«SLAM технологія в доповненій реальності»

Додаток А

Лістинг програми «Olive tree»

Листів 5

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

Черкаси, 2018

using System;

using System.Collections.Generic;

using UnityEngine.EventSystems;

namespace UnityEngine.XR.iOS

{

public class UnityARHitTestExample : MonoBehaviour

{

public Transform m\_HitTransform;

bool HitTestWithResultType (ARPoint point, ARHitTestResultType resultTypes)

{

List<ARHitTestResult> hitResults = UnityARSessionNativeInterface.GetARSessionNativeInterface ().HitTest (point, resultTypes);

if (hitResults.Count > 0)

{

foreach (var hitResult in hitResults)

{

Debug.Log ("Got hit!");

m\_HitTransform.position = UnityARMatrixOps.GetPosition (hitResult.worldTransform);

m\_HitTransform.rotation = UnityARMatrixOps.GetRotation (hitResult.worldTransform);

Debug.Log (string.Format ("x:{0:0.######} y:{1:0.######} z:{2:0.######}", m\_HitTransform.position.x, m\_HitTransform.position.y, m\_HitTransform.position.z));

return true;

}

}

return false;

}

// Update is called once per frame

void Update ()

{

var touch = Input.GetTouch (0);

if (Input.touchCount > 0 && m\_HitTransform != null) {

if (touch.phase == TouchPhase.Began || touch.phase == TouchPhase.Moved) {

if (!EventSystem.current.IsPointerOverGameObject (0)) {

transform.localScale = new Vector3 (5, 5, 5);

var screenPosition = Camera.main.ScreenToViewportPoint (touch.position);

ARPoint point = new ARPoint

{

x = screenPosition.x,

y = screenPosition.y

};

// prioritize reults types

ARHitTestResultType[] resultTypes =

{

ARHitTestResultType.ARHitTestResultTypeExistingPlaneUsingExtent,

// if you want to use infinite planes use this:

//ARHitTestResultType.ARHitTestResultTypeExistingPlane,

ARHitTestResultType.ARHitTestResultTypeHorizontalPlane,

ARHitTestResultType.ARHitTestResultTypeFeaturePoint

};

foreach (ARHitTestResultType resultType in resultTypes)

{

if (HitTestWithResultType (point, resultType))

{

return;

}}}}}}}}

using System;

using UnityEngine;

using UnityEngine.XR.iOS;

using System.Collections;

using System.Collections.Generic;

using System.Runtime.InteropServices;

public class UnityPointCloudExample : MonoBehaviour

{

public uint numPointsToShow = 100;

public GameObject PointCloudPrefab = null;

private List<GameObject> pointCloudObjects;

private Vector3[] m\_PointCloudData;

public void Start()

{

UnityARSessionNativeInterface.ARFrameUpdatedEvent += ARFrameUpdated;

if (PointCloudPrefab != null)

{

pointCloudObjects = new List<GameObject> ();

for (int i =0; i < numPointsToShow; i++)

{

pointCloudObjects.Add (Instantiate (PointCloudPrefab));

}

}

}

public void ARFrameUpdated(UnityARCamera camera)

{

m\_PointCloudData = camera.pointCloudData;

}

public void Update()

{

if (PointCloudPrefab != null && m\_PointCloudData != null)

{

for (int count = 0; count < Math.Min (m\_PointCloudData.Length, numPointsToShow); count++)

{

Vector4 vert = m\_PointCloudData [count];

GameObject point = pointCloudObjects [count];

point.transform.position = new Vector3(vert.x, vert.y, vert.z);

}

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.XR.iOS;

public class PointCloudParticleExample : MonoBehaviour {

public ParticleSystem pointCloudParticlePrefab;

public int maxPointsToShow;

public float particleSize = 1.0f;

private Vector3[] m\_PointCloudData;

private bool frameUpdated = false;

private ParticleSystem currentPS;

private ParticleSystem.Particle [] particles;

// Use this for initialization

void Start () {

UnityARSessionNativeInterface.ARFrameUpdatedEvent += ARFrameUpdated;

currentPS = Instantiate (pointCloudParticlePrefab);

frameUpdated = false;

}

public void ARFrameUpdated(UnityARCamera camera)

{

m\_PointCloudData = camera.pointCloudData;

frameUpdated = true;

}

// Update is called once per frame

void Update () {

if (frameUpdated) {

if (m\_PointCloudData != null && m\_PointCloudData.Length > 0) {

int numParticles = Mathf.Min (m\_PointCloudData.Length, maxPointsToShow);

ParticleSystem.Particle[] particles = new ParticleSystem.Particle[numParticles];

int index = 0;

foreach (Vector3 currentPoint in m\_PointCloudData) {

particles [index].position = currentPoint;

particles [index].startColor = new Color (1.0f, 1.0f, 1.0f);

particles [index].startSize = particleSize;

index++;

}

currentPS.SetParticles (particles, numParticles);

} else {

ParticleSystem.Particle[] particles = new ParticleSystem.Particle[1];

particles [0].startSize = 0.0f;

currentPS.SetParticles (particles, 1);

}

frameUpdated = false;

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ModelController : MonoBehaviour

{

public void ScaleBigger() {

transform.localScale += new Vector3 (1, 1, 1);

}

public void ScaleSmaller() {

if (transform.localScale.x > 1) {

transform.localScale -= new Vector3 (1, 1, 1);

}

}

public void RotateRight(){

transform.RotateAroundLocal (new Vector3 (0, 1, 0), 5);

}

public void RotateLeft(){

transform.RotateAroundLocal (new Vector3 (0, 1, 0), -5);

}

}