Imagine WebAR Face Tracker

Version:	1.1.1
WebGL Demos:	https://webar.imaginerealities.com.au/ft/demo
Contact Support:	https://imaginerealities.com.au/contact-support
Publisher's Website:	https://imaginerealities.com.au
Unity Forums Thread:	Coming soon
Discord Community:	https://discord.gg/ypNARJJEbB

WebGL Introduction

The web is one of the most promising platforms for augmented reality, because it allows users to experience AR without the need to download a standalone application. And most, if not all, WebAR plugins for Unity require developers to pay on a subscription and/or per view basis.

Imagine WebAR Image Tracker is an augmented reality plugin for Unity WebGL which allows developers to implement AR experiences for the web. This plugin also allows developers to host their own AR experiences like any other Unity WebGL build.

WebGL applications built using this plugin are able to run on both desktop and mobile browsers. And since AR is mostly experienced through mobile devices, we are giving mobile browsers a higher priority.

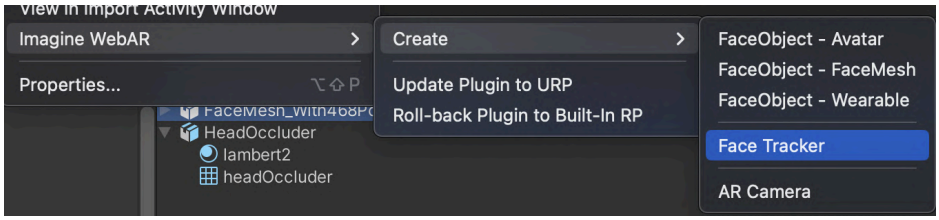
Render Pipelines supported are Built-In RP and URP. Though, some rendering features are not supported (see **Current Limitations**) or still experimental.

The plugin consists mainly of two modules: [1] a computer-vision (CV) module, written in Javascript, which uses the MediaPipe Face Tracking solution. [2] A Unity Editor module which provides the tools necessary to create your Face tracking AR Scene in Unity.

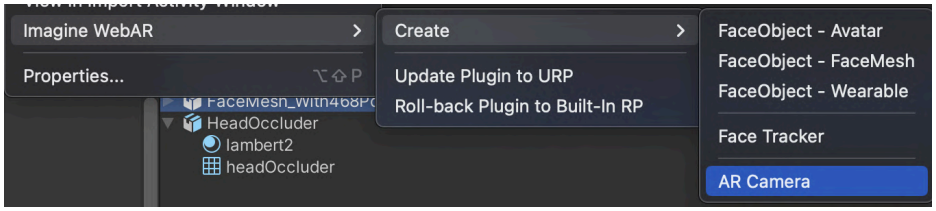
Setting up your AR Scene in Unity

A simple AR scene can be set up in a few minutes. Tutorial video can be found here:
To set up your first scene:

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



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



4.) Now we are ready to add **Face Objects** to our tracker.

Adding Face Objects to your Tracker

When the tracker detects a face, it overlays a corresponding faceObject in front of the camera.
This can be a virtual wearable, face mask, avatar, or a mix of everything. You can have at most 4 faceObjects in your scene. Note that the more faces you track simultaneously, the more performance intensive your experience becomes.

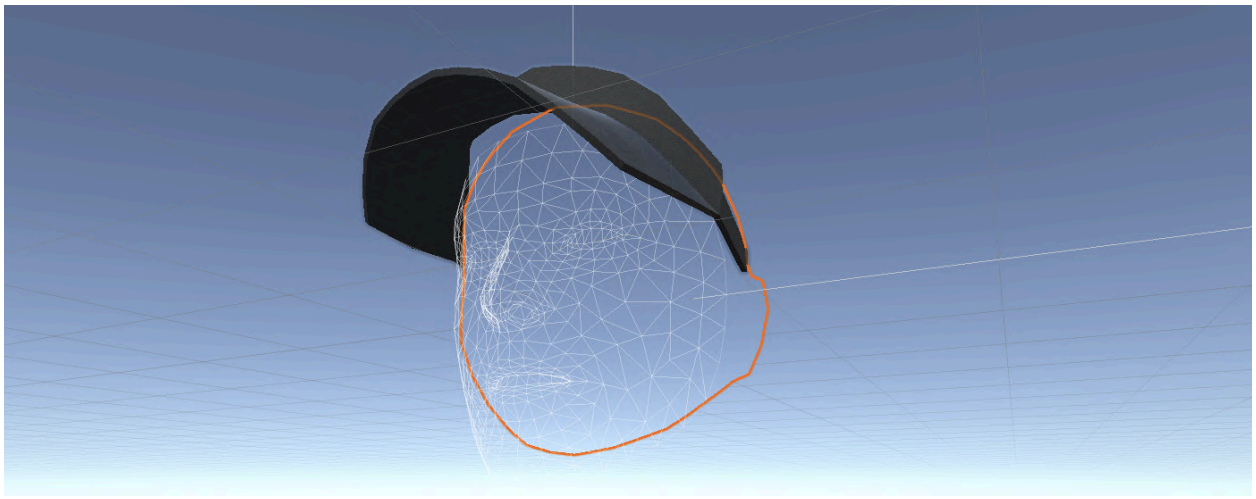
To add a FaceObject, you can use our predefined templates or create your own. FaceObjects should have a unique **Face Index** (0-3) for each of the 4 players.

Face Object - Wearable

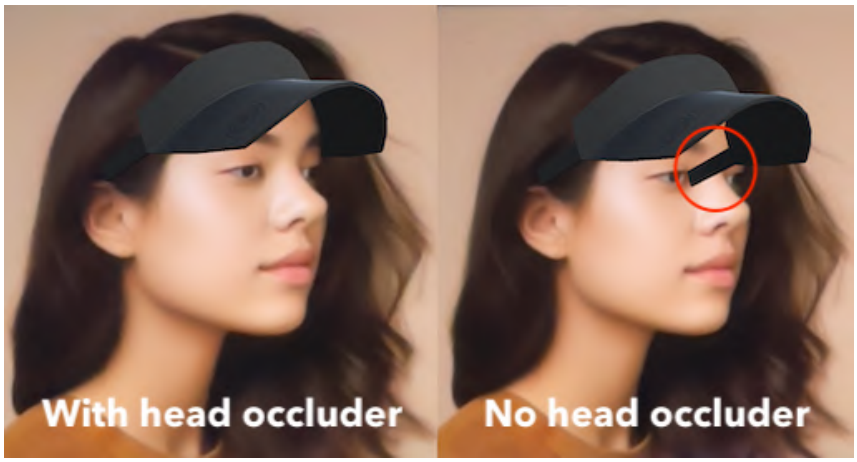
Virtual wearables and head decorations are the simplest to implement because we are simply anchoring 3D models to the detected face. Simply parent your gameObjects in your FaceObject

To add a wearable face object template, simply go to **Assets>Imagine WebAR>Create>Face Object - Wearable.** Then drag in your 3D models as a child of the faceObject. Scale, rotate and position your objects as they would naturally look when worn.

Head Occluder

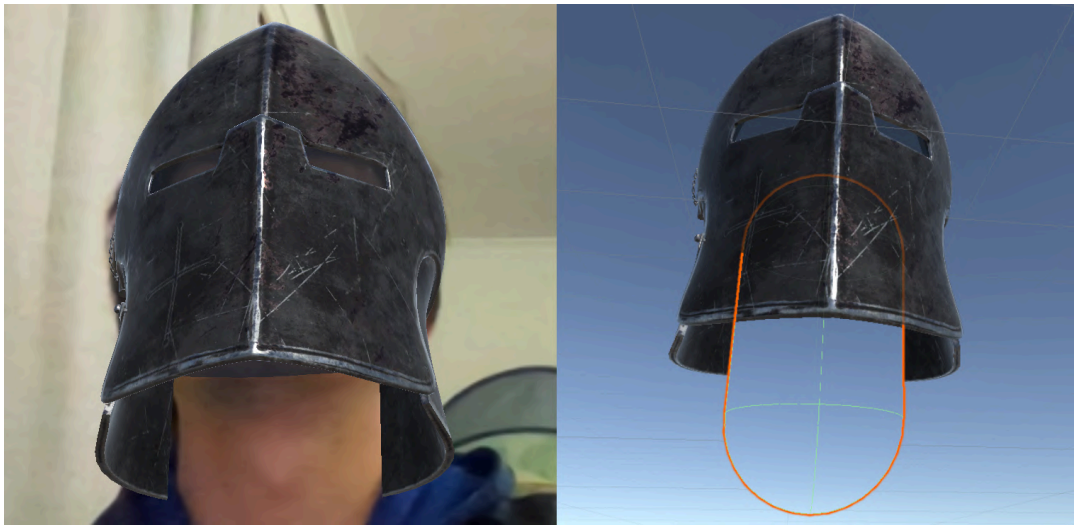


The head occluder will mask out 3D model parts that are normally occluded by the head. The head occluder material uses the Imagine/DepthMask shader.



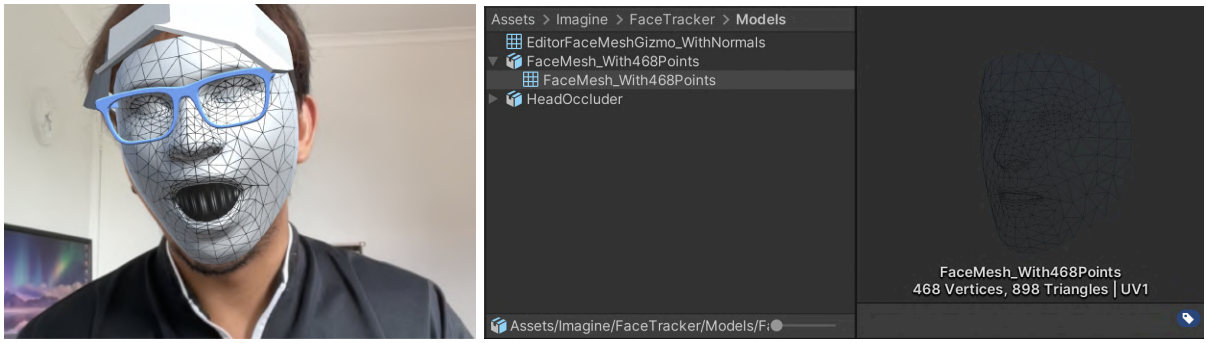
Note: To mask out transparent objects without any issues, you will need to use ARCamera>VideoPlaneMode>TexturePtr in tandem with the head occluder.

This concept can be further applied to occlude other parts of the body as needed. The example below shows a neck occluder.

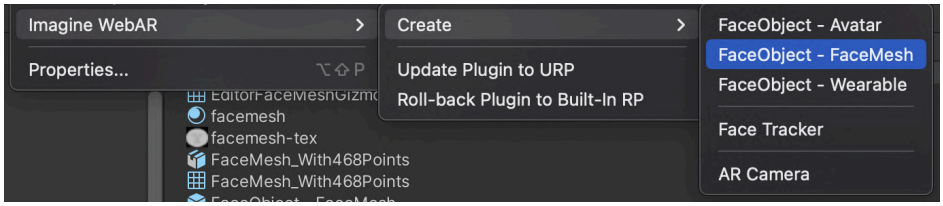


Face Object - FaceMesh

Some face tracking experiences require a mesh that deforms with the player's face. These experiences include various face paints and face masks. This functionality only works with our custom-built face mesh model (Assets/Imagine/FaceTracker/Models/FaceMesh_With468Points.fbx) which has exactly 468 matching that of the face tracker model.

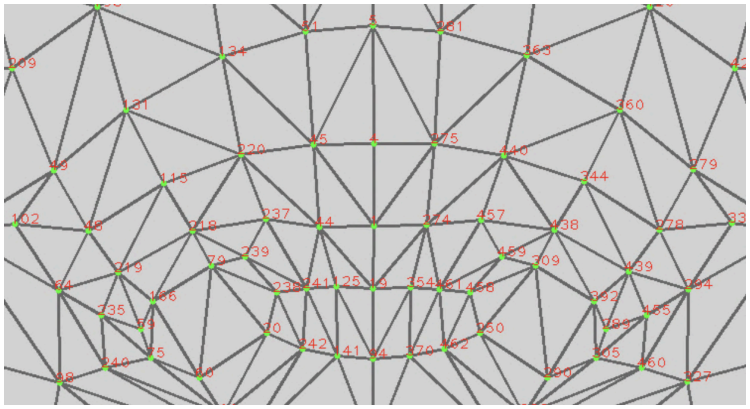


To add a face mesh object template, simply go to **Assets>Imagine WebAR>Create>Face Object - FaceMesh.** Then replace the material's texture as needed.



The best approach is to duplicate and paint over the provided face mesh UV image in Assets/Imagine/FaceTracker/Textures/facemesh-tex.png

Note that the face mesh feature can be more performance intensive compared to other features as it computes for the position of 465 vertices every frame. Thus it's not recommended to have more than one faceObject using the face mesh, otherwise, the experience can slow down older devices.



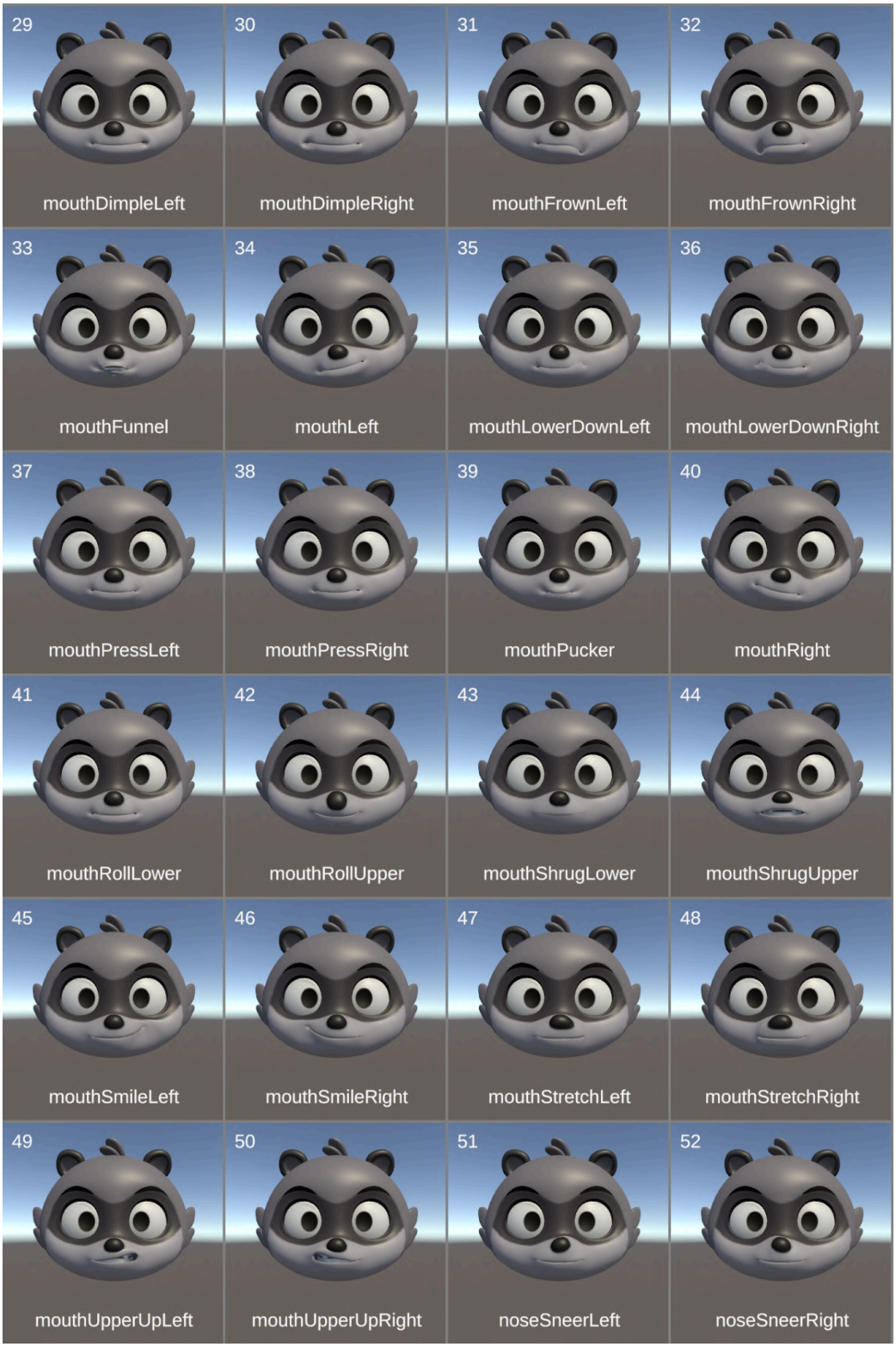
To map the exact location of each vertex index from the mesh, download [Mediapipe's face landmark vertices mapping here]

Face Object - Avatar

A popular use case of the face tracker is to animate avatars. The plugins allow you to synchronize your custom avatar's blendshapes to the blendshapes detected by the face tracker thus seamlessly animating facial expressions in real-time.

This can all be done in the **ShapesMap** property under the FaceTracker's Blendshapes Settings. You can utilize up to 52 different blendshapes from Mediapipe to animate your custom avatar.


1
_neutral
2
browDownLeft
3
browDownRight
4
browInnerUp
5
browOuterUpLeft
6
browOuterUpRight
7
cheekPuff
8
cheekSquintLeft
9
cheekSquintRight
10
eyeBlinkLeft
11
eyeBlinkRight
12
eyeLookDownLeft
13
eyeLookDownRight
14
eyeLookInLeft
15
eyeLookInRight
16
eyeLookOutLeft
17
eyeLookOutRight
18
eyeLookUpLeft
19
eyeLookUpRight
20
eyeSquintLeft
21
eyeSquintRight
22
eyeWideLeft
23
eyeWideRight
24
jawForward
25
jawLeft
26
jawOpen
27
jawRight
28
mouthClose



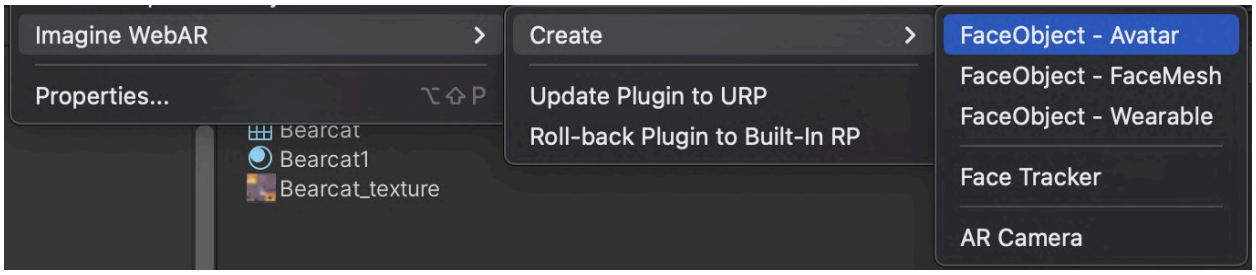
Notes:

- Link to download Mediapipe's racoon head model: https://assets.codepen.io/9177687/raccoon_head.glb
- *mouthClose* is extrapolated to cancel-out *jawOpen.* You don't need to follow mediapipe's example here.

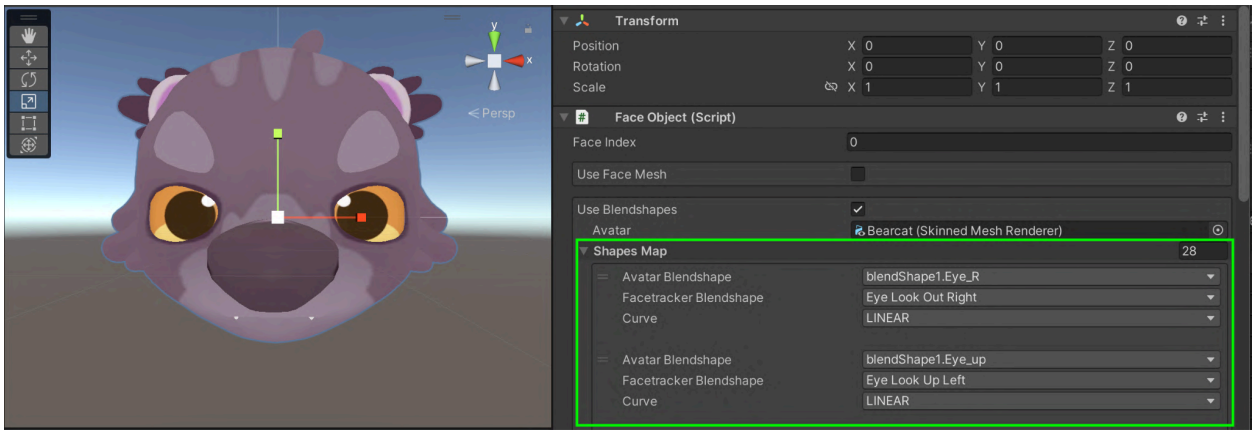
Custom Avatar Blendshapes

Mapping blendshapes to your custom avatar is straightforward. You don't need to implement all the 52 blendshapes from mediapipe, and your avatar model can have less than those. Follow the steps below to setup your custom avatar's blendshapes:

1.) Create a faceObject avatar from our template by going to **Assets>Imagine WebAR>Create>Face Object - Avatar.** Or you can optionally setup your own by creating an empty gameObject and adding a **FaceObject** script and enabling **Use Blendshapes** flag.



2.) Drag in your custom avatar face model in the scene (or optionally use our bearcat avatar). Then drag this gameObject in the **Avatar** property of the FaceObject.

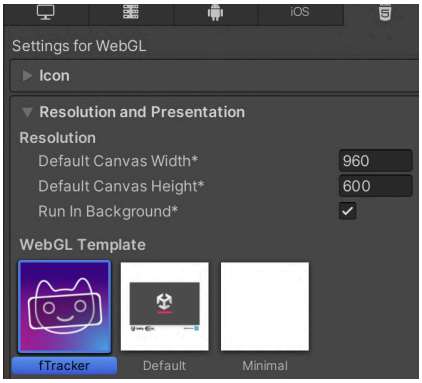


3.) Add items in the **Shapes Map** property of the FaceObject. Each item will correspond to one blendshape on your custom avatar. Match the correct blendshapes by setting **Avatar Blendshape** and selecting its corresponding **FaceTracker Blendshape**. You can optionally set the **Curve** property to control the "responsiveness" of your blendshapes.

Building your AR Scene

After setting up all your faceObjects, you can now build 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 **fTracker** template in **Player Settings>Resolution and Presentation>WebGL Template**



2.) Build and Run your project.

3.) Allow access to your device camera and stand in front of the webcam. You should see the face objects anchored in your face in AR.

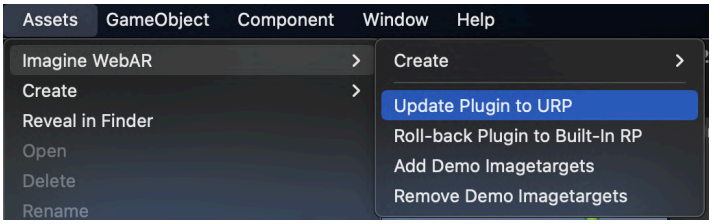
Updating the plugin to URP

Imagine WebAR also supports the Universal Render Pipeline for WebGL. To update the plugin to URP, follow the steps below:

1.) Create a new Unity project using the **3D(URP)** template.

2.) Import the plugin.

3.) Go to **Assets>Imagine WebAR>Update Plugin to URP**



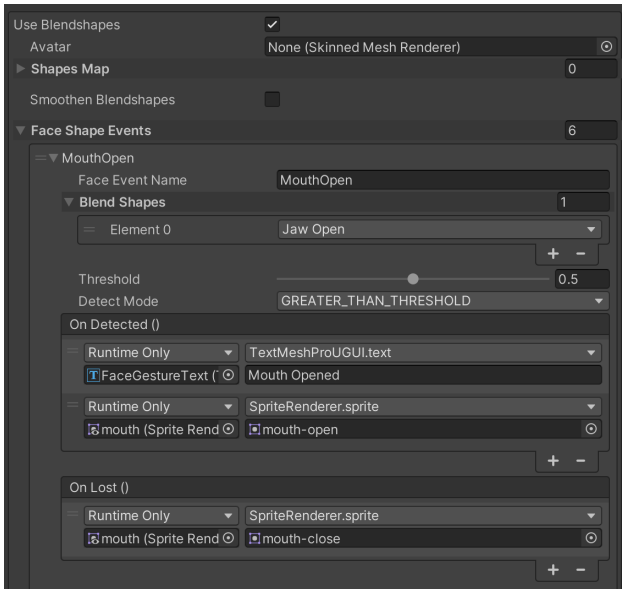
4.) Wait for the reimport to finish.

Note: Failure to do this step when building in URP can cause black-screen or other camera issues.

Face Events

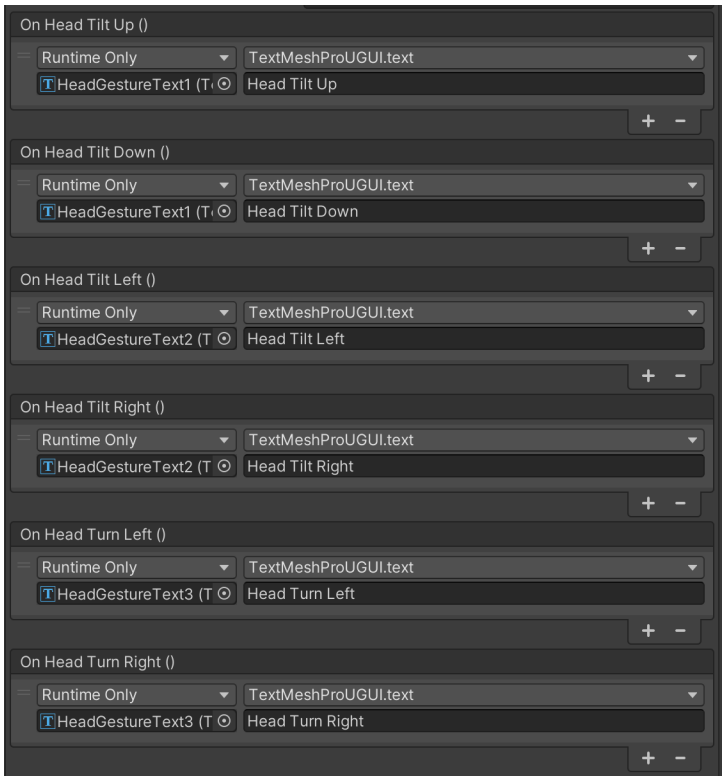
You can set up face events in the FaceTracker to quickly catch user gestures such as MouthOpen, EyeBlink etc. You can even combine 2 gestures to get an average gesture value such as: MouthSmileLeft and MouthSmileRight, to get a MouthSmile. Try the steps below as an example:

1.) Create a new scene and create a *FaceTracker*, *ARCamera* and *FaceObject*.
2.) Select your FaceObject and enable **Use Blendshapes** in the inspector
3.) Under **Face Shape Events**, create a new element, and set **FaceEventName** to “Mouth Open”
4.) Under **Blendshapes**, add a new element and select **Jaw Open**.
5.) Set **Threshold** to 0.5 and **Detect Mode** to **Greater than Threshold**. This will allow the face event to trigger once the jaw open blendshape value goes over your threshold.
6.) Then add a new event in **OnDetected** and map it to your script function.
7.) You can also add a catch the **OnLost** event and map it to another script function.

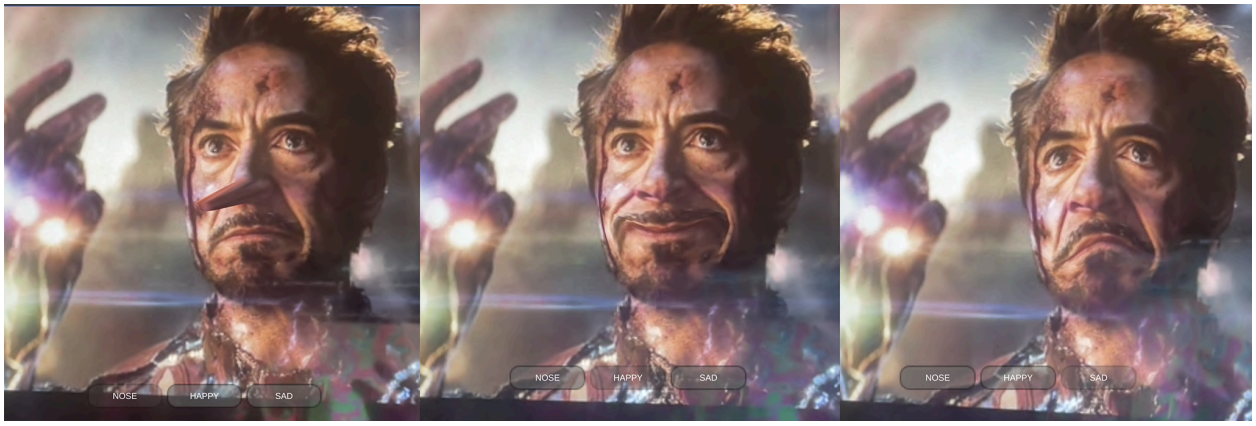


To combine blendshapes (like MouthSmileLeft and MouthSmileRight, to get a MouthSmile), simply select more than one blendshape in the **Blendshapes** list.

To use head angles, such as controlling a character in a game using head tilt, you can use **HeadAngleEvents** where you can detect head tilt-up, tilt-down, tilt-left, tilt-right, turn-left, turn-right.

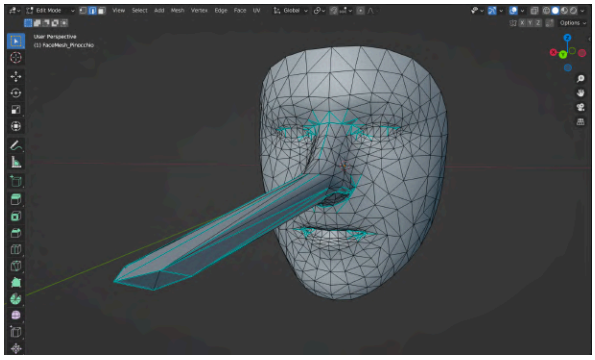


Face Deformation

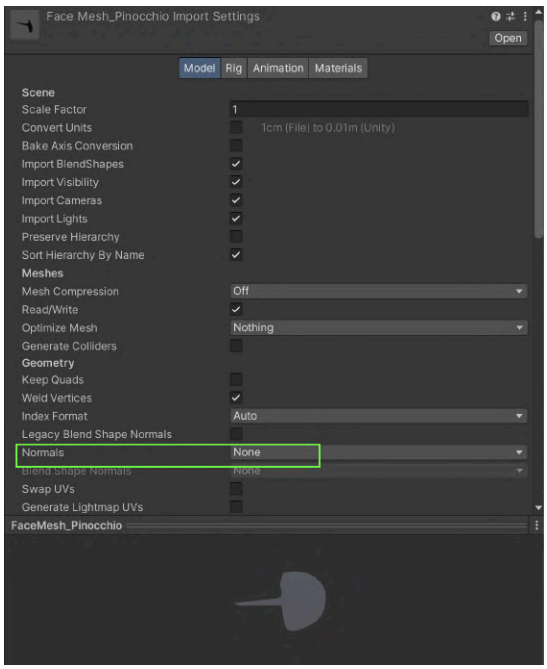


Face deformation allows you to manipulate the vertices of your facemesh using a second mesh (deformation mesh) while keeping the texture on.
Currently this is only possible in Built-In Render Pipeline, but we aim to port the functionality to URP as well. You are only limited to a single face deformation per scene for now (eg. 2 players with different deformations are not possible).

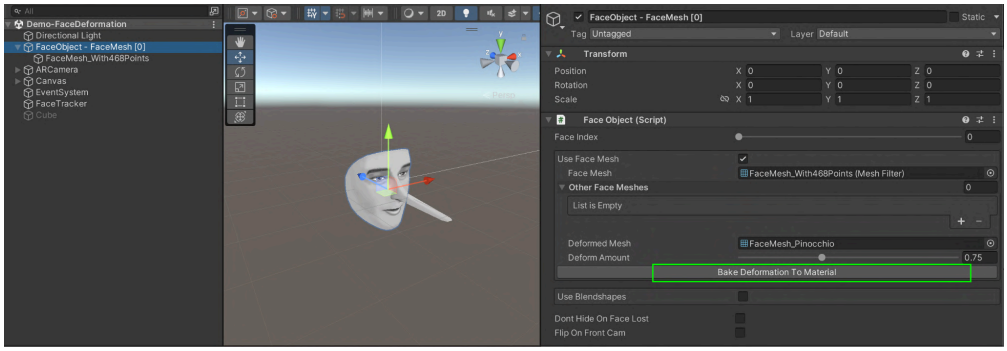
1. To do this, make a copy of our **FaceMesh_With468Points.fbx**, which will serve as our **deformedMesh.** Edit this mesh in your preferred 3D editing software to your liking. Just make sure to preserve the vertex count and order.



Also, when importing your deformedMesh back to Unity, make sure to set **Normals** to **None** in the import settings. Otherwise, Unity will add more vertices to bake a new Normal/UV geometry and will break the vertex references.

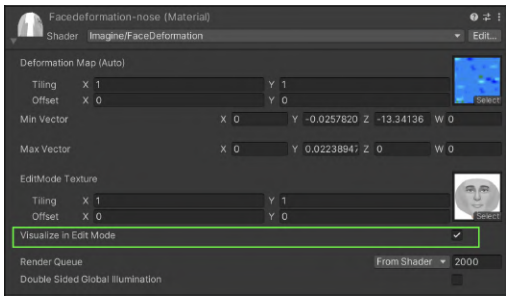


2. Then in your FaceObject inspector, enable **Use FaceMesh**, and under **DeformedMesh**, drag your new mesh. Set the deformation strength by dragging the **Deform Strength** slider.



3. Then lastly, click **Bake Deformation to Material** and choose the path to save your *deformation map*. This will also change your face mesh's material shader to **Imagine/FaceDeformation**, as well as apply the deformation map and vectors.

4. You can view the effect in Editor by enabling **Visualize in Edit Mode** in your deformation material



5. Build and run your project.

WebAR Best Practices

Lightweight AR experiences and game optimizations

Another thing to consider is that WebAR runs in web browsers with very limited processing power compared to other platforms. So it is important to choose a lightweight experience and well-optimized to run even on low-tier mobile devices across a wide range of browsers and versions. Consider the following simplifications:

- Use low-res sprite sheets and/or low-poly models
- Use simple/unlit shaders whenever possible
- Avoid real-time image effects and post-processing
- Use simple animations instead of physics simulations
- Total memory consumed less than 300MB

FPS and Overheating

It is also very important to consider the impact of frame rate to the tendency of the device to heat-up. AR experiences, in general, consume more resources (such as camera and calculations) than standard games, thus putting more strain on device hardware and causing them to heat up, especially on high frame rates and longer play sessions. And In response to overheating, devices will cap the framerate to avoid any serious damage.

From our testing, mid-tier devices (eg. iPhone 8) is able to run a simple AR scene at 60 FPS for 3-5 minutes straight, before the frame rate starts to drop due to overheating.

Hence, it is best to keep your frame rate as low as possible (30 FPS or less is recommended for WebAR experiences in mobile). As well as designing your AR experience to be playable in small intervals (2-3 minute sessions) while keeping your game as optimized as possible.

FaceTracker Settings

Tracker Cam

- Drag your scene's AR Camera in this field

FaceObjectGizmoType

- The face object gizmo type that helps you visualize the position of the face in the editor.
- Choose between Mesh, Wireframe, Mesh+Wireframe or None
- Deactivating gizmos in your editor settings will also deactivate the faceObject gizmo

FaceObjectGizmoColor

- The color of the faceGizmo.
- Best when transparency is around 20%, otherwise, can occlude your objects

EditorDebugMode

- Since AR only runs in the WebGL build, and not in the Editor, use editor buttons to manually trigger faceObject detections and the keyboard for moving the camera

FaceObject Settings

Your FaceObjects can be customized with several properties. We have created prefab templates for common use cases such as wearables, face mesh, and avatars. But in other cases, you will be customizing it for your specific needs.

Face Index

- The *unique* index of the player's face being tracked. Up to 4 players only

Use Face Mesh

- Enable this flag to allow mesh deformation based on face vertices

- This can be computationally expensive, and may run slow on older devices, especially when tracking multiple faces.

FaceMesh

- The MeshFilter to deform and overlay the user's face
- Only works with our custom face mesh model (FaceMesh_With468Points.fbx)

OtherFaceMeshes

- If you need to layer your face effects such as in face makeup, you can use more than one face mesh

DeformedMesh

- Drag your deformed face model here to implement face deformation

DeformStrength

- The amount of deformation to use

Use Blendshapes

- Enable this flag to allow reading blendshapes for your avatar

Avatar

- The skinned mesh renderer of your avatar

ShapesMap

- The mapping of your avatar blendshapes to their corresponding facetracker blendshapes from mediapipe

ShapesMap Item > Avatar Blendshape

- Your custom avatar blendshape

ShapesMap Item > FaceTracker Blendshape

- The corresponding blendshape from mediapipe's 52 blendshapes (See **FaceObject - Avatar** section)

ShapesMap Item > Curve

- The response curve of the blendshape
- Can either be Linear, QuadEaseIn, QuadEaseOut, QuadEaseInOut

Smoothen Blendshapes

- Minimize abrupt transitions by lerping/interpolating blendshape values

Smooth Factor

- The smoothing factor when lerping blendshape values

FaceShapeEvents

- This is used for creating custom gestures and triggering events when they are detected.

FaceShapeEvents > FaceEventName

- The name of the face event

FaceShapeEvents > Blendshapes

- The list of blendshapes to listen to
- Multiple blendshape values will get averaged

FaceShapeEvents > Threshold

- The threshold value to reference

FaceShapeEvents > DetectMode

- Fire-up your event when the blendshape value becomes GREATER or LESS than the threshold value

FaceShapeEvents > OnDetected

- The UnityEvent to fire up when gesture gets detected

FaceShapeEvents > OnLost

- The UnityEvent to fire up when gesture gets lost

CompensateScaleOnMouthOpen

- By default, opening the mouth can affect the computed position of the FaceObject. When the mouth is fully open, the face appears larger because it appears nearer to the camera, than when it is closed. This flag will allow you to compensate for this effect by scaling down the selected objects simultaneously.

ScaleOnMouthFullyOpen

- The approximate target scale when the mouth is fully open. Note, this will be different across different user faces, so there is no “perfect” value

ScaleTransformsOnMouthOpen

- The list of transforms to scale when compensating for mouth openness

DontHideOnFaceLost

- If this is enabled, the faceObject will not be disabled/hidden

FlipOnFrontCam

- By default, the front camera inverts the webcam image on the x-axis, Enabling this flag will also invert the scale of your faceObject to match this.

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 slightly 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.

Tracker - Scripting API

You can also use these API calls to control the WebGL tracker:

FaceTracker.StartTracker
Manually start the tracker. This is useful when you need to restart the tracker after stopping it.

FaceTracker.StopTracker
Manually stop your tracker. This is useful when you want to display non-AR related content in your scene.

ARCamera.PauseCamera
Temporarily pauses the camera.
Note: currently tracked objects will freeze in place if tracker is not stopped before calling this method.

ARCamera.UnpauseCamera
Resumes the camera.

ARCamera.FlipCamera
Toggle the camera from front to back, or vice versa.

Other Features:

ARShadowShader
Easily add shadow planes to your image target experiences

ChromaCutoutShader
Easily integrate green-screen/chroma videos to your AR experiences.

SyncVideoSound

Play Videoplayer and AudioSource simultaneously (because video with sound does not play in iOS Safari)

ScreenshotManager.GetScreenshot ()

Capture and display screenshots and allow users to save them to gallery or share to their social media apps

TextureDownloader.DownloadTexture (Texture2D texture)

Allows the users to download textures as a png or jpeg file. Useful for sharing image targets directly from the web browser for printing.
Note: Textures need to be **Uncompressed** and **Read/Write** enabled.

Current Limitations

Frame rate and Performance

Since WebGL supports a wide variety of devices and web browsers, the actual frame rate will be determined by the processing capability of the user device. During testing we have achieved 45-55 fps on newer iPhones (iPhone X, iPhone 14) while 12-24 fps on older devices (such as iPhone 8 and Samsung S8)

CV Source code

Source code of the CV module is not included by default, mainly because we cannot yet guarantee and provide support on its functionality and performance once customized by other developers.

Editor Simulation

Unfortunately, we do not yet have any means to test AR functionality in the Unity Editor. However, we plan to implement this feature in future versions.

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.

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.

Uncaught TypeError: Cannot read properties of undefined (reading 'FOV')

- Check your **Player Settings>Resolution and Presentation** and make sure that have selected the correct WebGL template.

Works in localhost, but webcam does not open when build is hosted

- Please make sure that you're hosting with SSL(using https).

Unity doesn't start - stuck in loading screen/white screen

- If you are seeing this error or something similar
  Uncaught SyntaxError: Invalid or unexpected token (at WebGL.framework.js.br:1:2)
  Try building without compression in PlayerSettings>Publishing Settings

Known Issues:

Visit the #bug-reports channel in our discord

Need Help?

Visit the #support-channel in our discord

Change Notes:

Version 1.0.2

• [ADDED] Face Paint shader (Transparent without self-occlusion) for Built-In RP

Version 1.0.1

• [ADDED] Demo scenes for avatar blendshapes, face mesh, virtual wearables, and multiple faces
• [ADDED] MenuItems for streamlined prefab creation
• [ADDED] ARCameraGlobalSettings which allows you to choose Front/Back Camera Facing modes

• [FIXED] Bug where face objects don't show up after closing and reopening a new scene
• [FIXED] Bug where camera FOV is wrong after closing and reopening a new scene
• [ADDED] Improved documentation
• Other minor improvements and cleanup

Version 1.0.0
• First release