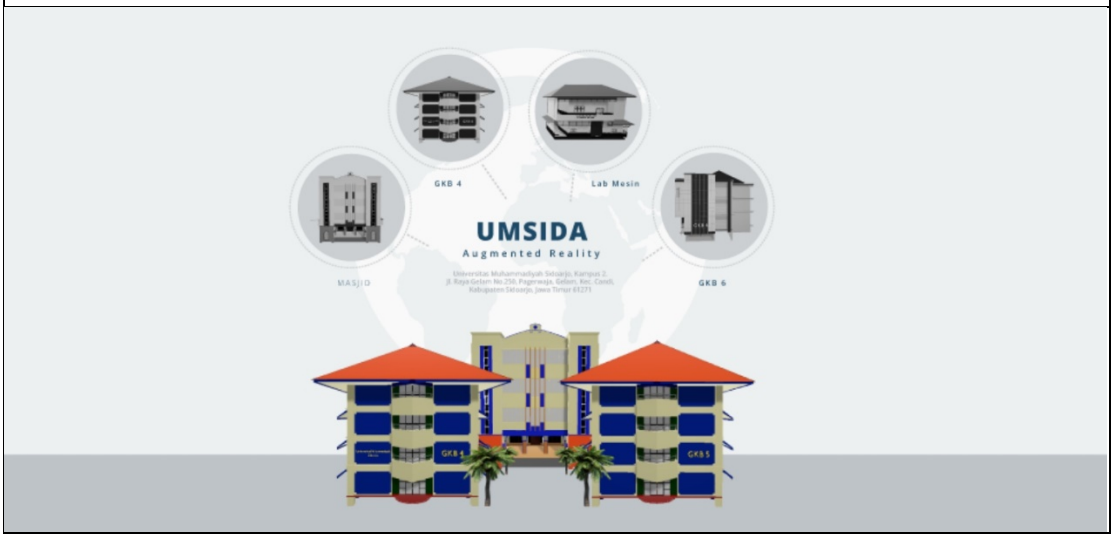
FILE CIPTAAN HKI *Virtual* 3D Kampus 2 Universitas Muhammadiyah Sidoarjo Berbasis *Augmented Reality*

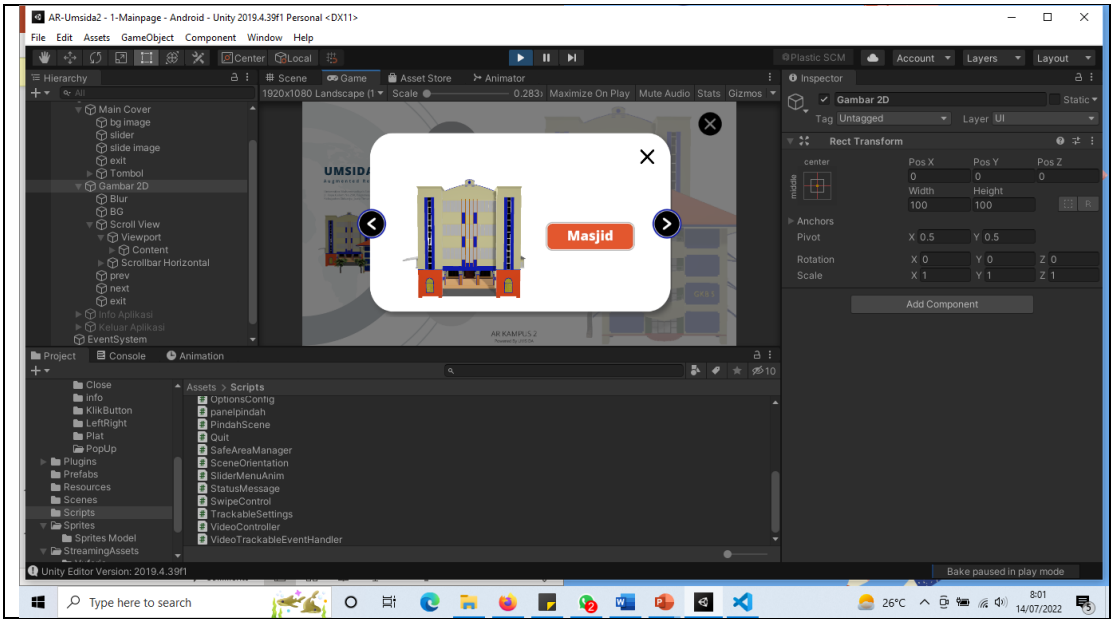
1. SPLASH SCREEN

```
/*============================================================================
Copyright (c) 2016-2017 PTC Inc. All Rights Reserved.
Vuforia is a trademark of PTC Inc., registered in the United States and other
countries.
============================================================================*/
using UnityEngine;
using UnityEngine.UI;
using System.Collections;

public class AsyncSceneLoader : MonoBehaviour
{
   #region PUBLIC_MEMBERS
   public float loadingDelay = 5.0F;
   #endregion //PUBLIC_MEMBERS

   #region MONOBEHAVIOUR_METHODS
   void Start()
   {
      StartCoroutine(LoadNextSceneAfter(loadingDelay));
   }
   #endregion //MONOBEHAVIOUR_METHODS

   #region PRIVATE_METHODS
   private IEnumerator LoadNextSceneAfter(float seconds)
   {
      yield return new WaitForSeconds(seconds);

      UnityEngine.SceneManagement.SceneManager.LoadScene(UnityEngine.SceneManagement.SceneManag
er.GetActiveScene().buildIndex + 1);
   }
   #endregion //PRIVATE_METHODS
}
```



2. SWIPE CONTROL (2D)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class SwipeControl: MonoBehaviour
{
   public GameObject scrollbar;
   float scroll_pos = 0;
   float [] pos;
   int posisi = 0;

   void Start()
   {

   }

    public void prev(){
      if (posisi > 0){
         posisi -= 1;
         scroll_pos = pos [posisi];
      }
   }

   public void next(){
      if (posisi < pos.Length - 1){
         posisi += 1;
         scroll_pos = pos [posisi];
      }
   }



   // Update is called once per frame
   void Update()
   {
      pos = new float[transform.childCount];
      float distance = 1f / (pos.Length - 1f);
      for (int i = 0; i < pos.Length; i++) {
         pos [i] = distance * i;
      }

      if (Input.GetMouseButton (0)) {
         scroll_pos = scrollbar.GetComponent<Scrollbar> ().value;
      } else {
         for (int i = 0; i < pos.Length; i++) {
            if (scroll_pos < pos [i] + (distance / 2) && scroll_pos > pos [i] - (distance / 2)) {
               scrollbar.GetComponent<Scrollbar> ().value = Mathf.Lerp
(scrollbar.GetComponent<Scrollbar>().value, pos[i], 0.15f);
               posisi = i;

            }
         }
      }

   }
}
```

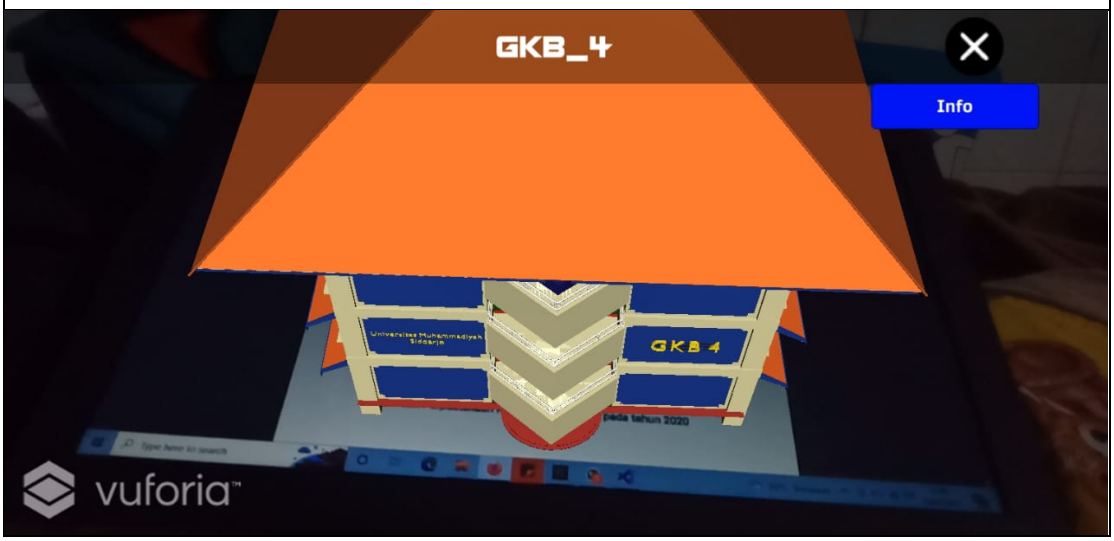


3. AR CAMERA

```
/*============================================================================
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;

/// <summary>
/// A custom handler that implements the ITrackableEventHandler interface.
///
/// Changes made to this file could be overwritten when upgrading the Vuforia version.
/// When implementing custom event handler behavior, consider inheriting from this class instead.
/// </summary>
public class DefaultTrackableEventHandler : MonoBehaviour, ITrackableEventHandler
{
    #region PROTECTED_MEMBER_VARIABLES

    protected TrackableBehaviour mTrackableBehaviour;
    protected TrackableBehaviour.Status m_PreviousStatus;
    protected TrackableBehaviour.Status m_NewStatus;

    #endregion // PROTECTED_MEMBER_VARIABLES

    #region UNITY_MONOBEHAVIOUR_METHODS

    protected virtual void Start()
    {
        mTrackableBehaviour = GetComponent<TrackableBehaviour>();
        if (mTrackableBehaviour)
            mTrackableBehaviour.RegisterTrackableEventHandler(this);
    }

    protected virtual void OnDestroy()
    {
```

```
    if (mTrackableBehaviour)
      mTrackableBehaviour.UnregisterTrackableEventHandler(this);
  }

  #endregion // UNITY_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)
  {
    m_PreviousStatus = previousStatus;
    m_NewStatus = newStatus;

    if (newStatus == TrackableBehaviour.Status.DETECTED ||
      newStatus == TrackableBehaviour.Status.TRACKED ||
      newStatus == TrackableBehaviour.Status.EXTENDED_TRACKED)
    {
      Debug.Log("Trackable " + mTrackableBehaviour.TrackableName + " found");
      OnTrackingFound();
    }
    else if (previousStatus == TrackableBehaviour.Status.TRACKED &&
        newStatus == TrackableBehaviour.Status.NO_POSE)
    {
      Debug.Log("Trackable " + mTrackableBehaviour.TrackableName + " lost");
      OnTrackingLost();
    }
    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 PROTECTED_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 // PROTECTED_METHODS
}
```



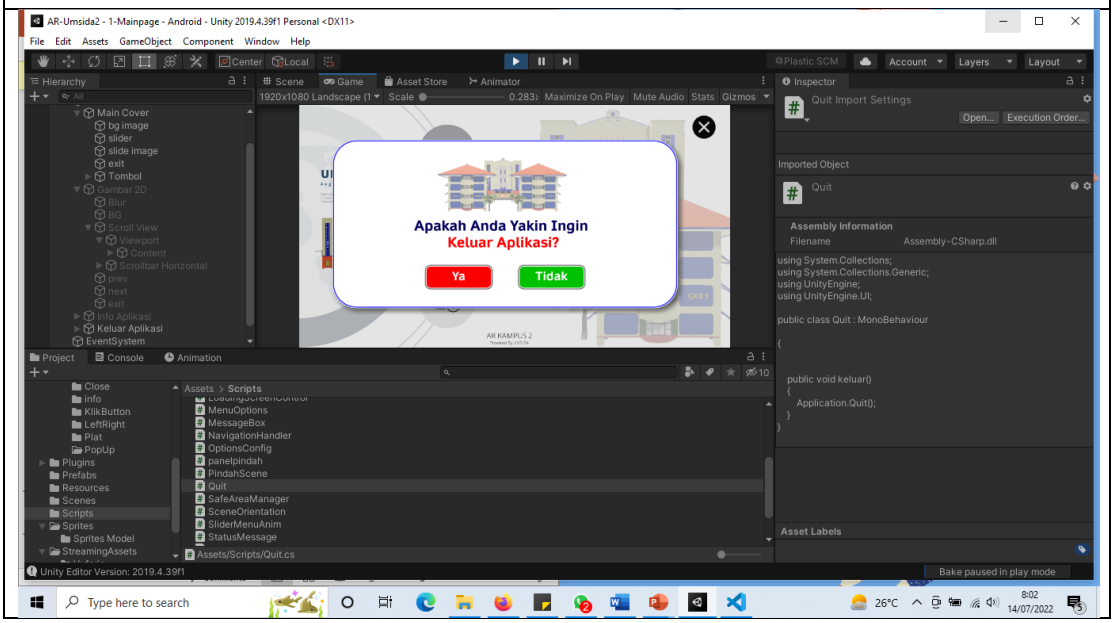
4. PINDAH SCENE

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class PindahScene : MonoBehaviour
{
    public void Play(string scene_name) {
        SceneManager.LoadScene(scene_name);
    }
}
```

5. KELUAR

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Quit : MonoBehaviour
```

```
{

   public void keluar()
   {
      Application.Quit();
   }
}
```



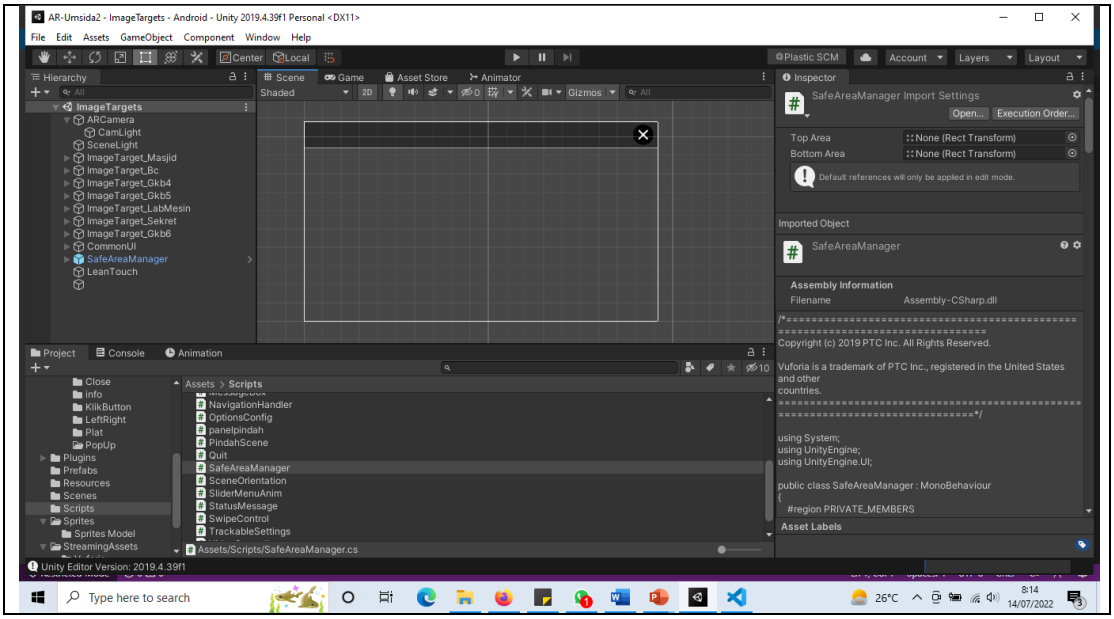
6. SAVE AREA

```
/*============================================================================
Copyright (c) 2019 PTC Inc. All Rights Reserved.

Vuforia is a trademark of PTC Inc., registered in the United States and other
countries.
============================================================================*/

using System;
using UnityEngine;
using UnityEngine.UI;

public class SafeAreaManager : MonoBehaviour
{
   #region PRIVATE_MEMBERS

   [System.Serializable]
   class SafeAreaRect
   {
      public RectTransform rectTransform = null;
      [Header("Apply Safe Area Constraints")]
      public bool top = false;
      public bool bottom = false;
   }

   [Header("Global Unsafe Area Settings (Per-Scene)")]
   [Tooltip("Unsafe Area Colors can be changed programmatically at runtime.")]
   [SerializeField] RectTransform topArea = null;
   [SerializeField] RectTransform bottomArea = null;
   [SerializeField] Color topAreaColor;
   [SerializeField] Color bottomAreaColor;
```

```
    [Tooltip("Safe Area Margin reduces the Safe Area by the specified amount at the Top/Bottom boundaries. " +
            "It is useful for testing Safe Area Behaviour in PlayMode.")]
    [Range(0,100)] // Max range value is arbitrary for example purposes
    [SerializeField] private int SafeAreaMargin = 0;

    [Header("Apply Safe Area Constraints to RectTransforms")]
    [SerializeField] SafeAreaRect[] safeAreaRects = null;

    ScreenOrientation lastOrientation;
    Rect lastSafeArea = new Rect(0, 0, 0, 0);
    Rect safeArea;
    Image topAreaImage = null;
    Image bottomAreaImage = null;
    bool colorsChanged => (topAreaColor != topAreaImage.color) || (bottomAreaColor != bottomAreaImage.color);

    #endregion // PRIVATE_MEMBERS

    #region MONOBEHAVIOUR_METHODS

    void Awake()
    {
        if (!topArea || !bottomArea)
        {
            Debug.LogWarning("Either topArea or bottomArea is null. Programmatically getting the required references.");
            SetAreaRectTransforms();
        }

        // cache our unsafe area image components
        this.topAreaImage = this.topArea.GetComponent<Image>();
        this.bottomAreaImage = this.bottomArea.GetComponent<Image>();

        // Set the unsafe area colors using Inspector values
        SetAreaColors(this.topAreaColor, this.bottomAreaColor);

        this.safeArea = GetSafeArea();
    }

    void SetAreaRectTransforms()
    {
        var images = GetComponentsInChildren<Image>();
        if (images.Length != 2)
        {
            Debug.LogError($"SafeAreaManager must have exactly two children with Image components attached.");
            return;
        }

        topArea = images[0].rectTransform;
        bottomArea = images[1].rectTransform;
    }

    Rect GetSafeArea()
    {
        return new Rect(
            Screen.safeArea.x,
            Screen.safeArea.y + this.SafeAreaMargin,
            Screen.safeArea.width,
            Screen.safeArea.height - (this.SafeAreaMargin * 2));
    }
```

```
    void Start()
    {
        this.lastOrientation = Screen.orientation;

        Refresh();
    }

    void Update()
    {
        Refresh();
    }

    #endregion // MONOBEHAVIOUR_METHODS

    #region PRIVATE_METHODS

    void Refresh()
    {
        this.safeArea = GetSafeArea();

        if ((this.safeArea != this.lastSafeArea) || (Screen.orientation != this.lastOrientation))
        {
            ApplySafeArea();
            UpdateUnsafeArea();
        }

        if (this.colorsChanged)
        {
            SetAreaColors(this.topAreaColor, this.bottomAreaColor);
        }
    }

    void ApplySafeArea()
    {
        this.lastSafeArea = this.safeArea;
        this.lastOrientation = Screen.orientation;

        foreach (SafeAreaRect areaRect in this.safeAreaRects)
        {
            var anchorMin = this.safeArea.position;
            var anchorMax = this.safeArea.position + this.safeArea.size;

            anchorMin.x /= Screen.width;
            anchorMin.y = areaRect.bottom ? anchorMin.y / Screen.height : 0;
            anchorMax.x /= Screen.width;
            anchorMax.y = areaRect.top ? anchorMax.y / Screen.height : 1;

            if (Screen.orientation == ScreenOrientation.LandscapeLeft ||
                Screen.orientation == ScreenOrientation.LandscapeRight)
            {
                anchorMin.x = 0;
                anchorMax.x = 1;
            }

            areaRect.rectTransform.anchorMin = anchorMin;
            areaRect.rectTransform.anchorMax = anchorMax;
        }
    }

    void UpdateUnsafeArea()
    {
        var anchorMin = this.safeArea.position;
        var anchorMax = this.safeArea.position + this.safeArea.size;
```

```
            anchorMin.x /= Screen.width;
            anchorMin.y = anchorMin.y / Screen.height;
            anchorMax.x /= Screen.width;
            anchorMax.y = anchorMax.y / Screen.height;

            SetUnsafeAreaSizes(anchorMin.y, anchorMax.y);

            SetAreaColors(this.topAreaColor, this.bottomAreaColor);
        }

        void SetUnsafeAreaSizes(float safeAreaAnchorMinY, float safeAreaAnchorMaxY)
        {
            this.topArea.anchorMin = new Vector2(0, safeAreaAnchorMaxY);
            this.topArea.anchorMax = Vector2.one;

            this.bottomArea.anchorMin = Vector2.zero;
            this.bottomArea.anchorMax = new Vector2(1, safeAreaAnchorMinY);
        }

        #endregion // PRIVATE_METHODS

        #region PUBLIC_METHODS

        public void AddSafeAreaRect(RectTransform rect, bool applyTopConstraint, bool
    applyBottomConstraint)
        {
            Array.Resize(ref this.safeAreaRects, this.safeAreaRects.Length + 1);
            this.safeAreaRects[this.safeAreaRects.Length - 1] = new SafeAreaRect
            {
                rectTransform = rect,
                top = applyTopConstraint,
                bottom = applyBottomConstraint
            };

            ApplySafeArea();
        }

        public void SetAreasEnabled(bool topAreaEnabled, bool bottomAreaEnabled)
        {
            this.topAreaImage.enabled = topAreaEnabled;
            this.bottomAreaImage.enabled = bottomAreaEnabled;
        }

        /// <summary>
        /// Sets the area colors programmatically and updates Inspector colors.
        /// </summary>
        /// <param name="topColor">Top color.</param>
        /// <param name="bottomColor">Bottom color.</param>
        public void SetAreaColors(Color topColor, Color bottomColor)
        {
            // update Inspector-level colors to match programmatic ones
            this.topAreaColor = topColor;
            this.bottomAreaColor = bottomColor;

            // assign the colors
            this.topAreaImage.color = this.topAreaColor;
            this.bottomAreaImage.color = this.bottomAreaColor;
        }

        #endregion // PUBLIC_METHODS
    }
```


AR-Umsida2 - ImageTargets - Android - Unity 2019.4.39f1 Personal <DX11>
File Edit Assets GameObject Component Window Help
Hierarchy
ImageTargets
ARCamera
CamLight
SceneLight
ImageTarget_Masjid
ImageTarget_Bc
ImageTarget_Gkb4
ImageTarget_Gkb5
ImageTarget_LabMesin
ImageTarget_Sekret
ImageTarget_Gkb6
CommonUI
SafeAreaManager
LeanTouch
Scene
Game
Asset Store
Animator
Inspector
SafeAreaManager Import Settings
Open...
Execution Order...
Top Area
None (Rect Transform)
Bottom Area
None (Rect Transform)
Default references will only be applied in edit mode.
Imported Object
SafeAreaManager
Assembly Information
Filename
Assembly-CSharp.dll
Copyright (c) 2019 PTC Inc. All Rights Reserved.
Vuforia is a trademark of PTC Inc., registered in the United States and other countries.
using System;
using UnityEngine;
using UnityEngine.UI;
public class SafeAreaManager : MonoBehaviour
#region PRIVATE_MEMBERS
Asset Labels
Project
Console
Animation
Assets > Scripts
NavigationHandler
OptionsConfig
panelpindah
PindahScene
Quit
SafeAreaManager
SceneOrientation
SliderMenuAnim
StatusMessage
SwipeControl
TrackableSettings
Assets/Scripts/SafeAreaManager.cs
Unity Editor Version: 2019.4.39f1
Type here to search
26°C
8:14
14/07/2022