Table of Contents

Quick Start

Please note that the code provided in this page is *purely* for learning purposes and is far from perfect. Remember to null-check all responses!

AndroidManifest Setup

You will have to define the following permissions in your Android Manifest:

```
<uses-permission android:name="android.permission.CAMERA" android:required="true"/>
<uses-permission android:name="horizonos.permission.HEADSET_CAMERA"
android:required="true"/>
```

This package cannot request these permissions for you during runtime, you will have to do that manually.

Project Setup

Add an instance `UCameraManager` to the first scene that is loaded in your app. You can do this by right-clicking in the scene Hierarchy -> under Quest Camera -> click on Quest Camera Manager. This will add an instance of `UCameraManager` with the correct YUV 4:2:0 to RGBA converting compute shader. This is required as the Meta Quest's camera streams in the YUV format. `UCameraManager` is persistent across scenes, so you do not have to create more instances of it.

Example Usage

The following script will display the camera stream to a `RawImage`:

```
using System.Collections;
using UnityEngine;
using UnityEngine.UI;
using Uralstech.UXR.QuestCamera;

public class CameraTest : MonoBehaviour
{
    [SerializeField] private RawImage _rawImage;

    private IEnumerator Start()
    {
        // Check if the current device is supported.
        if (!CameraSupport.IsSupported)
        {
            Debug.LogError("Device does not support the Passthrough Camera API!");
            yield break;
        }

        // Get a camera device ID.
        string currentDevice = UCameraManager.Instance.CameraDevices[0];

        // Get the supported resolutions of the camera and choose the highest resolution.
        Resolution highestResolution = default;
        foreach (Resolution resolution in
UCameraManager.Instance.GetSupportedResolutions(currentDevice))
        {
            if (resolution.width * resolution.height > highestResolution.width
* highestResolution.height)
                highestResolution = resolution;
        }

        // Open the camera.
        CameraDevice camera = UCameraManager.Instance.OpenCamera(currentDevice);
        yield return camera.WaitForInitialization();

        // Check if it opened successfully
        if (camera.CurrentState != NativeWrapperState.Opened)
        {
            Debug.LogError("Could not open camera!");

            // Very important, this frees up any resources held by the camera.
            Destroy(camera.gameObject);
            yield break;
        }

        // Create a capture session with the camera, at the chosen resolution.
        CameraDevice.CaptureSessionObject sessionObject =
camera.CreateCaptureSession(highestResolution);
        yield return sessionObject.CaptureSession.WaitForInitialization();

        // Check if it opened successfully
        if (sessionObject.CaptureSession.CurrentState != NativeWrapperState.Opened)
        {
            Debug.LogError("Could not open camera session!");

            // Both of these are important for releasing the camera and session resources.
            Destroy(sessionObject.GameObject);
            Destroy(camera.gameObject);
            yield break;
        }

        // Set the image texture.
        _rawImage.texture = sessionObject.TextureConverter.FrameRenderTexture;
    }
}
```

Breaking Changes Notice

If you've just updated the package, it is recommended to check the *changelogs* for information on breaking changes.