```
/
*============================================================================
============
Copyright (c) 2017 PTC Inc. All Rights Reserved.

Copyright (c) 2010-2014 Qualcomm Connected Experiences, Inc.
All Rights Reserved.
Confidential and Proprietary - Protected under copyright and other
laws.
============================================================================
============*/

using UnityEngine;
using Vuforia;

// need to import video functionality
using UnityEngine.Video;
[RequireComponent(typeof(VideoPlayer))]

/// <summary>
///     A custom handler that implements the
ITrackableEventHandler interface.
/// </summary>
public class DefaultTrackableEventHandler : MonoBehaviour,
ITrackableEventHandler
{
    #region PRIVATE_MEMBER_VARIABLES

    protected TrackableBehaviour mTrackableBehaviour;
     // setup the videoPlayer object
     private VideoPlayer videoPlayer;

    #endregion // PRIVATE_MEMBER_VARIABLES

    #region UNTIY_MONOBEHAVIOUR_METHODS

    protected virtual void Start()
    {
        mTrackableBehaviour = GetComponent<TrackableBehaviour>();
        if (mTrackableBehaviour)

mTrackableBehaviour.RegisterTrackableEventHandler(this);

          // add the following 4 lines to get the reference to the
video player component of the plane that the video is attached to
          // IMPORTANT: set "Video_plane" to the name of the plane
game object you attached your video to
          GameObject video = GameObject.Find ("Video_plane");
          videoPlayer = video.GetComponent<VideoPlayer>();
          videoPlayer.Play();
          videoPlayer.Pause();
          // see the VideoPlayer Scripting API for more ideas on
```

```
which functions you can use in your code
          // for example changing the playback speed or jumping to
a speicific point in time:
          // https://docs.unity3d.com/ScriptReference/
Video.VideoPlayer.html

    }

    #endregion // UNTIY_MONOBEHAVIOUR_METHODS

    #region PUBLIC_METHODS

    /// <summary>
    ///     Implementation of the ITrackableEventHandler function
called when the
    ///     tracking state changes.
    /// </summary>
    public void OnTrackableStateChanged(
        TrackableBehaviour.Status previousStatus,
        TrackableBehaviour.Status newStatus)
    {
        if (newStatus == TrackableBehaviour.Status.DETECTED ||
            newStatus == TrackableBehaviour.Status.TRACKED ||
            newStatus ==
TrackableBehaviour.Status.EXTENDED_TRACKED)
        {
                //Debug.Log("Trackable " +
mTrackableBehaviour.TrackableName + " found");
                // Play the video:
                Debug.Log("Play!");
                OnTrackingFound();
                videoPlayer.Play();
          }
          else if (previousStatus ==
TrackableBehaviour.Status.TRACKED &&
                newStatus == TrackableBehaviour.Status.NOT_FOUND)
          {
                //Debug.Log("Trackable " +
mTrackableBehaviour.TrackableName + " lost");
                // Pause the video, if using Stop() the video would
always play back from the beginning again
                Debug.Log("Stop!");
                OnTrackingLost();
                videoPlayer.Pause();
        }
        else
        {
            // For combo of previousStatus=UNKNOWN +
newStatus=UNKNOWN|NOT_FOUND
            // Vuforia is starting, but tracking has not been lost
or found yet
            // Call OnTrackingLost() to hide the augmentations
```

```
            OnTrackingLost();
        }
    }

    #endregion // PUBLIC_METHODS

    #region PRIVATE_METHODS

    protected virtual void OnTrackingFound()
    {
        var rendererComponents =
GetComponentsInChildren<Renderer>(true);
        var colliderComponents =
GetComponentsInChildren<Collider>(true);
        var canvasComponents =
GetComponentsInChildren<Canvas>(true);

        // Enable rendering:
        foreach (var component in rendererComponents)
            component.enabled = true;

        // Enable colliders:
        foreach (var component in colliderComponents)
            component.enabled = true;

        // Enable canvas':
        foreach (var component in canvasComponents)
            component.enabled = true;
    }


    protected virtual void OnTrackingLost()
    {
        var rendererComponents =
GetComponentsInChildren<Renderer>(true);
        var colliderComponents =
GetComponentsInChildren<Collider>(true);
        var canvasComponents =
GetComponentsInChildren<Canvas>(true);

        // Disable rendering:
        foreach (var component in rendererComponents)
            component.enabled = false;

        // Disable colliders:
        foreach (var component in colliderComponents)
            component.enabled = false;

        // Disable canvas':
        foreach (var component in canvasComponents)
            component.enabled = false;
    }
```

```
    #endregion // PRIVATE_METHODS
}
```