

МИНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

кафедра програмних засобів

Інструкція з завантаження, встановлення та роботи з Unity з
використанням шолому віртуальної реальності Oculus Rift

2021

1 ЗАВАНТАЖЕННЯ ТА ВСТАНОВЛЕННЯ UNITY

1.1 Завантаження та встановлення Unity Hub

Для роботи з Unity перш за все необхідно завантажити та встановити Unity Hub.

Unity Hub – це нова програма, розроблена для спрощення робочого процесу. Це справжній центр управління Unity-проектами, в якому можна легко шукати, завантажувати та організовувати компоненти редактора Unity. Крім того, з ним просто знаходити нові можливості для своїх проектів, наприклад, за допомогою функції шаблонів.

Завантажити Unity Hub можна з офіційного сайту Unity, за посиланням: <https://unity3d.com/get-unity/download>. Відкривши сторінку завантаження Unity необхідно натиснути кнопку «Download Unity Hub» (рис. 1.1).

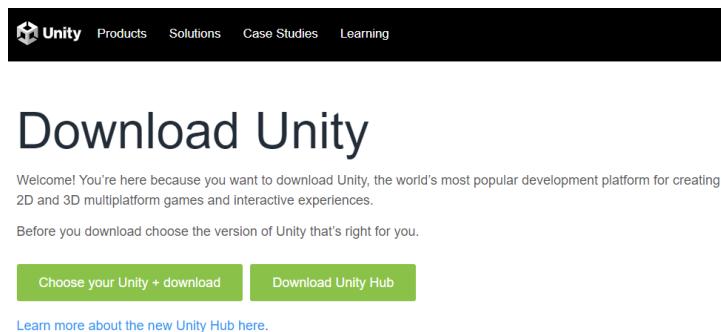


Рисунок 1.1 – Сторінка завантаження Unity

Має розпочатися завантаження файлу інсталяції «UnityHubSetup.exe» (рис. 1.2). Після успішного завершення процесу завантаження необхідно відкрити файл та розпочати встановлення програми.

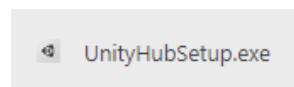


Рисунок 1.2 – Завантажений файл інсталяції «UnityHubSetup»

У вікні ліцензійної угоди необхідно ознайомитися з угодою користувача та натиснути на кнопку «Приймаю» (рис. 1.3).

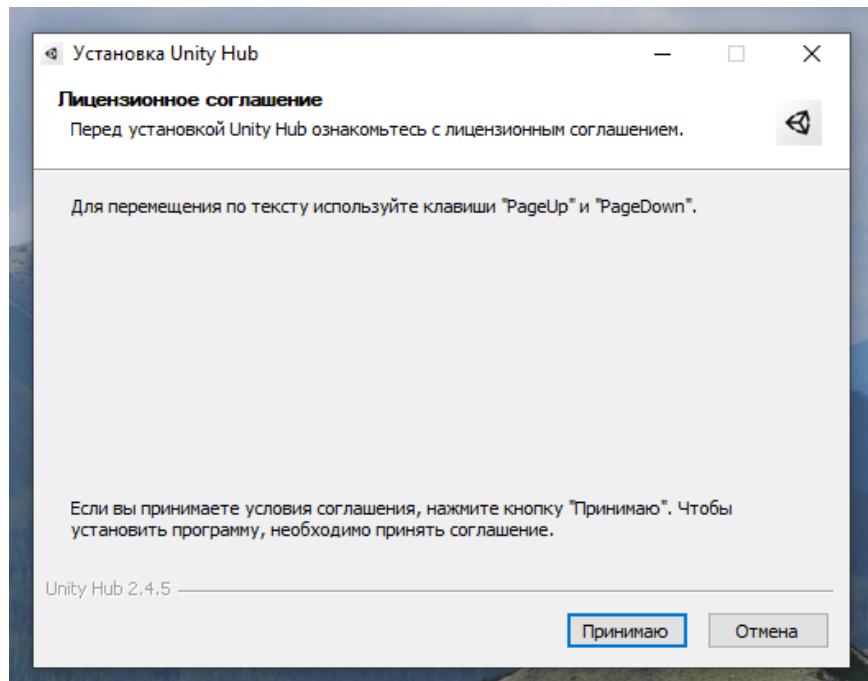


Рисунок 1.3 – Погодження з ліцензійної угодою користувача

Далі необхідно вибрати папку на жорстокому диску, в яку буде встановлено Unity Hub та натиснути на кнопку «Встановити» (рис. 1.4).

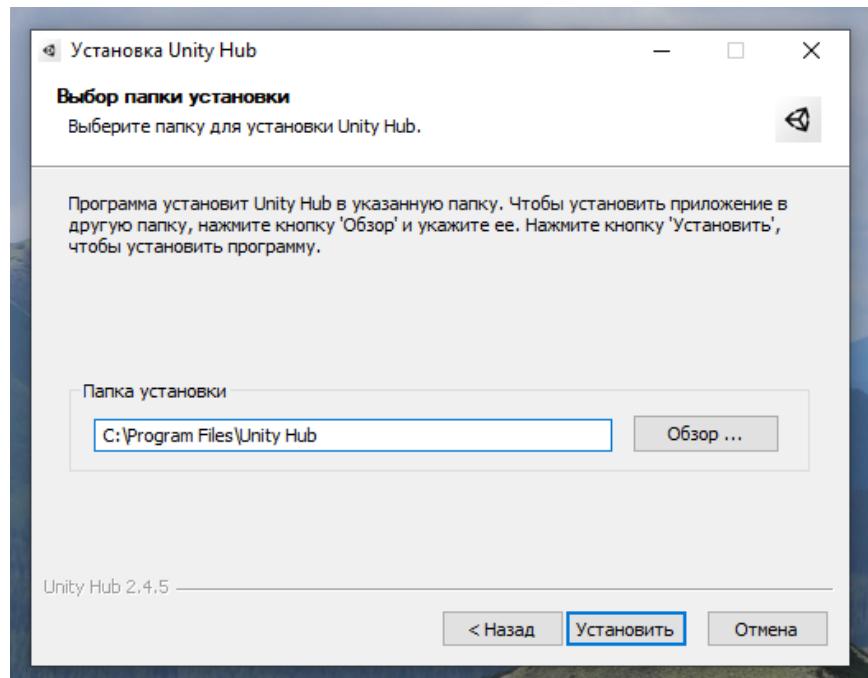


Рисунок 1.4 – Вибір папки для встановлення Unity Hub

Розпочнеться процес встановлення Unity Hub. Встановлення займає декілька хвилин часу (рис. 1.5).

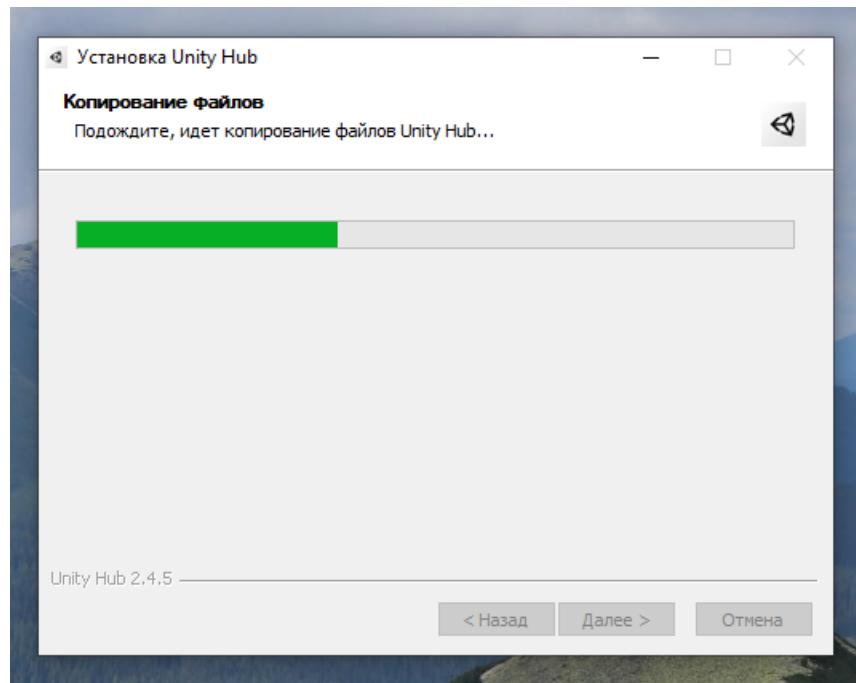


Рисунок 1.5 – Процес встановлення Unity Hub

Після завершення встановлення програми буде відкрито вікно з повідомленням про успішне завершення роботи майстра установки Unity Hub. Необхідно натиснути на кнопку «Готово» (рис. 1.6).

Unity Hub успішно встановлено. Має відкритися головне вікно програми.

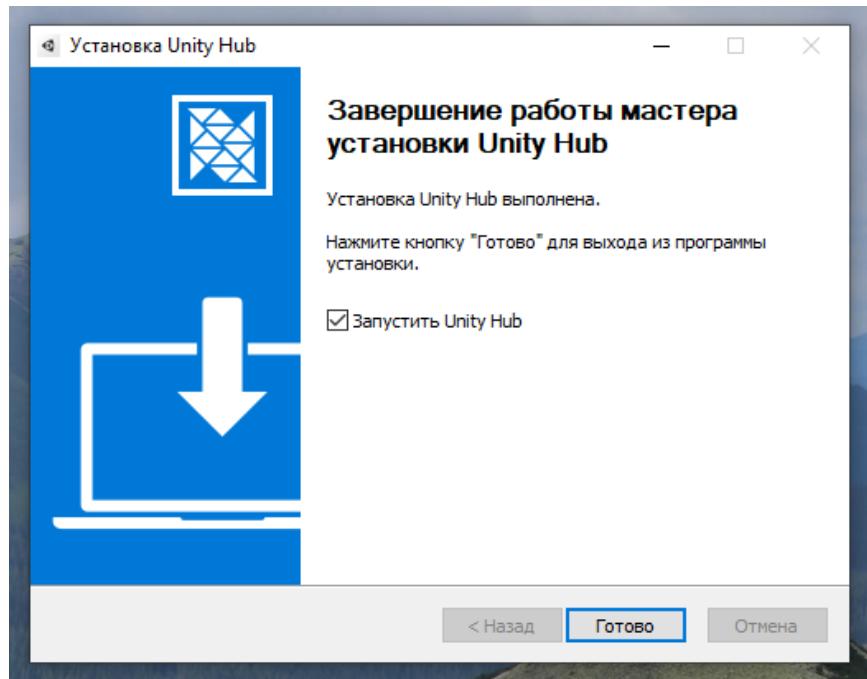


Рисунок 1.6 – Успішне завершення встановлення Unity Hub

1.2 Налаштування ліцензій користувача в Unity Hub

Після успішного встановлення Unity Hub, його буде відкрито (рис. 1.7). Для роботи з Unity необхідно створити акаунт, або увійти у вже існуючий. Для цього необхідно натиснути на іконку аватара в правій верхній частині вікна.

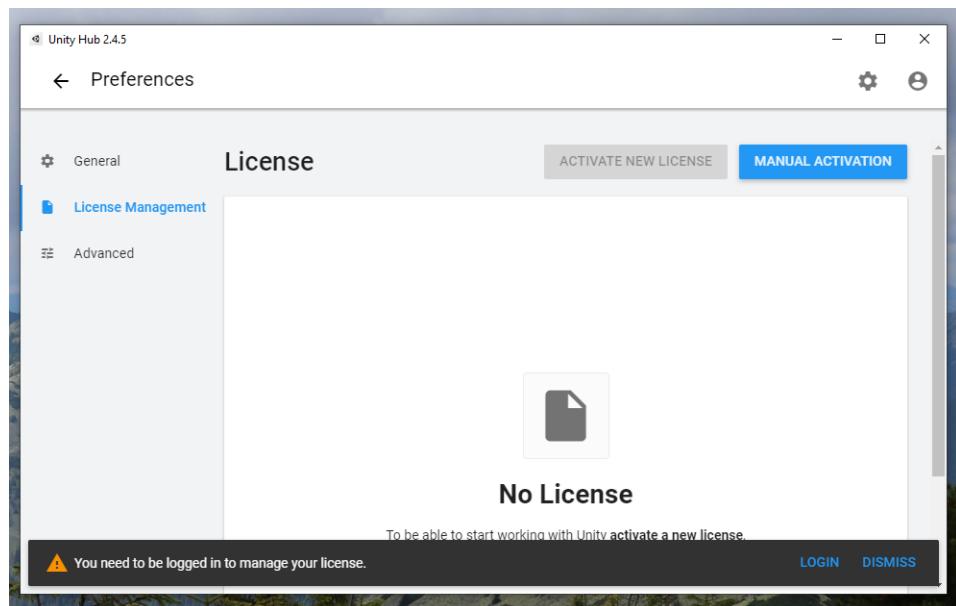


Рисунок 1.7 – Відкритий Unity Hub після встановлення

У відкритій панелі необхідно натиснути на «Sign in» та увійти у вже існуючий акаунт, або створити новий (рис. 1.8).

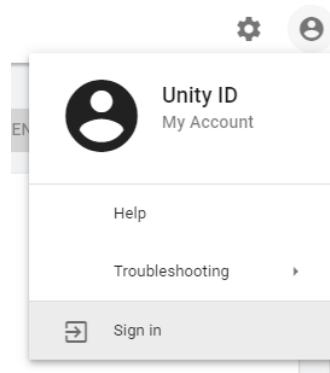


Рисунок 1.8 – Панель користувача

Після входу в акаунт в панелі користувача буде відображенено інформацію про нього та його аватар.

Для роботи з Unity необхідно мати хоча б одну ліцензію користувача на використання програми. Є як платні так і безкоштовні постійні ліцензії. Для перевірки ліцензій необхідно відкрити сторінку «License Management» в меню «Preferences», або в панелі користувача натиснути на кнопку «Manage license» (рис. 1.9).

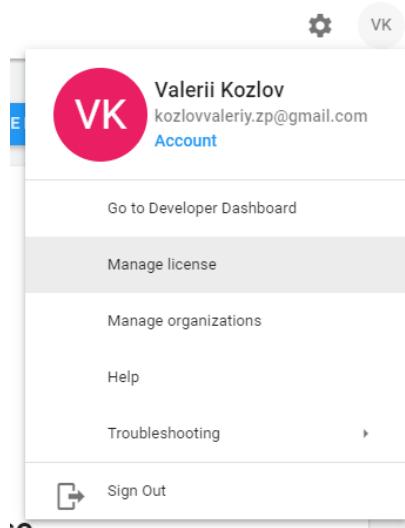


Рисунок 1.9 – Панель користувача після входу в акаунт

Для отримання ліцензій необхідно натиснути на кнопку «Activate new license» на сторінці «License Management» в меню «Preferences» (рис. 1.10).

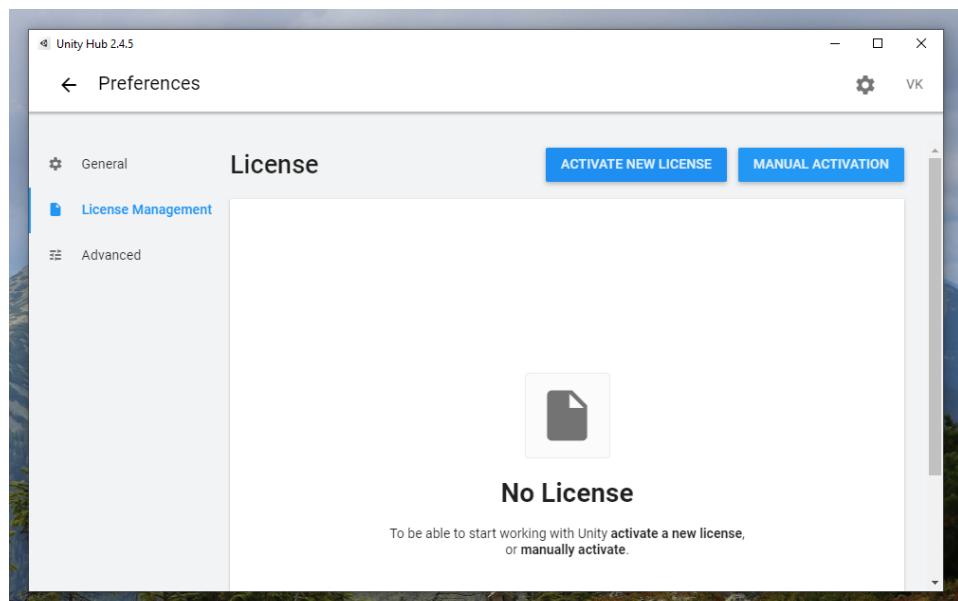


Рисунок 1.10 – Сторінка «License Management» в меню «Preferences»

Для отримання безкоштовної персональної ліцензії для навчання у вікні «New License Activation» необхідно вибрати пункт «Unity Personal» та підпункт «I don't use Unity in a professional capacity» та натиснути кнопку «Done» (рис. 1.11).

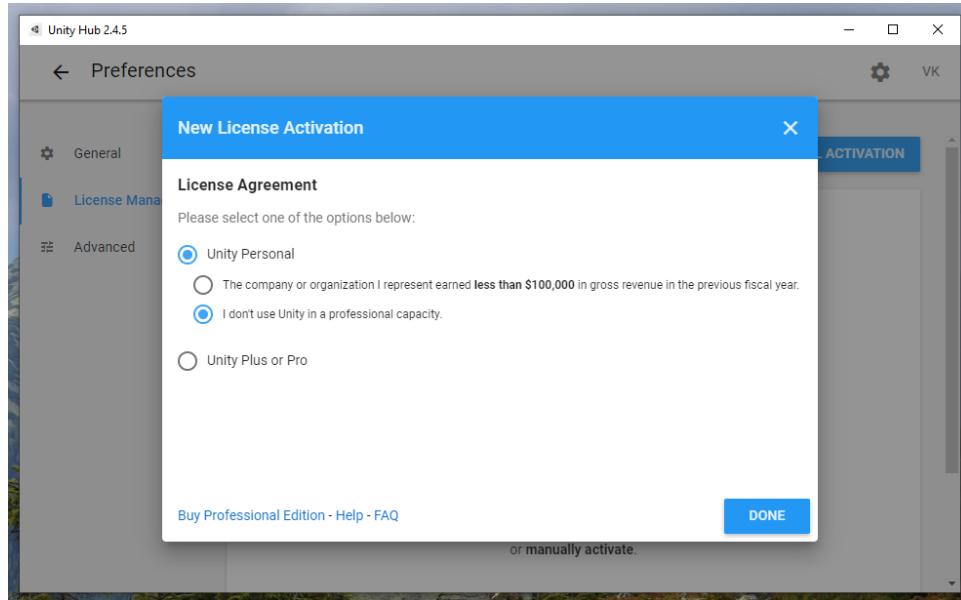


Рисунок 1.11 – Вікно «New License Activation»

Після активації ліцензії її буде відображенено на сторінці «License Management» в меню «Preferences» (рис. 1.12).

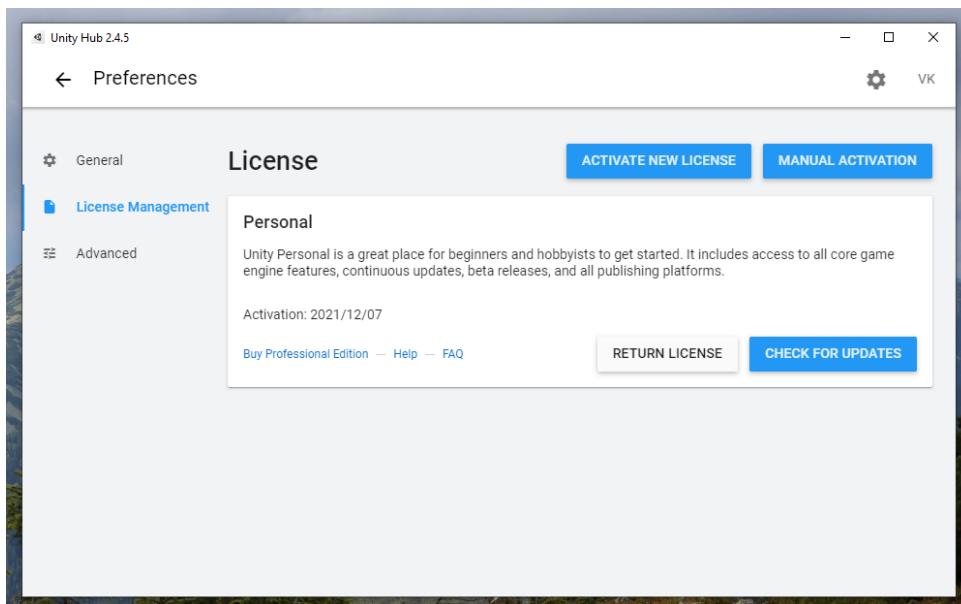


Рисунок 1.12 – Сторінка «License Management» після додавання ліцензії

1.3 Завантаження необхідної версії Unity та створення первого проекту

Для встановлення будь-якої версії Unity необхідно перейти на вкладку «Installs». У відкритій вкладці необхідно натиснути кнопку «Locate» для вказання, що необхідна версія вже встановлена на комп’ютері та обрати папку з її розташуванням, або кнопку «Add» для завантаження та додавання версії з ресурсів Unity. Натиснемо кнопку «Add» (рис. 1.13)

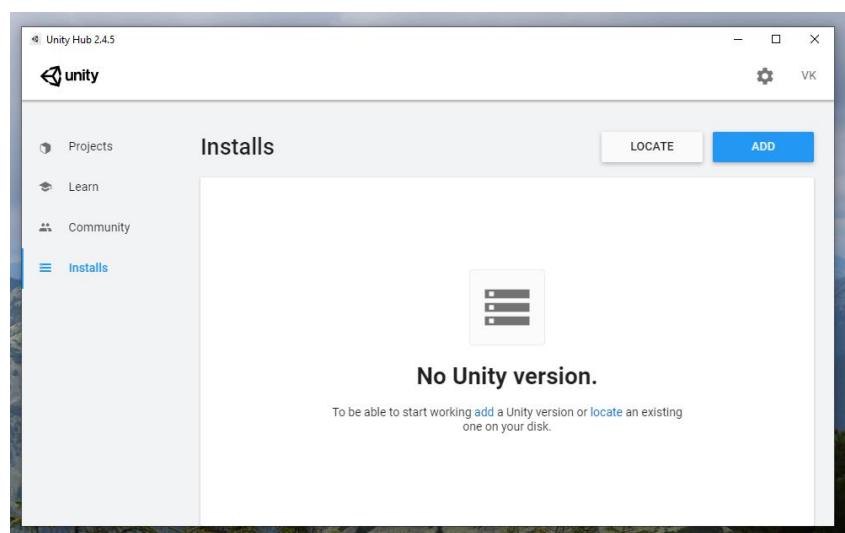


Рисунок 1.13 – Вміст вкладки «Installs»

У вікні «Add Unity Version» можна вибрати необхідну версію Unity та встановити її натиснувши кнопку «Next» (рис. 1.14).

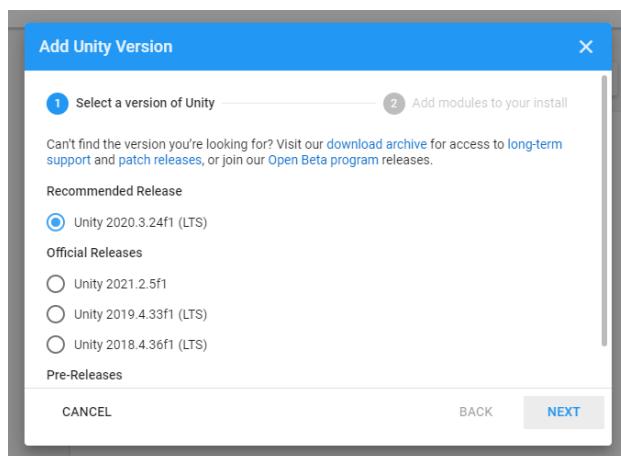


Рисунок 1.14 – Вікно «Add Unity Version»

Для завантаження іншої старшої версії Unity необхідно перейти на сторінку архівних версій Unity на офіційному сайті за посиланням <https://unity3d.com/get-unity/download/archive>. З наведеного списку вибрati необхідну версію середовища та натиснути кнопку «Unity Hub» зі значком завантаження (рис. 1.15). У відкритому вікні браузера натиснути кнопку «Відкрити застосунок «Unity Hub» (рис. 1.16)



Рисунок 1.15 – Версія Unity 2019.3.0 з архіву версій

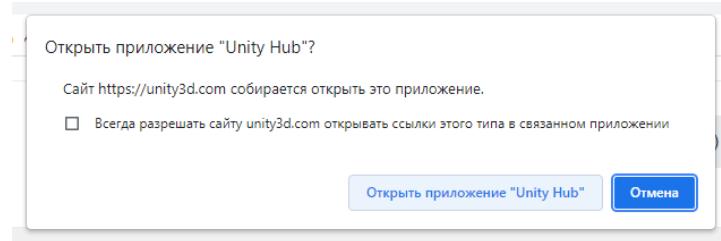


Рисунок 1.16 – Запит браузера на відкриття Unity Hub

У відкритому вікні «Add Unity Version» в Unity Hub вибрati необхідні пакети для завантаження (рекомендація встановити Visual Studio, якщо вона в вас не встановлена) та натиснути кнопку «Install» (рис. 1.17).

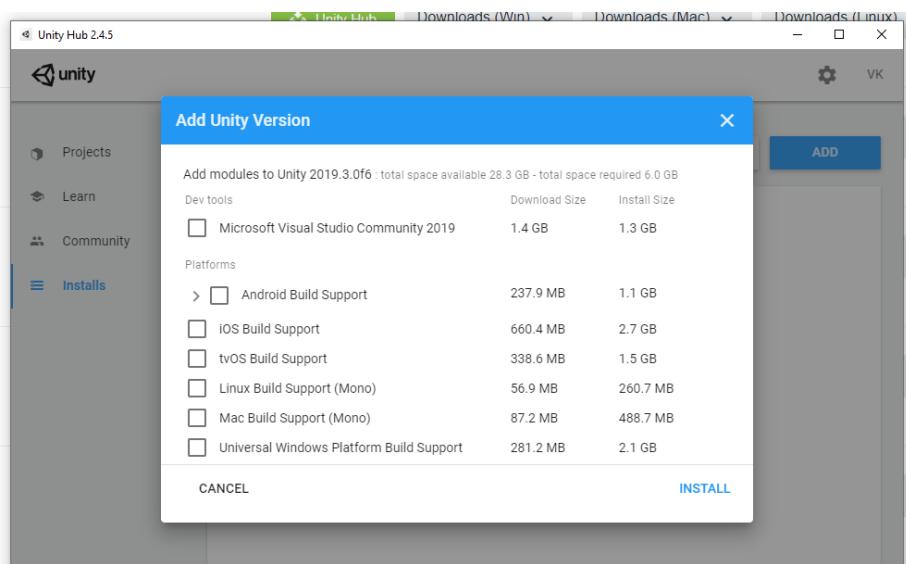


Рисунок 1.17 – Додавання нової версії Unity в Unity Hub

У вкладці «Installs» буде відображено процес завантаження вибраної версії Unity (рис. 1.18).

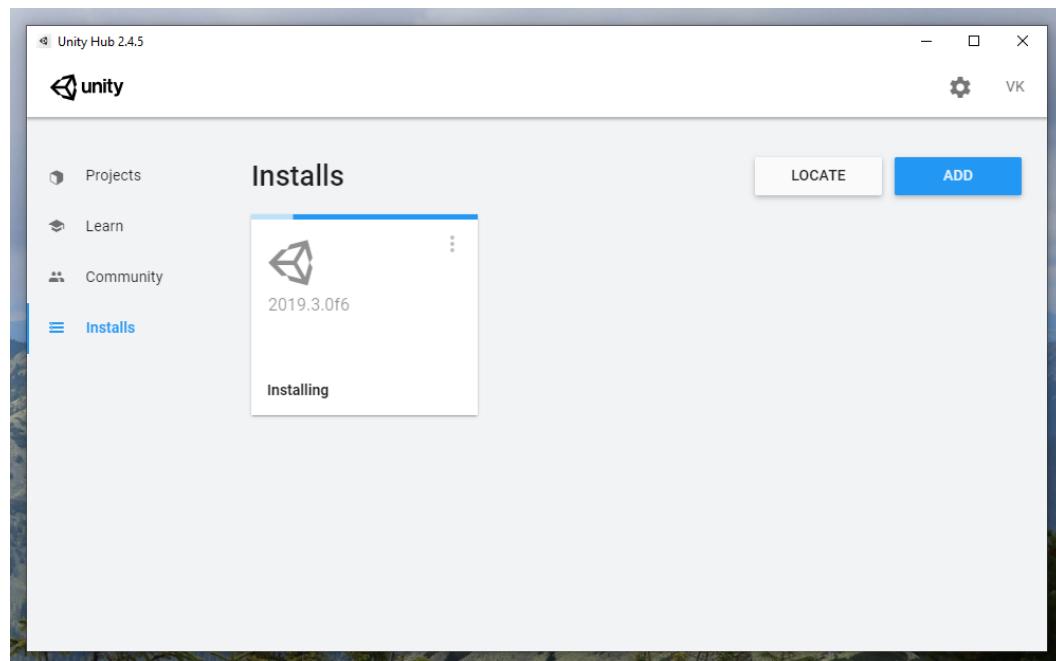


Рисунок 1.18 – Вкладка «Installs» під час завантаження версії Unity

Після завантаження та встановлення необхідної версії Unity вона стане активною для використання та створення проектів. Елемент версії у вкладці «Installs» стане активним (рис. 1.19).

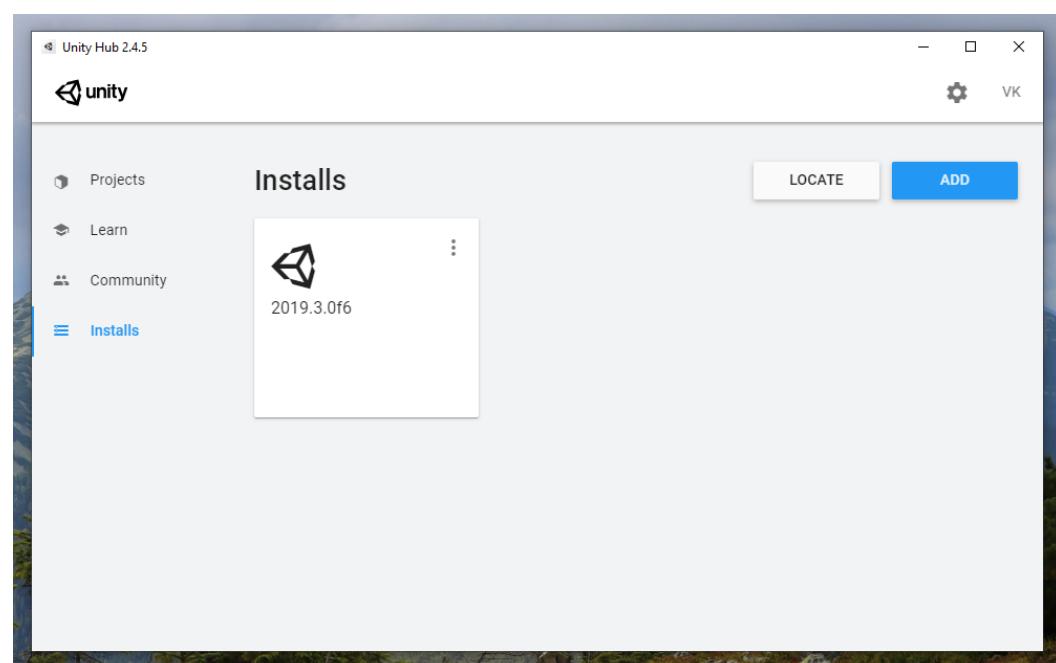


Рисунок 1.19 – Вкладка «Installs» після завантаження версії Unity

Для створення нового проекту необхідно перейти у вкладку «Projects» та натиснути кнопку «Add» для додавання існуючого проекту до Unity Hub або «New» для створення нового проекту (рис. 1.20).

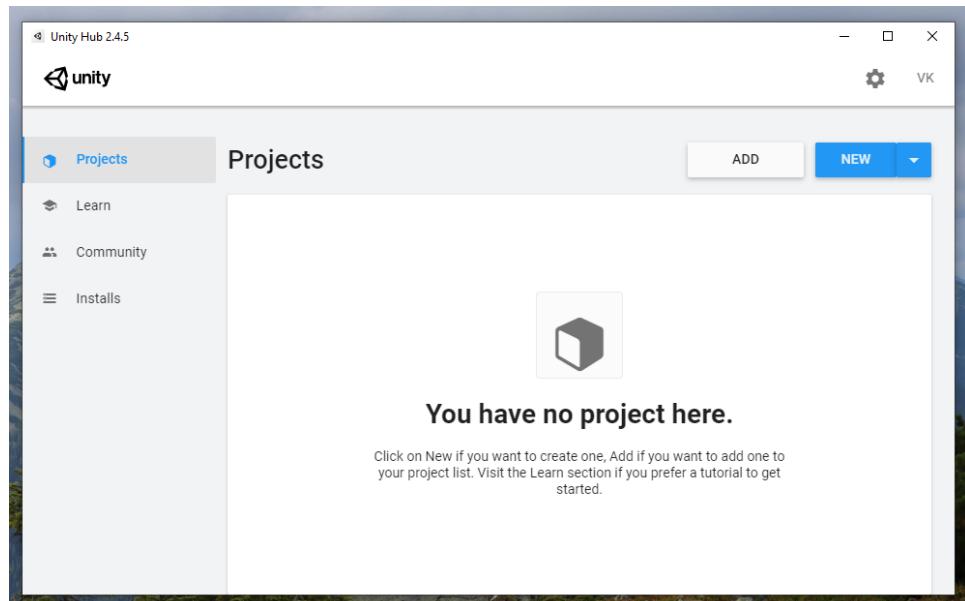


Рисунок 1.20 – Вкладка «Projects» в Unity Hub

У вікні створення нового проекту необхідно вибирати бажаний шаблон, ввести назву проекту та його місце розташування в пам'яті (рис. 1.21). Створимо проект з шаблону «Universal Project Template».

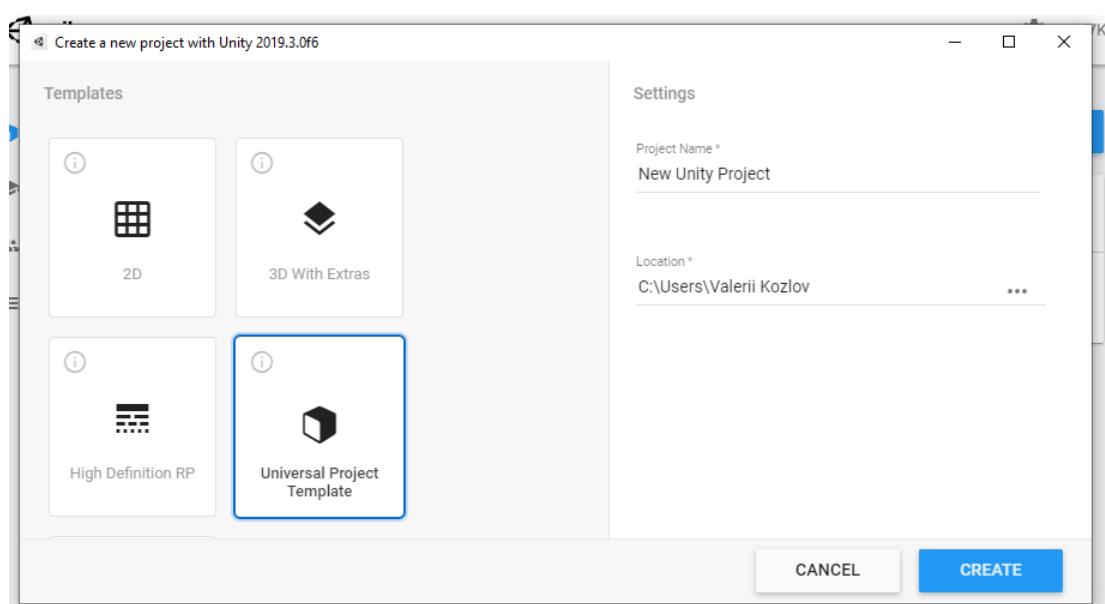


Рисунок 1.21 – Вікно створення нового проекту

В результаті створення проєкту відкрито головний інтерфейс середовища Unity з головною сценою вибраного шаблону (рис. 1.22).

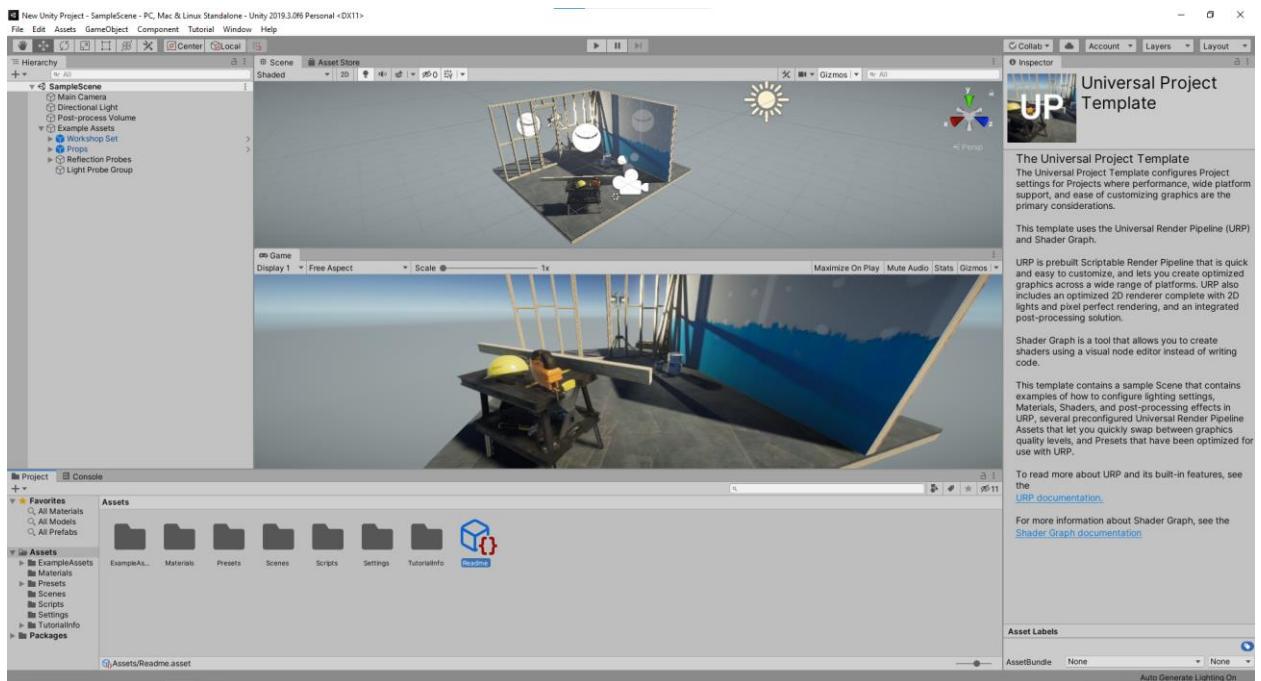


Рисунок 1.22 – Середовище розробки застосунків Unity

2 РОБОТА З VR ТА OCULUS RIFT В ПРОЄКТІ НА UNITY

Для роботи з шоломом віртуальної реальності Oculus Rift в середовищі розробки Unity небхідно підключити пристрій до комп'ютера та запустити програмне забезпечення Oculus (рис. 2.1).

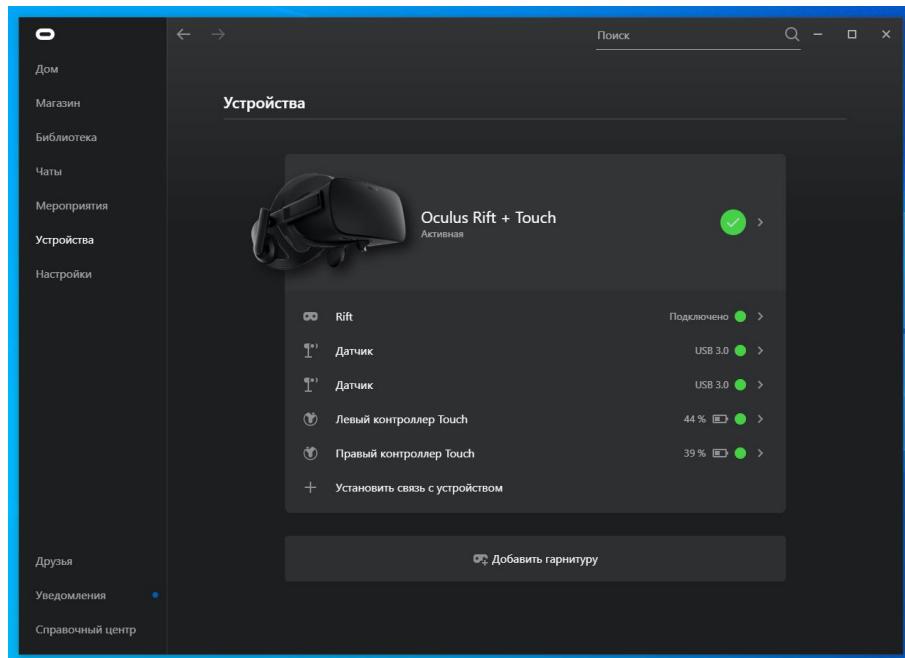


Рисунок 2.1 – Підключений Oculus Rift в програмі Oculus

Для можливості запуску сторонніх застосунків та розробленого програмного забезпечення необхідно дозволити запуск з невідомих джерел на Oculus Rift. Для цього в програмі Oculus необхідно перейти загальні налаштування («Settings» - «General») та активувати пункт «Unknown Sources» натиснувши на повзунок (рис. 2.2).

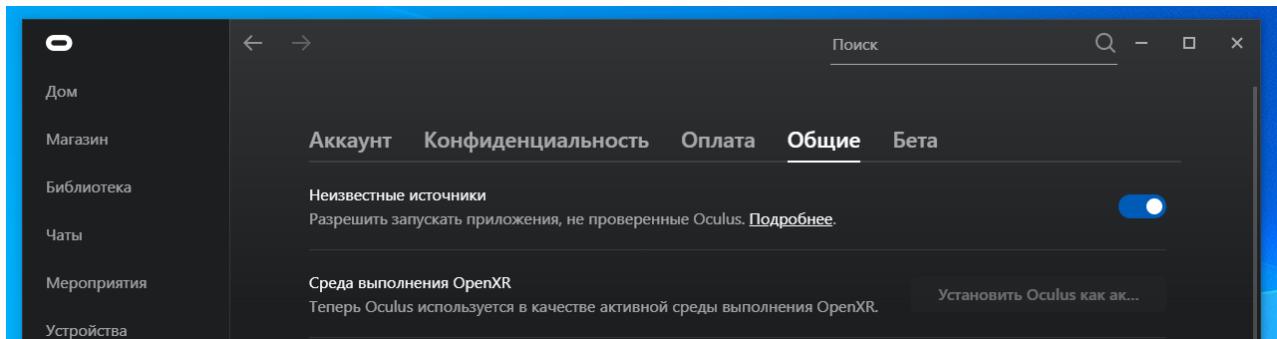


Рисунок 2.2 – Налаштування параметрів запуску з невідомих джерел

Розпочнемо роботу з Oculus Rift в середовищі розробки Unity зі створення нового проекту. За допомогою Unity Hub створимо новий проект натиснувши на кнопку «New» попередньо вибравши встановлено версію Unity 2019.3 (рис. 2.3).

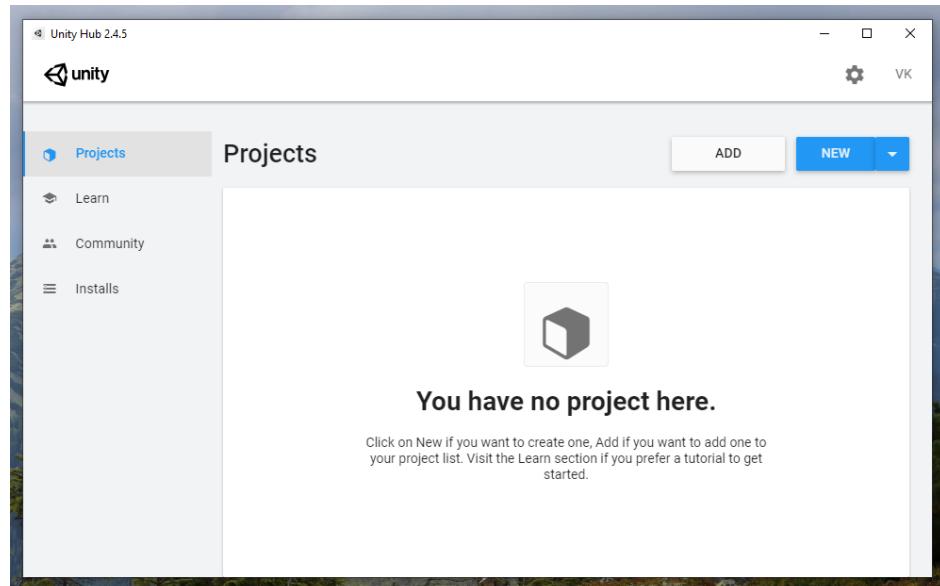


Рисунок 2.3 – Вкладка «Projects» в Unity Hub

В якості шаблону вибираємо 3D, встановлюємо необхідну назву для проекту та його місце розташування. Для створення проекту натискаємо на кнопку «Create» (рис. 2.4).

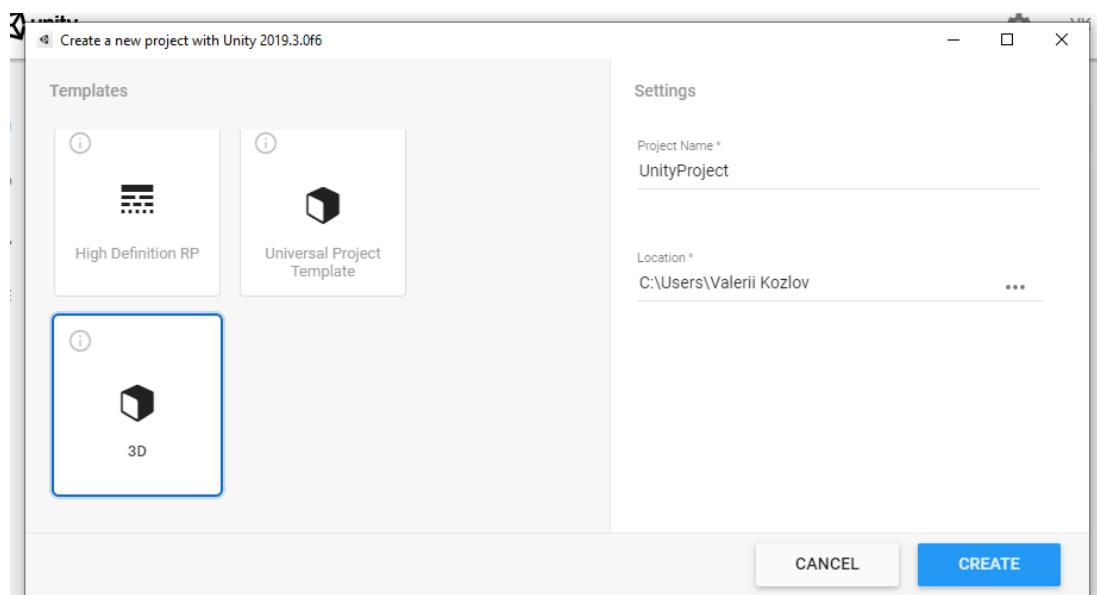


Рисунок 2.4 – Створення нового проекту з шаблоном 3D

Після імпорту необхідних пакетів (рис. 2.5) буде відкрито головний інтерфейс середовища Unity з головною сценою вибраного шаблону (рис. 2.6)

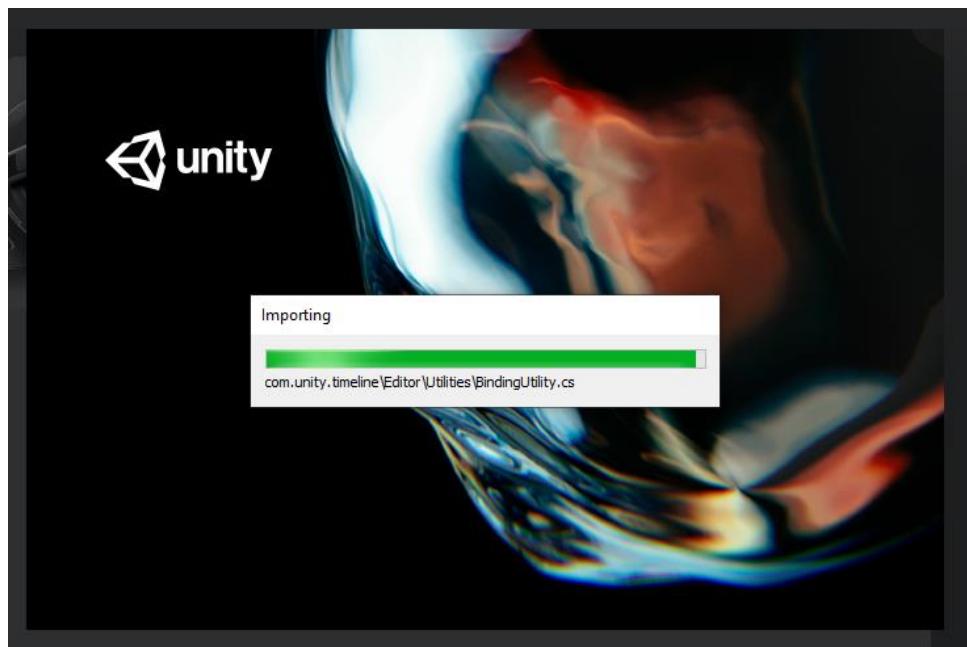


Рисунок 2.5 – Імпорт пакетів перед відкриттям середовища розробки

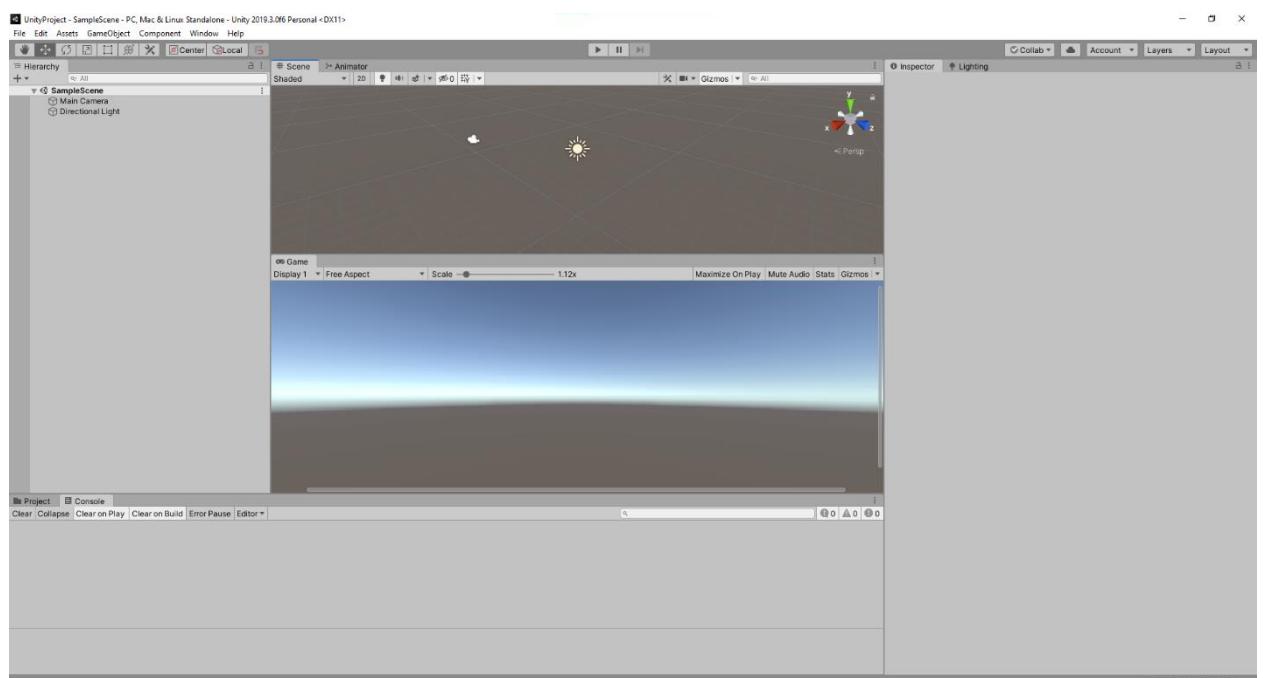


Рисунок 2.6 – Головний інтерфейс середовища Unity з шаблоном 3D

Налаштуємо створений проект на можливість роботи з шоломами віртуальної реальності. Для цього необхідно відкрити менеджер пакетів та встановити необхідні налаштування.

Відкриємо менеджер пакетів натиснувши на пункт «Package Manager» у вкладці «Window» (рис. 2.7).

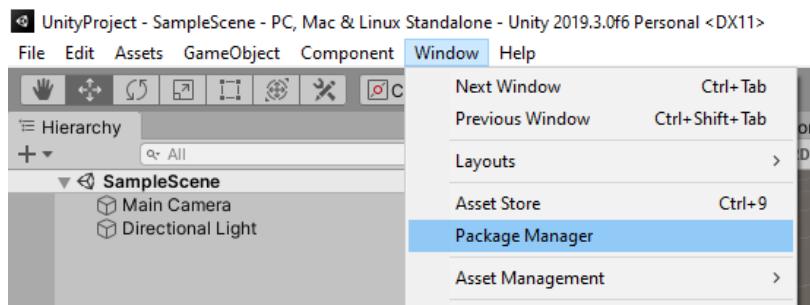


Рисунок 2.7 – Відкриття менеджера пакетів

Вибираємо відображення попереднього перегляду в налаштуваннях відображеніх пакетів (рис. 2.8).

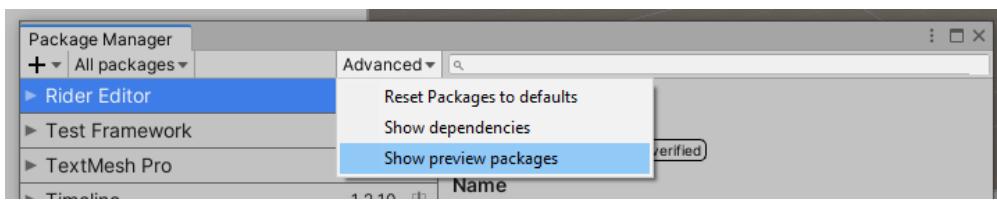


Рисунок 2.8 – Відображення попереднього перегляду пакетів

Для роботи з VR використовують пакет «XR Interaction Tool». XR Interaction Toolkit – це високорівнева система взаємодії на основі компонентів для створення Virtual Reality (віртуальної реальності) та Augmented Reality (доповненої реальності). Він забезпечує структуру, яка робить 3D-взаємодії та інтерфейс користувача доступними за допомогою використання подій введення Unity. Ядром цієї системи є набір базових компонентів Interactor та Interactable, а також менеджер взаємодії, який пов’язує ці два типи компонентів разом. Він також містить допоміжні компоненти, які можна використовувати для розширення функціональних можливостей для малювання візуальних елементів і підключення до власних подій взаємодії.

Знаходимо пакет «XR Interaction Tool» та натискаємо кнопку «Install» для завантаження в проект (рис. 2.9).

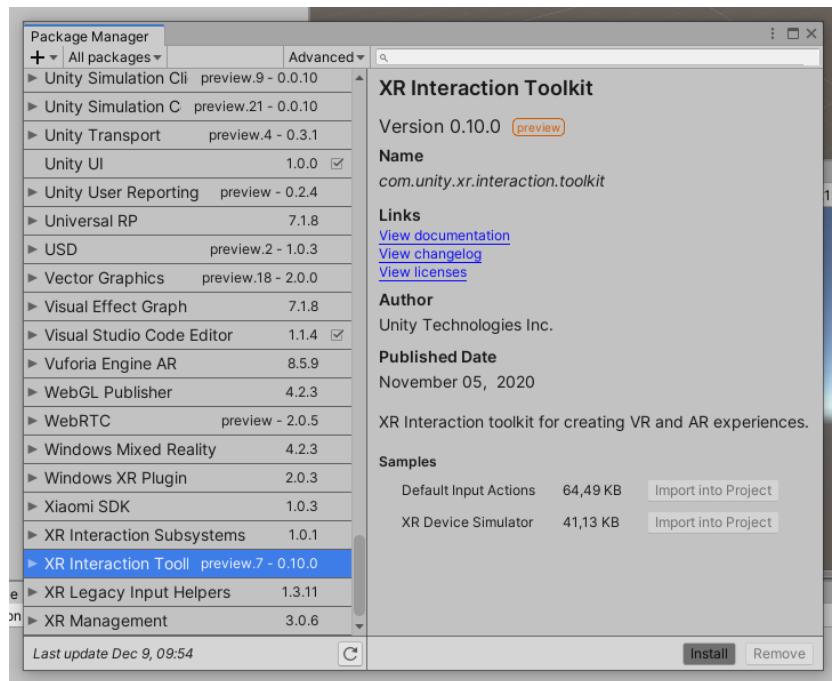


Рисунок 2.9 – Встановлення пакета «XR Interaction Tool»

Також для взаємодії з шоломами віртуальної реальності необхідно встановити SDK (Software Development Kit, комплект для розробки програмного забезпечення). Для цього відкриємо налаштування проєкту «Edit» - «Project Settings...» (рис. 2.10).

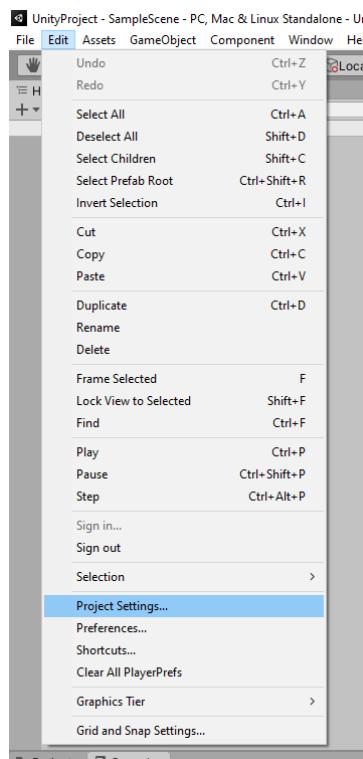


Рисунок 2.10 – Відкриття налаштувань проєкту

В налаштуваннях проєкту відкриваємо вкладку «XR Plugin Management» та натискаємо на кнопку «Install XR Plugin Management» для встановлення (рис. 2.11).

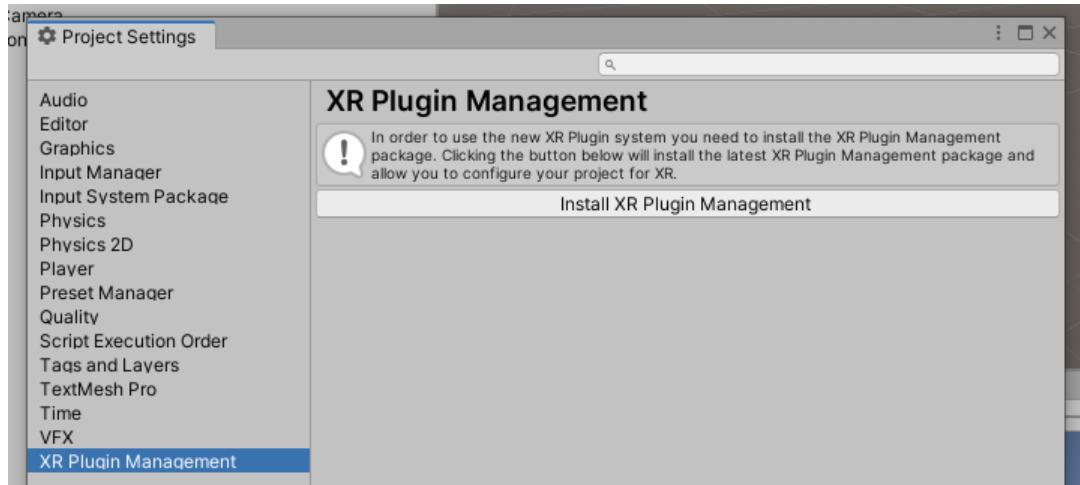


Рисунок 2.11 – Встановлення плагін «XR Plugin Management»

Після встановлення «XR Plugin Management» активуємо пункт «Initialize XR on Startup» для запуску XR плагінів під час відкриття проєкту за замовчанням (рис. 2.12).

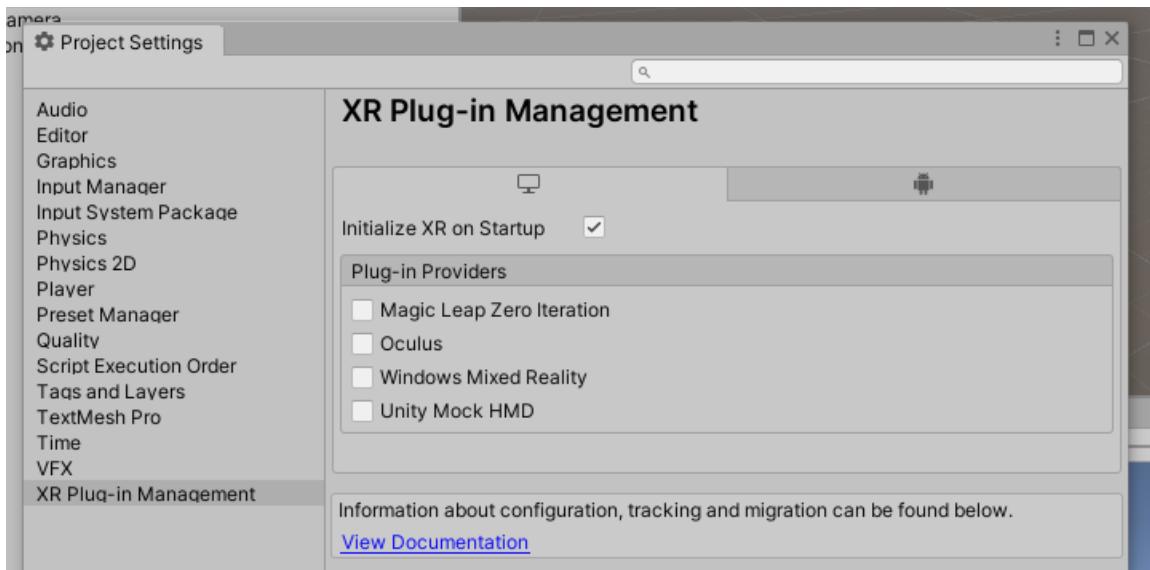


Рисунок 2.12 – Встановлений плагін «XR Plugin Management»

В налаштуваннях проєкту переходимо на вкладку Player та в пункті «XR Settings» активуємо підпункт «Virtual Reality Supported» (рис. 2.13).

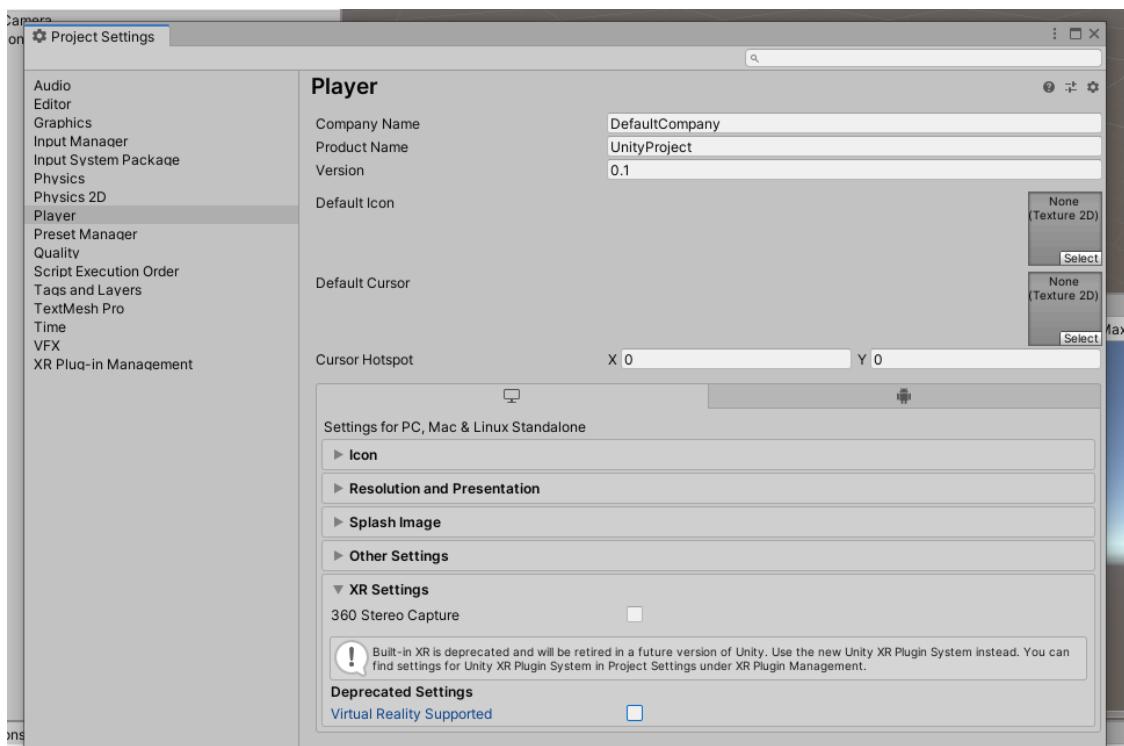


Рисунок 2.13 – Встановлення підтримки віртуальної реальності

Після встановлення підтримки віртуальної реальності з'явився пустий список SDK. Для додавання комплекту для розробки програмного забезпечення під різні шоломи віртуальної реальності необхідно натиснути на значок плюсу в правій нижній частині списку та вибрати необхідний набір розробника (рис. 2.14). В проєкт можна додати декілька SDK. Для цього встановлюємо їх по черзі.

Для роботи з різними шоломами необхідні різні набори розробника. Наприклад для роботи з Oculus Rift необхідно встановити Oculus SDK, а для роботи з Valve Index загальну бібліотеку OpenVR. Так як ми працюємо з обома цими шоломами, то встановимо Oculus SDK та OpenVR.

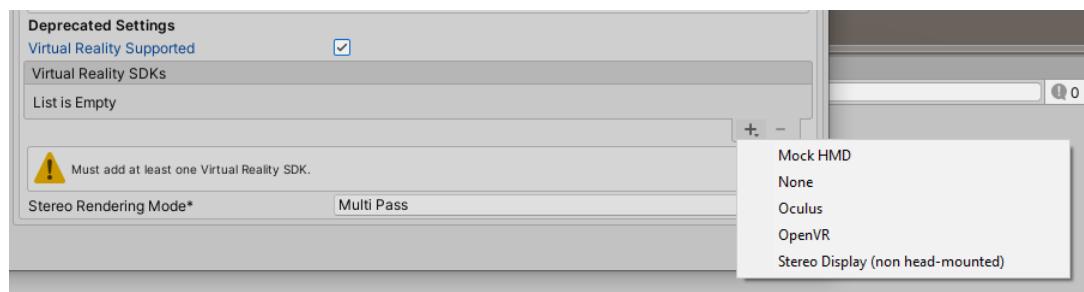


Рисунок 2.14 – Додавання наборів розробника (SDK)

Під час встановлення вибраних наборів необхідно підтвердити видалення попередньо встановлених версій натиснувши кнопку «Ok» (рис.2.15).

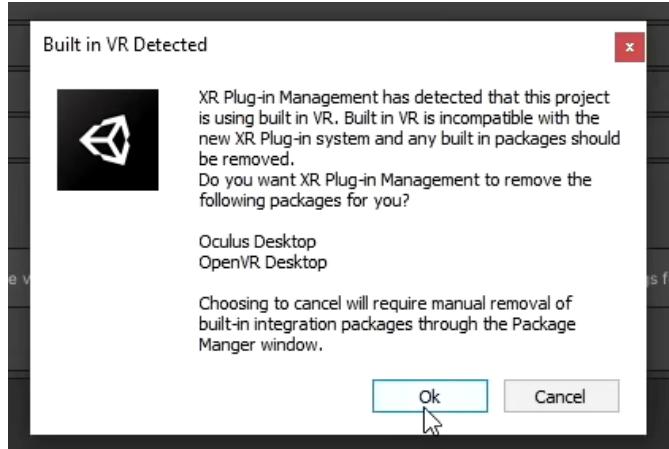


Рисунок 2.15 – Видалення попередньо встановлених версій наборів

В результаті виконаних дій встановлено необхідні набори розробника для роботи з шоломами віртуальної реальності Oculus Rift та Valve Index (рис. 2.16). Продовжимо роботу з проєктом.

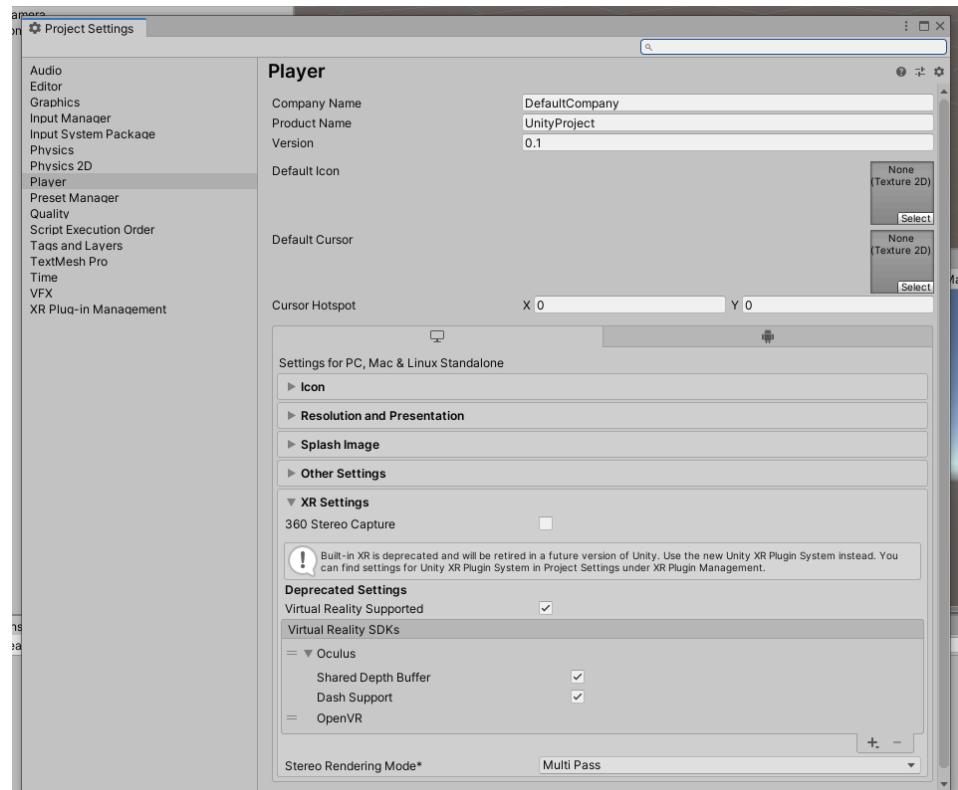


Рисунок 2.16 – Встановлені набори розробника в «XR Settings»

Додамо в сцену площину, яка буде виконувати роль полу для гравця. Для цього створимо об'єкт «Plane» натиснувши правою кнопкою миші в ієрархії проєкту та вибравши «Plane» в пункті «3D Object» (рис. 2.17-18).

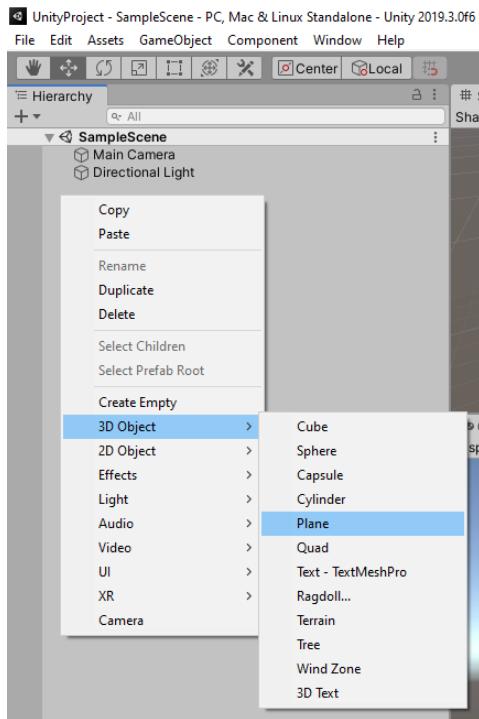


Рисунок 2.17 – Створення 3D-об'єкта площини

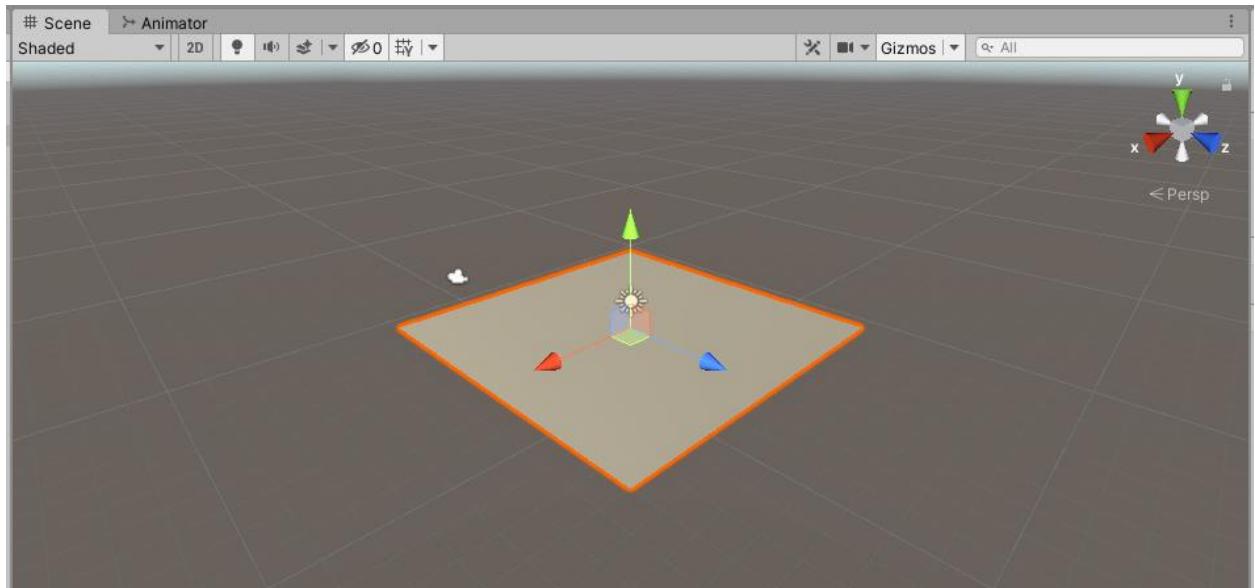


Рисунок 2.18 – Створена площа в сцені

В правій частині середовища розробки Unity відображене вікно з встановленими параметрами створеного об'єкта (рис. 2.19).

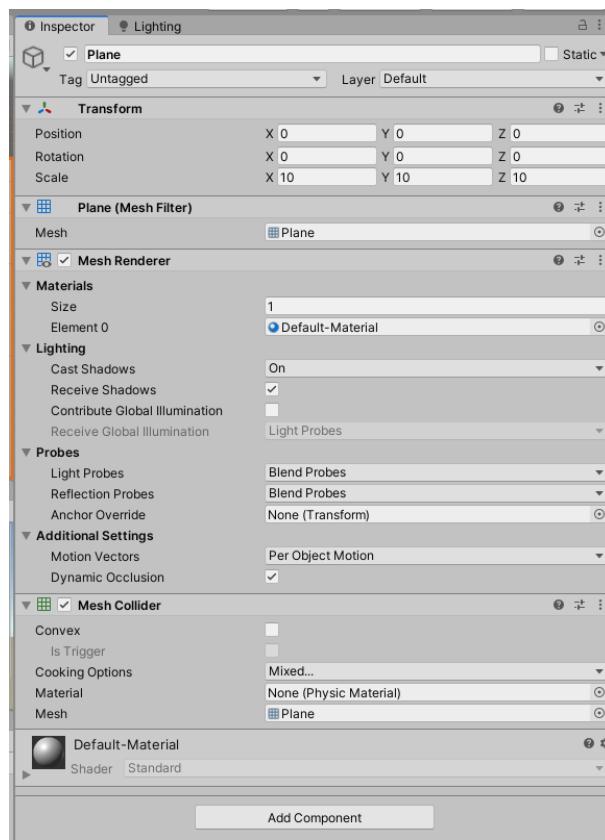


Рисунок 2.19 – Параметри створеної площини

Створимо матеріл для об'єкта та встановимо його цій площині. Для цього натиснемо правою кнопкою миші в папці «Assets» вікна «Project» та виберемо «Create» - «Material» (рис. 2.20).

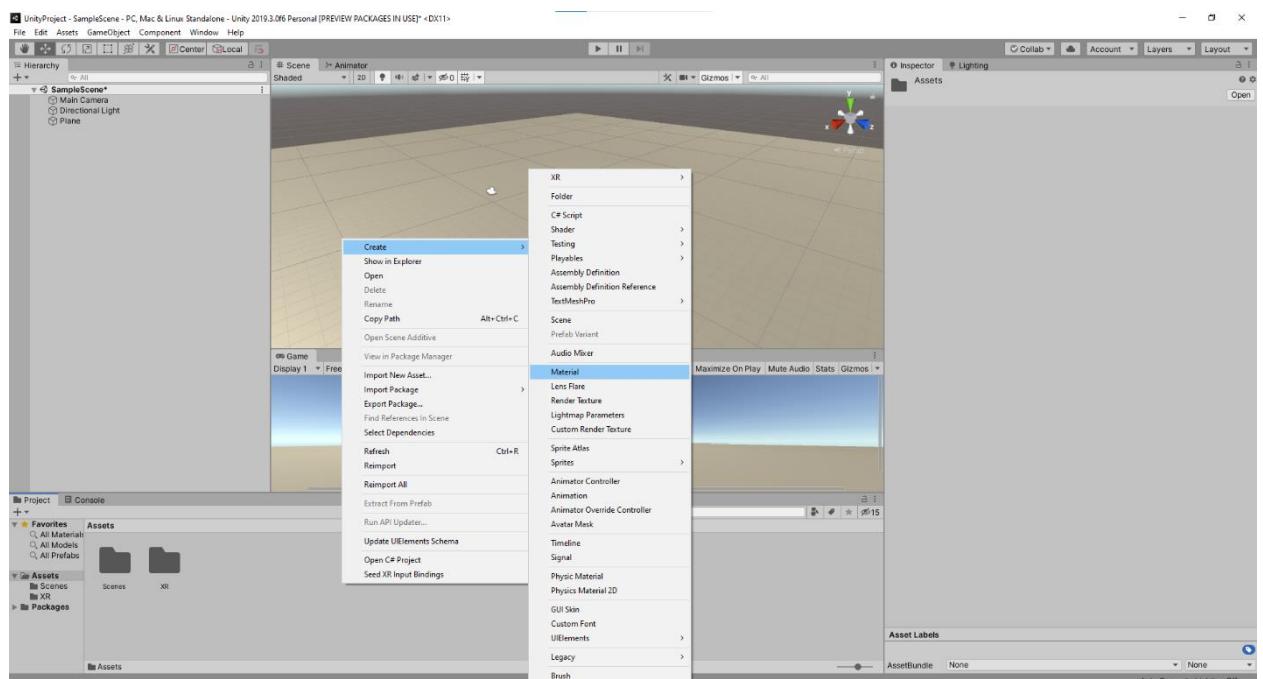


Рисунок 2.20 – Створення матеріалу для об'єкта

Змінемо назву для створеного матеріалу (рис. 2.21) та встановимо йому необхідну структуру та колір (рис. 2.22).

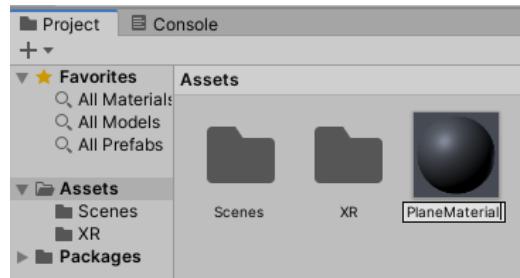


Рисунок 2.21 – Зміна назви матеріалу

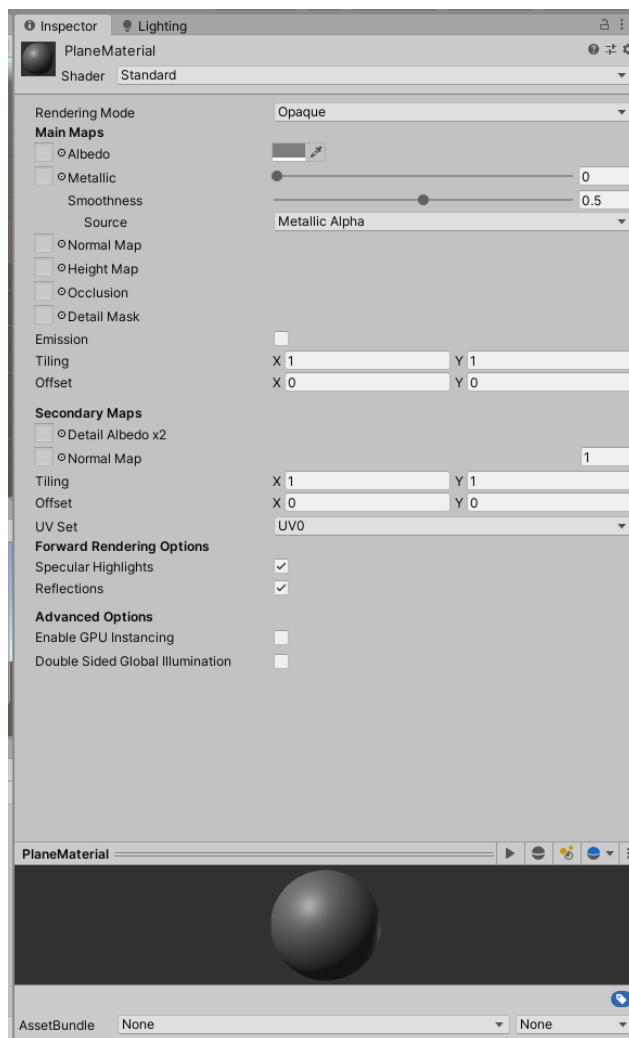


Рисунок 2.22 – Налаштування параметрів матеріалу

Встановимо створений матеріал в якості матеріалу для площини. Виберемо площину та перетягнемо матеріал з папки «Assets» в пункт налаштувань площини «Materials» (рис. 2.23).

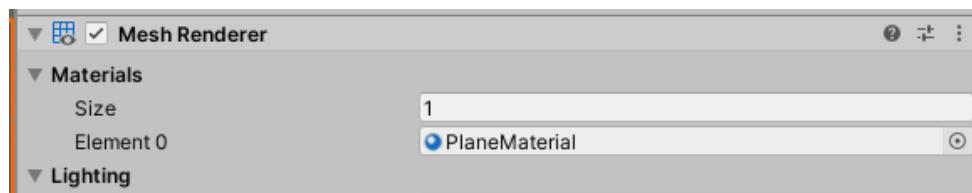


Рисунок 2.23 – Встановлення матеріалу

Видалимо створену за замовчанням камеру (рис. 2.24).

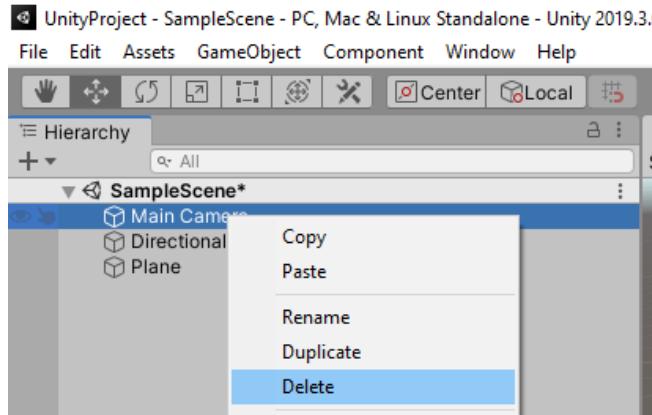


Рисунок 2.24 – Видалення об’єкта з ієархії проекта

Після видалення камери у вікні «Game» з’явиться повідомлення «No cameras rendering» (рис. 2.25).

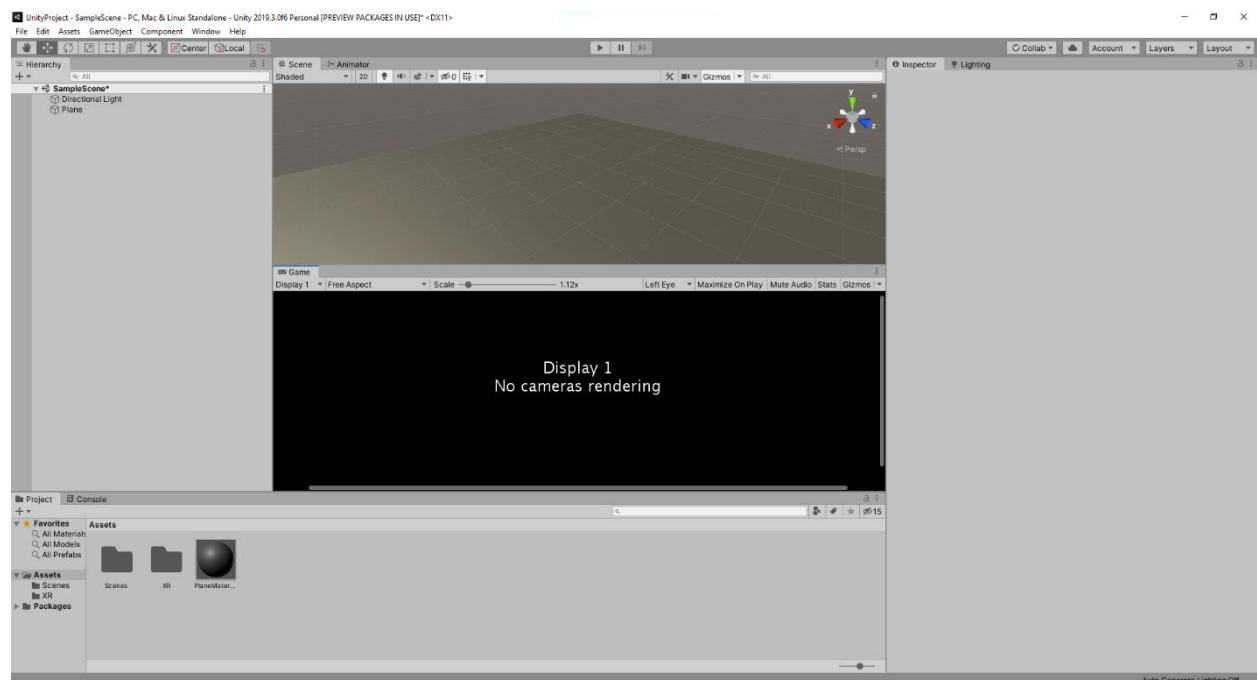


Рисунок 2.25 – Відображення інформації про відсутність рендеру камери

Створимо камеру, яка буде видавати зображення для користувача і буде його очима. Для цього створюємо в сцені пустий об'єкт (рис. 2.26).

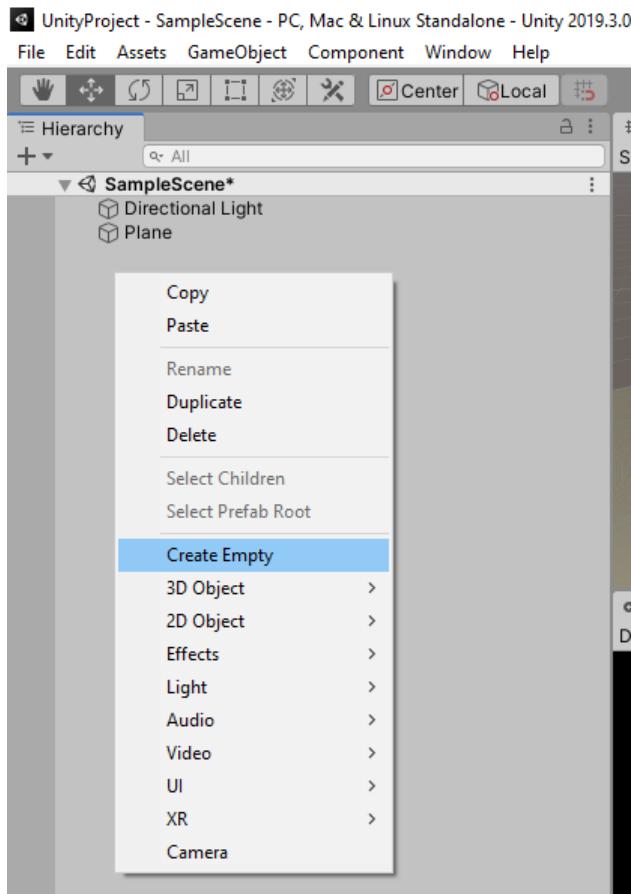


Рисунок 2.26 – Створення пустого об'єкта

Перейменуємо створений об'єкт в «VR Rig» (рис. 2.27).

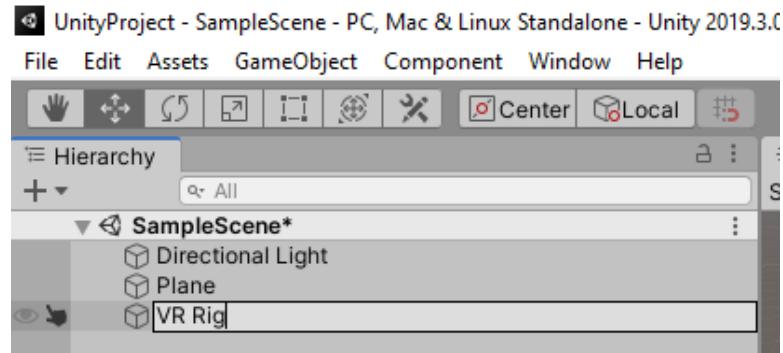


Рисунок 2.27 – Перейменування об'єкта

В параметрах «VR Rig» натискаємо на кнопку «Add Component» та додаємо компонент «XR Rig» (рис. 2.28-29).

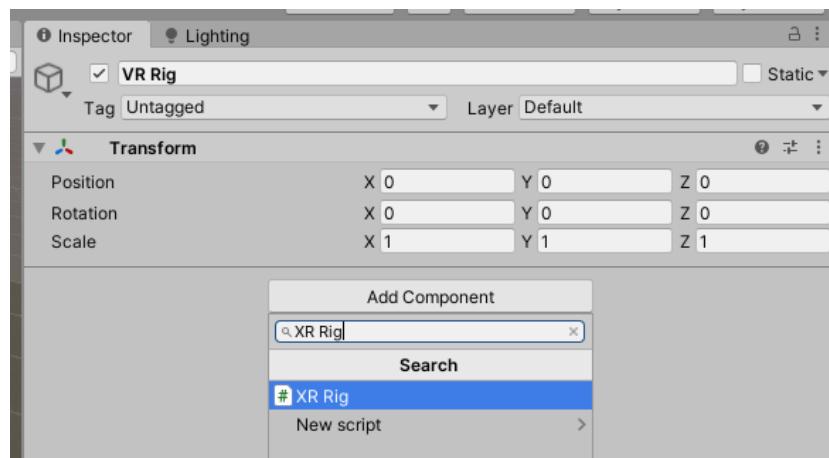


Рисунок 2.28 – Додавання компонента до об’єкта

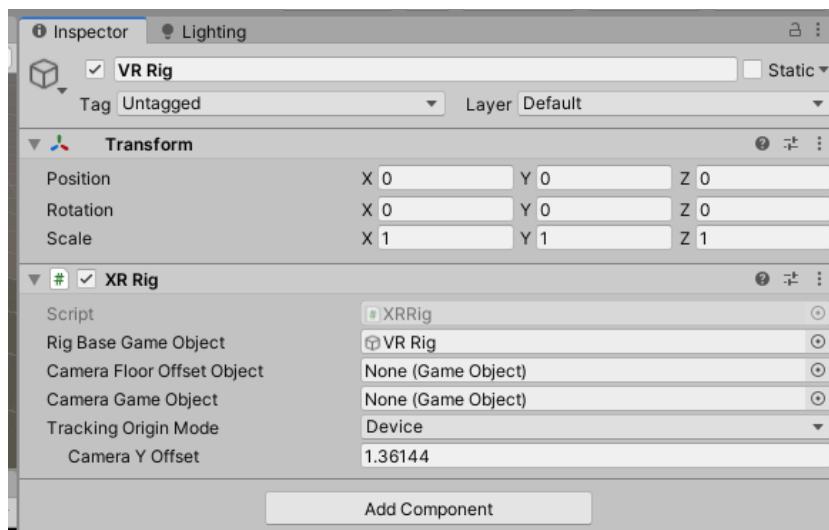


Рисунок 2.29 – Доданий компонент «XR Rig» в об’єкт «VR Rig»

Створюємо дочірній пустий об’єкт для «VR Rig» та перейменовуємо його в «Camera Offset» (рис. 2.30).

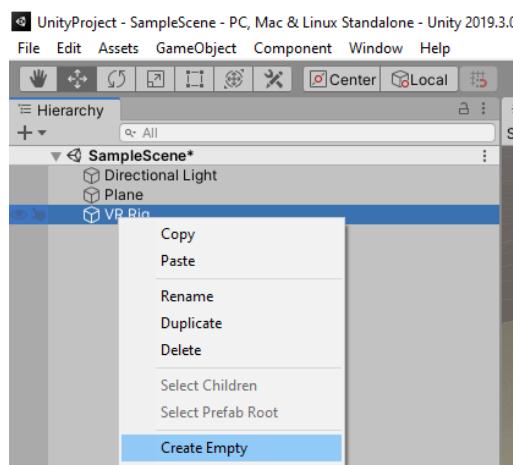


Рисунок 2.30 – Додавання дочірнього об’єкта

Для «Camera Offset» створюємо дочірній об'єкт типу «Camera» (рис. 2.31) та перейменовуємо його в «VR Camera» (рис. 2.32).

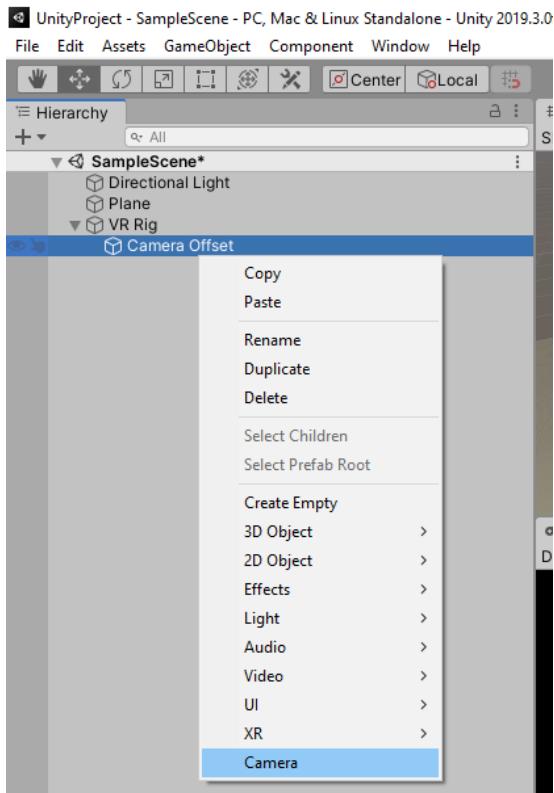


Рисунок 2.31 – Створення об'єкта «Camera»

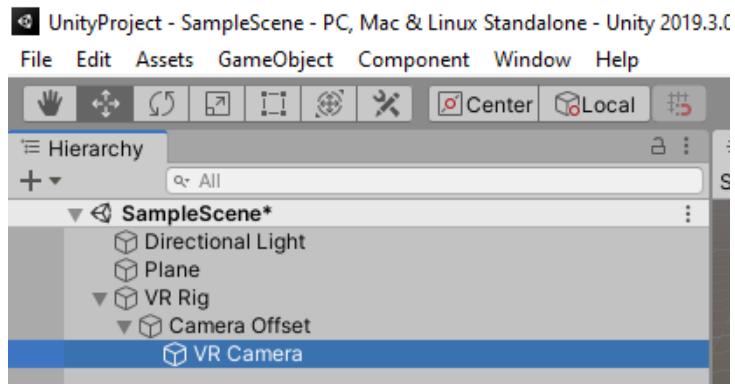


Рисунок 2.32 – Структура доданих об'єктів

До об'єкта «VR Camera» додаємо компонент «Tracked Pose Driver» (рис. 2.33) та виконуємо його налаштування (рис. 2.34).

В якості пристрою встановлюємо «Generic XR Device» – для того, щоб можна було підключити будь-який шолом і автоматично підтягнути налаштування під нього. В якості джерела позиції камери використовуємо

«Center Eye – HMD Reference» – для відображення середньої позиції очей, а в якості типу відслудковування «Rotation And Position» – позиціонування та повороти.

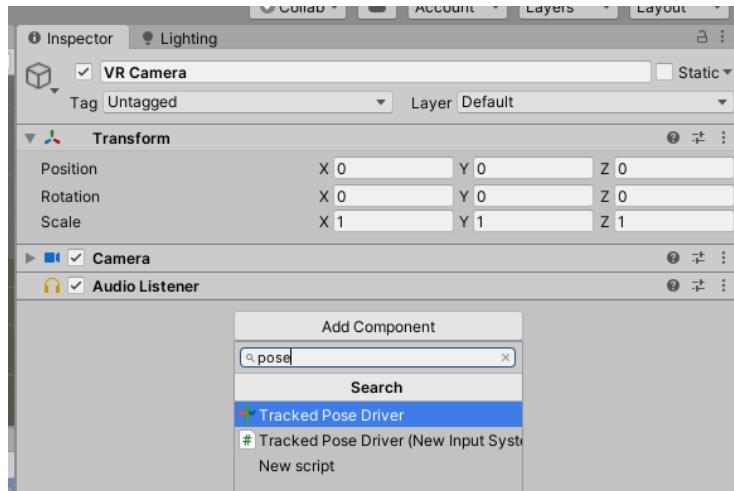


Рисунок 2.33 – Додавання компонента «Tracked Pose Driver»

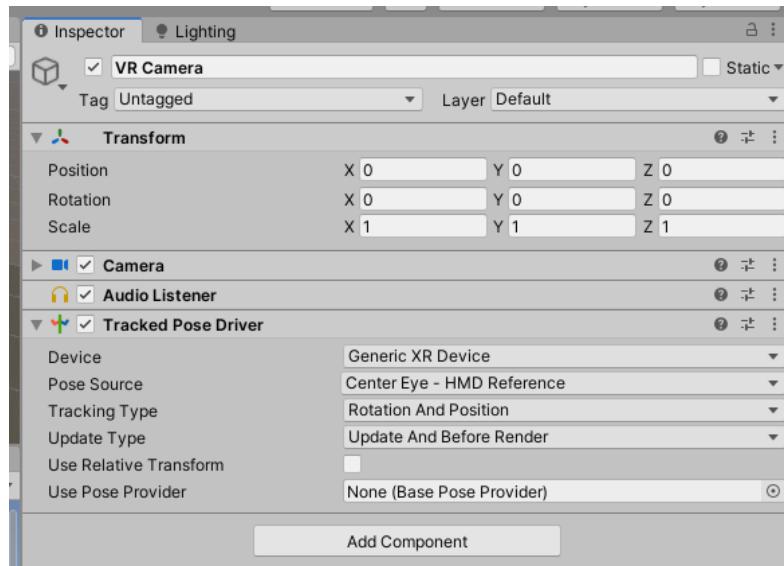


Рисунок 2.34 – Налаштування компонента «Tracked Pose Driver»

В налаштуваннях об’єкта «VR Rig» в якості базового ігрового об’єкта встановлюємо «VR Rig», в якості відступу камери від полу встановлюємо об’єкт «Camera Offset», в якості ігрового об’єкта камери встановлюємо об’єкт «VR Camera». Як режим відстеження джерела встановлюємо «Floor», для того, щоб позиція Camera Offset відраховувалася від точки полу, налаштованої до запуску Unity (рис. 2.35).

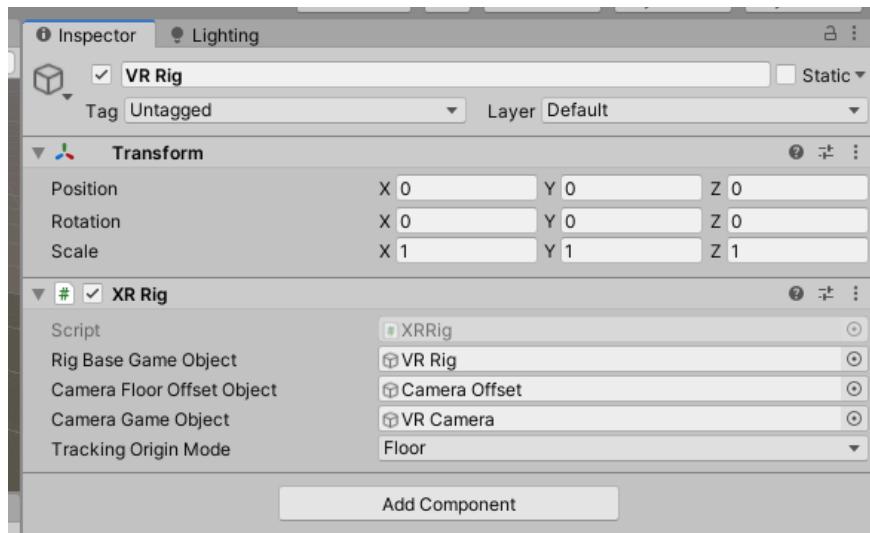


Рисунок 2.35 – Налаштування об’єкта «VR Rig»

Після виконаних налаштувань, можна запустити розроблений застосунок. Під час обертання головою, буде змінюватися картинка, тобто застосунок буде реагувати на повороти голови користувача та на його переміщення в просторі.

Додамо відображення рук, для наглядного прикладу зробимо їх у вигляді простих кубів.

Спочатку необхідно створити пусті об’єкти які будуть контейнерами для моделей рук. В об’єкті «Camera Offset» створюємо два дочірніх пустих об’єкти. Перейменуємо їх у «Left Hand» та «Right Hand» (рис. 2.36).



Рисунок 2.36 – Ієархія проєкту з пустими об’єктами для додавання рук

Виділимо обидві руки (натиснути та утримувати клавішу Ctrl та лівою клавішею миші вибрати необхідні об’єкти) та в їх параметрах додамо компонент «XR Controller (Device-based)» (рис. 2.37).

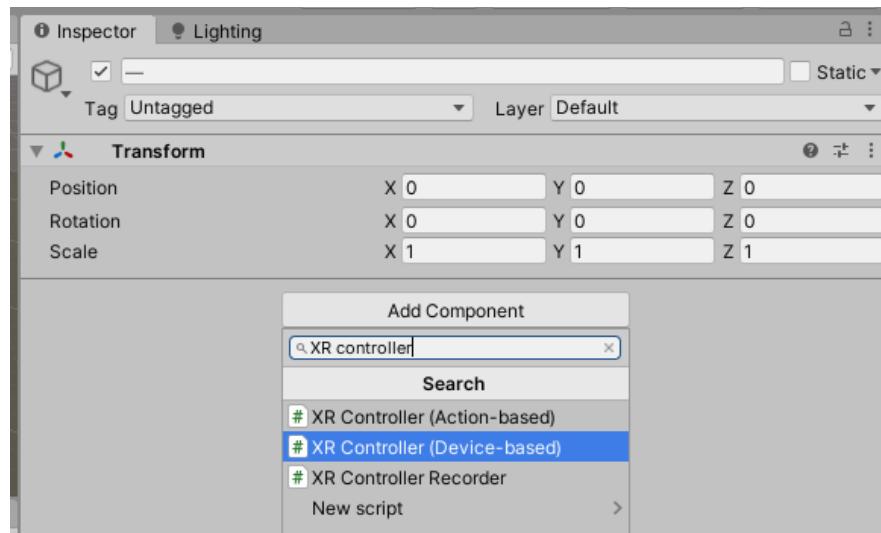


Рисунок 2.37 – Додавання компонента «XR Controller (Device-based)»

Далі для кожної руки встановимо необхідне значення параметра «Controller Node». Для лівої руки встановимо значення «Left Hand», а для правої – «Right Hand» (рис. 2.38).

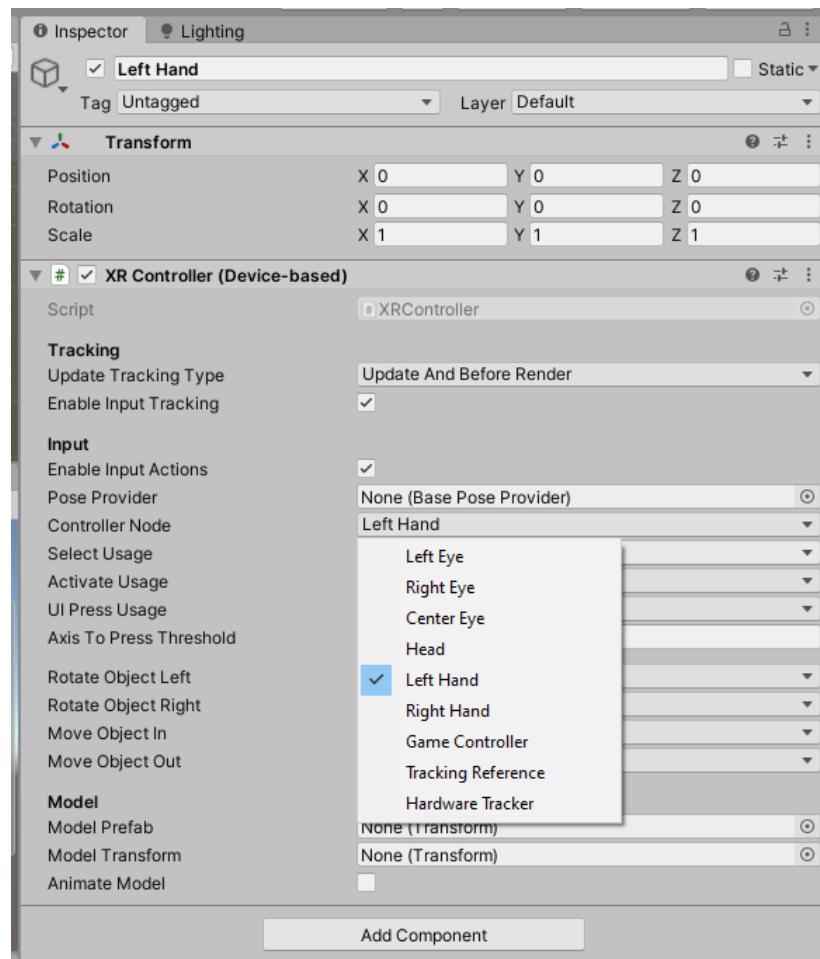


Рисунок 2.38 – Встановлення параметра «Controller Node»

У разі запуску застосунку у вікні сцени буде відображене положення контролера в просторі сцени (рис. 2.39).

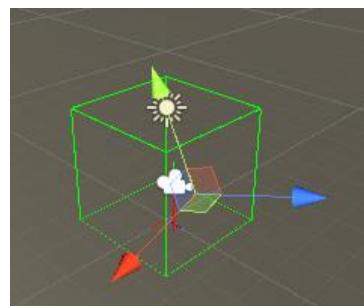


Рисунок 2.39 – Відображення контролера в просторі сцени

Є два основних терміни, які використовуються в Unity. Це асset (asset) та префаб (prefab).

Асsetи - це компоненти, які є графікою, звуковим супроводом або скриптами. Вони прикріплюються до об'єктів і становлять важливу частину гри.

Префаб – це особливий тип асsetів, що дозволяє зберігати весь ігровий об'єкт з усіма компонентами та значеннями властивостей. Префаб виступає в ролі шаблону для створення екземплярів об'єкта, що зберігається в сцені.

Створимо префаб (prefab) для зовнішнього вигляду рук. Для цього в сцені створимо пустий об'єкт та назовемо його «Controller». Для створеного об'єкта створимо дочірній елемент типу «3D Object» – «Cube».

Розроблена ієархія об'єктів проекту наведена на рисунку 2.40.

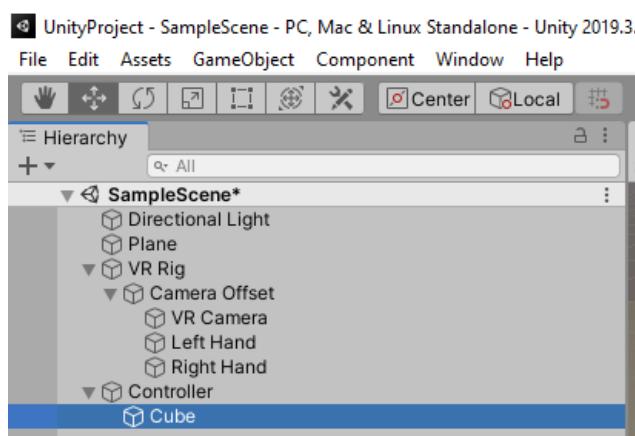


Рисунок 2.40 – Розроблена ієархія об'єктів

Встановимо необхідні параметри для кубу (рис. 2.41)

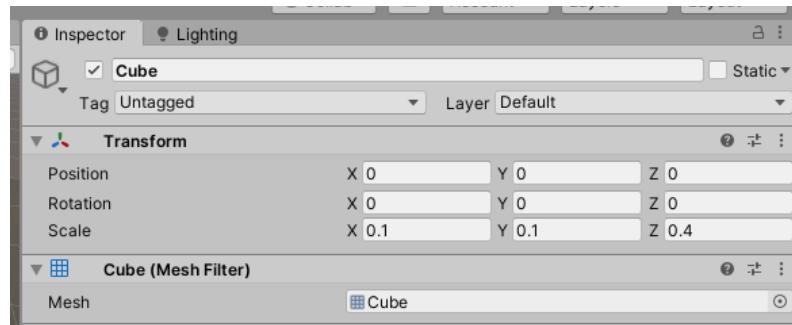


Рисунок 2.41 – Встановлення параметрів кубу

Для того, щоб об’єкт «Controller» став префабом необхідно перетягнути його лівою кнопкою миші зі сцени у папку «Assets» вікна «Project» (рис. 2.42) та видалити цей об’єкт зі сцени.

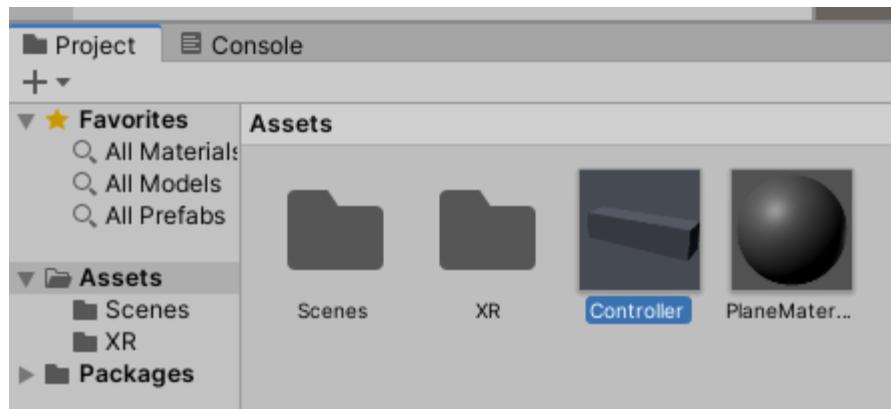


Рисунок 2.42 – Об’єкт сцени тепер є префабом

Далі необхідно виділити об’єкти, що відповідають за руки (рис. 2.43).

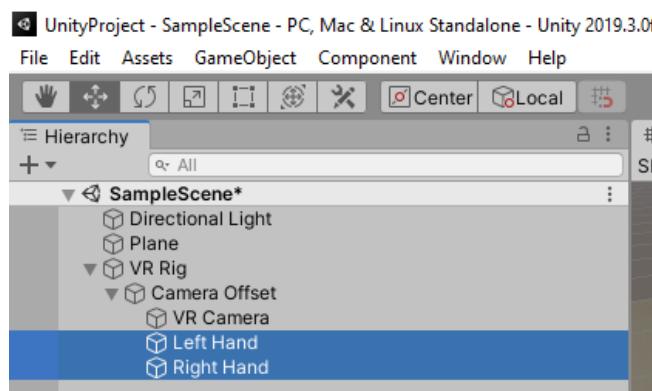


Рисунок 2.43 – Виділені об’єкти, що відповідають за руки

В параметрах «Left Hand» та «Right Hand» переходимо до компонента «XR Controller (Device-based)» та в якості «Model Prefab» встановлюємо створений нами префаб (рис. 2.44).

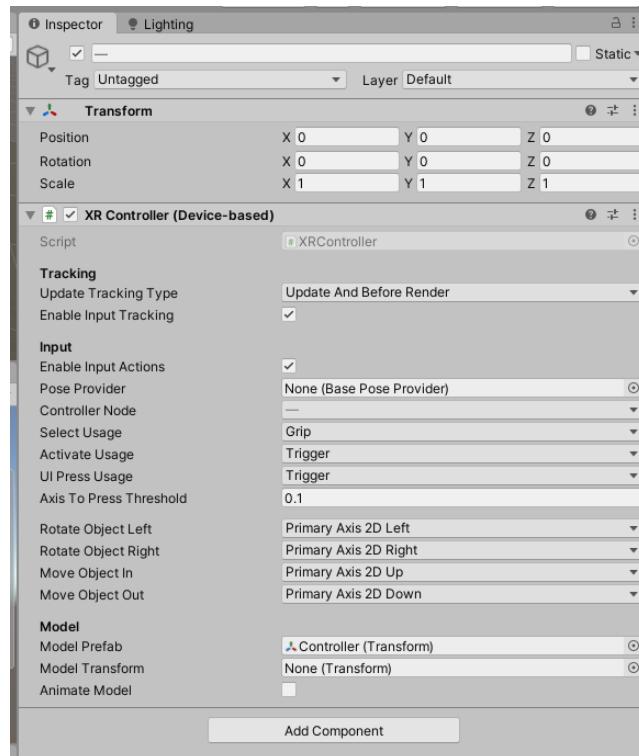


Рисунок 2.44 – Встановлення префабу для «Left Hand» та «Right Hand»

Після запуску застосунку можна переміщувати контролери, позиція префаба в ігрому просторі буде змінюватися відповідно до рухів контролерів (рис. 2.45).

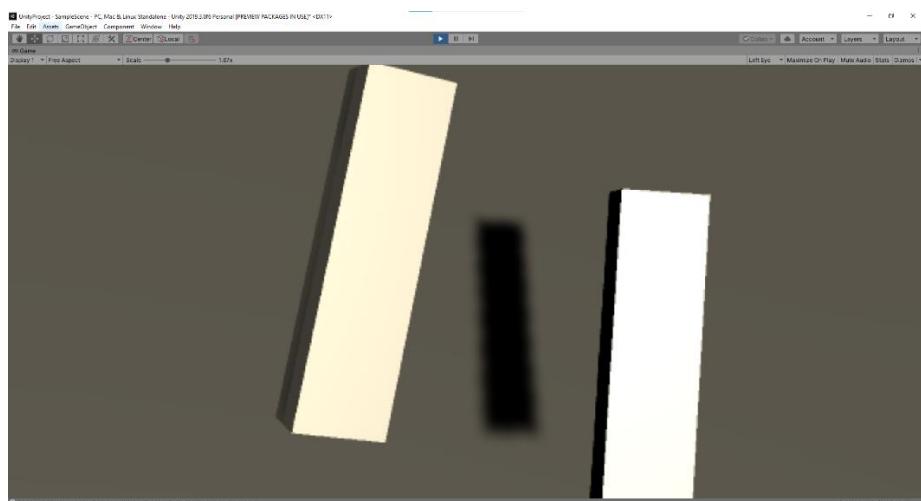


Рисунок 2.45 – Запущений застосунок та відображені моделі рук

Додамо до сцени модель фізичного об'єкта, з яким зможе взаємодіяти користувач.

В сцені створимо 3D-об'єкт «Cube» та розмістимо поверх нього 3D-об'єкт «Sphere». Встановимо йому параметри, як зображенено на рисунку 2.46.

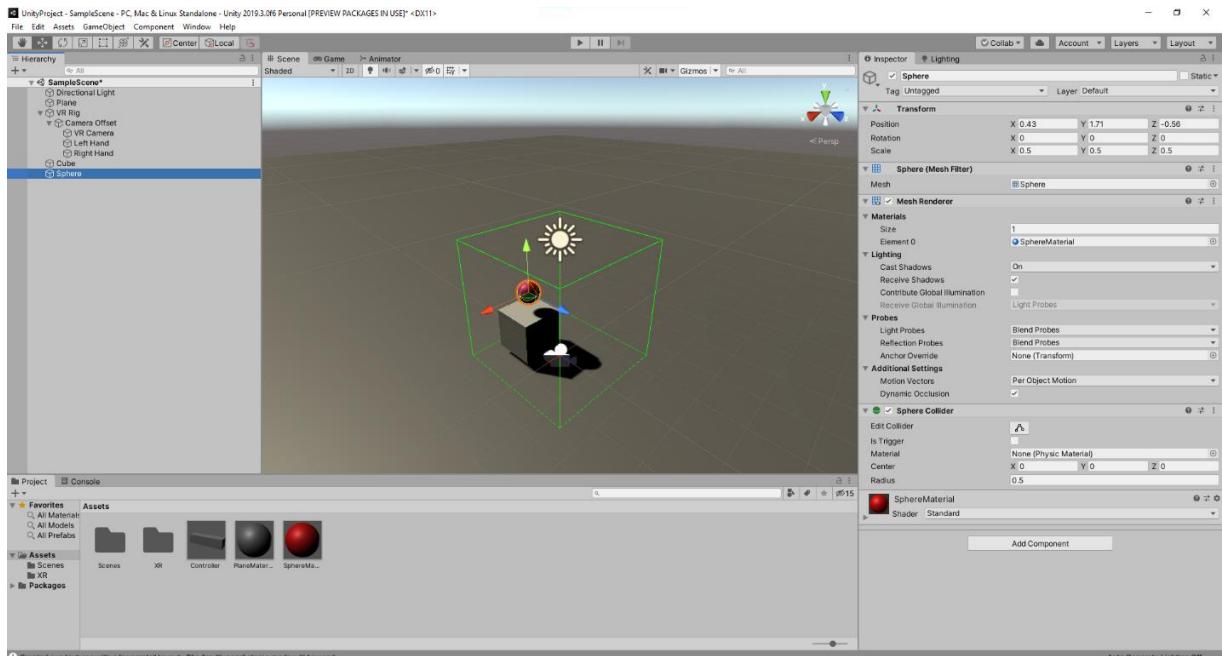


Рисунок 2.46 – Сцена з 3D-об'єктами «Cube» та «Sphere»

Додамо компоненти «XR Direct Interactor» та «Sphere Collider» до об'єктів, що відповідають за контролери (руки), та встановимо відповідні параметри (рис. 2.47).

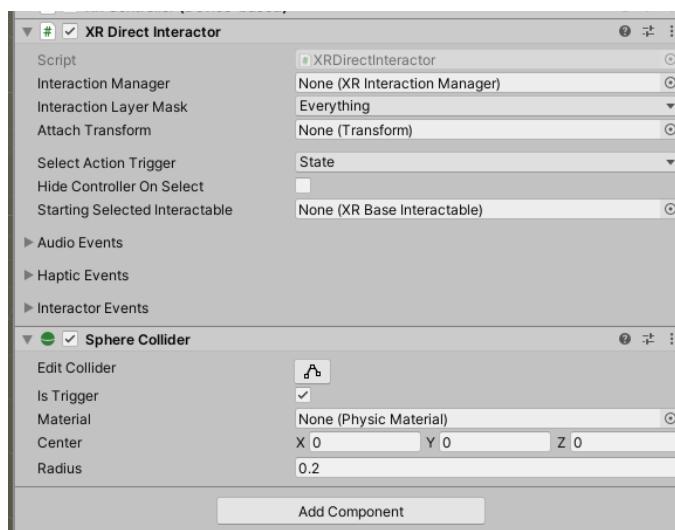


Рисунок 2.47 – Налаштування «XR Direct Interactor» та «Sphere Collider»

Для об'єкта «Sphere» додамо компоненти «Rigidbody» для моделювання фізичного об'єкта та «XR Grab Interactable» для можливості взаємодії з користувачем (рис. 2.48). Запустимо застосунок та перевіримо можливості взаємодії з шаром (рис. 2.49).

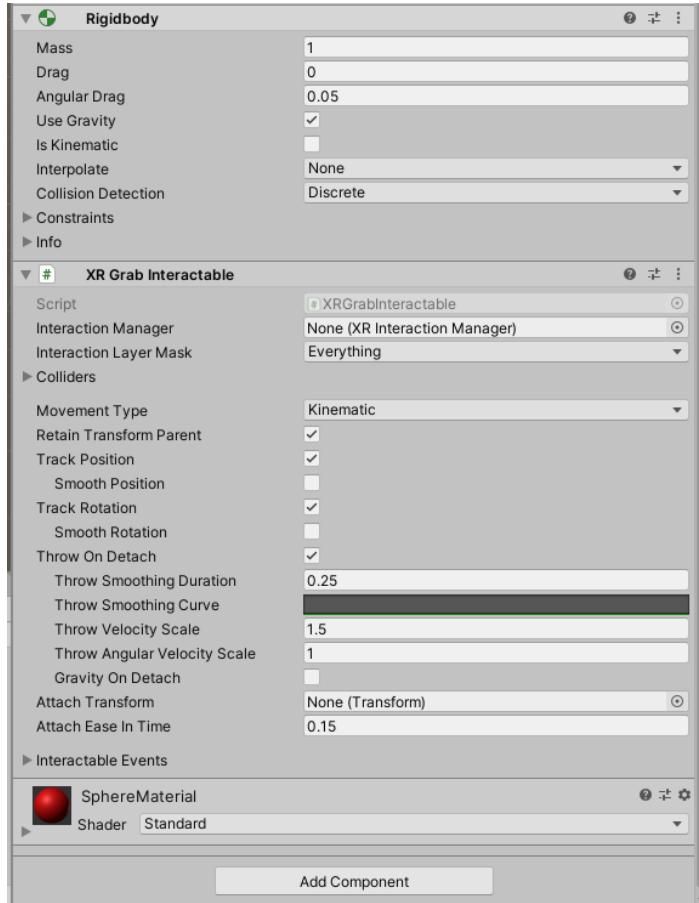


Рисунок 2.48 – Налаштування «Rigidbody» та «XR Grab Interactable»

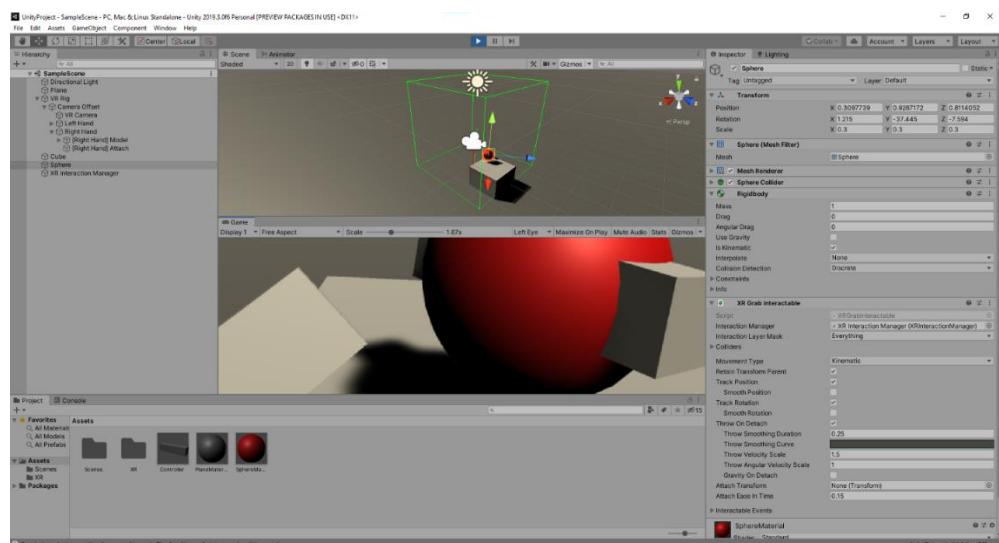


Рисунок 2.49 – Взаємодія з об'єктами в застосунку

В будь-якому проекті на Unity можна використовувати вже готові асети, а не створювати нові скрипти чи проектувати 3D-об'єкти. Додамо об'єкти з асету, імпортувавши його з магазину асетів Unity.

Для відкриття магазину асетів необхідно вибрати пункт «Window» - «Asset Store» (рис. 2.50).

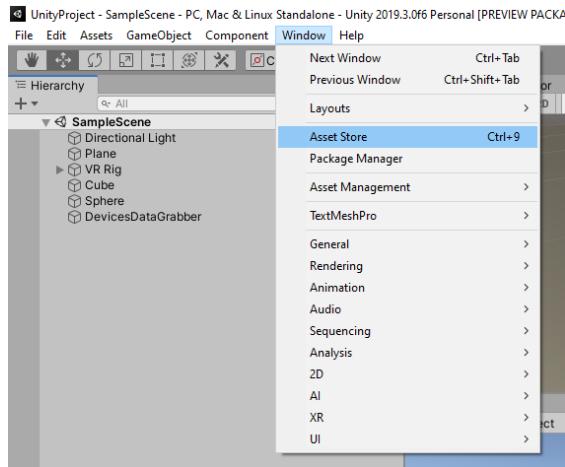


Рисунок 2.50 – Відкриття магазину асетів Unity

У відритом вікні з магазином асетів можна вибрати необхідний, або додати його до власної бібліотеки. Для імпорту асету, його перш за все необхідно завантажити. Натиснемо на кнопку «Download» бажаного асету або префабу (2.51). Після завантаження асету на комп’ютер необхідно натиснути кнопку «» для початку імпорту в проект (рис. 2.52).

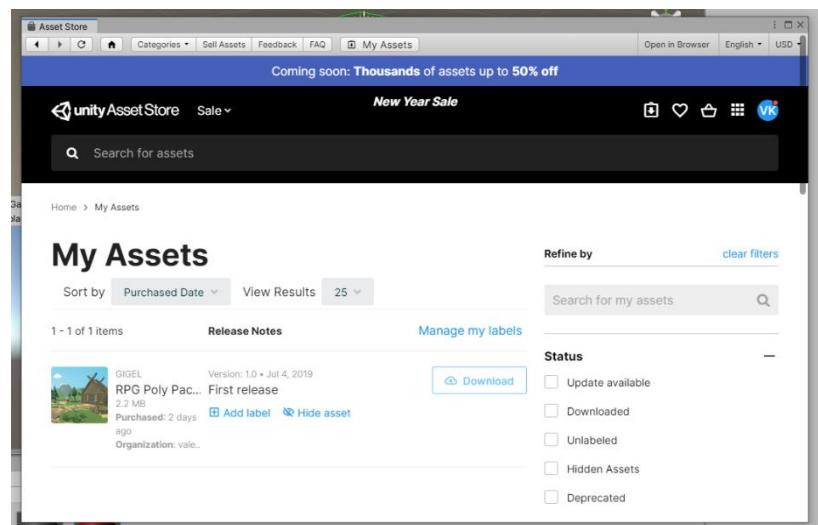


Рисунок 2.51 – Завантаження асету

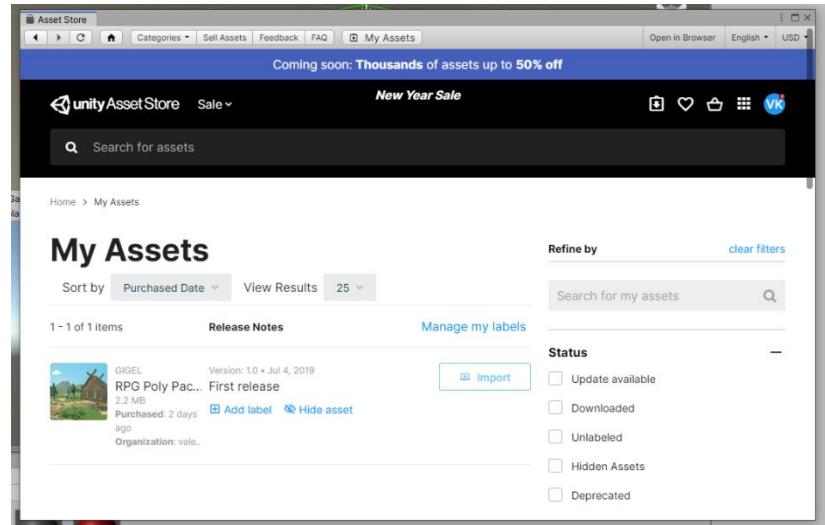


Рисунок 2.52 – Імпорт асету

У відкритому вікні імпорту пакету Unity необхідно вибрати бажані компоненти до імпорту та натиснути кнопку «Import» (рис. 2.53).

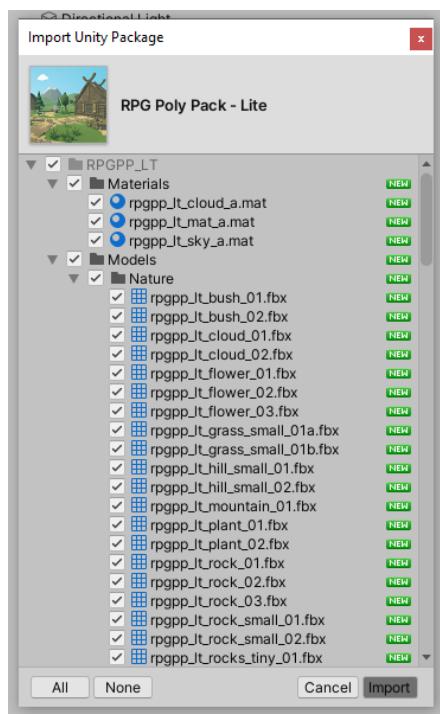


Рисунок 2.53 – Імпорт асету в проект

Додамо до нашого проекту вже готову сцену. Для цього відкриємо імпортовані частини асету, відкриємо сцену (рис. 2.54), виділимо необхідні об'єкти та скопіюємо їх (рис. 2.55). Відкриємо сцену нашого проекту та вставимо скопійовані об'єкти (рис. 2.56). Налаштуємо їх параметри.

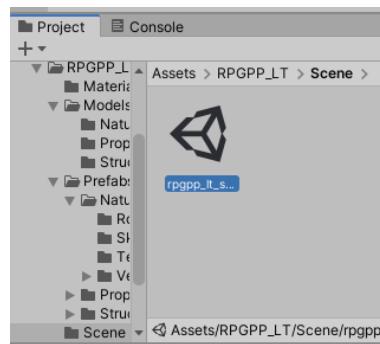


Рисунок 2.54 – Відкриття сцени з асетеу

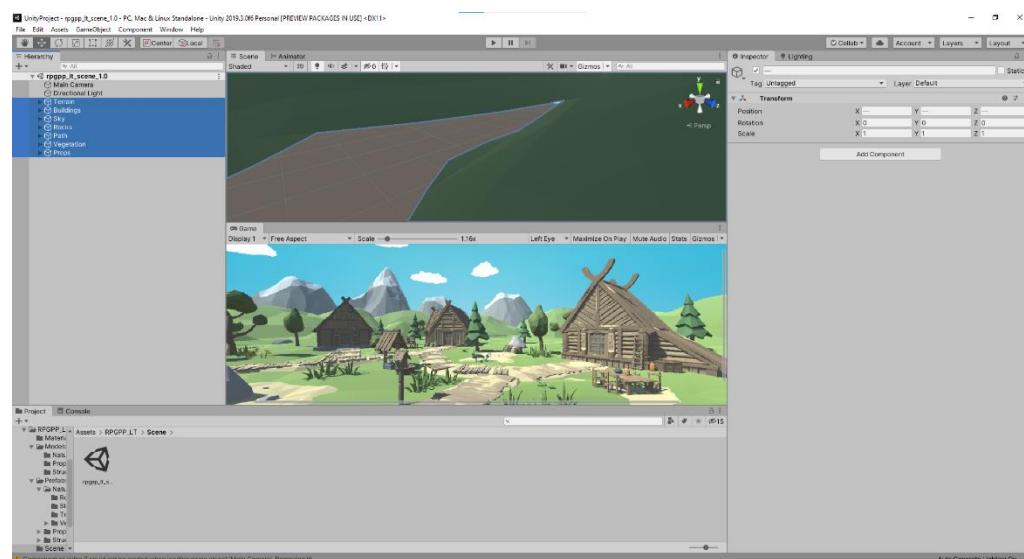


Рисунок 2.55 – Копіювання об’єктів сцени

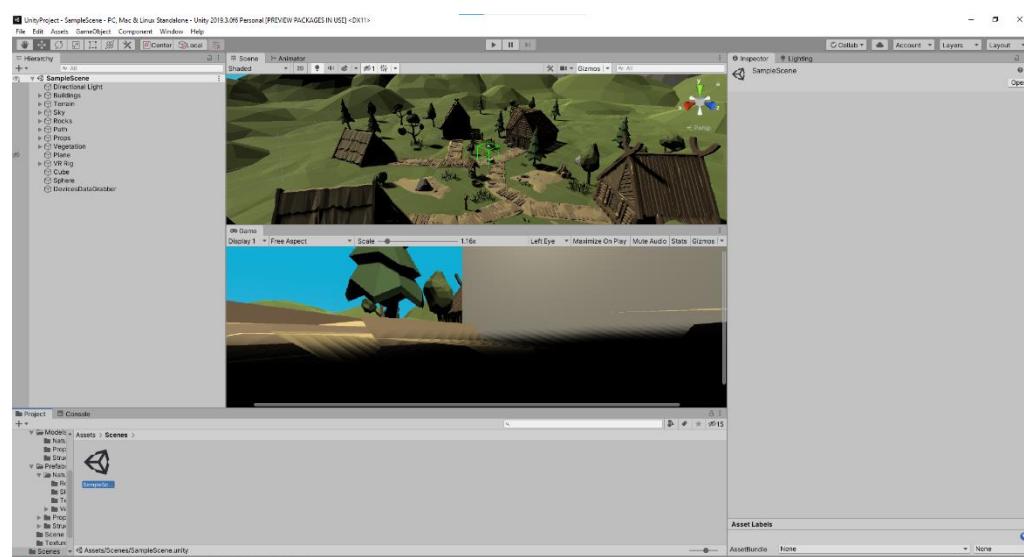


Рисунок 2.56 – Додавання скопійованих об’єктів до головної сцени

Запустивши проект переконаємося в тому, що нові об’єкти правильно відображаються.

Для додавання можливості зчитування даних з пристройів, а саме: шолому та контролерів, необхідно написати код та зберегти його в скриптах, які будуть автоматично запускатися під час роботи застосунку.

Перш за все необхідно встановити відповідні налаштування проєкту, для кращої роботи зі скриптами. Для цього відкрито налаштування переваг проєкту: «Edit» - «Preferences» (рис. 2.57).

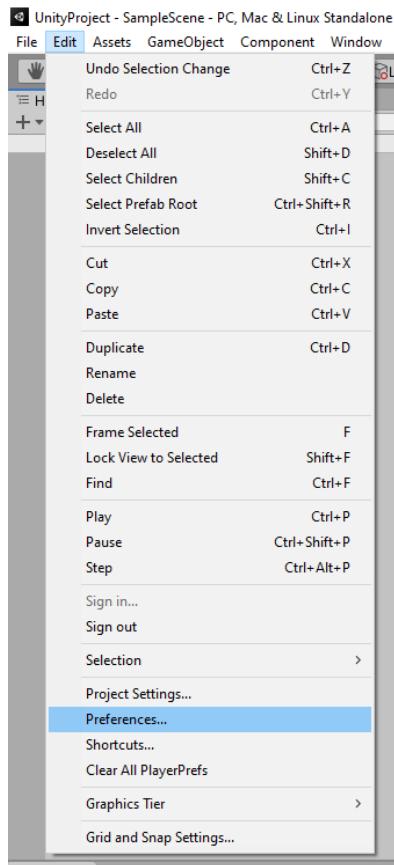


Рисунок 2.57 – Налаштування переваг проєкту

Далі необхідно відкрити вкладку зі сторонніми інструментами та в якості інструмента для роботи зі скриптами обрати бажану сторонню програму. Було обрано Visual Studio 2019 Community Edition, завдяки його перевагам та простоті використання для роботи з мовою C# (скрипти в середовищі розробки Unity зазвичай пишуться мовою C#). Також необхідно активувати пункт «Generate all .csproj file» для автоматичного створення необхідних для правильної роботи проєкта Visual Studio файлів та належної роботи доповнювача коду IntelliSense (рис. 2.58).

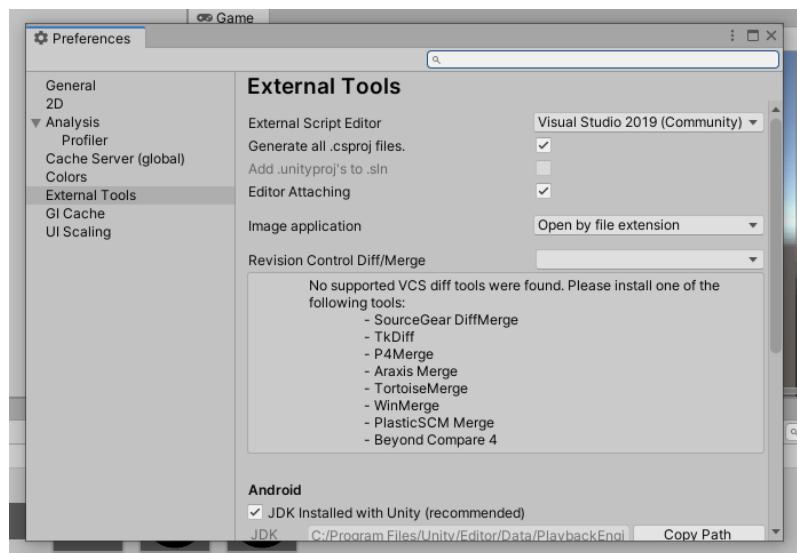


Рисунок 2.58 – Налаштування використовуваних сторонніх інструментів

Для створення скрипту та додавання його автоматичного запуску зі сценою необхідно створити пустий об'єкт в цій сцені (рис. 2.59), додати до нього компонент «Script» («New script») (рис. 2.60) та ввести для нього бажану назву (рис. 2.61). Параметри створеного об'єкта наведено на рисунку 2.62.

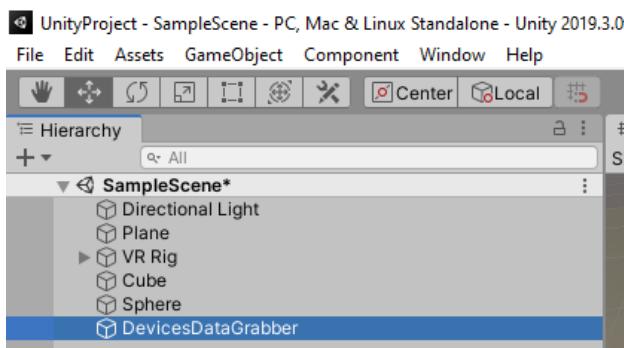


Рисунок 2.59 – Додавання пустого об'єкта для додавання скрипту

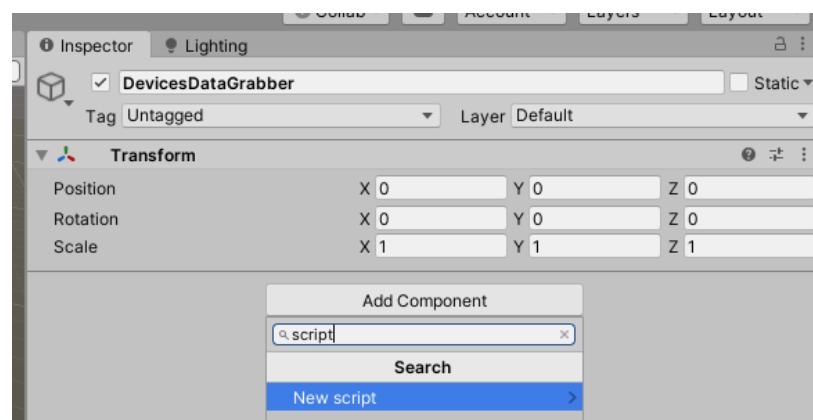


Рисунок 2.60 – Додавання компонента «Script»

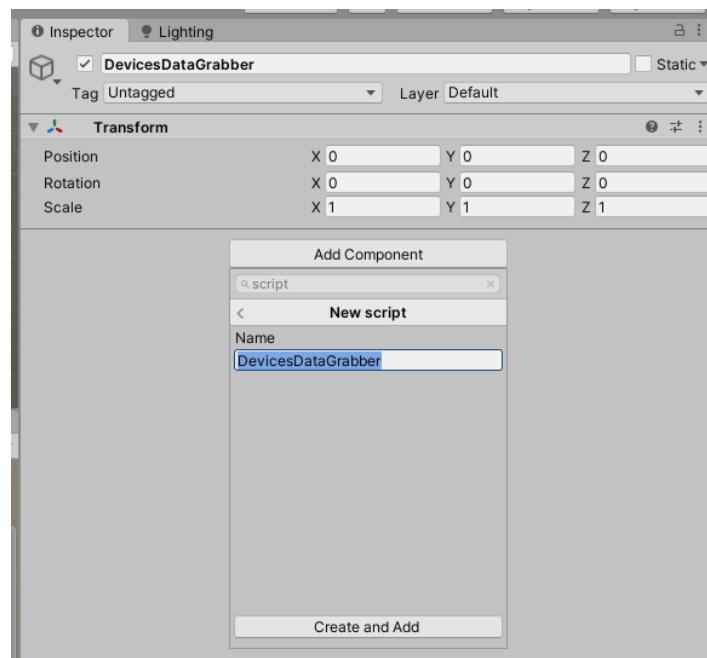


Рисунок 2.61 – Задання бажаної назви для створюваного скрипта

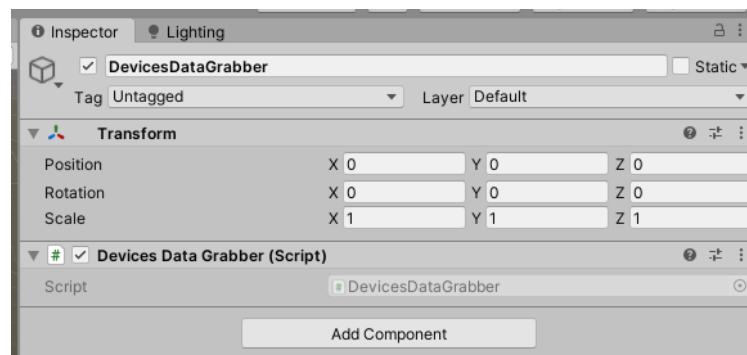


Рисунок 2.62 – Пустий об’єкт з прив’язаним скриптом

Після додавання компонента «Script» створено скрипт з введеною назвою, який розташовано у вікні «Project» в папці «Assets» (рис. 2.63). Для відкриття скрипта з метою редагування необхідно натиснути двічі лівою кнопкою миші на файлі зі скриптом (рис. 2.64).



Рисунок 2.63 – Створений скрипт в папці «Assets»

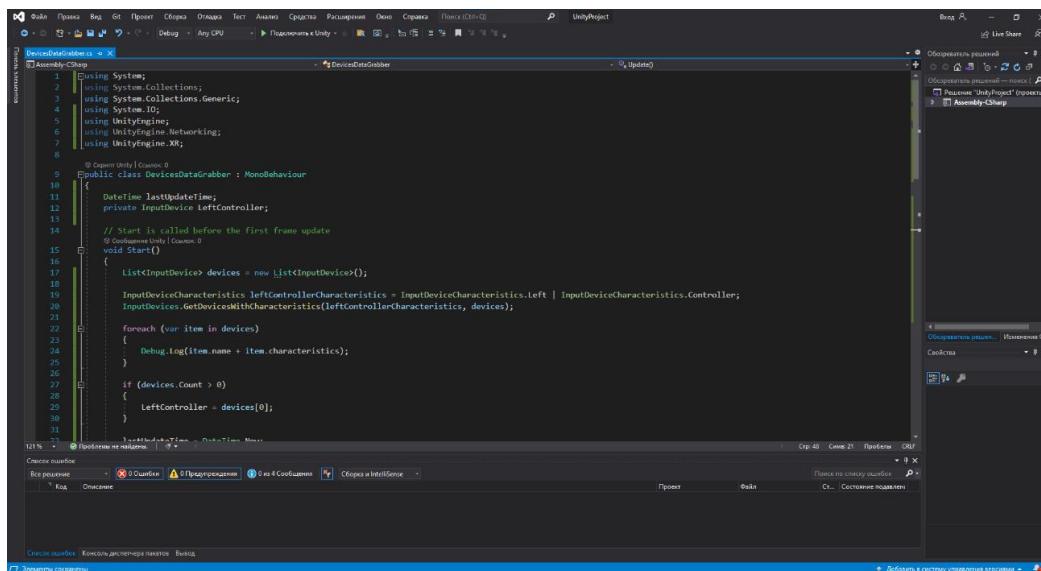


Рисунок 2.64 – Відкритий скрипт для редагування в Visual Studio

Для шолому віртуальної реальності Oculus Rift під час роботи проєкту в середовищі розробки Unity можна отримати доступ до даних з пристрой, які автоматично підтягуються за допомогою SDK Oculus (рис. 2.65-66).

XR Interaction Debugger			
Input Devices			
Name	Role	Type	Value
Devices			
Oculus Rift	Generic		
TrackingState		System.UInt32	63
IsTracked		System.Boolean	True
DevicePosition		UnityEngine.Vector3	(0.5, 1.2, 0.1)
DeviceRotation		UnityEngine.Quaternion	(-0.2, 0.0, 0.2, -1.0)
DeviceVelocity		UnityEngine.Vector3	(0.0, 0.0, 0.0)
DeviceAngularVelocity		UnityEngine.Vector3	(0.1, 0.1, 0.0)
DeviceAcceleration		UnityEngine.Vector3	(0.1, -0.2, 0.0)
DeviceAngularAcceleration		UnityEngine.Vector3	(-3.1, -6.0, -1.9)
LeftEyePosition		UnityEngine.Vector3	(0.5, 1.2, 0.1)
LeftEyeRotation		UnityEngine.Quaternion	(-0.2, 0.0, 0.2, -1.0)
LeftEyeVelocity		UnityEngine.Vector3	(0.0, 0.0, 0.0)
LeftEyeAngularVelocity		UnityEngine.Vector3	(0.1, 0.1, 0.0)
LeftEyeAcceleration		UnityEngine.Vector3	(0.1, -0.2, 0.0)
LeftEyeAngularAcceleration		UnityEngine.Vector3	(-3.1, -6.0, -1.9)
RightEyePosition		UnityEngine.Vector3	(0.5, 1.2, 0.1)
RightEyeRotation		UnityEngine.Quaternion	(-0.2, 0.0, 0.2, -1.0)
RightEyeVelocity		UnityEngine.Vector3	(0.0, 0.0, 0.0)
RightEyeAngularVelocity		UnityEngine.Vector3	(0.1, 0.1, 0.0)
RightEyeAcceleration		UnityEngine.Vector3	(0.1, -0.2, 0.0)
RightEyeAngularAcceleration		UnityEngine.Vector3	(-3.1, -6.0, -1.9)
CenterEyePosition		UnityEngine.Vector3	(0.5, 1.2, 0.1)
CenterEyeRotation		UnityEngine.Quaternion	(-0.2, 0.0, 0.2, -1.0)
CenterEyeVelocity		UnityEngine.Vector3	(0.0, 0.0, 0.0)
CenterEyeAngularVelocity		UnityEngine.Vector3	(0.1, 0.1, 0.0)
CenterEyeAcceleration		UnityEngine.Vector3	(0.1, -0.2, 0.0)
CenterEyeAngularAcceleration		UnityEngine.Vector3	(-3.1, -6.0, -1.9)

Рисунок 2.65 – Дані з шолому віртуальної реальності Oculus Rift

XR Interaction Debugger			
Input Devices Interactables Interactors			
Devices			
Name	Role	Type	Value
▼ Oculus Touch Controller - Left	LeftHanded		
Primary2DAxis		UnityEngine.Vector2	(0.0, 0.0)
Trigger		System.Single	7,152557E-07
Grip		System.Single	7,152557E-07
IndexTouch		System.Single	0
ThumbTouch		System.Single	0
PrimaryButton		System.Boolean	False
SecondaryButton		System.Boolean	False
GripButton		System.Boolean	False
MenuButton		System.Boolean	False
Primary2DAxisClick		System.Boolean	False
PrimaryTouch		System.Boolean	False
SecondaryTouch		System.Boolean	False
TriggerButton		System.Boolean	False
Primary2DAxisTouch		System.Boolean	False
Thumbrest		System.Boolean	False
TrackingState		System.UInt32	42
IsTracked		System.Boolean	False
DevicePosition		UnityEngine.Vector3	(0.0, 0.0, 0.0)
DeviceRotation		UnityEngine.Quaternion	(-0.6, -0.1, 0.2, -0.7)
DeviceVelocity		UnityEngine.Vector3	(0.0, 0.0, 0.0)
DeviceAngularVelocity		UnityEngine.Vector3	(0.0, 0.0, 0.0)
DeviceAcceleration		UnityEngine.Vector3	(0.0, 0.0, 0.0)
DeviceAngularAcceleration		UnityEngine.Vector3	(0.3, 0.4, -0.3)

Рисунок 2.66 – Дані одного з контролерів Oculus Touch Controller

З шолому віртуальної реальності Oculus Rift в середовищі розробки Unity можна отримати такі дані:

- TrackingState – представляє значення, які відстежуються для цього пристрою;
- IsTracked – інформує розробника, чи відстежується пристрій на даний момент;
- DevicePosition – положення пристрою в просторі;
- DeviceRotation – обертання пристрою в просторі;
- DeviceVelocity – швидкість руху пристрою;
- DeviceAngularVelocity – кутова швидкість пристрою в форматі кутів Ейлера;
- DeviceAcceleration – прискорення пристрою;
- DeviceAngularAcceleration – кутове прискорення пристрою, в форматі кутів Ейлера;

- LeftEyePosition – положення точки лівого ока користувача в просторі;
- LeftEyeRotation – обертання точки лівого ока користувача в просторі;
- LeftEyeVelocity – швидкість руху точки лівого ока користувача;
- LeftEyeAngularVelocity – кутова швидкість точки лівого ока користувача в форматі кутів Ейлера;
- LeftEyeAcceleration – прискорення точки лівого ока користувача;
- LeftEyeAngularAcceleration – кутове прискорення точки лівого ока користувача, в форматі кутів Ейлера;
- RightEyePosition – положення точки правого ока користувача в просторі;
- RightEyeRotation – обертання точки правого ока користувача в просторі;
- RightEyeVelocity – швидкість руху точки правого ока користувача;
- RightEyeAngularVelocity – кутова швидкість точки правого ока користувача в форматі кутів Ейлера;
- RightEyeAcceleration – прискорення точки правого ока користувача;
- RightEyeAngularAcceleration – кутове прискорення точки правого ока користувача, в форматі кутів Ейлера;
- CenterEyePosition – положення точки посередині між зіницями очей користувача в просторі;
- CenterEyeRotation – обертання точки посередині між зіницями очей користувача в просторі;
- CenterEyeVelocity – швидкість руху точки посередині між зіницями очей користувача;
- CenterEyeAngularVelocity – кутова швидкість точки посередині між зіницями очей користувача в форматі кутів Ейлера;
- CenterEyeAcceleration – прискорення точки посередині між зіницями очей користувача;
- CenterEyeAngularAcceleration – кутове прискорення точки посередині між зіницами очей користувача, в форматі кутів Ейлера.

З контролерів Oculus Touch Controller в середовищі розробки Unity можна отримати такі дані:

- TrackingState – представляє значення, які відстежуються для цього пристрою;
- IsTracked – інформація про розробника, чи відстежується пристрій на даний момент;
- DevicePosition – положення пристрою в просторі;
- DeviceRotation – обертання пристрою в просторі;
- DeviceVelocity – швидкість руху пристрою;
- DeviceAngularVelocity – кутова швидкість пристрою в форматі кутів Ейлера;
 - DeviceAcceleration – прискорення пристрою;
 - DeviceAngularAcceleration – кутове прискорення пристрою, в форматі кутів Ейлера;
 - Primary2DAxis – положення основного джойстика на пристрії;
 - Trigger – положення управління, схожого на спусковий гачок, натискається вказівним пальцем;
 - Grip – положення управління, схожого на спусковий гачок, натискається середнім пальцем;
 - IndexTouch – дотик вказівним пальцем до кнопки Trigger;
 - ThumbTouch – дотик пальцем до кнопки Grip;
 - PrimaryButton – натискання на основну кнопку контролера (кнопка контролера [X/A]);
 - SecondaryButton – натискання на додаткову основну кнопку контролера (кнопка контролера [Y/B]);
 - GripButton – повне натискання на кнопку Grip;
 - MenuButton – натискання на кнопку Start на контролері;
 - Primary2DAxisClick – натискання на кнопку джойстика;
 - PrimaryTouch – дотик до основної кнопки контролера (кнопка контролера [X/A]);

- SecondaryTouch – дотик до додаткової основної кнопки контролера (кнопка контролера [Y/B]);

- TriggerButton – повне натискання на кнопку Trigger;
- Primary2DAxisTouch – дотик до кнопки джойстика;
- Thumbrest – дотик великим пальцем до панелі контролера.

Позиціонування шолому в просторі відбувається відносно центральної точки ігрової зони, отриманої під час налаштування шолому в застосунку Oculus.

За висотою нульова точка розташовується на рівні полу (теж визначається під час налаштування ігрової зони в застосунку Oculus).

Положення контролерів в просторі відповідно до знайденої документації розраховується залежно від положення шолому в просторі (тобто координатою 0 для контролера є точка знаходження шолому в просторі).

Приклад скрипту для зчитування набору даних з пристрій Oculus Rift та запису у локальний CSV файл та скріншотів ігрового вікна в PNG файли наведено в пункті 3.

На рисунку 2.67 наведено зображення написаного проекту.

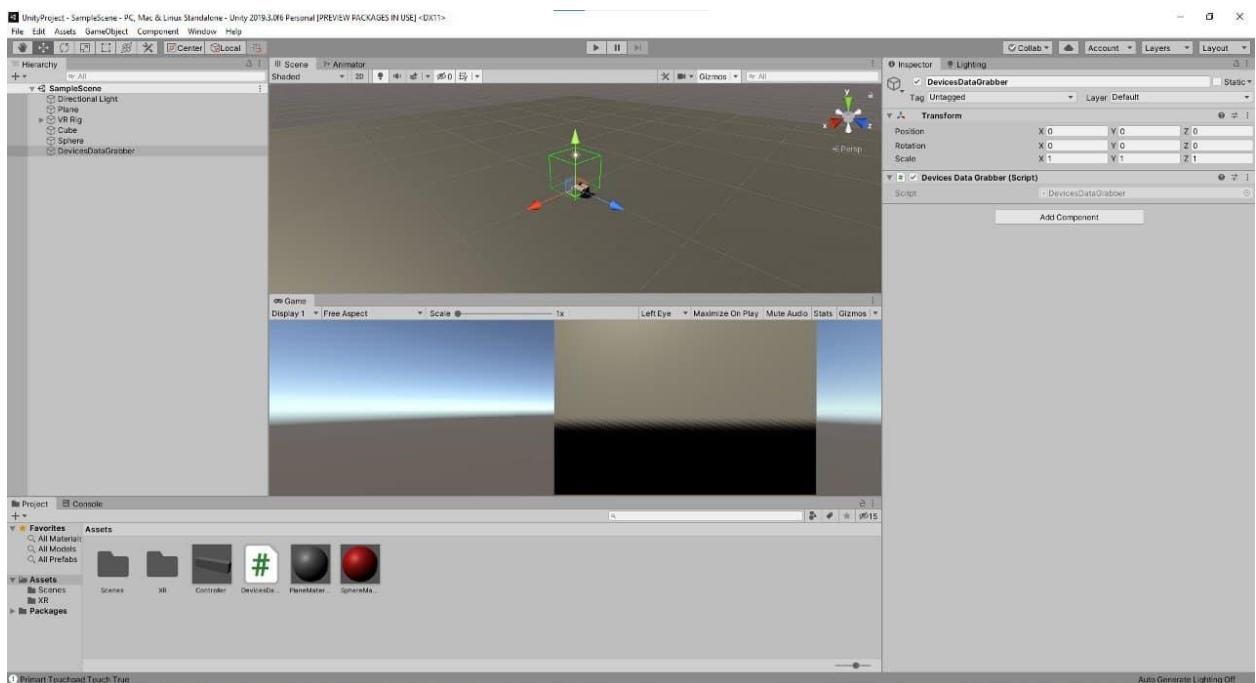


Рисунок 2.67 – Зображення написаного проекту

Приклад зчитаних наборів даних з пристрой OculusRift та записаних у локальний CSV файл та скріншотів ігрового вікна в PNG файли наведено на рисунку 2.68.

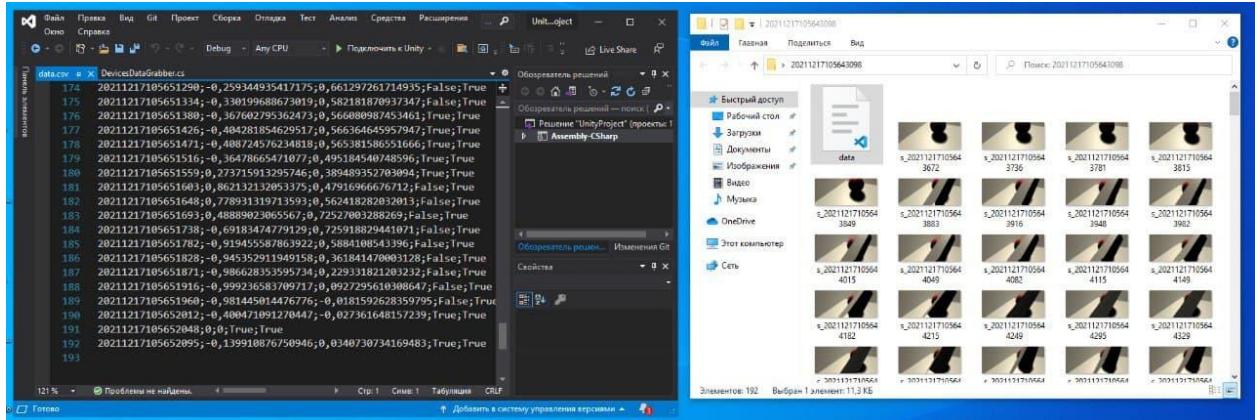


Рисунок 2.68 – Збережені дані пристрой та скріншоти в пам’яті комп’ютера

3 ФАЙЛІ СКРИПТІВ

3.1 Файл DeviceDataGrabber

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.XR;

public class DevicesDataGrabber : MonoBehaviour
{
    DateTime lastUpdateTime;

    private InputDevice LeftController;
    //private InputDevice RightController;

    private string directoryPath = "C:\\\\Users\\\\Admin\\\\Desktop\\\\";
    private string screenshotPath;
    private string csvExportFilePath;

    private Dictionary<string, string> devicesData = new Dictionary<string,
    string>();

    // Start is called before the first frame update
    void Start()
    {
        List<InputDevice> devices = new List<InputDevice>();

        InputDeviceCharacteristics leftControllerCharacteristics =
        InputDeviceCharacteristics.Left | InputDeviceCharacteristics.Controller;
        InputDevices.GetDevicesWithCharacteristics(leftControllerCharacteristics,
        devices);

        foreach (var item in devices)
        {
            Debug.Log(item.name + item.characteristics);
        }

        if (devices.Count > 0)
        {
            LeftController = devices[0];
        }

        devicesData["date"] = "";
        devicesData["left_controller_primary2daxis_x"] = "";
        devicesData["left_controller_primary2daxis_y"] = "";
        devicesData["left_controller_primary2daxistouch"] = "";
        devicesData["left_controller_primary2daxisclick"] = "";

        lastUpdateTime = DateTime.Now;

        directoryPath += lastUpdateTime.ToString("yyyyMMddHHmmssfff");
        csvExportFilePath = directoryPath + "\\\" + "data.csv";
        screenshotPath = directoryPath + "\\\" + "s_";

        if (!Directory.Exists(directoryPath))
        {
    
```

```

        Directory.CreateDirectory(directoryPath);
    }

    using (StreamWriter sw = File.CreateText(csvExportFilePath))
    {
        string headerString = "";
        foreach (var str in devicesData.Keys)
        {
            headerString += str + ";";
        }
        headerString = headerString.TrimEnd(';');

        sw.WriteLine(headerString);
    }
}

// Update is called once per frame
void Update()
{
    DateTime updateTime = DateTime.Now;

    if ((updateTime - lastUpdateTime).TotalMilliseconds >= 100)
    {
        string currentTime = updateTime.ToString("yyyyMMddHHmmssfff");

        ScreenCapture.CaptureScreenshot(screenshotPath + currentTime + ".png");
        Debug.Log("ScreenshotCaptured");

        LeftController.TryGetFeatureValue(CommonUsages.primary2DAxis, out Vector2 primary2dAxisValue);
        Debug.Log("Primart Touchpad " + primary2dAxisValue);

        LeftController.TryGetFeatureValue(CommonUsages.primary2DAxisClick, out bool primary2daxisclick);
        Debug.Log("Primart Touchpad Click " + primary2daxisclick);

        LeftController.TryGetFeatureValue(CommonUsages.primary2DAxisTouch, out bool primary2daxistouch);
        Debug.Log("Primart Touchpad Touch " + primary2daxistouch);

        ControllerData leftControllerData = new ControllerData();
        leftControllerData.date = updateTime;
        leftControllerData.primary2daxis_x = primary2dAxisValue.x;
        leftControllerData.primary2daxis_y = primary2dAxisValue.y;
        leftControllerData.primary2daxisclick = primary2daxisclick;
        leftControllerData.primary2daxistouch = primary2daxistouch;

        devicesData["date"] = updateTime.ToString("yyyyMMddHHmmssfff");
        devicesData["left_controller_primary2daxis_x"] =
leftControllerData.primary2daxis_x.ToString();
        devicesData["left_controller_primary2daxis_y"] =
leftControllerData.primary2daxis_y.ToString();
        devicesData["left_controller_primary2daxistouch"] =
leftControllerData.primary2daxisclick.ToString();
        devicesData["left_controller_primary2daxisclick"] =
leftControllerData.primary2daxistouch.ToString();

        using (StreamWriter sw = File.AppendText(csvExportFilePath))
        {
            string dataString = "";
            foreach (var key in devicesData.Keys)
            {

```

```
        dataString += devicesData[key] + ";"  
    }  
    dataString = dataString.TrimEnd(';');  
  
    sw.WriteLine(dataString);  
}  
}  
}  
  
[Serializable]  
public class ControllerData  
{  
    [SerializeField]  
    public DateTime date;  
    [SerializeField]  
    public double primary2daxis_x;  
    [SerializeField]  
    public double primary2daxis_y;  
    [SerializeField]  
    public bool primary2daxisclick;  
    [SerializeField]  
    public bool primary2daxistouch;  
}  
}
```