Додаток Г

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

**Дослідження технологій SLAM в доповненій реальності**

Текст програми «AR Car»

482 ЧДТУ 817137-04 11 04

Листів 6

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

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

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

Черкаси – 2018

using System;

using UnityARInterface;

using UnityEngine;

namespace MonoBehaviours.Controllers

{

public class ARInterfaceController : ARBase

{

[SerializeField] public float \_actualObjectScale;

private float \_currentObjectSize;

private bool \_isObjectContainerPlaced;

private int \_layerMask;

private int \_planeLayer;

public ARController ArController;

public GameObject HitParentContainer;

public float MaxObjectScaleValue;

public float MinObjectScaleValue;

public ARPointCloudVisualizer PointCloudVisualizer;

public static event Action<bool> ObjectSuccessfullyPlacedEvent;

private void OnEnable()

{

SubscribeOnEvents();

}

private void OnDisable()

{

UnsubscribeFromEvent();

}

private void Start()

{

Init();

}

private void Update()

{

PlaceObjectOnPlane();

}

private void Init()

{

ArController = GetFirstEnabledControllerInChildren();

\_planeLayer = GetComponent<ARPlaneVisualizer>().planeLayer;

\_layerMask = 1 << \_planeLayer;

}

private void SubscribeOnEvents()

{

GesturesTouchController.PinchInEvent += OnPinchIn;

GesturesTouchController.PinchOutEvent += OnPinchOut;

GesturesTouchController.FingerSwipeEvent += OnRotate;

GesturesTouchController.TwoFingersSwipeEvent += OnChangePosition;

CarInstructionsUiController.InstructionsClosed += HidePointCloud;

CarInstructionsUiController.ScanningInstructionsTought += ShowPointCloud;

CarArViewUiController.ResetButtonClick += OnResetRecognition;

CarArViewUiController.ActualSizeButtonClick += OnSetActualSize;

}

private void UnsubscribeFromEvent()

{

GesturesTouchController.PinchInEvent -= OnPinchIn;

GesturesTouchController.PinchOutEvent -= OnPinchOut;

GesturesTouchController.FingerSwipeEvent -= OnRotate;

GesturesTouchController.TwoFingersSwipeEvent -= OnChangePosition;

CarInstructionsUiController.InstructionsClosed -= HidePointCloud;

CarInstructionsUiController.ScanningInstructionsTought -= ShowPointCloud;

CarArViewUiController.ResetButtonClick -= OnResetRecognition;

CarArViewUiController.ActualSizeButtonClick -= OnSetActualSize;

}

public void PlaceObjectOnPlane()

{

if (Input.GetMouseButton(0) && !\_isObjectContainerPlaced)

{

ConstructRayAndPlaceObjectContainer();

}

}

private void ConstructRayAndPlaceObjectContainer()

{

var camera = GetCamera();

var ray = camera.ScreenPointToRay(Input.mousePosition);

RaycastHit rayHit;

if (Physics.Raycast(ray, out rayHit, float.MaxValue, \_layerMask))

{

ArController.pointOfInterest = HitParentContainer.transform.position;

ArController.AlignWithPointOfInterest(rayHit.point);

if (PointCloudVisualizer.enabled)

{

\_isObjectContainerPlaced = true;

// Raise successful place event

if (ObjectSuccessfullyPlacedEvent != null)

{

ObjectSuccessfullyPlacedEvent(true);

}

}

}

}

public void OnReset()

{

if (ObjectSuccessfullyPlacedEvent != null)

{

ObjectSuccessfullyPlacedEvent(false);

}

\_isObjectContainerPlaced = false;

}

public void OnRotate(float rotateAngle)

{

ArController.rotation = Quaternion.AngleAxis(rotateAngle, Vector3.up);

}

public float GetCurrentScale()

{

return ArController.scale;

}

public void OnChangePosition()

{

ConstructRayAndPlaceObjectContainer();

}

public void OnResetRecognition()

{

OnReset();

SetActualSize();

}

public void OnSetActualSize()

{

SetActualSize();

}

public void ShowPointCloud()

{

PointCloudVisualizer.enabled = true;

}

public void HidePointCloud()

{

PointCloudVisualizer.enabled = false;

}

#region Instance

private static ARInterfaceController \_instance;

public static ARInterfaceController Instance

{

get

{

if (\_instance == null)

{

\_instance = FindObjectOfType<ARInterfaceController>();

}

return \_instance;

}

}

#endregion

#region Model input manipulation

/// <summary>

/// PinchIn event handler.

/// </summary>

/// <param name="zoom">Zoom.</param>

private void OnPinchIn(float zoom)

{

ZoomOut(zoom);

}

/// <summary>

/// PinchOut event handler.

/// </summary>

/// <param name="zoom">Zoom.</param>

private void OnPinchOut(float zoom)

{

ZoomIn(zoom);

}

/// <summary>

/// Decrease scale.

/// </summary>

/// <returns>The in.</returns>

/// <param name="val">Value.</param>

public float ZoomIn(float val)

{

if (ArController.scale - val > MinObjectScaleValue)

{

ArController.scale -= val;

}

return ArController.scale;

}

/// <summary>

/// Encrease scale.

/// </summary>

/// <returns>The out.</returns>

/// <param name="val">Value.</param>

public float ZoomOut(float val)

{

if (ArController.scale + val < MaxObjectScaleValue)

{

ArController.scale += val;

}

return ArController.scale;

}

/// <summary>

/// Set the actual scale which declarated in variable ActualObjectSize.

/// </summary>

public void SetActualSize(float? actualSize = null)

{

ArController.scale = actualSize ?? \_actualObjectScale;

}

#endregion

}

}