGlobalSettings which allows you to choose Front/Back Camera Facing modes. Note: 6DOF is not supported on front camera
• [FIXED] Bug where auto-placed 3DOF rotate undesirably towards the North even if useCompass property is disabled
• [FIXED] Bug where 6DOF and 3DOF experiences rotate in the y-axis after the device moves quickly
• [FIXED] Bug where GPS pins do not lerp into the correct position in WebGL builds
• [FIXED] Compass-related log spams in iOS devices

Version 1.4.0
• [FIXED] Bug where frame rate is significantly lower on high-resolution devices. Also boosted overall performance and framerate in general across all devices
• [ADDED] Experimental Geolocation Feature that works in tandem with 3DOF mode. You can now anchor GameObjects to GPS coordinates which users can explore when they are in the vicinity
• [ADDED] Experimental Compass Feature which is required in Geolocation
• [ADDED] Experimental 3DOF Orbit Mode
• [ADDED] UseExtraSmoothing property for 3DOF mode to smoothen out motion sensor noise
• [ADDED] Experimental Interoperability with Image Tracker. Can now be used in the same project with Image Tracker version 1.7.0

• [FIXED] Material issue where Chroma videos tend to get cropped out in iOS 17.
• [FIXED] Potential bug where camera feed turns dark/black

Version 1.3.1
• [FIXED] Race condition bug, where camera feed is white when the user immediately opens the webcam, right after allowing camera permissions
• Other minor improvements

Version 1.3.0
• [ADDED] TexturePointer VideoBackground mode for AR Camera. This supports PostProcessing and HDR for both URP and BuiltIn RP.
• [ADDED] Non-AR landing scene. ARCamera is now being started from within Unity (instead of index.html).
• ARCamera is now separated from wTracker. This opens up the possibility of interoperability between other Imagine Trackers.
• [FIXED] Bug where camera is zoomed in some browsers, when Accessibility>Zoom is enabled in the mobile device
• [FIXED] SyncVideoSound.cs - Black texture when videoplayer is still loading

Version 1.2.2
• [ADDED] Experimental screenshot capture functionality
• [ADDED] Y-axis billboarding mode for Billboard.cs
• [FIXED] WorldTracker prefab setup with Pinch-To-Scale
• [FIXED] Debug Camera movement and rotation sensitivity in Editor

Version 1.2.1
• [FIXED] Fixed bug where tracker may stop working after the site tab is placed into the background
• [FIXED] Console log spams when tracking
• [ADDED] Feature to pause/unpause the camera (3DOF will still run even when camera is paused)

Version 1.2.0
• [FIXED] Fixed bug which contributes a major drift in 6DOF
• [FIXED] Fixed a major memory leak in 3DOF that causes crashes after a few minutes
• [FIXED] Fixed several minor memory leaks in 6DOF
• [ADDED] Support in switching between 6DOF and 3DOF
• [ADDED] 3DOF VFX demo scene
• Major refactoring and overall improvement of WorldTracker.cs
• Minor optimizations for 6DOF
• Placement phase is now using 3DOF calculations which boosts the frame rate
• [ADDED] Experimental Screenshot Feature

Version 1.1.0
• [FIXED] Fixed bug where placed object does not always face the AR camera
• [FIXED] Fixed bug where UI becomes unresponsive when the url bar is hidden in Safari

• [FIXED] Fixed bug where UI becomes unresponsive when permission screen is displayed in Brave
• [FIXED] Fixed bug where UI is scaled up in Samsung Internet Browser
• [ADDED] Handling when user denies camera and/or motion sensor permission
• [ADDED] Main landing page for demos
• [ADDED] Sample chroma video experience
• Exposed webcam initialization calls and improved html template
• Breaking Changes: app.js is replaced with wtracker.js. Refactoring needed for custom index.html templates from 1.0.1 and below.

Version 1.0.1
• [FIXED] Fixed a JSON parsing bug causing the tracker to behave erratically on some countries and regions
• [FIXED] Fixed object getting skewed when opening the experience
• Minor UX Improvement for WorldTracker Inspector
• Added **OnResetOrigin** and **OnPlaceOrigin** event delegates
• Added Lists for **ShowObjectsOnPlace** and **ShowObjectsOnReset** gameObjects
• Added ARCamera class to support Experimental **WebcamTexture** and **DataURLTexture** video backgrounds
• Added API's to start and stop the WorldTracker

Version 1.0.0
• First release