ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
АВТОМАТИЗОВАНИХ СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка механізмів віртуальної реальності для використання їх у навчальному процесі з предмету математики початкових класів»

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Ірина БУЛАЦАН
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав:
здобувач вищої освіти Ірина БУЛАЦАН
група ІСДМ-62

Керівник: Калинюк А.М.
науковий ступінь, вчене звання к.ф.-м.н., доцент

Рецензент:
науковий ступінь, вчене звання _____
Ім'я, ПРІЗВИЩЕ

Київ 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем

Ступінь вищої освіти Магістр

Спеціальність Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІПЗАС

_____ Каміла СТОРЧАК
«_____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Булацан Ірині Романівні
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Розробка механізмів віртуальної реальності для використання їх у навчальному процесі з предмету математики початкових класів

керівник кваліфікаційної роботи Алла КАЛИНЮК, к.ф.-м.н., доцент
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» 10.2023р. №145

2. Строк подання кваліфікаційної роботи «29» грудня 2023р.

3. Вихідні дані до кваліфікаційної роботи: Unity, Visual Studio 2019, науково-технічна література з питань, пов'язаних з темою работи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Дослідити ключові елементи та технології роботи
Проаналізувати гарнітуру зручну для використання
Розробка середовища та механізмів взаємодії
Висновки

5. Перелік графічного матеріалу: *презентація*

1. Ключові технології, що лежать в основі віртуальної реальності
2. Порівняльний аналіз гарнітури для VR
3. Розроблений проект в Unity

6. Дата видачі завдання«19» жовтня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	19.10-05.11.23	
2	Вивчення матеріалів для подальшої взаємодії з ними	05.11-12.11.23	
3	Аналіз існуючих гарнітур та впроваджених проектів	13.11-19.11.23	
4	Вибір технологій та середовища проектування	20.11-25.11.23	
5	Розробка\тестування програми	27.11-03.12.23	
6	Вступ, висновки, реферат	04.12-10.12.23	
7	Розробка обов‘язкових демонстраційних матеріалів	11.12-20.12.23	
8	Попередній захист роботи	21.12-29.12.23	

Здобувач вищої освіти _____ Ірина БУЛАЦАН
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Алла КАЛИНЮК
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра 83 стор., 3 табл., 20 рис., 3 дод., 36 джерел.

Мета роботи – розробка механізмів віртуальної реальності з метою створення інтерактивної взаємодії користувача з об’єктами віртуальної реальності

Об’єкт дослідження – розробка механізмів віртуальної реальності

Предмет дослідження – технології та механізми віртуальної реальності в середовищі Unity

Короткий зміст роботи: У роботі проведено дослідження існуючих технологій віртуальної реальності. Проаналізовано основні методи взаємодії користувача з об’єктами в іммерсивному середовищі. Визначено оптимальна для використання гарнітура та платформа для розробки. Проведено роботу над створенням віртуального середовища та механізмів взаємодії в ньому за допомогою джойстика або з використанням рук для забезпечення реалістичного досвіду.

КЛЮЧОВІ СЛОВА:ТЕХНОЛОГІЇ ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ, ІММЕРСИВНІСТЬ, СЕНСОРИ, ВІДСЛІДКОВУВАННЯ РУХІВ, ГАРНІТУРА, ПАРАМЕТРИ.

ABSTRACT

Text part of the master's qualification work: 83 pages, 20 pictures, 3 table, 3 attachments, 36 sources.

The purpose of the work is to develop virtual reality mechanisms in order to create interactive user interaction with virtual reality objects.

Object of research - development of virtual reality mechanisms.

Subject of research - technologies and mechanisms of virtual reality in the Unity environment.

Summary of the work: The research of existing technologies of virtual reality is carried out in the work. The main methods of user interaction with objects in an immersive environment are analyzed. The optimal headset for use and the development platform have been determined. Work has been carried out on the creation of a virtual environment and interaction mechanisms in it with the help of a joystick or using hands to provide a realistic experience.

KEYWORDS: VIRTUAL REALITY TECHNOLOGIES, IMMERSIVITY, SENSORS, MOTION TRACKING, HEADSET, PARAMETERS.

ЗМІСТ

ВСТУП

Сучасний світ неперервно вдосконалює технологічні засоби, розширюючи їхні можливості й застосування в освіті. Однією з інновацій є використання віртуальної реальності (VR) в навчанні. Віртуальна реальність - це технологічне досягнення, яке створює імерсивне середовище, де користувачі можуть взаємодіяти з віртуальним світом через спеціальне обладнання, таке як VR-окуляри й контролери[1]. Імерсивні технології - технології повного або часткового занурення у віртуальний світ або різні види змішання реальної і віртуальної реальності[2].

Актуальність данної теми підтверджується тим, що на сьогоднішній день вже технології віртуальної реальності починають застосовуватись, тому покращення взаємодії користувача з об’єктами є важливим для розвитку данної області технологій.

В освітній сфері механізми взаємодії створенні з урахуванням використання джойстика, в данній роботі наукова новизна полягає в розробці механізмів взаємодії за допомогою як джойстика так і рук, для можливості отримувати більш реалістичний досвід або для дітей в яких моторика рук ще не готова до використання джойстика.

З технічної точки зору, розробка механізмів віртуальної реальності для навчання математики охоплює створення віртуальних об'єктів, сценаріїв та інтерактивних завдань, які допомагають учням легше розуміти абстрактні математичні поняття. Важливо також розробити ефективні методи навчання в цьому інтерактивному середовищі.

Ця магістерська робота спрямована на дослідження, проектування та створення технічних аспектів віртуальної реальності для математичного навчання на початковому рівні. Вона розглядає апаратне та програмне забезпечення, яке необхідне для реалізації VR-середовища в освіті, а також розробляє методику використання цих інструментів для досягнення кращих навчальних результатів та стимулювання інтересу учнів до математики[3].

РОЗДІЛ 1 КЛЮЧОВІ ТЕХНОЛОГІЇ, ЩО ЛЕЖАТЬ В ОСНОВІ ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ

Віртуальна реальність є інтерактивною технологічною системою, яка створює враження присутності користувача у віртуальному середовищі. Віртуальна реальність створюється завдяки комбінації технологій, які використовуються для візуалізації та забезпечення взаємодії з віртуальним середовищем. Ці середовища часто зображують тривимірний простір, який може бути як реалістичним, так і уявним. По розмірах він може бути макроскопічним або мікроскопічним. Також базується таке середовище на подібних до реальних фізичних законах динаміки або на уявній динаміці. Безліч сценаріїв, для зображення яких можна використовувати VR, робить його широко застосовним до багатьох сфер освіти. Ключовою особливістю віртуальної реальності є те, що вона дозволяє мультисенсорну взаємодію з візуалізованим простором.

Віртуальна реальність ґрунтується на кількох ключових технологіях, які спільно створюють імерсивний віртуальний досвід.

1.1 Гарнітура відображання для віртуальної реальності

Основним компонентом віртуальної реальності є постійне 3D-візуальне представлення середовища, яке передає відчуття глибини. Щоб створити цю глибину, апаратне забезпечення віртуальної реальності використовує 3D-дисплей високої роздільної здатності який знаходиться в окулярах або шоломі для віртуальної реальності.

Щоб створити ілюзію глибини, потрібно створити окреме зображення для кожного ока, одне трохи зміщене відносно іншого, щоб імітувати паралакс. Приклад зміщеного зображення показано на рис. 1.1

Рисунок 1.1 – Два дисплеї, в яких зображення трохи зміщене одне відносно іншого

Паралакс - це явище, яке виникає при спостереженні об'єктів з різних точок огляду і полягає в тому, що об'єкт здається змінювати своє положення відносно інших об'єктів або фону. Ця зміна положення об'єкта спричиняється різницею в кутах огляду об'єкта від різних спостерігачів або точок спостереження. Якщо закрити одне око і подивитесь на об'єкт перед собою, а потім закриєте інше око і подивитесь на той же об'єкт, то можна помітити, що об'єкт видно з іншого кута[4]. Це приклад паралакса який використовується для створення віртуальної реальності (через наші очі трохи віддалені один від одного). Щоб створити хорошу ілюзію, також потрібно спотворити зображення, щоб краще імітувати сферичну форму ока, використовуючи техніку, відому як бочкоподібна дисторсія. Бочкоподібна дисторсія (спотворення) - явище, коли точки в полі зору виглядають візуально ближчими чим ближче вони до центру і навпаки, збільшення зображення зменшується із збільшенням відстані від оптичної осі (рис. 1.2). Візуальний ефект ніби зображення нанесене довкола бочки, через це і така назва. Лінзи риб'яче око, які мають напівсферичне поле зору, використовують цей тип дисторсії як спосіб відображення нескінченно широкої площини об'єкта у скінченну площу зображення[5].

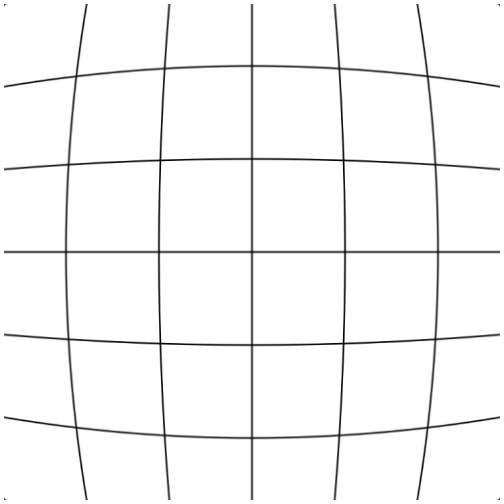


Рисунок 1.2 - Бочкоподібна дисторсія

Гарнітуру можна розділити на три типи:

а) Смартфон, який обробляє та виводить зображення. Вони високопродуктивні та здатні самостійно обробляти тривимірні зображення. Дисплеї смартфонів мають досить високу роздільну здатність. Практично кожен смартфон має датчики, що дозволяють визначати положення пристрою.

б) Зовнішній пристрій, який обробляє зображення, такий як комп’ютер або ігрова консоль, який передає інформацію на гарнітуру на голові користувача. Сама ж ця гарнітура відповідає за передачу положення завдяки датчикам положення.

в) Автономно працюючі окуляри віртуальної реальності. Шоломи є основним компонентом VR з повним зануренням, оскільки не тільки забезпечують об'ємне зображення та стереозвучання, а й частково ізолюють користувача від навколишньої реальності[6].

1.2 Обладнання для відстеження руху

Відстеження рухів голови та оновлення візуальної сцени в реальному часі змушує мозок повірити, що він знаходиться в іншому місці. Це імітує те, що

відбувається, коли людина дивиться навколо в реальному світі. Відстеження рухів голови відбувається за допомогою високошвидкісного блоку вимірювання інерції. Блок вимірювання інерції - це електронний пристрій або система датчиків, яка використовується для вимірювання і повідомлення про швидкість, орієнтацію та гравітаційні сили об'єкта[7].

Зазвичай пристрій складається з кількох типів датчиків:

Акселерометр - вимірює лінійне прискорення вздовж однієї або кількох осей, що може бути використано для визначення швидкості та орієнтації пристрою.

Гіроскоп - вимірює швидкість обертання або кутову швидкість навколо однієї або декількох осей. Гіроскопи допомагають визначити зміни в орієнтації.

Магнітометр - вимірює силу та напрямок магнітного поля в навколишньому середовищі. Цей датчик часто використовується разом з акселерометрами та гіроскопами, щоб надати більш точний показник орієнтації.

Блок вимірювання інерції поєднує дані з цих датчиків для оцінки положення, швидкості та орієнтації об'єкта в реальному часі.

1.3 Пристрої управління

Щоб створити переконливе відчуття занурення, окуляри повністю закривають очі користувача, відрізаючи його від зовнішнього світу. Це створює ситуацію коли користувачу доводиться взаємодіяти з віртуальним світом не бачачи своєї миші чи клавіатури під час їх використання. З цієї причини існує багато типів пристроїв для взаємодії, щоб забезпечити більше відчуття занурення:

а) Датчики відстеження руху. Датчики відстеження руху призначені для визначення руху користувача або об'єктів у реальному світі та передачі цієї інформації у VR-систему, щоб забезпечити відповідну реакцію або візуальне сприйняття. Такі системи можуть використовувати бездротові трекери й базову станцію для створення тривимірного простору, в якому користувач може вільно рухатися та взаємодіяти з віртуальними об'єктами. Або використовує бездротові контролери, які користувач тримає в руках, і

дозволяє взаємодіяти з об'єктами в віртуальному світі.

б) Контролери. Контролери в системах віртуальної реальності є пристроями, які користувач може тримати й використовувати для взаємодії з віртуальним світом. Ці контролери зазвичай оснащені різними сенсорами та кнопками, які дозволяють користувачеві керувати об'єктами, переміщатися в віртуальному середовищі, взаємодіяти з об'єктами та виконувати інші дії.

в) Рукавичка. Це пристрій взаємодії між людиною та комп’ютером, який носять як рукавичку. Для запису фізичних даних, наприклад згинання пальців, використовуються різні сенсорні технології. Часто принцип роботи наступний, пристрій відстеження руху, наприклад магнітний або інерційний пристрій відстеження, приєднується для фіксації глобальних даних про положення та обертання рукавички. Потім ці рухи інтерпретуються програмним забезпеченням, яке супроводжує рукавичку, тому будь-який рух може означати будь-яку кількість речей. Тоді жести можна розділити на корисну інформацію, наприклад для розпізнавання мови жестів або інших символічних функцій. Рукавички високого класу також можуть забезпечувати тактильний зворотний зв’язок, який є імітацією відчуття дотику.

г) Системи трекінгу рук. Системи трекінгу рук в системах віртуальної реальності дозволяють відстежувати рухи та позицію рук користувача в реальному часі та відтворювати цю інформацію в віртуальному світі. Це дозволяє користувачам взаємодіяти з віртуальними об'єктами та середовищем за допомогою їх реальних рук. Ось декілька типів систем трекінгу рук, які використовуються: оптичні камери та сенсори, що встановлюються на гарнітурі або в кімнаті, які реєструють рухи рук користувача на основі зображень і відтворюють їх у віртуальному середовищі; інфрачервоні датчики для відстеження рухів рук, які виробляють інфрачервоні сигнали й вимірюють їх відбиття від рук, що дозволяє точно визначати позицію рук у просторі; ультразвукові системи

використовують високочастотні звукові хвилі для вимірювання відстані до рук користувача, можуть бути встановлені на гарнітурі або в кімнаті й відстежувати рухи рук в реальному часі; магнітні датчики, які реєструють зміни в магнітному полі, коли користувач рухає руками (ця технологія може бути особливо корисною для відстеження рухів під водою або через інші перешкоди); акселерометри та гіроскопи, можуть використовуватися для відстеження рухів рук користувача, вимірюючи прискорення і кутову швидкість рухів[8].

Візуальний приклад геймпада для віртуальної реальності відображено нижче (рис.1.3).

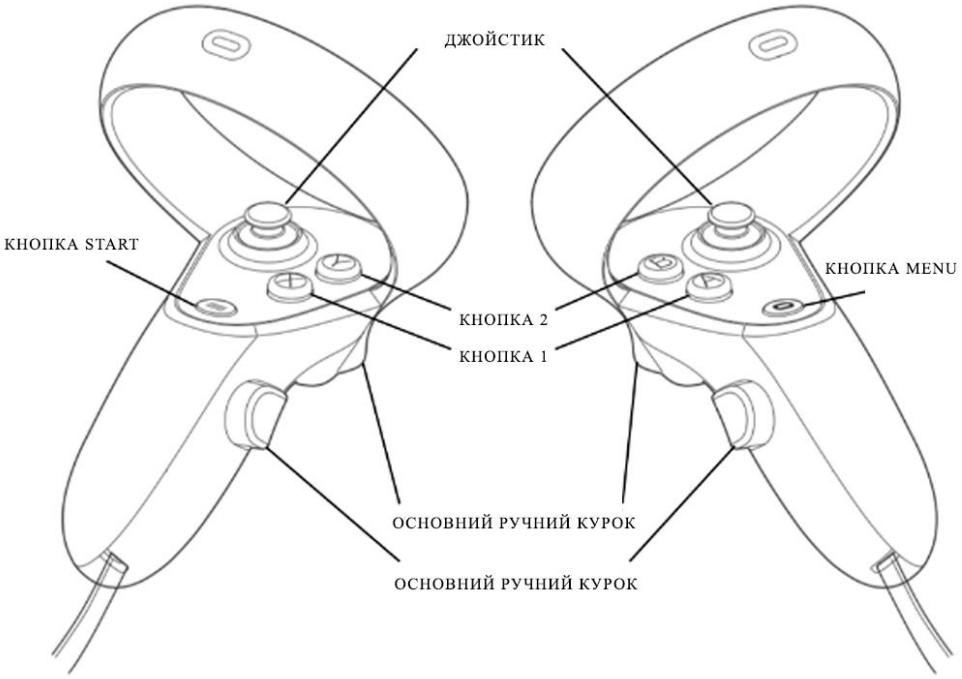


Рисунок 1.3 - Приклад геймпада

1.4 Технології відтворення звуку

Користувачі віртуальної реальності можуть чути звук завдяки різним пристроям та технологіям, які створюють імерсивний аудіо досвід. Ось деякі з найпоширеніших пристроїв для відтворення звуку:

а) Звукові бандажі. Деякі VR-гарнітури мають вбудовані звукові бандажі або паси, які розташовані навколо голови користувача та надають аудіофайли

через динаміки, які розташовані біля вух. Прикладом є HTC Vive Deluxe Audio Strap, який є доповненням до HTC Vive і надає звук та комфорт за рахунок вбудованих навушників.

б) Звукові системи віртуальної реальності. Деякі платформи містять спеціально розроблені звукові системи, які можуть виробляти просторовий звук із відтворенням напрямленості. Це може бути звукова панель, вбудована в гарнітуру або інший аудіокомпонент. Прикладом є Oculus Quest 2, який має вбудовані динаміки для просторового звуку.

в) Звукові мережі та процесори. Деякі VR-системи використовують звукові мережі та процесори для обробки та відтворення аудіофайлів, створюючи просторовий звук та оброблюючи звукові ефекти. Прикладом є Oculus Rift S, який використовує технологію Oculus Audio SDK для створення просторового звуку із відтворенням враховуючи напрямок.

г) Звукові акустичні панелі. В деяких системах можуть використовуватися акустичні панелі або динаміки, розташовані в околицях користувача, щоб створити просторовий звук і надати імерсивний аудіальний досвід. Прикладом є платформа Teslasuit, яка використовує акустичні панелі, розташовані на спеціальному комбінезоні, щоб створити імерсивний аудіо досвід.

Звук відображається в віртуальній реальності за допомогою різних технологій та методів.

Просторовий звук (також відомий як 3D-звук або звукова просторова імітація) є ключовим аспектом звуку в віртуальній реальності. Ця технологія створює враження, що звукові джерела розташовані у просторі, навколо користувача. Вона використовує спеціальні алгоритми та апаратне забезпечення, такі як просторові навушники, для відтворення звуків із різних напрямків. Прикладом є навушники Logitech G333 VR Gaming Earphones, надають можливість відчувати, що звукові джерела навколо вас розташовані у просторі, що додає імерсивність до віртуального досвіду[9].

Просторове аудіо дозволяє точно визначити положення звукових джерел у

віртуальному середовищі та налаштувати їх відповідно до рухів користувача. Зазвичай це досягається за допомогою спеціальних мікрофонів та алгоритмів обробки сигналу. Прикладом є програмне рішення Dolby Atmos for VR, яке може використовувати вбудовані мікрофони в гарнітурах для точного визначення положення звукових джерел у просторі. Таким чином, звук може змінювати своє положення відповідно до рухів користувача, створюючи просторовий звуковий ефект[10].

Відстеження голови - це процес визначення положення та руху голови користувача в реальному часі. Ця інформація використовується для налаштування звуку відповідно до положення голови, що дозволяє користувачам оточувати звукові джерела, обертаючи голову. Прикладом є HTC Vive Pro зі станціями відстеження Lighthouse, користувач може відчувати, що звукові джерела рухаються відповідно до руху його голови. Це створює ілюзію того, що звук навколо користувача переміщується разом з його рухами[11].

Технології акустичного моделювання використовуються для створення реалістичних звукових ефектів у віртуальних світах. Вони враховують акустичні характеристики об'єктів та середовища, що оточує користувача, та відтворюють звуки відповідно до цих характеристик. Прикладом може бути програмне забезпечення Wwise або FMOD, може використовуватися в іграх та VR-додатках для створення акустичних ефектів, які відтворюються відповідно до характеристик конкретного віртуального середовища.

Технологія підсилення звуку використовується для створення масштабних аудіо ефектів. Вона може підсилювати звуки залежно від контексту або подій у віртуальному середовищі, щоб підсилити імерсивний ефект. Прикладом є Sony PlayStation VR з технологією 3D Audio, яка підсилює певні звуки залежно від їхнього положення у віртуальному просторі, роблячи їх більш імерсивними[12].

РОЗДІЛ 2 ПЛАТФОРМИ ДЛЯ РОЗРОБКИ МЕХАНІЗМІВ ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ

Розробка механізмів віртуальної реальності вимагає спеціальних платформ та інструментів. Такі платформи часто мають спеціалізовані інструменти та рішення, які допомагають створювати рухомі об'єкти, фізичні ефекти та інтерактивні механізми з врахуванням особливостей віртуальної реальності. Велика частина роботи націлена на оптимізацію для конкретних VR-платформ, таких як Oculus, HTC Vive, PlayStation VR і інші. Використання спеціалізованих платформ дозволяє забезпечити найкращу продуктивність та сумісність із конкретними пристроями. Платформ для розробки мають вбудовані рішення для моделювання фізики та створення інтерактивних механізмів. Це дозволяє легко додавати фізичні ефекти та ігрову логіку проектів. Багато платформ інтегруються з різними сторонніми інструментами й плагінами, що дозволяє розширити функціональність і створити складніші механізми та інтерактивні об'єкти. Популярні платформи для розробки мають активну спільноту розробників та доступ до широкого спектра документації, уроків і підтримки.

2.1 Unity3D

Unity - це кросплатформний ігровий движок, розроблений компанією Unity Technologies, вперше анонсований і випущений у червні 2005 року. Її можна використовувати для створення тривимірних і двовимірних ігор, а також для інтерактивного моделювання та інших вражень.

Вона підтримує багато різних платформ, включаючи Windows, macOS, iOS, Android, Linux, PlayStation, Xbox, Nintendo Switch, веб (WebGL), а також платформи віртуальної реальності, такі як Oculus Rift, HTC Vive, PlayStation VR і інші. Це дозволяє створювати проекти, які можуть бути розгорнуті на різних пристроях і операційних системах[13].

C# є найбільш популярною мовою програмування для Unity3D. Вона є офіційною мовою програмування Unity та надає доступ до всіх функцій та можливостей платформи, більше можливостей для роботи зі сторонніми бібліотеками та плагінами, що може бути корисним для розширення функціональності. C# використовується для створення скриптів та програм, які контролюють поведінку об'єктів у грі, створення ігрової логіки та багато іншого. C# має велику спільноту розробників, тому легко знайти багато ресурсів, навчальних матеріалів та документації для навчання та розв'язання проблем у C#. Відомі онлайн-курси та книги також часто орієнтовані на C#. JavaScript ще одна мова програмування, яка раніше підтримувалася в Unity, але в більшості випадків зараз рекомендується використовувати C#. UnityScript подібна до JavaScript, але має свої особливості. Попри те, що JavaScript все ще можна використовувати, рекомендується переходити на C#, оскільки ця мова має більшу підтримку та переваги. Наприклад, C# зазвичай є швидшою мовою порівняно з JavaScript. Також JavaScript вже не активно оновлюється та підтримується в нових версіях. Ще однією мовою використання може бути Boo. Він має синтаксис, який дуже схожий на Python. Якщо розробник знайомий з Python, то буде легше вивчати та працювати з Boo. Unity дозволяє використовувати Boo і C# разом в одному проекті, використовувати C# для складних алгоритмів та взаємодії зі системними компонентами Unity, а Boo - для швидкого прототипування іншої логіки. Boo має систему типізації, яка може бути корисною для зменшення помилок та поліпшення проектування коду та вбудовану підтримку мережевого програмування.

Unity має потужний тривимірний і двовимірний графічний двигун, який дозволяє створювати реалістичні об'єкти, використовувати різноманітні ефекти, текстури та освітлення. Існує потужна система шейдерів і матеріалів, яка дозволяє створювати реалістичні текстури, освітлення та спеціальні ефекти. Можна створювати власні шейдери або використовувати готові. Графічний двигун Unity дозволяє створювати різні типи анімації, включаючи скелетну анімацію, інтерполяцію між кадрами, анімаційні контролери та інші методи. Це особливо

важливо для створення рухомих персонажів та об'єктів. Unity підтримує різні техніки візуалізації, включаючи візуалізацію за допомогою лінійного запропонування і візуалізації з використанням буфера глибин. Це дозволяє отримувати високоякісні графічні зображення з реалістичним освітленням та тінями.

Є вбудована система фізики, яка дозволяє створювати реалістичну фізичну поведінку об'єктів. Що охоплює рух, зіткнення, гравітацію та інші фізичні закони. Підтримка різних видів зіткнень, включаючи зіткнення між об'єктами, тригери, тобто зони, які спрацьовують при входженні об'єкта в них, та рей касти, тобто промені, які використовуються для виявлення перешкод. Можна призначати фізичні властивості матеріалу об'єктам, що визначатимуть їхню поведінку. Наприклад, можна встановити коефіцієнт тертя, масу та інші фізичні властивості для об'єктів. Створювати динамічні об'єкти, які реагують на сили й взаємодію з іншими об'єктами.

Unity підтримує скелетну анімацію, де об'єкти анімуються за допомогою скелета або кісток. Можна створювати рухи персонажів і контролювати їх за допомогою анімаційних контролерів. Надаються інструменти для створення анімаційних контролерів, які дозволяють переходити між анімаціями, відтворювати їх у відповідь на події та управляти рухом об'єктів. Створювати анімаційні кліпи для різних дій і рухів, і потім об'єднувати їх в анімаційні контролери для реалізації складних анімаційних сценаріїв. Також є можливість створювати анімацію шляхом програмування, де керування анімацією об'єктів відбувається за допомогою скриптів. Підтримується анімація частинок, що дозволяє створювати ефекти, такі як дим, вогонь, блискавки тощо.

Має вбудовану інтегровану розробницьку середу, яка називається Unity Editor. Інтегроване розробницьке середовище (IDE) в Unity3D - це інструмент, який надає розробникам зручну робочу область для створення ігор та інтерактивних додатків. Надається візуальний інтерфейс для створення та редагування сцен. Розробники можуть перетягувати та розміщувати об'єкти на сцені, налаштовувати їх властивості та розміщення. Панель інспектор об'єктів де можна переглядати та

змінювати властивості обраного об'єкта, призначати компоненти, шейдери, матеріали та інші параметри. Unity використовує систему компонентів, де кожен об'єкт може мати різні компоненти, які визначають його функціональність. Наприклад, персонаж може мати компонент для управління фізикою, компонент анімації та компонент для здоров'я. Середовище дозволяє створювати скрипти та програми за допомогою візуальних інструментів, таких як графи поведінки та візуальні скрипти. Це полегшує роботу для розробників, які не володіють програмуванням. Unity Editor дозволяє переглядати та тестувати проекти на різних платформах безпосередньо в середовищі розробки. Можна перемикати платформи для перегляду, як проект працює на різних пристроях. Допомагає керувати ресурсами, такими як моделі, текстури, анімації та інші активи, включаючи можливість імпорту та оптимізації цих ресурсів.

2.1.1 Огляд основних компонентів для реалізації для реалізації VR програми в Unity

Щоб створювати програми VR, Unity має плагін під назвою XR Plug-in Management. Плагін в Unity - це набір коду, створеного поза Unity, який створює функціональні можливості в Unity. В Unity можна використовувати два типи плагінів: керовані плагіни (керовані збірки .NET, створені за допомогою таких інструментів, як Visual Studio) і рідні плагіни (бібліотеки власного коду для певної платформи). Unity підтримує такі плагіни:

а) Oculus для Oculus Rift, Meta Quest 2 і Quest Pro.

б) OpenXR для будь-якого пристрою з середовищем виконання OpenXR, включаючи гарнітури Meta, гарнітури Vive, Valve SteamVR, HoloLens, Windows Mixed Reality та інші.

в) PlayStation VR (доступна для зареєстрованих розробників PlayStation) для пристроїв Sony PS VR і PS VR2.

г) Макет HMD для імітації гарнітури VR у перегляді режиму відтворення Unity Editor[14].

Також Unity має XR Interaction Toolkit - це високорівнева компонентна

система взаємодії для створення досвіду віртуальної та доповненої реальності. Має структуру, яка робить взаємодію 3D і UI доступною через події введення Unity. Ядром цієї системи є набір базових компонентів Interactor і Interactable, а також менеджер взаємодії, який об’єднує ці два типи компонентів. Він також містить компоненти, які можна використовувати для пересування та малювання візуальних елементів. XR Interaction Toolkit містить набір компонентів, які підтримують наступні завдання взаємодії:

а) Кросплатформенний вхід контролера XR: Meta Quest (Oculus), OpenXR, Windows Mixed Reality тощо.

б) Навести, вибрати, захопити об’єкт.

в) Тактильний зворотний зв'язок через контролери XR.

г) Візуальний зворотний зв’язок (відтінки/лінії) для вказівки на можливі та активні взаємодії.

д) Базовий інтерфейс Canvas для взаємодії з контролерами XR.

е) Утиліта для взаємодії з XR Origin, камерою віртуальної реальності для роботи зі стаціонарною та кімнатною віртуальною реальністю[15].

Пакет Unity Input System не тільки підтримує доступ до керування користувача з кнопок і джойстиків контролера VR, але також надає доступ до даних відстеження XR і тактильних даних. Пакет Input System потрібен, якщо використовується XR Interaction Toolkit або плагін OpenXR. Цей пакет дозволяє користувачам керувати грою чи програмою за допомогою пристрою, дотику або жестів[16].

Пакет XR Hands визначає API, який дозволяє отримати доступ до даних відстеження рук із пристроїв, які підтримують відстеження рук. Щоб отримати доступ до даних відстеження рук, також потрібно ввімкнути плагін, який реалізує підсистему відстеження рук XR. Пакет XR Hand забезпечує:

а) XRHandSubsystem. Визначає інтерфейс підсистеми XR для даних відстеження рук.

б) Функція OpenXR HandTracking - ця функція реалізує XRHandSubsystem

плагіна OpenXR.

в) Функція Open XR Meta Aim Hand - ця функція надає дані з розширення XR_FB_hand_tracking_aim до специфікації OpenXR. Це розширення забезпечує базове розпізнавання жестів.

г) Структура XRHand - дані для окремої відстежуваної руки.

д) Структура XRHandJoint - дані для окремого суглоба або іншої відстежуваної точки руки.

е) Клас MetaAimHand - дані про жести, захват і прицілювання з функції Meta Aim Hand OpenXR[17].

Audio Spatializer SDK використовує «фізичні» характеристики сцени, такі як відстань і кут між AudioListener і AudioSource, щоб змінити властивості звуку, що передається користувачеві. Просторовість може покращити сприйняття того, що звук походить із певного місця сцени. Аудіосистема Unity підтримує просторування за допомогою плагінів, створених за допомогою Unity Audio Spatializer SDK. Unity не надає жодних вбудованих плагінів об’ємного простору, але кілька реалізацій плагінів доступні в сторонніх SDK для 3D-аудіо. Ці аудіо SDK зазвичай надають додаткові компоненти та інструменти Unity для 3D-аудіо.

Нижче наведено неповний список аудіо SDK сторонніх виробників, які надають плагіни просторового просторового аудіо Unity:

а) Microsoft Spatializer. Створено компанією Microsoft для платформ Windows, Android.

б) Oculus Spatializer Unity. Створено компанією Oculus для платформ Windows, Android.

в) Qualcomm 3D Audio Plugin for Unity. Створено компанією Qualcomm для платформ Windows, Android.

г) Steam Audio. Створено компанією Steam для платформ Windows, MacOS, Linux, Android.

д) 3DSP Audio SDK. Створено компанією Vive для платформ Windows, Android[18].

2.2 Unreal Engine

Unreal Engine - платформа, розроблена компанією Epic Games, вперше представлених у 1998 році. Написаний на C++ і має високий ступінь портативності, підтримуючи широкий спектр настільних, мобільних, консольних і віртуальних платформ. Останнє покоління, Unreal Engine 5, було запущено у квітні 2022 року. Його вихідний код доступний на GitHub. Однією з його головних особливостей є Nanite, механізм, який дозволяє імпортувати в ігри високодеталізований вихідний фотографічний матеріал. Технологія віртуалізованої геометрії Nanite дозволяє Epic скористатися перевагами минулого придбання Quixel, найбільшої у світі бібліотеки фотограмметрії станом на 2019 рік. Фотограмметрія - це наука та технологія отримання перевіреної інформації про фізичні об’єкти та навколишнє середовище за допомогою процесу запису, вимірювання та інтерпретації фотографічних зображень і моделей електромагнітного випромінювання та інших явищ. Nanite може імпортувати майже будь-яке інше тривимірне представлення об’єктів і середовищ, включаючи ZBrush і CAD - моделі, дозволяючи використовувати ресурси плівкової якості.

Lumen - ще один компонент, система глобального освітлення та відображень Unreal Engine 5, призначена для консолей нового покоління. Він використовує повністю динамічний конвеєр непрямого освітлення, що означає, що геометрія сцени, матеріал і властивості світла можуть змінюватися в будь-який час. Завдяки цьому Lumen значно покращує робочий процес художників. Це не тільки означає, що освітлення оновлюється миттєво, але й усуває час очікування на створення, щоб досягти остаточної якості освітлення. На додаток до цього, сцени з освітленням Lumen не потребують кубічних карт відображення, оскільки Lumen повністю замінює інші методи та здатний відображати геометрично точні відображення[19].

В Unreal Engine 5 додано ще один компонент віртуальні карти тіней. Віртуальні карти тіней - це новий метод відображення тіней, який

використовується для забезпечення узгодженого відображення тіней із високою роздільною здатністю, який працює з активами плівкової якості та великими динамічно освітленими відкритими світами. Відображення тіней або проєкція тіней - це процес, за допомогою якого тіні додаються до тривимірної комп'ютерної графіки. Тіні створюються шляхом перевірки видимості пікселя з джерела світла шляхом порівняння пікселя з зображенням глибини огляду джерела світла, що зберігається у формі текстури. Віртуальні карти тіней були розроблені з наступними цілями:

а) Значно збільшити роздільну здатність тіні, щоб відповідати високодеталізованій геометрії Nanite.

б) М'які тіні з розумними, контрольованими витратами на продуктивність.

в) Просте рішення, яке працює за замовчуванням з обмеженою кількістю необхідних налаштувань.

г) Замінити багато технік затінення стаціонарного світла єдиним уніфікованим шляхом.

Концептуально віртуальні карти тіней - це просто карти тіней із дуже високою роздільною здатністю. У поточній реалізації вони мають віртуальну роздільну здатність 16k x 16k пікселів. Відрізки використовуються для подальшого збільшення роздільної здатності для спрямованого світла. Щоб підтримувати високу продуктивність за прийнятної вартості пам'яті, розділяють карту тіней на фрагменти, розміром 128x128 кожна. Сторінки виділяються та відтворюються лише за потреби для затінення пікселів на екрані на основі аналізу буфера глибини. Фрагменти зберігаються в кеш-пам'яті між кадрами, якщо вони не стають недійсними рухомими об'єктами або світлом, що додатково покращує продуктивність[20].

Unreal Engine підтримує кілька мов програмування для розробки ігор та інтерактивних додатків. C++ є основною мовою програмування для розробки в Unreal Engine. Велика частина двигуна та графічних можливостей Unreal Engine розроблена в C++. Розробники можуть використовувати C++ для створення високопродуктивних ігор та налаштовувати графічні можливості гри. Також є

можливість розробки гри за допомогою графічної мови програмування під назвою Blueprint Visual Scripting. Система візуальних сценаріїв Blueprint в Unreal Engine - це повна система сценаріїв ігрового процесу, заснована на концепції використання інтерфейсу на основі вузлів для створення елементів ігрового процесу в Unreal Editor. Як і в багатьох поширених мовах сценаріїв, він використовується для визначення об'єктноорієнтованих класів або об’єктів у механізмі. Ця система є надзвичайно гнучкою та потужною, оскільки надає дизайнерам можливість використовувати практично повний спектр концепцій та інструментів, які зазвичай доступні лише програмістам. Крім того, спеціальна розмітка Blueprint, доступна в C++ реалізації Unreal Engine, дозволяє програмістам створювати базові системи, які можуть бути розширені дизайнерами. В січні 2022 року, Unreal Engine 5 вперше представив експериментальну підтримку Python через плагін "Python Editor Script Plugin." Ця можливість дозволяє використовувати Python для створення додаткових інструментів та сценаріїв для розробки, але це **е**кспериментальна підтримка[21].

Проекти Unreal Engine, можуть бути розгорнуті на різних платформах. Наприклад на операційних системах на операційних систем Windows, macOS, Linux, iOS та Android. Підтримує ігрові консолі, такі як Xbox, PlayStation та Nintendo. Unreal Engine дозволяє створювати проекти для різних віртуальних реальностей, таких як Oculus Rift, HTC Vive, PlayStation VR, і інших. Також для доповненої реальності на пристроях, таких як HoloLens та мобільні пристрої з підтримкою ARKit та ARCore. Має підтримку для відтворення у веб-браузерах за допомогою технології WebGL.

2.2.1 Огляд основних компонентів для реалізації VR програми в Unreal Engine

Починаючи з Unreal Engine 4.12, режим редагування віртуальної реальності тепер включено до версії двигуна, доступної через Epic Games Launcher. Unreal Editor VR - це інструмент для розробки віртуальної реальності у Unreal Engine, який дозволяє розробникам створювати та редагувати вміст віртуальної реальності безпосередньо в імітації віртуальної реальності. Він забезпечує

інтерактивну імерсивну середу для створення та редагування сцен та об'єктів, які використовуються у віртуальних іграх, симуляторах, архітектурних візуалізаціях та інших проектах. Основні функції Unreal Editor VR включають:

а) Віртуальний інтерфейс. Можна користуватися спеціальним інтерфейсом для доступу до різних інструментів, меню, панелей та налаштувань редагування.

б) Маніпуляція об'єктами. Можна переміщати, обертати, масштабувати та редагувати об'єкти у віртуальному просторі, використовуючи контролери.

в) Редагування сцен. Можна створювати, копіювати, видаляти та редагувати об'єкти, світло, матеріали, текстури та інші компоненти сцени в режимі реального часу.

г) Візуалізація та налаштування освітлення. Unreal Editor VR дозволяє налаштовувати освітлення в сцені, а також відображати, як воно виглядає в різних умовах.

д) Перевірка проекту. Можна тестувати свій проект безпосередньо в імітації віртуальної реальності, щоб переконатися, що все працює як слід.

Interactor Hand - це один із компонентів, який може бути використаний для створення взаємодії між рукою гравця та об'єктами у віртуальній реальності в Unreal Engine. Цей компонент допомагає симулювати руку гравця та взаємодію з об'єктами в VR-середовищі. Зазвичай це використовується для створення ігор або додатків для VR, де гравці можуть взаємодіяти з об'єктами віртуального світу за допомогою власних рук.

Основні функції компонента "Interactor Hand" містять:

а) Взаємодія з об'єктами. Використовувати руку гравця, яка моделюється за допомогою "Interactor Hand", для взаємодії з об'єктами у VR-середовищі. Це може мати збирання, переміщення, викидання та різні інші маніпуляції об'єктами.

б) Деталізована анімація руки. Цей компонент зазвичай охоплює докладну анімацію руки гравця, включаючи рухи пальців, долонь і зап'я́стя, щоб

створити більш реалістичний ефект.

в) Відстеження руху гравця. "Interactor Hand" використовує відстеження руху VR-гарнітури та контролерів, щоб точно реплікувати рух рук гравця в ігровому середовищі.

г) Взаємодія з інтерактивними об'єктами. Налаштовувати об'єкти у вашій грі чи додатку, щоб вони реагували на взаємодію руки гравця, наприклад, виконували реакцію на дотик, затискання або перетягування.

д) Можливість взаємодії з меню та інтерфейсами. "Interactor Hand" може використовуватися для взаємодії з меню, інтерфейсами та іншими що керує елементами в VR-середовищі.

Універсальний інструмент для перетворень Universal Gizmo - це інтерактивний віртуальний засіб, призначений для переміщення, обертання та масштабування об'єктів під час роботи в режимі віртуальної реальності. Є доступними як локальні, так і глобальні версії цього інструмента, які можуть бути перемкнуті за допомогою VR Mode Radial Menu. Елементи Універсального інструмента для перетворень мають червоний, зелений і синій кольори, що відповідають осям X, Y і Z, відповідно. Точно так само, як у стандартному редакторі, існують інші інструменти, які можуть бути використані специфічно для переміщення, обертання та масштабування ваших об'єктів. Локальні та глобальні версії цих інструментів доступні через VR Mode Radial Menu[22].

Меню радіальних команд VR Mode - це інтерактивне меню, яке використовується під час роботи в режимі віртуальної реальності в Unreal Engine. Це меню дозволяє користувачам легко отримувати доступ до різних інструментів та функцій для редагування проекту під час роботи в імерсивному VR-середовищі. Меню радіальних команд містить опції та інструменти для редагування та керування проектом. Ось деякі можливі опції та функції, які можуть бути доступні у цьому меню:

а) Універсальний інструмент для перетворень. Можливість перемикати між інструментами для переміщення, обертання та масштабування об'єктів у VR-середовищі.

б) Переміщення між локальними та глобальними координатами. Вибір між роботою у локальній системі координат об'єкта або глобальній системі координат сцени.

в) Інструменти для редагування матеріалів. Можливість редагувати матеріали та текстури об'єктів у VR.

г) Перехід між режимами редагування. Вибір між режимами редагування сцени, об'єктів чи інших параметрів проекту.

д) Навігація та переміщення. Опції для переміщення та навігації в VR-середовищі, такі як переміщення гравця, відстеження руху та інші.

е) Загальні параметри і налаштування. Доступ до різних загальних параметрів та налаштувань проекту, які можуть бути змінені під час роботи в VR[23].

РОЗДІЛ 3 ОБҐРУНТУВАННЯ ВИБОРУ ГАРНІТУРИ ТА ПЛАТФОРМИ ДЛЯ РОЗРОБКИ

3.1 Порівняльний аналіз характеристик та вибір гарнітури

Вибір між різними системами віртуальними реальностями, такими як Oculus, HTC Vive, PlayStation VR та інші, зазвичай залежить від потреб і вимог. Кожна з цих систем має свої переваги та недоліки, для мого проекту важливо враховувати кілька факторів при виборі: якість зображення, трекінг та мобільність. Нижче наведено порівняльні дані по якості зображення для таких гарнітур як Oculus Quest 2[24], HTC Vive[25], PlayStation VR[26] і Valve Index[27] (табл. 3.1).

Таблиця 3.1

Порівняльні дані якості зображення

	Роздільна здатність, пікселів	Поле зору, градусів	Оптика. Тип лінз	Оптика. Діаметр лінз	Оптика. Міжзіркова відстань	Частота оновлення
Oculus Quest 2	1832 x 1920	~ 100	асферичні лінзи	~ 40 мм	Ручне регулювання	90 Гц
HTC Vive	1080x1200	~ 110	асферичні лінзи	~ 28 мм	Ручне регулювання	90 Гц (можливо підвищити до 120 Гц)
PlayStation VR	960x1080	~ 100	асферичні лінзи	~ 36 мм	Ручне регулювання	90 Гц
Valve Index	1440x1600	~ 130	асферичні лінзи	~ 50 мм	Ручне регулювання	144 Гц

Загалом, Valve Index вражає своєю оптикою та швидкістю оновлення, що робить його однією з найкращих гарнітур. Нижче наведено порівняльні дані по трекінгу для тих самих гарнітур (табл. 3.2).

Таблиця 3.2

Порівняльні дані трекінгу

	Тип трекінгу	Кількість камер	Кількість базових станцій	Контролери
Oculus Quest 2	Внутрішній	чотири вбудовані камери	-	використовують внутрішній трекінг
HTC Vive	Зовнішній	-	2	використовують внутрішній трекінг і синхронізуються з базовими станціями.
PlayStation VR	Зовнішній	-	використовує PlayStation Camera для відстежування рухів користувача та контролерів	використовують внутрішній трекінг і синхронізуються з PlayStation Camera
Valve Index	Зовнішній	-	2	використовують внутрішній трекінг і синхронізуються з базовими станціями.

Зовнішній трекінг зазвичай вважається більш точним і надійним, але вимагає встановлення базових станцій. Внутрішній трекінг зручний шляхом відсутності необхідності встановлювати додаткове обладнання, але може бути менш точним в деяких ситуаціях.

Також потрібно порівняти характеристики автономності кожної з гарнітур (табл. 3.3). Зазначені розміри та вага відображають лише гарнітури без додаткового обладнання, такого як контролери, кабелі та інше.

Таблиця 3.3

Порівняльні дані автономності

	Автономність	Розмір	Потрібні додаткові прилади (додаткова вага)
Oculus Quest 2	Так	224 x 450 мм	Ні
HTC Vive	Ні	155,5 x 200,7 мм	Так
PlayStation VR	Ні	187×277 мм	Так
Valve Index	Ні	179,7 x 135,9 мм	Так

Загалом, автономні гарнітури, такі як Oculus Quest 2, зазвичай найбільш мобільні, оскільки вони не потребують підключення до комп'ютера або консолі. Гарнітури зовнішнього трекінгу, такі як HTC Vive, PlayStation VR і Valve Index, можуть надавати разючий імерсивний досвід, але вони менш мобільні через прив'язку до додаткового обладнання і проводів.

Оскільки для мого проекту важливими є автономна робота гарнітури без потреби підключення до комп'ютера або консолі, також якість зображення та частота оновлення кадрів має бути на достатньому рівні для зручного застосування для розробки проекту буде використовуватись Oculus Quest 2.

3.2 Обґрунтування вибору платформи для розробки механізмів віртуальної реальності

Unity легша платформою для вивчення в порівнянні з Unreal Engine. Має інтуїтивний і спрощений інтерфейс, який допомагає швидше зрозуміти основні концепції розробки. Надає інструменти та панелі для роботи з об'єктами, компонентами, анімацією та іншими аспектами проекту. Unity використовує мову програмування C#, яка є досить простою для вивчення та має багатий ресурс з багатьма навчальними матеріалами та прикладами. Має велику спільноту розробників та багато навчальних ресурсів, таких як відеоуроки, онлайн-курси,

форуми та документація. Це дозволяє швидше освоювати платформу. Unity Asset Store містить безліч готових активів, таких як моделі, текстури, звуки та інше, які можна легко інтегрувати у проект. Серед таких активів є досить багато й безкоштовних пропозицій. Unity надає безкоштовну версію для особистого використання та невеликих команд з обмеженими фінансовими ресурсами.

Для мого проекту в приорітеті є легкість навчання з великою базою навчальних ресурсів. Також ліцензовані безкоштовні готові об'єкти для сторення оточення, що пришвидшить роботу над виготовленням.Також ліцензовані безкоштовні готові об'єкти для створення оточення, що пришвидшить роботу над виготовленням. Також Unity підтримує розробку саме для гарнітури Oculus, яка також обрана для даної теми.

Враховуючи ці всі фактори розробка механізмів віртуальної реальності буде розроблятись на платформі Unity найновішої версії на момент розробки, а саме 2022.3.

РОЗДІЛ 4 ПРОЦЕС РОЗРОБКИ СЕРЕДОВИЩА ТА МЕХАМІЗМІВ ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ

4.1 Налаштування середовища для розробки

Спочатку створено новий проект в Unity Hub за шаблоном 3D (URP), який забезпечує швидку та просту ітерацію з оптимізованою найкращою у своєму класі графікою, яка працює на багатьох платформах.

Наступне, що потрібно зробити це перейти в Edit - Project Settings - XR Plagin Menegement та завантажити XR Plag-in Menegement. Цей пакет містить:

а) XRManagerSettings - це ScriptableObject, який можна використовувати для керування екземплярами XRLoader та їх життєвим циклом.

б) XRLoader - це базовий клас, від якого мають походити всі завантажувачі. Він надає базовий API, який XRManagerSettings може використовувати для керування життєвим циклом, і простий API, який можна використовувати для запиту певних підсистем у завантажувача.

в) XRConfigurationData - це атрибут, який дозволяє розміщувати налаштування збірки та виконання у вікні єдиних налаштувань.

г) XRPackageInitializationBase - допоміжний клас, який спрощує ініціалізацію пакета. Допомагає створити типові екземпляри XRLoader пакета та налаштування за замовчуванням під час інсталяції пакета.

д) XRBuildHelper - абстрактний клас, корисний для роботи з деякими шаблонами щодо переміщення налаштувань із редактора до середовища виконання.

е) XRGeneralSettings - містить налаштування, які застосовуються до всіх плагінів XR, а не до будь-якого окремого.

ж) Samples folder - містить реалізацію всіх частин XR Plug-in Management[28].

В завантаженому XR Plag-in Menegement обирається Plag-in Providers. Хоча в розробці механізмів віртуальної реальності я опираюсь на гарнітуру Oculus, але

розроблений матеріал буде доступний і на інших пристроях, тому для мого проекту потрібен OpenXR для Android та Windows, адже цей стандарт спрямований на спрощення розробки AR/VR, дозволяючи орієнтуватися на широкий спектр пристроїв AR/VR.

В OpenXR є можливість обрати для яких профілей буде взаємодія розроблятись. Обрано Meta Quest Touch Pro Controller Profile та Oculus Touch Controller Profile, тому що саме ці використовуються для Oculus[29]. Також обрано підтримку Meta Quest у розділі OpenXR Feature Groups.

На вкладці Window - Packege Meneger встановлюється пакет XR Interaction Toolkit версії 2.5.1(найновіший на момент розробки). Цей пакет є високорівневою компонентною системою. Надає структуру, яка робить взаємодію 3D і UI доступною через події введення Unity. Ядром цієї системи є набір базових компонентів Interactor і Interactable, а також менеджер взаємодії, який об’єднує ці два типи компонентів. Також містить компоненти, які можна використовувати для пересування та малювання візуальних елементів. В встановленому XR Interaction Toolkit, на вкладці Samples імпортується Starter Assets, який потрібен для оптимізації налаштування поведінки, включаючи стандартний набір дій введення та попередні налаштування для використання з XR Interaction Toolkit, які використовують систему введення.

Область для розробки в Unity розділена на основні компоненти необхідні для розробки, до яких можна додавати додаткові. Hierarchy (Ієрархія) в якій відображається кожен ігровий об’єкт у сцені, наприклад моделі, камери або префаби, інші сцени (рис. 4.1). Коли додаються або видаляються GameObjects у вікні Scene, вони також додаються або видаляються із вікна Hierarchy. Дочірні GameObjects успадковують рух і обертання батьківського GameObject.

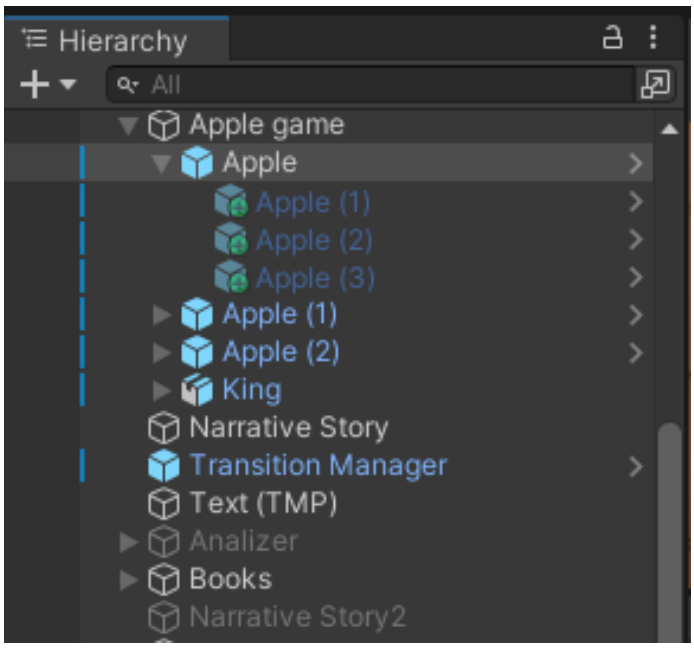


Рисунок 4.1 – Вікно «Ієрархія» в Unity

Scene (Сцена) містить гру чи програму повністю або частково (рис. 4.2).

Рисунок 4.2 – Вікно «Сцена» в Unity

Inspector (Вікно інспектора) використовується для перегляду та редагування властивостей і налаштувань майже всього в редакторі Unity, у тому числі GameObjects, компоненти Unity, ресурси, матеріали, налаштування та параметри редактора (рис. 4.3).

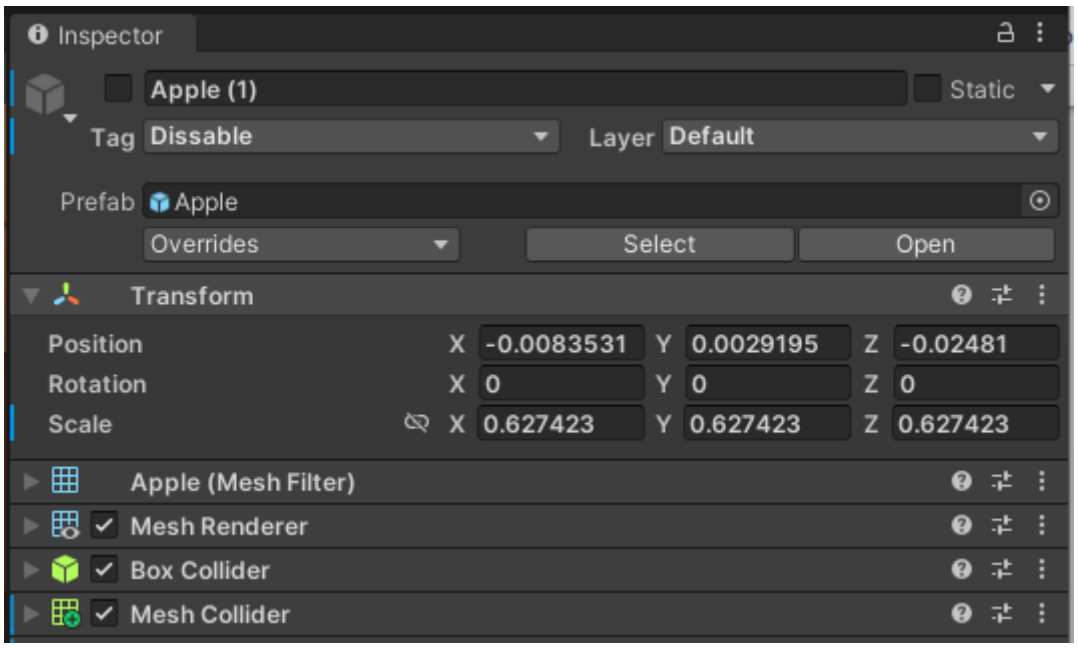


Рисунок 4.3 – Вікно «Інспектора» в Unity

Project (Вікно проекту) відображає всі файли, пов’язані з проектом, і є основним способом навігації та пошуку ресурсів та інших файлів проекту у програмі (рис. 4.4).

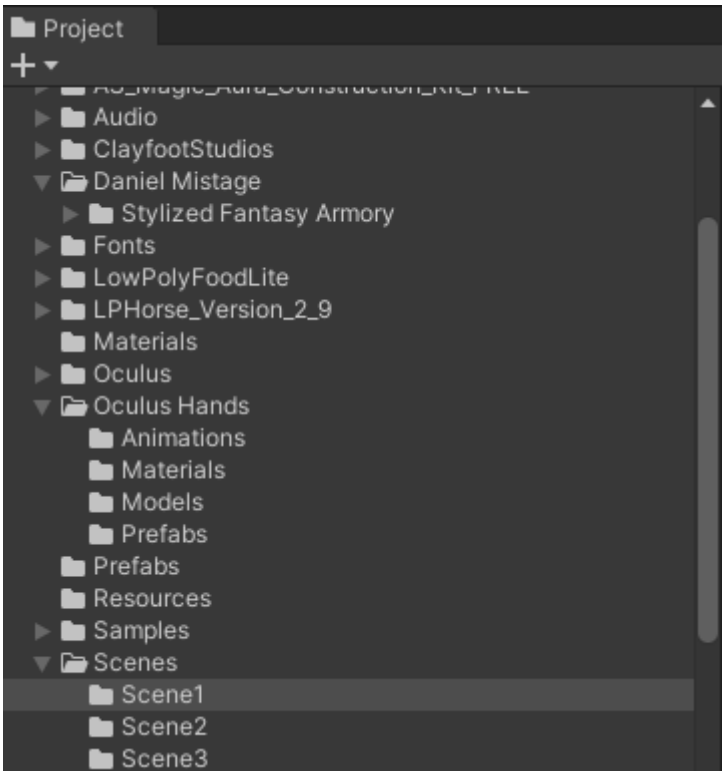


Рисунок 4.4 – Вікно «Проект» в Unity

4.2 Моделювання сцени

4.2.1 Завантаження та розміщення об’єктів

Моделі - це файли, що містять дані про форму та зовнішній вигляд. Файли моделей можуть містити різноманітні дані, зокрема сітки,матеріалів, ітекстури. Вони також можуть містити дані анімації для анімованих персонажів.

В Unity Asset Store обрано та завантажено відповідний до теми роботи набір «STYLIZED Fantasy Armory - Low Poly 3D Art», який містить готові моделі, префаби та текстури для роботи[30]. Для цього потрібно в Unity Asset Store обрати Add to My Assets, після чого в Package Manager імпортувати обраний набір моделей. Після завантаження, набір моделей знаходиться в Project – Assets, де зберігаються всі файли, пов’язані з проектом. Додано, необхідні для моделювання сцени, об’єкти в область Hierarchy, ці ж об’єкту починають відображатись в вікні Scene.

Розміщення об’єктів в вікні Scene може бути реалізовано двома шляхами. Перший, вибрати потрібний об’єкт та натиснути клавіші W, E, R після чого, навколо об’єкта з’явиться відповідне графічне накладання(gizmo) (рис. 4.5).

W – зміна положення в координатах X, Y і Z; E – обертання навколо осей X, Y і Z, вимірюється в градусах; R – масштабування вздовж осей X, Y і Z. Значення «1» — вихідний розмір (розмір, у якому об’єкт було імпортовано).

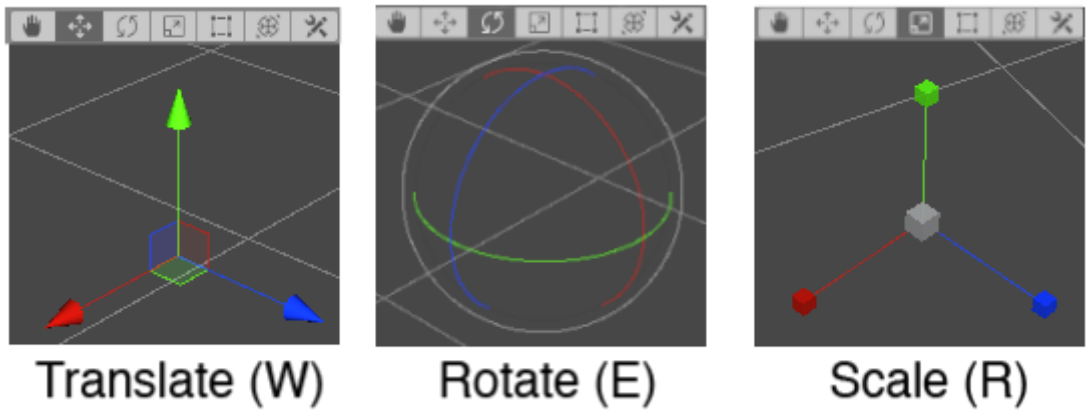


Рисунок 4.5 – Зовнішній вигляд для кожного типу трансформації

Другий, в вікні Inspector знаходиться компонент Transform, який має значення Position (x, y, z), Rotation(x, y, z) та Scale (x, y, z), куди можна вводити цифрові значення[31].

Використовуючи елементи трансформації необхідні об’єкти розміщено на сцені. Створено дерев’яну кімнату в якій є табурет та стіл за тематикою навчання дитини в школі або бібліотеці. Додано шафи з книжками та інші елементи для створення атмосфери більш реалістичного віртуального світу. Навпроти користувача знаходиться вікно з відображенням навколишнього середовища, щоб уникнути відчуття замкненості.

4.2.2 Види та налаштування джерел світла

Освітлення в Unity працює, наближаючи себе до того, як світло поводиться в реальному світі. Unity використовує детальні моделі того, як працює світло для більш реалістичного результату, або спрощені моделі для більш стилізованого результату. Пряме світло – це світло, яке випромінюється, потрапляє на поверхню один раз, а потім відбивається безпосередньо в датчик (наприклад, сітківку ока або камеру). Непряме світло – це все інше світло, яке в кінцевому підсумку відбивається на датчик, включаючи світло, яке кілька разів потрапляє на поверхні, і світло неба. Щоб досягти реалістичних результатів освітлення, потрібно імітувати пряме та непряме світло. Unity може обчислювати пряме освітлення, непряме освітлення або як пряме, так і непряме освітлення[32]. Для використання доступні декілька видів світла.

Point Light. Таке світло розташоване в точці простору і однаково розповсюджує світло в усіх напрямках. Напрямок світла, що падає на поверхню, — це лінія від точки контакту назад до центру світлового об’єкта. Інтенсивність зменшується з віддаленням від світла, досягаючи нуля в заданому діапазоні. Інтенсивність світла обернено пропорційна квадрату відстані від джерела. Це відоме як «закон зворотних квадратів» і схоже на те, як світло поводиться в реальному світі. Точкове освітлення корисне для імітації ламп та інших локальних джерел світла в сцені. Також можна використовувати його, щоб змусити іскру чи вибух переконливо освітлити оточення.

Spot Light (Прожекторне світло). Подібно до Point Light, Spot Light має певне розташування та діапазон, у якому світло падає. Однак точкове світло обмежене кутом, що призводить до конусоподібної області освітлення. Центр конуса вказує в напрямку вперед (Z) світлового об’єкта. Світло також слабшає на краях конуса прожектора. Розширення кута збільшує ширину конуса, а разом з цим збільшується розмір цього переходу, відомого як «півтінь». Прожектори зазвичай використовуються для штучних джерел світла, таких як ліхтарики, автомобільні фари та прожектори. Якщо напрямок керується сценарієм або анімацією, рухливий прожектор освітлюватиме лише невелику ділянку сцени та створюватиме драматичні світлові ефекти (рис. 4.6).

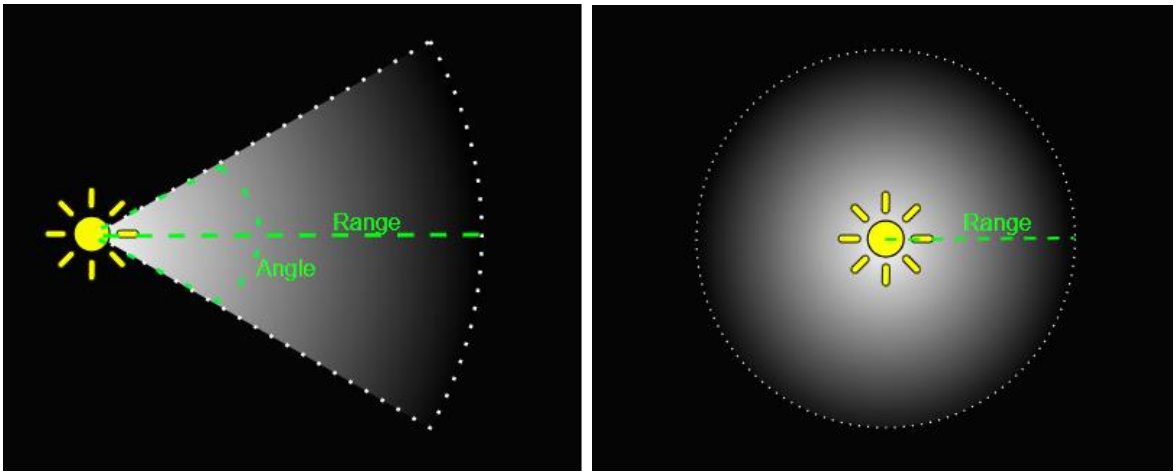


Рисунок 4.6 – Графічне зображення освітлення для Point Light та Spot Light

Directional Light (Спрямоване світло). Спрямоване світло корисне для створення таких ефектів, як сонячне світло у ваших сценах. Поводячись багато в чому як сонце, спрямоване світло можна розглядати як віддалені джерела світла, які існують нескінченно далеко. Спрямоване світло не має ідентифікованого положення джерела, тому світловий об’єкт можна розмістити будь-де на сцені. Усі об’єкти сцени освітлюються так, ніби світло завжди йде з одного напрямку. Відстань світла від цільового об’єкта не визначена, тому світло не зменшується. Якщо світло розташувати збоку, паралельно до землі, можна досягти ефекту заходу сонця. Крім того, направлення світла вгору призводить до того, що небо стає чорним, ніби зараз ніч. Якщо світло буде під кутом зверху, небо буде нагадувати денне світло (рис. 4.7).

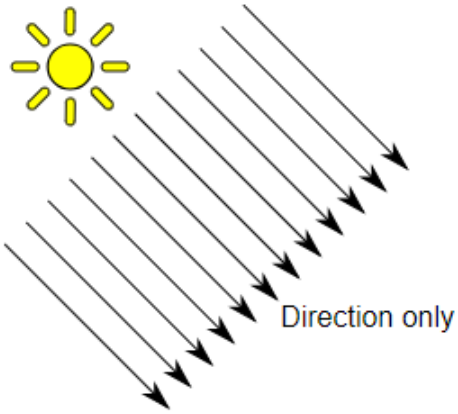


Рисунок 4.7 – Графічне зображення освітлення для Directional Light

Area Light (Зону освітлення). Ви можете визначити зону освітлення за допомогою однієї з двох форм у просторі: прямокутника або диска. Область освітлення випромінює світло з одного боку цієї форми. Випромінюване світло рівномірно поширюється в усіх напрямках по поверхні цієї форми. Властивість Range визначає розмір цієї форми. Інтенсивність освітлення, що забезпечується зональним світлом, зменшується зі швидкістю, яка визначається оберненим квадратом відстані від джерела світла (див. закон обернених квадратів). Оскільки цей розрахунок освітлення є досить інтенсивним для процесора, Area Lights недоступні під час виконання й можуть бути записані лише в карти освітлення. Оскільки зональне освітлення освітлює об’єкт одночасно з кількох різних напрямків, затінення має тенденцію бути більш м’яким і тонким, ніж інші типи світла. Можна використовувати його для створення реалістичного вуличного ліхтаря або групи ліхтарів поблизу гравця. Світло невеликої площі може імітувати менші джерела світла (наприклад, освітлення всередині будинку), але з більш реалістичним ефектом, ніж точкове світло (рис. 4.8).

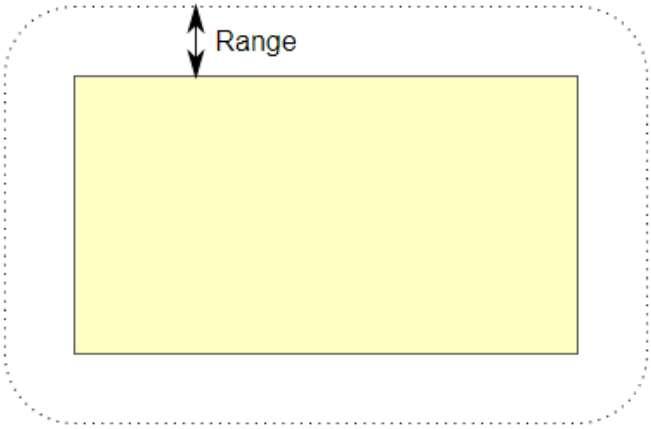


Рисунок 4.8 – Графічне зображення освітлення для Area Light

Під час моделювання сцени використано Directional Light для створення ефекту сонячного світла. Також додано Point Light для всіх додаткових джерел світла від світильників. Для всіх джерел світла виставлено персональні налаштування світла: тип, діапазон, кут, колір, режим, інтенсивність.

Тип показує поточний тип світла. Діапазон визначає, на яку відстань поширюється світло, що випромінюється від центру об’єкта (тільки для Point та Spot). Кут визначте кут (у градусах) біля основи конуса прожектора (тільки для Spot). Колір це колір випромінюваного світла.

Існує три режими: Baked, Realtime, Mixed. Якщо використовувати режим Baked, то Unity попередньо розраховує освітлення перед виконанням і не включає їх у будь-які розрахунки освітлення під час виконання. Для Realtime Unity обчислює та оновлює освітлення кожного кадру під час виконання. Unity не виконує попередніх обчислень для Realtime Lights. В режимі Mixed - Unity виконує деякі обчислення для Mixed Lights заздалегідь, а деякі – під час виконання.

Інтенсивність це налаштування яскравості світла. Значення за замовчуванням для Directional light становить 0,5 для всіх інших становить 1.

4.3 Принцип роботи контролерів та керування руками. Їх додавання та налаштування

XR Origin служить центром відстеження в сцені XR. Конфігурація XR Origin це набір GameObjects і компонентів, які працюють разом, щоб трансформувати дані з підсистем відстеження XR у світовий простір. Пристрої XR надають дані відстеження в реальних одиницях відносно точки, вибраної системою XR під час ініціалізації. Unity вирівнює XR Origin GameObject у сцені з цією початковою точкою. Початкова позиція XR Origin у сцені також визначає початкову позицію за замовчуванням камери. GameObjects, що представляють відстежувані сутності, такі як гарнітура користувача або портативний пристрій, є дочірніми елементами XR Origin GameObject в ієрархії сцени. Це означає, що зміни в положенні та обертанні на основі оновлень відстеження відносяться до XR Origin. Коли користувач рухається в реальному світі, ці дочірні GameObjects рухаються відносно XR Origin у сцені. XR Origin не рухається самостійно.

В меню GameObject - XR додано XR Origin (VR), який є частиною пакету XR Interaction Toolkit, що включає контролер GameObjects, налаштований для введення на основі дії. Після чого в вікно Hierarchy додано Origin (XR Rig), в якому знаходяться правий та лівий контролери. Ці контролери мають функцію XR Controller (Action-based), що має параметри для налаштувань, в які додано стандарті шаблони введення з Starter Assets. Нижче описано дії, які мають бути прив’язані до XR Controller (Action-based) та їх значення:

а) Position, Rotation, та Select. Основна взаємодія позиція, обертання та вибір.

б) Tracking State. Відстеження стану чи дійсні вхідні дані позиції та/або обертання.

в) Activate. Активація вибраного об'єкта.

г) UI Press. Взаємодія з елементами полотна інтерфейсу користувача.

д) Haptic Device. Ідентифікація пристрою для надсилання тактильних імпульсів.

е) Rotate та Translate Anchor. Маніпулювання обраним об’єктом на відстані.

Наступний крок, надати форму та текстуру контролерам. Для цього в батьківську папку Left Controller та Right Controller додано залежну з готовими префаб моделями завантажені з офіційного сайту Oculus[32], що мають готові налаштування та текстури.

OVRInput надає уніфікований API введення для кількох типів контролерів. Він використовується для запиту віртуального або необробленого стану контролера, наприклад кнопок, джойстиків, тригерів і ємнісних сенсорних даних. Для цього потрібно викликати OVRInput.Update() і OVRInput.FixedUpdate() один раз на кадр на початку будь-яких методів Update і FixedUpdate будь-якого компонента відповідно. OVRInput надає дані про позицію дотику та орієнтацію через GetLocalControllerPosition() і GetLocalControllerRotation(), які повертають Vector3 і Quaternion відповідно. Пози контролера повертаються системою відстеження та прогнозуються одночасно з гарнітурою. Ці пози в тій самій системі координат, що й гарнітура, відносно початкової пози центрального ока, і їх можна використовувавти для візуалізації рук або об’єктів у 3D-світі. Вони також скидаються за допомогою OVRManager.display.RecenterPose(), подібно до пози голови та очей.

Основним використанням OVRInput є доступ до стану введення контролера через Get(), GetDown() і GetUp(). Get() запитує поточний стан контролера. GetDown() запитує, чи було натиснуто контролер у цьому кадрі. GetUp() запитує, чи було звільнено контролер у цьому кадрі. Вони надають доступ до різних наборів елементів керування. Ці набори елементів керування представлені через перерахування, визначені OVRInput, а саме OVRInput.Button, що контролює радиційні кнопки на геймпадах, контролерах і кнопці «Назад»; OVRInput.Touch, що контролює ємнісно-чутливі поверхні керування знаходяться на контролері; OVRInput.NearTouch, що контролює чутливі до наближення поверхні керування, що знаходяться на контролері; OVRInput.Axis1D, що контролює одновимірні елементи керування, такі як тригери, які повідомляють про стан із плаваючою комою; OVRInput.Axis2D, що контролює двовимірні елементи керування, включаючи джойстики. Повідомляє про стан Vector2.

Контролери оснащені ємнісними поверхнями керування, які виявляють, коли пальці або великі пальці користувача здійснюють фізичний контакт (дотик), а також коли вони знаходяться в безпосередній близькості (NearTouch). Це дозволяє виявити кілька різних станів взаємодії користувача з певною поверхнею керування. Наприклад, якщо вказівний палець користувача повністю віддалено від поверхні керування, NearTouch для цього елемента керування повідомить помилку. Коли палець користувача наближається до елемента керування та наближається до нього, NearTouch повідомить істину до того, як користувач зробить фізичний контакт. Коли користувач здійснює фізичний контакт, дотик для цього елемента керування повідомляє про істинність. Коли користувач натискає тригер індексу вниз, кнопка для цього елемента керування повідомить істину. Ці окремі стани можна використовувати для точного виявлення взаємодії користувача з контролером і активації різноманітних схем керування.

При доступі до контролерів як комбінованої пари за допомогою OVRInput.Controller.Touch віртуальне відображення точно відповідає макету типового геймпада, розділеного на ліву та праву руки.

Тож до контролерів додано префаб Direct Interactor з XR Interaction Toolkit, який містить скрипт XR Direct Interactor. Цей код визначає сценарій C# для програми Unity. Зокрема, це клас під назвою розширює XR Base Controller Interactor. Він призначений для прямої взаємодії з об’єктами, з якими можна взаємодіяти за допомогою тригерів. Код покладається на тригери для виявлення інших об’єктів і взаємодії з ними. Ці тригери використовуються для оновлення набору дійсних цілей. Код дозволяє налаштувати параметри, пов’язані з фізикою, такі як маска шару фізики та взаємодія тригера. Сценарій обробляє тригерні події, такі як On Trigger Enter, On Trigger Stay і On Trigger Exit, для виявлення та підтримки контакту з взаємодіючими елементами. Перевіряє, чи GameObject має необхідну конфігурацію коллайдера для інтерактора. Загалом, цей сценарій надає гнучке й оптимізоване рішення для прямої взаємодії з об’єктами в програмі Unity, враховуючи продуктивність і можливість налаштування.

Для предметів з якими буде відбуватись взаємодія додано скрипт Disable

Grabbing Hand Model, який призначений для управління моделями рук у віртуальній реальності, перемикаючи їх видимість залежно від взаємодій (дод.А).

Окремо створено взаємодію з віртуальним середовищем, шляхом відстеження рук та використання руху як методу передачі даних для взаємодії. Використання рук як методу введення створює нове відчуття присутності, підвищує соціальну залученість і забезпечує більш природну взаємодію. Відстеження рук доповнює контролери і не призначене для заміни контролерів у всіх сценаріях, тому таку взаємодію додано лише для одного рівня. Існують деякі ускладнення, які виникають під час розробки взаємодії через руки. Наприклад внутрішні технологічні обмеження, як-от обмежений обсяг відстеження та проблеми з перешкодами. Віртуальні об’єкти не забезпечують тактильний зворотний зв’язок, на який ми покладаємося під час взаємодії з об’єктами реального життя. Вибір жестів руками, які активують систему без випадкових тригерів, може бути складним.

Максимально ефективно можна використовувати руку для примітивних взаємодій або основних цілей: націлювання(наведення), яке переміщує фокус на певний об’єкт; вибір, який дозволяє користувачам вибрати або активувати цей об’єкт; маніпуляції або переміщення, обертання або масштабування об’єкта в просторі.

Програма відображає руки так само, як і будь-який інший пристрій введення. Спочатку потрібно налаштувати камеру, вибрати руки як пристрій введення та додати збірку рук, щоб відобразити руки у формі за замовчуванням. На вкладці Inspector перейти до OVR Manager - Quest Features, а потім у списку Hand Tracking Support вибрати «Controllers and Hands». Або вибрати опцію «Hands Only», щоб використовувати руки як спосіб введення без будь-яких контролерів. Коли обрано контролери та опцію «Hands Only», Meta Quest автоматично додає <uses-permission android:name="com.oculus.permission.HAND_TRACKING" /> і <uses-feature android:name="oculus.software.handtracking" android:required="false" /> у файлі AndroidManifest.xml. Якщо програма підтримує контролери та руки, для android:required встановлено значення false, що означає, що програма надає

перевагу використанню рук, якщо вони присутні, але програма продовжує працювати з контролерами за відсутності рук. Якщо програма підтримує лише руки, android:required має значення true. Oculus автоматично додає обидва ці теги, і файл маніфесту Android не вимагає ручного оновлення. Для мого проекту обрано підтримку рук та контролів.

На вкладці Hierarchy розгорнути OVRCameraRig - TrackingSpace», щоб додати ручні збірні елементи під лівий і правий анкери. Перетягнути OVRHandPrefab під кожну ручну прив’язку на вкладці Hierarchy. На вкладці Hierarchy в розділі RightHandAnchor виберати OVRHandPrefab, а потім на вкладці Hierarchy у розділі OVR Hand, OVR Skeleton і OVR Mesh змінити тип руки на праву руку. Для лівого префабу не потрібно нічого робити, оскільки тип руки встановлюється на ліву руку автоматично. На вкладці Hierarchy виберати обидва збірні елементи OVR Hand, а потім на вкладці Inspector перевірити, що встановлено прапорці OVR Skeleton, OVR Mesh і OVR Mesh Renderer для відтворення рук у програмі. На цьому етапі програма може відображати руки як пристрій введення.

Щоб створити та відобразити анімовану 3D-модель рук, OVR Mesh Renderer поєднує дані, отримані OVR Skeleton і OVR Mesh. OVR Skeleton повертає позу зв’язування, ієрархію кісток і дані коллайдера капсули. OVR Mesh завантажує вказаний 3D-ресурс із середовища виконання Oculus і представляє його як сітку Unity Engine.

Фізичні капсули представляють об’єм кісток у руці, який використовується для ініціювання взаємодії з фізичними об’єктами та створення подій зіткнення з іншими твердими тілами у фізичній системі. На вкладці Hierarchy розгорнути OVRCameraRig - TrackingSpace, а потім виберати префаб OVR Hand, який буде використовувати для фізичної взаємодії. Для OVR Skeleton ввімкнути значення Enable Physics Capsules.

Стандартні моделі рук мають шкіру. Засіб візуалізації сітчастої оболонки надає властивості поверхонь, які визначають спосіб візуалізації моделі на сцені. Вигляд рук налаштавано, як напівпрозорі блакитного кольору.

Щоб створити багатий досвід використання рук, потрібно включити кілька

взаємодій, враховуючи, як розміщено об’єкт. Об’єкти ближнього поля знаходяться в межах досяжності. Пряма взаємодія, як-от тикання чи щипання, добре працює з цими об’єктами. Об’єкти далекого поля знаходяться поза межами досяжності руки і потребують трансляції променів, яка спрямовує проміння на об’єкти на великій відстані. Це дуже схоже на взаємодію сенсорного контролера. Але в моєму проекті, де всі необхідні об'єкти для взаємодії перебувають на столі перед користувачем достатньо прямої взаємодії.

Тикання та щипання - це жести в режимі реального часу, які дуже інтуїтивно зрозумілі для будь-якого користувача, щоб виконувати основні завдання, такі як встановлення фокусу, вибір або маніпулювання об’єктом у просторі. Тикання вимагає, витягнути та перемістити палець до об’єкта, поки палець не зіткнеться з об’єктом у просторі. API OVR Skeleton і OVR Hand надають інформацію, необхідну для відтворення повністю сформульованого представлення реальних рук користувача у віртуальній реальності без використання контролерів: інформація про кістки, розташування та орієнтація руки та пальця, сила щипків, поза вказівника для інтерфейсу користувача, відстеження, розмір руки, системний жест для відкриття універсального меню[34].

4.4 Додавання об’єктів та їх конфігурація для взаємодії

4.4.1 Глобальні налаштування об’єктів для взаємодії

Щоб з об’єктами можна було взаємодіяти потрібно додати декілька класів один з яких це Rigidbody - клас програмування, визначений у ядрі механізму, доступ до якого здійснюється за допомогою сценаріюю Додавання компонента Rigidbody до об’єкта поставить його рух під контроль фізичного механізму Unity. Навіть без додавання будь-якого коду об’єкт Rigidbody буде тягнутися вниз під дією сили тяжіння та реагуватиме на зіткнення з вхідними об’єктами, якщо також присутній правий компонент Collider. Rigidbody також має API сценаріїв, який дозволяє застосовувати силу до об’єкта та контролювати його фізично реалістичним способом. Наприклад, поведінку автомобіля можна визначити

силами, які прикладаються колесами. З огляду на цю інформацію, фізичний механізм може впоратися з більшістю інших аспектів руху автомобіля, тому він буде реалістично прискорюватися та правильно реагувати на зіткнення.

Поширеною проблемою, використовуючи Rigidbody, є те, що фізика гри здається уповільненою. Насправді це пов’язано з масштабом, використаним для моделей. Параметри гравітації за замовчуванням припускають, що одна світова одиниця відповідає одному метру відстані. У нефізичних іграх немає великої різниці, якщо всі моделі мають 100 одиниць завдовжки, але при використанні фізики вони розглядатимуться як дуже великі об’єкти. Якщо використовується великий масштаб для об’єктів, які мають бути малими, здається, що вони падають дуже повільно – фізичний механізм вважає, що це дуже великі об’єкти, які падають на дуже великі відстані. Пам’ятаючи про це, потрібно створювати об’єкти більш-менш схожими в масшатабі як в реальному житті.

Клас Rigidbody дає можливість встановлювати індивідуальні налаштування для твердих тіл. Маса об'єкта (за замовчуванням у кілограмах). Який опір повітря впливає на об'єкт під час руху від сил. 0 означає відсутність опору повітря, а нескінченність змушує об’єкт негайно зупинятися. Який опір повітря впливає на об’єкт під час обертання. 0 означає відсутність опору повітря. Обирати чи дії на об'єкт гравітація, чи ні. Налаштування кінематики, інтерполяції, запобігання проходженню об’єктів, що швидко рухаються, крізь інші об’єкти без виявлення зіткнень. Встановлювати обмеження руху твердого : зупиняти вибіркове переміщення Rigidbody по осях X, Y і Z світу; зупиняти вибіркове обертання Rigidbody навколо локальних осей X, Y і Z[35].

Наступний клас, який потрібно додано це XR Grab Interactable. Інтерактивний компонент, який підключається до системи взаємодії (через XRInteractionManager), щоб забезпечити базову функціональність захоплення об’єкта. Може приєднатися до вибраного інтерактора та слідувати за ним, дотримуючись фізики (і успадковувати швидкість, коли його відпускають). Далі описано налаштування, які можна встановлювати для цього класу.

Менеджер XR Interaction Manager, з яким цей клас спілкується. Обирати

коллайдери для взаємодії з цим класом (якщо пусте, використовуватимуться будь-які дочірні колайдери). Контролює, скільки інтеракторів можуть вибрати цей інтеракт. Визначати, як об’єкт переміщується, коли його вибрано, або через встановлення швидкості Rigidbody, переміщення кінематичного Rigidbody під час Fixed Update, або шляхом безпосереднього оновлення Transform кожного кадру. Встановити щоб об’єкт слідував позиції Interactor, коли вибрано. Застосовувати згладжування, стежачи за положенням, коли вибрано об’єкт. Змушує цей об’єкт мати силу тяжіння під час відпускання.

Додано Mesh Collider, який бере Mesh Asset і створює свій Collider на основі цього Mesh. Це точніше для виявлення зіткнень, ніж використання примітивів для складних сіток. Меш-коллайдери, позначені як Convex, можуть стикатися з іншими сітчастими коллайдерами. Mesh Collider будує своє представлення зіткнення з Mesh, приєднаного до GameObject, і зчитує властивості приєднаного Transform, щоб правильно встановити його положення та масштаб. Перевагою цього є те, що можна зробити форму коллайдера точно такою ж, як форма видимої сітки для GameObject, що створює більш точні та автентичні зіткнення. Однак ця точність пов’язана з більшими накладними витратами на обробку, ніж зіткнення за участю примітивних коллайдерів (таких як Sphere, Box і Capsule), тому краще використовувати Mesh Colliders економно. Грані в сітках зіткнень односторонні. Це означає, що GameObjects можуть проходити крізь них з одного боку, але стикатися з ними з іншого.

Під час використання Mesh Collider існують деякі обмеження. GameObjects, які мають компонент Rigidbody, підтримують лише сітчасті коллайдери, у яких увімкнено параметр Convex, фізичний механізм може імітувати лише опуклі сітчасті колайдери. Щоб Mesh Collider працював належним чином, у Mesh має бути активовано читання/запис за будь-якої з цих обставин: трансформація Mesh Collider має негативне масштабування (наприклад, (–1, 1, 1)), а сітка є опуклою; перетворення Mesh Collider має перекіс або зріз (наприклад, коли повернуте перетворення має масштабоване батьківське перетворення); для прапорців параметрів приготування Mesh Collider встановлено будь-яке значення, окрім

стандартного. Не слід змінювати геометрію сітки, яка використовується для коллайдерів, оскільки фізичний механізм має перебудовувати внутрішню структуру прискорення зіткнення сітки щоразу, коли змінюється сітка. Це спричиняє значні накладні витрати на продуктивність. Для сіток, які повинні стикатися та змінюватися під час виконання, часто краще наближати форму сітки за допомогою примітивних колайдерів, таких як капсули, сфери та ящики.

Додано компонент Mesh Renderer, який відтворює сітку. Він працює з компонентом Mesh Filter на тому самому GameObject; Mesh Renderer відтворює сітку, на яку посилається Mesh Filter. Щоб візуалізувати сітку, що деформується, потрібно використовувати засіб візуалізації Skinned Mesh. У коді C# клас MeshRenderer представляє компонент Mesh Renderer. Клас MeshRenderer успадковує більшу частину своєї функціональності від класу Renderer. Таким чином, цей компонент має багато спільного з іншими компонентами, які успадковують Renderer, наприклад Line Renderer і Trail Renderer. Цей компонент має наступні налаштування: матеріали,освітлення, світлове відображення, зонди, додаткові налаштування. У розділі «Матеріали» перераховано всі матеріали, які використовує цей компонент. Розділ «Освітлення» містить властивості, пов’язані з освітленням. Розділ «Відображення світла» містить властивості, що стосуються карт освітлення. Цей розділ відображається лише тоді, коли для параметра «Отримувати глобальне освітлення» встановлено значення Карти світла. Розділ «Зонди» містить властивості, пов’язані зі світловими зондами та зондами відбиття.

Тож, маючи ці класи при наведені на об’єкт джойстиком та затиснувши кнопку, предмет буде взято в руку, також предмет має гравітацію, тому його можна штовхати, кидати, обертати.

4.4.2 Створення інтерактивної зони для зникнення об’єктів

Для інтерактивну вирішено додати об’єкт, який в програмі названо «Tresh», в якому будуть зникати предмети, що потрапляють всередину.

Для цього для даного батьківського компоненту додано модель ящика для якого присвоєно клас Mesh Renderer, що описаний в попередньому розділі та

налаштовано параметри. Цього класу буде достатньо, адже планується, що інші предмети мають потрапляти всередину об’єкта ,а не відбиватись від нього, тому створювати йому колайдер не потрібно.

Додано головний елемент завдяки якому буде відбуватись зникнення предметів – куб. Йому додано колайдер та проставлено галочку для Is Trigger. Тригер не реєструє зіткнення з вхідним Rigidbody. Натомість він надсилає повідомлення OnTriggerEnter, OnTriggerExit і OnTriggerStay, коли тверде тіло входить або виходить із тригерного обсягу.

Створено новий скрипт Trigger Zone, код показано на рис 4.9.

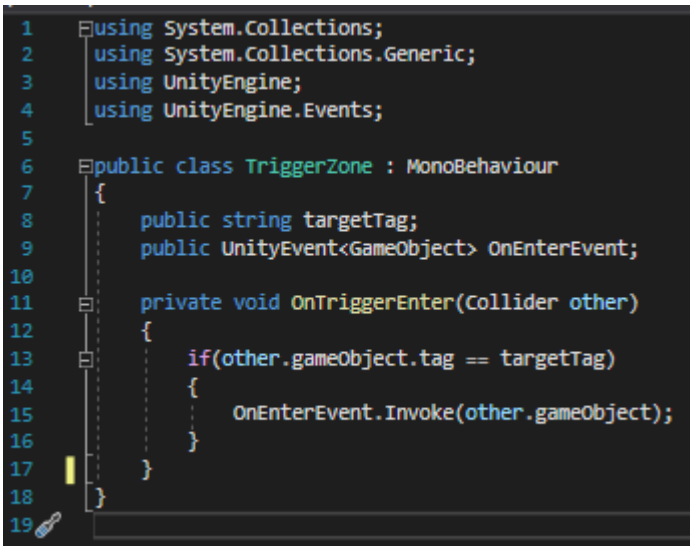


Рисунок 4.9 – Скрипт Trigger Zone

Загалом, цей скрипт призначено для прикріплення до GameObject у Unity як компоненту, створюючи зону тригера. Якщо інший GameObject потрапляє в цю зону та має вказаний тег, це викличе подію (OnEnterEvent), яку можна використовувати для виконання конкретних дій або змін поведінки.

Додано новий скрипт Trash Can, призначений для вимикання видимості об'єктів, які потрапляють у зону тригера, код показано на рис 4.10.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class TrashCan : MonoBehaviour
{
    private void Start()
    {
        GetComponent<TriggerZone>().OnEnterEvent.AddListener(InsideTrash);
    }
    public void InsideTrash(GameObject go)
    {
        go.SetActive(false);
    }

}
```

Рисунок 4.10 – Скрипт Trash Can

Наступним етапом є додавання візуального ефекту, який буде символізувати активацію зони та можливість зникнення об’єктів. Переходячи по шляху GameObject - 3D Object, створено 3D-модель чотирикутника. За замовчуванням читирикутник - це квадрат із ребрами в одну одиницю, розділений на два трикутники та орієнтований у площині x та y локального простору координат. Їх можна використовувати чотирикутник як екран для зображення або фільму. Також можна використовувати чотирикутники для реалізації простого графічного інтерфейсу та інформаційних дисплеїв, частинок, спрайтів і самозваних зображень як замінників твердих об’єктів, які виглядають на відстані. Додано клас Mesh Renderer.

У вікні Project – Assets додано Create – Shader Graph – URP – Unlit Shader Graph. Unlit Master Stack дозволяє створювати шейдери, на які не впливає освітлення. Цей тип матеріалу Master Stack має власний набір параметрів графіка. Через зв’язок між налаштуваннями та блоками це впливає на те, які блоки мають відношення до графіка. Далі описано про блоки, які цей тип матеріалу Master Stack додає за замовчуванням, і які властивості блоків налаштовуються для параметрів графіка цього типу матеріалу Master Stack. При створені нового Unlit Master Stack, Vertex Context за замовчуванням містить такі блоки: position (позиція вершини простору об’єкта на вершину), normal (нормальна вершина простору об’єкта на вершину), tangent (дотична до вершини простору об’єкта на вершину). При створені нового Unlit Master Stack, Fragment Context за

замовчуванням містить такі блоки: Base Color (базовий колір матеріалу); Emission (колір світла, яке випромінює поверхня цього матеріалу); Alpha (значення альфа матеріалу, це визначає прозорість матеріалу - очікуваний діапазон 0–1).

Залежно від налаштувань графіки, Shader Graph може додавати наступні блоки до контексту фрагмента. Alpha Clip Threshold - обмеження значення альфа-каналу, яке HDRP використовує, щоб визначити, чи відтворювати кожен піксель. Якщо значення альфа-каналу пікселя дорівнює обмеженню або перевищує його, HDRP візуалізує піксель. Якщо значення нижче за обмеження, HDRP не відтворює піксель. Значення за замовчуванням 0,5; Alpha Clip Threshold Depth Postpass - обмеження значення альфа-каналу, яке HDRP використовує для прозорого проходу глибини. Якщо значення альфа-каналу пікселя дорівнює або перевищує це обмеження, HDRP візуалізує піксель. Якщо значення нижче за обмеження, HDRP не відтворює піксель. Значення за замовчуванням 0,5; Alpha Clip Threshold Depth Prepass - обмеження значення альфа, яке HDRP використовує для попереднього проходу прозорої глибини. Якщо значення альфа-каналу пікселя дорівнює або перевищує це обмеження, HDRP візуалізує піксель. Якщо значення нижче за обмеження, HDRP не відтворює піксель. Значення за замовчуванням 0,5; Alpha Clip Threshold Shadow - обмеження значення альфа-каналу, яке використовує HDRP, щоб визначити, чи має відтворювати тіні для пікселя. Якщо значення альфа-каналу пікселя дорівнює або перевищує це обмеження, HDRP візуалізує піксель. Якщо значення нижче за обмеження, HDRP не відтворює піксель. Значення за замовчуванням 0,5; Depth Offset - значення, на яке шейдер збільшує глибину фрагмента. Для отримання реалістичного результату цей блок вимагає від вас введення результату вузла відображення паралаксної оклюзії; Distortion - екранний простір у кожному напрямку, який HDRP спотворює світло, яке проходить через матеріал. Наприклад, якщо встановити значення (1, 0), результат буде зміщено на 1 піксель праворуч; Distortion Blur - інтенсивність розмиття ефекту спотворення; Shadow Tint - колір тіні та непрозорість.

Для створеного Shader Graph створюється анімація для цього через Create

Node додано Twirl - ефект обертання зображення спотворює відтворене зображення в межах круглої області. Пікселі в центрі кола повертаються на заданий кут; поворот для інших пікселів у колі зменшується з відстанню від центру, зменшуючись до нуля на краю кола..

Create Node — це те, як створити вузли в Shader Graph. Він фільтрує доступні вузли, щоб показати лише ті, які використовують тип даних вибраного ребра. У ньому буде перераховано всі доступні порти на цих вузлах, які відповідають цьому типу даних.

Додано Voronoi - шум Вороного генерується шляхом обчислення відстані між пікселем і решіткою точок. Шляхом зсуву цих точок на псевдовипадкове число, кероване вхідним кутовим зміщенням, можна створити кластер комірок. Масштаб цих комірок і результуючий шум контролюються вхідною щільністю комірок. Вихідні клітинки містять необроблені дані клітинок. Також додано Time – параметр часу завдяки якому ефект водовороту стає рухомим.

У межах Shader Graph вузол Color дозволяє додавати колір до ефекту, використовувати його для зразків інших кольорів і багато іншого.

Вузол Sample Texture 2D робить вибірку ресурсу Texture 2D і повертає значення кольору Vector 4. Можна вказати UV-координати для зразка текстури та використовувати вузол Sampler State для визначення конкретного стану Sampler.

Вузли є основною частиною сценаріїв у Visual Scripting. Вузол може прослуховувати події, отримувати значення змінної, змінювати компонент GameObject тощо. Вузли відображаються як блоки в редакторі графіків. Тому всі вищевказані елементи поєднано між собою за допомогою вузлів (рис. 4.11).

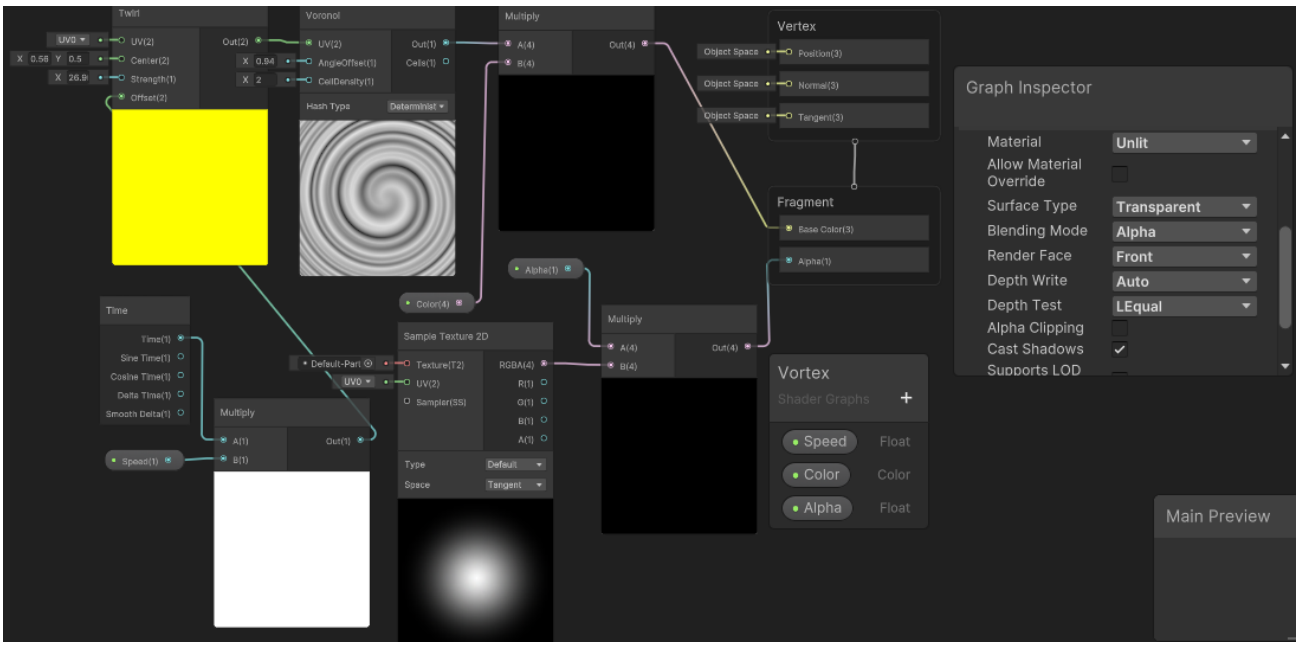


Рисунок 4.11 – Пов’язані компоненти Shader Graph вузлами

З налаштованого Shader Graph створено матеріал та присвоєно чотирикутнику.

Результатом є предмет, який виключає видимість об’єктів, що потрапили в середену колайдеру з відповідно указаним тегом, та візуальний ефект коловороту.

4.4.3 Створення об’єктів для першої сцени та налаштування їх взаємодії

Для створення інтерактивної взаємодії, на якій побудовано сюжет першої сцени потрібно створити палочку та три яблука. Під час доторкання паличкою до яблук, яблуко розділиться на три рівні части, що буде імітувати принцип множення. Щоб імітувати розділення яблука на три рівні частини під час доторкання до нього паличкою у віртуальному середовищі, можна використати наступний підхід:

а) Створення яблука та палички. Спочатку потрібно створити моделі яблука та палички, які будуть взаємодіяти у віртуальному середовищі.

б) Розробка скрипту у Unity. Створити скрипт, який призначений для обробки взаємодії між паличкою та яблуком. Цей скрипт має виявляти дотик паличкою до яблука та реагувати на цю подію.

в) Імітація розділення яблука. Коли паличка доторкається до яблука, скрипт має викликати операцію розділення яблука на три частини, що імітуватиме

принцип множення. Це можна виконати шляхом реалізації анімації або візуальних змін моделі яблука, яка розділиться на три частини під час взаємодії з паличкою.

г) Взаємодія з колайдерами та скриптами. Під час доторкання паличкою до яблука можна використовувати колайдери для виявлення цієї взаємодії. Скрипт, що відповідає за розділення яблука, буде активований, коли колайдер палички зіштовхується з колайдером яблука.

д) Відображення результату розділення. Після виявлення дотику паличкою до яблука та запуску процесу розділення, потрібно візуально показати, як яблуко розбивається на три частини.

Для створення палички, до проекту додано модель чарівної палочки з параметрами Rigibody для фізичних властивостей, Disable Grabbing Hand Model для того щоб моделі руки ставали невидимими під час тримання цього об’єкту та XR Grab Interactable, щоб можна було брати в руку предмет. До моделі додано залежний об’єкт Attach Transform, який потрібен в параметрі XR Grab Interactable - Attach Transform, щоб відкалібрувати позицію моделі під час тримання її в руках.

Створено новий скрипт Stick, який імітує невидимий лазер віртуальному середовищі, спрацьовує метод Break для об'єктів, які перетинаються з лазером на вказаній відстані distance (дод Б).

Для створення яблук, до проекту додано модель яблука Apple (пізніше остаточний результат буде розмножено методом копіювання) з параметрами Mesh Renderer, Mesh Colider, Rigibody, XR Grab Interactable, Disable Grabbing Hand Model. З такими ж параметрами створено залежна об’єкти яблук Apple1, Apple2, Apple3, за замовченням вони дезактивовані, тому що всі мають свій колайдер. Apple - це головний батьківський об’єкт, який буде зникати, а на його місці з’являться залежні Apple1, Apple2, Apple3.

Створено скрипт Breakable, який дезактивує об’єкт якому присвоєний, та активує указані об’єкти в параметрі bracable Pieces (рис. 4.11).

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;

public class Breakable : MonoBehaviour
{
    public List<GameObject> breakablePeices;
    public UnityEvent OnBreak;

    void Start()
    {
        foreach (var item in breakablePeices)
        {
            item.SetActive(false);
        }
    }
    public void Break()
    {
        foreach (var item in breakablePeices)
        {
            item.SetActive(true);
            item.transform.parent = null;
        }

        OnBreak.Invoke();
        gameObject.SetActive(false);
    }
}

```

Рисунок 4.11 – Скрипт Breakable для активації\дизактивації указаних об’єктів

4.4.4 Створення об’єктів для другої сцени та налаштування їх взаємодії

Для створення інтерактивної взаємодії, на якій побудовано сюжет другої сцени потрібно створити книги та область для ідентифікації потрібної. Під час піднесення книги до ідентифікаційної області, буде відображено кольоровий ідентифікатор в залежності від указаного тега. Якщо відпустити книгу на ідентифікаційною областю, то книга залишиться в застиглому стані. Для створення цього сценарію знадобиться:

а) Створення об'єктів та налаштування тегів. Створити книги та область для ідентифікації. Позначити кожну книгу тегом, що відповідає її статусу або категорії.

б) Створення скрипту для обробки взаємодії.

в) Візуальна ідентифікація книги. Під час підняття книги до ідентифікаційної області скрипт повинен змінювати колір або візуальні атрибути книги залежно від її тегу.

Для створення аналізуючої області створено чотирикутник, якому

присвоєно параметри Mesh Renderer, Box Collider. Додано залежні об’єкти Socket Interactor та Socket Interactor2, яким присвоєно параметри Sphere Collider та XRSocketTagInteractor.

XR Socket Interactor використовується для утримання інтерактивних елементів через сокет. Цей компонент не призначений для приєднання до контролера (отже, не походить від XRBaseControllerInteractor, на відміну від XRDirectInteractor і XRRayInteractor) і натомість завжди намагатиметься вибрати інтерактивний елемент, на який він наводить курсор (хоча не виконуватиме ексклюзивний вибір цього інтерактивного елемента). Socket визначається як ціль для конкретного взаємодіючого елемента.

XRSocketTagInteractor наслідує функціонал базового інтерактора від Unity XR Interaction Toolkit, додаючи умови для взаємодії лише з об'єктами, які мають певний тег (рис. 4.12).

```
1
2   using System.Collections;
3   using System.Collections.Generic;
4   using UnityEngine;
5   using UnityEngine.XR.Interaction.Toolkit;
6
7
8   public class XRSocketTagInteractor : XRSocketInteractor
9   {
10      public string targetTag;
11
12      public override bool CanHover(IXRHoverInteractable interactable)
13      {
14          return base.CanHover(interactable) && interactable.transform.tag == targetTag;
15      }
16
17      public override bool CanSelect(IXRSelectInteractable interactable)
18      {
19          return base.CanSelect(interactable) && interactable.transform.tag == targetTag;
20      }
21  }
```

Рисунок 4.12 – Скрипт XRSocketTagInteractor

В Hover Mesh Material встановлено колір сітки, який буде відображатись під час наведення книги. В Target Tag прописано тег на який буде впливати скрипт.

Для моделі книги додано параметри Mesh Renderer, Mesh Colider, Box Colider, Rigibody, XR Grab Interactable, Disable Grabbing Hand Model та проставлено тег.

4.4.5 Створення об’єктів для третьої сцени та налаштування їх взаємодії

Для створення інтерактивної взаємодії, на якій побудовано сюжет третьої сцени потрібно завантажити модель казанка в якого параметр Mesh Renderer та моделі трьох різних по розміру мішків. Після перенесення мішка до казана, мішок дезактивується, а в казанку буде додано монети в кількості залежній від розміру мішка. Для більш реалістичного відчуття взаємодія в третій сцені буде відбуватись без джойстика, тільки руками. Моделям мішків присвоєно параметр Mesh Renderer, Box Collider, Grabbable, Active Ob. Скрипт Active Ob (рис. 4.13) дезактивує моделі указані в braekablePieces, моделі монет, до моменту поки мішок стає дезактивованим, після чого монети активуються. Візуально це виглядає як при переміщенні мішка, мішок зникає, а символічні монети з’являються і падають в казан. Основна мета скрипту Grabbable - взаємодіяти з об'єктами у віртуальній реальності через Oculus SDK, надаючи можливість захоплення та трансформації цих об'єктів залежно від кількості точок захоплення та призначених трансформерів.

Для захоплення об’єктів руками, моделям MoneyBag додано залежні Visuals, HandGrabInteractable, HandGrabInteractable_mirror. Для Visuals присвоєно додано скрипт InteractableGroupView, цей код дозволяє об'єднувати та управляти групою інтерактивних об'єктів (в параметр Interactables додано HandGrabInteractable та HandGrabInteractable_mirror) у віртуальній реальності через Oculus SDK, визначати їхні стани та взаємодію з оточуючим середовищем.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;

public class ActiveOb : MonoBehaviour
{
    public List<GameObject> breakablePieces;
    public UnityEvent OnBreak;

    void Start()
    {
        SetBreakablePiecesActive(false); // Викликаємо метод для встановлення стану breakablePieces
    }

    private void OnDisable()
    {
        SetBreakablePiecesActive(true); // Викликаємо метод для встановлення стану breakablePieces
    }

    private void SetBreakablePiecesActive(bool active)
    {
        foreach (var piece in breakablePieces)
        {
            piece.SetActive(active);
        }
    }

    public void Break()
    {
        SetBreakablePiecesActive(true); // Включаємо всі частини при руйнуванні
        OnBreak.Invoke();

        gameObject.SetActive(false); // Деактивуємо об'єкт, до якого доданий цей скрипт
    }
}
```

Рисунок 4.13 – Скрипт Active Ob

HandGrabInteractable та HandGrabInteractable_mirror мають параметр Hand Grab Interactable, цей код дозволяє визначати параметри захоплення об'єкта рукою у віртуальній реальності через Oculus SDK у фреймворку Unity, включаючи різні типи захоплення, правила руху та позицій захоплення рук. Для наслідування Rigibody указано головний батьківський елемент MoneyBag, для типів захоплень указано залежні елементи HandGrab Point. Для HandGrab Point додано скрипти HandGrabPose - цей код визначає клас, який використовується для визначення точок захоплення (grip points) для рук у віртуальній реальності в середовищі Unity, особливо для взаємодії з об'єктами через Oculus SDK. Даний клас забезпечує логіку обчислення оптимальних позицій для рук користувача під час захоплення об'єктів; та CylinderGrabSurface – скрипт визначає поведінку поверхонь захоплення, специфічно циліндричних поверхонь, навколо об'єктів. Відрегулювання позиції HandGrabInteractable та HandGrabInteractable_mirror відбувається під час тестового запуску застосунку.

4.5 Додавання звуку в проект

Аудіофункції Unity включають повний тривимірний просторовий звук, мікшування та мастеринг у реальному часі, ієрархії мікшерів, знімки та попередньо визначені ефекти. Аудіосистема Unity може імпортувати більшість стандартних форматів аудіофайлів, відтворювати звуки в 3D-просторі та застосовувати додаткові ефекти, такі як відлуння та фільтрація. Щоб імітувати вплив положення, Unity вимагає, щоб звуки виходили з аудіоджерел, приєднаних до об’єктів. Потім видавані звуки вловлюються аудіослухувачем, прикріпленим до іншого об’єкта, найчастіше головної камери. Тоді Unity може симулювати ефекти відстані та положення джерела від об’єкта слухача та відтворювати їх. Unity не може обчислити відлуння виключно з геометрії сцени, але можна імітувати їх, додавши аудіофільтри до об’єктів.

Unity може імпортувати аудіофайли у форматах AIFF, WAV, MP3 та Ogg так само, як і інші ресурси перетягнувши файли на панель проекту. Імпортувати аудіофайл, щоб створити аудіокліп, який потім можна перетягнути до аудіоджерела або використати зі сценарію.

Для музики Unity також підтримує модулі трекерів, які використовують короткі звукові зразки, які можна організувати для відтворення мелодій. Можна імпортувати модулі трекера з файлів .xm, .mod, .it і .s3m і використовувати їх так само, як і інші аудіокліпи.

Unity може отримати доступ до мікрофонів комп’ютера за допомогою сценарію та створювати аудіокліпи шляхом прямого запису. Клас Microphone надає API для пошуку доступних мікрофонів, запиту їхніх можливостей, а також запуску та завершення сеансу запису[36].

Починаючи з Unity 5.0 аудіодані відокремлюються від фактичних аудіокліпів. Аудіокліпи лише посилаються на файли, що містять аудіодані, і в імпортері аудіокліпів є різні комбінації параметрів, які визначають, як кліпи завантажуються під час виконання.

Модулі відстеження — це, по суті, просто пакети зразків аудіо, які були

змодельовані, упорядковані та секвеновані програмно. Доріжки - це партитури, які містять інформацію про те, коли грати на інструментах, на якій висоті та гучності, і на основі цього можна відтворити мелодію та ритм оригінальної мелодії.

Unity Audio Mixer дозволяє мікшувати різні аудіоджерела, застосовувати до них ефекти та виконувати мастеринг. У вікні відображається аудіомікшер, який, по суті, є деревом груп аудіомікшерів. Група Audio Mixer — це, по суті, суміш аудіо, сигнальний ланцюг, який дозволяє застосовувати ослаблення гучності та корекцію висоти; він дозволяє вставляти ефекти, які обробляють звуковий сигнал, і змінювати параметри ефектів. Існує також механізм надсилання та повернення для передачі результатів з однієї шини на іншу.

Аудіокліпи містять аудіодані, які використовуються джерелами звуку. Unity підтримує моно-, стерео- та багатоканальні аудіоресурси до восьми каналів. Ресурси модуля трекера поводяться так само, як і будь-які інші аудіоресурси в Unity, хоча в інспекторі імпорту активів попередній перегляд форми сигналу недоступний. Параметри аудіо кліпів:

а) Force To Mono. Параметр увімкнено, багатоканальне аудіо буде мікшовано до монодоріжки перед упаковкою

б) Normalize. Звук буде нормалізовано під час процесу мікшування Force To Mono

в) Load In Background. Завантаження кліпу відбуватиметься із затримкою в окремому потоці, не блокуючи основний потік

г) Ambisonic. Джерела звуку Ambisonic зберігають звук у форматі, який представляє звукове поле, яке можна обертати залежно від орієнтації слухача. Це корисно для 360-градусних відео та програм XR. Увімкніть цей параметр, якщо ваш аудіофайл містить звук, закодований за допомогою Ambisonic.

Аудіослухач діє як пристрій, схожий на мікрофон. Він отримує вхідний сигнал від будь-якого аудіоджерела в сцені та відтворює звуки через динаміки комп’ютера. Для більшості програм доцільніше приєднати слухавку до головної

камери. Якщо аудіослухач знаходиться в межах зони реверберації, реверберація застосовується до всіх звукових сигналів сцени. Крім того, аудіоефекти можна застосувати до слухача, і вони будуть застосовані до всіх чутних звуків у сцені.

Джерело звуку відтворює аудіокліп у сцені. Кліп можна відтворювати на аудіослухачі або через аудіомікшер. Джерело аудіо може відтворювати будь-який тип аудіокліпів і може бути налаштований для відтворення у 2D, 3D або як суміш (SpatialBlend). Аудіо можна поширювати між динаміками (стерео до 7.1) (Spread) і трансформувати між 3D і 2D (SpatialBlend). Це можна контролювати на відстані за допомогою кривих падіння. Крім того, якщо слухач знаходиться в одній або кількох зонах реверберації, реверберація застосовується до джерела. Індивідуальні фільтри можна застосувати до кожного аудіоджерела для ще багатшого звуку. Є три режими спаду: логарифмічний, лінійний і спеціальний спад. Custom Rolloff можна змінити, змінивши криву відстані об’єму, як описано нижче. Якщо ви спробуєте змінити функцію об’ємної відстані, коли для неї встановлено значення Logarithmic або Linear, тип автоматично зміниться на Custom Rolloff. Існує кілька властивостей аудіо, які можна змінювати залежно від відстані між джерелом аудіо та слухачем аудіо:

а) Volume амплітуда (0,0 - 1,0) на відстані.

б) Spatial Blend. 2D (оригінальне відображення каналів) до 3D (усі канали мікшуються до моно та послаблюються відповідно до відстані та напрямку).

в) Spread кут (градуси 0,0 - 360,0) на відстань.

г) Low-Pass (тільки якщо LowPassFilter під’єднано до AudioSource). Частота зрізу (22000,0–10,0) на відстані.

д) Reverb Zone кількість сигналу, спрямованого в зони реверберації.

Джерела звуку нічого не роблять без призначеного аудіокліпу. Кліп — це фактичний звуковий файл, який буде відтворено. Джерело схоже на контролер для запуску та зупинки відтворення цього кліпу та зміни інших властивостей звуку.

Щоб створити нове джерело звуку потрібно імпортувати аудіофайли у

проект Unity. Тепер це аудіокліпи. Створити джерело звуку GameObject (GameObject - Audio - Audio Source). Вибравши новий GameObject, вибрати Component - Audio - Audio Source. В Inspector знайти властивість «Аудіокліп» у компоненті «Джерело аудіо» та призначте кліп, перетягнувши його з вікна проекту або клацнувши піктограму маленького кола праворуч від властивості Inspector, а потім вибравши кліп зі списку. Для проекту завантажено різноманітні звуки та присвоєно предметам або додано як результат якоїсь дії в часолінію, налаштовано параметри.

4.6 Створення Timeline для покращення взаємодії

Вікно Timeline використовується, щоб створювати кат-сцени, кінематографічні сюжети та ігрові послідовності, візуально впорядковуючи доріжки та кліпи, пов’язані з GameObjects у сцені. Для кожної кат-сцени, кінематографічної або ігрової послідовності у вікні часової шкали зберігається наступне:

а) Ресурс часової шкали: зберігає доріжки, кліпи та записані анімації без посилань на конкретні GameObjects, які анімуються. Ресурс часової шкали зберігається в проекті.

б) Екземпляр часової шкали: зберігає посилання на певні ігрові об’єкти, які анімуються або на які впливає актив часової шкали. Ці посилання, які називаються прив’язками, зберігаються в сцені.

Щоб анімувати GameObject у сцені за допомогою ресурсу Timeline, потрібно створити екземпляр Timeline. Екземпляр Timeline пов’язує ресурс Timeline з GameObject у Scene через компонент Playable Director. Коли обрати об’єкт GameObject у сцені, яка має компонент Playable Director, екземпляр Timeline відображається у вікні Timeline. Прив’язки відображаються у вікні шкали часу та в компоненті Playable Director (вікно Inspector). Компонент Playable Director зберігає зв’язок між екземпляром Timeline та ресурсом Timeline. Компонент Playable Director контролює, коли відтворюється примірник Timeline,

як примірник Timeline оновлює свій годинник і що відбувається, коли примірник Timeline завершує відтворення.

Щоб імпортувати пакет Timeline потрібно:

а) Розгорнути Timeline у вікні Package Manager.

б) Вибрати поточний встановлений пакет Timeline.

в) Або розгорнути Samples, або вибрати вкладку Samples, щоб відобразити список доступних зразків проектів шкали часу.

г) Імпортувати пакет. Якщо пакет вже імпортовано, повторно імпортувати. Коли повторно імпортувати пакет, будь-які зміни, які вносились до пакету, його активів, сцен і файлів, перезаписуються оригінальним семплом.

Для даного проекту було створено Timeline для кожної окремої сцени. Завдяки цьому зрозуміла мета перебування користувача в сцені, приклад часолінії можна побачити (рис. 4.14). Далі буде описано створення та компоненти часолінії першої сцени.

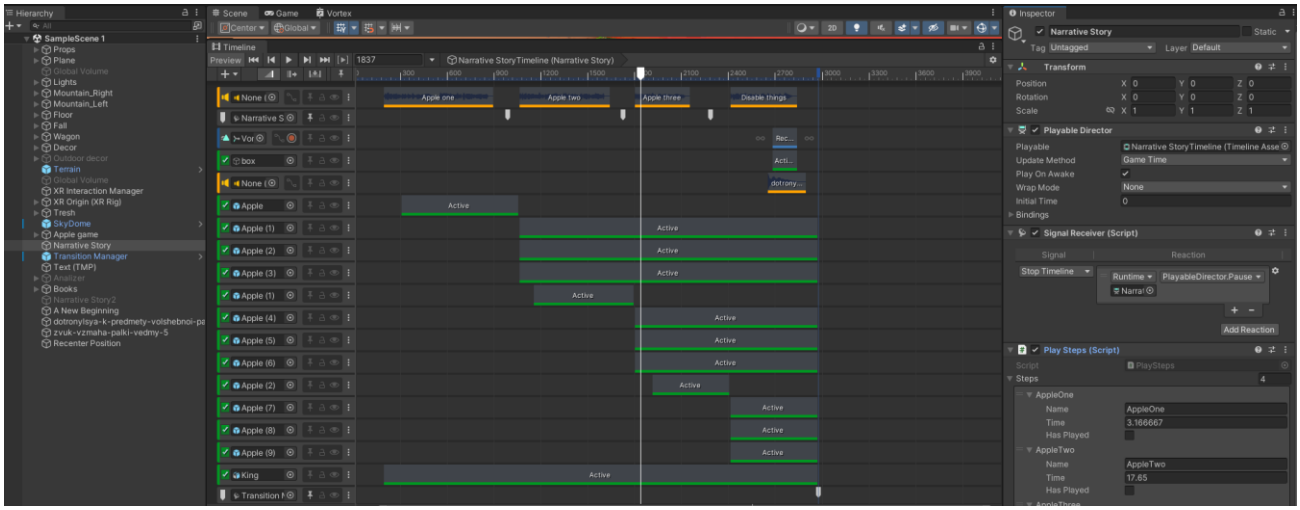


Рисунок 4.14 – Timeline першої сцени

Підготовлено звукові записи у форматі .wav того що відбувається в сцені і що потрібно зробити. Такі записи додано до часолінії як Audio Track та розміщено з часовими інтервалами. Перемішено об’єкти які мають з’явитись після або в процесі озвучання, це об’єкти яблук, палочки та ящика. Створено Animation Track для коловороту ящика. Animation Track - це частина системи анімацій в Unity, що використовується для керування анімаціями в таймлайні (Timeline). Треки анімацій дозволяють вбудовувати різноманітні анімації, які змінюють параметри

об'єктів у грі на протязі часу. Створювати комплексні анімаційні послідовності: додавати різні анімаційні кліпи до таймлайну і керувати їхнім часовим відтворенням та параметрами. Управляти параметрами об'єктів під час анімації: налаштовувати різні параметри об'єктів (такі як позиція, обертання, масштаб тощо) в процесі відтворення анімації. Синхронізувати анімації з іншими подіями: встановлювати час початку та завершення анімаційних кліпів у відповідності до інших подій в грі. Це дозволяє створювати ефект того що об’єкт повільно збільшується до необхідних розмірів, або переміщується під час відрізку часу.

Для того щоб сцена чекала дій користувача і тільки після цього продовжувала Timeline додано Signal Track, які будуть запускати кожен наступний запис .wav або після завершення завдання запускає Transition Manager, який повертає користувача в головне меню. Signal Track - це один із типів треків у системі таймлайнів (Timeline) Unity. Він використовується для відправлення сигналів (повідомлень) у певні моменти часу під час відтворення або редагування таймлайну. Основна мета Signal Track полягає в тому, щоб сповістити інші скрипти, компоненти або системи в грі про певні події часової лінії. Коли поточний час таймлайну досягає певного позначеного моменту, Signal Track може відправити сигнал, який буде сприйнятий іншими об'єктами або скриптами.

Створено скрипт PlaySteps (рис. 4.15). Цей код управляє відтворенням різних кроків часової лінії в Unity. У ньому визначено клас PlaySteps, який містить список об'єктів класу Step. Кожен об'єкт Step має ім'я, час та прапорець, який вказує, чи відтворений вже цей крок. Метод PlayStepIndex(int index) відтворює крок за його індексом у списку. Якщо крок ще не був відтворений (hasPlayed дорівнює false), то встановлюється прапорець hasPlayed в true, зупиняється відтворення поточного PlayableDirector, встановлюється час на вказаний час кроку та запускається відтворення. Головна мета цього коду - керувати послідовним відтворенням різних кроків у грі чи анімації в Unity.

Для кожного об’єкта з яким має взаємодіяти користувач після чого має продовжитись Timeline, додано в параметр скрипт PlaySteps та проставлено індекс в залежності від часу. Подібним чином створено часолінії для інших сцен.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Playables;

public class PlaySteps : MonoBehaviour
{
    PlayableDirector director;
    public List<Step> steps;

    void Start()
    {
        director = GetComponent<PlayableDirector>();
    }
    [System.Serializable]

    public class Step
    {
        public string name;
        public float time;
        public bool hasPlayed = false;
    }

    public void PlayStepIndex(int index)
    {
        Step step = steps[index];
        if (!step.hasPlayed)
        {
            step.hasPlayed = true;

            director.Stop();
            director.time = step.time;
            director.Play();

        }
    }

}
```

Рисунок 4.15 – Скрипт PlaySteps

4.7 Створення області з кнопками головного меню та налаштування переходу до сцен

Для проекту додано головну сцену з меню, що є місцем, де користувач може отримати доступ до сцен. Додано панель меню, де є список сцен до яких можна перейти та кнопка виходу з застосунку.

Створено головний батьківський об’єкт Game Menu UI, який має компонент Rect Transform, що є двовимірним аналогом компонента Transform. Якщо Transform представляє одну точку, Rect Transform представляє прямокутник, усередині якого можна розмістити елемент інтерфейсу користувача. Якщо батько Rect Transform також є Rect Transform, дочірнє Rect Transform також може

вказати, як воно має бути розташоване та розмір відносно батьківського прямокутника. Також має параметр Canvas - це область, усередині якої мають бути всі елементи інтерфейсу. Canvas — це ігровий об’єкт із компонентом Canvas на ньому, і всі елементи інтерфейсу користувача мають бути дочірніми для такого Canvas. Додано компонент Canvas Scaler використовується для керування загальним масштабом і щільністю пікселів елементів інтерфейсу користувача в Canvas. Це масштабування впливає на все, що знаходиться під полотном, включаючи розмір шрифту та рамки зображення.

Має скрипт TrackedDeviceGraphicRaycaster представляє собою реалізацію спеціального GraphicRaycaster для використання з XR Interaction Toolkit у Unity. Його основна мета - виконувати променеве лиття (ray casting) на Canvas за допомогою трекованого пристрою в XR. Основні елементи коду це Tracked Device Graphic Raycaster - клас, який успадковує Base Raycaster та реалізує деякі специфічні для XR Interaction Toolkit методи. Perform Raycasts - метод, який виконує променеве лиття на Canvas залежно від даних, які надаються в Pointer Event Data. Sorted Spherecast Graphics та Sorted Raycast Graphics - методи, що визначають як графічні об'єкти на Canvas будуть перевірятися для зіткнень за допомогою променевого лиття. Вони сортують графічні об'єкти в порядку віддаленості та виконують перевірку для знаходження перетину променя з областями Rect Transform. Process Sorted Hits Results - метод, який обробляє результати променевого лиття та формує список об'єктів Raycast Result для подальшого використання. Цей код є складовою частиною підтримки просторових інтерфейсів користувача в середовищі XR, дозволяючи виконувати взаємодію з графічними об'єктами на Canvas за допомогою трекованих пристроїв.

Створено скрипт GameStartMenu (дод В). Основна його мета - керування меню гри та переходом між різними сторінками цього меню. В Main Menu Buttons зроблено зв’язок між об’єктами кнопок відповідно до назв функцій коду (рис. 4.16).

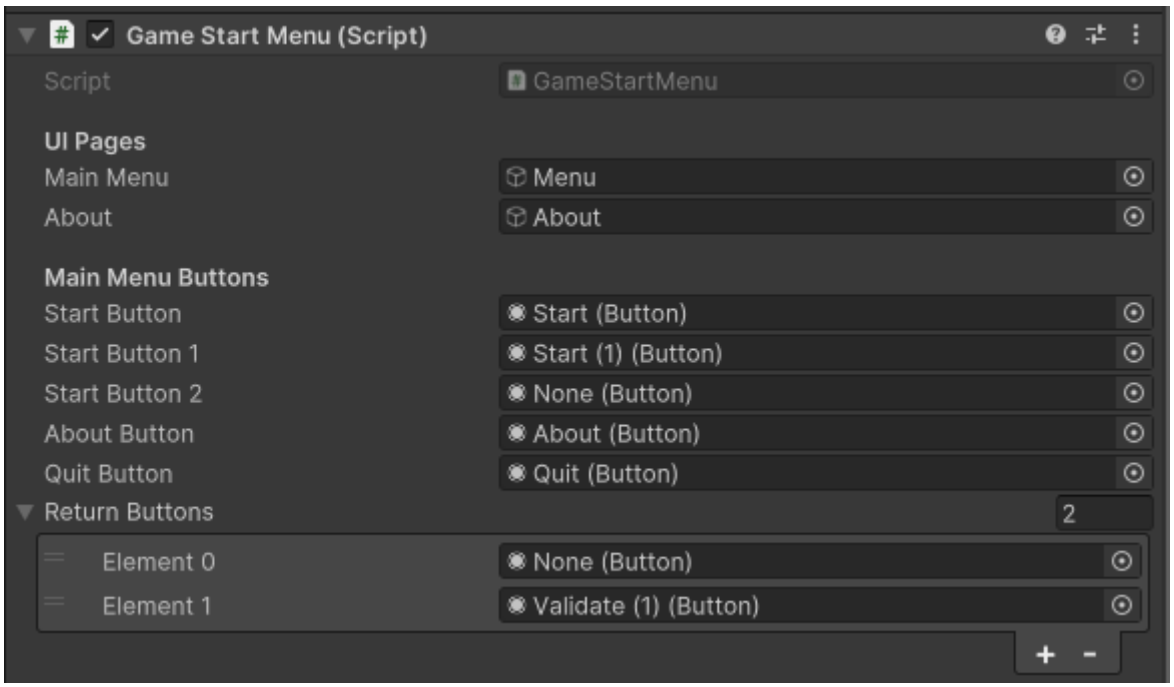


Рисунок 4.16 – Параметри GameStartMenu

В батьківському компоненті Game Menu UI знаходиться об’єкт Panel, який має власний Rect Transform. Також має Canvas Renderer - відтворює графічний об’єкт інтерфейсу користувача, що міститься в Canvas та не має властивостей, представлених в інспекторі.

Наступний елемент це Menu, який має лише Rect Transform. Menu це батьківський компонент для дочірніх Start, Start (1), Start (2), що ще кнопками для інтерактивну переходу до інших сцен. Кожна з цих кнопок має параметри Rect Transform, Canvas Renderer, Image, Button, Start, UIAudio.

Елемент Image відображає неінтерактивне зображення для користувача. Він подібний до елемента керування Raw Image, але пропонує більше можливостей для анімації зображення та точного заповнення прямокутника елемента керування. Однак вимагає, щоб його текстура була спрайтом, тоді як Raw Image може приймати будь-яку текстуру.

Елемент керування Button реагує на натискання користувача та використовується для ініціювання або підтвердження дії.

UIAudio використовується для програвання аудіо відповідно до різних дій користувача на UI-елементах в грі, коли відбуваються певні події. OnPointerClick: Метод, який викликається при кліку на UI-елементі. Якщо clickAudioName не

порожній, відтворюється звук, який відповідає цьому події. OnPointerEnter: Викликається при наведенні курсора на UI-елемент. Якщо hoverEnterAudioName не порожній, відтворюється аудіо для події наведення курсора на елемент. OnPointerExit: Викликається при відведенні курсора від UI-елементу. Якщо hoverExitAudioName не порожній, відтворюється аудіо для події відведення курсора від елементу. Отже, цей код дозволяє додавати звукові ефекти до взаємодії користувача з UI-елементами, наприклад, звук кліку при натисканні кнопки, звук при наведенні курсора на кнопку та звук при відведенні курсора від кнопки.

Останній компонент Plane, який має Rect Transform, Mesh Renderer, Mesh Collider. В налаштуваннях встановлено напівпрозорий білий колір заднього фону.

Для кожного з елементів Start, Start (1), Start (2) додано залежний компонент TextMeshPro, що використовує розширені методи рендеринга тексту разом із набором спеціальних шейдерів; забезпечуючи суттєві покращення візуальної якості, одночасно надаючи користувачам неймовірну гнучкість, коли справа доходить до стилізації та текстурування тексту. TextMeshPro забезпечує покращений контроль над форматуванням і макетом тексту за допомогою таких функцій, як символи, слова, інтервали між рядками та абзацами, кернінг, вирівняний текст, посилання, доступні понад 30 тегів Rich Text, підтримка кількох шрифтів і спрайтів, користувацькі стилі тощо. Оскільки геометрія, створена TextMeshPro, використовує два трикутники на символ, як і текстові компоненти Unity, ця покращена візуальна якість і гнучкість забезпечуються без додаткових витрат на продуктивність.

ВИСНОВКИ

У магістерській роботі проаналізовано платформи, механізми та технології віртуальної реальності та розроблено проект для реалізації механізмів взаємодії. Розробку інформаційної системи було виконано у середовищі Unity з додатково написаними скриптами у Microsoft Visual Studio 2019 при використанні мови програмування C #. Дана інформаційна система повинна значно полегшити процес навчання учнів, зробити його цікавішим та доступнішим візуально та практично для розуміння з можливістю відтворення наглядних прикладів. У розробленому продукту реалізовано такі функціональні можливості:

а) брати предмети;

б) переміщати предмети;

в) кидати предмети;

г) видаляти предмети за рахунок переміщення їх у сеціально відведене для цього місце;

д) збільшувати кількість моделей яблук;

е) переміщатись в просторі кімнати.

У першій другій та третій частині роботи було розглянуте поняття віртуальної реальності та необхідно можливих компонентів та принципів роботи. Виділено основні платформи для розробки проектів та описано можливості та принципи роботи. Було проведено аналіз найвідоміших гарнітур віртуальної реальності, описано переваги та недоліки кожної для пошуку оптимального варіанту для використання.

В четвертій частині здійснено розробку проекту віртуальної реальності з описом елементів платформи Unity, які використовувались під час розробки. В результаті проведеної роботи розроблено різноманітні механізми взаємодії користувача з віртальними елементами чи моделями, додано можливість взаємодії через джойстик чи використовуючи руки.1

ПЕРЕЛІК ПОСИЛАНЬ

1. Virtual reality | wikipedia.org. URL: https://en.wikipedia.org/wiki/Virtual_reality.
2. Імерсивні технології | wikipedia.org. URL: https://uk.wikipedia.org/wiki/Імерсивні_технології.
3. Virtual reality education. Affective, Interactive and Cognitive Methods for E-Learning Design. Creating an Optimal education Experience, Christou.C, pp. 228–243(2010).
4. Паралакс | wikipedia.org. URL: https://uk.wikipedia.org/wiki/Паралакс.
5. Дисторсія | wikipedia.org. URL: https://uk.wikipedia.org/wiki/Дисторсія.
6. Virtual Reality, VR. IT-Enterprise – your one-stop ecosystem for digital transformation | it.ua. URL: https://www.it.ua/knowledge-base/technology-innovation/virtualnaja-realnost-vr.
7. Інерційний вимірювальний пристрій Unity (game engine) | wikipedia.org. URL: https://uk.wikipedia.org/wiki/Інерційний_вимірювальний_пристрій.
8. Best VR Controller in 2023, Level up your VR experience with the best VR controllers | pcguide.com. URL: https://www.pcguide.com/vr/guide/best-controller/
9. Logitech, G333 VR Gaming Earphones | logitechg.com. URL: https://www.logitechg.com/en-us/products/gaming-audio/g333-vr-headphones.981-001002.html
10. Dolby Atmos is making virtual reality incredibly realistic | techradar.com. URL: https://www.techradar.com/news/wearables/dolby-atmos-is-making-virtual-reality-incredibly-realistic-1286268
11. VIVE, Audio SDK | vive.com. URL: https://hub.vive.com/storage/3dsp/
12. Wireless Headset | playstation.com. URL: https://www.playstation.com/en-us/accessories/pulse-3d-wireless-headset/
13. Unity (game engine), Unity (game engine) | wikipedia.org. URL: https://en.wikipedia.org/wiki/Unity_(game_engine)
14. XR Interaction Toolkit | unity3d.com. URL:

https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.3/manual/index.html

15. VR development in Unity | unity3d.com. URL: https://docs.unity3d.com/Manual/VROverview.html
16. Input System | unity3d.com. URL: https://docs.unity3d.com/Packages/com.unity.inputsystem@1.5/manual/index.html
17. XR Hands | unity3d.com. URL: https://docs.unity3d.com/Packages/com.unity.xr.hands@1.2/manual/index.html
18. Audio Spatializers | unity3d.com. URL: https://docs.unity3d.com/Manual/VRAudioSpatializer.html
19. Lumen in UE5: Let there be light | unrealengine.com. URL: https://www.unrealengine.com/en-US/blog/lumen-in-ue5-let-there-be-light
20. Virtual Shadow Maps | unrealengine.com. URL: https://docs.unrealengine.com/5.0/en-US/virtual-shadow-maps-in-unreal-engine/
21. Blueprints Visual Scripting | unrealengine.com. URL: https://docs.unrealengine.com/5.3/en-US/blueprints-visual-scripting-in-unreal-engine/
22. Gizmos | unrealengine.com. URL: https://docs.unrealengine.com/5.0/en-US/transforming-actors-in-vr/
23. VR Mode Radial Menu | unrealengine.com. URL: https://docs.unrealengine.com/5.0/en-US/vr-mode-radial-menu/
24. Get started with Meta Quest 2 | meta.com. URL: https://www.meta.com/quest/products/quest-2/tech-specs/#tech-specs
25. Vive | vive.com. URL: https://www.vive.com/us/product/vive-flow/overview/
26. Vr | playstation.com. URL: https://www.playstation.com/en-us/ps-vr/tech-specs/
27. Headset | valvesoftware.com. URL: https://www.valvesoftware.com/en/index/headset
28. About the XR Plug-in Management package | unity3d.com. URL: https://docs.unity3d.com/Packages/com.unity.xr.management@3.2/manual/index.html
29. Input in Unity OpenXR | unity3d.com. URL: https://docs.unity3d.com/Packages/com.unity.xr.openxr@1.8/manual/input.html

30. STYLIZED Fantasy Armory - Low Poly 3D Art | unity.com. URL: https://assetstore.unity.com/packages/3d/environments/fantasy/stylized-fantasy-armory-low-poly-3d-art-249203
31. Transforms | unity3d.com. URL: https://docs.unity3d.com/2019.4/Documentation/Manual/class-Transform.html
32. Types of light | unity3d.com. URL: https://docs.unity3d.com/Manual/Lighting.html
33. Oculus Hand Models | oculus.com. URL: https://developer.oculus.com/downloads/package/oculus-hand-models/?locale=en_GB
34. Set Up Hand Tracking | oculus.com. URL: https://developer.oculus.com/documentation/unity/unity-handtracking/
35. Rigidbody | oculus.com. URL: https://docs.unity3d.com/2021.2/Documentation/Manual/class-Rigidbody.html
36. Audio overview | unity3d.com. URL: https://docs.unity3d.com/Manual/AudioOverview.html

ДОДАТОК А

Скрипт призначений для управління моделями рук у віртуальній реальності, перемикаючи їх видимість залежно від взаємодій.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR;
using UnityEngine.XR.Interaction.Toolkit;

public class DisableGrabbingHandModel : MonoBehaviour
{
   // Поля для моделей рук
   public GameObject leftHandModel; // Модель лівої руки
   public GameObject rightHandModel; // Модель правої руки

   void Start()
   {
      // Отримання посилання на компонент взаємодії з об'єктами віртуальної
реальності
      XRGrabInteractable grabInteractable = GetComponent<XRGrabInteractable>();

      // Додання слухачів подій для взаємодії з об'єктами
      grabInteractable.selectEntered.AddListener(HideGrabbingHand); //
Приховування руки при взаємодії з об'єктом
      grabInteractable.selectExited.AddListener(ShowGrabbingHand); // Показ руки
після взаємодії з об'єктом
   }

   // Метод для приховування моделі руки при взаємодії з об'єктом
   public void HideGrabbingHand(SelectEnterEventArgs args)
   {
      // Перевірка тегу об'єкта взаємодії
      if(args.interactorObject.transform.tag == "Left Hand") // Якщо взаємодія з лівою
рукою
      {
         leftHandModel.SetActive(false); // Приховати модель лівої руки
      }
      else if(args.interactorObject.transform.tag == "Right Hand") // Якщо взаємодія з
правою рукою
      {
         rightHandModel.SetActive(false); // Приховати модель правої руки
      }
   }
```

```
    // Метод для показу моделі руки після виходу з взаємодії з об'єктом
    public void ShowGrabbingHand(SelectExitEventArgs args)
    {
        // Перевірка тегу об'єкта взаємодії
        if (args.interactorObject.transform.tag == "Left Hand") // Якщо виходить з лівої
руки
        {
            leftHandModel.SetActive(true); // Показати модель лівої руки
        }
        else if (args.interactorObject.transform.tag == "Right Hand") // Якщо виходить з
правої руки
        {
            rightHandModel.SetActive(true); // Показати модель правої руки
        }
    }

    void Update()
    {
        // Порожній метод Update, який викликається кожен кадр у грі
    }
}
```

ДОДАТОК Б

Скрипт імітує лазер, спрацьовує метод Break, якщо об'єкти перетинаються з лазером на вказаній відстані.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR;
using UnityEngine.XR.Interaction.Toolkit;

public class Stick : MonoBehaviour
{
    // Параметри для використання луча (ray)
    public LayerMask layerMask; // Шар, на якому працює луч
    public Transform shootSource; // Точка виходу луча
    public float distance = 10; // Максимальна відстань луча

    private bool rayActivate = false; // Прапорець для активації луча

    // Метод, який викликається при запуску скрипту
    void Start()
    {
        // Отримання компонента для взаємодії з об'єктами віртуальної реальності
        XRGrabInteractable grabInteractable = GetComponent<XRGrabInteractable>();

        // Додання слухачів подій для початку та завершення взаємодії з об'єктом
        grabInteractable.selectEntered.AddListener(x => StartShoot()); // Початок
взаємодії
        grabInteractable.selectExited.AddListener(x => StopShoot()); // Завершення
взаємодії
    }

    // Метод для активації луча при взаємодії з об'єктом
    public void StartShoot()
    {
        rayActivate = true; // Встановлення прапорця для активації луча
    }

    // Метод для деактивації луча при завершенні взаємодії з об'єктом
    public void StopShoot()
    {
        rayActivate = false; // Скидання прапорця для деактивації луча
    }
```

```
    // Метод, який викликається кожен кадр у грі
    void Update()
    {
        if (rayActivate) // Якщо луч активований
            CheckForTouch(); // Перевірка на дотик об'єктів
    }

    // Метод для перевірки на дотик об'єктів лучем
    void CheckForTouch()
    {
        // Отримання всіх колайдерів, які перетинають луч на вказаній відстані та
шарі
        Collider[] colliders = Physics.OverlapSphere(shootSource.position, 0.1f,
layerMask);

        // Обхід усіх знайдених колайдерів
        foreach (Collider collider in colliders)
        {
            // Відправка повідомлення "Break" до об'єкта, що зіткнувся з лучем
            collider.gameObject.SendMessage("Break",
SendMessageOptions.DontRequireReceiver);
        }
    }
}
```

ДОДАТОК В

Скрипт що відповідає за управління стартовим меню гри

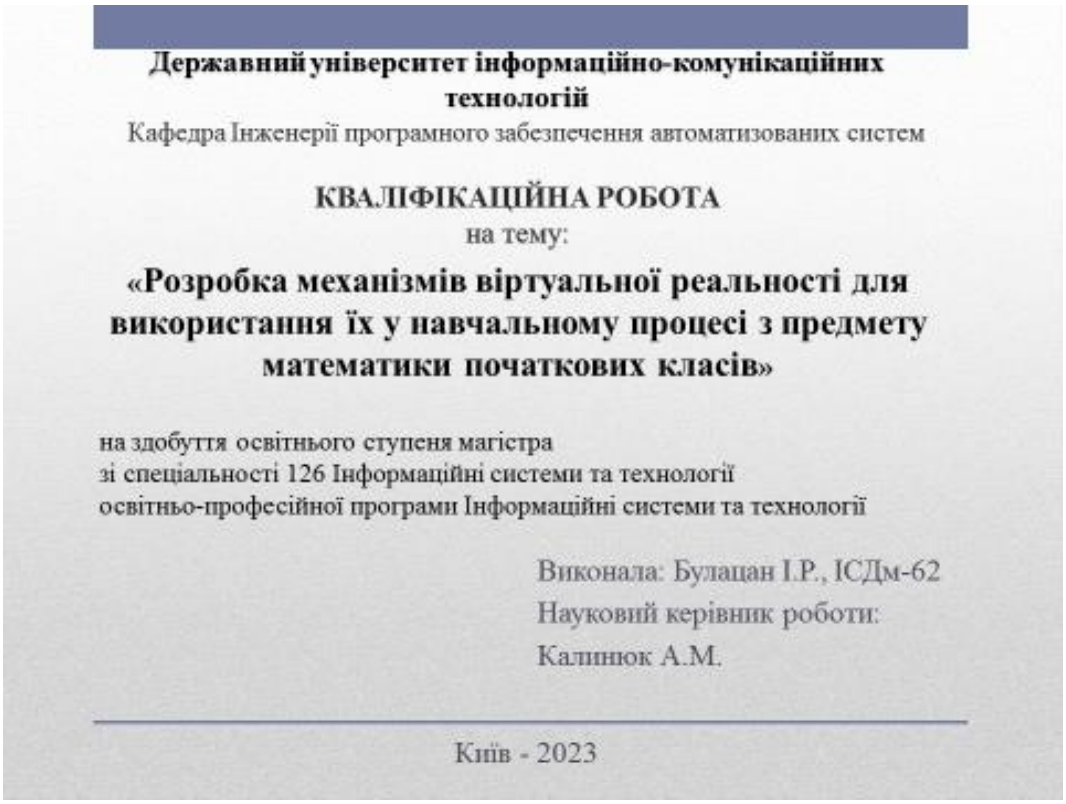
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class GameStartMenu : MonoBehaviour
{
    [Header("UI Pages")]
    public GameObject mainMenu; // Головне меню
//    public GameObject options; // Меню опцій
    public GameObject about; // Меню "Про гру"

    [Header("Main Menu Buttons")]
    public Button startButton; // Кнопка старту гри 1
    public Button startButton1; // Кнопка старту гри 2
    public Button startButton2; // Кнопка старту гри 3
//    public Button optionButton; // Кнопка опцій
    public Button aboutButton; // Кнопка "Про гру"
    public Button quitButton; // Кнопка виходу з гри

    public List<Button> returnButtons; // Список кнопок повернення до головного меню

    // Метод, який викликається при запуску скрипту
    void Start()
    {
        EnableMainMenu(); // Встановлення головного меню в початковий стан

        // Прив'язка подій до кнопок
        startButton.onClick.AddListener(StartGame); // Обробка події початку гри 1
        startButton1.onClick.AddListener(StartGame1); // Обробка події початку гри 2
        startButton2.onClick.AddListener(StartGame2); // Обробка події початку гри 3
//        optionButton.onClick.AddListener(EnableOption); // Обробка події відкриття меню опцій
        aboutButton.onClick.AddListener(EnableAbout); // Обробка події відкриття меню "Про гру"
        quitButton.onClick.AddListener(QuitGame); // Обробка події виходу з гри

        // Прив'язка подій до кнопок повернення до головного меню
        foreach (var item in returnButtons)
        {
```

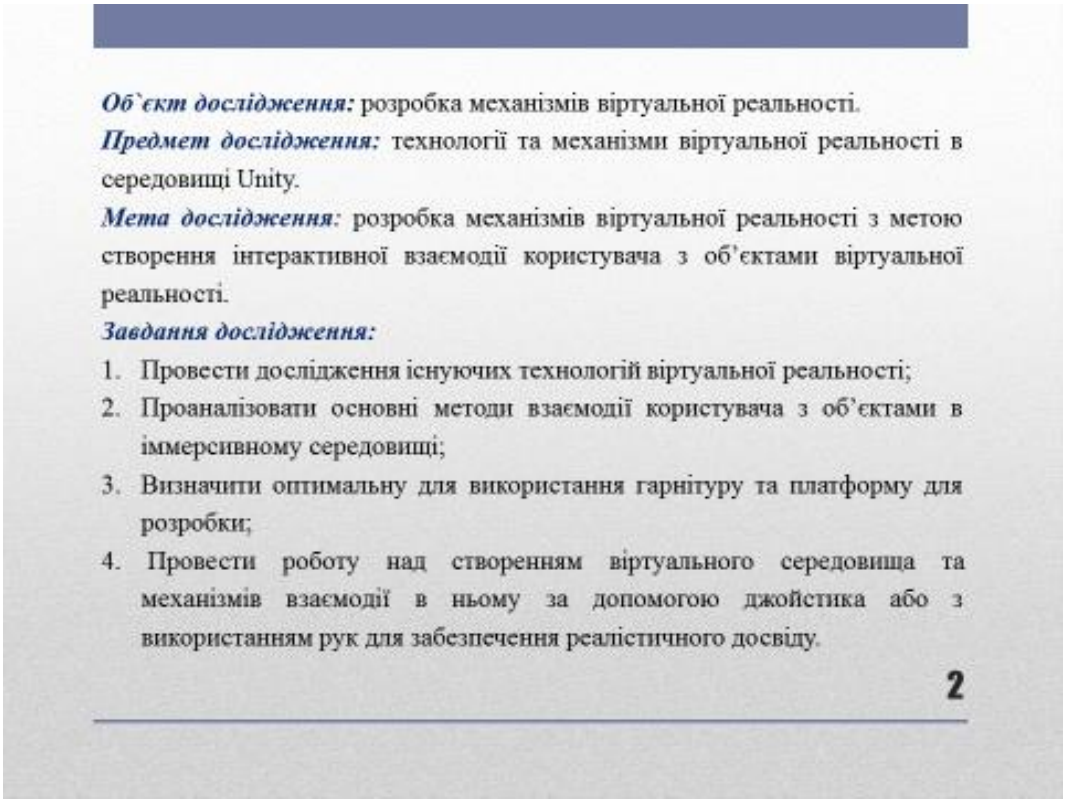
```
            item.onClick.AddListener(EnableMainMenu);
        }
    }

    // Метод для виходу з гри
    public void QuitGame()
    {
        Application.Quit();
    }

    // Методи для початку гри з вибраним номером
    public void StartGame()
    {
        HideAll(); // Приховання всіх меню
        SceneTransitionManager.singleton.GoToSceneAsync(1); // Запуск гри 1
    }

    public void StartGame1()
    {
        HideAll(); // Приховання всіх меню
        SceneTransitionManager.singleton.GoToSceneAsync(2); // Запуск гри 2
    }

    public void StartGame2()
    {
        HideAll(); // Приховання всіх меню
        SceneTransitionManager.singleton.GoToSceneAsync(3); // Запуск гри 3
    }

    // Метод для приховання всіх меню
    public void HideAll()
    {
        mainMenu.SetActive(false); // Приховання головного меню
//      options.SetActive(false); // Приховання меню опцій
        about.SetActive(false); // Приховання меню "Про гру"
    }

    // Метод для відображення головного меню
    public void EnableMainMenu()
    {
        mainMenu.SetActive(true); // Відображення головного меню
//      options.SetActive(false); // Приховання меню опцій
        about.SetActive(false); // Приховання меню "Про гру"
    }
```

```
    // Метод для відображення меню "Про гру"
    public void EnableAbout()
    {
        mainMenu.SetActive(false); // Приховання головного меню
//        options.SetActive(false); // Приховання меню опцій
        about.SetActive(true); // Відображення меню "Про гру"
    }
}
```

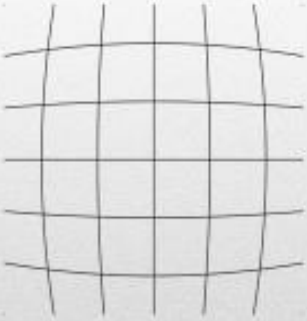
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)





Гарнітура для відображання



Бочкоподібна дисторсія

Два дисплеї, в яких зображення трохи зміщене одне відносно іншого

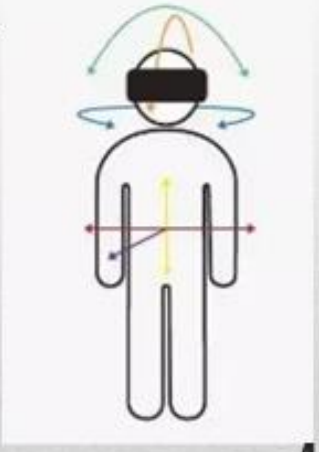
3

Обладнання для відстеження руху

Акселерометр
для визначення швидкості та орієнтації пристрою

Гіроскоп
для визначення зміни в орієнтації

Магнітометр
вимірює силу та напрямок магнітного поля



4

Пристрої керування

Рукавичка
- дротова
- бездротова

Джойстик
- дротовий
- бездротовий

Рука
- датчики зовнішні
- датчики в гарнітурі

5

Технології відтворення звуку

Звукові бандажі/навушники розташовані навколо голови користувача та надають аудіофайли через динаміки, які розташовані біля вух

Звукові системи вбудовані можуть виробляти просторовий звук із відтворенням напрямленості

6

Технології відтворення звуку

Звукові акустичні динаміки
розташовані в околицях користувача, щоб створити просторовий звук

Динаміки
розташовані навколо користувача окремо-стоячі або прикріплені до стін

7

Порівняльні дані якості зображення

	Роздільна здатність, пікселів	Поле зору, градусів	Оптика. Тип лінз	Оптика. Діаметр лінз	Оптика. Міжзіркова відстань	Частота оновлення
Oculus Quest 2	1832 x 1920	~ 100	асферичні лінзи	~ 40 мм	Ручне регулювання	90 Гц
HTC Vive	1080x1200	~ 110	асферичні лінзи	~ 28 мм	Ручне регулювання	90 Гц (можливо підвищити до 120 Гц)
PlayStation VR	960x1080	~ 100	асферичні лінзи	~ 36 мм	Ручне регулювання	90 Гц
Valve Index	1440x1600	~ 130	асферичні лінзи	~ 50 мм	Ручне регулювання	144 Гц

Valve Index має найкращі показники по полю зору, діаметр лінз, частота оновлення.

8

Порівняльні дані трекінгу

Зовнішній трекінг більш точний і надійний, але вимагає встановлення базових станцій.

Внутрішній трекінг зручний тому що не вимагає встановлювати додаткове обладнання, але може бути менш точним.

	Тип трекінгу	Кількість камер	Кількість базових станцій	Контролери
Oculus Quest 2	Внутрішній	чотири вбудовані камери	-	використовують внутрішній трекінг
HTC Vive	Зовнішній	-	2	використовують внутрішній трекінг і синхронізуються з базовими станціями
PlayStation VR	Зовнішній	-	використовує PlayStation Camera для відстежування руху користувача та контролерів	використовують внутрішній трекінг і синхронізуються з PlayStation Camera
Valve Index	Зовнішній	-	2	використовують внутрішній трекінг і синхронізуються з базовими станціями

9

Порівняльні дані автономності

	Автономність	Розмір	Потрібні додаткові прилади (додаткова вага)
Oculus Quest 2	Так	224 x 450 мм	Ні
HTC Vive	Ні	155,5 x 200,7 мм	Так
PlayStation VR	Ні	187×277 мм	Так
Valve Index	Ні	179,7 x 135,9 мм	Так

Автономні гарнітури, зазвичай **найбільш мобільні**, оскільки вони не потребують підключення до комп'ютера або консолі.

Гарнітури зовнішнього трекінгу, можуть надавати разючий імерсивний досвід, але вони **менш мобільні** через прив'язку до додаткового обладнання і проводів.

10

Меню

Відео 0

- Управління через джойстик.
- Можна обрати потрібний пункт меню та перейти в сцену або вийти з застосунку.

11

Сцена 1.

Відео 1

- Управління через джойстик.
- Можна взяти предмети, підкинути їх, схопити у повітрі, перемістити, штовхнути.
- Завдання полягає у наглядному прикладі множення і ділення. Потрібно взяти палочку та розщепити по черзі три яблука.

12

Сцена 1.

Приклад скриптів розробленених для сцени.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class TrashCan : MonoBehaviour
{
    private void Start()
    {
        GetComponent<TriggerZone>().OnEnterEvent.AddListener(InsideTrash);
    }
    public void InsideTrash(GameObject go)
    {
        go.SetActive(false);
    }
}
```

Скрипт Trash Can

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;

public class TriggerZone : MonoBehaviour
{
    public string targetTag;
    public UnityEvent<GameObject> OnEnterEvent;

    private void OnTriggerEnter(Collider other)
    {
        if(other.gameObject.tag == targetTag)
        {
            OnEnterEvent.Invoke(other.gameObject);
        }
    }
}
```

Скрипт Trigger Zone

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;

public class Breakable : MonoBehaviour
{
    public List<GameObject> breakablePeices;
    public UnityEvent OnBreak;

    void Start()
    {
        foreach (var item in breakablePeices)
        {
            item.SetActive(false);
        }
    }
    public void Break()
    {
        foreach (var item in breakablePeices)
        {
            item.SetActive(true);
            item.transform.parent = null;
        }

        OnBreak.Invoke();
        gameObject.SetActive(false);
    }
}
```

Скрипт Breakable для активації\дизактивації указаних об'єктів

13

Сцена 2.

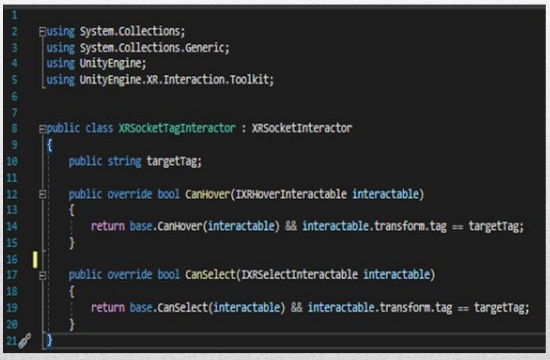
Відео 2

- Управління через джойстик.
- Можна взяти предмети, підкинути їх, схопити у повітрі, перемістити, штовхнути.
- Завдання полягає у розпізнаванні фігур. Потрібно взяти книгу з відповідною фігурою та перемістити у область визначення. Якщо фігура вірна – колір зелений, якщо не вірна – червоний.

14

Сцена 2.

Приклад скрипту розробленого для сцени.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR.Interaction.Toolkit;


public class XRSocketTagInteractor : XRSocketInteractor
{
    public string targetTag;

    public override bool CanHover(IXRHoverInteractable interactable)
    {
        return base.CanHover(interactable) && interactable.transform.tag == targetTag;
    }

    public override bool CanSelect(IXRSelectInteractable interactable)
    {
        return base.CanSelect(interactable) && interactable.transform.tag == targetTag;
    }
}
```

Скрипт XRSocketTagInteractor

15

Сцена 3.

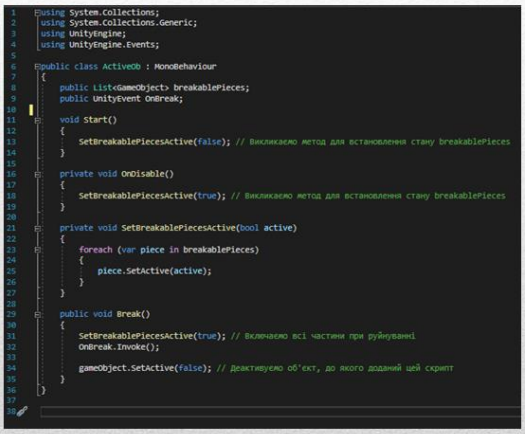
Відео 3

- Управління через джойстик або руками.
- Можна взяти предмети, підкинути їх, схопити у повітрі, перемістити, штовхнути.
- Завдання полягає у наглядному відтворенні заповнення обєму. Потрібно взяти мішок (задумо що з монетами) та перемістити в казан, після чого він заповниться.

16

Сцена 3.

Приклад скрипту розробленого для сцени.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;

public class ActiveOb : MonoBehaviour
{
    public List<GameObject> breakablePieces;
    public UnityEvent OnBreak;

    void Start()
    {
        SetBreakablePiecesActive(false); // Викликаємо метод для встановлення стану breakablePieces
    }

    private void OnDisable()
    {
        SetBreakablePiecesActive(true); // Викликаємо метод для встановлення стану breakablePieces
    }

    private void SetBreakablePiecesActive(bool active)
    {
        foreach (var piece in breakablePieces)
        {
            piece.SetActive(active);
        }
    }

    public void Break()
    {
        SetBreakablePiecesActive(true); // Включаємо всі частини при руйнуванні
        OnBreak.Invoke();

        gameObject.SetActive(false); // Деактивуємо об'єкт, до якого доданий цей скрипт
    }
}
```

Скрипт Active Ob

17

Висновки

У магістерській роботі проаналізовано платформи, механізми та технології віртуальної реальності та розроблено проект для реалізації механізмів взаємодії.
Розробку інформаційної системи було виконано у середовищі Unity з додатково написаними скриптами у Microsoft Visual Studio 2019 при використанні мови програмування C #.
Реалізовано функціональні можливості: брати, переміщати, кидати, видаляти предмети, збільшувати кількість моделей яблук, переміщатись в просторі кімнати.

Робота пройшла апробацію.
Стаття на тему: «Дослідження принципів та механізмів віртуальної реальності та використання їх у навчальному процесі з предмету математики початкових класів»
Теза на тему: «Переваги та недоліки використання технологій віртуальної реальності в процесі навчання»

18