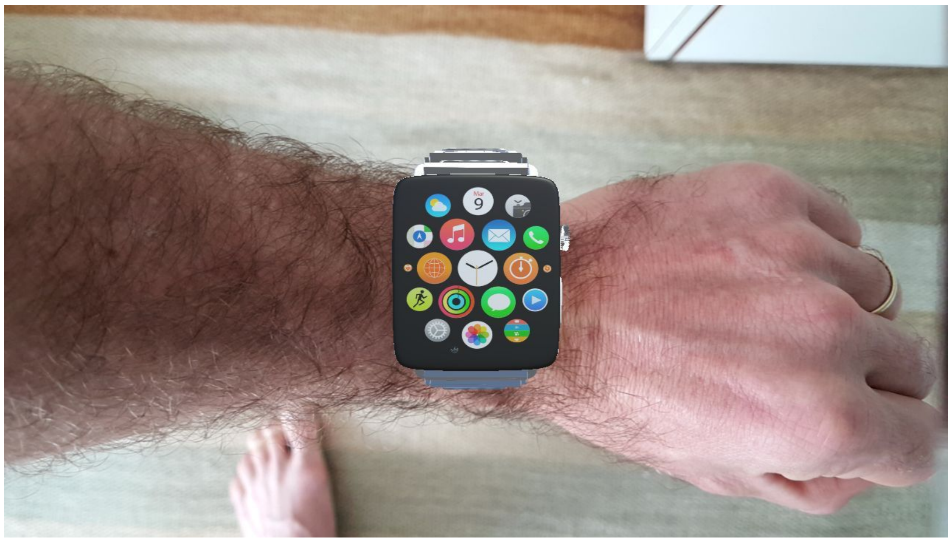
Mixing Mobile Camera and 3D Geometry

In this document I'll explain how to access your mobile camera through Unity API and mix video and 3D geometry on the same application.

Results

The result of this tutorial is an Android app that will capture information from the camera and directly play it on the phone screen. On top of the video there will be rendered a custom 3D model.



Screenshot of the application, the watch is a 3D model being rendered on top of the video.

Let’s get started!

You will need a fresh Unity Project, I’m using for this example Unity 2017.1.0f3. Next we must create a script to communicate with the camera and get the image from the phone and bring it to Unity.

I called this script *PhoneCameraInterface* it has the following overall structure.

```
using UnityEngine;

using UnityEngine.UI;

public class PhoneCameraInterface : MonoBehaviour

{

    /*

        ---------------------------------------------------------------------------

        # Phone Camera Interface #

        This class will take the video stream form the cellphone and display it on a

        WebCamTexture.

        ---------------------------------------------------------------------------

    */

    /* A control variable, is true when there is a camera properly set up. */

    private bool isCameraAvailable;

    /* The camera input stream.  */

    private WebCamTexture backCamera;

    /* Where the video will be displayed.  */

    private Texture defaultBackground;

    /* That UI image that will contain the defaultBackground Texture.  */

    public RawImage background;

    /* Component to modify the aspect ratio accordingly.  */

    public AspectRatioFitter fit;

}
```

Setting everything up...

Now we have the basic script set up let's work on the Start function where we will hook the camera to the in game texture.

First we get the background texture from the UI game object, then get a list of the webcam devices available. If there is no camera we return.

```
/* Get the background texture from UI game object. */

defaultBackground = background.texture;



/* Get a list of the webcam devices available. */

WebCamDevice[] devices = WebCamTexture.devices;



/* Case there is no camera. */
if (devices.Length == 0)
{

  Debug.Log("Could not find any camera.");

  isCameraAvailable = false;

  Return;

}
```

Now we loop through the devices and get the back camera of the phone.

```
/* This loop cycles through the devices and get the back camera. */
for (int i = 0; i < devices.Length; i++)
{
    if(!devices[i].isFrontFacing)
    {
      /* Create the webcamTexture that we will use given the display information. */
      backCamera = new WebCamTexture(devices[i].name, Screen.width, Screen.height);

     }
}
```

```
 /* If we couldn't find the back camera for some reason. */
if (backCamera == null)
{
   Debug.Log("Unable to find the back camera.");

   return;
}
```

At this point everything is set-up and ready for the camera to be started.

```
/*  After the set up, everything is ready to start the camera and get information from it.
*/
backCamera.Play();

/* Set the UI texture as the backCamera WebCamTexture. */
background.texture = backCamera;

/* There is a camera available. */
isCameraAvailable = true;
```

Scale, rotation and orientation...

That was enough for the Start function we just need to set up the ratio, orientation and scale of the texture on the Update function and we are ready to go!

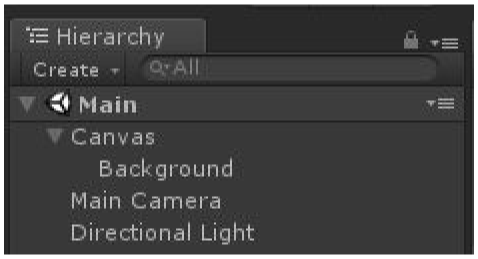
```
    void Update()
    {
        /* If there is no camera set up, we return. No need to run the rest of the code.*/
        if (!isCameraAvailable)
            return;

        /* This next bit of code will fix the ratio, scale and orientation of the camera so
it fits well the
           background transform. */

        // Ratio
        float ratio = (float)backCamera.width / (float)backCamera.height;
        fit.aspectRatio = ratio;

        // Scale
        float scaleY = backCamera.videoVerticallyMirrored ? -1f : 1f;
        background.rectTransform.localScale = new Vector3(1f, scaleY, 1f);

        // Angle
        int orient = -backCamera.videoRotationAngle;
        background.rectTransform.localEulerAngles = new Vector3(0, 0, orient);


    }
```

On Unity…

On an empty Unity scene add a *Raw Image* on an empty canvas.

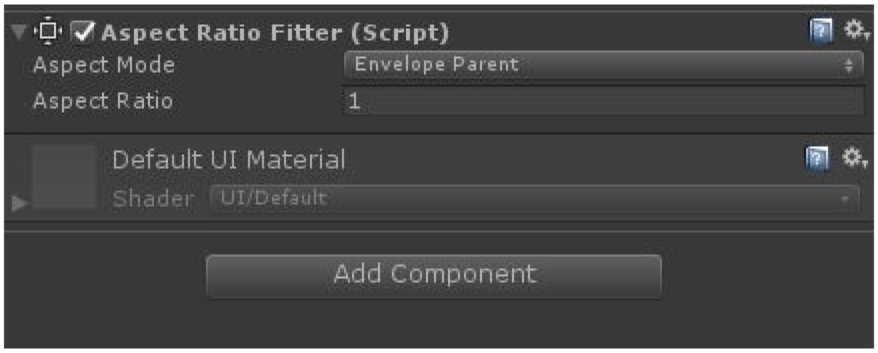
1. Right click on the Hiearchy tab.
2. Select UI
3. Select Raw Image

Rename it from Raw Image to Background.



Result

On the *Background* game object we add the component *Aspect Ratio Fitter* and set it's **aspect mode** to *Envelope Parent.*

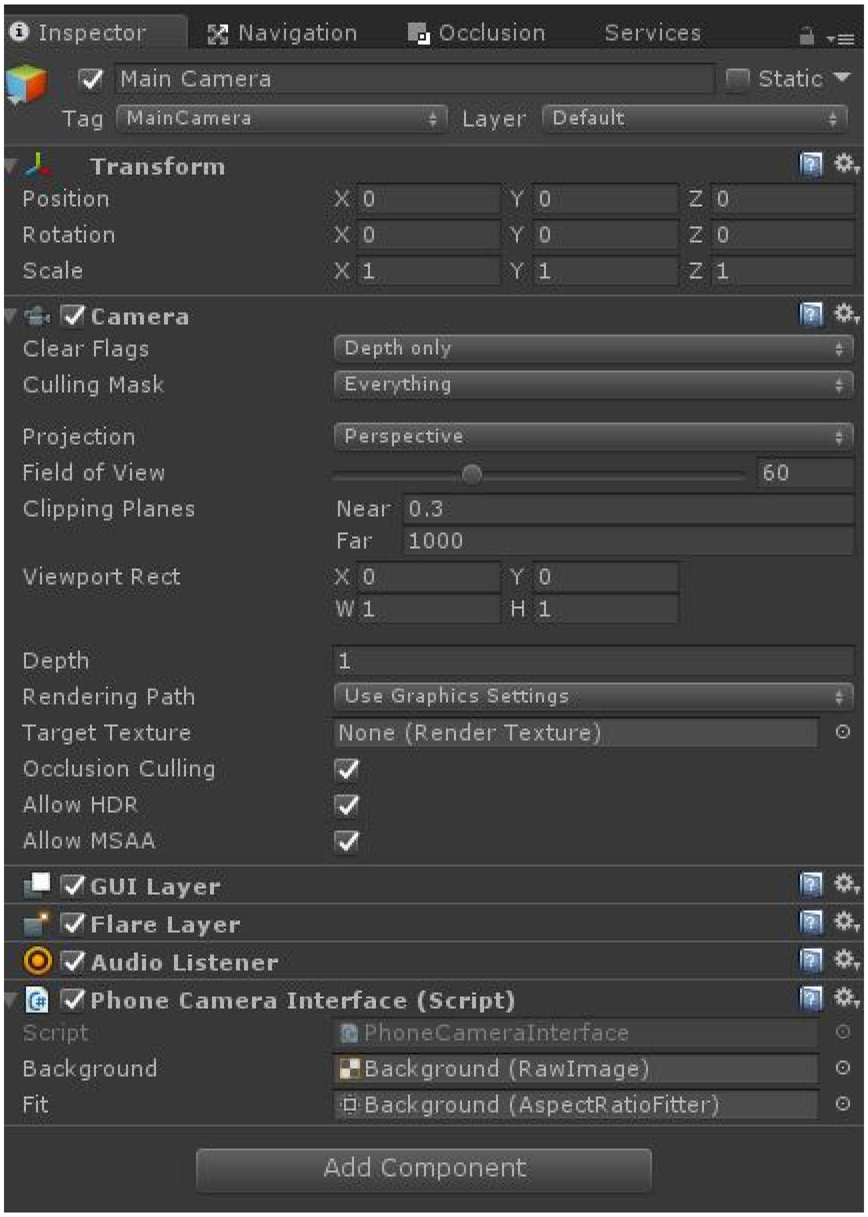


Aspect Ratio Fitter (Script)
Aspect Mode: Envelope Parent
Aspect Ratio: 1

Default UI Material
Shader: UI/Default

Now on the *Main Camera* object add the script we created *PhoneCameraInterface.* On the script we hook the *Background* game object on both *Background* and *Fit* variables.

Still on the Main Camera, switch **Clear Flags** to *Deph Only* and set **Depth** to *1*. That camera will render the 3D object and will render on top of the background camera. The following picture shows how it should look like.



Main Camera ✓ — Static
Tag: MainCamera — Layer: Default

Transform

	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

Camera ✓

Clear Flags	Depth only
Culling Mask	Everything
Projection	Perspective
Field of View	60
Clipping Planes	Near 0.3, Far 1000
Viewport Rect	X 0, Y 0, W 1, H 1
Depth	1
Rendering Path	Use Graphics Settings
Target Texture	None (Render Texture)
Occlusion Culling	✓
Allow HDR	✓
Allow MSAA	✓

GUI Layer ✓

Flare Layer ✓

Audio Listener ✓

Phone Camera Interface (Script) ✓

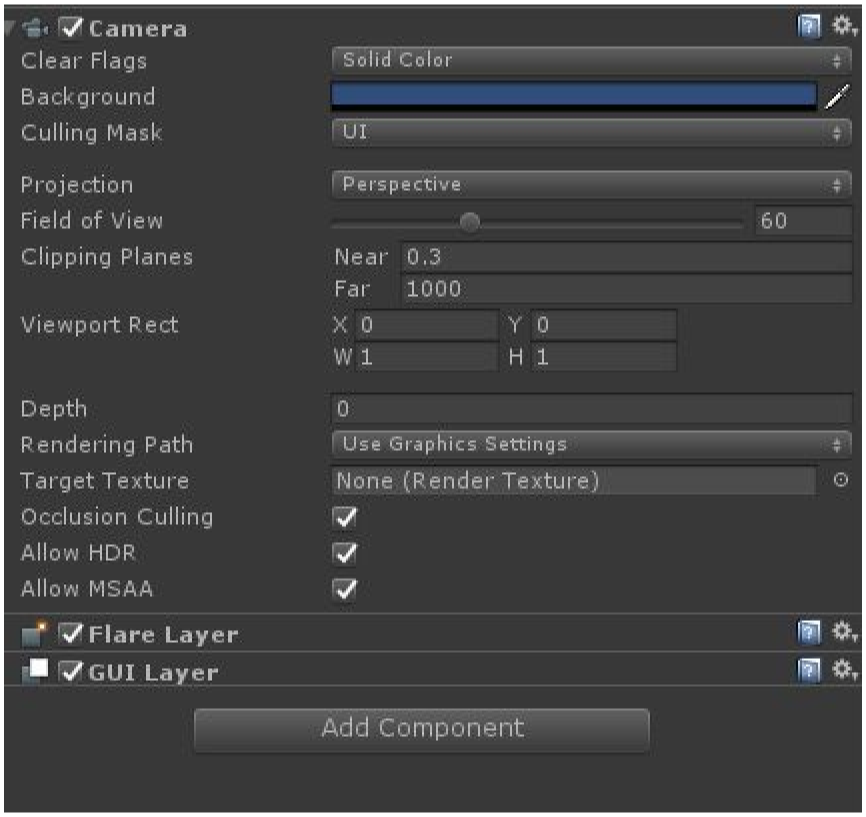
Script	PhoneCameraInterface
Background	Background (RawImage)
Fit	Background (AspectRatioFitter)

A camera for the background

Okay now lets make a camera just for the background video. Create a camera :

1. Right click on the Hiearchy tab.
2. Select Camera

On the new camera change **Clear Flags** to *Solid Color.* Change its depth to 0 and sets its Culling mask only to UI. Double check with the picture below if you got everything right.



Camera

Clear Flags	Solid Color
Background	🟦
Culling Mask	UI
Projection	Perspective
Field of View	60
Clipping Planes	Near 0.3
	Far 1000
Viewport Rect	X 0 Y 0
	W 1 H 1
Depth	0
Rendering Path	Use Graphics Settings
Target Texture	None (Render Texture)
Occlusion Culling	✓
Allow HDR	✓
Allow MSAA	✓

Flare Layer ✓

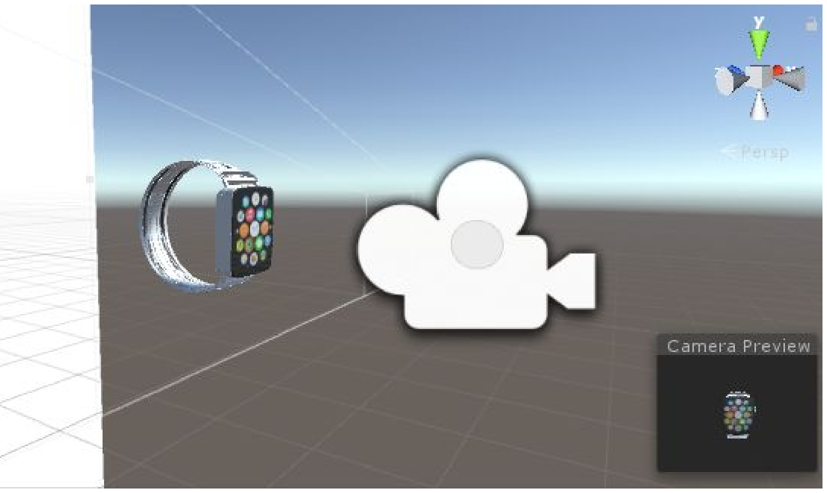
GUI Layer ✓

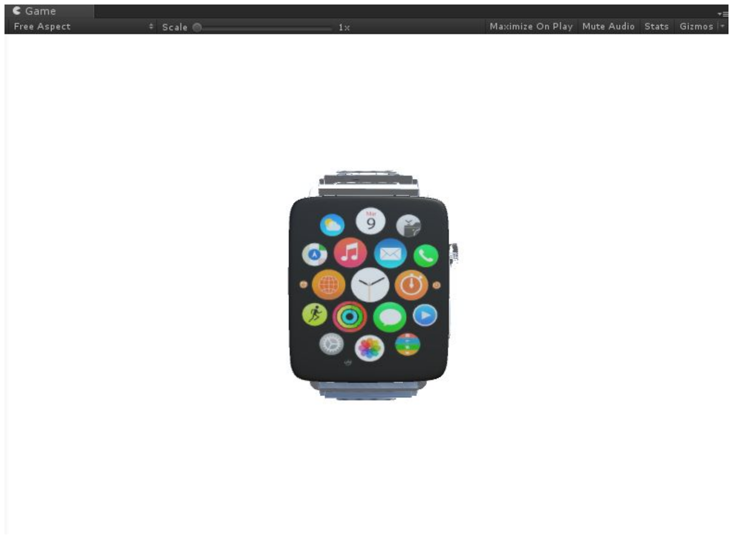
Add Component

Remember to place this camera on the exact same position as your Main Camera.

Adding geometry

Okay for my case I decided to use a simple 3D model for a watch you can pick whichever object you feel like using. Place it in front of the main camera in a way it looks fine and you are good to go.





Conclusions

This was a simple and quick tutorial to for activating the cell phone camera, hope you guys enjoyed it. It’s possible to find the source files for this Unity project on my webpage (http://www.yetanothergamer.com/2017/08/05/tutorial-mixing-mobile-camera-and-3d-geometry/).

Special thanks to N3K for his tutorials on Unity.