Міністерство освіти і науки України
Тернопільський національний педагогічний університет
імені Володимира Гнатюка

Фізико-математичний факультет
Кафедра інформатики та методики її навчання

Кваліфікаційна робота

ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ІНТЕГРОВАНОГО 3D-СИМУЛЯТОРА НА ОСНОВІ СУЧАСНИХ ТЕХНОЛОГІЙ ГЕЙМДИЗАЙНУ

Спеціальність 122 Комп'ютерні науки
Освітня програма «Комп'ютерні науки»

Здобувача вищої освіти освітньо-кваліфікаційного рівня «магістр»
Джуги Дениса Євгенійовича

НАУКОВИЙ КЕРІВНИК:
кандидат фізико-математичних наук,
доцент
Мартинюк Сергій Володимирович

РЕЦЕНЗЕНТ:
кандидат технічних наук,
доцент кафедри кібербезпеки,
декан ФКІТ Західноукраїнського
національного університету
Якименко Ігор Зіновійович

Тернопіль – 2025

АНОТАЦІЯ

Джуга Д. Є. Проєктування та реалізація інтегрованого 3D-симулятора на основі сучасних технологій геймдизайну. Кваліфікаційна робота на здобуття освітнього ступеня «магістр» зі спеціальності 122 Комп'ютерні науки. ТНПУ ім. В. Гнатюка. Тернопіль, 2025. 62 с.

У кваліфікаційній роботі досліджено технології створення 3D-ігрових симуляторів із використанням відкритих програмних засобів Godot Engine та Blender. Проаналізовано сучасні методи й інструменти розробки ігрових проєктів, їх переваги та особливості, зокрема функціонал Godot для реалізації фізики, анімації та взаємодії в 3D-просторі, а також можливості Blender у створенні тривимірних моделей і анімаційного контенту. У практичній частині розроблено прототип 3D-гри-симулятора, який забезпечує інтуїтивне управління, реалістичну фізику та різні ігрові режими. Робота обґрунтовує використання відкритого програмного забезпечення і популяризації сучасних технологій у галузі комп'ютерних наук та освіти.

Ключові слова: 3D гра-симулятор, Godot Engine, Blender, розробка ігор, тривимірна графіка, ігрові технології, відкритий код.

ABSTRACT

Dzhuha D. Ye. Design and implementation of an integrated 3D-simulator based on modern game design technologies. Master's thesis for the MS degree in the specialty 122 Computer Science. Ternopil Volodymyr Hnatiuk National Pedagogical University. Ternopil, 2024. Ternopil, 2025. 62 p.

The master's thesis explores technologies for creating 3D-game simulators using open-source software Godot Engine and Blender. It analyzes modern methods and tools for game project development, their advantages and features, including the functionality of Godot for implementing physics, animation, and interaction in 3D-space, as well as Blender's capabilities for creating three-dimensional models and animated content. The practical part features the development of a prototype 3D-game simulator that offers intuitive control, realistic physics, and multiple game modes. The work justifies the use of open-source software to reduce costs and promote modern technologies in computer science and education.

Keywords: 3D game-simulator, Godot Engine, Blender, game development, 3D-graphics, gaming technologies, open-source.

ЗМІСТ

ВСТУП

У сучасному світі ігрова індустрія займає велику частку ринку інформаційних технологій, стрімко розвиваючись та охоплюючи все ширшу аудиторію. Постійно зростаючий попит користувачів на якісні та сучасні ігрові продукти стимулює розробників до впровадження нових технологій і підходів у процесі створення ігрових проєктів. 3D ігри-симулятори, які надають гравцеві змогу заглибитися у віртуальні світи, забезпечивши при цьому реалістичність, посідають особливе місце серед таких продуктів.

Одним із сучасних програмних продуктів для розробки ігор є рушій Godot Engine, який виділяється поміж аналогів своїм відкритим кодом, легкістю у використанні, зручним та інтуїтивно зрозумілим інтерфейсом та підтримкою написання програмного коду кількома мовами, такими як GDScript, C# та C++. Завдячуючи своїй кросплатформеності та великій активній спільноті розробників з усього світу, Godot здобуває все більшу популярність серед невеликих ігрових студій та інді-розробників. У свою чергу Blender є потужним редактором для створення та редагування тривимірної графіки, що дозволяє моделювати деталізовані об’єкти, персонажі тощо для своїх проєктів. Також Blender має вбудовані інструменти для створення анімаційних сцен і візуальних ефектів. Поєднання можливостей Blender для створення ігрового контенту та Godot для реалізації взаємодії дає можливість створювати сучасні 3D ігри-симулятори.

Актуальність розробки 3D гри-симулятора з використанням Godot та Blender зумовлена кількома чинниками. По-перше, затребуваність на ігрові симулятори в сучасних умовах постійно зростає, оскільки вони не тільки здатні розважати гравців, а й тренувати навички, навчати та моделювати ситуації. наближені до реальності. Зокрема, симулятори різних видів використовують в освіті, авіації, медицині, військовій справі й інших галузях. По-друге, поєднання можливостей Godot та Blender дозволяє звести до мінімуму витрати на розробку, так як обидва пакети програмного забезпечення є безкоштовними та мають відкритий код. По-третє, використання сучасних інструментів забезпечує

популяризацію відкритого програмного забезпечення та вчить початківців працювати з гнучкими й універсальними середовищами розробки та редакторами. Отже, розробка 3D гри-симулятора із застосуванням Godot та Blender є актуальною як для задоволення потреби в сучасних симуляторах, так і для вдосконалення навичок розробників у використанні відкритих інструментів для створення тривимірного контенту й ігрових проєктів.

Метою дослідження є розробка 3D гри-симулятора із використанням ігрового рушія Godot та інструмента моделювання Blender, що забезпечуватиме наближену до реальності фізику, інтуїтивне управління та різноманітні режими гри.

Встановлена мета обумовлює такі **завдання**:

1. Проаналізувати сучасні методи й інструменти для розробки ігрового проєкту.

2. Дослідити функціонал і можливості Godot та Blender для розробки 3D гри-симулятора.

3. Розробити ігровий проєкт з використанням Godot та Blender.

Об'єкт дослідження – процес створення 3D гри-симулятора.

Предмет дослідження – методи й інструменти розробки 3D гри-симулятора з використання сучасного програмного забезпечення Godot Engine та Blender.

Для досягнення поставленої мети було використано такі методи аналізу: теоретичний аналіз літературних джерел з розробки ігрових проєктів; порівняльний аналіз можливостей сучасних ігрових рушіїв; проєктування та розробка ігрових механік і графіки; тестування розробленого симулятора на практиці.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків і списку використаних джерел.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНИХ МЕТОДІВ ТА ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ ІГРОВОГО ПРОЄКТУ

1.1 Вплив вибору інструментів розробки для створення гри

Вибір інструментів розробки є одним із ключових факторів, що визначає ефективність створення ігрового проєкту. Від правильного підбору рушія, мов програмування та додаткового програмного забезпечення залежить якість кінцевого продукту, час розробки та ресурси, необхідні для реалізації ідеї.

Основні аспекти впливу вибору інструментів:

1. Продуктивність і оптимізація.

Обраний рушій має підтримувати цільові платформи, забезпечувати високу продуктивність і можливість оптимізації для слабших пристроїв. Наприклад, рушії на кшталт Unreal Engine надають фотореалістичну графіку, але вимагають потужного обладнання. Godot натомість є більш легким і дозволяє створювати оптимізовані 3D-ігри навіть на середньому обладнанні.

2. Гнучкість і масштабованість.

Якщо гра планується як невеликий інді-проєкт, доцільно обирати рушій, який дозволяє швидко реалізувати ідею без складного налаштування (наприклад, Godot або Unity). Для великих проєктів із реалістичною графікою та складними механіками краще підійде Unreal Engine, оскільки він пропонує більше інструментів для оптимізації та масштабування.

3. Вартість розробки.

Використання платних рушіїв та інструментів збільшує загальну вартість проєкту. Безкоштовні та open-source рішення, такі як Godot та Blender, значно зменшують витрати, що є важливим для інді-розробників і студентських проєктів [30].

4. Зручність навчання та використання.

Важливим фактором є наявність зрозумілого інтерфейсу, навчальних матеріалів і документації. Наприклад, Godot пропонує інтуїтивну структуру

проєктів на основі сцен і вузлів, що спрощує процес навчання для новачків. Blender, у свою чергу, є універсальним інструментом для моделювання, але вимагає більше часу для опанування.

5. Сумісність із додатковими інструментами.

Ігровий рушій має підтримувати імпорт моделей з популярних редакторів, таких як Blender, 3ds Max чи Maya. Наприклад, Godot добре інтегрується з Blender, що дозволяє швидко переносити 3D-моделі й анімації у проєкт.

6. Спільнота та підтримка.

Активна спільнота розробників полегшує процес вирішення проблем, а також надає готові плагіни та ресурси. Чим більша база користувачів, тим більше прикладів і рішень доступно. Unreal Engine та Unity мають найбільші спільноти, але Godot стрімко набирає популярність завдяки своїй простоті та відкритому коду.

Правильний вибір інструментів визначає не лише якість гри, а й ефективність усього процесу розробки. Для створення 3D-симулятора з мінімальними витратами доцільно використовувати Godot Engine як рушій та Blender для моделювання, оскільки ці інструменти забезпечують баланс між функціональністю, простотою, продуктивністю та безкоштовністю.

1.2 Аналіз сучасних ігрових рушіїв для розробки 3D-ігор

Проведемо аналіз найвідоміших сучасних ігрових рушіїв, зокрема таких, як Unreal Engine, Unity, Godot. Кожен з цих рушіїв має свої унікальні можливості та функціонал, що робить їх привабливими для проєктів різних типів. Звернемо увагу на такі параметри:

- основний функціонал та характеристики;
- переваги та недоліки;
- приклади успішних ігор, створених з їх допомогою.

Почнемо з гіганта гейм-індустрії Unreal Engine.

Unreal Engine – це ігровий рушій, розроблений компанією Epic Games. Був розроблений як середовище розробки шутерів від першої особи, натомість зараз це багатофункціональний інструмент для створення реалістичних ігрових

проєктів різних жанрів, візуалізації архітектурних проєктів і кіновиробництва. Завоював прихильність серед розробників завдяки реалістичній графіці, гнучкості в написанні коду й іншим потужним інструментам.

Ключовими можливостями Unreal Engine є:

- програмування ігрових механік, використовуючи власну візуальну мову програмування Blueprints для швидкого прототипування;
- технологія віртуалізованої геометрії Nanite, яка забезпечує використання високополігональних 3D об’єктів без втрати продуктивності;
- динамічне глобальне освітлення ігрових сцен Lumen в реальному часі;
- інструмент для створення реалістичних людських персонажів MetaHuman Creator;
- підтримка експорту проєктів для різних платформ: Windows, macOS, Linux, консолі, iOS та Android.

Перевагами Unreal Engine над іншими ігровими рушіями є:

- фотореалістична графіка;
- потужні інструменти для розробки проєктів високого рівня;
- широка спільнота розробників;
- велика кількість навчальних матеріалів як в платній, так і вільно поширюваній версіях;
- гнучкість у програмуванні ігрової логіки з використання C++ та Blueprints;
- можливість використовувати програмне забезпечення у повному обсязі безкоштовно, поки дохід від проєкту не досягне 1 мільйона доларів.

Основні мінуси рушія:

- високі системні вимоги;
- великі розміри файлів і проєктів;
- складність у навчання для початківців, особливо у C++.

За допомогою Unreal Engine було розроблено багато різноманітних проєктів, починаючи від атмосферних пригодницьких платформерів, на зразок

Little Nightmares (рис. 1.1) до багатокористувацьких гігантів ігрової індустрії таких як PlayerUnknown's BattleGrounds (рис. 1.2) [1].

Рис. 1.1. Little Nightmares

Рис. 1.2. PlayerUnknown's BattleGrounds

Наступним проаналізуємо не менш відомий ігровий рушій Unity. Unity – це відомий багатоплатформний рушій, розроблений компанією Unity Technologies. Використовують для створення як малих, так і великих 2D і 3D ігор, додатків віртуальної та доповненої реальності. Підтримує мову програмування C# та побудову ігрової логіки, використовуючи візуальне програмування Visual Scripting, що забезпечує зручність як для новачків, так і для досвідчених розробників [4].

Особливості Unity:

- наявність рендерингу URP та HDRP з високою продуктивністю або фотореалістичністю;
- можливість розширювати власні проєкти за допомогою сторонніх пакетів з Unity Asset Store або з відкритих джерел;
- швидке прототипування проєктів з використання візуальної мови програмування Visual Scripting;
- підтримка експорту ігор і додатків на понад 25 платформ, включаючи ПК, мобільні пристрої, консолі та VR/AR.

Переваги рушія:

- велика спільнота розробників;
- багато відкритих і платних матеріалів для навчання;
- відносно невеликий поріг входу для новачків;
- активна підтримка від розробників;

- можливість реалізовувати ігрову логіку, використовуючи як C#, так і Visual Scripting;
- безкоштовний, допоки річний дохід проєкту не досягне відмітки в 100 тисяч доларів.

Мінуси Unity:

- обмеження в продуктивності для реалізації великих ігрових проєктів AAA-рівня;
- незважаючи на безкоштовність рушія, деякі важливі функції та можливості доступні лише після оформлення платних підписок;
- складність у налаштуванні деяких потрібних у розробці інструментів для початківців.

У середовищі розробки Unity було створено багато цікавих малих і великих проєктів, починаючи з стилізованої гри на соціальну дедукцію Among Us (рис. 1.3), закінчуючи проєктом Genshin Impact з великим відкритим світом і гарною графікою (рис. 1.4).

Рис. 1.3. Among Us

Рис. 1.4. Genshin Impact

Проаналізуємо ще один з відомих ігрових рушіїв – Godot.

Godot Engine – безкоштовний багатофункціональний рушій з відкритим кодом, створений аргентинськими розробниками Хуаном Лінетським і Аріелем Манзуром, зараз розробляється співтовариством Godot Engine Community. Зазвичай використовують для створення невеликих 2D та 3D ігрових проєктів [6].

Основні особливості Godot:

- гра складається з сцен, які в свою чергу складаються з вузлів, або інших сцен, що спрощує структуру всього проєкту;

- підтримує кілька мов програмування, такі як: C#, GDScript (мова спеціально створена для Godot).
- підтримує експорт на платформи Windows, macOS, Linux, Android, iOS, Web і консолі;
- можливість модифікації середовища розробки під власні потреби за допомогою інструмента GDExtension.

Плюси Godot Engine:

- малі апаратні вимоги;
- інтуїтивно зрозумілий інтерфейс;
- має власну мову програмування GDScript, яка є досить легкою для опанування її новачками;
- швидке прототипування завдяки системі сцен і GDScript;
- маленький розмір скомпільованих проєктів, що ідеально підходить для мобільних або просто невеликих проєктів;
- розповсюджується за ліцензією MIT, яка дозволяє змінювати рушій і користуватися без обмежень [12].

Мінуси рушія:

- менша продуктивність порівняно з Unreal Engine та Unity;
- менша спільнота та кількість плагінів, ніж у популярнчших рушіях;
- не підходить для розробки великих AAA-проєктів через недостатню підтримку реалістичної графіки.

Хоча більшість користувачів Godot інді-розробники, але це не є перешкодою для розкриття його потенціалу. Наведемо кілька ігор, вартих уваги: пригодницька гра з елементами головоломки – Deponia Doomsday (рис. 5), фантастичний бойовик-симулятор Planeten verteidigungs kanonen kommandant (PVKK) (рис. 1.6).

Рис. 1.5. Deponia Doomsday

Рис. 1.6. PVKK

1.3 Дослідження програмного забезпечення для роботи з 3D-графікою

Жодна команда розробників при створенні повноцінної 3D гри не може обійтися без програми для 3D-моделювання, оскільки графіка є не менш важливим аспектом гри, ніж продуманий сценарій або ігрові механіки. Дослідимо можливості, переваги та недоліки 3D редакторів, а також наведемо кілька прикладів ігрових проєктів, у яких їх застосовували.

Першим дослідимо інструмент від компанії розробників програмного забезпечення Autodesk – 3ds Max.

3ds Max – це потужний пакет інструментів для 3D-моделювання, анімації, рендерингу, та візуалізації. Він широко використовується індустрії ігор, кіновиробництві, дизайні інтер’єрів та візуалізації архітектурних проєктів. 3ds Max є основним інструментом для роботи з тривимірною графікою у багатьох компаніях завдяки гнучким інструментам для моделювання, ефективним рендер-рушіям та інтеграції з іншими продуктами від Autodesk.

Опишемо основний функціонал програми 3ds Max:

- **3D-моделювання**: підтримує різних видів моделювання, таких як полігональне, NURBS та сплайн-моделювання. Має потужний модифікаторний стек для недеструктивного моделювання, та інструменти ProBoolean і ProCutter для редагування твердих тіл [12];
- **анімація**: наявна ефективна система кісток та морфінгу. Містить інструменти MassFX для процедурної анімації та фізики, також CAT та Biped для швидкої анімації персонажів;

- **рендеринг**: Має вбудовані рендер-рушії Arnold, Scanline та ART, а також підтримує використання сторонніх інструментів V-Ray, Corona та інших;
- **візуалізація**: підтримує реалістичні матеріали PBR та освітлення GI, HDRI, IBL;
- **скриптинг**: має вбудовану мову сценаріїв для автоматизації різних задач і підтримує розширення функціоналу та інтеграції з сторонніми програмами з допомогою Python;
- **інтеграція**: легко інтегрується з іншими програмами Autodesk: Revit, AutoCAD та Maya.
- **експорт**: підтримує велику кількість форматів файлів для експорту, такі як FBX, OBJ, STL, DWG тощо.

Переваги 3ds Max:

- обширний інструментарій для деталізованого 3D-моделювання;
- висока якість рендеру, завдяки підтримці передових рушіїв для рендеру;
- інтеграція з екосистемою програм Autodesk;
- велика кількість плагінів для покращення робочого процесу.

Недоліки програми:

- складений інтерфейс і багатофункціональність роблять його важким у вивченні для початківців;
- працює тільки на операційній системі Windows;
- дорога ліцензія, що робить його недоступним для більшості інді-розробників;
- потребує потужного апаратного обладнання для роботи з складними проєктами.

3ds Max використовувався для створення 3D-моделей персонажів та оточення, у найрізноманітніших проєктах, таких як серія пригодницьких ігор Assassin’s Creed (рис. 1.7) та в серії шутерів від першої особи Call of Duty (рис. 1.8).

Рис. 1.7. Assassin's Creed

Рис. 1.8. Call of Duty

Другим популярним інструментом для створення тривимірної графіки є Autodesk Maya.

Maya – це один із найпотужніших професійних пакетів для 3D-моделювання, анімації, рендерингу та симуляцій, який широко використовується у кіноіндустрії, геймдеві та візуалізації. Maya є стандартом для багатьох великих студій, завдяки своїй гнучкості, великій кількості інструментів і підтримці передових технологій.

Основні можливості Maya:

- **3D-моделювання**: Maya підтримує полігональне NURBS та сплайн-моделювання, а також має розширений набір інструментів для скульптингу. Є можливість застосування процедурних методів моделювання для оптимізації робочого процесу;
- **анімація**: Містить потужний набір засобів для створення анімацій персонажів: система кісток, інверсна кінематика, інструменти для ріггінгу та морфінгу. Додатково є інструменти для процедурної анімації та симуляцій тканин, волосся і динаміки рідин;
- **рендеринг**: вбудований рушій Arnold забезпечує високоякісний фізично коректний рендеринг, також підтримується інтеграція зі сторонніми рушіями, такими як V-Ray та Redshift [11];
- **скриптинг та автоматизація**: Maya підтримує мови MEL (Maya Embedded Language) та Python, що дозволяє автоматизувати робочі процеси та розширювати функціонал програми;

- **інтеграція**: легко поєднується з іншими продуктами Autodesk та популярними ігровими рушіями (Unreal Engine, Unity), що робить її універсальним інструментом для розробників ігор та аніматорів.
- **експорт**: підтримує стандартні формати для ігрових рушіїв (FBX, OBJ, Alembic), що дозволяє безперешкодно переносити моделі й анімації.

Переваги Maya:

- великий набір інструментів для моделювання й анімації на професійному рівні;
- розвинена система ріггінгу та процедурних симуляцій;
- висока якість рендерингу завдяки Arnold;
- широкі можливості автоматизації та кастомізації.

Недоліки Maya:

- висока вартість ліцензії, що робить програму недоступною для більшості інді-розробників;
- складний інтерфейс і значний час на освоєння;
- високі системні вимоги, особливо при роботі зі складними сценами.

Maya активно використовують у створенні високобюджетних ігор і фільмів. Наприклад, вона застосовувалася при розробці таких проєктів, як The Last of Us Part II (рис. 1.9) та Spider Man Remastered (рис. 1.10).

Рис. 1.9. The Last of Us Part II

Рис. 1.10. Spider Man Remastered

Третім інструментом, який ми розглянемо, є Blender.

Blender – це безкоштовний і відкритий програмний комплекс для роботи з 3D-графікою, який включає повний набір інструментів для моделювання, анімації, рендерингу, створення візуальних ефектів і відеомонтажу. Завдяки своїй

універсальності, активній спільноті та постійному розвитку, Blender є одним із найпопулярніших рішень серед інді-розробників та фрилансерів.

Основні можливості Blender:

- **3D-моделювання**: підтримує полігональне та скульптурне моделювання, має інструменти для створення складних форм та органічних об’єктів, присутні модифікатори для процедурного моделювання;
- **анімація**: наявна система ріггінгу, кісткових структур, морфінгу, а також вбудований рушій для симуляцій тканин, рідин, волосся та частинок;
- **рендеринг**: Blender має два вбудовані рендер-рушії – Cycles (фізично коректний трасувальник променів) та Eevee (реальний час, підходить для прев’ю й ігор) [2];
- **візуальні ефекти та композитинг**: інтегрований нодовий редактор для створення складних матеріалів, шейдерів та постобробки;
- **відеомонтаж**: Blender містить базовий нелінійний редактор для роботи з відео;
- **скриптинг**: підтримка мови Python для створення аддонів, автоматизації та кастомізації робочих процесів;
- **інтеграція з ігровими рушіями**: підтримує формати FBX, OBJ, GLTF, що дозволяє легко переносити моделі в Godot, Unity та Unreal Engine.

Переваги Blender:

- повністю безкоштовний і з відкритим кодом (ліцензія GNU GPL);
- постійне оновлення та розвиток за рахунок активної спільноти;
- широкий набір інструментів, що дозволяє виконувати весь цикл створення 3D-контенту в одній програмі;
- велика кількість безкоштовних навчальних матеріалів та плагінів.

Недоліки Blender:

- вимагає значного часу для освоєння через багатий функціонал;
- у деяких випадках поступається професійним платним продуктам (наприклад, у сфері VFX або кінопроєктів високого рівня);

- при роботі зі складними сценами може спостерігатися зниження продуктивності.

Blender використовують у створенні як інді-проєктів, так і в професійних студіях. Його застосовували для розробки ігор Sonic Colors: Ultimate (рис. 1.11) та No Man's Sky (рис. 1.12), а також для створення анімаційних короткометражок від Blender Foundation.

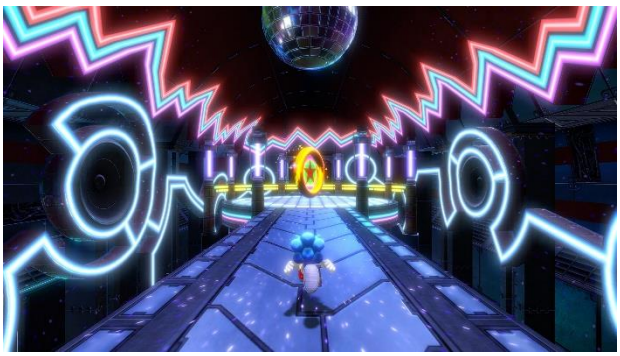

Рис. 1.11. Sonic Colors: Ultimate

Рис. 1.12. No Man's Sky

1.4 Огляд популярних аналогів симуляторів

У сучасній ігровій індустрії симулятори займають особливе місце, оскільки вони дозволяють користувачам занурюватися у максимально реалістичні умови та відтворювати різноманітні сценарії з реального життя. Такі ігри не лише виконують розважальну функцію, а й застосовуються у навчанні, тренуваннях і навіть професійній підготовці. Ринок симуляторів охоплює широкий спектр напрямів - від авіації та автоперегонів до управління складними механізмами чи транспортом. У цьому підрозділі буде розглянуто декілька популярних прикладів симуляторів, які стали еталонними у своїх категоріях і слугують орієнтиром для створення нових проєктів, зокрема симулятора польоту дронів.

Liftoff: FPV Drone Racing (рис. 1.13) – один із найвідоміших симуляторів польоту дронів, розроблений бельгійською студією LuGus Studios. Гра орієнтована на початківців і професійних пілотів FPV-дронів, забезпечуючи реалістичну фізику польоту та велику кількість налаштувань.

Рис. 1.13. Відображення геймплею Liftoff

Основні особливості:

- реалістична модель польоту, що враховує вагу дрону, характеристики моторів і пропелерів;
- великий вибір готових дронів і можливість їх кастомізації;
- різноманітні карти: від відкритих просторів до міських трас;
- підтримка мультиплеєра, що дозволяє змагатися з іншими гравцями;
- інтеграція з реальними пультами управління (наприклад, FrSky).

Переваги:

- висока реалістичність фізики;
- широкі можливості налаштувань дронів;
- велика спільнота гравців і модифікацій;
- можливість тренуватися перед реальними польотами.

Недоліки:

- високі системні вимоги;
- обмежений вибір середовищ у базовій версії (решта через DLC).

Liftoff є одним із найбільш збалансованих симуляторів польоту дронів, що поєднує реалістичність та ігрову складову. Він підходить як для навчання новачків, так і для тренувань професійних пілотів FPV-дронів.

DRL Simulator (рис. 1.14) – це офіційний симулятор від Drone Racing League, однієї з найбільших міжнародних організацій з перегонів дронів. Він

створений як для розважального використання, так і для професійної підготовки пілотів, які беруть участь у реальних чемпіонатах DRL.

Рис. 1.14. Відображення геймплею DRL Simulator

Основні особливості:

- реалістична фізика польоту, що відтворює поведінку FPV-дронів на високих швидкостях;
- офіційні траси DRL, а також можливість створення власних трас;
- підтримка реальних пультів керування (Taranis, Spektrum та інші);
- режим кар’єри з поступовим ускладненням завдань;
- мультиплеєр з онлайн-турнірами та рейтингами.

Переваги:

- точне відтворення характеристик перегонових дронів;
- наявність офіційного кіберспортивного компоненту;
- можливість відбору до реальних змагань через гру;
- великий вибір трас і сценаріїв.

Недоліки:

- орієнтація переважно на перегонові сценарії, що обмежує універсальність;
- вища складність для новачків порівняно з іншими симуляторами;
- доволі висока ціна порівняно з аналогами.

DRL Simulator можна вважати найбільш професійним симулятором для підготовки пілотів дронів. Він поєднує в собі реалістичність польоту та

кіберспортивну складову, що робить його популярним як серед геймерів, так і реальних спортсменів.

Microsoft Flight Simulator (рис. 1.15) – один з найвідоміших і найстаріших авіаційних симуляторів, розроблений компанією Microsoft. Його перша версія з'явилася ще в 1982 році, і з того часу проєкт постійно вдосконалюється, задаючи стандарт реалістичності для всіх симуляторів польотів. Остання версія вирізняється застосуванням сучасних технологій хмарних обчислень і реальних супутникових знімків для створення максимально наближеної до реальності картини світу.

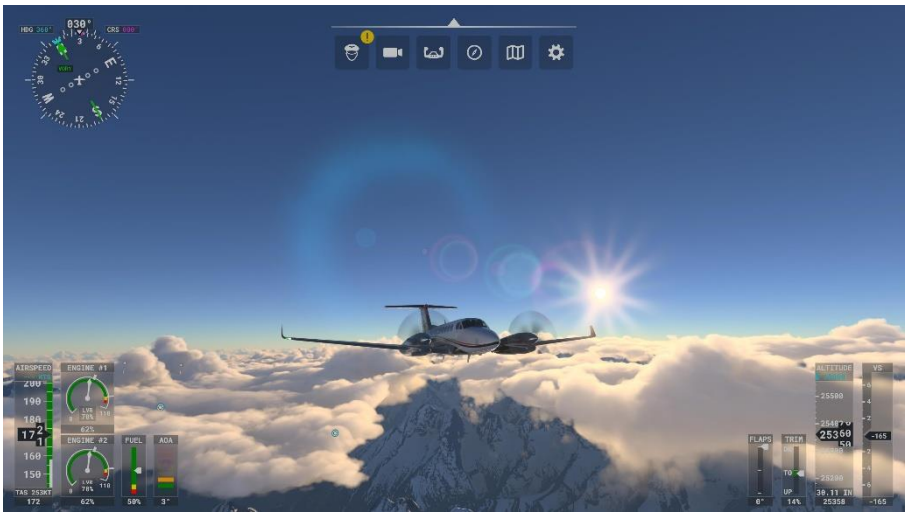

Рис. 1.15. Відображення геймплею Microsoft Flight Simulator

Основні особливості:

- деталізовані тривимірні моделі літаків, створені з високою точністю;
- реалістична фізична модель польоту, яка враховує аеродинаміку та погодні умови;
- використання даних Bing Maps для створення глобальної карти Землі;
- динамічна зміна часу доби та реалістичні погодні умови в режимі реального часу;
- можливість керування як малими літаками, так і великими пасажирськими авіалайнерами.

Переваги:

- максимальна реалістичність польотів;
- найбільша віртуальна карта серед симуляторів;

- широкий вибір літаків і аеропортів;
- використання у навчальних цілях для підготовки пілотів.

Недоліки:

- дуже високі системні вимоги;
- складність у навчанні для початківців;
- великий обсяг гри та потреба в швидкому інтернеті для завантаження даних.

Microsoft Flight Simulator вважається еталоном серед авіаційних симуляторів. Його рівень реалістичності робить його не лише розважальною грою, а й професійним інструментом для навчання. Саме цей проєкт задає планку якості, до якої прагнуть інші симулятори польотів.

Euro Truck Simulator 2 (рис. 1.16) – один із найпопулярніших транспортних симуляторів, розроблений компанією SCS Software. Гра дозволяє гравцеві відчути себе водієм вантажівки, виконуючи перевезення вантажів дорогами Європи. ETS2 поєднує в собі реалістичне керування транспортом із елементами бізнес-симулятора, оскільки гравець може розвивати власну логістичну компанію.

Рис. 1.16. Відображення геймплею Euro Truck Simulator 2

Основні особливості:

- реалістична фізика руху вантажних автомобілів;
- велика мапа, що охоплює більшість європейських країн;
- можливість тюнінгу та модернізації вантажівок;

- система розвитку бізнесу: найм водіїв, купівля гаражів та транспортних засобів;
- підтримка модифікацій, що значно розширює функціонал гри.

Переваги:

- високий рівень занурення завдяки деталізованим моделям вантажівок і трас;
- поєднання симуляції керування з економічною складовою;
- велика активна спільнота гравців і модмейкерів;
- підтримка мультиплеєра (через сторонні модифікації).

Недоліки:

- нерідко зауважується одноманітність ґеймплею;
- базова версія має обмежену кількість країн, решта доступна через DLC;
- середня реалістичність фізики порівняно зі спеціалізованими симуляторами.

Euro Truck Simulator 2 є яскравим прикладом того, як симулятор може поєднувати розважальну та навчальну складову. Гра навчає плануванню, управлінню ресурсами та водінню, залишаючись водночас масовим і доступним продуктом.

Висновки для першого розділу

У ході аналізу сучасних методів та інструментів для розробки ігрового проєкту було розглянуто ключові аспекти, що впливають на якість та ефективність створення 3D ігор-симуляторів. Зокрема, встановлено, що правильний вибір рушія та програмного забезпечення для роботи з 3D-графікою визначає не лише технічні можливості майбутнього продукту, але й його економічну доцільність і швидкість реалізації.

Дослідження найбільш поширених рушіїв – Unreal Engine, Unity та Godot – показало, що кожен із них має переваги та недоліки. Unreal Engine забезпечує найвищий рівень реалістичності та функціоналу, проте вимагає значних ресурсів і досвіду. Unity поєднує універсальність і зручність, проте має обмеження у великих проєктах. Godot вирізняється простотою, відкритим кодом та

оптимальністю для інді-розробки, що робить його доцільним вибором для створення навчальних і дослідницьких симуляторів.

Щодо програм для роботи з 3D-графікою, то 3ds Max і Maya залишаються провідними професійними інструментами, що застосовуються у великих студіях. Водночас Blender завдяки відкритому коду, багатому інструментарію й активній спільноті став одним із найпопулярнішим серед незалежних розробників і освітніх проєктів.

Огляд сучасних симуляторів (Liftoff, DRL Simulator, Microsoft Flight Simulator, Euro Truck Simulator 2) підтвердив, що ігри цього жанру поєднують реалістичність із навчальним і тренувальним потенціалом, а також задають високі стандарти якості, на які варто орієнтуватися при створенні власного проєкту.

Таким чином, для реалізації поставленої мети доцільно поєднати Godot Engine як рушій та Blender як засіб для створення графічного контенту, що забезпечить баланс між функціональністю, простотою освоєння, продуктивністю та доступністю.

РОЗДІЛ 2
ДОСЛІДЖЕННЯ ФУНКЦІОНАЛУ ТА МОЖЛИВОСТЕЙ GODOT ТА BLENDER ДЛЯ РОЗРОБКИ 3D ГРИ-СИМУЛЯТОРА

2.1 Godot Engine як основа створення 3D-симулятора

Godot Engine є відкритим ігровим рушієм, який поєднує в собі багатофункціональність, модульність і простоту у використанні. Його архітектура побудована на принципах об’єктно-орієнтованого програмування та компонентної системи, що дозволяє розробникам створювати комплексні ігрові проєкти, не занурюючись у низькорівневі технічні аспекти рендерингу або фізики. Центральним елементом рушія є сцена (Scene), яка виступає базовим контейнером для об’єктів гри. Кожна сцена може містити кілька вузлів (Node), кожен з яких виконує певну функцію, наприклад, рендеринг 3D-моделі, обробку фізики або управління логікою персонажа (рис. 2.1). Така структура дозволяє гнучко комбінувати різні елементи гри та повторно використовувати компоненти у багатьох сценах, що особливо важливо при розробці 3D-симуляторів зі складними ландшафтами та численними об’єктами.

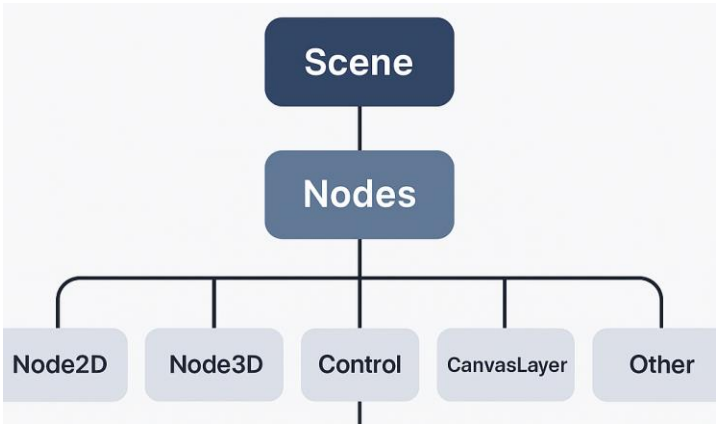

Рис. 2.1. Схема ієрархії сцен Godot

Ключовими компонентами рушія є рендеринг, фізичний движок, система вводу та скриптова система. Рендеринг у Godot підтримує як 2D, так і 3D-графіку, з використанням OpenGL та Vulkan, що забезпечує високу продуктивність навіть на середньому апаратному забезпеченні. Фізичний движок включає модулі для симуляції тіл, колізій, гравітації та інших взаємодій у

просторі, що дає змогу реалізовувати реалістичну поведінку об’єктів без додаткових сторонніх бібліотек. Система вводу дозволяє обробляти події від клавіатури, миші, джойстиків та сенсорних екранів, що робить рушій придатним для розробки різних типів ігор – від класичних симуляторів до VR-проєктів. Скриптова система, представлена мовами GDScript, C# та VisualScript, дозволяє програмувати логіку гри на різних рівнях абстракції, забезпечуючи як швидку розробку прототипів, так і гнучку інтеграцію складних алгоритмів.

Таблиця 2.1.

Ключові компоненти Godot

Компонент	**Призначення**	**Переваги**
Рендеринг	Відображення 2D та 3D графіки	Підтримка OpenGL та Vulkan, висока продуктивність на середньому обладнанні
Фізичний движок	Симуляція тіл, колізій, гравітації та взаємодій у просторі	Реалістична поведінка об’єктів без потреби в сторонніх бібліотеках
Система вводу	Обробка подій з клавіатури, миші, джойстиків, сенсорних екранів	Гнучкість для різних платформ, включно з VR
Скриптова система	Програмування логіки гри на різних рівнях абстракції	Підтримка GDScript, C#, VisualScript; швидка розробка прототипів і складних алгоритмів

Ще одним важливим аспектом архітектури є система сигналів (Signals), яка реалізує патерн спостерігача. Це дозволяє вузлам взаємодіяти між собою без жорсткого зв’язку, підвищуючи модульність та спрощуючи тестування окремих компонентів. Завдяки такому підходу, розробник може легко додавати нові функції або змінювати існуючі без ризику порушити роботу інших частин проєкту. Наприклад, у 3D-симуляторі дронів сигнали можуть використовуватися для повідомлення про зіткнення дрона з об’єктами або завершення проходження траси, що дозволяє гнучко реалізувати механіку гри.

Окрему увагу варто приділити редактору Godot, який інтегрований з рушієм і забезпечує візуальне управління сценами та ресурсами. Редактор дозволяє редагувати об’єкти у 3D-просторі, керувати матеріалами, анімаціями та фізикою без необхідності постійного програмування. Таке поєднання архітектури

рушія та зручного редактора значно скорочує час розробки та підвищує ефективність роботи команди.

Таким чином, архітектура Godot Engine поєднує модульність, гнучкість та простоту інтеграції різних систем, що робить його оптимальним інструментом для розробки 3D-симуляторів. Основні компоненти рушія забезпечують повний цикл створення гри – від розробки графічних об’єктів до реалізації фізики та логіки взаємодії. Використання сигналів і компонентної системи дозволяє створювати складні проєкти без надлишкового зв’язування елементів, що підвищує масштабованість і зручність подальшої підтримки (рис. 2.2).

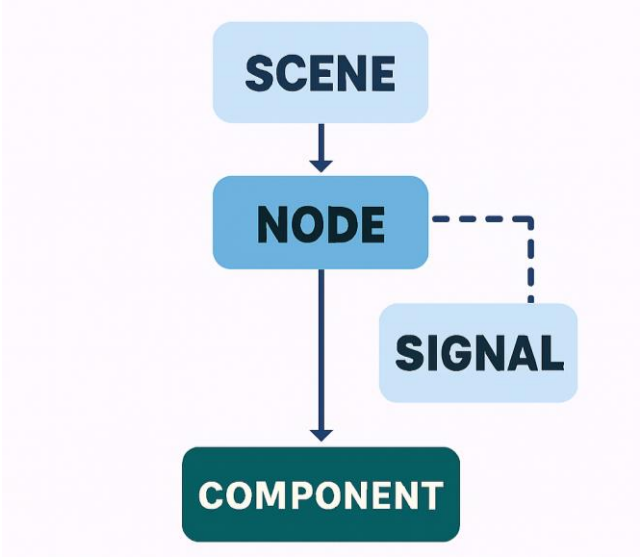

Рис. 2.2. Архітектура Godot

Однією з ключових особливостей Godot Engine є його здатність ефективно працювати з 3D-графікою, фізикою та логікою гри через інтегровані скриптові мови. Рушій надає повний набір інструментів для створення тривимірного світу, включаючи підтримку моделей, текстур, матеріалів, освітлення та камер. Це дозволяє розробникам реалізовувати візуально насичені та технічно складні 3D-симулятори без потреби у використанні сторонніх графічних движків [22].

Godot використовує вузлову систему для організації 3D-об’єктів. Кожен об’єкт у сцені може мати кілька вузлів, що відповідають за:

- рендеринг геометрії (MeshInstance, CSGShape);
- колізії та фізичні властивості (RigidBody, CollisionShape);
- анімацію (AnimationPlayer, Skeleton);

- управління камерою та світлом (Camera, Light).

Таблиця 2.2.

Основні вузли 3D у Godot та їх призначення

Вузол	Призначення
Node3D	Базовий вузол для позиціонування, обертання та масштабування в 3D-просторі
MeshInstance3D	Відображення 3D-моделі (мешу) у сцені
Camera3D	Визначає точку огляду сцени; рендерить зображення на екран
Light3D	Джерело освітлення (Directional, Omni, Spot)
CollisionShape3D	Визначає форму для фізичних взаємодій (колізій)
RigidBody3D	Фізичне тіло з масою, яке реагує на сили та зіткнення
StaticBody3D	Нерухоме фізичне тіло, що бере участь у колізіях
Area3D	Визначає зону для тригерів, зон дії або впливу на фізику
AnimationPlayer	Відтворення анімацій для властивостей вузлів
Skeleton3D	Структура кісток для скіннінгу та анімації персонажів
NavigationRegion3D	Визначає область для навігації агентів (AI)
Particles3D	Генерація візуальних ефектів (диму, вогню, магії тощо)

Фізичний движок Godot забезпечує реалізацію реалістичних взаємодій у просторі, включаючи симуляцію гравітації, зіткнень, тертя й об’єктних сил. Для 3D-симуляторів це особливо важливо, оскільки користувач очікує природну поведінку об’єктів у світі. Серед особливостей фізичного движка можна виділити:

- підтримку різних типів тіл: RigidBody, StaticBody, KinematicBody;
- моделювання колізій у реальному часі;
- підтримку джойстиків та контролерів для фізичного керування персонажами або транспортними засобами;
- вбудовану систему сил та імпульсів для реалістичної динаміки.

Для програмної частини логіки гри Godot пропонує GDScript – власну скриптову мову, а також підтримку C# та VisualScript [9]. GDScript синтаксично

нагадує Python, що робить його легким для освоєння та швидким у прототипуванні. Основні переваги використання GDScript:

- простота та читабельність коду;
- тісна інтеграція з вузловою системою Godot;
- можливість швидкого тестування і відлагодження;
- підтримка сигналів і сцен для подій та взаємодії об’єктів.

C# у Godot дозволяє реалізовувати складні алгоритми та використовувати багатопоточність, що підвищує продуктивність проєкту, особливо при розрахунках фізики або обробці великої кількості об’єктів у сцені. Застосування C# доцільне для великих симуляторів або ігор із високою логічною складністю [15].

Ще однією цікавинкою є використання AnimationTree та AnimationPlayer, які дозволяють будувати комплексні анімаційні системи для персонажів та об’єктів без складного коду. Сценарії гри можна пов’язувати зі станами анімацій, фізики та подій, що забезпечує синхронізацію візуальних та логічних елементів.

Таблиця 2.3.

Порівняння GDScript і C# у Godot

Мова	*Переваги*	*Недоліки*
GDScript	Спеціально створена для Godot, тісно інтегрована з редактором; Простий синтаксис, схожий на Python; Швидкий прототипінг та легке навчання.	Менша продуктивність порівняно з C#; Обмежена функціональність поза Godot; Відсутність статичної типізації (хоча частково підтримується).
C#	Вища продуктивність Потужна екосистема .NET Статична типізація, підтримка LINQ, async/await; Краще підходить для великих проєктів.	Потребує встановлення Mono; Менша інтеграція з редактором Godot; Складніший синтаксис для новачків.

Таким чином, поєднання вузлової системи, потужного фізичного движка та скриптових мов у Godot дозволяє створювати багаторівневі 3D-симулятори з

високим рівнем реалізму та інтерактивності. Вбудовані інструменти та логіка взаємодії компонентів дозволяють розробнику контролювати поведінку кожного об’єкта, забезпечуючи ефективну та масштабовану розробку ігор.

Для сучасних 3D-симуляторів базового функціоналу рушія часто недостатньо, особливо якщо проєкт передбачає складні ландшафти або взаємодію з хмарними сервісами. У Godot є можливість значно розширити стандартні можливості за допомогою сторонніх аддонів та API, серед яких ключовими для даного проєкту були Terrain3D та Godot Firebase [14].

Terrain3D дозволяє ефективно генерувати та редагувати рельєф у 3D-просторі без необхідності ручного моделювання кожного елемента ландшафту (рис. 2.3). Це особливо важливо для симуляторів із відкритими світами, де природність та масштабність середовища є критичною для ігрового досвіду. Аддон надає інструменти для налаштування висотного рельєфу, текстурування поверхні та застосування різних матеріалів, що дозволяє швидко створювати гори, долини, озера та інші природні об’єкти. Використання Terrain3D значно скорочує час розробки та дозволяє підтримувати гнучкість дизайну, оскільки зміни рельєфу можна виконувати безпосередньо у редакторі Godot, переглядаючи результат у реальному часі.

Рис. 2.3. Інтерфейс плагіну Terrain3D

Інший важливий інструмент – Godot Firebase, який забезпечує інтеграцію рушія з хмарною платформою Firebase. Це дозволяє реалізовувати збереження

даних користувача, керування статистикою, а також синхронізацію ігрового прогресу у реальному часі. Завдяки API Godot Firebase можна організувати взаємодію гри з сервером без потреби створювати власну інфраструктуру, що значно спрощує процес розробки та робить проєкт масштабованим.

Головна перевага використання Godot Firebase полягає в тому, що він забезпечує повний цикл роботи з даними: від аутентифікації користувача та зберігання даних у базі до отримання та оновлення інформації у реальному часі. Це особливо важливо для симуляторів, де користувачі можуть взаємодіяти з об’єктами гри, а результати їх дій мають миттєво відображатися в системі.

Варто зазначити, що застосування сторонніх аддонів та API накладає певні обмеження і вимагає уважного підходу до сумісності та продуктивності. Terrain3D може потребувати оптимізації рельєфу для роботи на слабших пристроях, а Godot Firebase – налаштування безпеки та правил доступу до бази даних. Проте правильно організована інтеграція цих інструментів дозволяє значно підвищити якість проєкту та розширити його функціональні можливості без значного збільшення обсягу коду.

Таким чином, використання Terrain3D та Godot Firebase дозволяє поєднати візуальну реалістичність 3D-середовища з функціональністю хмарних сервісів, що робить розробку симулятора більш ефективною та масштабованою. Ці інструменти демонструють гнучкість Godot Engine та його здатність адаптуватися під потреби сучасних 3D-ігор.

2.2 Blender у процесі створення ігрового контенту

Blender є потужним універсальним інструментом для створення 3D-моделей, що робить його незамінним компонентом у процесі розробки ігрового контенту. Однією з головних переваг Blender є поєднання різноманітних методів моделювання в одному середовищі, що дозволяє розробнику ефективно працювати з об’єктами будь-якої складності – від простих предметів інтер’єру до складних механізмів чи транспортних засобів [10].

Процес моделювання у Blender включає кілька ключових підходів:

- полігональне моделювання – створення об’єктів за допомогою полігонів, що дає точний контроль над формою та деталізацією моделі;
- скульптинг – метод, що імітує роботу скульптора, дозволяючи створювати високодеталізовані органічні форми;
- моделювання на основі кривих і NURBS – зручне для створення гладких і симетричних поверхонь;
- модифікатори – автоматизація рутинних операцій, таких як дзеркальне відображення, субдивізія поверхні або створення повторюваних елементів.

Другим важливим аспектом є текстурування та матеріали, що визначають зовнішній вигляд об’єктів у грі. Blender надає широкий спектр інструментів для роботи з матеріалами:

- UV-розгортка – процес проєкції поверхні 3D-моделі на плоску площину для точного нанесення текстур.
- Вули матеріалів (Shader Nodes) – гнучка система для створення складних матеріалів та ефектів освітлення.
- Bake-текстури – дозволяє «випікати» тіні, нормалі або AO у текстури, оптимізуючи продуктивність гри.

Особливо важливою у процесі розробки 3D-симуляторів є інтеграція моделювання та текстурування з ігровим рушієм. Для цього використовують формати експорту, такі як .glb/.gltf, .fbx та інші, що забезпечують збереження геометрії, матеріалів та анімаційних даних. Використання правильного формату експорту дозволяє максимально зберегти якість моделі та забезпечує сумісність із Godot, зменшуючи ризик втрати даних при перенесенні об’єктів у рушій.

Таким чином, моделювання та текстурування у Blender є базовими етапами створення ігрового контенту. Використання різних методів моделювання, налаштування матеріалів і текстур, а також правильний вибір формату експорту дозволяють створювати високоякісні 3D-об’єкти, які легко інтегруються у 3D-симулятор на рушії Godot. Крім того, цей підхід забезпечує гнучкість і масштабованість проєкту, дозволяючи швидко вносити зміни у дизайн та зовнішній вигляд об’єктів без потреби повторного створення моделей.

Анімація та ріггінг є ключовими аспектами розробки 3D-контенту для ігор, оскільки вони визначають динаміку та реалістичність руху об’єктів у віртуальному світі. Blender пропонує потужний набір інструментів для створення анімацій, що дозволяє реалізовувати як прості рухи предметів, так і складні поведінкові моделі персонажів. Центральним компонентом є система ріггінгу, яка забезпечує побудову скелета для 3D-моделі, дозволяючи анімувати її без спотворення геометрії.

Ріггінг включає кілька важливих етапів:

- створення скелету (Armature) – розміщення кісток у відповідності з анатомією об’єкта або механізму;
- Weight painting – визначення впливу кісток на вершини моделі для плавного руху (рис. 2.4);
- IK/FK системи – інверсна та пряма кінематика для більш природних та керованих рухів.

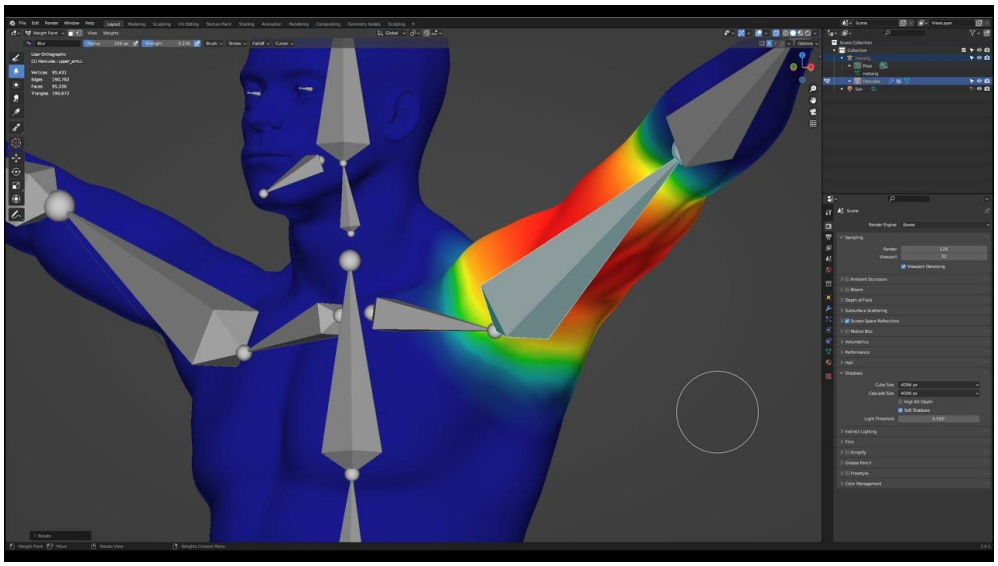

Рис. 2.4. Визначення впливу кісток на модель

Система анімації Blender дозволяє створювати складні послідовності рухів за допомогою Animation Editor та NLA Editor, що забезпечує керування ключовими кадрами та об’єднання кількох анімацій в єдиний цикл. Для ігрових симуляторів це особливо важливо, оскільки дозволяє реалізовувати взаємодії між об’єктами, динамічні події та реакції персонажів на зміну середовища.

Візуальні ефекти (VFX) у Blender доповнюють анімацію, створюючи атмосферу та підвищуючи візуальну привабливість гри [27]. Серед основних можливостей можна виділити:

- Particle System – генерація частинок для імітації диму, вогню, води або пилу (рис. 2.5);
- Modifiers та Shader Nodes – створення складних матеріалів та спеціальних ефектів;
- Simulation Tools – фізично коректні симуляції тканин, рідин і газів.

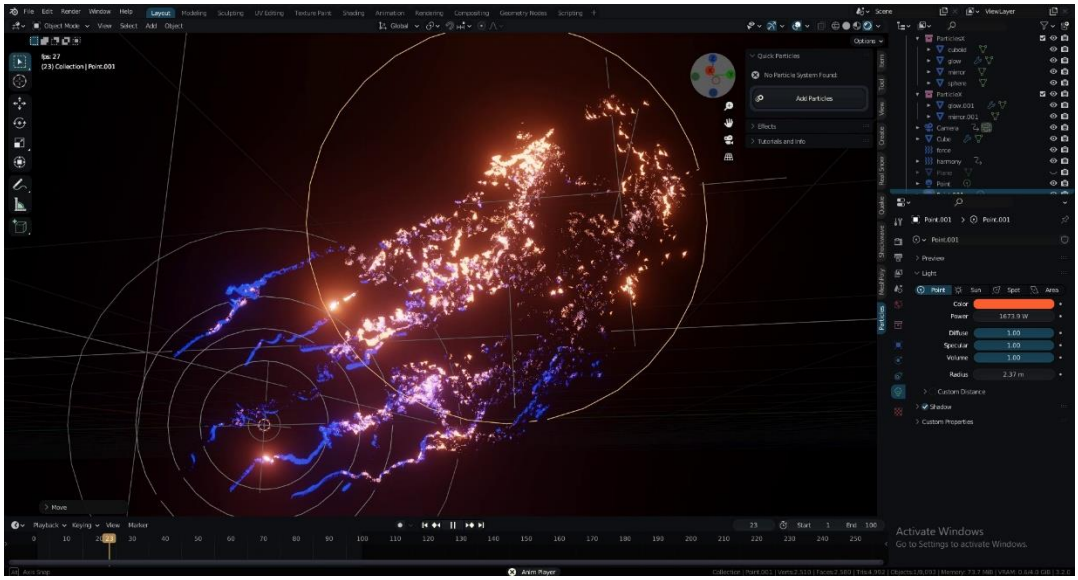

Рис. 2.5. Демонстрація Particle System

Особливу увагу в створенні ігрових 3D-об’єктів займає інтеграція анімацій з рушієм Godot. Після ріггінгу та анімації у Blender моделі можна експортувати у форматі .glb/.gltf, що дозволяє зберегти анімаційні дані та матеріали без втрати якості. Завдяки цьому у Godot можна безпосередньо використовувати готові анімації, прив’язуючи їх до сценаріїв гри та сигналів рушія, що забезпечує синхронізацію руху об’єктів з логікою симулятора.

Таким чином, анімація, ріггінг та створення візуальних ефектів у Blender є невід’ємною частиною процесу створення ігрового контенту. Використання цих інструментів дозволяє досягти високого рівня реалізму рухів, забезпечити інтерактивність та підвищити загальну якість 3D-симулятора, що робить Blender ефективним та універсальним рішенням для розробників ігор.

2.3 Хмарні технології в сучасній ігровій розробці

Сучасні ігрові проєкти часто вимагають комплексного підходу до збереження та обробки даних, особливо якщо мова йде про багатокористувацькі симулятори або ігри з хмарним збереженням прогресу [3]. У цьому контексті платформу Firebase можна розглядати як універсальний інструмент для реалізації бекенд-функціоналу, який дозволяє зменшити витрати часу на розробку власних серверних рішень та зосередитися на створенні ігрового контенту.

Firebase пропонує низку сервісів, які забезпечують повний цикл роботи з даними:

- Realtime Database – хмарна база даних, що дозволяє зберігати та отримувати дані у режимі реального часу;
- Firestore – більш сучасна база даних, яка підтримує складні запити та масштабування;
- Authentication – інструменти для аутентифікації користувачів через електронну пошту, соціальні мережі та інші платформи;
- Cloud Storage – зберігання великих файлів, таких як зображення, моделі та відео;
- Cloud Functions – серверна логіка, що виконується у хмарі у відповідь на події, наприклад, оновлення бази даних або запит користувача.

Однією з ключових переваг Firebase є можливість швидкої інтеграції з рушіями ігор, такими як Godot, через спеціальні API та аддони [7]. Це дозволяє реалізовувати:

- збереження прогресу гравців у реальному часі;
- синхронізацію ігрових станів між різними пристроями;
- збір та аналіз статистики користувачів;
- керування внутрішньоігровими даними, включаючи рейтинги, інвентар та налаштування.

Крім того, Firebase забезпечує масштабованість і надійність. Хмарні сервери автоматично адаптуються до кількості користувачів, що дозволяє розробникам уникнути проблем із продуктивністю при зростанні аудиторії.

Вбудовані засоби безпеки дозволяють встановлювати правила доступу до даних, контролюючи права читання та запису для різних груп користувачів.

Таблиця 2.4

Засоби безпеки

Аспект	Переваги	Обмеження
Аутентифікація	Просте підключення через Google, Facebook, email, тощо	Обмежена кастомізація UI та логіки входу
База даних (Firestore / RTDB)	Реальний час, масштабованість, offline-режим	Може бути дорогим при великому обсязі запитів або складних структур
Хмарне сховище	Зберігання зображень, аудіо, відео, сейвів	Обмеження на розмір файлів, потреба в додатковій безпеці
Хостинг	Швидкий деплой для веб-ігор або супутніх сервісів	Не підходить для великих ігрових клієнтів або серверної логіки
Функції (Cloud Functions)	Серверна логіка без власного сервера, обробка подій	Затримки cold start, обмеження на тривалість виконання
Push-сповіщення	Легка інтеграція для мобільних ігор	Обмежена кастомізація на різних платформах
Аналітика	Вбудована статистика, поведінка гравців, конверсії	Не завжди достатньо гнучка для глибокого аналізу
Crashlytics	Виявлення помилок у реальному часі	Працює лише з мобільними платформами (Android/iOS)
Масштабованість	Автоматичне масштабування сервісів	Вартість зростає зі збільшенням навантаження
Інтеграція з Godot / Unity	Плагіни та SDK для популярних рушіїв	Потребує додаткової конфігурації, не всі функції доступні “з коробки”

Використання Firebase також має певні обмеження. Зокрема, робота з великою кількістю складних запитів у Realtime Database може потребувати оптимізації структури даних, а Cloud Functions можуть створювати затримки при масових операціях. Крім того, платформа є хмарною, що означає залежність від стабільного інтернет-з’єднання для доступу до даних, що слід враховувати при розробці симуляторів для офлайн-режиму.

Попри ці обмеження, Firebase залишається ефективним рішенням для сучасної розробки ігор, оскільки забезпечує високу гнучкість, швидке розгортання бекенд-функціоналу та інтеграцію з різними платформами. Це робить його ключовим інструментом при створенні ігор із хмарним збереженням, синхронізацією даних та багатокористувацькими функціями.

Таким чином, використання Firebase у процесі розробки ігор дозволяє поєднати зручність хмарного сервісу з можливістю гнучкого налаштування даних, забезпечуючи надійність, масштабованість та ефективну роботу ігрового проєкту.

Інтеграція платформи Firebase з ігровим рушієм Godot відкриває широкі можливості для реалізації хмарного збереження даних, синхронізації ігрового прогресу та обробки інформації у режимі реального часу. Використання Godot Firebase API дозволяє безпосередньо підключати функціонал хмарних сервісів до логіки гри, що значно спрощує розробку багатокористувацьких ігор та симуляторів з великими базами користувачів.

Однією з головних переваг такого підходу є можливість автоматизації процесів збереження та завантаження даних. Через Godot Firebase можна організувати миттєве оновлення стану гри, що дозволяє гравцям продовжувати гру з будь-якого пристрою, зберігаючи усі налаштування, досягнення та інвентар. Крім того, платформа забезпечує зручне управління користувацькою аутентифікацією, що дозволяє безпечно ідентифікувати гравців і прив’язувати дані до конкретних акаунтів.

Інтеграція також відкриває можливості для аналітики та моніторингу поведінки користувачів. За допомогою Firebase можна збирати статистичні дані, наприклад, частоту входів, тривалість ігрових сесій, найпопулярніші дії або об’єкти у грі. Це дозволяє розробникам приймати обґрунтовані рішення щодо балансування геймплею, оптимізації контенту та планування оновлень.

У практичних сценаріях використання Godot Firebase включає: хмарне збереження прогресу, синхронізацію станів ігрового світу між гравцями, ведення рейтингових таблиць та інтеграцію внутрішньо-ігрових нагород (рис. 2.6). Ці

функції дозволяють створювати більш інтерактивні та масштабовані ігрові продукти, не потребуючи розробки власного серверного забезпечення.

Рис. 2.6. Практичні сценарії використання Firebase у 3D-симуляторі

Водночас інтеграція з Firebase вимагає уваги до оптимізації запитів і структури даних. Наприклад, надмірна кількість одночасних звернень до бази даних або складні запити можуть створювати затримки у відображенні змін у грі. Також важливо забезпечити безпеку доступу до даних, використовуючи правила доступу та шифрування. Незважаючи на ці виклики, правильно організована інтеграція забезпечує стабільну роботу симулятора, знижує навантаження на локальні ресурси та підвищує зручність для користувачів.

Таким чином, інтеграція Firebase з Godot демонструє ефективність хмарних технологій у сучасній розробці ігор, поєднуючи зручність хмарного збереження, аналітичні можливості та високий рівень інтерактивності. Це робить Firebase незамінним інструментом для симуляторів і багатокористувацьких проєктів, де важлива синхронізація, масштабованість та надійність обробки даних.

Використання хмарних сервісів у розробці 3D-симуляторів стає дедалі більш актуальним, оскільки забезпечує ефективне управління даними, масштабованість проєкту та гнучкість у впровадженні нових функцій. Основною перевагою є можливість централізованого зберігання та обробки великих обсягів інформації, що включає дані користувачів, параметри ігрових об’єктів та

результати симуляцій. Це дозволяє розробникам зосередитися на створенні контенту та ігрової логіки, не витрачаючи ресурси на підтримку власних серверних рішень.

Серед інших важливих переваг можна виділити:

- синхронізацію прогресу користувачів у реальному часі, що підвищує зручність для гравців і забезпечує безперервність ігрового процесу;
- масштабованість та адаптацію до навантаження, що дозволяє обслуговувати великі аудиторії без значного збільшення технічних витрат;
- вбудовані засоби безпеки та контролю доступу, які забезпечують захист персональних даних і запобігають несанкціонованому доступу до хмарних ресурсів.

Водночас інтеграція хмарних технологій пов’язана з певними викликами. По-перше, залежність від стабільного інтернет-з’єднання може впливати на досвід користувачів у регіонах із нестабільним доступом до мережі. По-друге, обробка великої кількості запитів одночасно може створювати затримки та вимагати оптимізації структури бази даних і логіки обробки подій. По-третє, деякі складні функції, наприклад, реалістичні фізичні симуляції або взаємодія численних об’єктів у реальному часі, можуть потребувати значних ресурсів, що вимагає балансування між локальними та хмарними обчисленнями.

Ключовим фактором ефективного використання хмарних сервісів у 3D-симуляторах є правильне планування та оптимізація даних. Це включає обмеження кількості запитів, використання кешування для часто запитуваних даних та застосування асинхронних функцій для мінімізації затримок. Крім того, важливо поєднувати хмарні сервіси з локальною обробкою, щоб забезпечити безперервність ігрового процесу навіть у випадках тимчасових проблем із мережею (рис. 2.7).

Рис. 2.7. Баланс локальної обробки та хмарних сервісів у 3D-симуляторі

Таким чином, використання хмарних технологій у 3D-симуляторах відкриває широкі можливості для масштабування та підвищення якості ігрового досвіду, одночасно накладаючи певні технічні та організаційні вимоги. Зважений підхід до інтеграції, оптимізації та безпеки дозволяє максимально ефективно поєднати переваги хмарних платформ з особливостями конкретного проєкту, забезпечуючи стабільну та надійну роботу симулятора.

Висновки для другого розділу

У другому розділі було детально розглянуто теоретичні аспекти використання основних інструментів для розробки 3D-ігрового проєкту, а саме рушія Godot, редактора Blender і хмарної платформи Firebase. Аналіз показав, що Godot забезпечує гнучку архітектуру та широкі можливості для роботи з фізикою, анімацією та сценаріями, що робить його ефективним інструментом для створення інтерактивних симуляторів. Blender у свою чергу дозволяє створювати високоякісний 3D-контент, включно з моделюванням, текстуруванням, ріггінгом і візуальними ефектами, забезпечуючи гнучку інтеграцію з рушієм через стандартизовані формати експорту. Використання хмарних сервісів, таких як Firebase, надає можливість централізованого збереження даних, синхронізації

прогресу користувачів і обробки інформації у реальному часі, що є особливо важливим для багатокористувацьких проєктів. Разом ці інструменти формують комплексне середовище для розробки 3D-симулятора, дозволяючи поєднувати художню якість контенту з технічною функціональністю та масштабованістю проєкту. Таким чином, теоретичний аналіз показав доцільність використання зазначених платформ і технологій як основи для успішної реалізації ігрового проєкту.

РОЗДІЛ 3
РОЗРОБКА ІГРОВОГО ПРОЄКТА З ВИКОРИСТАННЯМ GODOT ТА BLENDER

3.1 Підготовка середовища розробки й інтеграція інструментів

Початковим етапом практичної реалізації симулятора стало створення нового проєкту в середовищі Godot Engine. Цей рушій було обрано завдяки своїй відкритості, стабільності, простому інтерфейсу та зручності у роботі з тривимірною графікою. Крім того, Godot забезпечує високу продуктивність навіть на середніх за характеристиками пристроях, що є важливою перевагою для симуляторів, орієнтованих на точну фізику та плавний рендеринг сцени.

Створення проєкту починалося з головного вікна рушія, де користувач задає назву, місце збереження та тип майбутнього середовища. Було обрано назву FPV Drone Simulator і створено окрему робочу директорію. Після цього рушій автоматично формує базову структуру файлів, з якою надалі працює розробник.

Для зручності навігації та впорядкування даних структура була оптимізована вручну – додано окремі папки для різних типів контенту:

- scenes – містить усі сцени гри, включно з головним меню, рівнями та службовими сценами;
- scripts – використовується для збереження логіки гри, реалізованої через мову GDScript;
- assets – для моделей, текстур, шрифтів та графічних матеріалів;
- ui – для елементів інтерфейсу користувача, кнопок, панелей і текстових полів;
- sounds – для звукових ефектів і музики.

Така структура дозволяє швидко орієнтуватися у великому проєкті й підтримувати порядок під час активної розробки (рис. 3.1).

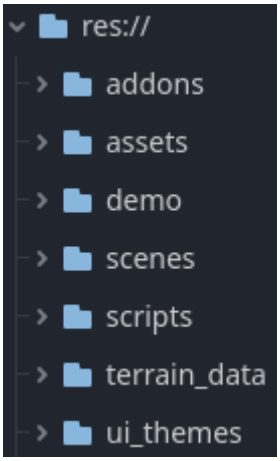

Рис. 3.1. Файлова структура проєкту

Далі було виконано базове налаштування параметрів проєкту, що визначають поведінку рушія, роздільну здатність, фізику та інтерфейс. У меню Project → Project Settings встановлено такі параметри:

•Display → Window: роздільна здатність 1920×1080, що відповідає стандарту Full HD;

•VSync: увімкнено для синхронізації кадрів з частотою оновлення монітора, що усуває розриви зображення;

•Physics → Common → Physics FPS: підвищено до 120, аби забезпечити плавність симуляції руху;

•Display → Stretch Mode: встановлено значення *Viewport*, щоб гра правильно масштабувалася під різні екрани.

Наступним кроком стало налаштування системи вводу (Input Map), яка дозволяє створювати власні події для керування ігровими об’єктами. Через вкладку Input Map було додано основні дії [23]:

•throttle_up / throttle_down – підйом і зниження тяги;

•pitch_up / pitch_down – нахил уперед і назад;

•roll_left / roll_right – керування креном дрона;

•yaw_left / yaw_right – поворот навколо вертикальної осі;

•reset – відновлення положення дрона у випадку зіткнення або втрати стабільності.

Завдяки цьому вже на ранньому етапі проєкт отримав готову схему керування, яку можна використовувати під час створення ігрових механік (рис. 3.2).

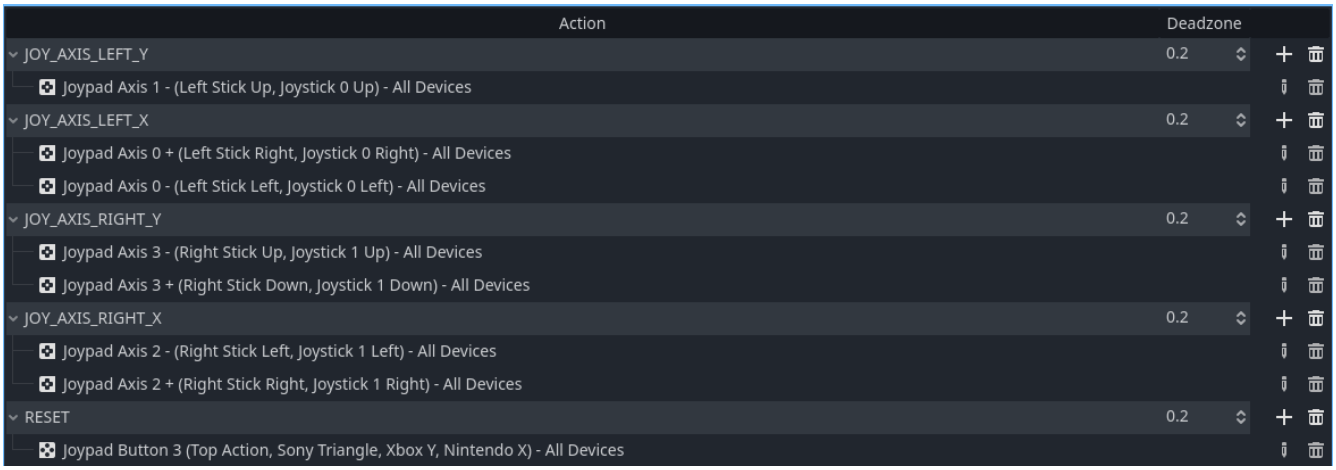

Рис. 3.2. Вкладка Input Map проєкту

Після налаштування вводу було створено початкову сцену проєкту. У Godot будь-який проєкт складається зі сцен, що є логічними контейнерами для об’єктів. У цьому випадку створено головну сцену main.tscn, яка згодом стала основною точкою запуску симулятора. Вона містить такі базові елементи:

- DirectionalLight3D – джерело освітлення, що імітує сонячне світло;
- WorldEnvironment – для керування атмосферними ефектами, туманом і фоном;
- Camera3D – базова камера, спрямована на простір, де надалі розміщуватиметься дрон;
- StaticBody3D із площиною – тимчасова поверхня для тестування руху об’єктів.

На етапі первинного налаштування також було важливо оптимізувати параметри відображення, щоб уникнути надмірного навантаження на систему. У вкладці Rendering зменшено значення параметра Shadow Atlas Size, а також активовано Frustum Culling – відсікання об’єктів, які не потрапляють у поле зору камери. Це забезпечує стабільний FPS під час відтворення сцен з великою кількістю деталей.

Щоб підготувати середовище до імпорту 3D-моделей, було створено тестову сцену з простим освітленням, куди згодом імпортувалися об’єкти з

Blender. Вона використовувалася для перевірки масштабу моделей, якості матеріалів і роботи текстур. Такий підхід дозволив уникнути проблем з різницею у масштабі та нормалях між Blender і Godot ще до етапу основної інтеграції.

Після завершення налаштувань проєкт отримав чітку структуру, налаштовану систему керування, оптимізовані параметри фізики та відображення. Це забезпечило стабільну основу для подальших етапів розробки: створення моделей, побудови сцени, реалізації польоту дрона й інтеграції мережевих можливостей.

Після налаштування базового середовища розробки наступним етапом стало імпортування у рушій Godot тривимірних моделей, створених у Blender. Цей процес є ключовим, адже саме він забезпечує візуальну частину симулятора – від дрона до елементів навколишнього середовища. Blender було обрано як основний інструмент 3D-моделювання завдяки його відкритості, високій точності у роботі з геометрією та можливості оптимізації моделей під ігрові рушії.

Перед початком експорту важливо було дотриматися правильної підготовки моделей. Кожен об’єкт – корпус дрона, пропелери, елементи карти (будівлі, дерева, перешкоди) – проходив етапи оптимізації:

- перевірка нормалей (усі поверхні повинні бути спрямовані назовні);
- об’єднання дрібних об’єктів для зменшення кількості draw calls;
- зменшення кількості полігонів, особливо для другорядних елементів сцени;
- призначення матеріалів і текстур безпосередньо у Blender, щоб уникнути ручного переналаштування у Godot (рис. 3.3).

Рис. 3.3. Модель дрона в робочому середовищі Blender

Для експорту моделей використовувався стандартний формат .glb (GL Transmission Format Binary), який найкраще підходить для обміну даними між Blender та Godot [20]. Цей формат зберігає не лише геометрію об’єктів, але й матеріали, анімації, налаштування освітлення та UV-розгортки. Процес експорту виконувався через меню File → Export → glTF 2.0, де було обрано параметри:

- Format: GLB Binary (.glb);
- Include → Selected Objects: лише вибрані об’єкти;
- Transform → +Y Up: щоб відповідати системі координат Godot;
- Apply Modifiers: активовано для збереження всіх змін геометрії;
- Animation: вимкнено, оскільки рух дрона реалізується у рушії.

Після експорту моделі зберігалися в каталозі /assets/models/, звідки імпортувалися до рушія. Godot автоматично розпізнає формат .glb і створює відповідні ресурси: Mesh, Material, CollisionShape тощо. Це дає змогу одразу розміщувати об’єкти на сцені без додаткового редагування.

На етапі імпорту проводилася перевірка масштабу й орієнтації моделей. Blender і Godot мають різні одиниці вимірювання, тому під час першого імпорту масштаб дрона й інших об’єктів міг відрізнятися. Для вирішення цієї проблеми у Blender масштаб усіх моделей було приведено до 1 : 1, а осі вирівняно під систему координат Godot (вісь Y спрямована вгору, Z – уперед).

Далі виконувалося налаштування матеріалів і освітлення. Хоча формат GLB переносить основні властивості матеріалів, Godot може сприймати їх із

невеликими відмінностями у кольорі чи блиску. Тому після імпорту для кожного об’єкта створювалися окремі матеріали типу StandardMaterial3D, у яких вручну коригувалися параметри [5]:

- Albedo – основний колір або текстура;
- Roughness – ступінь шорсткості поверхні;
- Metallic – рівень металевості;
- Emission – світіння світлодіодних елементів на корпусі дрона.

Окрему увагу було приділено розміщенню моделей на ігровій карті. Для цього створювалася сцена environment.tscn, у якій розміщувалися всі елементи оточення: рельєф місцевості, будівлі, перешкоди, маркери цілей. Кожна модель розташовувалася вручну для досягнення максимальної реалістичності простору польоту. При цьому враховувалися баланс продуктивності та якості – наприклад, об’єкти, що знаходяться далеко від зони гравця, мають спрощену геометрію.

У процесі інтеграції перевірялася відповідність текстур і матеріалів, а також коректність тіней і відбиттів. Для цього в Godot використовувалися тимчасові джерела світла (DirectionalLight3D і OmniLight3D), що допомагали виявити проблеми з нормалями чи некоректними UV-розгортками. Усі знайдені недоліки виправлялися у Blender, після чого модель повторно експортувалася.

Завдяки чітко налагодженому процесу експорту й імпорту вдалося досягти високої точності відображення моделей у Godot, зберігши їхній візуальний стиль, масштаб і фізичну реалістичність. На цьому етапі було сформовано основу візуальної частини симулятора – зібрано повноцінну ігрову сцену, що відображає реалістичне середовище польоту дрона. Ця база стала підґрунтям для подальшого впровадження фізики, системи керування та геймплейних режимів.

Після створення структури проєкту й інтеграції основних моделей постала необхідність розширити функціональні можливості рушія Godot. Для цього були використані два аддони – Terrain і Godot Firebase, які забезпечили створення реалістичної карти та збереження даних користувачів у хмарному сховищі. Їхнє підключення дало змогу суттєво розширити базові можливості рушія без розробки власних систем із нуля.

Підключення та налаштування Terrain3D

Аддон Terrain3D використовується для створення великих відкритих просторів із деталізованим рельєфом, що є важливим елементом симулятора польоту дрона. Стандартні інструменти Godot не забезпечують достатньої гнучкості у побудові складних ландшафтів, тому застосування цього розширення дозволило досягти більш реалістичного результату.

Після завантаження аддона його було додано до каталогу /addons/ у структурі проєкту. Далі, через меню Project → Project Settings → Plugins, аддон активується – після цього в панелі створення вузлів стає доступним новий тип об’єкта Terrain3D.

Було створено окрему сцену terrain_scene.tscn, де розміщено компонент Terrain3D, який слугує основою для ігрової карти. Через інспектор задавалися параметри висоти, розміру тайлів і текстур. Для рельєфу було використано висотну карту, створену у Blender, що дозволило відтворити плавні підйоми, долини та природні нерівності поверхні (рис. 3.4).

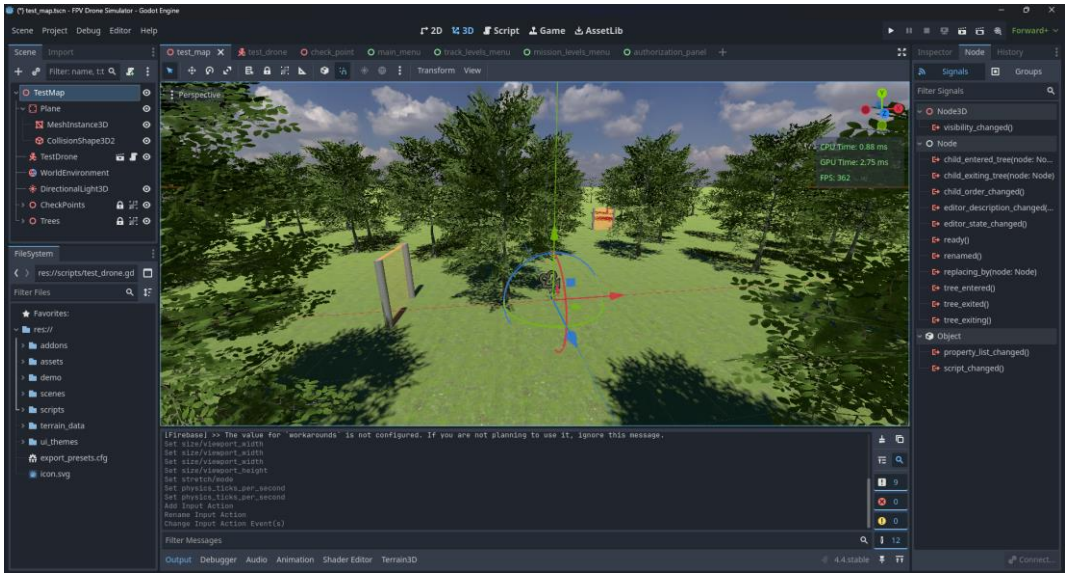

Рис. 3.4. Тестова мапа для польотів

Щоб досягти реалістичного вигляду поверхні, у Terrain3D налаштовувалися кілька шарів текстур – трава, ґрунт, скелі. Для кожного шару визначалися власні параметри масштабування та змішування, що створило плавний перехід між типами поверхні. У результаті отримано природний вигляд місцевості без різких меж між матеріалами.

Додатково для покращення продуктивності активовано Level of Detail (LOD) – систему, яка автоматично спрощує геометрію ландшафту на віддалених ділянках. Це дозволило підтримувати стабільну частоту кадрів навіть на великих картах, що є критичним для симулятора реального часу.

Підключення та налаштування Godot Firebase

Для збереження даних користувача, авторизації та синхронізації прогресу було підключено аддон Godot Firebase. Його інтеграція забезпечила взаємодію рушія з сервісом Firebase Realtime Database, дозволяючи кожному гравцеві мати власний профіль та зберігати статистику польотів у хмарі.

Після завантаження архіву аддона його вміст було розміщено в папці /addons/firebase/. Як і у випадку з Terrain, підключення виконувалося через Project → Project Settings → Plugins, де плагін було активовано. Після цього в інспекторі з'явився новий вузол FirebaseApp, який використовується для зв'язку з базою даних.

Далі налаштовувалися параметри підключення до хмарного сервісу. У консольній панелі Firebase Console було створено новий проєкт із назвою FPV Drone Simulator та додано веб-конфігурацію, що містить API-ключ, ідентифікатор проєкту та адресу бази даних. Ці параметри було внесено до відповідного конфігураційного файлу Godot, який зберігається в каталозі /config/firebase_settings.cfg.

Основними функціями, які реалізовуються через Firebase, є:

- реєстрація й авторизація користувачів (з використанням email і пароля);
- збереження прогресу гравця, включно з пройденими місіями та результатами;
- завантаження даних при вході – користувач відновлює свій стан незалежно від пристрою.

Під час тестування перевірялося коректне з'єднання з Firebase: при реєстрації нового користувача створювався відповідний запис у базі, а після проходження рівня змінювалися дані про його досягнення. Це дозволило

підтвердити стабільність інтеграції й правильність налаштування мережевих параметрів.

Важливим аспектом було також забезпечення безпеки даних. Для цього у Firebase застосовано правила доступу, що обмежують перегляд і зміну інформації лише авторизованими користувачами. Це гарантує, що кожен гравець має доступ лише до власних даних, а сторонні користувачі не можуть їх змінювати або переглядати.

Завдяки підключенню аддонів Terrain3D і Godot Firebase проєкт отримав не лише реалістичне тривимірне середовище, але й сучасну систему взаємодії з хмарною базою даних. Такий підхід поєднує в собі дві важливі складові сучасного симулятора – візуальну глибину і інтерактивну персоналізацію користувача. Це створює фундамент для реалізації повноцінного ігрового досвіду з можливістю збереження та продовження прогресу з будь-якого пристрою.

3.2 Створення ігрового контенту та функціональних систем

Процес створення 3D-моделей у Blender став одним із ключових етапів розробки симулятора, адже саме візуальна складова формує перше враження гравця та впливає на загальне занурення в гру. Основна мета цього етапу полягала у створенні реалістичних, але водночас оптимізованих моделей, які забезпечують баланс між якістю відображення та продуктивністю під час виконання симуляції в рушії Godot.

На початковому етапі було змодельовано кілька типів дронів, які різнилися за зовнішнім виглядом і функціональністю. Основна модель – це FPV-дрон із чотирма роторами, що відповідає реальним гоночним дронам, використаним як прототип. Вона була створена з урахуванням деталей конструкції: корпус, пропелери, камера, батарея, а також дрібні елементи, як-от антени та кріплення. Це дозволило досягти високої реалістичності візуального відображення, що є важливим для симуляційного типу гри.

Після створення моделі дрона увага була зосереджена на оточенні, яке формує атмосферу гри. Карта була розроблена у вигляді тренувального полігону, що містив такі елементи, як будівлі, дерева, перешкоди, цілі для ураження, а

також різноманітні об’єкти навколишнього середовища – скелі, вежі, дороги. Для кожного об’єкта застосовувалася модульна структура, що полегшувала процес розміщення сцен у Godot та дозволяла швидко змінювати конфігурацію рівня.

У процесі створення ландшафту було використано систему матеріалів і текстур, які допомагають відтворити різні типи поверхонь – ґрунт, метал, бетон, траву. З метою підвищення продуктивності були застосовані карти нормалей і baked-тіні, що імітують складне освітлення без додаткового навантаження на рушій.

Важливим аспектом стало також оптимізування моделей перед експортом у Godot. Для цього були проведені такі кроки:

- зменшення кількості полігонів без суттєвої втрати якості;
- об’єднання дрібних об’єктів у єдині меші;
- перевірка правильності нормалей і UV-розгортки;
- створення колізійних сіток для фізичної взаємодії дронів з об’єктами.

У результаті було отримано набір готових ігрових моделей, які гармонійно поєднуються між собою за стилем і масштабом. Завдяки цьому вдалося створити реалістичне середовище, що сприяє повному зануренню гравця в процес керування дроном. Blender став основним інструментом не лише для моделювання, а й для технічної підготовки контенту до подальшої інтеграції в рушій Godot, забезпечуючи високий рівень деталізації при оптимальному навантаженні на систему.

Після створення 3D-моделей дронів та оточення наступним ключовим етапом розробки стало впровадження системи фізики польоту дрона та керування ним у Godot. Основною метою цього підпункту було забезпечити реалістичну поведінку дрона в просторі та точне відтворення його рухів, що є критично важливим для симулятора FPV-польотів.

Для реалізації польоту дрона було обрано вузол CharacterBody3D, який дозволяє поєднувати фізику руху з контролем користувача, одночасно підтримуючи точні колізії та обмеження швидкості [2]. Використання CharacterBody3D забезпечило зручне управління дроном у тривимірному

просторі, дозволяючи моделювати його поведінку, враховуючи гравітацію, прискорення й інерцію.

Основні аспекти фізики польоту включали:

• підйом та тяга дрона: користувач може регулювати висоту польоту за допомогою дій «throttle up» та «throttle down». Система розраховує підйомну силу, враховуючи масу дрона та поточне прискорення, що дозволяє підтримувати стабільний політ на заданій висоті;

```
1. func _handle_lift(lift_input: float, delta: float) -> void:
2.     var effective_input = clamp(lift_input, 0.0, 1.0)
3.     var available_power = power * effective_input
4.
5.     if effective_input >= _min_lift_input:
6.         _current_vertical_velocity.y = available_power * pow(effective_input, 3) * (1 - abs(transform.basis.y.normalized().y - 1) * 2) / (mass * gravity)
7.     elif effective_input > 0:
8.         _current_vertical_velocity.y = -gravity + (gravity * (effective_input / _min_lift_input))
9.     else:
10.        _current_vertical_velocity.y -= gravity * delta
11.        if is_on_floor():
12.            _current_vertical_velocity.y = 0
13.
14.    velocity.y = lerp(velocity.y, _current_vertical_velocity.y, delta * 5.0)
```

•нахили та крени: об’єкт реагує на вхідні команди нахилу вперед-назад (pitch) і в сторони (roll), що забезпечує плавний поворот та контроль орієнтації у просторі. Нахили реалізовані через зміну локальної орієнтації CharacterBody3D, із врахуванням обмежень для уникнення неприродних позицій;

```
1. func _handle_movement(pitch_input: float, roll_input: float, delta: float) -> void:
2.     pitch_input = pow(pitch_input, 3)
3.     roll_input = pow(roll_input, 3)
4.
5.     var target_rotation = Vector3(deg_to_rad(pitch_input * 90), rotation.y, deg_to_rad(roll_input * 90))
6.     rotation = rotation.lerp(target_rotation, stability * delta)
7.
8.     var forward_direction = transform.basis.z.normalized()
9.     var right_direction = -transform.basis.x.normalized()
10.    var target_velocity = Vector3.ZERO
```

```
11.     target_velocity += forward_direction * pitch_input * (power *
sin(abs(rotation.x))) / mass
12.     target_velocity += right_direction * roll_input * (power * sin(abs(rotation.z))) /
mass
13.
14.     velocity.x = lerp(velocity.x, target_velocity.x, delta)
15.     velocity.z = lerp(velocity.z, target_velocity.z, delta)
```

- повороти навколо вертикальної осі (yaw): дрон обертається у горизонтальній площині відповідно до команд користувача, що дозволяє змінювати напрямок польоту без втрати стабільності.

```
1. func _handle_rotation(yaw_input: float) -> void:
2.     rotate_y(-yaw_input * rotation_speed)
```

Система також враховує інерцію та опір повітря, що дозволяє відтворити більш реалістичну фізику. Наприклад, при різкій зміні напряму дрон не змінює положення миттєво, а плавно повертається, як це відбувається з реальними FPV-моделями [19]. Такий підхід додає динаміки та відчуття реалістичного керування.

Важливою складовою є обробка колізій і взаємодія з оточенням. CharacterBody3D дозволяє визначати зіткнення дрона з об’єктами сцени, такими як будівлі, дерева або перешкоди на трасі. Система фізики реагує на удари, підтримуючи контроль користувача та моделюючи відскоки або падіння при сильних зіткненнях.

Для зручності керування та тестування різних сценаріїв польоту було додано регулювання параметрів руху, таких як максимальна швидкість, прискорення, сила підйому та чутливість до команд керування. Це дозволяє тонко налаштовувати поведінку дрона, забезпечуючи як більш реалістичний, так і більш ігровий стиль польоту залежно від режиму гри.

Крім того, було реалізовано систему стабілізації дрона, яка підтримує його горизонтальне положення під час зависання та невеликих маневрів. Це дозволяє гравцеві легше контролювати дрон під час польоту, знижуючи ймовірність випадкових падінь і забезпечуючи більш плавну симуляцію.

У підсумку, реалізація фізики польоту та керування через CharacterBody3D забезпечила інтерактивний і реалістичний досвід управління дроном. Гравець

отримує точний контроль над рухом, відчуття маси та інерції, а також стабільність під час польоту, що є ключовою вимогою для симулятора FPV. Така система слугує основою для подальшого впровадження ігрових режимів, рівнів та завдань, забезпечуючи відчуття реального польоту в обраному віртуальному середовищі.

Після налаштування фізики польоту та керування дроном наступним важливим етапом стало створення ігрових режимів, що формують безпосередній геймплей симулятора. Головною метою цього підпункту було реалізувати два базові режими: політ по маршруту та знищення цілей, які разом забезпечують різноманітність завдань і дозволяють користувачеві тренувати навички керування дроном у різних умовах.

Політ по маршруту

Цей режим орієнтований на точність та швидкість проходження дистанцій. Він передбачає, що гравець керує дроном, рухаючись за заздалегідь визначеним маршрутом, який складається з контрольних точок або «чекпоінтів». Кожен чекпоінт визначається у просторі сцени як невидимий тригер, при перетині якого система фіксує прогрес гравця.

Основна мета цього режиму – навчити користувача швидкому реагуванню на команди керування, правильному балансуванню нахилів і прискорення дрона. Для досягнення цього були створені траси різної складності: від прямих та коротких маршрутів для початкового рівня до складних трас із поворотами та зміною висоти для досвідчених користувачів.

Для підвищення ігрового інтересу додано систему таймеру й оцінки результату. Час проходження кожного маршруту фіксується, що дозволяє порівнювати досягнення гравця, а також відслідковувати прогрес у навчальних завданнях. Крім того, для полегшення орієнтації на складних трасах були додані візуальні підказки, які допомагають гравцеві тримати правильний курс без порушення занурення в симуляцію.

Знищення цілей (FPV-симуляція)

Другий режим має більш активний і динамічний характер. Він моделює FPV-сценарій, у якому гравець керує дроном для ураження цілей на карті. У

цьому випадку дрон оснащений умовним «озброєнням», яке імітує взаємодію з об’єктами цілей у просторі. Мета цього режиму – тренування координації рухів, швидкого реагування та точності при управлінні дроном у складних умовах.

Цілі розташовані на різній висоті та відстані, що дозволяє створювати різні рівні складності. Кожна ціль реалізована як спеціальний об’єкт із визначеними тригерами, що реагують на контакт із «пострілом» або взаємодію дрона. Після успішного ураження ціль змінює стан або зникає, що надає гравцеві зворотний зв’язок і стимулює досягати кращого результату.

Для обох режимів було реалізовано систему відновлення та повторного запуску, яка дозволяє гравцеві швидко перезапустити рівень у випадку падіння дрона або пропуску контрольної точки. Це підвищує комфорт і ефективність тренувального процесу, забезпечуючи безперервність геймплею та можливість повторювати спроби для покращення навичок.

Особливості реалізації геймплею

Обидва режими інтегруються із загальною системою фізики та керування, що забезпечує узгодженість і реалістичність рухів. Для досягнення максимального занурення важливою складовою стало налаштування відповідності швидкості дрона, інерції та маневреності у різних умовах польоту. Наприклад, на маршруті з перешкодами максимальна швидкість трохи зменшена для полегшення контролю, тоді як у режимі знищення цілей швидкість можна регулювати самостійно, підкреслюючи динамічність завдань.

Кожний режим має власну систему візуального та звукового зворотного зв’язку, яка допомагає гравцеві оцінювати ефективність польоту. У режимі FPV це, наприклад, підсвічування цілей або зміна кольору при успішному ураженні, а у режимі маршруту – підказки для проходження чекпоінтів.

Завдяки поєднанню цих двох режимів, симулятор забезпечує комплексне навчання та розвагу одночасно: гравець відпрацьовує точність польоту, координацію рухів, швидкість реакції та прийняття рішень у різних умовах. Така структура ігрових режимів формує основу для подальшого розширення симулятора і додавання нових завдань та сценаріїв польоту.

3.3 Реалізація інфраструктури й оптимізація проєкту

Одним із ключових компонентів симулятора стало впровадження системи користувачів, яка дозволяє гравцям створювати власні профілі, входити до гри та зберігати прогрес у хмарі. Це забезпечує персоналізований досвід, можливість продовжувати гру на різних пристроях та відстежувати досягнення в різних ігрових режимах.

Для реалізації системи користувачів використано можливості Firebase, що дозволяють організувати авторизацію, реєстрацію та збереження даних у хмарі. Основна структура роботи системи передбачає три ключові етапи.

1. *Реєстрація користувача* (рис. 3.5). На першому етапі новий гравець створює акаунт, заповнюючи форму з основними даними – електронною поштою та паролем. Після підтвердження введеної інформації система перевіряє її коректність і надсилає запит до Firebase, де створюється унікальний запис користувача в базі даних. Кожен запис містить персональні дані, а також базові параметри прогресу – пройдені рівні, результати польотів, досягнуті цілі та налаштування.

Рис. 3.5. Інтерфейс панелі реєстрації

2. *Авторизація користувача*. Для входу у гру гравець вводить свої облікові дані, після чого система перевіряє їх через Firebase. У разі успішної авторизації відбувається завантаження персональної інформації та стану прогресу з бази даних. Це забезпечує безперервність гри незалежно від пристрою, на якому

запускається симулятор, і дозволяє користувачу відновити останній стан гри після перерви.

3. *Збереження прогресу у хмарі*. Після кожного проходження рівня або зміни стану гри, дані користувача автоматично зберігаються у Firebase. Це включає інформацію про виконані завдання, час проходження маршруту, досягнення у режимі FPV та налаштування управління дроном. Система реалізована так, щоб оновлення даних відбувалося у реальному часі, забезпечуючи актуальний стан гри для будь-якої сесії.

У результаті реалізації цієї системи кожен гравець отримує персоналізований досвід гри, де прогрес зберігається незалежно від платформи, а взаємодія з грою стає більш гнучкою і комфортною. Система користувачів створює фундамент для подальшого розвитку симулятора, включаючи статистику досягнень, рейтинги та можливість інтеграції нових соціальних функцій.

Після створення основних моделей, інтеграції фізики та розробки ігрових режимів, наступним важливим етапом стала оптимізація продуктивності рівнів та візуального оформлення сцени. У симуляторах FPV, де швидкість реакції та точність управління критично впливають на геймплей, надмірне навантаження на систему може призвести до падіння кадрів, спотворення руху дрона або затримок у відображенні сцени. Тому на цьому етапі було необхідно збалансувати якість графіки та продуктивність рушія.

Оптимізація моделей та текстур

Першим кроком було зменшення складності моделей без втрати візуальної якості. Основні методи оптимізації включали:

- зменшення кількості полігонів на другорядних об’єктах, таких як дерева, будівлі та деталі оточення;
- об’єднання дрібних елементів у єдині меші для зменшення кількості draw calls;
- використання LOD (Level of Detail) для моделей дронів та об’єктів оточення, що автоматично зменшувало деталізацію на віддалених від камери об’єктах.

Текстури також були оптимізовані. Для великих об’єктів застосовувалися спрайтові та baked-текстури, що дозволяє зберегти деталізацію без надмірного використання відеопам’яті. Для дрібних або віддалених об’єктів використовувалися текстури меншого розміру та меншої глибини кольору, що значно знижувало навантаження на систему.

Оптимізація сцени та рендеру

Другим важливим напрямком стало налаштування сцени в Godot. Для цього були застосовані такі підходи:

- групування об’єктів у сцени та інстанси: великі рівні розбито на окремі підсцени, що дозволяє завантажувати та відображати лише потрібні ділянки карти;
- використання Occlusion Culling: приховування об’єктів, що не потрапляють у поле зору камери, що зменшує кількість обчислень для рендеру [8];
- регулювання параметрів освітлення: замість динамічного глобального освітлення для всіх об’єктів застосовувалися статичні lightmaps та комбіновані джерела світла, що створюють реалістичну атмосферу без зайвого навантаження на процесор.

Оптимізація фізики та взаємодії

Також було виконано оптимізацію фізичних обчислень. Для цього:

- обмежено кількість активних колізійних об’єктів у сцені;
- визначено пріоритети обробки колізій: дрони та цілі мають високий пріоритет, другорядні об’єкти – низький;
- використано спрощені колізійні сітки для складних моделей, що значно зменшує обчислювальне навантаження без втрати точності взаємодії.

Візуальне оформлення та баланс продуктивності

Для збереження високої якості візуального оформлення при оптимізації продуктивності було застосовано комбінований підхід до матеріалів і ефектів:

- використовувалися прості PBR-матеріали для основних моделей;

•детальні шейдери застосовувалися лише для об’єктів, що найбільш помітні для гравця;

•для створення атмосфери та глибини застосовувалися декоративні елементи, які незначно впливають на продуктивність, наприклад, частково статичні ефекти пилу або диму.

У результаті проведеної оптимізації досягнуто:

•стабільна частота кадрів під час польоту дрона на всіх тестових рівнях;

•збереження високої якості візуального оформлення;

•плавний та реалістичний геймплей без затримок управління.

Таким чином, оптимізація продуктивності та візуальної частини сцени забезпечила баланс між реалістичним середовищем, точністю симуляції та комфортом користувача, що є критично важливим для тренувального FPV.

Після завершення всіх етапів розробки симулятора дронів важливим заключним кроком стало тестування та налагодження проєкту. Цей етап забезпечує стабільність, коректність роботи всіх компонентів та комфортний геймплей для користувача. Тестування проводилося на декількох рівнях: перевірка функціональності, фізики, геймплейних режимів та оптимізації продуктивності.

Тестування фізики та керування дроном

Першим етапом було перевірено реалістичність польоту дрона та точність системи керування. Тестувалося:

•коректне реагування дрона на команди користувача (підйом, крен, нахили, обертання);

•поведінка під час різких маневрів, зокрема перевірка інерції та стабілізації;

•взаємодія з об’єктами оточення та відпрацювання колізій.

Виявлені неточності, наприклад занадто різкі повороти або нестабільне зависання на місці, були відкориговані шляхом налаштування параметрів фізики, таких як сила підйому, опір повітря та прискорення.

Перевірка геймплейних режимів

Другим важливим напрямком було тестування ігрових режимів – політ по маршруту та знищення цілей. Ключові аспекти перевірки включали:

- коректне проходження чекпоінтів на маршруті, фіксація часу проходження та відображення прогресу;
- взаємодія з цілями у FPV-режимі та коректне спрацьовування тригерів при «ураженні»;
- стабільна робота таймерів, підрахунок результатів та оновлення прогресу користувача у хмарі Firebase.

У ході тестування були виявлені дрібні помилки, такі як пропуск деяких тригерів або затримки в оновленні прогресу. Вони були усунені через корекцію логіки обробки подій у сценах та налаштування пріоритетів обчислень.

Оптимізація продуктивності та налагодження

На фінальному етапі тестування особлива увага приділялася продуктивності та стабільності сцени. Перевірялися:

- частота кадрів на різних рівнях та при різній кількості об’єктів;
- поведінка LOD та Occlusion Culling для забезпечення плавності рендеру;
- швидкість завантаження сцен та відновлення прогресу користувача після запуску гри.

Завдяки цьому було виявлено та усунуто незначні проблеми з падінням кадрів на великих рівнях та оптимізовано ресурси для плавного відображення фізики польоту дрона та об’єктів оточення.

Висновки щодо тестування

Після всебічного тестування та налагодження симулятор дронів:

- коректно обробляє керування та фізику польоту;
- стабільно працює обидва геймплейні режими;
- зберігає та відновлює прогрес користувача у хмарі;
- підтримує високу продуктивність навіть на складних рівнях.

Таким чином, проведене тестування підтвердило готовність проєкту до використання, забезпечивши стабільну, реалістичну та комфортну симуляцію польоту дронів для навчання та розваг.

Також під час проходження виробничої практики в компанії ТОВ «ТРЕКЛАМ» було проведено фахові консультації з розробниками підприємства щодо концепції та технічної реалізації проєкту, який розробляється в межах даної магістерської роботи. У ході обговорень особливу увагу було приділено питанням архітектури програмного забезпечення, оптимізації продуктивності 3D-сцен, організації взаємодії між ігровими модулями, а також практичним аспектам використання хмарних сервісів для збереження даних користувачів. Отримані рекомендації мали прикладний характер і були враховані на етапі доопрацювання проєкту, що сприяло підвищенню його стабільності, масштабованості та відповідності сучасним вимогам до програмних продуктів у сфері інтерактивних 3D-симуляторів.

Висновки для третього розділу

Реалізація інтегрованого 3D-симулятора дронів продемонструвала ефективність сучасних підходів до розробки ігор і навчальних моделей у віртуальному середовищі. У процесі роботи було забезпечено створення реалістичних тривимірних моделей дронів та елементів оточення, що поєднують високий рівень деталізації з оптимізованою структурою для забезпечення стабільної продуктивності. Реалізація фізики польоту та системи керування через CharacterBody3D дозволила досягти точного відтворення поведінки дронів у просторі, забезпечивши природну інерцію, стабілізацію та взаємодію з об’єктами середовища, що є важливим для симуляційного досвіду.

Створення різноманітних ігрових режимів забезпечило комплексний підхід до розвитку навичок користувача, поєднуючи завдання з точності та швидкості польоту з динамічними сценаріями у стилі FPV. Впровадження системи користувачів з можливістю авторизації, реєстрації та збереження прогресу у хмарі забезпечило персоналізовану взаємодію з симулятором, гарантуючи безпеку даних та можливість відновлення гри на різних пристроях.

Значна увага приділялася оптимізації продуктивності та візуального оформлення сцени, що дозволило зберегти високий рівень графічної деталізації при стабільній частоті кадрів. Використання методів оптимізації моделей, текстур, LOD, Occlusion Culling та спрощених колізійних сіток забезпечило баланс між реалістичністю середовища й ефективністю роботи рушія. Проведене тестування та налагодження підтвердило коректність фізики польоту, стабільність ігрових режимів і надійність системи користувачів.

Загалом, досягнуті результати демонструють можливість комплексного поєднання тривимірного моделювання, реалістичної фізики, інтерактивного геймплею та хмарних технологій для створення ефективного та продуктивного середовища для навчання і тренування навичок управління дронами, а також підтверджують практичну доцільність використання сучасних інструментів геймдизайну в розробці навчальних симуляторів.

ЗАГАЛЬНІ ВИСНОВКИ

. У ході виконання кваліфікаційної роботи було комплексно досліджено процес створення інтегрованого 3D-симулятора на основі сучасних технологій геймдизайну, що поєднує інструменти Godot Engine, Blender і хмарні сервіси Firebase. Проведений аналіз сучасних рушіїв, методів моделювання та підходів до побудови симуляційних систем дав змогу визначити оптимальну комбінацію технологій для реалізації ефективного, продуктивного та масштабованого програмного продукту.

У результаті дослідження було підтверджено, що використання відкритих інструментів – Godot та Blender – дозволяє забезпечити повний цикл розробки 3D-симулятора: від створення візуального контенту до реалізації інтерактивної фізики та логіки геймплею. Godot продемонстрував високу гнучкість у роботі з тривимірними сценами, фізичною моделлю, анімацією та системами керування, тоді як Blender забезпечив можливість створення деталізованих моделей із подальшою оптимізацією для реального часу.

Практична реалізація симулятора показала ефективність інтеграції різних технологічних компонентів. Створені режими польоту – маршрути з чекпоінтами та сценарії FPV – дозволили сформувати багатофункціональне середовище для тренування навичок керування дроном. Налаштована модель фізики польоту забезпечує реалістичну поведінку апарата, включно з інерційністю, маневровістю та взаємодією з оточенням. Інтеграція Firebase дала змогу реалізувати систему користувачів, збереження прогресу та хмарну синхронізацію, що підвищує зручність і масштабованість застосунку.

Проведене тестування підтвердило стабільність роботи розробленої системи, коректність механік, надійність хмарних сервісів і достатню продуктивність проєкту навіть у складних сценах. Оптимізація моделей, рельєфу та логіки рендерингу забезпечила плавність роботи симулятора в режимі реального часу. Загалом виконана робота продемонструвала доцільність використання відкритих інструментів у побудові навчальних ігрових систем та показала, що поєднання Godot, Blender і хмарних технологій може стати ефективною основою для створення сучасних симуляторів. Отримані результати підтвердили поставлену мету й завдання дослідження, а також створили технічне підґрунтя для подальшого розвитку проєкту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Autodesk 3ds Max Documentation URL: https://help.autodesk.com/view/3DSMAX/2025/UKR/ (дата звернення: 11.06.2025).

2. Blender Python API URL: https://docs.blender.org/api/current/ (дата звернення: 15.03.2025).

3. CharacterBody3D in Godot URL: https://docs.godotengine.org/en/stable/classes/class_characterbody3d.html (дата звернення: 22.07.2025).

4. Firebase Documentation URL: https://firebase.google.com/docs (дата звернення: 08.05.2025).

5. GLTF 2.0 Specification URL: https://www.khronos.org/gltf/ (дата звернення: 12.01.2025).

6. Godot 4.0 3D Rendering URL: https://docs.godotengine.org/en/stable/tutorials/3d/ (дата звернення: 30.09.2025).

7. Godot Engine URL: https://godotengine.org/ (дата звернення: 04.06.20250.

8. Godot Firebase URL: https://github.com/RandomNinjaAtk/GodotFirebase (дата звернення: 19.08.2025).

9. Godot GDExtension URL: https://docs.godotengine.org/en/stable/tutorials/scripting/gdextension/ (дата. звернення: 27.02.2025).

10. GDScript Reference URL: https://docs.godotengine.org/en/stable/tutorials/scripting/gdscript/ (дата звернення: 14.10.2025).

11. Blender URL: https://www.blender.org/ (дата звернення: 21.04.2025).

12. Maya User Guide URL: https://knowledge.autodesk.com/support/maya (дата звернення: 09.12.2024).

13. MIT License for Godot URL: https://godotengine.org/license/ (дата звернення: 17.09.2025).

14. OpenGL Programming Guide : the official guide to learning OpenGL / D. Shreiner [та ін.]. 9th ed. Addison-Wesley, 2019. 1056 p.

15. Terrain3D Plugin for Godot URL: https://github.com/TokisanGames/Terrain3D (дата звернення: 03.07.2025).

16. Unity Manual URL: https://docs.unity3d.com/Manual/index.html. Дата звернення: 25.01.2025.

17. Адоніна А. М. Blender 3D. Повне керівництво. Київ : BookShops, 2023. 450 с.

18. Бичківський О. О. Міжнародне право : підручник. Львів : ЛНУ, 2020. 320 с.

19. Булаєнко М. Як створюється FPV-симулятор. GameDev DOU. 2025. URL: https://gamedev.dou.ua/articles/fpv-battleground-interview/ (дата звернення: 28.03.2025).

20. Вивчаємо Blender : практичний посібник зі створення 3D-сцен. Київ : Читайка, 2024. 380 с.

21. Зінчук Т. О. Переваги та недоліки ігрових рушіїв. Наука НГУ. 2022. № 1. С. 87–95.

22. Іваненко Д. Г. Фізика в ігрових рушіях Godot та Unity. Праці конференції "IT в освіті". 2024. С. 112–120.

23. Кот Ю. П. Основи 3D-графіки та анімації : монографія. Київ : НАУ, 2020. 256 с.

24. Лісовий В. М. Програмування ігрових симуляторів : підручник. Львів : ЛНУ ім. І. Франка, 2022. 180 с.

25. Лялюк О. Ю. Основи організації і діяльності місцевих рад в Україні. Київ : НУОУ, 2021. 200 с.

26. Островка Д. В. Інформаційна технологія синтезу тривимірного зображення користувача для мобільних систем доповненої реальності : дис. ... канд. техн. наук. Львів : ЛПНУ, 2023. 180 с.

27. Петренко О. М. VFX у Blender : монографія. Одеса : ОНУ, 2023. 150 с.

28. Пономаренко А. С. 3D-моделювання в Blender для ігор. Харків : ХНУРЕ, 2021. 210 с.

29. Турута О. В. Інформаційні технології в освіті. Харків : ХНУРЕ, 2022. 250 с.

30. Шепель О. В. Open-source технології в геймдеві. Вісн. ТНПУ. Комп'ют. науки. 2023. № 2. С. 45–52.