ГУАП

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | Д. А. Булгаков |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| ДОПОЛНЕННАЯ РЕАЛЬНОСТЬ С ПРОСТРАНСТВЕННЫМ КАРТИРОВНИЕМ |
| по курсу: Разработка виртуальной и дополненной реальности |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4941 |  |  |  | Н.С. Горбунов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

**Цель работы:** знакомство с технологией пространственного картирования ARCore. Изучение способов визуализации пространства и размещения объектов в дополненной реальности по навигационным меткам. Создание пользовательского интерфейса для взаимодействия с объектами.

**Подготовка к работе:**

Установка пакета дополнений ARCore. Скачав свежую версию пакета необходимо установить его в проект Unity через Package Manager.

В пакет входят такие плагины как:

* XR Plugin Management
* XR Legacy Input Helpers
* AR Foundation
* ARCore XR Plugin

Настройка проекта. В Build settings необходимо переключить платформу с ПК на Android. (рис. 1)

В Player Settings необходимо выставить настройки связанные с работой приложения. К примеру, выставить минимальную версию Android 7.0 Nougat, включить поддержку ARM64. (рис. 2)

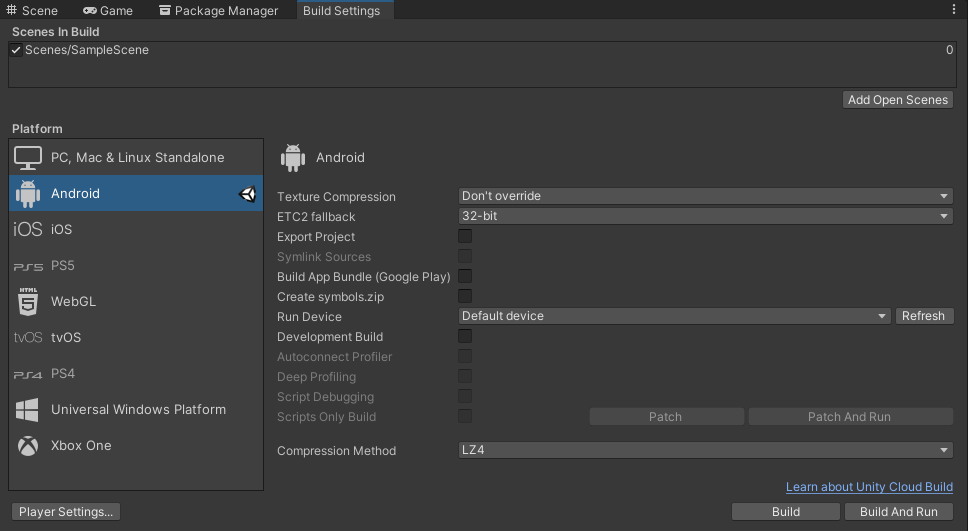


Рисунок 1. Build Settings

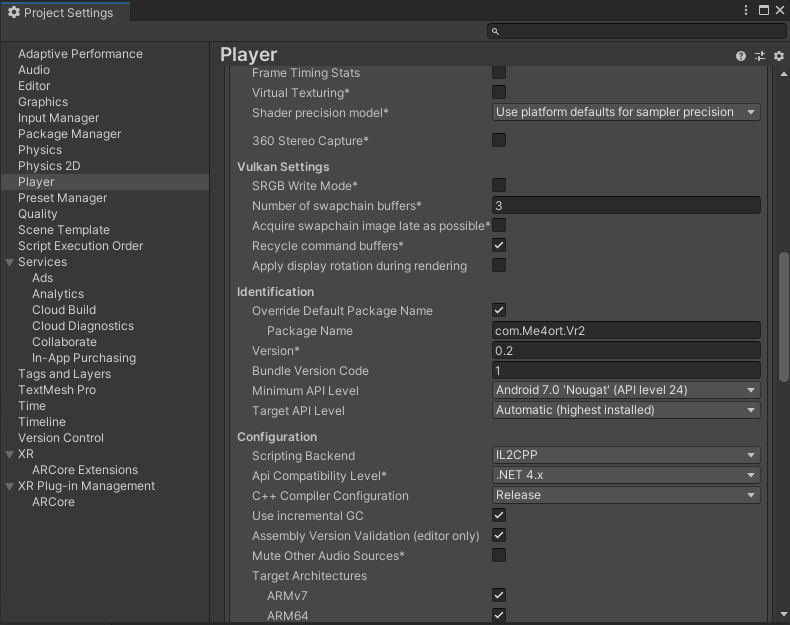


Рисунок 2. Player settings

Необходимы шагом является добавление специализированных объектов *AR Session Origin*  и *AR Session*. *AR Session Origin* устанавливает начало координат в дополненном окружении, а *AR Session* инициализирует сессию дополненной реальности. (рис. 3)

Для удобства работы нужно создать папки в директории Assets, для хранения элементов в процессе работы. (рис. 4)

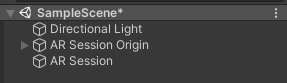


Рисунок 3. Окно иерархии с созданными объектами



Рисунок 4. Структура размещения файлов

**Выполнение работы**

К объекту **«AR Session Origin»** в окне Иерархии надо добавить компонент **«AR Plane Manager»**. Он отвечает за отображение плоскостей, которые AR-приложение строит на основании данных с камеры телефона. На построенную в дополненной реальности плоскость пользователь может помещать изображения или 3D-модели.

**Настройка отображения плоскости:**

К пустому обекту под названием **«Plane»** необходимо добавить стандартные скрипты ARCore «**AR Plane**» и «**AR Plane Mesh Visualizer**», а также стандартные компоненты «**Mesh Filter**», «**Mesh** **Renderer**» и «**Mesh Collider**». (рис. 5)

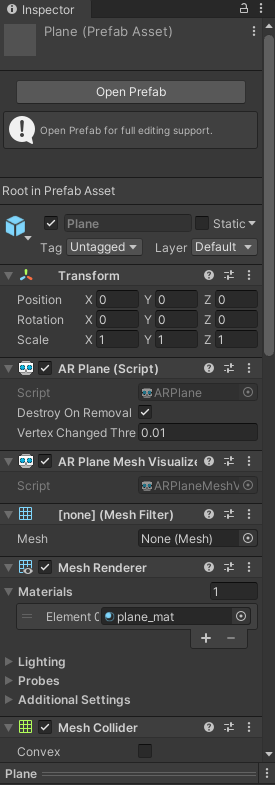


Рисунок 5. Объект «Plane»

«**Mesh** **Filter**» отвечает за построение геометрической сетки объекта. «**Mesh** **Renderer**» за её отображение на экране (визуализацию), а «**Mesh** **Collider**» добавляет "физическую" оболочку, т.е. позволяет обрабатывать коллизии с другими объектами сцены. Компонент «**Line** **Renderer**» понадобится для обрисовки контура плоскости линиями, лучше уменьшить ширину контурной линии с **1** до **0.02**, чтобы она не была слишком толстой. В этом компоненте надо настроить следующие параметры:

* задать желаемый цвет линии (Color);
* задать количество вершин на углах (Corner Vertices), чтобы они выглядели сглаженными;
* отключить использование мирового пространства (Use World Space);
* отключить тени (Lighting –> Cast Shadows) и (Lighting –> Receive Shadows);
* добавить стандартный материал линии (Materials –> Default-Line)

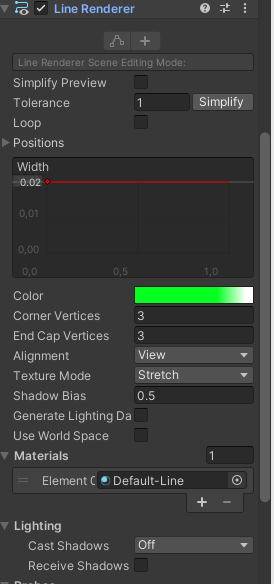


Рисунок 6. Настройка «Line Renderer»

Материал для линии должен быть полупрозрачным, чтобы не перекрывать собой объекты реального мира. Поэтому нужно изменить режим визуализации материала (параметр «**Rendering Mode**») на прозрачный («**Transparent**») и в настройках цвета установить значение альфа-канала на **50**. (рис.7)

Объект «**Plane**» необходимо сохранить как префаб и добавить в компонент «**AR Plane Manager**».

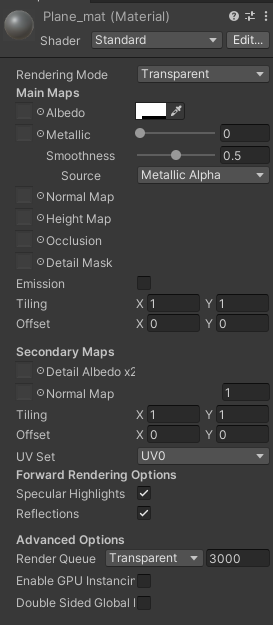


Рисунок 7. Настройка материала



Рисунок 8. Отображение плоскости



Рисунок 9. Отображение нескольких плоскостей

**Добавление навигационной метки**

В реальном проекте не нужно постоянно визуализировать плоскости, на которые пользователь мог бы проецировать 3D-объекты. Вместо этого проще сделать сами плоскости невидимыми, но добавить на них визуальную метку, которая будет показывать, куда можно поставить 3D-объект.

Примитив-плоскость «**NavMark**» станет маркером. На него надо наложить текстуру с изображением самой метки (рис. 10)

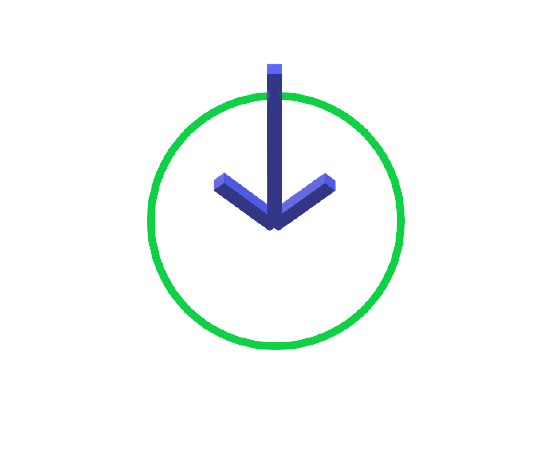


Рисунок 10. Изображение маркера

Спрайт необходимо поместить в папку проекта и изменить тип картинки с «**Default**» на «**Sprite** **(2D and UI)**». Спрайт переместить в канал «Albedo» материала и изменить тип отображения (Rendering Mode) на прозрачный (Transparent). Изменить тип шейдера материала со стандартного (Standard) на неосвещённый/прозрачный. (рис. 11)

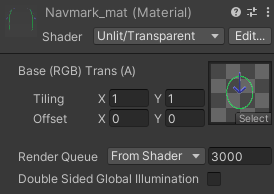


Рисунок 11. Настройки материала маркера

Это нужно, чтобы метка была полупрозрачной и не реагировала на освещение в сцене. (рис. 12)

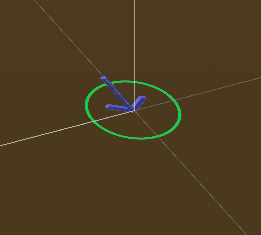


Рисунок 12. Отображение маркера в сцене

**Программирование отображения метки**

Для решения задачи корректного размещения метки на виртуальной плоскости так же подойдёт технология отбрасывания лучей (Raycast, рейкаст). Требуется выпустить луч из камеры и с его помощью засечь плоскость, после этого в месте контакта луча с плоскостью отображается навигационная метка. На объект «**AR Session Origin**» необходимо добавить компонент (скрипт) «**AR Raycast Manager**». Он отвечает за испускание лучей и детектирование их столкновений с преградами.

Далее добавить новый скрипт «**SceneManager**» и применить этот скрипт к объекту «**AR Session Origin**».

Листинг 1. SceneManager.cs

using System.Collections;

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 primitive;

// 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 видимой

navMark.SetActive(true);

// Позиции метки присваивается значение места пересечения луча с плоскостью

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

}

}

void Update()

{

DisplayNavMark();

}

}

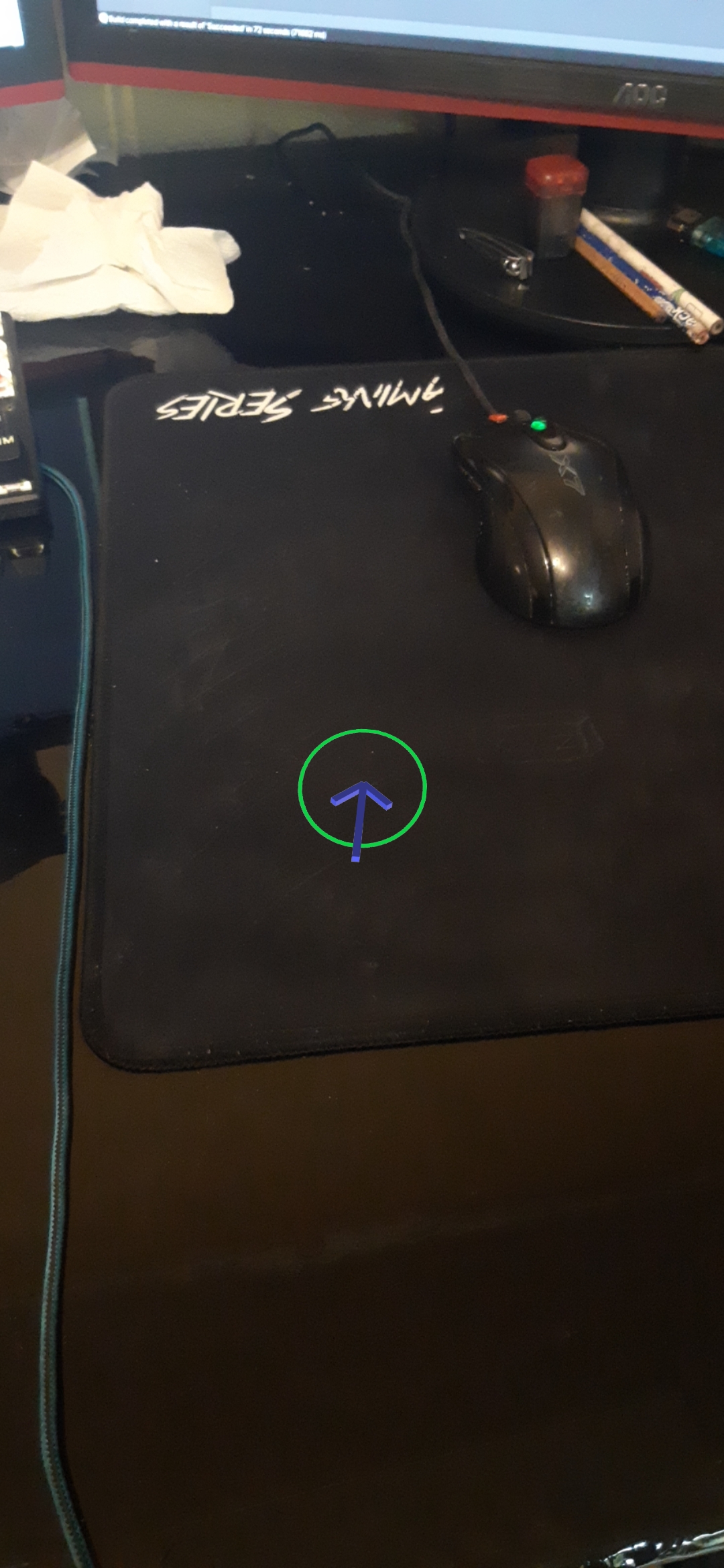


Рисунок 13. Отображение маркера в приложении

**Размещение объектов на навигационной метке**

Навигационная метка позволяет пользователю точнее контролировать место размещения объекта, поскольку визуально обозначает место в пространстве, где появится 3D-модель. В данном примере в качестве объекта используется префаб куба радиусом 0,02м. Необходимо изменить код скрипта «**SceneManager**»

Листинг 2.

using System.Collections;

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 primitive;

// 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 видимой

navMark.SetActive(true);

// Позиции метки присваивается значение места пересечения луча с плоскостью

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

}

// Если зафиксировано однократное нажатие на экран и фаза является начальной

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

{

// Создаём копию префаба объекта

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

}

}

void Update()

{

DisplayNavMark();

}

}

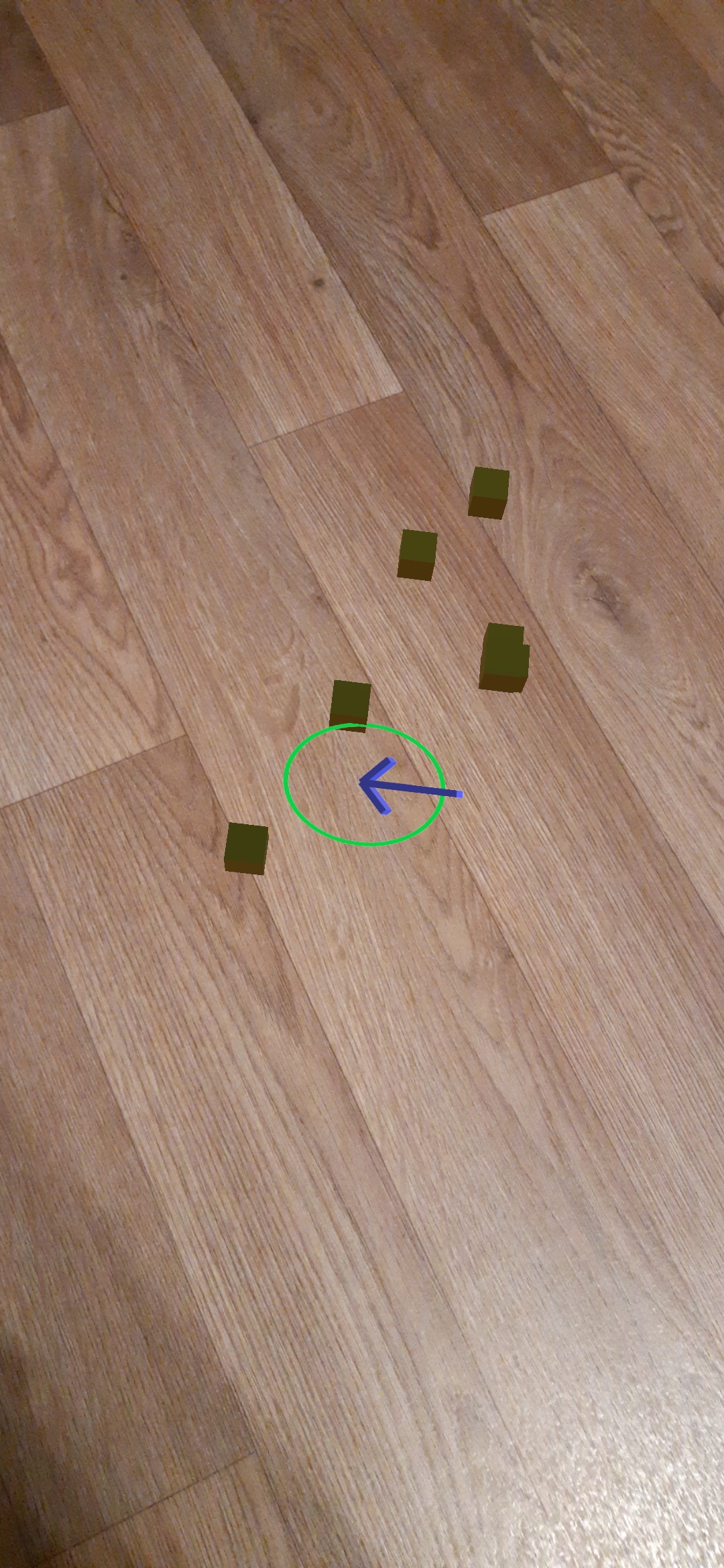


Рисунок 14. Создание объектов

**Выбор проецируемого объекта через UI**

В сцену необходимо добавить две кнопки («**UI –> Button**»). Одна из них будет называться «**SpawnCube**» и отвечать за появление куба, а вторая – «**SpawnSphere**» – размещать в комнате примитив-сферу.

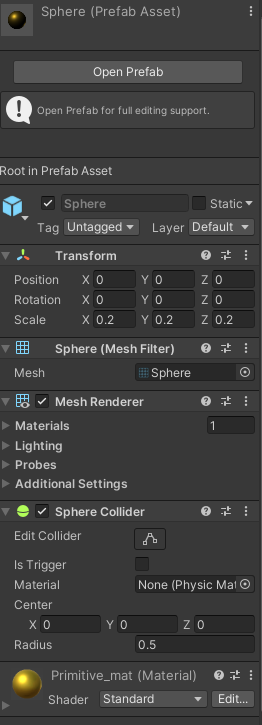


Рисунок 15. Префаб-сфера

В разделе просмотра «**Game**» можно настроить разрешение экрана (к примеру, 2560x1440 Portrait) и выставить кнопки по своему усмотрению.

Далее надо создать префаб-сферу наложить на неё материал и установить масштаб 0.2.

За обработку нажатия на кнопки будет отвечать скрипт «**ObjectSelector**». Его надо накинуть на каждую из кнопок.

Листинг 3.

using System.Collections;

using System.Collections.Generic;

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 в переменную

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 SceneManager

sceneManager.primitive = selectedObject;

}

}

В скрипт надо поместить префаб объекта, который необходимо создавать по нажатию.

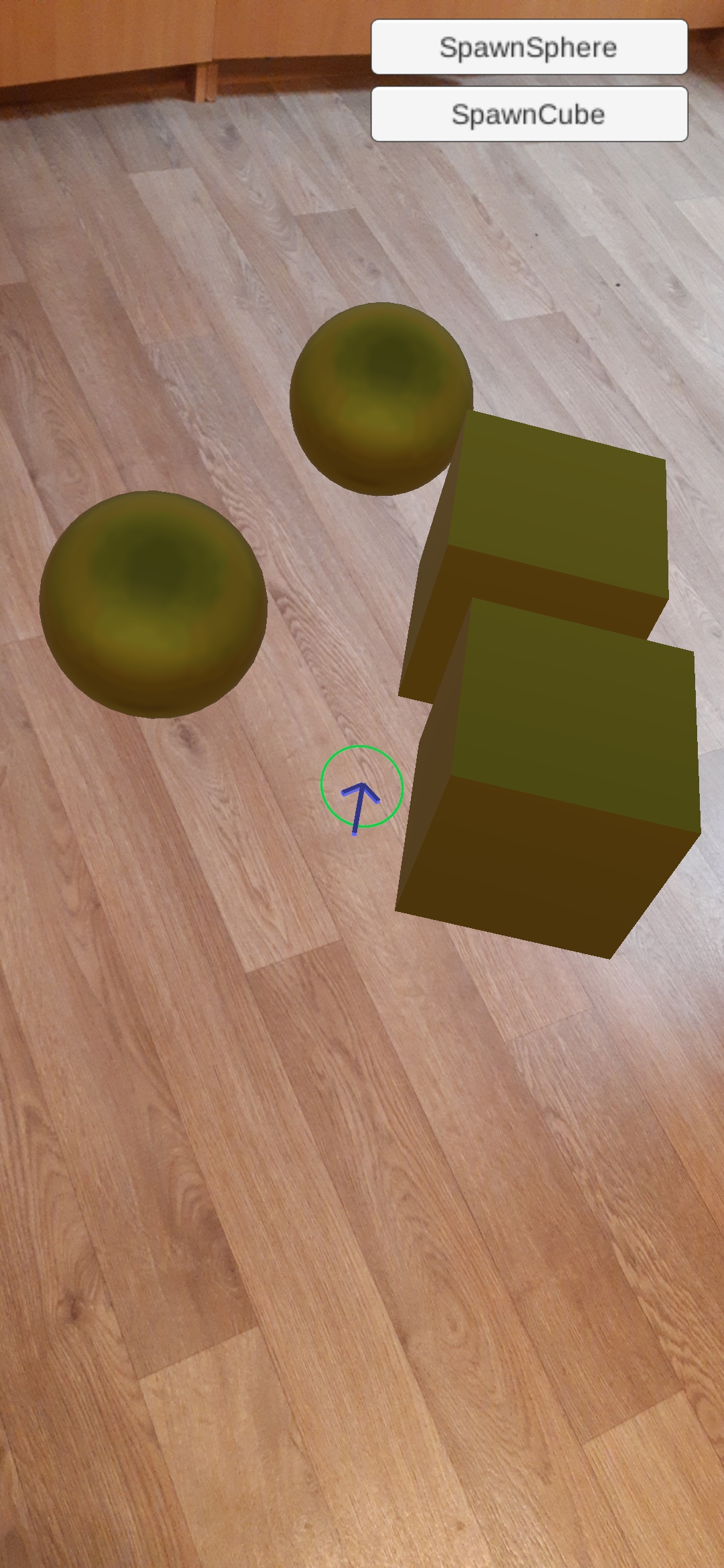


Рисунок 16. Результат лабораторной работы

Вывод: в результате выполнения лабораторной работы были изучены методы работы с плоскостями в AR. Проверены способы отображения формируемых плоскостей, отображение метки на плоскости с помощью Raycast, создание объектов в AR-окружении по нажатию на экран.