«SLAM технологія в доповненій реальності»

Додаток Б

Лістинг програми «AR Mersedes»

Листів 7

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

Черкаси, 2018

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class ScaleModel : MonoBehaviour {

public Material CarMat;

public Animator LeftDoor;

public Animator RightDoor;

public Animator Hood;

public GameObject LeftLights;

public GameObject GightLights;

public Material BackLights;

public GameObject ColorPanel;

public bool isPlus = false;

public bool isMinus = false;

// Use this for initialization

void Start () {

}

// Update is called once per frame

void Update () {

if (isPlus)

{

transform.localScale = new Vector3 (transform.localScale.x + 0.01f, transform.localScale.y + 0.01f, transform.localScale.z + 0.01f);

}

if (isMinus)

{

if (transform.localScale.x >= 0.04f) {

transform.localScale = new Vector3 (transform.localScale.x - 0.01f, transform.localScale.y - 0.01f, transform.localScale.z - 0.01f);

}

}

}

public void PlusScale()

{

isPlus = true;

}

public void StopPlusScale()

{

isPlus = false;

}

public void MinusScale()

{

isMinus = true;

}

public void StopMinusScale()

{

isMinus = false;

}

public void OpenDoors()

{

LeftDoor.SetTrigger ("OpenLeftDoor");

RightDoor.SetTrigger ("OpenRightDoor");

}

public void CloseDoors()

{

LeftDoor.SetTrigger ("CloseLeftDoor");

RightDoor.SetTrigger ("CloseRightDoor");

}

public void OpenHood ()

{

Hood.SetTrigger ("OpenHood");

}

public void CloseHood()

{

Hood.SetTrigger ("CloseHood");

}

public void OpenColorPanel()

{

ColorPanel.SetActive (true);

}

public void CloseColorPanel()

{

ColorPanel.SetActive (false);

}

public void TurnOnLights()

{

LeftLights.SetActive (true);

GightLights.SetActive (true);

BackLights.color = new Color32 (255,0,0,255);

}

public void TurnOffLights()

{

LeftLights.SetActive (false);

GightLights.SetActive (false);

BackLights.color = new Color32 (255,0,0,75);

}

public void PlusSize()

{

transform.localScale = new Vector3 (transform.localScale.x + 0.02f, transform.localScale.y + 0.02f, transform.localScale.z + 0.02f);

}

public void MinusSize()

{

if (transform.localScale.x >= 0.04f) {

transform.localScale = new Vector3 (transform.localScale.x - 0.02f, transform.localScale.y - 0.02f, transform.localScale.z - 0.02f);

}

}

public void Red()

{

CarMat.color = new Color32 (219,17,16,255);;

}

public void Blue()

{

CarMat.color = new Color32(16,48,157,255);

}

public void White()

{

CarMat.color = new Color32 (240,240,240,255);;

}

public void Black()

{

CarMat.color = new Color32(25,25,25,255);

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.XR.iOS;

using UnityEngine.UI;

public class PointCloudParticleExample : MonoBehaviour {

public ParticleSystem pointCloudParticlePrefab;

public int maxPointsToShow;

public float particleSize = 1.0f;

public AudioController AudioController;

private Vector3[] m\_PointCloudData;

private bool frameUpdated = false;

private ParticleSystem currentPS;

private ParticleSystem.Particle [] particles;

private bool \_isDetectedOnes = true;

// 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);

if (\_isDetectedOnes) {

AudioController.SurfaceDetected ();

}

\_isDetectedOnes = false;

} else {

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

particles [0].startSize = 0.0f;

currentPS.SetParticles (particles, 1);

}

frameUpdated = false;

}

}

public void HideParticles() {

currentPS.Clear (true);

}

}

using System;

using System.Collections.Generic;

using UnityEngine.EventSystems;

namespace UnityEngine.XR.iOS

{

public class UnityARHitTestExample : MonoBehaviour

{

public Transform m\_HitTransform;

public GameObject Car;

public GameObject PlusSize;

public GameObject MinusSize;

public GameObject Remote;

public GameObject PointCloudParticleExample;

public AudioController AudioController;

private bool \_isPlacedOnes = true;

private bool \_isPlaced = false;

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 () {

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

{

var touch = Input.GetTouch(0);

if (!\_isPlaced) {

if (touch.phase == TouchPhase.Began && !EventSystem.current.IsPointerOverGameObject (0)) {

\_isPlaced = true;

if (\_isPlacedOnes) {

AudioController.CarPlaced ();

}

\_isPlacedOnes = false;

PointCloudParticleExample.GetComponent<PointCloudParticleExample> ().HideParticles ();

PointCloudParticleExample.SetActive (false);

Car.SetActive (true);

PlusSize.SetActive (true);

MinusSize.SetActive (true);

Remote.SetActive (true);

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;

}

}

}

}

}

}

}

}