

Лабораторна робота №2

Технології просторової орієнтації в AR

Мета: Ознайомити з полями класу ARPlane для аналізу та класифікації відстежених поверхонь, отримати практичні навички із розпізнавання зображень у просторі та застосування події trackablesChanged класів ARPlaneManager та ARTrackedImageManager для відслідковування змін створених поверхонь та знайдених зображень, навчитись візуально перекривати 3D моделі реальними об'єктами із середовища.

Література

<https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@6.5/api/UnityEngine.XR.ARFoundation.ARPlane.html>

<https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@6.5/api/UnityEngine.XR.ARFoundation.ARTrackable.html>

<https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@6.5/api/UnityEngine.XR.ARFoundation.ARTrackableManager-5.html>

<https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@6.5/api/UnityEngine.XR.ARFoundation.AROcclusionManager.html>

Методичні рекомендації

Компонент ARPlaneManager пакунку ARFoundation створює поверхні на основі реального середовища, однак вони не являють собою прості 3D площини. Компонент ARPlane, що присутній в них, зберігає інформацію про їх орієнтацію в просторі та класифікацію, що дозволяє розрізняти їх під час виконання необхідних дій.

Клас ARPlane має поля, представлені в таблиці 1, що можуть стати в нагоді під час реалізації алгоритмів.

Таблиця 1. Поля класу ARPlane

Назва	Клас/Тип	Опис
alignment	PlaneAlignment (Enum)	Орієнтація (вирівнювання) площини в просторі. Зазвичай набуває значень HorizontalDown, HorizontalUp та Vertical. Поверхні, що не є вертикальними або горизонтальними, набувають значення NotAxisAligned.
boundary	NativeArray<Vector2>	Масив локальних координат вершин поверхні, розміщених на одній площині.
center	Vector3	Центр поверхні у координатах сцени.
centerInPlaneSpace	Vector2	Центр поверхні у її локальних координатах.

Продовження таблиці 1

Назва	Клас/Тип	Опис
<code>classifications</code>	<code>PlaneClassifications</code> (Enum)	Підтримується невеликим переліком пристроїв. Класифікація поверхні (підлога, стіл, стілець тощо), яка була визначена під час розпізнавання компонентом <code>ARPlaneManager</code> .
<code>infinitePlane</code>	<code>Plane</code>	Нескінченна площина, на якій лежить поверхня.
<code>normal</code>	<code>Vector3</code>	Нормаль поверхні.
<code>size</code>	<code>Vector2</code>	Довжина та ширина поверхні. У випадку вертикальної площини значення вектора <i>x</i> – висота, а значення <i>y</i> – ширина.
<code>subsumedBy</code>	<code>ARPlane</code>	Батьківська поверхня, яка поглинула поточну. Зазвичай, якщо значення <code>null</code> – поверхня все ще існує на ігровій сцені.
<code>trackableId</code>	<code>TrackableId</code>	Ідентифікатор поверхні з власним типом. Унаслідований від класу <code>ARTrackable</code> .
<code>transform</code>	<code>Transform</code>	Дані поверхні щодо положення, повороту, співвідношення тощо на ігровій сцені. Унаслідований від класу <code>MonoBehaviour</code> .
<code>vertexChangedThreshold</code>	<code>float</code>	Найбільша відстань в метрах, на яку може зміститись вершина.

Всі поля класу можуть пригодитись під час розробки AR додатку, однак найчастіше будуть враховуватись поля `alignment`, `boundary`, `size`, `trackableId` та `transform`, так як вони надають необхідну інформацію.

Методичні вказівки лабораторної роботи 1 (сторінки 12-13) описують отримання за допомогою методу `raycastManager.Raycast()` список зіткнень променю з поверхнями у вигляді класів `ARRaycastHit`. Отриманий клас містить поле `trackable`, значенням якого може стати клас `ARPlane`. Якщо виконати перевірку типу значення, з'явиться можливість зчитати необхідні дані та на їх основі виконати потрібні дії.

Приклад створення об'єктів на поверхнях, які є лише горизонтальними з орієнтацією вгору, представлений в лістингу нижче.

```
ARRaycastHit mainRayHit = rayHits[0];

if (mainRayHit.trackable is ARPlane plane)
{
    if (plane.alignment == PlaneAlignment.HorizontalUp)
    {
        Vector3 hitPosePosition = mainRayHit.pose.position;
        Quaternion hitPoseRotation = mainRayHit.pose.rotation;
        Instantiate(prefab, hitPosePosition, hitPoseRotation);
    }
}
```

Компонент `ARPlaneManager` дає можливість відслідковувати створення, оновлення та видалення поверхонь на ігровій сцені, що дозволяє попередньо налаштувати їх, або передавати в інші компоненти.

Це забезпечує подія `trackablesChanged` з батьківського класу `ARTrackableManager`, на яку можуть підписатись класи з методом, що містить параметр типу `ARTrackablesChangedEventArgs<T>`: `T` означає будь-який клас, що наслідує `ARTrackable`, а у випадку з `ARPlaneManager` таким класом є `ARPlane`. Подібний параметр містить наступні важливі поля, всі з яких мають тип `ReadOnlyList<T>`:

- `added` – список доданих об'єктів на сцену;
- `updated` – список оновлених об'єктів на сцені через зміну положення, розмірів, кількості вершин тощо;
- `removed` (тип `ReadOnlyList<KeyValuePair<TrackableId, T>>`) – список пар ідентифікатор-об'єкт, які було видалено зі сцени.

Для прикладу буде наведено метод `CheckPlanes()`, що виконує попереднє налаштування візуального відображення нових поверхонь через зміну кольору контуру в залежності від їх орієнтації (вертикальна, горизонтальна з поворотом вгору або вниз). Наведене в прикладі налаштування є доречним, якщо площина має лише контур та не має 3D моделі площини, адже в іншому випадку необхідно змінювати колір моделі через задання різних матеріалів. Лістинг коду наведено нижче.

```
public Color verticalColor = new Color(0.1f, 0.57f, 0.83f);
public Color horizontalUpColor = new Color(0.55f, 1f, 0.13f);
public Color horizontalDownColor = new Color(1f, 0.21f, 0.93f);

public void CheckPlanes(ARTrackablesChangedEventArgs<ARPlane> eventArgs)
{
    foreach (ARPlane plane in eventArgs.added)
    {
        LineRenderer lineRenderer =
            plane.gameObject.GetComponent<LineRenderer>();

        switch (plane.alignment)
        {
            case PlaneAlignment.Vertical:
                SetColor(verticalColor);
                break;
            case PlaneAlignment.HorizontalUp:
                SetColor(horizontalUpColor);
                break;
            case PlaneAlignment.HorizontalDown:
                SetColor(horizontalDownColor);
                break;
            default:
                break;
        }
    }

    void SetColor(Color color)
```

```

    {
        lineRenderer.startColor = color;
        lineRenderer.endColor = color;
    }
}

```

Підписатись на подію можна двома способами:

- через метод AddListener (в такому разі програмний код буде виглядати як planeManager.trackablesChanged.AddListener(CheckPlanes));
- вручну через інспектор шляхом вибору об'єкта, компонента та методу у відповідному полі компоненту ARPlaneManager (рис. 1).

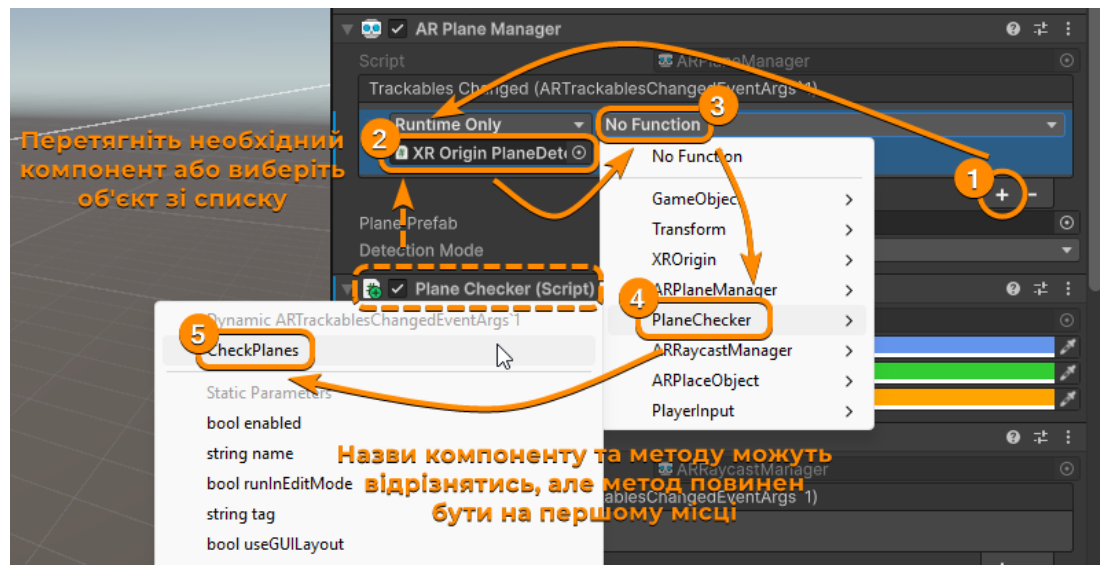


Рисунок 1 – Алгоритм підписання на подію через інспектор

Результат роботи вищевказаного прикладу демонструє рисунок 2. На ньому можна побачити сині, зелені та оранжеві багатокутники, що відповідають за вертикальні, горизонтальні з поворотом вгору та вниз поверхні.

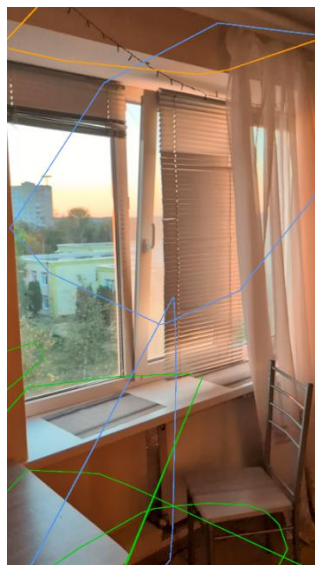


Рисунок 2 – Результат попереднього візуального налаштування поверхонь під час їх створення

Розпізнавання зображень – розділ доповненої реальності, що дозволяє створювати та змінювати 3D об'єкти в залежності від знайденого конкретного зображення. Таким методом створюють «живі» фотографії, відображають інтерактивні об'ємні об'єкти із зображень, реалізують колекціонування карток тощо.

В пакунку ARFoundation ігрового рушія Unity є компонент ARTrackedImageManager. Він дозволяє розпізнавати зображення з будь-яких поверхонь (найкраще – з рівних матових поверхонь), на основі яких виконуються необхідні дії. При розпізнаванні на сцені створюється об'єкт з компонентом ARTrackedImage, який оновлюється в залежності від положення зображення в просторі.

Умовою для коректної роботи розпізнавання зображень є якість та кількість унікальних патернів. Зображення з примітивними фігурами, приклад яких наведений на рисунку 3, має малу кількість ключових точок, тому відстежити їх буде неможливо. Також ARCore виконує оцінку кожного зображення на валідність для відстеження системою, з максимальним значенням в 100 балів. Якщо оцінка нижча за 75 балів, зображення може некоректно відстежуватись.



Рисунок 3 – Зображення, що має малу кількість ключових точок

Для роботи розпізнавання зображень потрібно виконати наступні дії:

1) Створити в проєкті батьківську папку Resources та дочірню папку Tracking. В останню необхідно додати зображення формату PNG. Для прикладу буде використано зображення з рисунку 4, оригінальний файл якого надається викладачем.

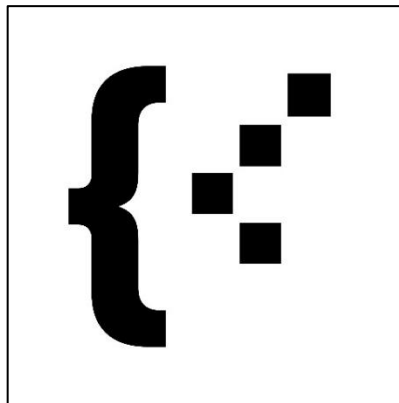


Рисунок 4 – Зображення для тестування відстеження

2) Перевірити валідність зображень для відстеження за допомогою додавання файлу `ARCoreImageEvaluator.cs` до проекту та його запуску через **Tools** → **ARCore** → **Evaluate Images** (рис. 5). Файл надається викладачем, а результат виводиться в консоль ігрового рушія. Якщо отримана оцінка є нижчою за рекомендовану або виводиться помилка, зображення необхідно доопрацювати або замінити на більш детальне.

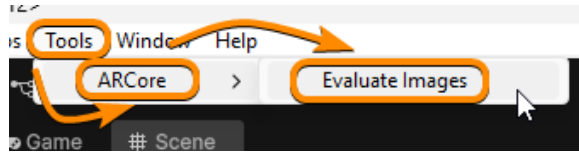


Рисунок 5 – Алгоритм перевірки валідності зображень для відстеження

3) Створити в папці з минулого кроку файл **Reference Image Library**, що знаходиться в контекстному меню за шляхом **Create** → **XR** → **Reference Image Library** (рис. 6).

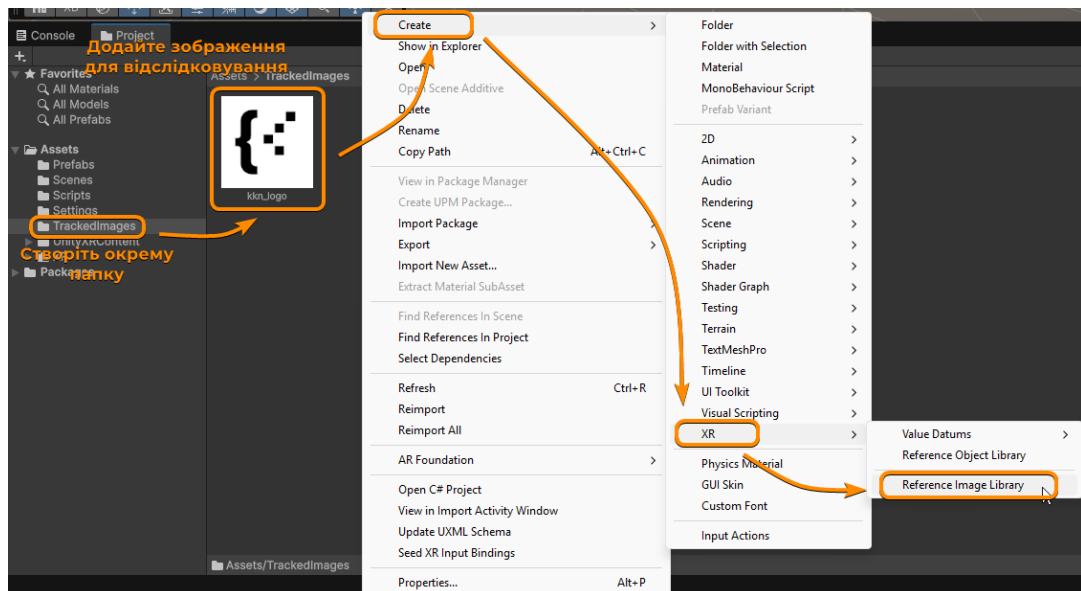


Рисунок 6 – Алгоритм додавання файлу **Reference Image Library**

4) Додати до **Reference Image Library** зображення для відстеження через кнопку **Add Image**, вказати файл зображення, завантажене в кроці 1, та вказати фізичні розміри зображення в метрах (рис. 7).

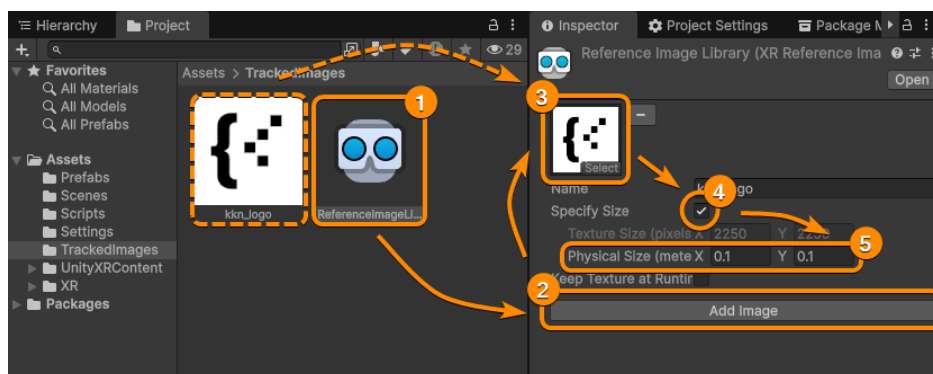


Рисунок 7 – Алгоритм додавання зображення для відстеження в файл **Reference Image Library**

- 5) Додати до сцени порожній об'єкт XR Origin, якщо він відсутній.
- 6) Додати до XR Origin компонент ARTrackedImageManager (рис. 8).

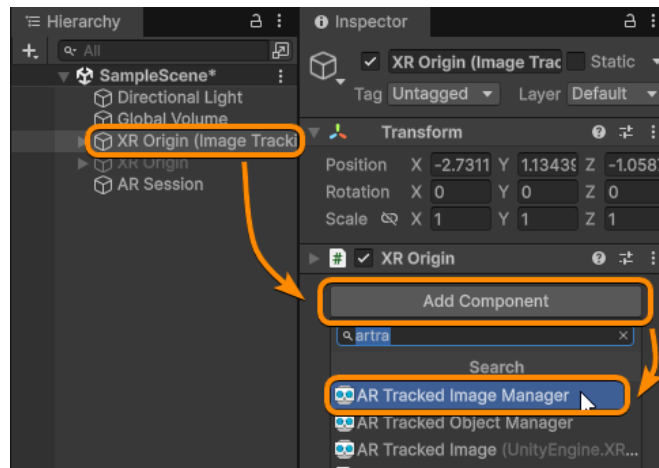


Рисунок 8 – Алгоритм додавання компонента ARTrackedImageManager до об'єкта XR Origin

- 7) Налаштувати компонент ARTrackedImageManager, вказавши створений в минулих кроках файл Reference Image Library, максимальну кількість рухомих зображень на сцені, а також префаб (рис. 9).

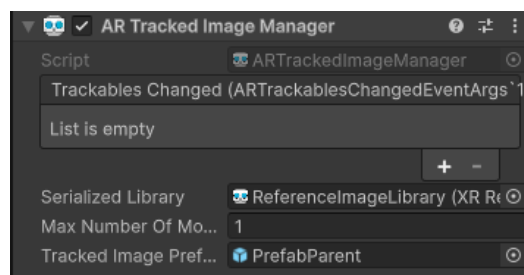


Рисунок 9 – Приклад налаштування компонента ARTrackedImageManager

При коректному налаштуванні компоненту ARTrackedImageManager буде отримано результат, представлений на рисунку 10. Тестування роботи розпізнавання зображень працює лише на реальному пристрої, тому кожна перевірка потребує створення збірки.

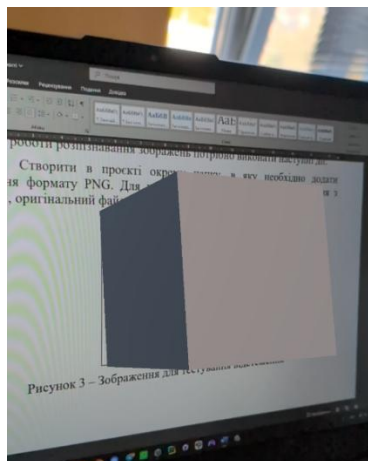


Рисунок 10 – Приклад роботи розпізнавання зображення з налаштованим компонентом ARTrackedImageManager

Префаб, що створюється компонентом ARTrackedImageManager, додається до об'єкта знайденого зображення з компонентом ARTrackedImage як дочірній, тому його положення відповідає положенню зображення в просторі. Цього достатньо для реалізації «живих» та об'ємних фото, колекціонування тощо, однак інші сценарії потребують інший підхід.

Компонент ARTrackedImageManager, як і ARPlaneManager, має подію trackablesChanged (T – ARTrackedImage), який дозволяє відстежувати зображення (див. с. 3). З компоненту ARTrackedImage можна отримати transform, на основі якого можна створювати незалежні об'єкти.

Прикладом може стати метод CheckImage(), що перевіряє всі оновлені зображення. Якщо зображення зустрічається вперше, він створює префаб з його положенням на сцені, однак об'єкт не додається до зображення та обертання залишається незмінним.

Рекомендовано оновлювати положення об'єкта по зображенню певний проміжок часу (в прикладі це пів секунди). Такі дії потрібні для визначення більш точного положення зображення в просторі компонентом ARTrackedImageManager.

Лістинг методу CheckImage() представлено нижче.

```
[SerializeField] private GameObject prefab;
private Dictionary<TrackableId, GameObject> objectsByImage = new();
private Dictionary<GameObject, float> timersByObject = new();

public void CheckImage(
    ARTrackablesChangedEventArgs<ARTrackedImage> eventArgs)
{
    foreach (var image in eventArgs.updated)
    {
        if (!objectsByImage.TryGetValue(
            image.trackableId,
            out var createdObject))
        {
            objectsByImage.Add(image.trackableId,
                Instantiate(
                    prefab,
                    image.transform.position,
                    Quaternion.identity)
                );

            timersByObject.Add(
                objectsByImage[image.trackableId],
                Time.time + 0.5f
            );

            continue;
        }

        if (timersByObject[createdObject] >= Time.time)
            createdObject.transform.position = image.transform.position;
    }
}
```

Перед тестуванням роботи подібного підходу спершу необхідно прибрати будь-який префаб з відповідного поля компоненту ARTrackedImageManager. В іншому випадку, створяться два об'єкти.

Результатом роботи CheckImage() представлений на рисунку 11. Приклад представляє лише об'єкт, що висить у повітрі, однак на його місці може бути об'єкт з будь-якою реалізацією, яка необхідна розробнику.

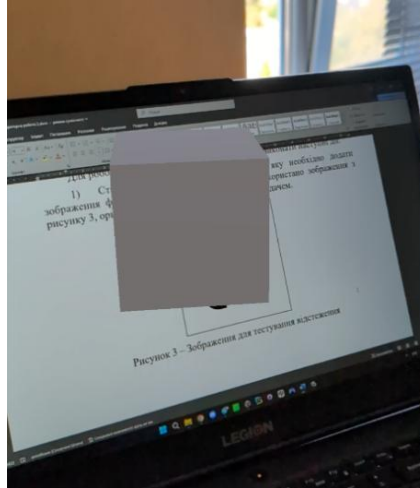


Рисунок 11 – Результат створення незалежного об'єкта після розпізнавання зображення методом CheckImage()

Перекривання (Occlusion) 3D моделей реальними об'єктами із середовища дозволяє краще погрузити користувача в додаток або симулювати вигляд об'єктів в просторі, в залежності від цілей додатку.

Основою перекривання є аналіз глибини середовища, в результаті якого отримується мапа глибини (depth map). Це може бути як двовимірний масив просторових точок, так і зображення з градієнтом. Мапа формується при аналізі елементів навколишнього середовища та того, наскільки вони перемістились за певний проміжок часу. Об'єкти, що знаходяться близько, мають відчутне зміщення положення, а дальні об'єкти зміщуються на декілька поділок.

Зазвичай, в простих додатках використовується останній варіант з градієнтом сірого, де світлі об'єкти розміщені близько до камери, а темні - далі. Приклад зображення з градієнтом сірого представлений на рисунку 12.

Об'єктам на ігровій сцені також задається певний градієнт в залежності від їх віддаленості від віртуальної камери. Під час аналізу мапи глибини та об'єкта відносно неї задається маска, яка перекриває об'єкт там, де середовище стоїть ближче, тобто пікселі або точки є світлішими за об'єкт.



Рисунок 12 – Приклад мапи глибини з градієнтом сірого

Якість мапи глибини залежить від апаратного забезпечення пристрою, на якому буде запускатись додаток, та способу програмного аналізу. Звичайні пристрої, які мають лише камеру та електронні пристрої виміру, дають посередню якість мапи та можуть не точно враховувати об'єкти, які з'явилися перед камерою, тому об'єкти можуть «обрізатись» не зовсім якісно. Пристрої, що мають апаратні датчики глибини (LiDAR або ToF), дають кращі результати.

В пакунку ARFoundation є готовий компонент AROcclusionManager, який додається до дочірньої камери об'єкта XR Origin та застосовує маску до об'єктів відносно мапи глибини. Алгоритм додавання компонента представлено на рисунку 13.

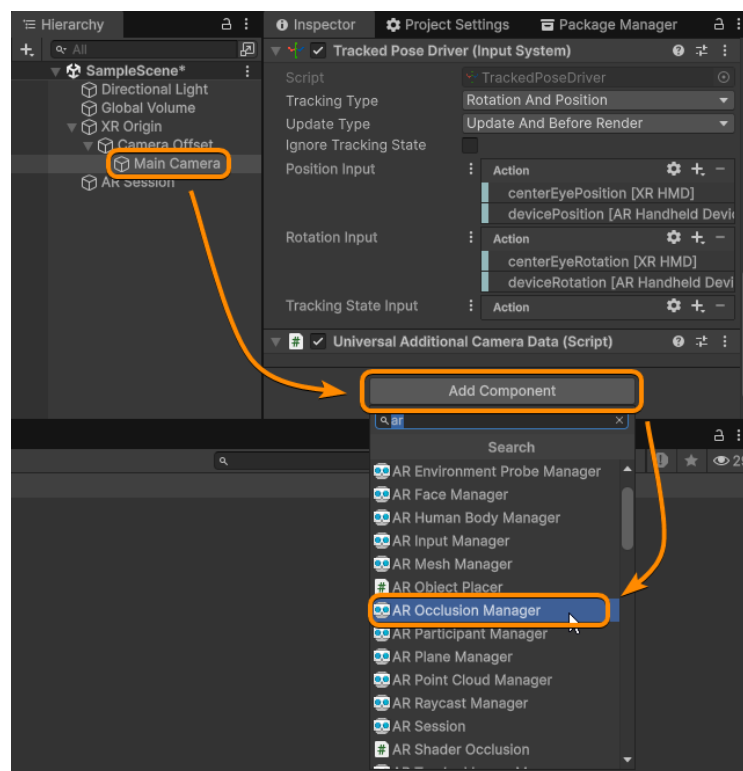


Рисунок 13 – Алгоритм додавання компонента AROcclusionManager до камери об'єкта XR Origin

AROcclusionManager має налаштування для швидкодії та якості зчитування мапи глибини та його застосування на об'єктах, а також відстеження людей за допомогою сегментації. Чим потужніший пристрій, тим вищі параметри можна виставити. Приклад налаштування компонента представлений на рисунку 14.

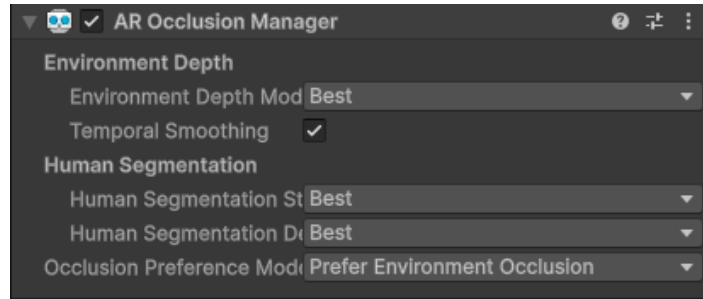


Рисунок 14 – Приклад налаштування компонента AROcclusionManager

Тестування роботи перекривання доступне на комп'ютері, однак через створення симуляції камери 3D сценою мапа глибини формується з великою точністю, що не відповідає роботі функції на реальному пристрої.

Приклад роботи додатку з розміщення об'єктів, в який було додано функцію перекривання, представлено на рисунку 15. Він наглядно демонструє перекривання 3D об'єкта реальними предметами лише у випадку їх візуального розміщення перед ним.



а

б

Рисунок 15 – Вигляд 3D об'єкта в AR додатку з реальними об'єктами перед ним (а) та без перешкод зверху (б)

Зміст роботи

Завдання 1. Створити додаток з розміщення меблів та предметів у просторі:

- додаток повинен розпізнавати поверхні в просторі, визначати їх орієнтацію (вертикальна, горизонтальна з поворотом вгору або вниз) та розміри;
- реалізувати візуальну відмінність поверхонь кольором в залежності від їх параметрів;
- додати можливість обирати поверхню та ставити на неї об'єкти в залежності від її параметрів (орієнтація, розміри тощо), при цьому зупиняючи розпізнавання поверхонь на момент вищеописаної дії;
- в додатку реалізувати 3 категорії об'єктів (рівна поверхня, стіна, стеля), кожна з яких міститиме 5+ об'єктів, об'єкти повинні мати обмеження по розмірам поверхні (вказані вручну або аналізуючи розміри їх 3D моделей);
- реалізувати інтерфейс, через який користувач може обирати об'єкт для розміщення після вибору поверхні;
- дозволяється використовувати готові та власні 3D моделі, а також виконувати завдання за іншою тематикою за умови виконання всіх вищевказаних критеріїв;

Завдання 2. Реалізувати додаток з розпізнавання зображень:

- додати можливість розпізнавати 3+ зображення;
- кожне зображення має свій об'єкт/сцену, який додається при знаходженні зображення в просторі;
- мінімум одне зображення повинно мати інтерактивну сцену (ручне обертання 3D моделі, керування персонажем/технікою в просторі тощо);
- дозволяється використовувати готові та власні 3D моделі;

Завдання 3. До кожного додатку реалізувати перекривання створених 3D об'єктів реальними об'єктами з навколишнього середовища:

- завдання вважається виконаним у разі успішного перекривання 3D об'єкта меблями, предметами, стіною тощо.

Додатково. Виконати на вибір одне із запропонованих завдань:

- 1) Для сцени із завдання 1 додати можливість попереднього показу об'єкта в просторі перед його розміщенням, та обертати його, Також користувач повинен мати можливість редагувати положення та обертання існуючих об'єктів, враховуючи розміри поверхні та інших 3D моделей.

2) Реалізувати ігрову сцену з керованим персонажем, який може рухатись лише по розпізнаних горизонтальних поверхнях, а також стрибати або падати на поверхні, що розміщені вище або нижче відповідно. Поверхні повинні бути близько до персонажа, щоб він міг виконувати вищеописані дії, а також персонаж не повинен заходити за межі поверхонь.

3) Реалізувати AR портал для завдання 1 або для завдання 2. Вигляд portalу довільний (картина або діра в стіні, двері тощо). Навчальний матеріал по створенню моделі portalу в ігровому рушії знаходиться за посиланням: <https://youtu.be/Y1lzXWD8NN8>.

Контрольні питання:

1. За що відповідає компонент ARPlane?
2. Назвіть 3 основні значення поля alignment класу ARPlane.
3. Яке поле зберігає вершини поверхні?
4. За що відповідає trackable в класі ARRaycastHit?
5. Коли викликається подія trackablesChanged?
6. Які дані відправляє trackablesChanged підписаним методам?
7. Що таке Reference Image Library?
8. Який параметр компонента ARTrackedImageManager відповідає за кількість відстежуваних зображень в просторі?
9. Що таке AROcclusionManager?
10. Як працює перекривання 3D моделей реальними об'єктами із середовища?