МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 43

ОТЧЁТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Асс. | |  |  | | | |  | | Е. Е. Майн |
| должность, уч. степень, звание | |  | подпись, дата | | | |  | | инициалы, фамилия |
| ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №6 | | | | | | | | | | |
| ДОПОЛНЕННАЯ РЕАЛЬНОСТЬ С ИСПОЛЬЗОВАНИЕМ ТРЕКЕРОВ VIVE TRAKER | | | | | | | | | | |
| по дисциплине: ВИРТУАЛЬНАЯ И ДОПОЛНЕННАЯ РЕАЛЬНОСТЬ | | | | | | | | | | |
|  | | | | | | | | | | |
| РАБОТУ ВЫПОЛНИЛА | | | | | | | | | | |
| СТУДЕНТКА ГР. | 4931 | | |  | 30.11.2022 |  | | Е.Ю. Ильченко | | |
|  |  | | |  | подпись, дата |  | | инициалы, фамилия | | |
|  |  | | |  |  |  | |  | | |

Санкт-Петербург 2022

# **Цель работы**

Получение практических навыков подготовки сцены Unity для манипуляции над 3D-моделью объекта с помощью HTC Vive трекера. Ознакомление с методами генерации событий при столкновении коллайдеров объектов.

Порядок выполнения лабораторной работы  
1) Подключить к ПК USB-ретранслятор.  
2) В SteamVR подключить трекер VIVE.  
3) Создать в 3D-редакторе объект, повторяющий по форме и размеру  
геометрический объект по варианту (из таблицы 1).  
4) Открыть копию проекта Unity, который был создан при выполнении  
лабораторной работы No2 (проект для шлема VIVE Pro).  
5) Импортировать созданный объект в формате FBX в Unity, настроить  
материал.  
6) Применить и настроить вручную скрипт отслеживания трекера VIVE.  
7) Применить на созданный 3D-объект скрипт автоматической выдачи ID.  
8) Установить трекер VIVE на геометрический объект по варианту (из  
таблицы 1).  
9) Запустить проект, проверить правильность отслеживания перемещения  
и вращения 3D-объекта относительно физического макета.  
10) Реализовать взаимодействие объектов с использованием коллайдеров.  
11) Сформулировать выводы по проделанной работе.  
12) Продемонстрировать результат преподавателю.  
13) Оформить отчёт.

Вариант

|  |  |
| --- | --- |
| Вариант | 2 |
| Геометрический объект | Высокий цилиндр |
| Действие | При столкновении с другим объектом отобразить частицы |

# **Подготовка проекта**

Создали новый проект с пресетом 3D и провели все  
подготовительные операции для работы с ARCore, как описано в ЛР №2. Удалили из сцены камеру «Main Camera» (рис.1).

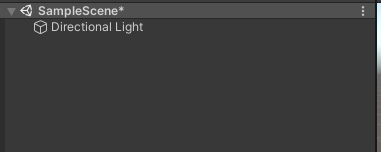


Рис.1 – Удаление «Main Camera»

Добавили в сцену объекты «AR Session Origin» и «AR Session» (рис.2).

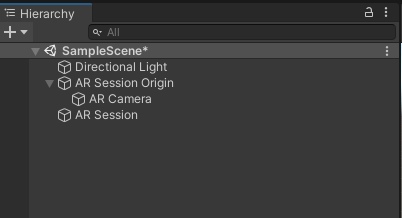


Рис.2 – Добавление в сцену AR – объектов

Добавили к камере «AR Camera», которая находится в дочерних  
элементах объекта «AR Session Origin» тэг «MainCamera» (рис.3).

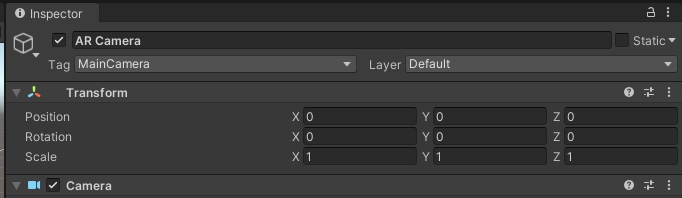


Рис.3 – Подготовка AR-камеры

# **Создание и визуализация плоскости пространства**

Добавили к объекту «AR Session Origin» компонент «AR Plane Manager».Создали пустой игровой объект под названием «Plane» в начале координат. Добавили к созданной "пустой" плоскости стандартные скрипты ARCore «AR Plane» и «AR Plane Mesh Visualizer» (см. рис. 4). Также добавили к плоскости стандартные компоненты «Mesh Filter», «Mesh Renderer» и «Mesh Collider».

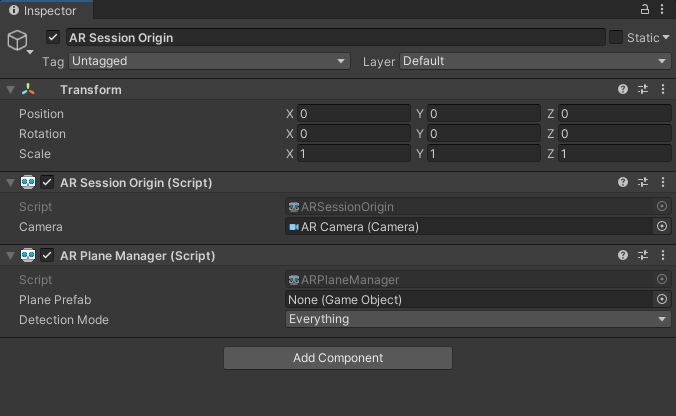


Рис.4 – Добавление компонентов для работы с плоскостями

Добавили компонент «Line Renderer», который понадобится для обрисовки контура плоскости линиями (см. рис. 5).

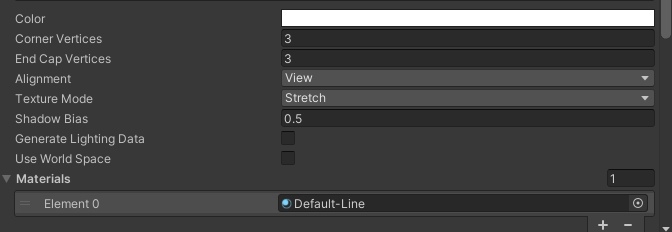


Рис.5 – Настройка параметров контурной линии

Создали материал для плоскости «plane\_mat» и добавили его на плоскость(см.рис.6).

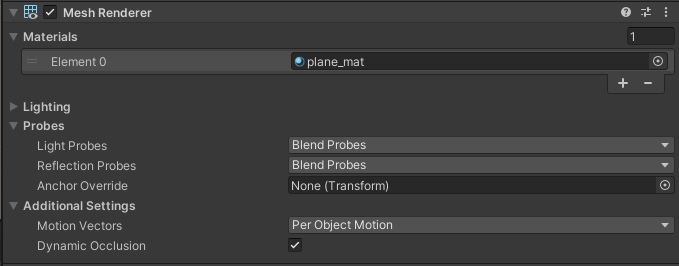


Рис.6 – Добавление материала

Изменили режим визуализации материала (параметр «Rendering Mode») на прозрачный («Transparent») и в настройках цвета установили значение альфа-канала на 50 (рис.7).

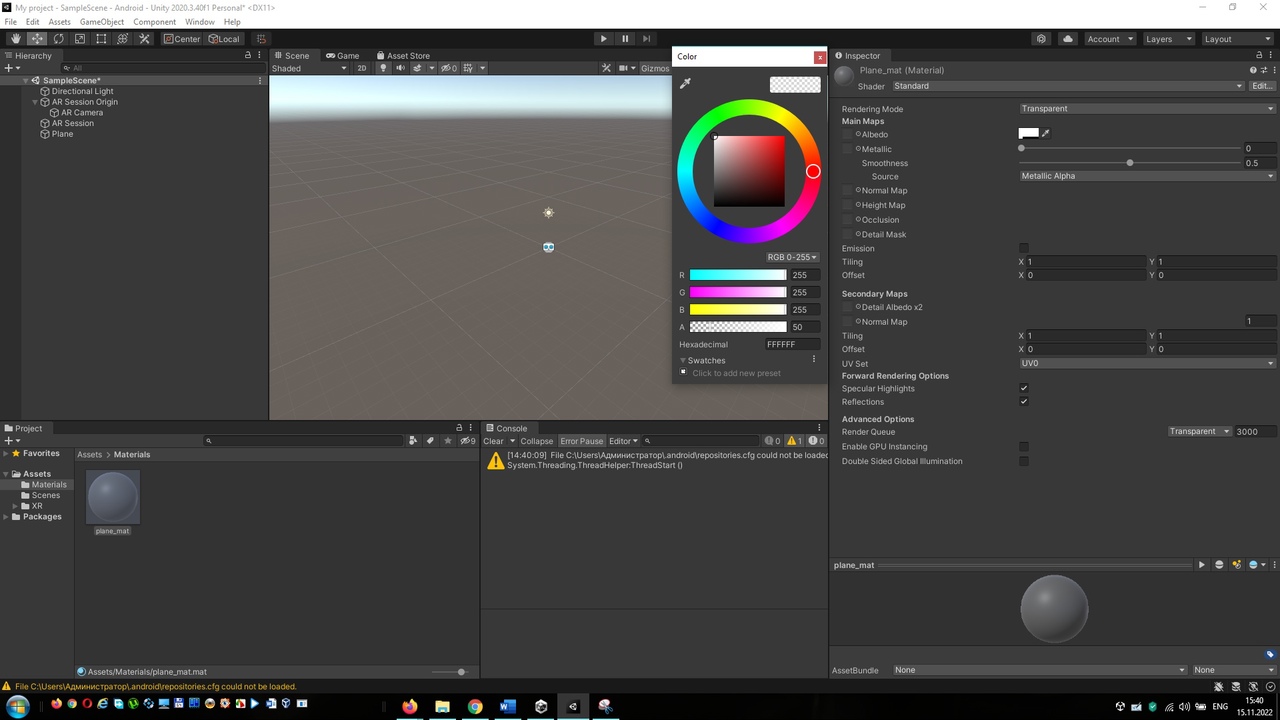


Рис.7 – Настройка материала для плоскости

Сделали билд и проверили полученный результат на телефоне. Теперь камера телефона сканирует пространство, а скрипт «AR Plane Manager» рисует на столе плоскость из префаба «Plane» (рис.8).



Рис.8 – Рисование плоскостей в дополненной реальности

# **Добавление навигационной метки**

Создали примитив-плоскость («GameObject –> 3D Object –> Plane»). Назвали её «NavMark». Поскольку метка – это плоская фигура, примитив-плоскость тут подойдёт как раз кстати. Создали материал «navmark\_mat» и применили к нему спрайт – 2D-картинку с альфа-каналом. Поместили спрайт в папку «Textures» в ассетах проекта и изменили тип картинки с «Default» на «Sprite (2D and UI)» (рис.9)

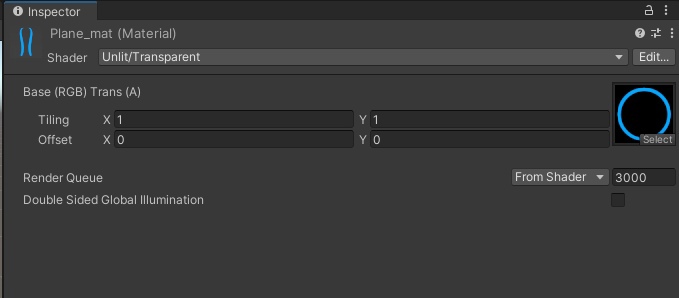


Рис.9 – Подготовка спрайта для навигационной метки

Переместили спрайт в канал «Albedo» материала и изменили тип отображения (Rendering Mode) на прозрачный (Transparent). Изменили тип шейдера материала со стандартного (Standard) на неосвещённый/прозрачный (Unlit/Transparent) (рис.10).

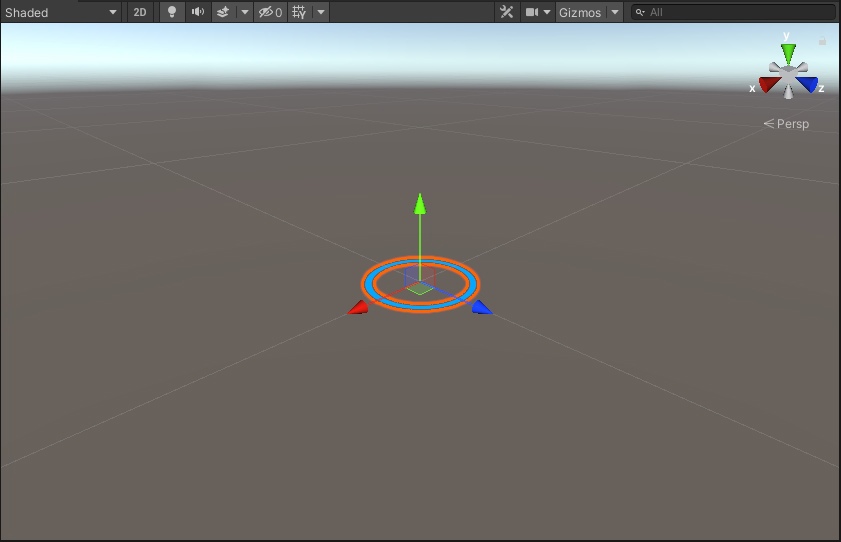


Рис.10 – Готовая навигационная метка

# **Программирование отображения**

На объект «AR Session Origin» добавили компонент (скрипт) «AR Raycast Manager». Он отвечает за испускание лучей и детектирование их столкновений с преградами. Создади папку «Scripts» в ассетах проекта и добавьте в неё новый скрипт «SceneManager». Применили этот скрипт к объекту «AR Session Origin» (рис.11).

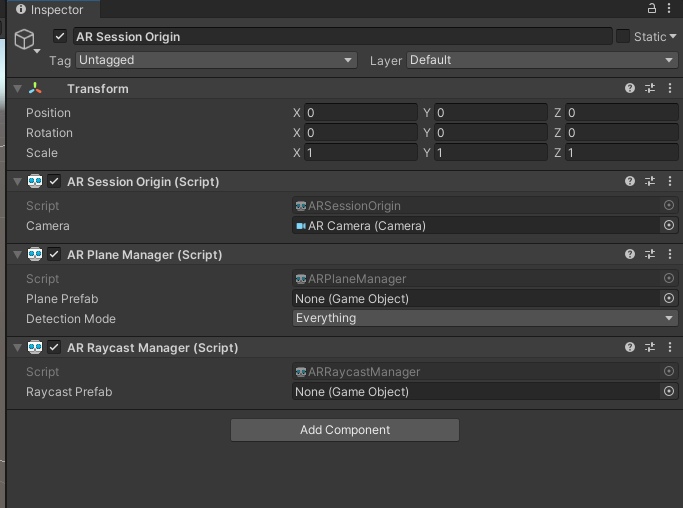


Рис.11 – Добавление скриптов

# **Размещение объектов на навигационной метке**

Дополнили метод «DisplayNavMark» из скрипта «SceneManager». В Инспекторе поместили объект «Sphere». Сделали предварительный билд, чтобы проверить результат работы скрипта. При каждом нажатии на экран в центре метки будет появляется  
сфера (рис.12.).

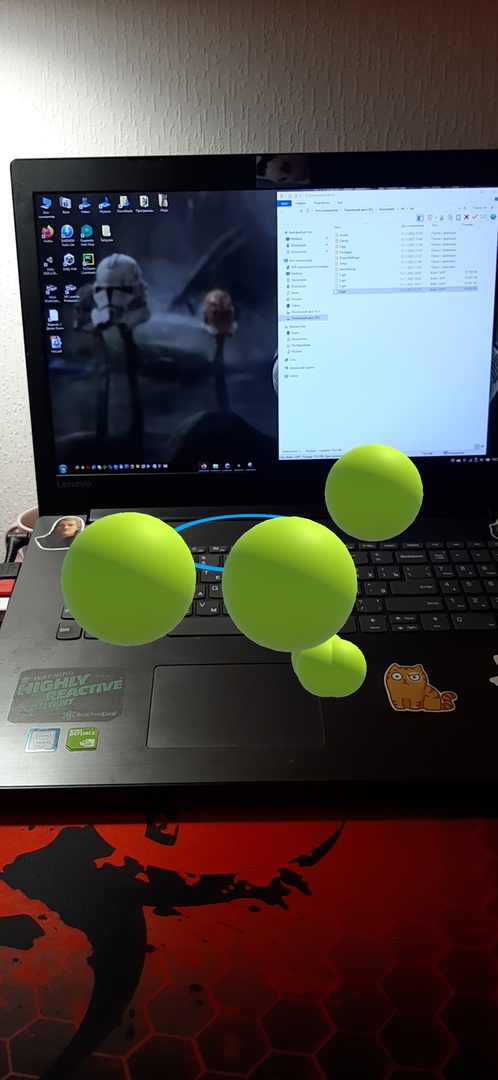


Рис.12 – Появление объектов на отметке при нажатии на экран

# **Выбор проецируемого объекта через UI**

Добавили в сцену две кнопки. Одна из них называется «SpawnCube» и отвечать за появление куба, а вторая – «SpawnSphere» – размещает в комнате примитив-сферу (рис.13).

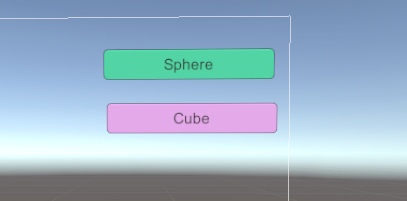


Рис.13 - Элементы UI для выбора объектов

Создали примитив-куб (Cube) в начале координат, наложили на него материал и установите масштаб 0.2. Сохранили куб в префаб. Создали новый скрипт «ObjectSelector» и применили его к обеим кнопкам. Подключили библиотеку UnityEngine.UI. Сделали билд APK-файла и установили его на телефон. Теперь мы можем выбирать, какой из двух примитивов следует разместить на метке при нажатии на экран телефона (рис. 14)

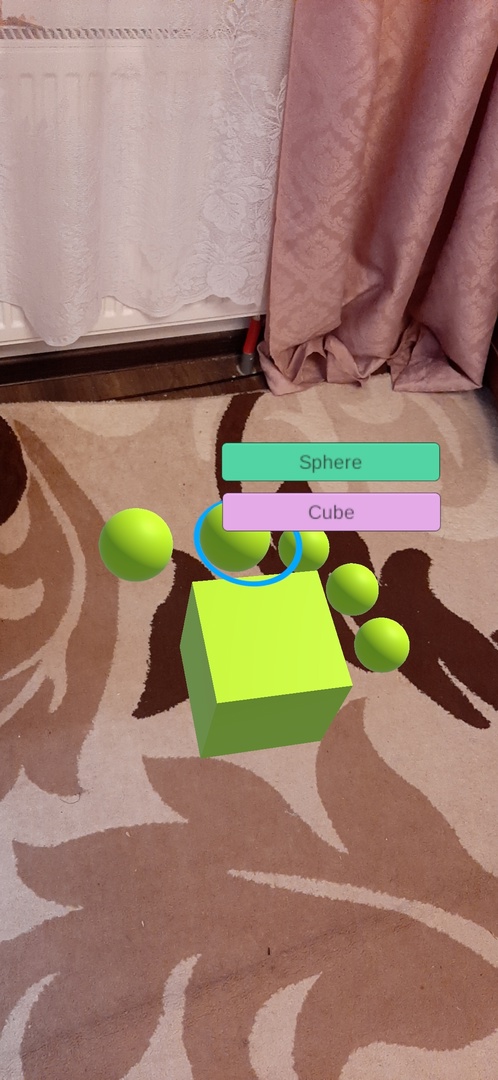


Рис.14 – Выбор объекта для размещения в AR

# **Вывод**

Мы ознакомились с технологией пространственного картирования ARCore. Изучили способы визуализации пространства и размещения объектов в дополненной реальности по навигационным меткам. Также научились создавать пользовательский интерфейс для взаимодействия с объектами.

# **ПРИЛОЖЕНИЕ 1**

Скрипт, реализующий функционал взаимодействия с объектом

using System.Collections.Generic;

using UnityEngine;

// Подключаем библиотеки для AR

using UnityEngine.XR.ARFoundation;

using UnityEngine.XR.ARSubsystems;

public class SceneManager : MonoBehaviour

{

// Объявляем переменную для навигационной метки

public GameObject navMark;

public GameObject primitiveObject;

// Объявляем переменную для менеджера лучей

private ARRaycastManager raycastScript;

void Start()

{

// Поиск и присваивание компонента менеджера рейкаста

raycastScript = GetComponent<ARRaycastManager>();

// Отключить отображение метки при старте

navMark.SetActive(false);

}

void DisplayNavMark()

{

// Список объектов (плоскостей), в которые попал луч.

List<ARRaycastHit> hits = new List<ARRaycastHit>(); // Запускаем луч из центра экрана.

// Информация помещается в контейнер (массив) hits // Фиксируем попадания луча на плоскости Planes

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

if (hits.Count > 0)

{

// Если луч пересёк плоскость, сделать метку снова

navMark.SetActive(true);

// Позиции метки присваивается значение места

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

// Если зафиксировано однократное нажатие на экран и фаза

if (Input.touchCount > 0 && Input.touches[0].phase == TouchPhase.Began &&!UnityEngine.EventSystems.EventSystem.current.IsPointerOverGameObject(0))

{

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

}

}

}

void Update()

{

DisplayNavMark();

}

}

## **ПРИЛОЖЕНИЕ 2**

Скрипт, реализующий функционал взаимодействия с объектом

using UnityEngine;

using UnityEngine.UI;

public class ObjectSelector : MonoBehaviour

{

// Объявляем переменную типа SceneManager

private SceneManager sceneManager;

// Объявляем переменную кнопки

private Button button;

// Какой объект выбран

public GameObject selectedObject;

void Start()

{

// Находим в проекте объект SceneManager и записываем его

sceneManager = FindObjectOfType<SceneManager>();

// Получаем кнопку, на которую наложен данный скрипт

button = GetComponent<Button>();

// Добавляем слушатель событий при нажатии на кнопку

button.onClick.AddListener(SelectObjectToSpawn);

}

void SelectObjectToSpawn()

{

// Передаём информацию о выбранном объекте в Scene

sceneManager.primitiveObject = selectedObject;

}

}