Отчет по прохождению курса “ Курс по дополненной реальности для начинающих (ARCore)”

Выполнил: студент группы 4210 Гауиш М.Г.

Руководство разработчика:

1. Установить Unity версии 2021.3.21f1

2. Создать в Unity 3D проект

3. В настройках Build ставим платформу андроид

4. Заходим в Player settings

5. Меняем имя компании/продукта

6. Убираем Vulcan graphics API

7. Убираем галочку Multithreaded rendering

8. Ставим минимальную версию андроид 7

9. Заходим в Package manager

10. Загружаем AR Foundations

11. Загружаем ARCore

12. Удаляем стандартную камеру

13. Добавляем AR session origin и AR Session

14. AR camera ставим тэг Main Camera

15. В AR session origin добавляем компонент AR Plane Manager

16. Создаем пустой объект называем ARPlane в нулевой позиции

1. Добавляем ему компоненты AR Plane и AR Plane Mesh Visualizer
2. Добавляем ему Mesh collider, renderer, filter и Line renderer
3. Line renderer width ставим 0.02 и ставим цвет границы
4. Ставим углы Line Renderer по 5, убираем галочку на use world space и убираем тени
5. Добавляем стандартный материал
6. Создаем папку для материалов
7. Создаем материал ARPlane
8. Делаем его прозрачным, и альфа канал ставим в 50
9. В Mesh renderer закидываем этот материал
10. Закидываем полученный ARPlane в AR Session origin

17. Создаем маркер

1. Создаем 3d plane
2. Скачиваем или сами рисуем круглый маркер
3. Создаем материал для маркера
4. Ставим материал Unlit\transparent для прозрачности
5. Выбираем наш маркер и ставим его в текстуру
6. Применяем маркер к плоскости
7. Ставим подходящий размер

18. Создаем трассировку лучей для маркера

* 1. Добавляем в AR session origin –AR Reycast Manager
  2. Создаем папку для скриптов
  3. Создаем скрипт ProgrammManager
  4. Пишем код:

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.XR.ARFoundation;

using UnityEngine.XR.ARSubsystems;

public class ProgrammManager : MonoBehaviour

{

private ARRaycastManager ARRaycastManagerScript;

List<ARRaycastHit> hits = new List<ARRaycastHit>();

[Header("Put your planeMarker here")]

[SerializeField] private GameObject PlaneMarkerPrefab;

public GameObject ObjectToSpawn;

[Header("Put ScrollView here")]

public GameObject ScrollView;

private GameObject SelectedObject;

[SerializeField] GameObject MaketShell;

[SerializeField] private GameObject EndText;

[SerializeField] private Camera ARCamera;

private Vector2 TouchPosition;

private Quaternion YRotation;

public bool ChooseObject = false;

public bool Rotation;

public bool Recharging;

public int Strikes;

void Start()

{

ARRaycastManagerScript = FindObjectOfType<ARRaycastManager>();

EndText.SetActive(false);

PlaneMarkerPrefab.SetActive(false);

ScrollView.SetActive(false);

}

// Update is called once per frame

void Update()

{

if (ChooseObject)

{

ShowMarkerAndSetObject();

}

MoveObjectAndRotation();

if (Strikes > 2)

{

EndText.SetActive(true);

}

if (Recharging)

{

MaketShell.SetActive(false);

}

else

{

MaketShell.SetActive(true);

}

}

void ShowMarkerAndSetObject()

{

List<ARRaycastHit> hits = new List<ARRaycastHit>();

ARRaycastManagerScript.Raycast(new Vector2(Screen.width / 2, Screen.height / 2), hits, TrackableType.Planes);

// show marker

if (hits.Count > 0)

{

PlaneMarkerPrefab.transform.position = hits[0].pose.position;

PlaneMarkerPrefab.SetActive(true);

}

// set object

if (Input.touchCount > 0 && Input.touches[0].phase == TouchPhase.Began)

{

Instantiate(ObjectToSpawn, hits[0].pose.position, ObjectToSpawn.transform.rotation);

MaketShell = GameObject.Find("MaketShell");

ChooseObject = false;

PlaneMarkerPrefab.SetActive(false);

}

}

void MoveObjectAndRotation()

{

if(Input.touchCount > 0)

{

Touch touch = Input.GetTouch(0);

TouchPosition = touch.position;

// Select object

if (touch.phase == TouchPhase.Began)

{

Ray ray = ARCamera.ScreenPointToRay(touch.position);

RaycastHit hitObject;

if (Physics.Raycast(ray, out hitObject))

{

if (hitObject.collider.CompareTag("UnSelected"))

{

hitObject.collider.gameObject.tag = "Selected";

}

}

}

SelectedObject = GameObject.FindWithTag("Selected");

if (touch.phase == TouchPhase.Moved && Input.touchCount == 1 )

{

// Rotate object with one finger

if (Rotation)

{

YRotation = Quaternion.Euler(0f, -touch.deltaPosition.x \* 0.1f, 0f);

SelectedObject.transform.rotation = YRotation \* SelectedObject.transform.rotation;

}

// Move Object

else

{

ARRaycastManagerScript.Raycast(TouchPosition, hits, TrackableType.Planes);

SelectedObject.transform.position = hits[0].pose.position;

}

}

// Rotate object with 2 fingers

if (Input.touchCount == 2)

{

Touch touch1 = Input.touches[0];

Touch touch2 = Input.touches[1];

if (touch1.phase == TouchPhase.Moved || touch2.phase == TouchPhase.Moved)

{

float DistanceBetweenTouches = Vector2.Distance(touch1.position, touch2.position);

float prevDistanceBetweenTouches = Vector2.Distance(touch1.position - touch1.deltaPosition, touch2.position - touch2.deltaPosition);

float Delta = DistanceBetweenTouches - prevDistanceBetweenTouches;

if (Mathf.Abs(Delta) > 0)

{

Delta \*= 0.1f;

}

else

{

DistanceBetweenTouches = Delta = 0;

}

YRotation = Quaternion.Euler(0f, -touch1.deltaPosition.x \* Delta, 0f);

SelectedObject.transform.rotation = YRotation \* SelectedObject.transform.rotation;

}

}

// Deselect object

if (touch.phase == TouchPhase.Ended)

{

if (SelectedObject.CompareTag("Selected"))

{

SelectedObject.tag = "UnSelected";

}

}

}

}

}

1. Установить модели катапульты и целей
   1. Перенести модели в папку моделей
   2. Добавить материалы и раскрасить катапульту и цели
   3. Сделать из моделей префабы
2. Дописываем скрипт
3. Добавляем окно выбора
   1. Добавляем Scroll View
   2. Убираем галку на горизонтальном списке
   3. Movement type ставим clamped
   4. Чувствительность на 25
   5. Убираем горизонтальный ползунок
   6. Spacing на 1
   7. Подогнать размеры под экран
   8. Добавляем в Content – vertical layout group и content filter size
   9. Добавляем Кнопку катапульты и мишени
   10. Ставим кнопкам картинки
   11. Пишем скрипт Choose object

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class ChooseObject : MonoBehaviour

{

private ProgrammManager ProgrammManagerScript;

private Button button;

public GameObject ChoosedObject;

void Start()

{

ProgrammManagerScript = FindObjectOfType<ProgrammManager>();

button = GetComponent<Button>();

button.onClick.AddListener(ChooseObjectFunction);

}

void ChooseObjectFunction()

{

ProgrammManagerScript.ObjectToSpawn = ChoosedObject;

ProgrammManagerScript.ChooseObject = true;

ProgrammManagerScript.ScrollView.SetActive(false);

}

}

1. Ставим ориентацию приложения горизонтальной в настройках
2. Добавляем кнопку добавить объект
3. Накидываем на кнопку скрипт AddObject

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class AddObject : MonoBehaviour

{

private Button button;

private ProgrammManager ProgrammManagerScript;

// Start is called before the first frame update

void Start()

{

ProgrammManagerScript = FindObjectOfType<ProgrammManager>();

button = GetComponent<Button>();

button.onClick.AddListener(AddObjectFunction);

}

// Update is called once per frame

void AddObjectFunction()

{

ProgrammManagerScript.ScrollView.SetActive(true);

}

}

1. Добавляем тэги selected и unselected
2. Добавляем катапульте и мишени Rigid body и Box collider
3. Добавить кнопку вращения объекта
4. Написать скрипт Rotation

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class Rotation : MonoBehaviour

{

private Button Button;

private ProgrammManager ProgrammManagerScript;

// Start is called before the first frame update

void Start()

{

ProgrammManagerScript = FindObjectOfType<ProgrammManager>();

Button = GetComponent<Button>();

Button.onClick.AddListener(RotationFunction);

}

// Update is called once per frame

void RotationFunction()

{

if (ProgrammManagerScript.Rotation)

{

ProgrammManagerScript.Rotation = false;

GetComponent<Image>().color = Color.red;

}

else

{

ProgrammManagerScript.Rotation = true;

GetComponent<Image>().color = Color.green;

}

}

}

1. Добавляем меши для катапульты
   1. Выбираем рычаг
   2. Добавляем ей mash collider
   3. Выбираем мэш из списка
   4. Добавляем rigid body
2. Прикрепляем рычаг к катапульте при помощи hinge joint
3. Добавляем коллайдеры для остальных частей
4. Добавляем кнопку Огня
5. Пишем скрипт для кнопки

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class FireScript : MonoBehaviour

{

private ProgrammManager ProgrammManagerScript;

private Button button;

private GameObject Beam;

private Rigidbody BeamRigidBody;

public float force;

// Start is called before the first frame update

void Start()

{

button = GetComponent<Button>();

button.onClick.AddListener(FireFunction);

ProgrammManagerScript = FindObjectOfType<ProgrammManager>();

}

// Update is called once per frame

void FireFunction()

{

Beam = GameObject.Find("Beam");

BeamRigidBody = Beam.GetComponent<Rigidbody>();

if (!ProgrammManagerScript.Recharging)

{

BeamRigidBody.AddForce(BeamRigidBody.transform.up \* force, ForceMode.Impulse);

ProgrammManagerScript.Recharging = true;

}

}

}

1. Пишем скрипт для запуска снаряда

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class ThrowShell : MonoBehaviour

{

private ProgrammManager ProgrammManagerScript;

private TrajectoryRenderer TrajectoryRenderScript;

[SerializeField] private GameObject ShellPrefab;

private GameObject ShellObject;

private Rigidbody ShellRigidBody;

private Vector3 speed;

private GameObject fieldobject;

private InputField filed;

private string ForceString;

private int force;

private Rigidbody CollisionRigidBody;

public AudioClip ThrowSound;

private AudioSource CatapultAudio;

// Start is called before the first frame update

void Start()

{

ProgrammManagerScript = FindObjectOfType<ProgrammManager>();

TrajectoryRenderScript = FindObjectOfType<TrajectoryRenderer>();

fieldobject = GameObject.Find("InputField");

filed = fieldobject.GetComponent<InputField>();

}

// Update is called once per frame

void Update()

{

ForceString = filed.text;

force = Int32.Parse(ForceString);

speed = transform.forward \* 2 + transform.up \* force;

TrajectoryRenderScript.ShowTrajectory(transform.position + new Vector3(0, 0.25f, 0), speed);

}

private void OnCollisionEnter(Collision collision)

{

ShellObject = Instantiate(ShellPrefab, transform.position + new Vector3(0, 0.25f, -0.05f), ShellPrefab.transform.rotation);

ShellRigidBody = ShellObject.GetComponent<Rigidbody>();

ShellRigidBody.AddForce(speed, ForceMode.Impulse);

CollisionRigidBody = collision.rigidbody;

CollisionRigidBody.AddForce(CollisionRigidBody.transform.up \* (-1), ForceMode.Impulse);

ProgrammManagerScript.Recharging = true;

CatapultAudio = GetComponent<AudioSource>();

CatapultAudio.PlayOneShot(ThrowSound, 1.0f);

}

}

1. Создаем ядро с Rigid body и Collider
2. Добавляем поле ввода силы запуска
3. Сделать линию траектории
   1. Создать plane
   2. Накинуть Line renderer
   3. Сделать толщину 0.01
   4. Ставим белый материал
   5. Пишем скрипт для траектории

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class TrajectoryRenderer : MonoBehaviour

{

private LineRenderer LineRendererComponent;

[SerializeField] private GameObject Catapult;

// Start is called before the first frame update

void Start()

{

LineRendererComponent = GetComponent<LineRenderer>();

}

private void Update()

{

Catapult = GameObject.Find("Catapult(Clone)");

}

public void ShowTrajectory(Vector3 origin, Vector3 speed)

{

Vector3[] points = new Vector3[100];

LineRendererComponent.positionCount = points.Length;

for (int i = 0; i < points.Length; i++)

{

float time = i \* 0.1f;

points[i] = origin + speed \* time + Physics.gravity \* time \* time / 2;

if (points[i].y < Catapult.gameObject.transform.position.y - 1)

{

LineRendererComponent.positionCount = i;

break;

}

}

LineRendererComponent.SetPositions(points);

}

}

1. Добавляем звук выстрела
   1. Добавляем камере audio listener
   2. Добавляем на катапульту источник звук
2. Создаем иконку победу
   1. Создаем иконку ТЫ ПОБЕДИЛ
   2. Добавляем в основной скрипт счетчик попаданий
   3. Добавляем скрпит для целей

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Strike : MonoBehaviour

{

private ProgrammManager ProgrammManagerScript;

private bool killed = false;

private void Start()

{

ProgrammManagerScript = FindObjectOfType<ProgrammManager>();

}

private void OnCollisionEnter(Collision collision)

{

if (!killed && collision.gameObject.name == "Shell(Clone)")

{

ProgrammManagerScript.Strikes += 1;

killed = true;

}

}

}

