ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ТЕХНОЛОГІЙ ЦИФРОВОГО РОЗВИТКУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Цифровий пізнавально-розважальний мобільний застосунок на основі Unreal Engine»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Технології цифрового розвитку»

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

Виконав: здобувач вищої освіти групи ТЦР-41

Владислав ГРИСИК

Керівник: доктор філософії Микола ГЕРЦЮК

Рецезент: _____

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Технологій цифрового розвитку

Ступінь вищої освіти «Бакалавр»

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Технологій цифрового розвитку»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Вікторія ЖЕБКА
«___» _____2026р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Грисику Владиславу Олександрович

1. Тема кваліфікаційної роботи: «Цифровий пізнавально-розважальний мобільний застосунок на основі Unreal Engine»
керівник кваліфікаційної роботи доктор філософії, Микола ГЕРЦЮК
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 року № 51.

2. Строк подання кваліфікаційної роботи «19» травня 2026 року

3. Вихідні дані до кваліфікаційної роботи: теоретичні та практичні матеріали з розробки runner-ігор, документація ігрового рушія Unreal Engine 5 та мови програмування C++.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
    1. Основні підходи до розробки ігор жанру runner та їх особливості.
    2. Проектування та архітектура інтерактивного ігрового середовища у Unreal Engine.
    3. Реалізація ігрових механік раннера та інтеграція пізнавального контенту про динозаврів у ігровий процес.

4. Тестування, оптимізація та підвищення якості користувацького досвіду в розробці мобільної гри.

5. Перелік ілюстративного матеріалу: *презентація*
    1. Актуальність теми та мета роботи
    2. Основні завдання для досягнення мети
    3. Аналіз аналогів
    4. Об’єкт та предмет дослідження
    5. Компонентна діаграма
    6. Методи дослідження
    7. Функціональні вимоги застосунку
    8. Нефункціональні вимоги
    9. Підтвердження роботи вимог
    10. Тестування застосунку
    11. Робота застосунку
    12. Апробація результатів дослідження.
    13. Висновок та подальші кроки

6. Дата видачі завдання «20» лютого 2026 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1.	Підготовчий етап: Збір інформації, аналіз літературних джерел, огляд жанру раннерів та ігор про динозаврів	21.02 – 25.02.2026	Виконано
2.	Формування мети та завдання роботи, визначення об’єкта і предмета дослідження	26.02 – 08.03.2026	Виконано
3.	Системи управління ігровими механіками та інтерактивними об’єктами	09.03 – 15.03.2026	Виконано
4.	Інтеграція механіки кількості м’яса та ймовірність купівлі іншого динозавра	16. 03 – 25.03.2026	Виконано
5.	Розробка графічних ресурсів для персонажів і середовища	26.03 – 07.04.2026	Виконано
6.	Тестування ігрового процесу, усунення помилок та оптимізація продуктивності	08.04 – 20.04.2026	Виконано
7.	Вступ, висновки, підготовка реферату та пояснювальної записки	21.04 – 27.04.2026	Виконано
8.	Розробка обов’язкових демонстраційних матеріалів	28.04 – 05.05.2026	Виконано

Здобувач вищої освіти _____ Владислав ГРИСИК

(підпис)

Керівник
кваліфікаційної роботи _____ Микола ГЕРЦЮК

(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 50 стор., 12 табл., 19 рис., 22 джерел.

Мета роботи – підвищення пізнавального контенту про доісторичний світ у жанрі runner для платформи Android.

Об’єкт дослідження – процес розробки мобільних ігрових застосунків жанру runner.

Предмет дослідження – методи і засоби розробки мобільних ігрових застосунків.

Короткий зміст роботи: у кваліфікаційній роботі проведено аналіз жанру runner та сучасних засобів розробки ігор. Вибір ігрового рушія Unreal Engine, як основної платформи розробки, був обґрунтований. Розроблено мобільну гру DinoRunner, визначено основні ігрові сценарії та реалізовано ключові ігрові механіки й системи. Реалізовано систему керування персонажем, процедурну генерацію перешкод та інтерфейс користувача. Особливо увагу було приділено впровадження пізнавального контенту про динозаврів в ігровий процес. Проведено тестування застосунку для мобільних пристроях різного типу.

Сферою використання є розробка мобільних ігор, освітньо-розважальні цифрові продукти, а також інтерактивні навчальні застосунки.

КЛЮЧОВІ СЛОВА: МОБІЛЬНА ГРА, RUNNER, UNREAL ENGINE, BLUEPRINTS, ДИНОЗАВРИ, ІГРОВІ МЕХАНІКИ, ПІЗНАВАЛЬНИЙ КОНТЕНТ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

UE – Unreal Engine

BP – Blueprints

GM – Game Mode

PC – Player Controller

WBP – Widget Blueprint

M – Material

MF – Material function

IMC – Input Mapping Context

IA – Input Action

GameDev – Game development (розробка ігор)

ПК – Персональний комп’ютер

ВСТУП

Актуальність теми. Ринок мобільних ігор є дуже динамічним та зростаючим, а жанр раннерів особливо популярний завдяки своїй доступності та коротким ігровим сесіям. Розробка DinoRunner, гри, яка поєднує раннер з інформацією та цікавими фактами про доісторичний світ, відповідає сучасним тенденціям в інноваційному дизайні ігор та цифровій освіті.

Мета роботи полягає у підвищенні пізнавального контенту про доісторичний світ у жанрі runner для платформи Android.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Проаналізувати особливості жанру runner та сучасні програмні засоби розробки мобільних ігор.

2. Обґрунтувати вибір ігрового рушія Unreal Engine як основного інструменту розробки.

3. Створити вимоги та спроектувати архітектуру застосунку DinoRunner, визначити сценарії використання та основні ігрові механіки.

4. Реалізувати систему керування персонажем, механізми процедурної генерації перешкод, інтерфейс користувача та модуль інтеграції пізнавального контенту.

5. Провести тестування застосунку та виконати оптимізацію його продуктивності мобільних пристроїв.

Об'єкт дослідження – процес розробки мобільних ігрових застосунків жанру runner із використанням сучасних ігрових рушіїв.

Предмет дослідження – модульні архітектурні рішення, план використання застосунку, проектування інтерфейсу користувача, алгоритми процедурної генерації рівнів, механіки керування персонажем та інтеграція освітнього контенту.

Методи дослідження. У процесі виконання роботи використано такі методи:

- порівняльний аналіз – для дослідження існуючих програмних рішень і вибору технологій;

- UML-моделювання – для формалізації сценаріїв використання;

- створення більшості застосунку на базі системи Blueprints;

- функціональне та продуктивнісне тестування – для перевірки коректності роботи застосунку та його оптимізації.

Галузь застосування – розробка мобільних ігрових застосунків, освітньо-розважальні цифрові продукти, інтерактивні навчальні системи для молодіжної аудиторії.

Розроблений застосунок може використовуватися як демонстраційний або портфельний проєкт, а також як приклад реалізації сучасних підходів до геймдизайну.

Практична новизна роботи основна у поєднанні отримання нової пізнавальної інформації у розважальному застосунку створеному із використанням ігрового рушія Unreal Engine.

Практична значущість є у розробленому мобільному застосуноку DinoRunner, який являє собою повнофункціональну гру з пізнавальним контентом для платформи Android, створену із використанням ігрового рушія Unreal Engine.

1 Аналіз проблеми та існуючих засобів її вирішення

1.1 Опис предметної галузі: сучасні ігри жанру Runner. Загальні відомості про жанр

Жанр runner характеризується мінімалістичним геймплеєм, у якому гравець керує персонажем, що безперервно рухається вперед, долаючи перешкоди.

Суть гри зводиться до одного: персонаж постійно рухається вперед, а гравець повинен встигнути реагувати на перешкоди, збираючи винагороди, якомога довше залишатися живим. Саме в цьому полягає ефективність мінімалістичного дизайну.

Схожі ігри існували і раніше, ще в епоху аркадних автоматів, але стрімке зростання популярності жанр пережив разом із появою смартфонів. Простота керування виявилась оптимальною для сенсорних екранів, а короткі ігрові сесії відповідають темпу повсякденного використання мобільних пристроїв.

Гравець, як правило, може робити лише кілька речей:

- стрибати;
- ухилятися від перешкод;
- прискорюватися;
- підбирати бонусні предмети.

Метою гравця є подолання максимальної відстані та набір найбільшої кількості очок. Доступність правил гри у поєднанні з інтенсивним темпом ігрового процесу забезпечує жанру runner високу популярність серед великого кола користувачів.

Із розвитком мобільних технологій runner-ігри зазнали значних якісних змін. Сучасний етап розвитку жанру характеризується переходом від спрощеної схематичної візуалізації до високотехнологічного графічного виконання, що включає детальну анімацію та розгалужені ігрові всесвіти з власним наративом.

Сучасні ігри жанру runner значно відрізняються від своїх перших представників завдяки розвитку мобільних технологій та зростанню вимог гравців. Вони поєднують деталізовану графіку, плавну анімацію персонажів і динамічний ігровий процес, що забезпечує комфортне та захопливе проходження. У багатьох проєктах використовується процедурна генерація рівнів, завдяки якій кожна нова спроба є унікальною та не повторює попередню. Для підтримання інтересу гравців додаються різноманітні бонуси, предмети та тимчасові покращення, які урізноманітнюють проходження та впливають на стиль гри. Крім цього, сучасні runner-ігри часто містять системи досягнень, рейтинги та таблиці рекордів, що створює змагальний елемент між користувачами. Важливою складовою також є тематичне оформлення світів, наявність унікальних персонажів і власної атмосфери, які формують впізнаваний стиль гри. Типовими представниками жанру є Subway Surfers[1] та Temple Run[2].

Незважаючи на індивідуальні особливості, кожен із цих проєктів демонструє важливу закономірність: використання обмеженого набору механік дозволяє створити масово популярний продукт із високим рівнем утримання аудиторії.

Ринок мобільних ігор демонструє стабільне збільшення інтересу до жанру runner, що підтверджує його високу адаптивність до специфіки портативних пристроїв. Жанр вирізняється високим рівнем універсальності, оскільки демонструє рівномірне охоплення та позитивне сприйняття серед представників різних демографічних і соціальних груп.

Основною аудиторією ігор жанру runner є користувачі, які надають перевагу динамічному та доступному ігровому процесу. До неї належать казуальні гравці, для яких важливими є короткі, але насичені ігрові сесії без складного та тривалого освоєння механік. Значну частину аудиторії також становлять молоді люди, яких приваблюють швидкий темп гри та зрозуміле керування. Окремою категорією є шанувальники мобільних аркад, для яких головною мотивацією стає прагнення покращити власний результат, встановити новий рекорд і зайняти вищі позиції у таблицях лідерів.

Популярність ігор цього жанру пояснюється декількома ключовими факторами. Одним із них є процедурна генерація перешкод і рівнів, завдяки якій кожне проходження відрізняється від попереднього та підтримує інтерес користувача до повторних спроб. Додатково високий темп ігрового процесу вимагає постійної концентрації уваги, що створює відчуття напруження та залучення до гри. Саме поєднання швидкої реакції, постійної динаміки та бажання перевершити попередній результат формує стійкий інтерес гравців до жанру runner.

Освітній та когнітивний потенціал

Попри уявну простоту, ігри жанру runner формують ряд когнітивних навичок. Дослідження у галузі game-based learning[3] підтверджують, що динамічний характер ігрового процесу сприяє розвитку таких здібностей:

- увагу та концентрацію – адже перешкоди з'являються раптово і часу на роздуми майже немає;

- швидкість прийняття рішень – у динамічному темпі гри доводиться діяти миттєво, покладаючись на реакцію;

- координацію рухів – точне керування персонажем стає вирішальним на складних ділянках рівня.

Окремі ігри йдуть ще далі й додають пізнавальні елементи прямо в ігровий процес – цікаві факти, інформаційні підказки, тематичні деталі світу. Включення освітніх елементів у ігровий контекст сприяє ненавмисному засвоєнню інформації в умовах позитивного емоційного стану користувача, що підвищує ефективність сприйняття.

Зазначений підхід реалізовано в застосунку DinoRunner шляхом інтеграції верифікованих пізнавальних фактів про динозаврів, що відображаються безпосередньо в ігровому процесі.

Соціокультурний вплив

Сучасні відеоігри зазнали суттєвої функціональної трансформації, еволюціонувавши з інструментів суто розважального характеру в медіум для трансляції соціокультурних парадигм та популяризації наукових знань. Згідно з

концепцією Джеймса Пола Гі [4], ігрове середовище забезпечує умови для нативної інтеграції освітнього контенту, що сприяє формуванню пізнавального інтересу без вираженого дидактичного тиску. Його концепція в галузі Game-based Learning [3] підтверджує, що імерсивність ігрового процесу дозволяє користувачеві отримувати комплексний когнітивний досвід, де засвоєння інформації відбувається органічно в межах ігрової логіки, що значно розширює межі традиційного користувацького сприйняття.

Деякі проєкти беруть за основу історичні або наукові сеттинги, бо інтерактивний формат сприймається набагато живіше, ніж текст у підручнику. Найкращими прикладами є:

- Assassin’s Creed: Discovery Tour (Ubisoft) [5]: Окремий освітній режим, який перетворює ігровий світ на інтерактивний музей. Користувачі можуть досліджувати архітектуру та побут Стародавнього Єгипту чи Греції без бойових механік.

- Foldit [6]: Експериментальний ігровий проєкт, де гравці допомагають розв'язувати реальні наукові завдання з біохімії (згортання білків), що демонструє можливість залучення геймерів до справжніх наукових досліджень.

- Ancestors: The Humankind Odyssey [7]: Гра, що реконструює процес еволюції ранніх предків людини, фокусуючись на біологічному розвитку та виживанні в умовах дикої природи Африки мільйони років тому.

Перспективи та тенденції розвитку жанру ігор «раннер»

Незмінна актуальність жанру «раннер» пояснюється високою адаптивністю до технологічних змін у галузі. На основі аналізу сучасних тенденцій[8][9] можна виділити кілька перспективних напрямків розвитку цього сегмента відеоігор:

Інтеграція технологій VR та AR: Інтеграція технологій VR та AR забезпечує новий рівень імерсивності для користувача, надаючи розробникам інструменти для формування принципово нових форматів взаємодії з ігровим середовищем та замінити традиційний ігровий процес на високоінтерактивне середовище з відчуттям присутності користувача у новому створеному світі.

Конвергенція жанрів: спостерігається тенденція до об’єднання Runner-ів з елементами інших жанрів, особливо сюжетних, рольових (RPG) та пригодницьких ігор. Такі доповнення дозволяють розширити глибину ігрового досвіду, не ускладнюючи базову механіку управління та керування персонажем.

Застосування систем штучного інтелекту (ШІ): Використання ШІ для динамічного регулювання складності дозволяє персоналізувати ігровий процес відповідно до ситуації конкретного користувача та створити баланс між викликом та доступністю.

Резюмуючи факти у тексті, можна стверджувати, що жанр «ранер» має хороший потенціал. Його подальший розвиток відбуватиметься шляхом технологічного ускладнення внутрішніх систем та доданням різних доповнень при збереженні фундаментальної концептуальної простоти.

1.2 Аналіз існуючих runner-ігор з елементами навчання

Перед розробкою DinoRunner було проведено аналіз існуючих мобільних додатків, що містять освітні або пізнавальні елементи зі схожою тематикою. Мета аналізу – виявити сильні сторони конкурентів, визначити незаповнені ніші та сформулювати конкурентні переваги власного продукту.

Dino Run (PixelJAM) [10]

Dino Run є одним із відомих представників жанру runner з тематикою динозаврів. Застосунок характеризується мінімалістичним графічним виконанням та обмеженим набором ігрових механік. Навколо динозавра лише небо та хмари. Освітній характер тут також у мінімальному вигляді: гра лише ознайомлює з назвами динозаврів але не надає жодної змістовної інформації про тварин. Перевагою є тематична цілісність або атмосфера, а недоліком – практична відсутність пізнавального контенту про тварин.

Endless Alphabet (Originator) [11]

Незважаючи на те, що Endless Alphabet не належить до жанру runner, цей застосунок є показовим прикладом успішного поєднання ігрового та навчального

контенту. Застосунок навчає словниковому запасу через міні ігри, але розрахований лише на частину малої вікової групи.

Jurassic World Alive (Ludia) [12]

Гра із елементами колекціонування динозаврів. Хоча це не runner, але вона є прямим конкурентом за цільовою аудиторією та тематикою. У грі містяться детальні картки з науковою інформацією про кожен вид динозавра, тривимірні моделі та статистику. Проте складність ігрового процесу та значні часові витрати орієнтують застосунок на більш зрілу аудиторію з відповідним рівнем залучення.

Dinosaur game (Google Chrome) [13]

Вбудована гра браузерів та мабуть, один із найвідоміших runner-ів у світі. Незважаючи на мінімалістичний дизайн і повну відсутність освітнього контенту, вона демонструє, наскільки ефективною може бути механіка runner. Мільярди сеансів та дуже багато часу гри підтверджують: гравці охоче повертаються до простої, але динамічної логіки та механіки.

Порівняльний аналіз

Зведену порівняльну характеристику розглянутих застосунків наведено в таблиці 1.1.

Таблиця 1.1

Порівняльний аналіз існуючих runner-ігор з елементами навчання

Застосунок	Жанр	Тематика	Освітній контент	Платформа
Dino Run	Runner	Динозаври	Мінімальний	PC/Web
Endless Alphabet	Навчальний	Алфавіт	Повний	iOS/Android
Jurassic World Alive	AR/Колекціонування	Динозаври	Повний	iOS/Android
Google Dino	Runner	Динозавр	Відсутній	Web
DinoRunner	Runner	Динозаври	Інтегрований	Android

За результатами аналізу можна зробити висновок, що на ринку мобільних ігор існує майже незаповнена ніша: runner-ігри з доісторичною тематикою. Вони або зовсім не містять освітнього контенту (Google Dino[13]), або пропонують тематику тварин без пізнавальної складової (DinoRun[10]). Застосунки з повноцінним освітнім контентом про динозаврів (Jurassic World Alive[12]) не є runner-іграми і орієнтовані на більш складний ігровий досвід.

1.3 Аналіз існуючих програмних продуктів для розробки ігор

Протягом останніх років спостерігається розвиток процесів розробки відеоігор, що зумовлено стрімкою еволюцією можливостей сучасних рушіїв для створення ігор або схожих додатків. Сучасні ігрові рушії надають розробникам готову екосистему для роботи з графікою, фізикою та анімацією, що дозволяє зосередити ресурси на проектуванні ігрової логіки та концептуальних рішень.

Усі існуючі інструменти – це універсальні рушії, такі як Unreal Engine[14], Unity [15] та Godot Engine[16], які добре працюють з різними проєктами.

Нижче розглянуто найпоширеніші інструменти, які реально використовуються при розробці runner-ігор – у тому числі таких, як DinoRunner.

Unreal Engine [14] – це безкоштовний рушій та один із найпотужніших рушіїв на ринку. Його репутація заслужена: рушій забезпечує дуже хорошу якість графіки і при цьому залишається достатньо гнучким для різних типів проєктів.

Основна мова розробки – C++, також існує система візуального скриптингу Blueprints. Вона дозволяє будувати ігрову логіку через блок-схеми, що суттєво знижує поріг входу. Крім того, UE має власний магазин, де можна знайти готові моделі, анімації, звуки та плагіни, що значно прискорює розробку.

Серед переваг варто відзначити високу продуктивність, підтримку одразу кількох платформ – ПК, мобільні пристрої, консолі – а також поєднання 2D і 3D елементів в одному проєкті. До недоліків відносяться висока складність вивчення та підвищені вимоги до оптимізації проектів для мобільних платформ.

Godot [16] – це безкоштовний рушій із відкритим вихідним кодом, який останніми роками стрімко набирає популярність серед інді-розробників. Він пропонує окремі, добре оптимізовані системи для 2D і 3D графіки, що робить його привабливим для різних проєктів.

Основна мова синтаксично схожа на Python, що робить її доступною для новачків. Паралельно підтримуються C# та C++ для тих, хто потребує більшої гнучкості.

Перевагами Godot є компактний редактор, висока швидкість роботи та повна відкритість вихідного коду. Мінуси теж є – спільнота поки менша, ніж у інших рушіїв, а бібліотека готових ресурсів і плагінів ще не така розгалужена.

Unity [15] – один із найпопулярніших інструментів для створення саме мобільних ігор. Він добре підходить для 2D і 3D проєктів, має величезну екосистему та активну спільноту розробників по всьому світу.

Основна мова – C#. Крім того, Unity має власний магазин ресурс, де можна знайти готові моделі, анімації, звуки та плагіни, що значно прискорює розробку.

До переваг належать універсальність, широка підтримка платформ і величезна кількість навчальних матеріалів. Серед недоліків – відносно високі вимоги до заліза і те, що для реалізації складної логіки без знань програмування не обійтися.

1.3.1 Недоліки та обмеження існуючих рішень

Аналіз наведених ігрових рушіїв показує, що жоден із них не є універсальним і повністю вдалим для всіх завдань. Кожен продукт має свої недоліки та обмеження:

- Unreal Engine [14] – надмірно складний для невеликих ігор, більшість можливостей залишаються невикористаними та має обмежені функціональні можливості для оптимізації гри.

- Godot [16] – зручний для створення, але має меншу кількість готових інструментів і прикладів, що ускладнює роботу.

- Unity [15] – хоч і універсальний, але вимагає досвіду роботи з C# та досить

потужного комп’ютера, що може бути проблемою для інді-проєктів.

З огляду на ці недоліки стає зрозуміло, що вибір рушія для розробки конкретної гри має враховувати не лише його технічні можливості, але й цілі проєкту, складність задуму та доступність інструментів.

2 ВИМОГИ ТА ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Вимоги до застосунку

Формування вимог до мобільного застосунку DinoRunner є важливим етапом проектування, оскільки саме вони визначають функціональні можливості системи, її якісні характеристики та загальну відповідність очікуванням користувачів.

Вимоги були сформовані на основі:

- аналізу предметної області;
- дослідження існуючих аналогів;
- визначення потреб цільової аудиторії;
- особливостей мобільних платформ.

З урахуванням цього всі вимоги до системи поділяються на дві основні групи: функціональні та нефункціональні.

2.1.1 Функціональні вимоги

Функціональні вимоги визначають основні можливості застосунку, тобто те, які дії система повинна виконувати та які сервіси надавати користувачеві.

У контексті гри DinoRunner функціональні вимоги охоплюють:

- ігрову логіку;
- взаємодію користувача з системою;
- обробку подій у грі;
- відображення інформації.

Особливу увагу приділено інтеграції пізнавального контенту, який є ключовою відмінністю застосунку від більшості конкурентів.

Зведений перелік функціональних вимог наведено в таблиці 2.1 (див. на ст. 23).

Таблиця 2.1

Перелік функціональних вимог

№	Функціональні вимоги	Опис
1.	Запуск гри	Користувач має можливість запуску гри з головного меню
2.	Керування персонажем	Гравець керує динозавром (стрибок, ухилення, рух)
3.	Генерація рівня	Рівень генеруються процедурно під час гри
4.	Система перешкод	На шляху гравця з’являються різні перешкоди
5.	Збір бонусів	Гравець може збирати предмети (наприклад, “м’ясо”)
6.	Підрахунок очок	Ведеться облік пройденої дистанції або очок
7.	Відображення UI	Інтерфейс відображає очки, стан гри та підказки
8.	Пауза гри	Можливість поставити гру на паузу
9.	Кінець гри	Відображення результату після завершення гри
10.	Освітній контент	Відображення фактів про динозаврів під час гри
11.	Меню	Наявність головного меню та навігації

Проведений аналіз функціональних вимог дозволяє зробити висновок, що система охоплює всі ключові аспекти класичної runner-гри, включаючи керування персонажем, генерацію рівнів, систему перешкод і підрахунок результатів.

Водночас важливою особливістю є наявність освітнього модуля, який реалізується через відображення фактів про динозаврів безпосередньо під час ігрового процесу. Такий підхід забезпечує органічне поєднання розваги та навчання без перевантаження користувача інформацією.

Крім того, передбачені функції паузи, перезапуску гри та навігації через меню, що підвищує зручність використання застосунку.

2.1.2 Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики системи, тобто те, як саме вона повинна виконувати свої функції. Вони є критично важливими для мобільних застосунків, оскільки безпосередньо впливають на користувацький досвід.

Для застосунку DinoRunner нефункціональні вимоги охоплюють такі аспекти:

- продуктивність;
- стабільність роботи;
- оптимізацію ресурсів;
- зручність користування;
- сумісність із пристроями.

Зведений перелік нефункціональних вимог наведено в таблиці 2.2.

Таблиця 2.2

Перелік нефункціональних вимог

№	Нефункціональна вимога	Опис
1.	Продуктивність	Гра повинна стабільно працювати на Android-пристроях (Здебільшого при 30-60 FPS)
2.	Оптимізація	Використання оптимізованих моделей (max 10000 вершин) і текстур (max 1024x1024)
3.	Час відгуку	Мінімальна затримка між діями користувача та реакцією гри (33-50 мс)
4.	Зручність використання	Інтуїтивно зрозумілий інтерфейс
5.	Надійність	Відсутність критичних помилок під час гри
6.	Масштабованість	Можливість додавання нових рівнів і механік
7.	Кросплатформеність	Можливість розширення на інші платформи
8.	Енергоспоживання	Оптимальне використання ресурсів мобільного пристрою (~ 50%)

Продовження таблиці 2.2

№	Нефункціональна вимога	Опис
9.	Графіка	Привабливий та оптимізований візуальний стиль
10.	Сумісність	Підтримка більшості сучасних Android-пристроїв (~Android 8+)

Аналіз нефункціональних вимог показує, що особлива увага приділяється оптимізації застосунку для мобільних пристроїв. Це включає використання оптимізованих 3D-моделей, текстур, а також технологій рівнів деталізації (LOD)[17], що дозволяє зменшити навантаження на пристрій.

Важливим аспектом є забезпечення стабільної роботи гри без затримок та збоїв, що досягається за рахунок оптимізації логіки гри та ефективного використання ресурсів системи.

Зручність користування реалізується через інтуїтивно зрозумілий інтерфейс і просте керування, що відповідає особливостям жанру runner та очікуванням користувачів.

Крім того, враховано можливість подальшого розширення функціоналу застосунку, що забезпечує його масштабованість і перспективність розвитку.

2.1.3 Програмні та апаратні вимоги

Визначення програмних та апаратних вимог є необхідною складовою процесу проєктування програмного забезпечення, оскільки вони встановлюють обмеження та умови функціонування системи, а також забезпечують її коректну реалізацію та подальший розвиток системи.

Для мобільного застосунку DinoRunner такі вимоги формуються з урахуванням особливостей ігрового рушія, мобільних платформ використання та

забезпечення стабільної роботи в умовах обмежених ресурсів (малий заряд батареї, включення режиму обмежених ресурсів телефону тощо).

Програмні вимоги

Програмні вимоги визначають сукупність програмних засобів, необхідних для розробки, налагодження, тестування та розгортання застосунку.

Розробка застосунку здійснюється з використанням сучасних інструментів, що забезпечують ефективну реалізацію ігрової логіки, графічного інтерфейсу та взаємодії з апаратними ресурсами пристрою.

Основними програмними компонентами є:

- ігровий рушій Unreal Engine 5 [14], який забезпечує повний цикл створення середовища ігор;

- мова програмування C++ та система візуального програмування Blueprints, що застосовуються для реалізації ігрової логіки;

- середовища розробки Visual Studio [18];

- програмний пакет Android SDK [19] для забезпечення сумісності із платформою Android;

- система контролю версій Git [20] для організації процесу розробки.

Зведені програмні вимоги наведено в таблиці 2.3.

Таблиця 2.3

Програмні вимоги

№	Компонент	Мінімальні вимоги
1.	Операційна система	Windows 10/11 (64-bit)
2.	Ігровий рушій	Unreal Engine 5.5
3.	IDE	Visual Studio 2022
4.	Android SDK	API Level 24+
5.	Система контролю версій	Git

Аналіз визначених програмних вимог свідчить про використання сучасного програмного стеку, який відповідає актуальним стандартам розробки мобільних ігор та забезпечує можливість подальшого розширення функціоналу застосунку.

Апаратні вимоги визначають мінімальні та рекомендовані характеристики технічних засобів, необхідних як для розробки застосунку, так і для його використання кінцевими користувачами.

З урахуванням специфіки розробки 3D-застосунків на основі Unreal Engine [14], до апаратного забезпечення розробника висуваються підвищені вимоги, зокрема щодо обчислювальної потужності процесора, обсягу оперативної пам’яті та продуктивності графічної підсистеми. Зведені вимоги до апаратного забезпечення персонального комп’ютера розробника наведено в таблиці 2.4.

Таблиця 2.4

Вимоги до апаратного забезпечення

№	Компонент	Мінімальні вимоги	Рекомендовані
1.	Процесор	Intel i5 / Ryzen 5	Intel i7 / Ryzen 7
2.	Оперативна пам’ять	8 GB	16 – 32 GB
3.	Відеокарта	GTX 1050 / аналог	RTX 2060+
4.	Накопичувач	SSD 256 GB	SSD 512 GB+

Використання рекомендованих характеристик дозволяє забезпечити ефективну роботу з ігровим рушієм, скоротити час компіляції та підвищити продуктивність процесу тестування.

Апаратні вимоги до кінцевого пристрою користувача формуються з урахуванням необхідності забезпечення стабільної роботи застосунку на широкому спектрі мобільних пристроїв середнього рівня продуктивності. Зведені вимоги наведено в таблиці 2.5.

Таблиця 2.5

Вимоги до мобільних пристроїв

№	Компонент	Мінімальні вимоги
1.	Операційна система	Android 7.0 (API 24)
2.	Процесор	4 ядра (1.8 GHz)
3.	Оперативна пам’ять	3 GB

Продовження таблиці 2.5

№	Компонент	Мінімальні вимоги
4.	GPU	Підтримка OpenGL ES 3.1
5.	Вільна пам’ять	200 MB
6.	Екран	HD (1280×720)

Визначені характеристики дозволяють досягти балансу між якістю графічного відображення та продуктивністю застосунку, що є критично важливим для мобільних ігор.

2.2 Опис IT-засобів для розробки: Unreal Engine, C++, Blueprints та супутніх технологій

Розробка сучасних ігор жанру runner базується на використанні спеціалізованих інструментів, які поєднують програмування, дизайн та інтеграцію графічних ресурсів.

Для створення гри DinoRunner основним середовищем розробки обрано Unreal Engine [14], який забезпечує широкий набір інструментів для створення ігрових проєктів різної складності.

Основними технологіями, що використовуються у процесі розробки, є:

- мова програмування C++ – використовується для реалізації складної логіки гри та оптимізації продуктивності;
- система візуального програмування Blueprints – дозволяє створювати ігрову логіку без написання коду, що значно прискорює процес розробки та є зручним для прототипування;
- інструменти роботи з анімацією – забезпечують створення плавних рухів персонажа, що є важливим для ігор жанру runner;
- система фізики рушія – використовується для реалізації взаємодії персонажа з перешкодами та ігровим середовищем.

Додатково можуть використовуватися сторонні інструменти для створення графіки, анімацій та звукового супроводу, що дозволяє підвищити якість кінцевого продукту.

Таким чином, використання Unreal Engine [14] забезпечує можливість реалізації повноцінної гри жанру runner з сучасними ігровими механіками, якісною графікою та високою продуктивністю.

2.3 Обґрунтування актуальності розробки «DinoRunner»

Розробка власної гри жанру runner «DinoRunner» є актуальною з кількох причин:

- освітня мета створення цієї гри дозволяє закріпити знання з програмування, розробки ігрової логіки, роботи з анімаціями, фізикою та створення різних компонентів до головного персонажа у середовищі Unreal Engine [14];

- популярність жанру. Ігри жанру runner є широко розповсюдженими на мобільних платформах завдяки простоті та динамічному ігровому процесу. Користувачі надають перевагу швидким ігровим сесіям, що що зумовлює актуальність розробки у цьому жанрі;

- заповнення ніші інді-проєктів. Сучасний ринок мобільних ігор активно розвивається, особливо завдяки інді-розробникам та їх проектам. Проєкт DinoRunner пропонує поєднання класичного раннера з пізнавальними елементами (інформацією та фактами про динозаврів), що робить його унікальним;

- дослідницький характер – у процесі розробки можна дослідити можливості Unreal Engine [14] для створення 3D ігор для смартфонів, зокрема використання системи Blueprints, а також оцінити ефективність рушія в умовах обмежених апаратних ресурсів мобільних пристроїв.

DinoRunner заповнює нішу пізнавального runner-а, поєднуючи доступну механіку із науково достовірною інформацією та фактами про динозаврів, що робить його унікальним на ринку мобільних ігор для широкої аудиторії.

2.4 Проектування архітектури гри DinoRunner

Архітектура гри DinoRunner побудована на основі можливостей ігрового рушія Unreal Engine [14] і відповідає принципам модульності та розділення відповідальності між різними класами та компонентами. Проєкт організовано таким чином, щоб кожен функціональний модуль був незалежним і міг розроблятися та тестуватися окремо, що суттєво спрощує підтримку та розширення застосунку.

Загальна концепція DinoRunner базується на класичній архітектурі runner-ігор, адаптованій під особливості рушія Unreal Engine [14].

Гра складається з чотирьох основних модулів, кожен з яких виконує чітко визначену функцію в загальній системі. Основні модулі програми:

- модуль головного персонажа відповідає за керування динозавром-гравцем: обробку вводу, анімації, фізику руху та визначення колізій з перешкодами. Цей модуль є визначальним з точки зору ігрового досвіду, оскільки персонаж безпосередньо реагує на дії користувача та визначає результат кожної ігрової сесії;
- модуль генерації рівня реалізує створення ігрового середовища: спавн перешкод, управління швидкістю руху та наростання складності в міру прогресу гравця. Алгоритм генерації враховує поточний час гри та рахунок, для забезпечення поступового наростання складності без різких змін;
- модуль інтерфейсу користувача містить усі екрани застосунку: головне меню, ігровий HUD із поточним рахунком, блоком інформації про динозавра та виведенням фактів про нього а також екран завершення гри та відображенням інформації про ігрову сесію. Реалізовано засобами UserWidget і UMG (Unreal Motion Graphics), яка є стандартним інструментом для побудови UI в Unreal Engine [14];
- модуль пізнавального контенту керує зберіганням і відображенням інформації та фактів про динозаврів. Інформація та факти зберігаються у структурах DataTable і відображаються гравцю у потрібний момент. Модуль інтегрований з ігровим HUD та системою підрахунку очок.

Графічно модулі виглядають як зображено на рисунку 2.1.

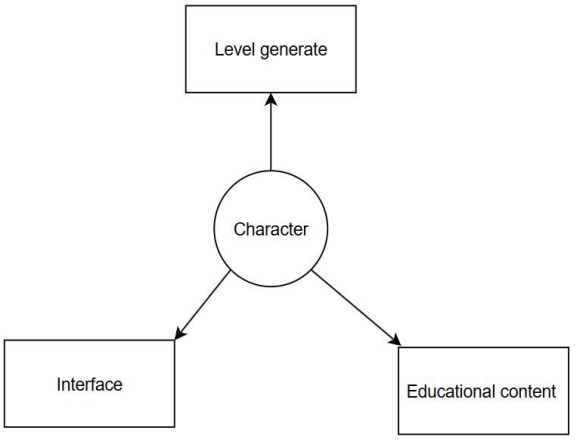


Рис. 2.1 Основні модулі

У процесі розробки DinoRunner застосовується комбінований підхід – поєднання системи візуального програмування Blueprints та мови програмування C++. Такий підхід є стандартною практикою при розробці проєктів в Unreal Engine [14] і дозволяє ефективно використовувати переваги обох інструментів.

Blueprints використовуються для реалізації ігрової логіки, керування анімаціями персонажа, налаштування поведінки об'єктів на рівні та побудови UI-віджетів. Візуальний підхід забезпечує швидке прототипування і зручну налагодження логіки без необхідності перекомпіляції проєкту. C++ застосовується для реалізації базових класів персонажа та ігрового режиму, логіки обробки колізій і системи підрахунку очок, де критична продуктивність.

Центральним елементом архітектури є клас ігрового режиму GM_DinoRunner, який ініціалізує гру, відстежує стан ігрової сесії та координує взаємодію між модулями. Основні класи проєкту наведено на рисунку 2.2 (див. ст. 31).

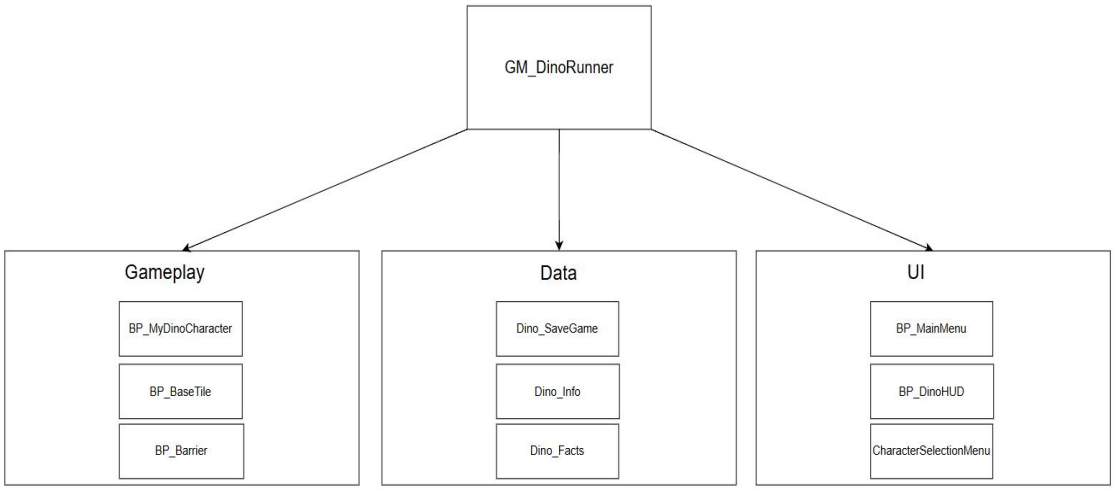


Рис. 2.2 Основні класи проекту

Взаємодія між модулями відбувається через колізію On Component Begin Overlap. Коли персонаж стикається з перешкодою, генерується подія "Death", яку отримує персонаж, зупиняє всі підрахування та передає UI-модулю для відображення екрана Game Over. Така подієво-орієнтована архітектура забезпечує слабку зв'язаність між модулями та спрощує розширення функціональності застосунку.

Для зберігання даних у грі використовуються такі підходи: рекорд гравця зберігається за допомогою системи SaveGame в Unreal Engine [14], що забезпечує його збереження між сесіями; факти про динозаврів зберігаються у DataTable у форматі рядків; поточний рахунок та стан гри зберігаються в GameMode як змінні сесії та обнуляються при кожному новому запуску.

Організація файлової структури проєкту в Unreal Engine [14] відповідає загальноприйнятим стандартам: ресурси згруповано за типами (Blueprints, Meshes, Materials, UI, DataTables), що забезпечує зручну навігацію та масштабованість проєкту.

2.5 Моделювання сценаріїв використання (Use Case)

Для формалізації вимог до системи та визначення взаємодії між гравцем і грою використано метод моделювання сценаріїв використання (Use Case). Цей підхід дозволяє наочно відобразити функціональні можливості застосунку з точки зору кінцевого користувача та забезпечує однозначне розуміння вимог на етапі проектування.

Основним актором у системі є Гравець – особа, яка безпосередньо взаємодіє з мобільним застосунком DinoRunner. Система не передбачає інших акторів, оскільки є однокористувацькою мобільною грою без мережевої складової. Основний план використання застосунку зазначено на рисунку 2.3.

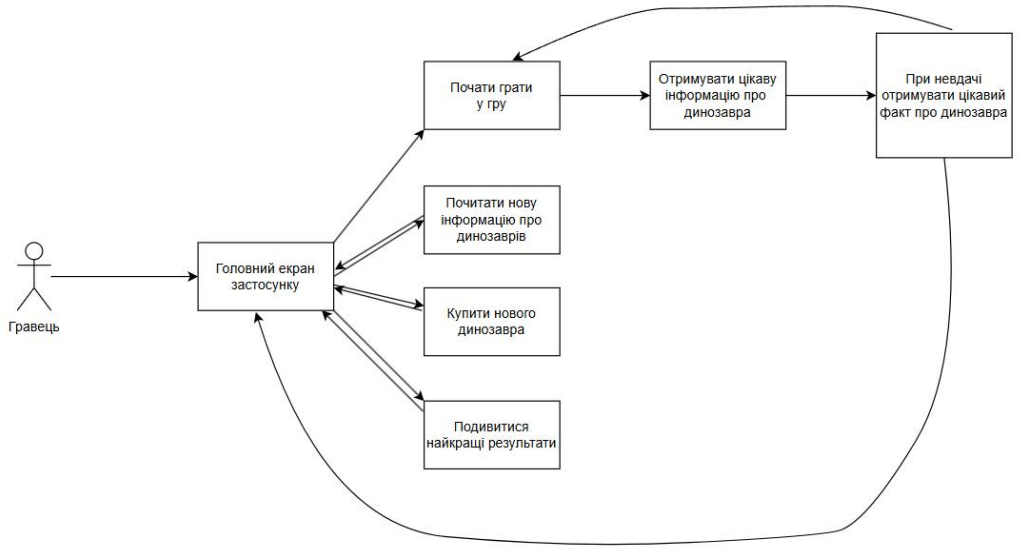


Рис 2.3 Основний план використання застосунку

Визначено такі основні варіанти використання системи гри.

UC-01. Запуск гри. Гравець натискає кнопку «Грати» і переходить в режим керування динозавром. Система ініціалізує ігровий рівень, розміщує персонажа у стартовій позиції та запускає генератор перешкод. Передумова: застосунок встановлено на Android-пристрої. Постумова: гравець перебуває в активній ігровій сесії.

UC-02. Керування персонажем. Під час активної ігрової сесії гравець натискає на екран для виконання стрибка або проводить свайп в бік для ухилення.

Система обробляє сенсорний ввід і передає команду класу BP_DinoCharacter, який змінює фізичний стан персонажа.

UC-03. Зіткнення з перешкодою. Персонаж стикається з перешкодою. Система виявляє колізію, зупиняє ігровий процес, відображає анімацію програшу та переходить до екрана Game Over з поточним рахунком і рекордом. Гравець може обрати перезапуск або повернення в меню.

UC-04. Перегляд рекорду та факту. Гравець переглядає свій поточний рекорд у головному меню або на екрані Game Over. Після проходження певної відстані система автоматично відображає картку з фактом про динозавра на екрані. Система зчитує збережений рекорд із SaveGame та відображає його.

UC-05. Перезапуск гри. На екрані Game Over гравець натискає «Грати знову». Система скидає всі параметри поточної сесії та повертається до стану UC-01 без повернення в головне меню.

UC-06. Вихід у головне меню. Гравець натискає відповідну кнопку на екрані Game Over або під час паузи. Система зупиняє поточну сесію, звільняє ресурси рівня та відображає головне меню.

Зведені характеристики всіх варіантів використання наведено в таблиці 2.6.

Таблиця 2.6

Варіанти використання системи DinoRunner

Ідентифікатор	Назва варіанту використання	Пріоритет
UC-01	Запуск гри	Високий
UC-02	Керування персонажем	Високий
UC-03	Зіткнення з перешкодою та завершення гри	Високий
UC-04	Перегляд рекорду та факту	Середній
UC-05	Перезапуск гри	Високий
UC-06	Вихід у головне меню	Середній

Аналіз варіантів використання дозволив визначити пріоритети розробки: насамперед реалізовано функціональність UC-01, UC-02, UC-06 та UC-03 як обов'язкову базу ігрового процесу. Варіанти UC-04 та UC-05 реалізовано на другому етапі як розширення, що надають грі унікальності та додаткової цінності для користувача.

2.6 Проектування інтерфейсу користувача

Інтерфейс користувача гри DinoRunner розроблено з урахуванням особливостей мобільної платформи Android та цільової аудиторії – казуальних гравців усіх вікових груп. Основними принципами проектування UI є мінімалізм, інтуїтивність та відповідність тематиці доісторичного світу.

Застосунок складається з основних віджетів: головного меню, ігрового екрана під час гри, віджета купівлі динозавра, інформації про динозавра та екрана завершення гри. Кожен екран реалізовано як окремий віджет в UE.

Головне меню є першим екраном, який бачить гравець після запуску застосунку.

Екран містить назву гри DinoRunner, розміщену праворуч у центральній третині екрану, виконану у стилізованому шрифті з тематичним оформленням. Нижче розміщено основні кнопки: «Грати», «Купівля», «Інформація» та «Рекорд». Кнопка «Грати» є домінуючим елементом і має більший розмір для зручності натискання на сенсорному екрані, приклад наведено на рисунку 2.4(див. ст. 36).

Рис. 2.4 Приклад головного меню

Ігровий екран є основним і найважливішим інтерфейсним рішенням застосунку. HUD (Heads-Up Display) спроектовано таким чином, щоб мінімально перекривати ігрове поле (Рис. 2.5). У верхньому куті розміщено поточний рахунок гравця, що оновлюється в реальному часі.

Рис. 2.5 Ігрове поле під час гри

Екран завершення гри відображається після зіткнення персонажа з перешкодою. Він містить напис «Гра завершена», поточний результат гравця та

рекордний результат. Картка факту про динозавра з'являється у центрі нижньої частини екрана і містить короткий пізнавальний текст. Екран прикладу зображено на рисунку 2.6.

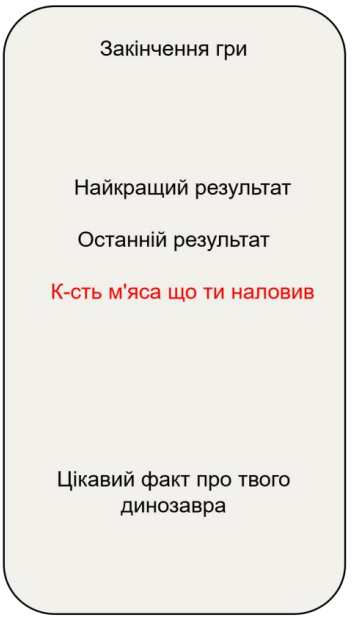


Рис. 2.6 Екран завершення гри

Екран купівлі динозавра – один із допоміжних екранів застосунку, який гравець відкриває перед початком гри(Рис. 2.7). У верхньому лівому куті відображається назва поточного обраного динозавра («Default Dino»), а у верхній центральній частині – кількість доступного ресурсу «Meats: 15», що є внутрішньою валютою для розблокування нових персонажів. У центрі екрана розміщено 3D-модель динозавра на фоні доісторичного пейзажу, що забезпечує занурення в тематику гри ще до початку ігрової сесії. Навігація між доступними персонажами реалізована через кнопки зі стрілками ліворуч і праворуч. У нижній частині екрана розміщено дві кнопки: «Back» для повернення в головне меню та «Selected» із зеленим підсвічуванням для підтвердження вибору. Зелений колір кнопки підтвердження є інтуїтивним сигналом для гравця і відповідає загальноприйнятим UX-стандартам мобільних застосунків(див. ст. 38).

Рис. 2.7 Екран купівлі динозавра

Екран пізнавального контенту – картка з інформацією про динозавра який обраний на даний момент. Картка займає більшу частину екрана і виконана в темно-оливковому кольорі, що гармоніює із загальною палітрою гри. У верхньому лівому куті розміщено кнопку закриття «Х» у червоному квадраті, яка дозволяє гравцеві швидко закрити картку і повернутися до гри. Центральна частина картки містить зображення динозавра, нижче – його назву жирним шрифтом і розгорнутий інформаційний текст білим кольором на темному фоні. Такий контраст забезпечує зручну читабельність на мобільних екранах у різних умовах освітлення. Зміст тексту є науково достовірним і написаний доступною мовою, що підтверджує освітню цінність застосунку.

Рис. 2.8 Екран інформації динозавра

Також є екран найкращих результатів. В ньому зображено назва динозавра який був активний в період найкращого результату, скільки очок вдалось зібрати та час щоб користувач міг усвідомлювати коли мозок та тіло працює найкраще.

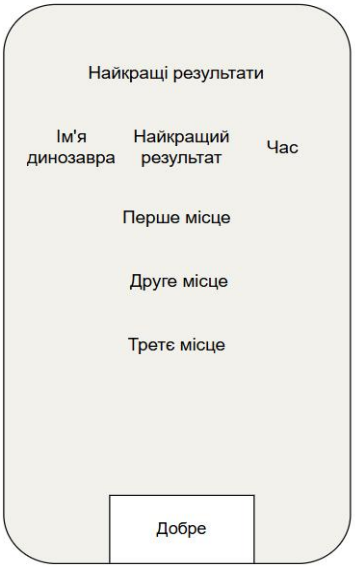


Рис. 2.9 Екран інформації динозавра

Кольорова палітра інтерфейсу побудована на теплих земляних тонах – коричневих, зелених і помаранчевих відтінках, що відповідає доісторичній тематиці. Текстові елементи виконано білим кольором із темною тінню для забезпечення контрасту на різних фонах. Усі інтерактивні елементи мають мінімальний розмір щодо доступності мобільних додатків.

2.7 Проектування ігрових механік

Ігрові механіки DinoRunner спроектовано на основі аналізу жанру runner та класичних рішень, що зарекомендували себе в успішних представниках жанру. Кожна механіка розглядається як окремий функціональний блок із чітко визначеними вхідними даними, логікою обробки та результатом.

Механіка руху персонажа є базовою і простою. Персонаж безперервно рухається вперед із константною початковою швидкістю. Рух реалізовано переміщенням персонажа та створенням ігрового світу – такий підхід є

стандартним у жанрі та спрощує реалізацію генерації рівня. Швидкість руху поступово збільшується.

Механіка стрибка реалізована через систему фізики Unreal Engine [14]. При натисканні на екран до персонажа застосовується миттєвий імпульс угору. Висота стрибка є фіксованою і визначена параметром LaunchVelocity. Після торкання землі лічильник стрибків скидається. Технічно стан персонажа визначається прапором IsGrounded, який встановлюється через OnHit-подію від колізії з підлогою.

Механіка процедурної генерації перешкод є ключовою для забезпечення варіативності ігрового процесу. Генератор BP_BaseTile розміщує перешкоди на відстані від персонажа за межами екрана. Відстань між перешкодами визначається випадковим чином у межах заданого діапазону. Тип перешкоди обирається з пулу доступних варіантів з різними ваговими коефіцієнтами: прості перешкоди мають вищу вагу на початку гри, складні комбінації – на пізніх стадіях. Перешкоди, що вийшли за межі екрана позаду персонажа, автоматично знищуються.

Механіка підрахунку очок базується на відстані, пройденій гравцем. Рахунок збільшується пропорційно часу, прожитому в активній ігровій сесії, з урахуванням поточної швидкості. Рекорд зберігається між сесіями засобами SaveGame та порівнюється з поточним результатом наприкінці кожної гри.

Механіка відображення фактів про динозаврів спрацьовує при досягненні порогових значень рахунку. Факт обирається з DataTable випадковим чином.

2.8 Особливості гри DinoRunner

DinoRunner – мобільний застосунок жанру runner, розроблений у межах цієї кваліфікаційної роботи. Гравець керує динозавром, який здійснює переміщення персонажа в ігровому середовищі крізь доісторичний світ і намагається уникнути різноманітних перешкод на своєму шляху. Атмосфера доісторичного світу задає особливий настрій і забезпечує тематичну відмінність від аналогів.

Ключовою відмінністю DinoRunner від більшості представників жанру є інтеграція верифікованих фактів про динозаврів безпосередньо в ігровий процес, що забезпечує органічне поєднання розважальної та освітньої складових.

Зазначена особливість відрізняє DinoRunner від типових представників жанру, що орієнтовані виключно на розважальний контент. DinoRunner може слугувати не лише джерелом розваги, але й інструментом підвищення зацікавленості користувача до вивчення природничої історії. Застосунок виконує функцію популяризації наукових знань, особливо серед молодої аудиторії, для якої подача фактів та короткої інформації є значно привабливішим за звичайні тексти у підручниках.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Реалізація системи керування персонажем

Персонаж гравця є центральним елементом ігрового процесу DinoRunner. Його реалізація вимагала вирішення кількох технічних задач: обробка сенсорного вводу на Android-пристрої, реалізація фізично коректного стрибка та ухилення, налаштування анімаційного стейт-машину та визначення колізій із перешкодами.

Базовий клас персонажа BP_MyDinoCharacter успадковується від стандартного класу ACharacter в Unreal Engine [14], що надає вбудовану підтримку фізики пересування, компонент капсульної колізії та компонент скелетної сітки для відображення 3D-моделі. Також визначено позицію камери в грі та вона рухається із такою ж швидкістю як обрана модель динозавра. Більшість ігрової логіки реалізовано в Blueprint-класі BP_MyDinoCharacter, що успадковується від C++-класу. Загальний приклад персонажу можемо побачити на рисунку 3.1.

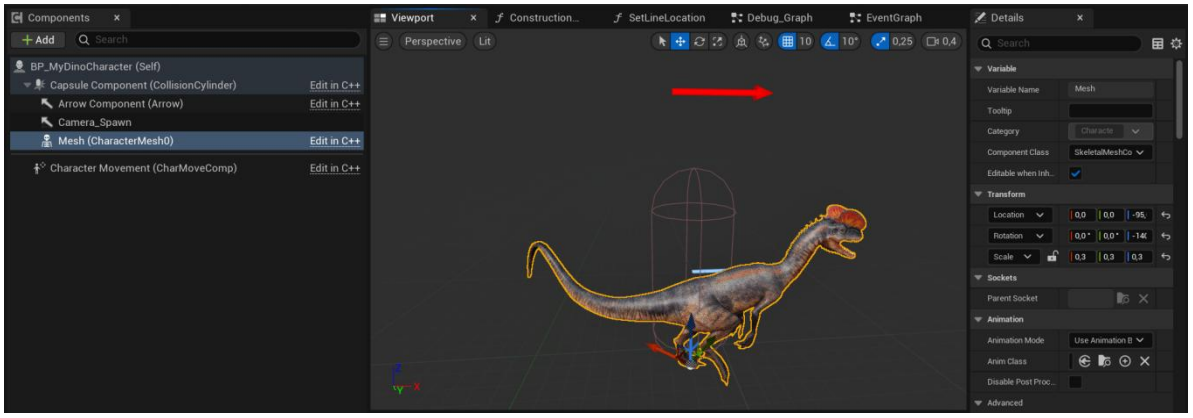


Рис. 3.1 Приклад персонажу в UE

Обробка сенсорного вводу реалізована через систему Input Touch, що відповідає рекомендованому підходу для обробки сенсорного вводу в Unreal Engine 5 [14]. Це дозволяє визначити початкову точку де користувач торкнувся

екрану і де він відпустив і за допомогою цього розуміємо чи це проведено пальцем ліворуч чи праворуч.

Реалізація механіки стрибка відбувається у кілька етапів. Якщо проведено пальцем знизу у вверх, то йде перевірка чи не знаходиться гравець вже у стрибку, якщо ні то викликається функція стрибку з UE.

Система визначення колізій базується на компоненті капсульної колізії персонажа та компонентах Box Collision перешкод. При виявленні перетину (OnComponentHit) між персонажем та тригером перешкоди перевіряємо чи то обрана модель персонажа і якщо це він то генерується подія Death. При цьому зупиняється підрахунок очок, гра переходить до стану паузи та відображається віджет GameOver, який повідомляє нам щодо закінчення сесії гри. В GameOver вносимо яку відстань подолано та яка кількість м’яса назбирана. Для уникнення хибних спрацювань введено короткочасну затримку після старту гри – перші 2 секунди колізії не обробляються.

Реалізацію персонажа було протестовано на Android-пристрої. Час відгуку на сенсорний ввід не перевищує 50 мс, що відповідає вимогам для забезпечення комфортного ігрового процесу. Стрибок і ухилення працюють коректно у всіх тестових сценаріях, включно з подвійним стрибком та ухиленням одразу після приземлення.

3.2 Реалізація генерації перешкод та рівнів

Процедурна генерація перешкод є ключовим технічним рішенням, що забезпечує нескінченність і варіативність ігрового процесу DinoRunner. Реалізація генератора SpawnObstacles створена у GM_DinoRunner тому що здебільшого там створюються такі частини та вирішує задачу динамічного створення ігрового середовища з урахуванням поточного рівня складності.

Архітектура генератора побудована на заздалегідь створеного набору об'єктів-перешкод, які завжди рандомно створюються та знищуються коли гравець їх подолав. Це робить гру непередбаченою та цікавою щодо проходження.

Алгоритм генерації перешкод працює наступним чином. Генератор відстежує позицію останньої розміщеної перешкоди. Коли ця перешкода наближається до персонажа на визначену відстань огляду, генератор активується та обирає наступну перешкоду. Тип перешкоди обирається за зваженим випадковим вибором з урахуванням поточного рівня складності.

Наростання складності реалізовано через систему рівнів DifficultLevel, що оновлюється щоразу при досягненні порогових значень рахунку. Рівень складності впливає на параметри CurrentSpeed та вагові коефіцієнти вибору типу перешкоди. Це забезпечує плавне зростання труднощів без різких стрибків, що негативно позначалися б на ігровому досвіді.

Ландшафт реалізовано за принципом нескінченного створення нового і видаленням старого ландшафту(рис 3.2).

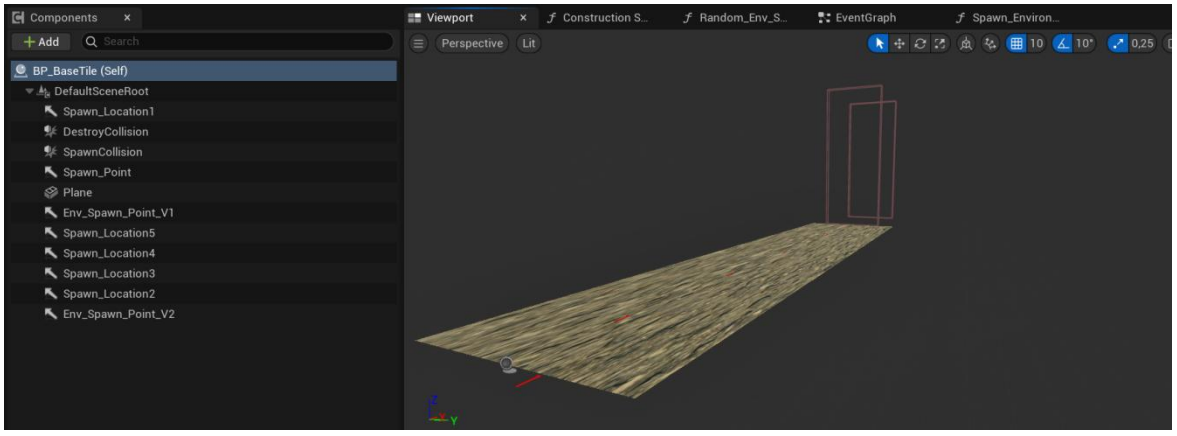


Рис. 3.2 Actor що створює ландшафт та перешкоди

Фонові елементи – небо, хмари, рослинність – реалізовані як окремі шари, що рухаються з різною швидкістю, створюючи ефект паралаксу. Швидкість руху кожного шару пропорційна його умовній «відстані від камери»: дальні елементи рухаються повільніше, ближні – швидше. Це підсилює відчуття глибини та занурення в ігровий світ. Ігровий світ зображено на рис 3.3 (див. ст. 45).

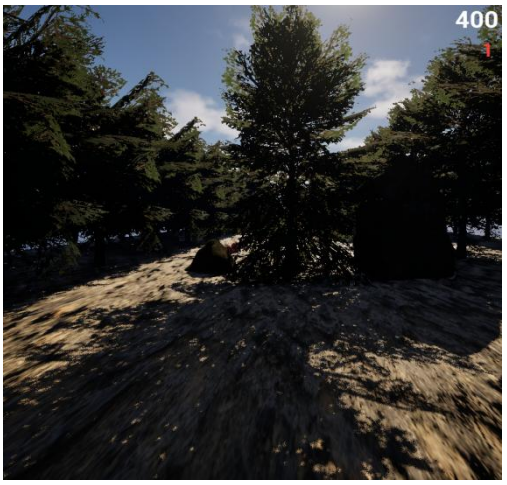


Рис. 3.3 Приклад ігрового світу

Під час тестування генератора перевірено коректність роботи при безперервній грі тривалістю 30 хвилин. Витоків пам'яті не виявлено, частота кадрів залишалася стабільною протягом усього тесту. Перевірено відсутність перекриваючих або непрохідних комбінацій перешкод – логіка мінімальних відступів між перешкодами виключає ситуації, коли гравець фізично не може пройти певну ділянку рівня.

3.3 Реалізація інтерфейсу користувача

Інтерфейс користувача DinoRunner реалізовано засобами системи UMG (Unreal Motion Graphics) – стандартного інструменту Unreal Engine [14] для побудови UI. Кожен екран застосунку являє собою окремий Widget Blueprint, що успадковується від класу UserWidget.

Головне меню (BP_MainMenu) реалізовано як повноекранний Widget, що відображається при старті застосунку. Приклад головного мені зображено на рисунку 3.4 (див. на ст. 46).

Рис. 3.4 Головне меню у грі

Ієрархія компонентів побудована на базі вертикального Box-контейнера з трьома секціями: заголовок, зона кнопок і зона рекорду. Кнопка «Грати» виконує роль основного тригера запуску ігрового процесу; при її активації викликається функція StartGame, у результаті чого ініціалізується ігровий HUD, а також відбувається деактивація або звільнення другорядних елементів інтерфейсу, які не є необхідними під час активної гри.

Ігровий HUD (BP_DinoHUD) є дуже важливішим інтерфейсним рішенням застосунку. HUD (Heads-Up Display) спроектовано таким чином, щоб мінімально перекривати ігрове поле. У верхньому куті розміщено поточний рахунок гравця, що оновлюється в реальному часі. Отже під час ігрового процесу користувач має можливість одночасно відстежувати ключові показники, зокрема подолану дистанцію, кількість зібраного м’яса, а також візуально контролювати головного персонажа та уникати перешкод, приклад зображено на рис. 3.5 (див. ст. 47).

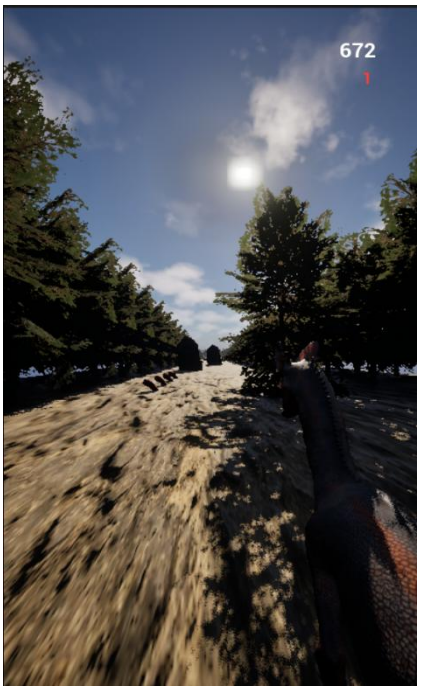


Рис. 3.5 Ігровий інтерфейс

Екран завершення гри (BP_GameOver) відображається після отримання події GameOver від основного класу GameMode. Екран реалізовано як окремий повноекранний Widget-компонент, який інтегрується в інтерфейс користувача та забезпечує плавний перехід між активним ігровим процесом і фінальним інформаційним вікном з анімованою появою (розмиття фону із затримкою 0.5 с після завершення гри). Особливістю реалізації є те, що після завершення гри гравець зберігає можливість спостерігати за моделлю свого динозавра та навколишнім ігровим світом. Також гравець отримує цікавий факт про його динозавра, що додає бажання проходити ігрові сесії знову. Виходить віджет завершення гри подає багато інформації: яка максимальна подолана дистанція, кількість метрів, пройдених під час останньої спроби, а також кількість знайденого м’яса за одну ігрову сесію.Загальний вигляд та приклад функціонування екрана завершення гри наведено на рисунку 3.6 (див. ст. 48).

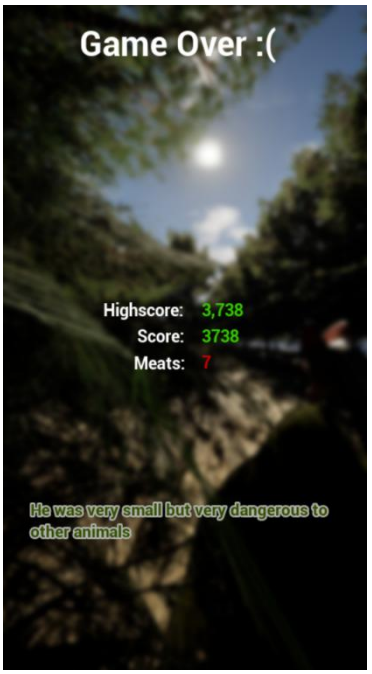


Рис. 3.6 Віджет завершення гри

Реалізація купівлі або зміни головного героя створена у віджеті WBP_CharacterSelectionMenu. У ньому користувач спочатку дізнається який в нас зараз обраний динозавр, які він має характеристики, ціну і тд. Натискаючи кнопку праворуч з’являється модель наступного динозавра з вищим індексом а якщо натискаємо кнопку ліворуч то з меншим. При зміні індексу змінюється 3д модель обраного динозавра і якщо користувач купив нового та натиснув “Select” то тоді зберігається новий індекс, і з новим індексом, і нова інформація, і цікаві факти, нове положення камери щодо нового динозавра, щоб з кожним гралося добре. І якщо користувач вже купив динозавра, а хоче обрати минулого то просто натискає кнопку “Select” та грає за минулого динозавра. Якщо ж не вистачає м’яса щоб купити нового героя то користувач не може його купити. Коли користувач перейшов вже на обрану модель то тоді трішки потемніє кнопка і буде написано “Selected”, тобто вже обрано (Рис. 3.7) і залишається повернутися на головний екран та займатися цікавою справою (див. ст. 49).

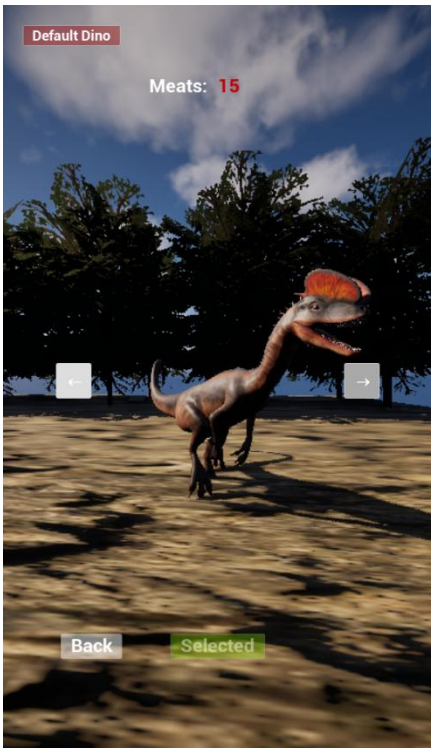


Рис. 3.7 Магазин динозаврів

Екран найкращих результатів користувача створено у віджеті WBP_BestResults. Даний інтерфейсний модуль призначений для виведення детальної інформації про максимальні досягнення гравця, що включає кількісні показники результативності, часові мітки фіксації рекордів, а також ідентифікатор ігрового персонажа (динозавра), за допомогою якого було досягнуто результат.

Стилістичне оформлення екрана узгоджено із загальною візуальною концепцією застосунку, проте акцент зміщено на інформаційні блоки з числовими даними для підвищення читабельності та ергономіки сприйняття. Семантика кольорів підпорядкована головному меню, що забезпечує цілісність користувацького досвіду. Архітектура інтерфейсу передбачає інтуїтивно зрозумілу навігацію, що дозволяє оперативно повернутися до головного екрана для ініціації нової ігрової сесії після аналізу поточної статистики. Приклад створеного віджету зображено на рис. 3.8 (див. ст. 50).

Рис. 3.8 Віджет найкращих результатів

Для забезпечення коректного масштабування інтерфейсу на різних розмірах екранів Android застосовано DPI Scaling. У налаштуваннях проєкту встановлено криву масштабування, що забезпечує відповідне збільшення UI-елементів на екранах з різною щільністю пікселів – від 4-дюймових бюджетних смартфонів до 7-дюймових планшетів.

Тестування інтерфейсу проводилось на пристроях з різними розмірами екрана та роздільною здатністю. Усі елементи коректно відображаються та зберігають функціональність. Час відгуку UI на натискання не перевищує 16 мс, що відповідає частоті кадрів 60 FPS.

3.4 Інтеграція пізнавального контенту про динозаврів

Інтеграція пізнавального контенту є ключовою особливістю DinoRunner, що відрізняє гру від типових представників жанру. Реалізація цього модуля вирішує задачу органічного введення освітньої інформації у розважальний контекст.

Для зберігання даних про динозаврів використовується структура даних

FDinoInfo, що містить такі поля: DinoName – рядок з назвою динозавра; DinoInfo – рядок із пізнавальною інформацією; DinoImage – посилання на текстуру із зображенням динозавра. Усі факти зберігаються у DataTable DT_DinoInfo, що є табличним активом Unreal Engine [14] з рядками типу FDinoInfo (приклад Рис. 3.9.)

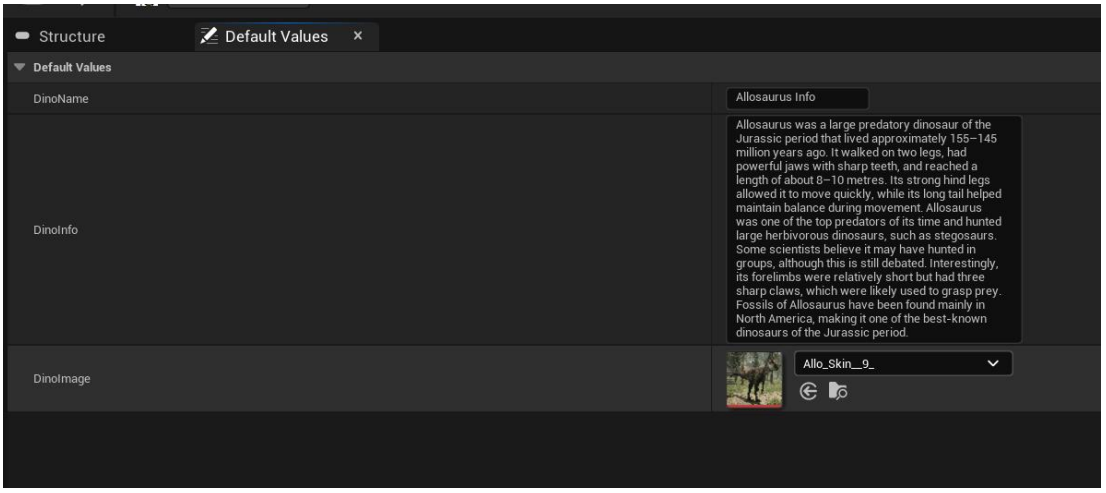


Рис. 3.9 Зберігання даних про динозаврів

Виклик даних створено у WPB_DinoInfo і в ньому створено гарний фон і розповідь про динозавра з його рисунком, ім’ям та інформацією про нього приклад зображено на рисунку 3.10.

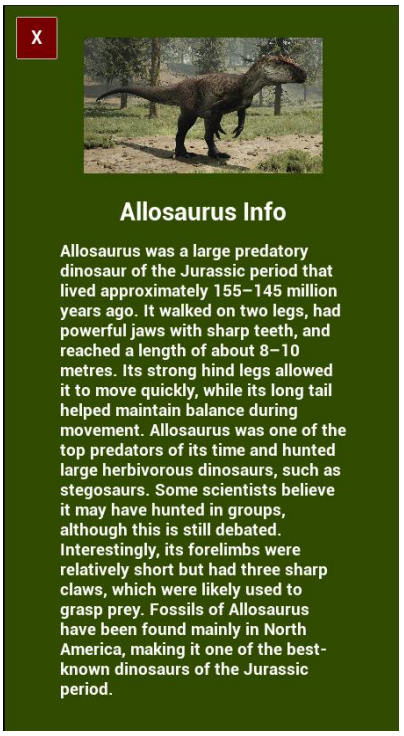


Рис. 3.10 Інформація про динозавра

Для зберігання даних про цікаві динозаврів використовується структура даних FDinoFact, що містить такі поля: DinoName – рядок з назвою динозавра; FactText – рядок із пізнавальним фактом. Усі факти зберігаються у DataTable DT_DinoFacts, що є табличним активом Unreal Engine [14] з рядками типу FDinoFact.

До бази фактів включено відомості про різних динозаврів, що охоплюють різні геологічні епохи. Кожен факт підібрано таким чином, щоб він був цікавим, достовірним і зрозумілим для широкої аудиторії. Приклади включених фактів наведено в таблиці 3.1.

Таблиця 3.1

Приклади фактів із бази даних DinoFacts

Динозавр	Ера	Факт
Велоцираптор	Крейдяний	Був дуже маленьким але дуже небезпечним для інших тварин
Тиранозавр	Крейдяний	Мав найсильніший укус серед наземних тварин – до 57 000 Н
Алозавр	Юрський	Наносив удари верхньою щелепою знизу вгору, щоб поранити здобич, а не покладався лише на силу укусу.

Функція BP_FactManager відповідає за управління відображенням фактів після ігрової сесії. При старті гри менеджер отримує повний список фактів з DT_DinoFacts про обрану модель динозавра та висвітлює їх при закінченні ігрової сесії. Логіка спрацювання показу факту реалізована у межах екранної форми завершення гри (GameOver_Screen) та працює з кожним динозавром.

Тестування модуля пізнавального контенту підтвердило коректну роботу всіх сценаріїв: факти відображаються у правильний момент, не повторюються протягом однієї сесії, картка коректно закривається як автоматично, так і при натисканні гравця. Текст фактів читається на екранах різних розмірів.

4 ТЕСТУВАННЯ ТА ОПТИМІЗАЦІЯ ЗАСТОСУНКУ

4.1 Стратегія та методи тестування мобільної гри

Тестування є обов'язковим етапом розробки програмного забезпечення, що дозволяє виявити та усунути дефекти до передачі продукту кінцевому користувачеві. Для гри DinoRunner розроблено комплексну стратегію тестування, що охоплює функціональну коректність ігрових механік, стабільність роботи на цільовій платформі Android та якість користувацького досвіду.

Тестування проводилось у три етапи. На першому етапі – в процесі розробки виконувалось компонентне тестування кожного модуля безпосередньо в редакторі Unreal Engine [14] з використанням режиму Play In Editor (PIE). Це дозволяло швидко виявляти й усувати дефекти ще до збірки під Android. На другому етапі тестування проводилось також у UE але у Standalone Game Mobile. У ньому було вперше протестовано на потужності як у смартфоні. На третьому етапі – після збірки проводилось тестування на реальному пристрої для перевірки коректності сенсорного вводу, продуктивності та відображення на конкретному екрані.

У рамках проєкту застосовано такі методи тестування. Функціональне тестування перевіряло відповідність реалізованих функцій проектним вимогам: чи коректно виконується стрибок, чи спрацьовує колізія з перешкодою, чи відображаються факти у потрібний момент. Тестування продуктивності (Performance Testing) вимірювало частоту кадрів (FPS) та споживання пам'яті під час активної ігрової сесії. Регресійне тестування виконувалось після кожного значного доопрацювання, щоб переконатися, що нові зміни не порушили раніше реалізований функціонал. Тестування сумісності перевіряло коректну роботу застосунку на пристроях з різними характеристиками екрану.

Тестове середовище включало редактор Unreal Engine 5 [14] для компонентного тестування, а також реальний Android-пристрій для фінального

тестування. Усі виявлені дефекти фіксувались у вигляді записів із описом кроків відтворення, очікуваного та фактичного результату.

4.2 Розробка та виконання тест-кейсів

Для верифікації функціональності застосунку DinoRunner проведено комплексне тестування, що охоплює основні сценарії взаємодії користувача з системою. Перевірка здійснювалася шляхом виконання набору тест-кейсів (ТК), кожен з яких оцінювався за критерієм відповідності фактичного результату очікуваному. Перелік виконаних тест-кейсів:

1. Перевірка коректності завантаження головного екрана та ініціалізації системних ресурсів.
2. Верифікація циклу анімації бігу та стабільності руху динозавра по ігровій зоні.
3. Перевірка реакції на вхідний сигнал та коректності траєкторії руху персонажа.
4. Тестування системи обробки зіткнень із перешкодами та ініціації завершення сесії.
5. Перевірка логіки інкрементації рахунку пропорційно пройденій дистанції.
6. Верифікація появи інформаційних вікон з фактами при досягненні визначених тригерів.
7. Перевірка працездатності кнопок переходу між екранами та налаштуваннями.
8. Перевірка відображення фінальної статистики та елементів керування після завершення гри.
9. Верифікація обнулення ігрових параметрів та повторної генерації рівня.
10. Тестування можливості активації/деактивації аудіоефектів у системному меню.
11. Перевірка запису максимального результату в пам'ять пристрою.
12. Верифікація коректного масштабування HUD-елементів під різні роздільні здатності екрана.
13. Перевірка алгоритму випадкового вибору та чергування пізнавальної інформації.
14. Тестування плавності переходів між станами Idle, Run та Jump.

15. Коректна деініціалізація процесів при виході із застосунку.

За результатами випробувань усі 15 тест-кейсів отримали статус Passed, що підтверджує повну відповідність розробленого програмного забезпечення встановленим вимогам. Виявлені на етапі розробки дефекти були успішно усунені, що детальніше описано в наступному підрозділі.

4.3 Виявлені дефекти та їх усунення

У процесі тестування було виявлено кілька дефектів різного ступеня критичності. Кожен дефект було задокументовано, проаналізовано та усунено до фінальної збірки застосунку.

Дефект BUG-01: подвійний стрибок спрацьовував коли модель була у повітрі. При швидких послідовних торканнях екрана персонаж міг виконати три і більше стрибки підряд, що порушувало баланс гри та дозволяло уникати більшості перешкод без зусиль. Причиною стала відсутність перевірки чи торкалась землі чи ні. Виправлення: додано умову IsJump перед застосуванням імпульсу, а скидання лічильника перенесено в подію OnLanded для гарантованого спрацювання.

Дефект BUG-02: деякі моделі були повернуті на 45°. Баг виник через помилкову систему координат у різних магазинах. Виправлення: спочатку експортувати потрібну 3д модель і тоді імпортувати її у проект під правильним кутом.

Дефект BUG-03: створення перешкод яких неможливо було уникнути на дуже високому ступеню складності. Виправлення стало додавання перешкод які можна уникнути за допомогою стрибка на високому рівні складності.

Окрім критичних дефектів, під час тестування виявлено кілька незначних проблем: незначні артефакти тіней на межах тайлів підлоги при максимальній швидкості (усунено коригуванням налаштувань Shadow Distance), а також незначне зміщення тексту рахунку при роздільній здатності нижче 720p (виправлено через налаштування якоря елемента в UMG).

4.4 Аналіз продуктивності та оптимізація

Продуктивність мобільної гри є критично важливим показником якості, оскільки нестабільна частота кадрів або перегрів пристрою призводять до негативного досвіду користувача та видалення застосунку. Для DinoRunner встановлено цільовий показник – стабільні 60 FPS протягом усієї ігрової сесії при помірному споживанні пам'яті (не більше 350 МБ RAM).

Вимірювання продуктивності проводилось засобами вбудованого профайлера Unreal Engine (команда stat fps, stat unit, memreport) [14] як в редакторі, так і на реальному пристрої через консоль з'єднання по USB. Додатково використовувався Android GPU Inspector для детального аналізу навантаження на графічний процесор.

За результатами початкового профілювання виявлено кілька вузьких місць. По-перше, надмірна кількість Draw Calls на кадр (понад 200) через відсутність інстансингу для однотипних перешкод. По-друге, текстури персонажів і фону мали роздільну здатність 2048×2048, що надмірно для мобільного пристрою. По-третє, постійне створення та знищення об'єктів-перешкод без Object Pooling спричиняло часті паузи збирача сміття.

Для усунення виявлених проблем застосовано комплекс оптимізаційних заходів, результати яких наведено в таблиці 4.1.

Таблиця 4.1

Результати оптимізації продуктивності DinoRunner

Захід оптимізації	Показник до	Показник після	Покращення
Увімкнення Instanced Static Mesh для перешкод	~230 Draw Calls	~85 Draw Calls	-63%
Зменшення роздільної здатності текстур до 512×512	~420 МБ RAM	~210 МБ RAM	-50%

Продовження таблиці 4.1

Захід оптимізації	Показник до	Показник після	Покращення
Впровадження Object Pooling для перешкод	Паузи до 80 мс	Паузи < 5 мс	-94%
LOD для 3D-моделей динозаврів	~45 FPS	~60 FPS	+33%
Стиснення текстур ETC2 (Android)	APK ~180 МБ	APK ~95 МБ	-47%
Вимкнення динамічних тіней для фону	GPU ~85%	GPU ~55%	-35%

Після виконання всіх оптимізаційних заходів застосунок стабільно працює на рівні 60 FPS протягом 30-хвилинного тестового сеансу. Розмір APK-файлу зменшено до 95 МБ, що спрощує завантаження через мобільний інтернет.

Додатково налаштовано параметри якості графіки в Unreal Engine [14] під мобільну платформу; налаштовано Mobile HDR відповідно до вимог платформи; встановлено Mobile Shader Permutation Reduction для зменшення часу компіляції шейдерів.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальне завдання розробки мобільного пізнавально-розважального застосунку жанру runner на основі ігрового рушія Unreal Engine [14]. Отримані результати повністю відповідають меті та завданням, визначеним у вступі.

1. Проведено аналіз жанру runner як одного з найпопулярніших напрямків мобільного геймінгу. Встановлено, що ключовими характеристиками жанру є простота керування, динамічний темп та процедурна генерація контенту. Виявлено перспективний тренд – інтеграція освітнього матеріалу в ігровий процес, що стало концептуальною основою DinoRunner. Проаналізовано три провідних ігрових рушії – Unreal Engine [14], Unity [15] та Godot Engine [16] – і обґрунтовано вибір Unreal Engine завдяки потужному графічному рушію, підтримці платформи Android та гнучкій системі Blueprints.

2. Спроектовано модульну архітектуру застосунку DinoRunner, що складається з чотирьох незалежних модулів: керування персонажем, генерації рівня, інтерфейсу користувача та пізнавального контенту. Побудовано модель варіантів використання сценаріями взаємодії гравця з системою. Розроблено концепцію інтерфейсу з п’яти екранів та спроектовано шість ключових ігрових механік із параметричною моделлю наростання складності.

3. Реалізовано повнофункціональний мобільний застосунок DinoRunner засобами Unreal Engine [14]. Система керування персонажем побудована на базі класу ACharacter з підтримкою сенсорного вводу через Enhanced Input – реалізовано стрибок та ухилення. Система процедурної генерації перешкод забезпечує нескінченний варіативний ігровий процес з динамічним наростанням складності. Інтерфейс користувача реалізовано засобами UMG з підтримкою DPI Scaling для коректного відображення на Android-пристроях різних розмірів.

4. Реалізовано унікальний модуль пізнавального контенту на основі DataTable із цікавими науково достовірними фактами про динозаврів різних

геологічних ер. Додатково реалізовано екран вибору персонажів із внутрішньоігровою валютою Meats, що розширює ігровий досвід та підвищує реграбельність.

5. Проведено комплексне тестування застосунку: виконано 15 тест-кейсів із результатом Passed, виявлено та усунено 3 функціональні дефекти. За результатами профілювання та оптимізації (інстансинг, LODs [17], стиснення текстур) досягнуто стабільних 60 FPS.

Апробацію результатів дослідження здійснено шляхом публікації тез доповідей у матеріалах всеукраїнських наукових конференцій, присвячених сучасним інформаційним технологіям та цифровому розвитку. Основні результати роботи висвітлено у темах: «Розвиток користувачів у жанрі runner» [21] та «Жанрова концентрація історичного контенту в мобільних іграх: причини та наслідки»[22], що підтверджує наукову та практичну значущість проведеного дослідження.

Перспективами подальшого розвитку застосунку є розширення бази пізнавального контенту, додавання нових динозаврів, підтримка платформи iOS, впровадження системи досягнень та локалізація інтерфейсу кількома мовами.

ПЕРЕЛІК ПОСИЛАНЬ

1. Subway Surfers [Електроний ресурс] Режим доступу: https://play.google.com/store/apps/details?id=com.kiloo.subwaysurf (дата звернення 22.02.2026)

2. Temple Run [Електроний ресурс] Режим доступу: https://play.google.com/store/apps/details/Temple_Run?id=com.imangi.templerun&hl=uk (дата звернення 22.02.2026)

3. Дослідження Game-based Learning [Електроний ресурс] Режим доступу: https://www.ebsco.com/research-starters/education/game-based-learning (дата звернення 23.02.2026)

4. Концепція Джеймса Пола Гі [Електроний ресурс] Режим доступу: https://blog.ufes.br/kyriafinardi/files/2017/10/What-Video-Games-Have-to-Teach-us-About-Learning-and-Literacy-2003.-ilovepdf-compressed.pdf (дата звернення 23.02.2026)

5. Discovery Tour від Ubisoft [Електроний ресурс] Режим доступу: https://store.ubisoft.com/ie/assassins-creed-discovery-tour?lang=uk-UA (дата звернення 25.02.2026)

6. Foldit https://fold.it/ (дата звернення 25.02.2026)

7. Ancestors: The Humankind Odyssey [Електроний ресурс] Режим доступу: https://ancestorsgame.com/en/ (дата звернення 25.02.2026)

8. Ринок віртуальної реальності в ігровій індустрії [Електроний ресурс] Режим доступу: https://www.grandviewresearch.com/industry-analysis/virtual-reality-in-gaming-market (дата звернення 26.02.2026)

9. Стан мобільних ігор [Електроний ресурс] Режим доступу: https://sensortower.com/blog/state-of-mobile-gaming-2024 (дата звернення 26.02.2026)

10. Dino Run. [Електроний ресурс] Режим доступу: https://play.google.com/store/apps/details?id=com.spiky.DinoRun&hl=uk (дата звернення 25.02.2026)

11. Endless Alphabet (Originator) [Електроний ресурс] Режим доступу:

https://play.google.com/store/apps/details?id=com.originatorkids.EndlessAlphabet&hl=uk (дата звернення 25.02.2026)

12. Jurassic World Alive (Ludia) [Електроний ресурс] Режим доступу: https://play.google.com/store/apps/details/Jurassic_World_Alive?id=com.ludia.jw2&hl=uk (дата звернення 25.02.2026)

13. Dinosaur Game Chrome [Електроний ресурс] Режим доступу: https://chromedino.com/ (дата звернення 25.02.2026)

14. Unreal Engine [Електроний ресурс] Режим доступу: https://www.unrealengine.com/en-US/unreal-engine-5 (дата звернення 10.03.2026)

15. Unity [Електроний ресурс] Режим доступу: https://unity.com/ (дата звернення 10.03.2026)

16. Godot [Електроний ресурс] Режим доступу: https://godotengine.org/uk/ (дата звернення 10.03.2026)

17. Рівень деталізації сітки (LOD) [Електроний ресурс] Режим доступу: https://docs.godotengine.org/uk/4.x/tutorials/3d/mesh_lod.html (дата звернення 02.04.2026)

18. Visual Studio [Електроний ресурс] Режим доступу: https://visualstudio.microsoft.com/ (дата звернення 12.03.2026)

19. Android SDK [Електроний ресурс] Режим доступу: https://developer.android.com/tools/releases/platform-tools (дата звернення 15.04.2026)

20. Git [Електроний ресурс] Режим доступу: https://git-scm.com/ (дата звернення 11.03.2026)

21. Грисик В.О., Герцюк М.М. Розвиток користувачів у жанрі runner. // Всеукраїнська науково-технічна конференція «Технології цифрового розвитку: програмні системи та децентралізація», 1 травня 2026 р., Збірник тез. Київ, 2026, (подано до друку)

22. Грисик В.О., Герцюк М.М. Жанрова концентрація історичного контенту в мобільних іграх: причини та наслідки. // VI Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2026 р., Збірник тез. Київ, 2026, (подано до друку).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ТЕХНОЛОГІЙ ЦИФРОВОГО РОЗВИТКУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Цифровий пізнавально-розважальний мобільний застосунок на основі Unreal Engine»

Виконав студент 4 курсу

Групи ТЦР-41

Грисик Владислав Олександрович

Керівник роботи

ст. в кафедри, доктор філософії, Герцюк Микола Модестович

Київ-2026

2

Мета, об’єкт та предмет дослідження

Мета роботи

Метою цієї роботи є підвищення пізнавального контенту про доісторичний світ у жанрі runner для платформи Android. (об’єкт і предмет)

Об’єкт дослідження

Процес розробки мобільних ігрових застосунків жанру runner за допомогою модульних компонентів з використанням сучасних ігрових рушіїв

Предмет дослідження

Методи і засоби розробки мобільних ігрових застосунків.

Основні завдання для досягнення мети

1

1. Аналіз жанру та аналогів

Огляд механік runner, аналіз подібних застосунків та Edutainment-рішень.

2

2. Формулювання вимог

Функціональні та нефункціональні вимоги до застосунку

3

3. Вибір ігрового рушія

Критерії вибору рушія що найкраще відповідає поставленим вимогам

4

4. Реалізація ключових модулів

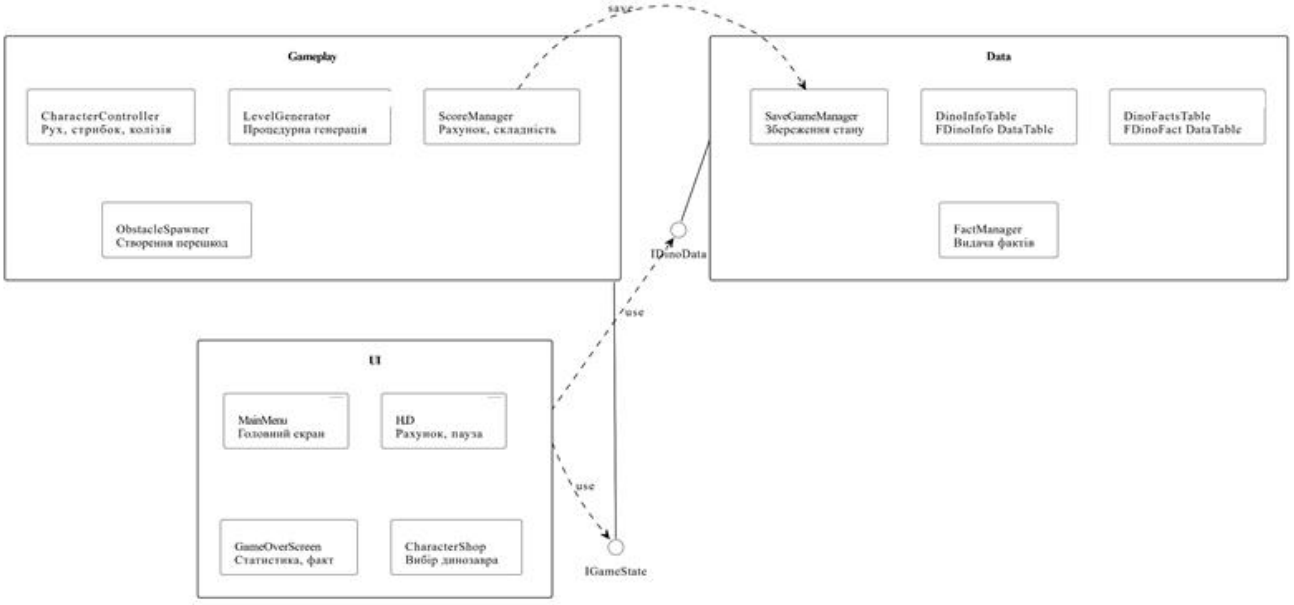
Управління персонажем, процедурна генерація перешкод, UI, інтеграція освітнього контенту.

5

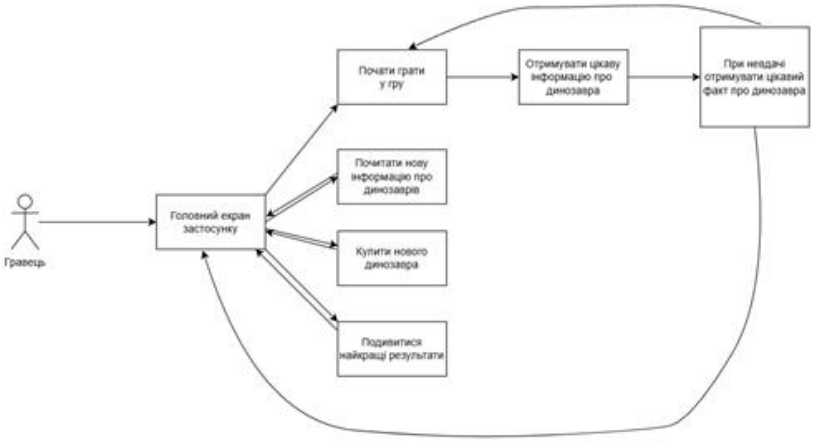
5. Тестування та оптимізація

Функціональне та нефункціональне тестування додатку та забезпечення 60 FPS на цільових пристроях.

Компонентна діаграма об’єкту дослідження



План використання застосунку



Проектування інтерфейсу користувача

Актуальність теми

Ринок мобільних ігор є дуже динамічним та зростаючим, а жанр раннерів особливо популярний завдяки своїй доступності та коротким ігровим сесіям. Розробка DinoRunner, гри, яка поєднує раннер з інформацією та цікавими фактами про доісторичний світ, відповідає сучасним тенденціям в інноваційному дизайні ігор та цифровій освіті.

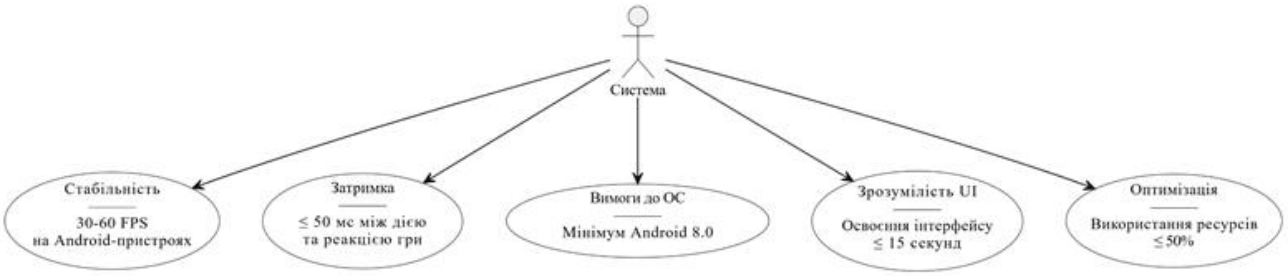
Аналіз аналогів

Застосунок	Жанр	Якість 3D моделей	Освітній контент	Платформа
Dino Run	Runner	Мінімальна	Мінімальний	PC/Android
Endless Alphabet	Навчальний	Відсутня	Повний	iOS/Android
Jurassic World Alive	AR/Колекціонування	Хороша	Повний	iOS/Android
DinoRunner	**Runner**	**Хороша**	**Повний**	**Android**

Функціональні вимоги застосунку

Системні вимоги до застосунку

Нефункціональні вимоги



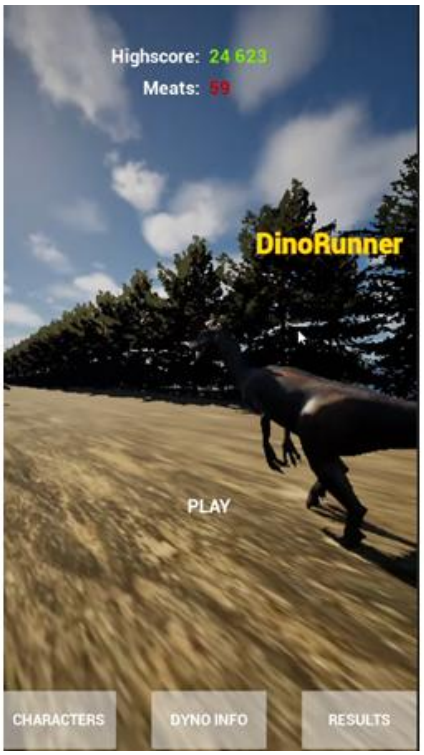
Ключові підтвердження роботи функціональних вимог:

- ✓ Модульна архітектура забезпечила хорошу взаємодію модулів Character, Level Generate, User Interface та компонентів. За допомогою цього було легко виявляти помилки та вирішувати їх.
- ✓ Edutainment ознайомлення з інформацією про динозаврів по бажанню користувача, поява фактів через відстань та на картці закінчення гри підтвердила збереження залученого гравця.

Підтвердження роботи нефункціональних вимог:

- ✓ За допомогою хорошої оптимізації застосунок працював у правильному діапазоні fps, затримки між діями користувача і реакцією гри та мав оптимальне використання енергії на пристроях.
- ✓ Оптимізація проводилась за допомогою:
  - + зменшення кількості вершин на моделі,
  - + зменшення якості текстур
  - + видалення непотрібних ефектів
  - + налаштування рушія для мобільного застосунку

Робота застосунку



Тестування застосунку

Тестування проходило у 3 етапи:

- ✓ Тестування у Unreal Engine
- ✓ Тестування у Standalone Game Mobile (UE але з обмеженням смартфону)
- ✓ Тестування на смартфонах

Тестування у UE складалось з:

- Тестування основних класів
- Тестування компонентів
- Тестування їх взаємодії під час гри

Тестування у Standalone Game Mobile (UE але з обмеженням смартфону)

- Тестування на склад 3д моделей
- Тестування тіней та ефектів
- Тестування налаштувань та оптимізації на смартфон

Тестування на смартфонах

- Перевірка проведених раніше тестувань
- Перевірка використання ресурсів смартфону

Апробація результатів дослідження

1. Грисик В.О., Герцюк М.М. Розвиток користувачів у жанрі runner. // Всеукраїнська науково-технічна конференція «Технології цифрового розвитку: програмні системи та децентралізація», 1 травня 2026 р., Збірник тез. Київ, 2026, (подано до друку)
2. Грисик В.О., Герцюк М.М. Жанрова концентрація історичного контентну в мобільних іграх: причини та наслідки. // VI Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті», 15 травня 2026 р., Збірник тез. Київ, 2026, (подано до друку).

Висновок та подальші кроки

У ході роботи було проведено аналіз жанру runner, досліджено сучасні аналоги та Edutainment-рішення, а також сформовано функціональні й нефункціональні вимоги до застосунку. Для реалізації проєкту обрано рушій Unreal Engine, який найкраще відповідає поставленим вимогам та потребам розробки. У застосунку реалізовано ключові модулі обраного жанру. Зокрема управління персонажем, генерацію перешкод, користувацький інтерфейс та інтеграцію освітнього контенту. Окрему увагу приділено тестуванню та оптимізації системи для забезпечення стабільної роботи на мобільних пристроях.

Дякую за увагу

Доповідь завершена чекаю на ваші питання

ДОДАТОК Б. РОБОТА ОСНОВНИХ МОДУЛІВ

Код GameMode – GM_DinoRunner

.h

```
#pragma once

#include "CoreMinimal.h"
#include "GameFramework/GameModeBase.h"
#include "GM_DinoRunner.generated.h"

class ABP_BaseTile;
class UUserWidget;

UCLASS()
class YOURGAME_API AGM_DinoRunner : public AGameModeBase
{
   GENERATED_BODY()

protected:
   virtual void BeginPlay() override;

public:
   UPROPERTY()
   ABP_BaseTile* BaseTileRef;

   UPROPERTY()
   FVector NextSpawnPoint;

   UPROPERTY(EditAnywhere)
   TSubclassOf<UUserWidget> MainMenuClass;

   UPROPERTY()
   UUserWidget* MainMenu;

   UFUNCTION()
   void Spawn_Tiles(int32 Amount);

   // HUD
   UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="UI")
   TSubclassOf<UUserWidget> PlayerHUDClass;

   UPROPERTY()
   UUserWidget* PlayerHUD_Ref;

   // Player
   UPROPERTY()
   class AMyDinoCharacter* Player_Ref;

   // Game state
   UPROPERTY()
   bool Game_Started = false;

   // Start Game
   UFUNCTION(BlueprintCallable)
   void StartGame();

   UFUNCTION()
   void IncreaseSpeed();

   UFUNCTION()
   void IncreasePoints();
};
```

.cpp

```
#include "GM_DinoRunner.h"
#include "Kismet/GameplayStatics.h"
#include "Blueprint/UserWidget.h"
#include "Engine/World.h"
#include "BP_BaseTile.h" // твій tile
#include "Components/ArrowComponent.h"

void AGM_DinoRunner::BeginPlay()
{
   Super::BeginPlay();

   // 1. Get all tiles
   TArray<AActor*> FoundTiles;
   UGameplayStatics::GetAllActorsOfClass(GetWorld(),
ABP_BaseTile::StaticClass(), FoundTiles);

   if (FoundTiles.Num() > 0)
   {
      // 2. Беремо перший
      BaseTileRef = Cast<ABP_BaseTile>(FoundTiles[0]);

      if (BaseTileRef)
      {
         // 3. Передаємо GameMode
         BaseTileRef->GameModeRef = this;

         // 4. Беремо Spawn Point
         if (BaseTileRef->Spawn_Point)
         {
            NextSpawnPoint = BaseTileRef->Spawn_Point->GetComponentLo
         }
      }
   }

   // 5. Spawn tiles
   Spawn_Tiles(10);

   // 6. Create Widget
   if (MainMenuClass)
   {
      APlayerController* PC = GetWorld()->GetFirstPlayerController();

      MainMenu = CreateWidget<UUserWidget>(PC, MainMenuClass);

      if (MainMenu)
      {
         UMainMenu* Menu = Cast<UMainMenu>(MainMenu);
         if (Menu)
         {
            Menu->GM_Dodger_Runner = this;
         }

         // 7. Add to viewport
         MainMenu->AddToViewport();
      }
   }
}

void AGM_DinoRunner::StartGame()
{
   // 1. Game Started
   Game_Started = true;

   APlayerController* PC = UGameplayStatics::GetPlayerController(this, 0);

   // 2. Create HUD
   if (PlayerHUDClass && PC)
   {
      PlayerHUD_Ref = CreateWidget<UUserWidget>(PC, PlayerHUDClass)

      if (PlayerHUD_Ref)
      {
         PlayerHUD_Ref->AddToViewport();
      }
   }

   // 3. Set Camera
   if (PC && Player_Ref && Player_Ref->Camera_Ref)
   {
      PC->SetViewTargetWithBlend(
         Player_Ref->Camera_Ref,
         0.3f // BlendTime
      );
   }

   // 4. Start gameplay systems
   IncreaseSpeed();
   IncreasePoints();
}
void AGM_DinoRunner::Spawn_Tiles(int32 Amount)
{
if (!TileClass) return;

for (int32 i = 0; i < Amount; i++)
{
// 1. Transform
FTransform SpawnTransform;
SpawnTransform.SetLocation(NextSpawnPoint);
SpawnTransform.SetRotation(FQuat::Identity);
SpawnTransform.SetScale3D(FVector(1.f));

// 2. Spawn Actor
ABP_BaseTile* SpawnedTile = GetWorld()->SpawnActor<ABP_BaseTile>(
TileClass,
SpawnTransform
);

if (!SpawnedTile) continue;

// 3. Set BaseTileRef
BaseTileRef = SpawnedTile;

// 4. We're passing GameMode
SpawnedTile->GameModeRef = this;

// 5. update NextSpawnPoint
if (SpawnedTile->Spawn_Point)
{
NextSpawnPoint = SpawnedTile->Spawn_Point->GetComponentLocation();
}

// 6. Call Main_Spawner
Main_Spawner();
}
}

void AGM_DinoRunner::Main_Spawner()
{
if (!BaseTileRef) return;

for (int32 i = 1; i <= 5; i++)
{
if (bCanSpawn)
{
// 1. Вибираємо випадковий індекс
int32 Index = FMath::RandRange(1, 5);

UArrowComponent* ChosenPoint = nullptr;

switch (Index)
{
case 1: ChosenPoint = BaseTileRef->Spawn_Location1; break;
case 2: ChosenPoint = BaseTileRef->Spawn_Location2; break;
case 3: ChosenPoint = BaseTileRef->Spawn_Location3; break;
case 4: ChosenPoint = BaseTileRef->Spawn_Location4; break;
case 5: ChosenPoint = BaseTileRef->Spawn_Location5; break;
default: break;
}

// 2. Set spawn point
if (ChosenPoint)
{
Obstacle_Spawn_Point = ChosenPoint;
```

```
        }

        // 3. Spawn obstacle
        Spawn_Obstacles();

        // 4. Toggle
        bCanSpawn = false;
        }
        else
        {
        bCanSpawn = true;
        }
        }
        }
        void AGM_DinoRunner::Spawn_Meats(int32 Amount, bool Already_Spawni
double Spawn_After_Location)
        {
          if (Already_Spawning)
          {
            if (SomeInternalCounter == 0)
            {
              Meat_Amount_To_Spawn = UKismetMathLibrary::RandomIntegerIn
3);
            }
            else
            {
              Meat_Amount_To_Spawn = SomeInternalCounter;
            }

            Spawn_Meats(Meat_Amount_To_Spawn, true, 300.0);
          }
          else
          {
            Meat_Amount_To_Spawn = Amount;
            SomeInternalCounter = Amount;

            bool bShouldStartSpawning = (Amount > 0);

            Spawn_Meats(Meat_Amount_To_Spawn, bShouldStartSpawning,
_After_Location);
          }
        }
        void AGM_DinoRunner::Set_Character()
        {
          USaveGame* LoadedGame =
:playStatics::LoadGameFromSlot(TEXT("Savegame"), 0);
          SaveGameRef = Cast<UBP_DinoSaveGame>(LoadedGame);

          if (SaveGameRef)
          {
            Current_Character_Index = SaveGameRef->Character_Index_Selected;

            if (SaveGameRef->Characters_Info.IsValidIndex(Current_Character_Index
            {
              FCharacter_Data_Sructure SelectedData = SaveGameRef-
cters_Info[Current_Character_Index];

              if (Player_Ref && Player_Ref->GetMesh())
              {
                Player_Ref->GetMesh()->SetSkeletalMeshAsset(SelectedData.Mesh);

                FVector NewScale = FVector(SelectedData.Scale);
                Player_Ref->GetMesh()->SetWorldScale3D(NewScale);
              }
            }
          }
        }
```

ДОДАТОК В. BLUEPRINTS ОСНОВНИХ МОДУЛІВ

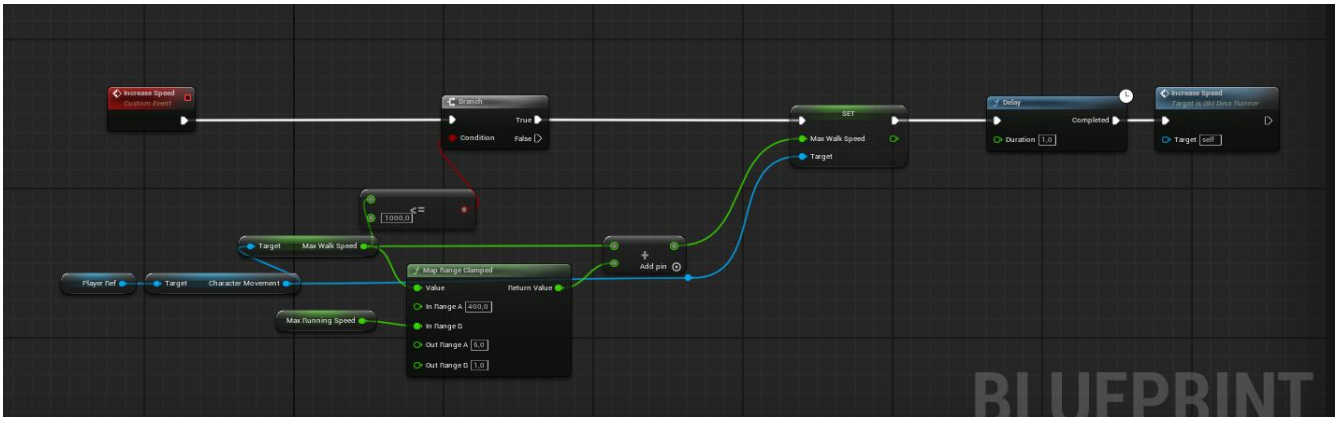


Рис. В.1 Збільшення швидкості

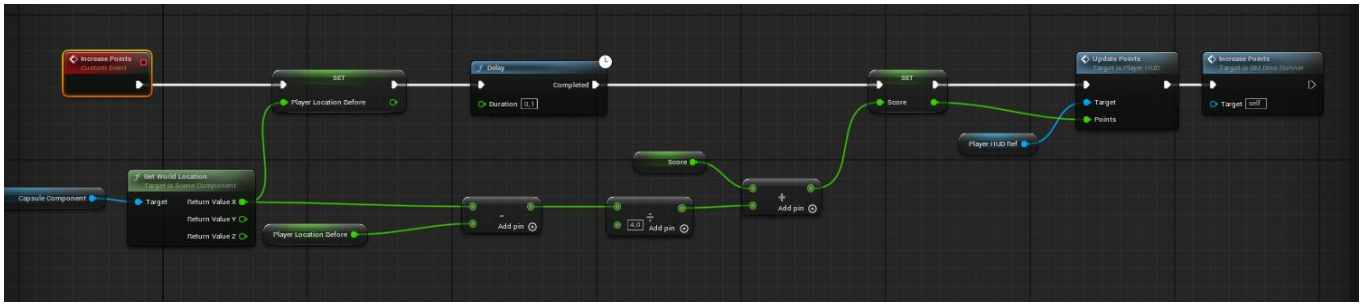


Рис. В.2 Збільшення рахунку

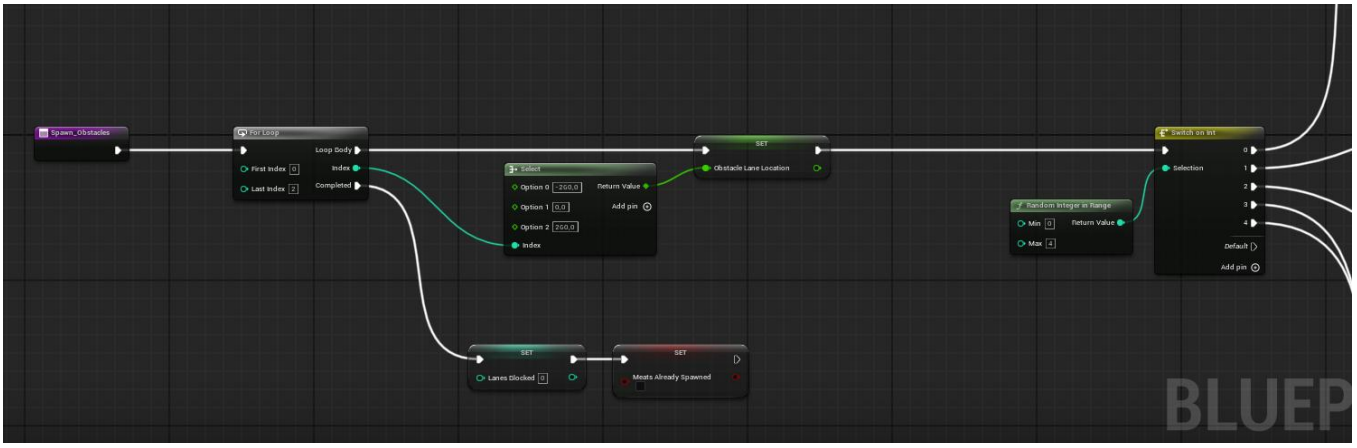


Рис. В.3 Випадкова поява перешкод

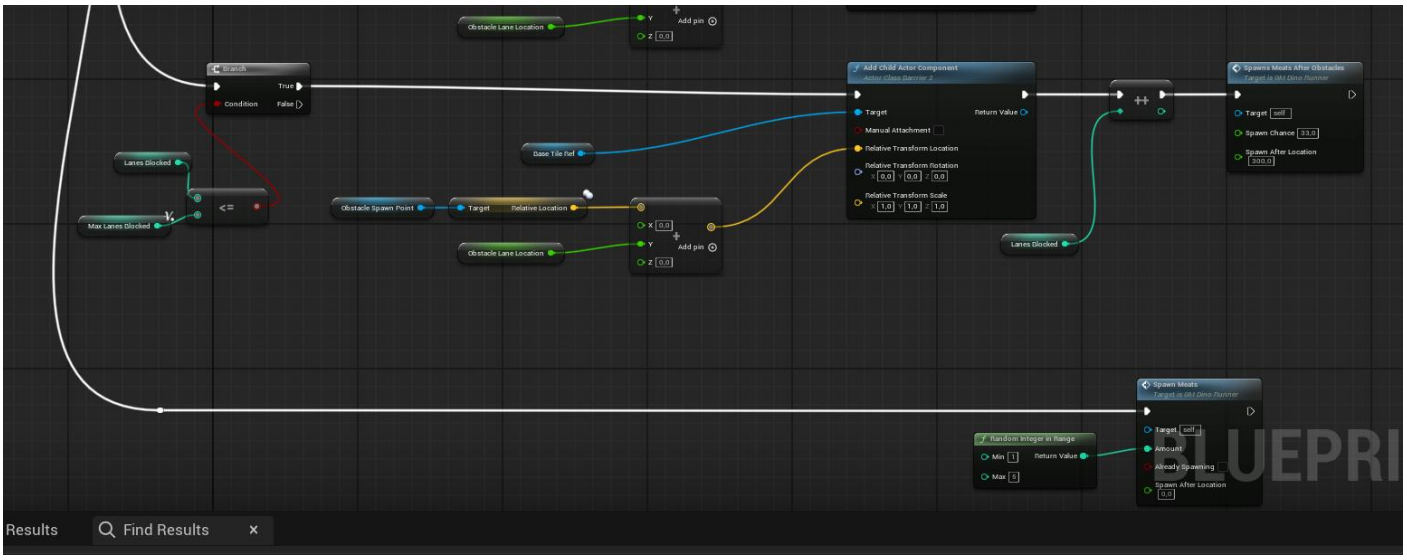


Рис. В.4 Поява перешкод на різних локаціях та м’яса з ними

Код головного персонажу

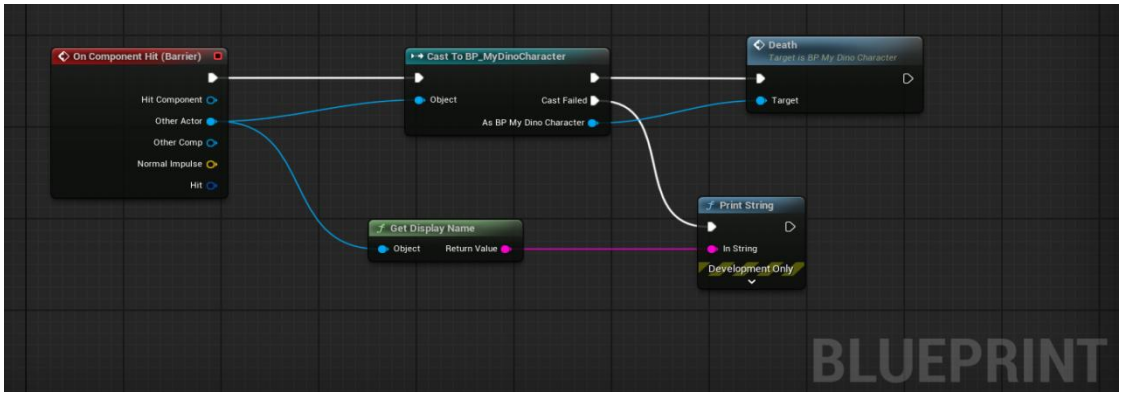


Рис. В.5 Виклик події Death

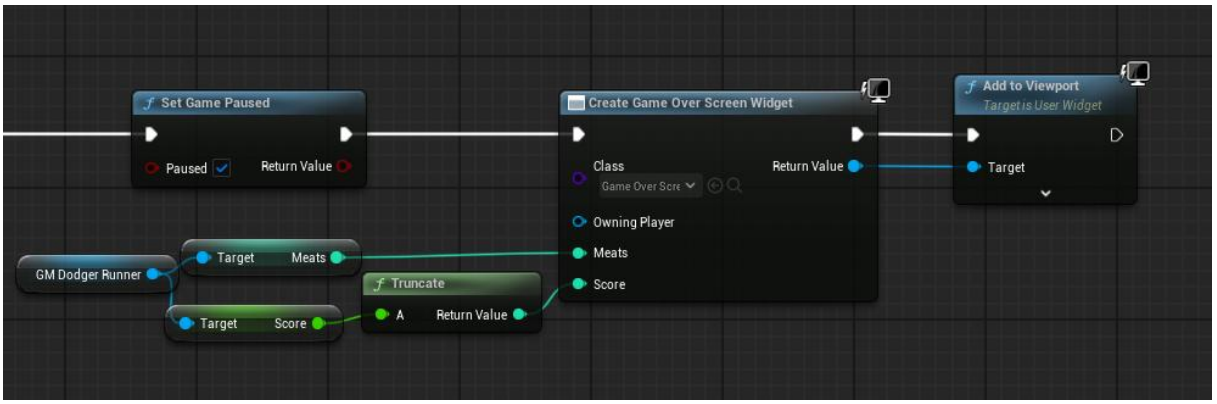


Рис. В.6 Перехід гри в паузу та віджет закінчення сесії

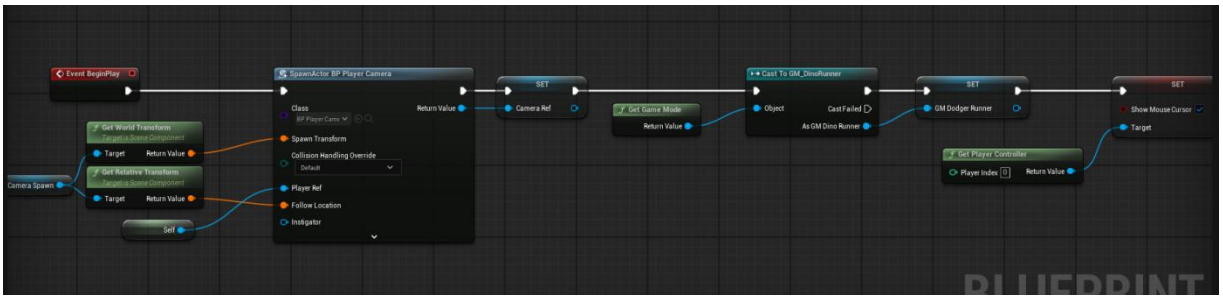


Рис. В.7 Додавання камери та ініціалізація GM

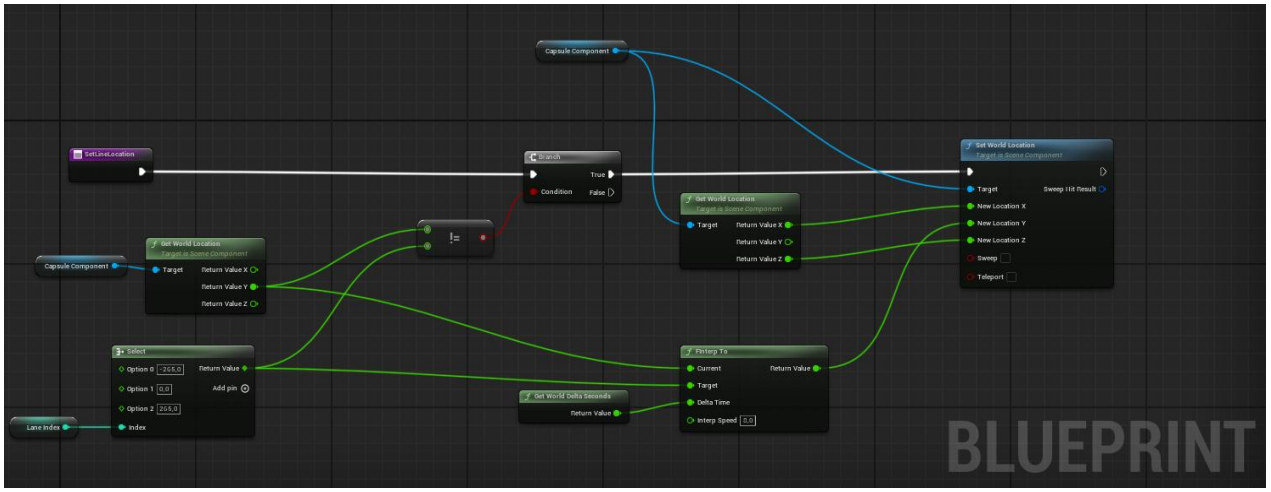


Рис. В.8 Функція пересування на частини мапи

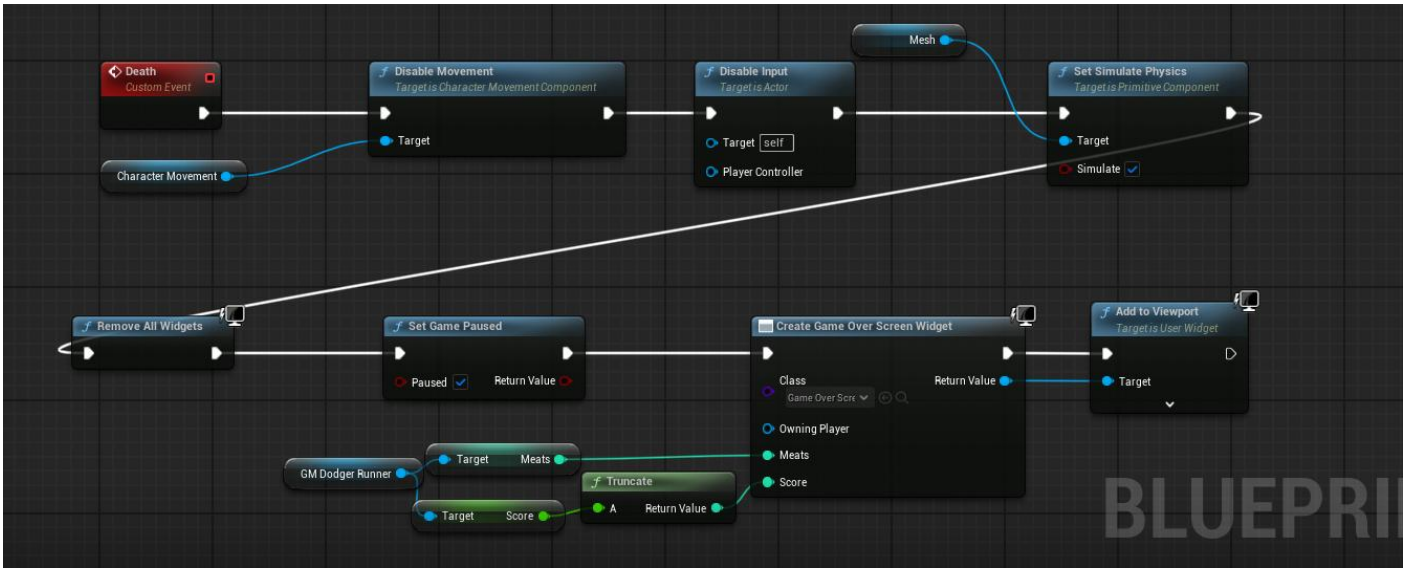


Рис. В.9 Подія Death

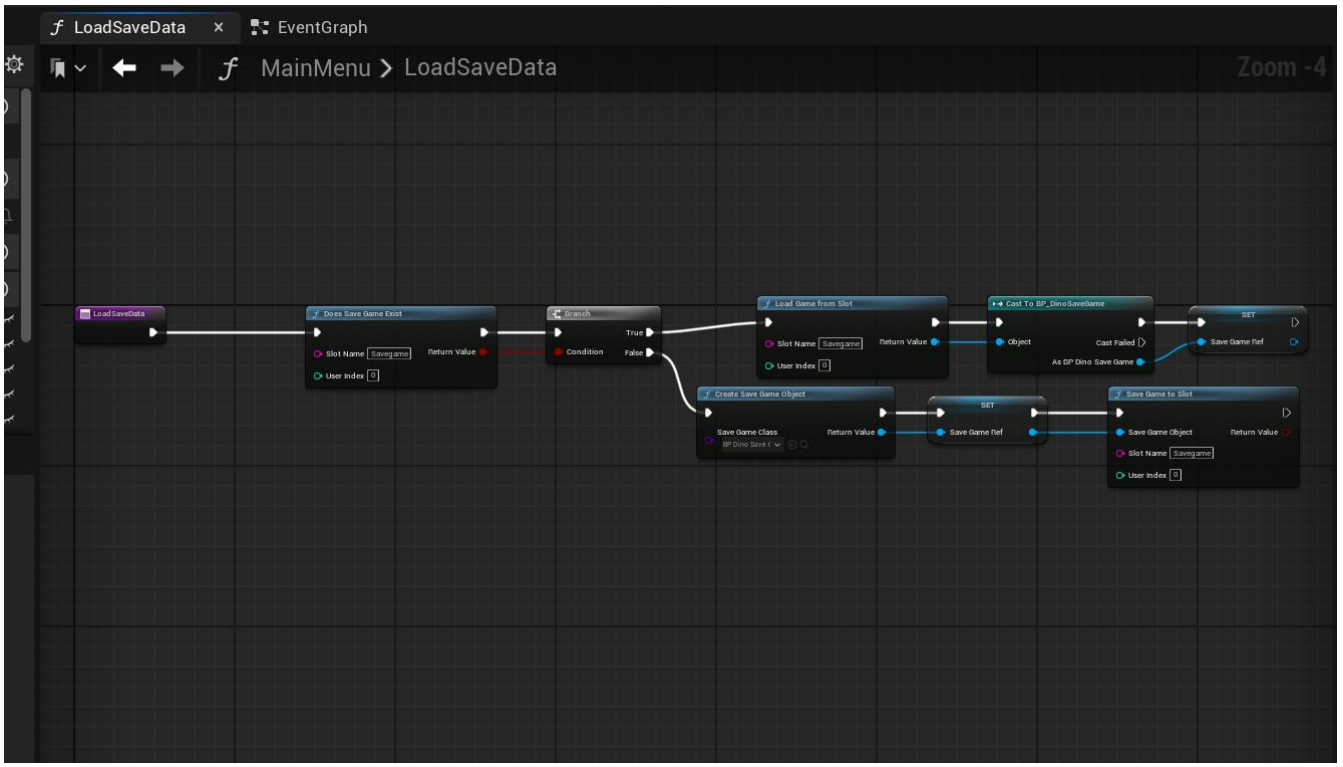


Рис. В.10 Завантаження інформації зі збережень