Додаток В

|  |
| --- |
| «Затверджую» |
| зав. кафедрою ІТП |
| Прокопенко Т.О. |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2018 р. |

**Дослідження технологій SLAM в доповненій реальності**

Текст програми «AR T-shirt»

482 ЧДТУ 87137-01 11 03

Листів 4

Розробник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Нечипоренко О. В.

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Карапетян А. Р.

Н. контроль \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Колесніков К. В.

Черкаси – 2018

using System;

using UnityEngine;

using Utilities;

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 Events

public event Action TargetFindEvent;

public event Action TargetLostEvent;

#endregion

#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;

Debug.Log("Trackable " + mTrackableBehaviour.TrackableName + " track");

if (newStatus == TrackableBehaviour.Status.DETECTED ||

newStatus == TrackableBehaviour.Status.TRACKED ||

newStatus == TrackableBehaviour.Status.EXTENDED\_TRACKED)

{

Debug.Log("Trackable " + mTrackableBehaviour.TrackableName + " found");

TargetFindEvent.InvokeSafe();

OnTrackingFound();

}

else if (previousStatus == TrackableBehaviour.Status.TRACKED &&

newStatus == TrackableBehaviour.Status.NO\_POSE)

{

Debug.Log("Trackable " + mTrackableBehaviour.TrackableName + " lost");

TargetLostEvent.InvokeSafe();

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

TargetLostEvent.InvokeSafe();

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

}