МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Острозька академія»
Навчально науковий інститут ІТ та бізнесу
Кафедра інформаційних технологій та аналітики даних

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня магістра

на тему: «**Управління проєктом створення мобільного ранера з елементами шутера на рушії Unreal Engine»**

Виконав: студент 2 курсу, групи МУП-2
другого (магістерського) рівня вищої освіти
спеціальності 122 Комп'ютерні науки
ОПП «Управління проєктами»
Євтушок Максим Едуардович

Керівник: ***Місай В.В., викладач, фахівець-практик кафедри ІТБ***

Рецензент: *кандидат технічних наук, доцент, доцент кафедри прикладної математики та кібербезпеки Донецького національного університету імені Василя Стуса*
Загоруйко Любов Василівна

РОБОТА ДОПУЩЕНА ДО ЗАХИСТУ

Завідувач кафедри інформаційних технологій та аналітики даних
_____ (проф., д.е.н. Кривицька О.Р.)

Протокол № 5 від « 04 » грудня 2025 р.

Острог, 2025

АНОТАЦІЯ
кваліфікаційної роботи
на здобуття освітнього ступеня магістра

Тема: *Управління проєктом створення мобільного ранера з елементами шутера на рушії Unreal Engine*

Автор: Євтушок Максим Едуардович

Науковий керівник: Місай В. В., *викладач, фахівець-практик кафедри ІТБ*

Захищена «……»…… 2025 року.

Пояснювальна записка до кваліфікаційної роботи: *с.68, рис.31, табл.1, джерел 29.*

Ключові слова: *Unreal Engine, Blueprints, мобільна гра, раннер, шутер, геймдизайн, управління IT-проєктами.*

Короткий зміст праці:

У цій кваліфікаційній роботі було розроблено мобільну гру жанру раннера з елементами шутера на рушії Unreal Engine 5.4. У процесі реалізації використовувалися сучасні інструменти геймдеву та управління проєктами, зокрема Unreal Engine, Blueprints, Git. Основною метою роботи стало створення динамічної гри з можливістю вибору персонажа, системою стрільби, збором бонусів та збереженням прогресу користувача. У проєкті реалізовано геймплейну логіку, адаптивний інтерфейс користувача, систему генерації рівнів, оптимізацію під мобільні пристрої та механізми управління проєктом. Особливу увагу приділено підвищенню продуктивності гри: застосовано оптимізовані матеріали, налаштовано LOD-профілі, використано техніки instancing. Проведений аналіз та результати розробки демонструють практичне застосування технологій Unreal Engine, а також відображають компетентності автора у сфері розробки мобільних ігор та управління IT-проєктами. Отримані результати можуть бути використані для подальшої розробки, масштабування або адаптації гри під інші платформи.

_____

ANNOTATION
of qualification paper
for master's degree

Theme: *Development of a Mobile Runner Game with Shooter Elements Using Unreal Engine.*

Author: *Yevtushok Maksym*

Supervisor: *Misay V. V., lecturer, specialist-practitioner at the ITB department*

Defended **"....."..... 2025.**

Explanatory note to the qualification work: *p.68, pic.31, table 1, sources29.*

Keywords: *Unreal Engine, Blueprints, mobile game, runner, shooter, game design, IT project management.*

Summary of the work:

In this qualification work, a mobile game in the runner genre with shooter elements was developed using the Unreal Engine 5.4 framework. The implementation involved modern gamedev and project management tools, including Unreal Engine, Blueprints, and Git. The main objective of the work was to create a dynamic game featuring character selection, a shooting system, bonus collection, and user progress saving. The project includes gameplay logic, an adaptive user interface, a level generation system, optimization for mobile devices, and core project management mechanisms. Special attention was paid to improving game performance by applying optimized materials, configuring LOD profiles, and using instancing techniques. The conducted analysis and development results demonstrate the practical application of Unreal Engine technologies and reflect the author's competencies in mobile game development and IT project management. The obtained results can be used for further development, scaling, or adaptation of the game for other platforms.

_____

ЗМІСТ

ВСТУП **3**
РОЗДІЛ 1
АНАЛІЗ ОСОБЛИВОСТЕЙ УПРАВЛІННЯ ПРОЄКТАМИ У СФЕРІ ГЕЙМДЕВУ ТА ОГЛЯД СУЧАСНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ МОБІЛЬНИХ ІГОР **5**
1.1. Особливості управління проєктами у сфері геймдеву 5
1.2.Аналіз сучасних рушіїв для створення ігор та обґрунтування вибору Unreal Engine 7
1.3. Огляд наявних аналогів 9
1.4. Постановка задачі 13
1.5. SWOT - аналіз 14
Висновки до розділу 1 17
РОЗДІЛ 2
ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ **19**
2.1. Аналіз предметної області 19
2.2. Архітектура гри 21
2.3. Розробка інтерфейсу користувача та системи меню 24
Висновки до розділу 2 26
РОЗДІЛ 3
ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ **28**
3.1. Засоби розробки 28
3.2. Вимоги до технічного та програмного забезпечення 29
3.3. Опис програмної реалізації 30
Висновки до розділу 3 52
РОЗДІЛ 4
ПІДГОТОВКА ГРИ ДО РЕЛІЗУ **54**
4.1 Реліз на різних платформах 54
4.2 Тестування та налагодження 55
4.3 Балансування геймплею та оптимізація інтерфейсу (UX/UI) 55
4.4 Маркетингова підготовка 56
Висновки до розділу 4 59
ВИСНОВКИ **61**
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ **64**

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

UE / UE5 / UE 5.4 - Unreal Engine

BP - Blueprints

HISM / ISM - Hierarchical Instanced Static Mesh / Instanced Static Mesh

LOD - Level of Detail

UI - User Interface

HUD - Heads-Up Display

UMG - Unreal Motion Graphics

FPS - Frames Per Second

GPU / CPU - Graphics Processing Unit / Central Processing Unit

QA - Quality Assurance

UX - User Experience

APK - Android Package Kit

PC - Personal Computer

VFX - Visual Effects

ВСТУП

Сьогодні ігрова індустрія є однією з найпотужніших галузей цифрової економіки, яка об’єднує розробників, дизайнерів, аналітиків, тестувальників та менеджерів проєктів у всьому світі. Ігри більше не сприймаються лише як розвага - вони стали повноцінним бізнесом, який формує мільярдні прибутки та створює тисячі робочих місць. Завдяки швидкому розвитку мобільних технологій і підвищенню продуктивності смартфонів, усе більшої популярності набувають мобільні ігри, що забезпечують миттєвий доступ до інтерактивного контенту та залучають мільйони користувачів щодня.

Розробка навіть простої гри є складним процесом, який потребує не лише технічних знань, а й ефективного управління проєктом. У сучасних умовах успішна реалізація ігрового продукту неможлива без планування етапів розробки, управління ризиками, контролю якості та організації роботи команди. Управління проєктом створення гри передбачає координацію багатьох аспектів - від підготовки концепції до тестування та випуску продукту.

У межах даної кваліфікаційної роботи створено мобільну гру в жанрі раннера з елементами шутера, розроблену на рушії Unreal Engine 5.4. У процесі розробки реалізовано рух персонажа, бойову систему зі стрільбою, механіку збору бонусів, систему збереження прогресу, а також інтерфейс користувача з меню паузи та відображенням очок. Для підвищення динамічності використано візуальні ефекти, а для покращення продуктивності - оптимізацію ресурсів і рівнів.

Для досягнення поставленої мети необхідно виконати такі завдання:

- Дослідити особливості управління проєктами у сфері геймдеву;
- Проаналізувати сучасні рушії для створення ігор та обґрунтувати вибір Unreal Engine 5.4;
- Огляд наявних аналогів
- Постановка задачі
- Розробити концепцію гри, визначити основні механіки та етапи реалізації;

- Аналіз предметної області
- Розробити інтерфейс користувача, систему меню;
- Засоби розробки
- Вимоги до технічного та програмного забезпечення
- Опис програмної реалізації
- Налаштувати оптимізацію продуктивності для мобільної платформи;
- Розробити календарний план і структуру управління проєктом;
- Провести тестування гри та оцінити ефективність реалізованого управління проєктом.
- Публікація гри в магазині ігор

Об’єктом дослідження є процес управління проєктом створення мобільної гри з використанням рушія Unreal Engine, а також застосування сучасних підходів до організації командної роботи у сфері розробки ігор.

Предметом дослідження є розробка мобільної гри в жанрі runner-shooter на рушії Unreal Engine для мобільних пристроїв.

РОЗДІЛ 1
АНАЛІЗ ОСОБЛИВОСТЕЙ УПРАВЛІННЯ ПРОЄКТАМИ У СФЕРІ ГЕЙМДЕВУ
ТА ОГЛЯД СУЧАСНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ МОБІЛЬНИХ ІГОР

1.1. Особливості управління проєктами у сфері геймдеву

Сфера розробки ігор (геймдев) є однією з найскладніших і найдинамічніших галузей IT-індустрії, у якій поєднуються творчість, програмна інженерія, дизайн, маркетинг та управління командами. На відміну від традиційних IT-проєктів, геймдев-проєкти мають підвищену невизначеність, високу залежність від користувацького досвіду і потребують ефективної координації між різними фахівцями - розробниками, художниками, сценаристами, геймдизайнерами та тестувальниками.

Особливістю управління проєктами у геймдеві є те, що успіх продукту визначається не лише технічною якістю, а й емоційним сприйняттям користувача. Це означає, що планування, контроль і реалізація задач повинні враховувати не тільки строки й бюджет, а й рівень залучення гравця, баланс складності, візуальну привабливість і цілісність ігрового процесу.

Управління геймдев-проєктом охоплює такі ключові напрями:

- **Керування життєвим циклом гри** - від ідеї, прототипу й препродакшену до релізу та пострелізної підтримки.
- **Планування контенту** - визначення масштабу гри, рівнів, персонажів, механік, систем досягнень і балансу.
- **Координація міждисциплінарних команд** - робота програмістів, художників, сценаристів і QA-фахівців має бути скоординована через систему управління завданнями.

- **Управління ризиками** - затримки з виробництвом контенту, технічна нестабільність, низька оптимізація або невідповідність очікуванням гравців.
- **Контроль якості (QA)** - багатоступеневе тестування, зокрема функціональне, графічне, балансне та UX-тестування.
- **Маркетингова складова** - планування промоційної кампанії, створення сторінок у магазинах, зворотний зв’язок із гравцями.

Залежно від масштабу проєкту, у геймдеві використовуються різні моделі управління:

- **Waterfall (каскадна модель)** - характерна для невеликих або одноосібних проєктів, коли послідовно виконуються всі етапи: планування, розробка, тестування, реліз.
- **Agile/Scrum** - найпоширеніший підхід у сучасних ігрових студіях, який передбачає гнучкість, ітеративну розробку й регулярні демонстрації результатів (спринти).
- **Kanban** - використовується для контролю потоку завдань і пріоритизації робіт під час активної фази розробки.

Система управління у геймдеві часто реалізується через такі інструменти, як Jira, Trello, Asana або Notion, які дозволяють вести беклог завдань, створювати спринти, контролювати дедлайни та відстежувати статус розробки [4]. Для спільної роботи з кодом і контентом застосовуються системи контролю версій (GitHub, Perforce), що забезпечують збереження змін і командну співпрацю.

Ключову роль у геймдев-проєкті відіграє менеджер проєкту (Project Manager), який координує всі процеси: від постановки завдань і контролю строків до комунікації з командою та стейкхолдерами. Менеджер відповідає не

лише за організацію роботи, а й за баланс між творчістю і дисципліною, що особливо важливо в креативному середовищі розробників ігор.

Таким чином, управління проєктами у сфері геймдеву поєднує принципи класичного проєктного менеджменту з методами гнучкої розробки, орієнтованої на гравця. Основним критерієм успіху виступає не лише виконання технічних вимог, а й досягнення високого рівня якості ігрового досвіду, що забезпечується ефективною командною взаємодією, адаптивним плануванням та постійним удосконаленням продукту.

1.2.Аналіз сучасних рушіїв для створення ігор та обґрунтування вибору Unreal Engine

Під час вибору рушія для створення гри типу runner-shooter ключовими критеріями стали продуктивність на Android/iOS і ПК (Windows), стабільні 50–60 FPS, підтримка Vulkan/Metal/DX12, адаптивна роздільність і пресети якості (Scalability). Важливими були засоби швидкого прототипування (візуальне скриптування, шаблони, Starter Content), модульність і масштабованість (розмежування логіки, UI, ефектів), зріла екосистема плагінів/документації, інструменти мобільної оптимізації (мобільні матеріали, контроль draw calls) і прозорі умови публікації у сторах. З огляду на ці вимоги порівнювалися Unity, Unreal Engine та Godot (додатково враховано CryEngine і GameMaker Studio як нішеві варіанти).

Unity вирізняється зручністю входу та великою бібліотекою ассетів, однак для top-tier 3D часто потребує більше ручної роботи зі шейдерами та ретельнішої оптимізації складних сцен. Godot приваблює відкритим кодом і легкістю, проте його стек 3D і мобільна оптимізація ще дозрівають, тому для інтенсивних 3D-екшенів поступається конкурентам. CryEngine дає преміальну графіку, але має вищі вимоги та складніший пайплайн; GameMaker Studio чудовий для 2D. Сукупно це робить Unreal Engine найбільш збалансованим

вибором для кросплатформної 3D-гри (мобайл + ПК) з насиченим геймплеєм і вимогами до кінематографічності.

У Unreal Engine 5.4 вирішальним стало швидке складання логіки через Blueprints із можливістю реалізовувати продуктивно критичні елементи на C++ [1]. Це дає змогу оперативно реалізувати ядро раннера (автобіг, lane-switch, стрибок/слайд) і шутерні механіки (спавн перешкод, колізії, постріли) без надмірного коду. Архітектура GameMode–Pawn–PlayerController–HUD задає чіткий каркас керування та станів гри: у проєкті це втілено у BP_RunnerGameMode, модулях HUD та Inputs. Окремо важливо, що UE забезпечує єдину кодову базу для мобільних і ПК-збірок: у проєкті використовуються абстракції введення (торкання/свайпи → клавіатура/миша/геймпад), пресети якості для слабких/середніх мобільних і потужних ПК, а також різні тарґети рендерера (Vulkan/Metal/DX12).

Зрілі інструменти профілювання та Mobile/Standalone Preview дозволяють вимірювати frame time, навантаження Game/Draw/RHI й пам’ять прямо в редакторі як для мобільної, так і для ПК-збірки. Це критично для раннера, де безперервний потік об’єктів може збільшувати draw calls і тиск на пам’ять. У межах проєкту застосовано стрімінг треку через BP_Floor, прибирання відпрацьованих об’єктів BP_ObstacleDestroyer, кешування та пулінг повторюваних елементів, керування матеріальними інструкціями. Runner_SaveGame забезпечує збереження рекордів, а UMG/HUD - інтерфейс очок, паузи і екрани вибору персонажа (CS_Map, BP_CS_Characters) із однаковою логікою для мобайлу та ПК (різниться лише метод введення).

З позиції керованості проєктом Unreal Engine мінімізує контекст-світчинг: логіка, VFX, UI та профілювання виконуються в єдиній екосистемі, що полегшує календарний план (спринти під прототипування Blueprints, окремі ітерації оптимізації, QA й білди Android + Windows). Головні ризики - розмір білду та крутіша навчальна крива - нівелюються компресією ресурсів, instanced meshes, вимкненням зайвих постпроцесів, ретельним LOD-менеджментом. У підсумку Unreal Engine 5.4 забезпечує баланс між високою візуальною якістю,

швидким прототипуванням, контрольованою оптимізацією та кросплатформеною підтримкою (мобільні пристрої та ПК), що робить його обґрунтованим вибором для розробки даної гри.

1.3. Огляд наявних аналогів

Огляд наявних аналогів доцільно будувати навколо двох груп: класичні мобільні раннери та гібриди раннера з бойовими елементами. Класичні представники - Temple Run 2 та Subway Surfers - задають канон жанру: нескінченна траса, короткі сесії на 1–3 хвилини, процедурне складання “секцій” дороги, керування жестами (свайп угору - стрибок, униз - ковзання, ліворуч/праворуч - зміна смуги), прості й зрозумілі бустери (магніт, множник очок, щит) [7]. Вони показують, що утримання користувача забезпечують передбачувана “основа” геймплею та постійні мікроцілі - щоденні місії, сезонні події, колекції предметів [8].

Рис. 1.1.Карта та геймплей Temple Run
Джерело: https://surl.lu/jqchrf

Подібний підхід до побудови ігрового процесу демонструє свою ефективність протягом багатьох років: гравець отримує зрозумілу мету, короткі та динамічні сесії, а також постійний прогрес за рахунок збирання очок та бонусів. Така структура робить гру доступною для широкої аудиторії - як для користувачів, які грають кілька хвилин у транспорті, так і для тих, хто готовий проводити більше часу, виконуючи завдання та поліпшуючи результати.

Успіх Temple Run та подібних проєктів визначив ключові вимоги до сучасних раннерів: висока швидкість реакції, плавна анімація, візуальна привабливість та інтуїтивне керування. Саме ці елементи стають основою при створенні мобільних ігор цього жанру та напряму впливають на подальший дизайн, технічні рішення та механіки, реалізовані у власному проєкті [7].

Рис. 1.2. Показ геймплея Subway Surfers
Джерело: https://minireview.io/runner/subway-surfers

Minion Rush розвиває базову петлю завдяки мікрорівням із варіативними умовами (пробігти N метрів, зібрати X монет, уникнути Y перешкод) і вираженій мета-грі: прогрес персонажа, косметика, тимчасові події. Це важливо для проєктування прогресії: простий раннер без мета-цілей швидко “вигорає”,

натомість система завдань і колекцій продовжує життєвий цикл. Для нашого проєкту це означає потребу в базовій таблиці завдань/досягнень і екранах відстеження прогресу в HUD.

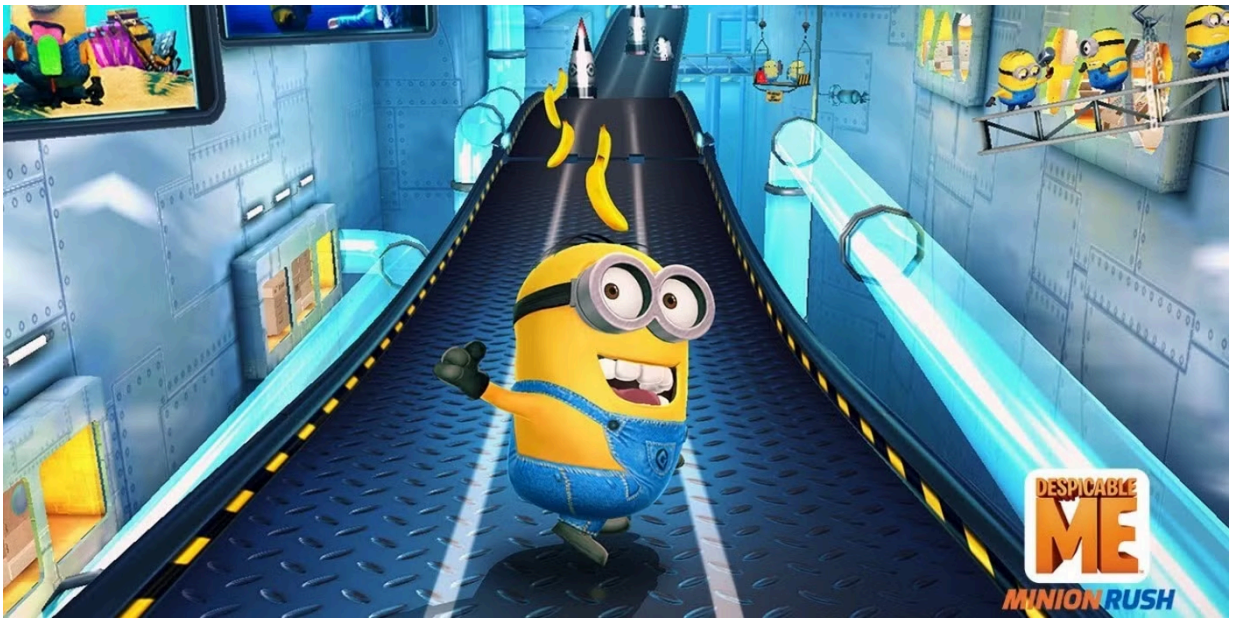

Рис. 1.3. Показ геймплея гри Minion Rush
Джерело: https://surli.cc/drdqfm

Гібриди раннера зі стрілянням демонструють різні способи інтеграції бою. Into the Dead поєднує безперервний біг від першої особи з дефіцитом боєзапасу та напругою ухилення/пострілу; тут стрільба - рідкісний, цінний ресурс, а траса підкидає моменти вибору (обійти чи витратити патрони). Zombie Highway робить акцент на прицільному знищенні загроз у русі та економному UX: мінімум кнопок, максимум автоматизації (автобіг + контекстні дії). Jetpack Joyride, попри сайд-скрол, показує, як перетворити стрільбу на фонову “постійно активну” дію з короткими піками сили (транспорт, щити), що добре масштабується під мобайл.

Окремої уваги заслуговують Blades of Brim і Lara Croft: Relic Run - приклади, де бойова складова (ближній бій/стрільба) поєднана з класичним lane-based рухом. У них удар/постріл зазвичай прив’язаний до смуги або до великого хитбокса ворога, аби не перевантажувати моторикою. Висновок для

нашого дизайну: на мобільних краще використовувати авто прицілювання або хітбокси.

Рис. 1.4. Загальне фото гри Blade's of Brim
Джерело: https://surl.li/erqqah

З погляду UX у більшості успішних аналогів дій небагато й вони взаємовиключні: свайпи керують рухом, тап - стрільбою/взаємодією. На ПК природними є WASD/стрілки для руху та клік/пробіл для стрільби/стрибка. Це підтверджує доцільність єдиного шару абстракції вводу: одна й та сама ігрова подія мапиться на різні пристрої керування. У нашому проєкті це реалізується через модулі Inputs і HUD, що спрощує кросплатформеність (мобайл + Windows).

Технологічно аналоги майже завжди використовують процедурну збірку траси з повторно використовуваних "секцій", object pooling (перешкоди, вороги, снаряди), рівні деталізації (LOD) та інстансинг для зменшення draw calls. Це критично для стабільного FPS на мобільних і напряму переноситься у нашу реалізацію: нескінченна доріжка збирається BP_Floor, прибирання відпрацьованих об'єктів виконує BP_ObstacleDestroyer, буст "магніт" реалізовано як BP_PowerUp_Magnet, стрільба - через BP_Projectile. Збереження

рекордів і прогресу підтримується локально (Runner_SaveGame), що відповідає практиці lightweight-проєктів без зовнішніх сервісів на старті.

Баланс складності в аналогах зазвичай нелінійний: швидкість поступово зростає, густина перешкод збільшується, а генератор секцій підмішує ризикованіші патерни. У бойових раннерах додатково налаштовують частоту появи цілей/патронів і “вікна” безпеки.

Моделі монетизації в топових раннерах типові: IAP на бустери/косметику, rewarded video за повтор/бонус, події й сезони. Навіть якщо проєкт навчальний, такі механіки задають структуру економіки: валюта (монети), синки (поліпшення, скіни), ліміти сесій (енергія або ускладнення). Це впливає на дизайн інтерфейсу (екран результатів, лідерборди, іконки трофеїв/рангів.

Підсумовуючи, аналіз аналогів підказує набір перевірених рішень для нашого runner-shooter: ядро з lane-based рухом і лаконічним управлінням, проста, але, базові підсилення (магніт, щит, множник), процедурна збірка траси з шаблонів, керований ріст складності. Всі ці висновки вже відображені в структурі проєкту: модульні Blueprints у папці Runner/Blueprints, карти MainMenu/CS_Map/Runner_Map/Loading для повного циклу, ресурси Meshes/Textures для економного рендерингу та система збережень Runner_SaveGame. Це забезпечує відповідність очікуванням користувачів жанру й готовність масштабувати продукт під мобільні пристрої та ПК.

1.4. Постановка задачі

Заплановано створити кросплатформну гру жанру runner-shooter на Unreal Engine 5.4 (повністю Blueprints) для мобільних пристроїв (Android/iOS) та ПК (Windows) з короткими динамічними сесіями, стабільною продуктивністю та різними профілями вводу (жести на мобільних, WASD/миша на ПК); передбачено: концепт і дизайн рівня з нескінченною трасою з модульних секцій та набір карт MainMenu/CS_Map/Runner_Map/Loading; систему керування

персонажем (автобіг, зміна смуг, стрибок/ковзання через Enhanced Input); механіку стрільби зі створенням снаряда у вигляді книг BP_Projectile; перешкоди зі спавном і параметрами складності; підсилення (монети, BP_PowerUp_Magnet) з налаштуванням дії й тривалості; інтерфейс UMG/HUD (очки, меню паузи/результатів, вибір персонажа/налаштування); прогрес і збереження); аудіо та VFX (звуки пострілу/підбору/ураження); оптимізацію для Mobile/PC; управління проєктом (календарний план, ризики, беклог/спринти, релізні білди Android + Windows) та підсумкову документацію (архітектура Blueprint-модулів, налаштування збірок, інструкції запуску). Цільова аудиторія: casual-гравці смартфонів, поціновувачі раннерів і легких шутерів, студенти/школярі та ПК-користувачі, яких мотивують рекорди/досягнення/лідерборди, а також аудиторія, що позитивно сприймає тематичну стилістику.

1.5. SWOT - аналіз

SWOT - аналіз	
Сильні сторони	**Слабкі сторони**
- Використання сучасного рушія Unreal Engine 5.4 - Кросплатформна підтримка (Android і Windows) - Візуальне програмування через Blueprints — швидке прототипування - Оптимізація через rendering, LOD, instancing та зменшення моделей у Mesh editor - Просте й інтуїтивне керування персонажем	- Обмежені ресурси та людський потенціал - Відсутність команди (робота індивідуально) - Мінімальний ігровий контент на старті - Відсутність повноцінної маркетингової стратегії - Обмежений час на тестування й

	доопрацювання
Можливості	**Загрози**
- Публікація в різних магазинах - Розширення контенту (нові карти, персонажі, режими) - Формування активної спільноти гравців - Використання гри як навчального або портфоліо-проєкту - Можливість комерціалізації через рекламу або внутрішні покупки	- Висока конкуренція серед runner-ігор - Технічні обмеження мобільних пристроїв - Можливі зміни політик публікаційних платформ - Короткий життєвий цикл жанру - Ризик зниження інтересу користувачів без оновлень

SWOT-аналіз проєкту дозволив комплексно оцінити його поточний стан, визначити внутрішні сильні та слабкі сторони, а також зовнішні можливості та загрози, які впливають на успіх розробки.

До сильних сторін належить використання сучасного рушія Unreal Engine 5.4, який забезпечує високу графічну якість, реалістичне освітлення та гнучкі інструменти розробки. Завдяки системі Blueprints реалізовано швидке прототипування і спрощене внесення змін без програмування мовою C++. Також перевагою є кросплатформність (Android і Windows), ефективна оптимізація через rendering, LOD, instancing та зменшення моделей у Mesh editor , а також інтуїтивне керування персонажем, що робить гру доступною широкій аудиторії.

Серед слабких сторін виділено обмеженість ресурсів - проєкт розроблявся індивідуально без повної команди, що обмежило обсяг контенту, художніх матеріалів і тестування. Відсутність маркетингової стратегії та просування також ускладнює позиціонування гри на ринку.

Можливості розвитку включають розширення гри за рахунок нових рівнів, персонажів і механік, публікацію на платформах Google Play, Amazon Store та Steam, а також створення активної спільноти користувачів. Проєкт може бути представлений як демонстраційний навчальний кейс або частина портфоліо розробника. Надалі можлива реалізація системи монетизації через внутрішні покупки або рекламу.

До загроз належать висока конкуренція серед мобільних runner-ігор, технічні обмеження мобільних пристроїв, а також ризики, пов’язані зі змінами політик публікаційних сервісів. Крім того, жанр раннерів має відносно короткий життєвий цикл, що потребує періодичних оновлень і розширення контенту для утримання гравців.

У цілому SWOT-аналіз показує, що гра Runner-Shooter має чітко визначені переваги й перспективи розвитку. Сильні сторони та можливості переважають існуючі ризики, а продумана стратегія поступового вдосконалення, технічних оновлень і розширення аудиторії дозволить забезпечити її подальше зростання та стабільне функціонування після релізу.

Висновки до розділу 1

У першому розділі було проведено теоретичний аналіз управління проєктами у сфері розробки ігор, досліджено специфіку геймдеву як особливого напряму IT-індустрії, визначено основні підходи до планування, організації та контролю виконання проєктів, а також обґрунтовано вибір рушія Unreal Engine 5 як технологічної бази для реалізації обраної теми. На основі вивчення літературних джерел, стандартів і методологій управління (PMBOK, Agile, Scrum, Kanban) встановлено, що розробка ігрових продуктів потребує поєднання класичних принципів проектного менеджменту з гнучкими підходами, орієнтованими на ітераційність, швидкий зворотний зв'язок і прототипування. Для ігрових студій ключовими є швидкість прийняття рішень, креативність команди, системне тестування й адаптація до змін, тому застосування Agile-практик є найбільш ефективним у порівнянні з каскадною моделлю.

Визначено, що життєвий цикл ігрового проєкту відрізняється від традиційних IT-проєктів через більшу частку творчих і дослідницьких завдань, високу частоту змін у процесі розробки та необхідність балансування між технічними, художніми та бізнес-аспектами. Типовими фазами геймдев-проєкту є: концепція (idea stage), передпродакшн, продакшн, альфа/бета-тестування та реліз. На кожному з цих етапів важливо підтримувати комунікацію між відділами дизайну, програмування, 3D-моделювання, звуку й тестування, що потребує ефективного управління командою та чіткого плану виконання робіт. Для невеликих інді-проєктів, подібних до створюваної гри жанру runner-shooter, найбільш доцільним є комбінований підхід - гнучке планування із короткими спринтами, регулярними перевірками результатів і постійною інтеграцією змін у фінальний продукт.

Проведено аналіз сучасних рушіїв для розробки ігор - Unity, Unreal Engine, Godot, CryEngine - із порівнянням їхніх функціональних можливостей, систем візуального програмування, оптимізації, рендерингу та підтримки мобільних платформ. Визначено, що Unreal Engine 5.4 є оптимальним вибором

для створення кросплатформної 3D-гри типу runner-shooter завдяки підтримці рушія Blueprints, високій якості графіки, системам візуальних ефектів (Niagara), гнучким інструментам роботи з UI (UMG), а також інтегрованим засобам профілювання й оптимізації. Важливою перевагою рушія є наявність інструментів для створення мобільних збірок під Android і Windows з єдиної кодової бази, що значно скорочує витрати часу на розробку та тестування.

У результаті аналізу ринку ігрових продуктів і жанрових аналогів (Temple Run 2, Subway Surfers, Minion Rush, Into the Dead) визначено, що основними факторами успіху ігор типу runner є простота керування, висока динаміка, короткі ігрові сесії, візуальна привабливість і система поступового прогресу з бонусами, досягненнями та внутрішньою економікою. Це дало змогу сформувати концепцію майбутнього проєкту як поєднання класичного раннера з базовими елементами шутера, де гравець може не лише ухилятися від перешкод, але й атакувати ворогів або руйнувати об’єкти, що підвищує динаміку геймплею та утримує увагу користувача.

Окрім того, у розділі проаналізовано технічні вимоги до майбутнього продукту: процедурна генерація траси, система колізій, механізм підсилень і збору монет, адаптивний інтерфейс та підтримка сенсорного і клавіатурного керування. Встановлено основні критерії успіху проєкту - стабільність роботи на мобільних пристроях середнього рівня, зручність управління, швидкий час завантаження, інтегрована система прогресу й можливість подальшого розширення контенту.

Таким чином, у результаті дослідження було визначено теоретичні засади управління проєктами у сфері розробки ігор, обґрунтовано вибір рушія Unreal Engine 5 як базової технології, сформовано концептуальну модель гри та визначено ключові завдання для її реалізації: розробку системи персонажа, генерації простору, бойової механіки, інтерфейсу та системи збережень.

РОЗДІЛ 2
ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Аналіз предметної області

Сучасний ринок мобільних і ПК-ігор насичений швидкими аркадами з короткими сесіями, де цінуються просте керування, виразний фідбек і постійна мотивація до «ще однієї спроби». Класичний runner дає зрозумілу основу (lane-based рух, наростання швидкості, мікроцілі), а додавання “кидання” підсилює емоційність, але вимагає продуманого балансу керування: на мобільних - мінімум жестів і щедрі хітбокси. Для впізнаваності й утримання потрібні: варіативність траси (процедурні секції), контроль складності, підсилення з відчутним впливом, легка мета-гра (рекорди, досягнення), стабільний FPS і швидкий повтор. Звідси випливають вимоги до гри та вісі, які в наступному розділі будуть детально розкриті як основні механіки та особливості геймплею:

1. **Рух і керування**: lane-based автобіг; стрибок/ковзання; профілі вводу (жести на мобільних; WASD/стрілки/пробіл/клік на ПК).
2. **Простір і генерація траси**: модульні секції, шаблони перешкод, правила стикування; параметри довжини/частоти для варіативності.
3. **Перешкоди**: класи загроз (статичні), патерни появи.
4. **Стрільба**: тип снаряду, візуальний/звуковий фідбек.
5. **Підсилення (power-ups)**: магніт.
6. **Прогрес і економіка сесії**: очки, дистанція, монети/валюта, досягнення; формули підрахунку та антивигорання.
7. **Динаміка складності**: щільність перешкод, частота бустерів; адаптація під навички гравця.
8. **UI/UX**: HUD (очки), підказки, меню паузи, екрани вибору персонажа; принцип «одна дія - один жест/клавіша».

9. **Аудіо та VFX**: сигнали подій (влучання, підбір), матеріальні анімації з обмеженнями для мобільних.
10. **Технічні засади продуктивності**: rendering, LOD, instancing та зменшення моделей у Mesh editor, контроль draw calls і пам’яті.

2.1.1. Визначення основних механік та особливостей геймплею.

Геймплей - це кросплатформний runner-shooter з короткими динамічними сесіями. Гравець біжить уперед по трасі з трьома смугами, уникає перешкод стрибком або ковзанням, за потреби перемикає смугу та кидає по цілях. Основна петля проста: рух, поява загрози, реакція (маневр або кидок), винагорода у вигляді очок і монет, зростання темпу та нова спроба побити рекорд.

Траса збирається з заздалегідь підготовлених секцій, щоб кожне проходження відрізнялося. На шляху трапляються статичні перешкоди, яких можна знищувати кидками. Стрільба на мобільних здійснюється простим тапом, на ПК. Для підтримки темпу використовуються підсилення: магніт для автоматичного збирання монет. Вони з’являються періодично та дають короткочасну перевагу.

Очки нараховуються за зібрані бали; після завершення спроби показується результат і рекорд, який зберігається локально. Складність зростає поступово: біг стає швидшим, перешкод і ворогів на трасі більше, але трапляються «вікна безпеки», щоб не перевантажувати гравця. Інтерфейс мінімальний: під час гри видно очки, дистанцію, монети та активні ефекти; є меню паузи, екран результатів і вибір персонажа. Звукові ефекти й легкі візуальні ефекти підкреслюють стрибки, постріли, підбір монет та активацію бустерів, не навантажуючи пристрій.

Проєкт працює на мобільних пристроях і на ПК з єдиним ігровим ядром. Різниця лише у способах керування та пресетах якості: на телефонах - жести і спрощені ефекти, на ПК - клавіатура/миша та вища деталізація. Такий набір механік дає зрозумілий, швидкий і повторюваний геймплей, який легко масштабувати новими секціями траси, видами першкод і підсиленнями.

2.2. Архітектура гри

Для побудови архітектури було сформовано ігрову діаграму модулів із розмежуванням відповідальностей та залежностей між підсистемами. Архітектура відображає логіку runner-shooter і спирається на типові ролі Unreal Engine: GameMode, Pawn/Character, PlayerController, HUD/UMG, Actor Components. Кожний модуль виконує чітко окреслену функцію та взаємодіє з іншими через публічні події/інтерфейси Blueprint. Узагальнену структуру показано на діаграмі (рис. 2.1) створено автором.

Основні модулі архітектури:

- **Control** - керування сесією гри: старт/пауза/рестарт, підрахунок очок, правила прогресії. Реалізація: BP_RunnerGameMode + PlayerController (обробка команд).
- **Movement** - рух персонажа: автобіг, перемикання смуг, стрибок, ковзання; колізії з перешкодами. Реалізація: CharacterBPs (Blueprints) з компонентами руху.
- **Combat** - бойова логіка: кидки, ураження цілей, фідбек попадань. Реалізація: BP_Projectile, хітбокси перешкод.
- **Scene Management** - побудова та прибирання траси, керування рівнями. Реалізація: BP_Floor (секції), BP_ObstacleDestroyer, карти MainMenu/CS_Map/Runner_Map/Loading.
- **Resources** - підсилення та колекційні об’єкти. Реалізація: BP_Coin, BP_PowerUp_Magnet, параметри дії/тривалості.

- **Stats** - очки, активні ефекти з таймерами. Реалізація: змінні GameMode/Character + оновлення HUD.
- **Core** - спільні сервіси: таймери, конфігурації пресетів якості (Mobile/PC).
- **UI (HUD/UMG)** - екранні елементи: очки/монети, індикатори ефектів, меню паузи/результатів, вибір персонажа. Реалізація: віджети UMG, папка HUD.
- **Saving** - збереження прогресу й налаштувань. Реалізація: Runner_SaveGame.
- **Inputs** - абстракція вводу. Реалізація: Enhanced Input у папці Inputs (жести для мобільних; WASD/миша для ПК).

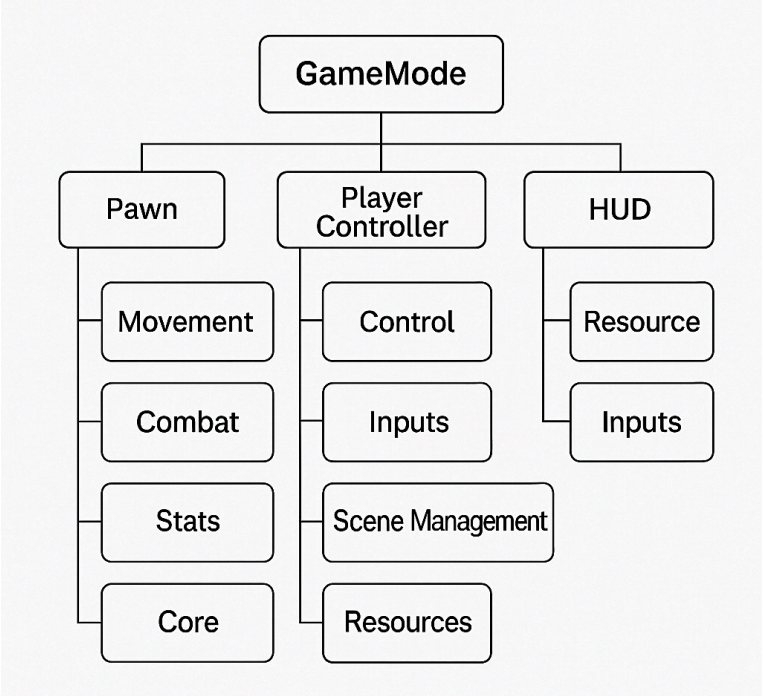

Рис. 2.1. Діаграма архітектури гри

Джерело: [створено автором]

Для правильної організації проєкту створено папки Blueprints із логічним групуванням скриптів/класів: CharacterBPs (логіка руху й станів героя), HUD (інтерфейс і віджети), Inputs (карти дій і контекстів), Obstacles (перешкоди та їх патерни), а також окремі Blueprint-класи BP_Floor, BP_Projectile, BP_PowerUp_Magnet, BP_ObstacleDestroyer, BP_RunnerGameMode,

Runner_SaveGame. Така структура спрощує навігацію, тестування та повторне використання компонентів. Приклад організації показано на рис. 2.2 створено автором у UE [12].

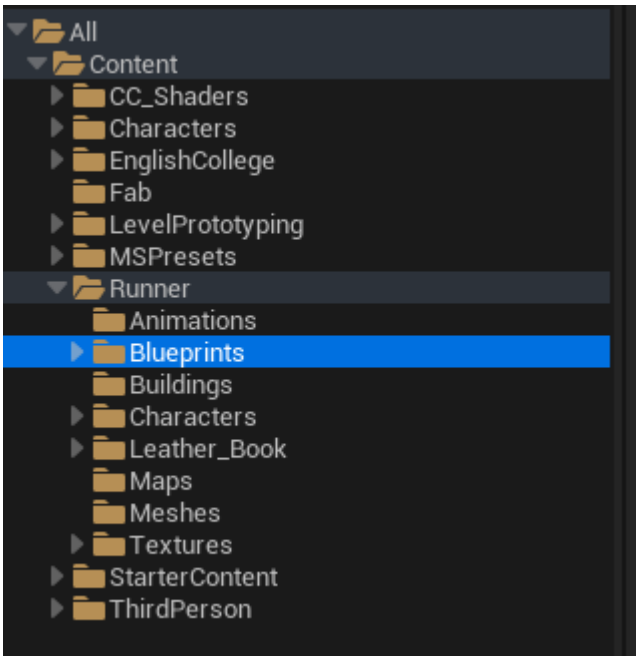

Рис. 2.2. Приклад організації папок.

Джерело: [створено автором]

Окремо впорядковано ігрові активи: папка Runner містить підкаталоги Animations, Characters (Kevin, YBot), Buildings, Meshes, Textures (у т.ч. Icon_Rank, Icon_Trophy для HUD), Maps (MainMenu, CS_Map, Runner_Map, Loading_). Розділення ресурсів за призначенням дає змогу підтримувати порядок і виконувати таргетовану оптимізацію (LOD, інстансинг, мобільні матеріали). Приклад структури активів наведено на рис. 2.3 створено автором.

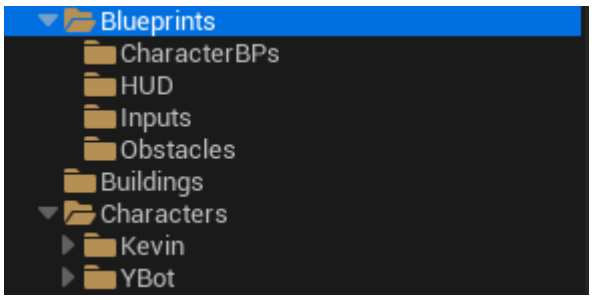

Рис. 2.3. Організація підпапок

Джерело: [створено автором]

Таким чином, архітектура гри поєднує чітку модульність (Control/Movement/Combat/Resources/Stats/Core/UI/Saving/Inputs/Scene Management) із практиками UE-проєктування (GameMode, Character, HUD/UMG, окремі Blueprint-класи для ключових об’єктів). Це забезпечує прозору взаємодію підсистем, спрощує профілювання продуктивності на мобільних і ПК та створює основу для масштабування контенту (нові секції траси, вороги, підсилення) без зміни ядра [12].

2.3. Розробка інтерфейсу користувача та системи меню

У межах проєкту я реалізував повний інтерфейс користувача та систему меню на основі UMG і Blueprint. Створено екрани MainMenu, Character Selection (CS_Map), Settings, Pause, Results і HUD для ігрової сцени Runner_Map. Головне меню забезпечує швидкий старт (Play), перехід до вибору персонажа (Character), налаштувань (Settings) та вихід (на ПК). На CS_Map реалізовано прев’ю й вибір героя (Kevin / YBot) з передаванням вибору в ігровий рівень та збереженням у Runner_SaveGame. У Settings додано регулювання гучності (Master/SFX/Music), чутливості введення (Touch/Mouse), перемикання пресетів якості для Mobile/PC та базові параметри доступності; зміни застосовуються без перезапуску рівня. Pause зупиняє гру, пропонує Resume/Settings/Restart/Main Menu. Results показує поточний результат, монети, кількість уражених цілей, а також відзначає досягнення іконками Icon_Trophy/Icon_Rank; передбачені кнопки Restart і Main Menu. Під час гри HUD відображає очки, монети та індикатори активних підсилень (магніт) із таймерами відліку.

Навігацію між екранами я організував через окремі Widget Blueprint та менеджер UI (Blueprint), що керує показом/приховуванням віджетів і станами. Дані HUD оновлюються подіями з BP_RunnerGameMode та персонажа. На мобільних реалізовано великі зони торкання (≥48 px), DPI-масштабування та

Safe Zone; на ПК додано гарячі клавіші. Ввід уніфіковано через Enhanced Input: жести (свайпи/тап) автоматично мапляться на дії, а на ПК використовується WASD/стрілки/Space/Mouse; Esc/Back відкриває паузу. Для продуктивності статичні елементи кешуються, анімації короткі, постпроцес у меню вимкнено на мобільних. У Runner_SaveGame зберігаються рекорд, сума монет, вибраний персонаж і налаштування користувача, які коректно відновлюються після перезапуску як на Android, так і у Windows-збірці. Структура ресурсів впорядкована: Runner/Blueprints/HUD, Runner/Blueprints/Inputs, Runner/Textures/UI (зокрема Icon_Rank, Icon_Trophy) та Runner/Maps (MainMenu, CS_Map, Loading, Runner_Map). Проведені перевірки підтвердили: масштабування HUD коректне на різних роздільностях, переходи між екранами відбуваються ≤0,5 с, а сценарії Play → Pause/Settings → Resume/Restart → Results → Main Menu працюють стабільно на мобільних і ПК.

Висновки до розділу 2

У другому розділі було розроблено й сформовано повну технічну основу майбутньої гри, що визначає її архітектуру, структуру контенту та взаємозв’язки між ключовими компонентами. На основі аналізу предметної області жанру runner-shooter сформульовано вимоги до механіки руху, бойової системи, складності рівнів, кросплатформності, оптимізації продуктивності та користувацького досвіду (UX). Враховано особливості обох платформ - мобільних пристроїв і персональних комп’ютерів, що дозволило забезпечити гнучкість системи керування, адаптацію інтерфейсу до різних роздільностей і баланс навантаження між графічними та логічними процесами.

У межах розділу спроєктовано модульну архітектуру гри в Unreal Engine 5.4, що включає підсистеми Control, Movement, Combat, Scene Management, Resources, Stats, Core, UI, Saving та Inputs. Кожна з них реалізована за допомогою конкретних Blueprint-класів, які забезпечують функціональну ізоляцію й гнучке розширення проєкту. Зокрема, BP_RunnerGameMode відповідає за основні стани гри (Ready, Playing, Paused, GameOver), CharacterBPs реалізують рух і поведінку персонажів, BP_Floor та BP_ObstacleDestroyer формують безперервну генерацію траси, а BP_Projectile і BP_PowerUp_Magnet забезпечують логіку атак і бонусів. Збереження прогресу реалізовано через Runner_SaveGame, а інтерфейс користувача - у системі HUD/UMG із підтримкою подій через Event Dispatchers.

Структура контенту гри впорядкована за принципом чіткого розділення за типами ресурсів. Основні директорії (Runner/Animations, Blueprints, Characters, Maps, Meshes, Textures) побудовані таким чином, щоб забезпечити швидку навігацію, можливість повторного використання елементів і спрощення командної роботи. У розділі також детально описано процес створення повного користувацького інтерфейсу та меню: MainMenu, CS_Map, Settings, Pause, Results і HUD. Ці елементи повністю інтегровані в ігровий цикл і мають адаптивну навігацію, оптимізовану для мобільного керування (через торкання) і ПК-контролів (через клавіатуру або мишу).

Важливою складовою є система керування, створена на базі Enhanced Input, яка об’єднує різні схеми управління в єдину конфігурацію. Це дозволяє гравцям комфортно взаємодіяти з грою на будь-якому пристрої без необхідності дублювати логіку вводу. Крім того, визначено й реалізовано правила генерації траси, розміщення перешкод і підсилень, що створює динамічний і повторюваний, але не монотонний ігровий досвід.

Також у межах архітектури передбачено механізми оптимізації продуктивності - використання object pooling, рівнів деталізації (LOD), instancing для повторюваних об’єктів, а також контроль draw calls і DPI scaling для стабільної роботи інтерфейсу на різних екранах. Завдяки цьому гра демонструє стабільну частоту кадрів і ефективне використання ресурсів навіть на пристроях середнього рівня.

Розроблена структура є масштабованою, логічно впорядкованою та легко підтримуваною, що спрощує додавання нового контенту, розширення геймплею й майбутню інтеграцію додаткових систем - таких як локалізація, досягнення або багатокористувацький режим. Отримані результати підтверджують, що архітектурна модель, побудована виключно на Blueprint-компонентах без залучення коду C++, є повністю придатною для створення повнофункціонального ігрового продукту.

Підсумовуючи, можна зазначити, що другий розділ став ключовим етапом формування технічного фундаменту проєкту. Він забезпечив практичну базу для подальшого тестування, оптимізації, балансування і підготовки до релізу гри на платформах Android та Windows. Розроблена система довела ефективність візуального підходу до програмування в Unreal Engine і підтверджує доцільність використання модульної архітектури для інді-проєктів жанру runner-shooter.

РОЗДІЛ 3
ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Засоби розробки

У проєкті використано набір інструментів, орієнтований на швидку збірку геймплею та кросплатформні збірки Android + Windows без застосування C++-коду. Базовим середовищем є Unreal Engine 5.4 з візуальним програмуванням Blueprints для всієї ігрової логіки (рух, кидків, підсилення, спавн перешкод, підрахунок очок), UMG для користувацького інтерфейсу (HUD, меню), Material Editor для легких візуальних ефектів і матеріалів, Enhanced Input для уніфікації введення (жести на мобільних та клавіатура/миша на ПК). Для побудови рівнів і керування сценою застосовано стандартні інструменти UE (Level/World Partition за потреби, Sequencer/CineCamera для переходів).

Графічні та аудіоресурси підготовлено з використанням бібліотек Fab (Unreal Marketplace) і Starter Content з подальшою мобільною оптимізацією (LOD, спрощені матеріали, компресія текстур); власні іконки й елементи UI (зокрема Icon_Rank, Icon_Trophy) опрацьовано у графічних редакторах. Для складання мобільних білдів застосовано Android SDK/NDK та вбудовані інструменти упаковки UE, для ПК-версії - стандартний Windows (Shipping/Development) build. Контроль версій і резервування виконувалися в Git (локальний репозиторій/хмарний хостинг), а планування робіт - у легкому таск-трекері (Trello/Jira/Notion) з відображенням беклогу, спринтів і контрольних точок. Така зв’язка інструментів забезпечила швидке прототипування на Blueprints, зручну роботу з інтерфейсом і ефектами, прозоре профілювання та стабільні кросплатформні збірки.

3.2. Вимоги до технічного та програмного забезпечення

Програмне забезпечення (для розробки)

1. Unreal Engine 5.4 (Blueprints, UMG, Enhanced Input).
2. Windows 10/11 64-bit (середовище розробника).
3. Android SDK/NDK + ADB (з Android Studio) - для складання та деплойменту мобільної версії; драйвери USB.
4. Git (локально/хмарно) - контроль версій і резервування.
5. Графічний/аудіоредактор за вибором.
6. Доступ до Fab (Unreal Marketplace) / Starter Content для ассетів.

Апаратні вимоги (робоча станція розробника)

Мінімальні:

- Процесор: 4-ядерний x86_64 (Intel Core i5 / AMD Ryzen 3).
- Оперативна пам’ять: 16 ГБ.
- Відеокарта: NVIDIA GTX 1060 / GTX 1650 або аналог з 4 ГБ VRAM, підтримка DX12/Vulkan.
- Накопичувач: SSD 256 ГБ+ (рекомендовано NVMe).
- ОС: Windows 10/11 64-bit.

Рекомендовані:

- Процесор: Intel Core i5-11400H / i5-12400F або Ryzen 5 5600(X) і вище (6+ ядер).
- Оперативна пам’ять: 32 ГБ (комфортна робота з UE Editor і збірками).
- Відеокарта: NVIDIA RTX 2060 / 3050 або вище з 6–8 ГБ VRAM.
- Накопичувач: NVMe SSD 512 ГБ+ (проєкт + кеш/DerivedDataCache).
- Дисплей: FullHD/1440p, бажано 120 Гц для тесту фреймрейту.

Вимоги до цільових платформ (для запуску гри)

Android (мінімум): Android 8.0+, 3 ГБ RAM, GPU з Vulkan 1.1 (рівень Adreno 618/Mali-G52), вільно 300–500 МБ.
Android (рекомендовано): Android 11+, 4–6 ГБ RAM, Adreno 640/660 або аналог.

Windows (мінімум): Windows 10/11 64-bit, CPU з 4 потоками, 8 ГБ RAM, GPU з DX12, 2–4 ГБ VRAM, 500 МБ вільного місця.
Windows (рекомендовано): 6-ядерний CPU, 16 ГБ RAM, GTX 1660/RTX 2060 або вище з 6 ГБ VRAM.

Такі вимоги гарантують стабільну роботу UE 5.4 під час розробки та комфортний FPS для релізних збірок гри на Android і Windows [19].

3.3. Опис програмної реалізації

3.3.1. Структура проєкту та організація контенту

Увесь ігровий контент я впорядкував у кореневій папці Runner, що дало чітку відповідність між логічними модулями й файловою структурою. У межах Runner/Blueprints зосереджено ігрову логіку: CharacterBPs для керування героєм, HUD для всіх UMG-віджетів, Inputs для Enhanced Input, Obstacles для перешкод і ворогів, а також окремі класи ядра - BP_RunnerGameMode, BP_Floor, BP_Projectile, BP_PowerUp_Magnet, BP_ObstacleDestroyer, Runner_SaveGame. Папка Maps містить MainMenu, CS_Map, два рівні завантаження Loading_ та основну карту Runner_Map. Візуальні ресурси розкладені по Meshes та Textures (у т.ч. Icon_Rank, Icon_Trophy для HUD), моделі персонажів - у Characters/ Kevin, YBot, декор бекґраунду - у Buildings. Така організація спростила навігацію, прискорила профілювання та ізолювала кросплатформні налаштування.

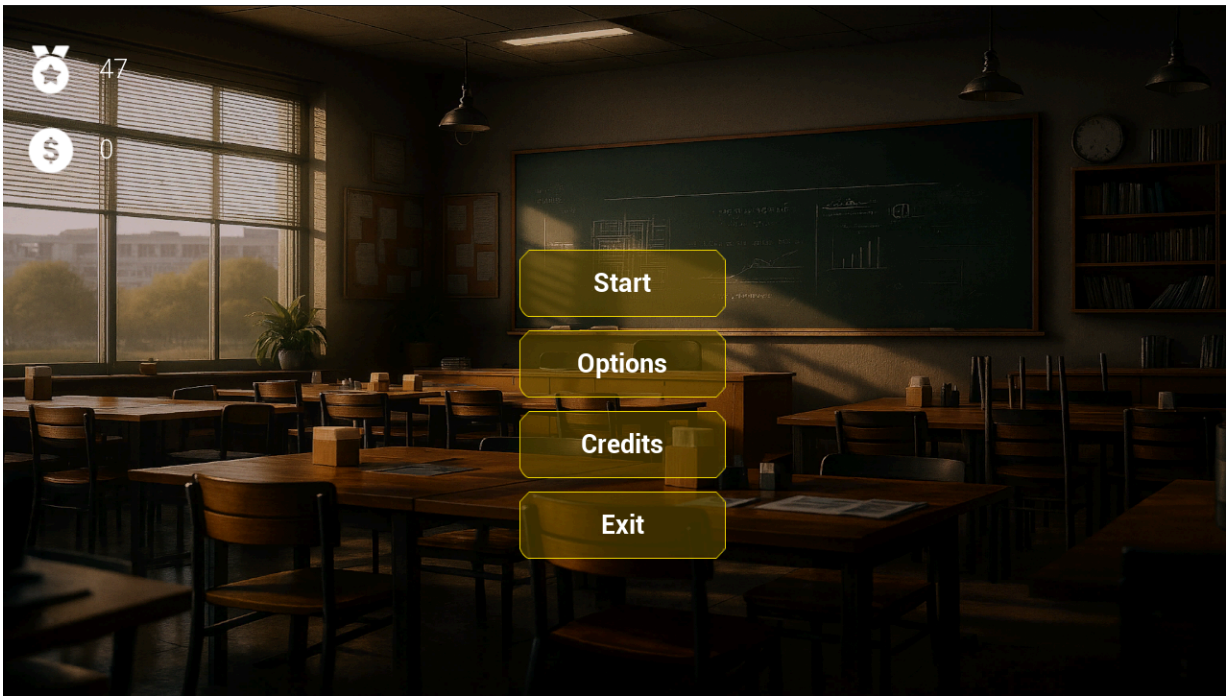

Рис. 3.1. Приклад головного меню

Джерело: [створено автором]

Ігровий цикл побудований як послідовність «MainMenu → CS_Map (вибір героя) → Loading → Runner_Map (гра) → Results → MainMenu/Restart». Рівні завантаження приховують ініціалізацію пулів об’єктів, скидання статистики та підготовку HUD. Вибір персонажа на CS_Map я зберігав у Runner_SaveGame і дублював у GameInstance, щоб уникати зайвого читання файлу під час переходів. На старті Runner_Map автоматично створюється HUD, підтягуються налаштування якості й звуку, і розпочинається сесія.

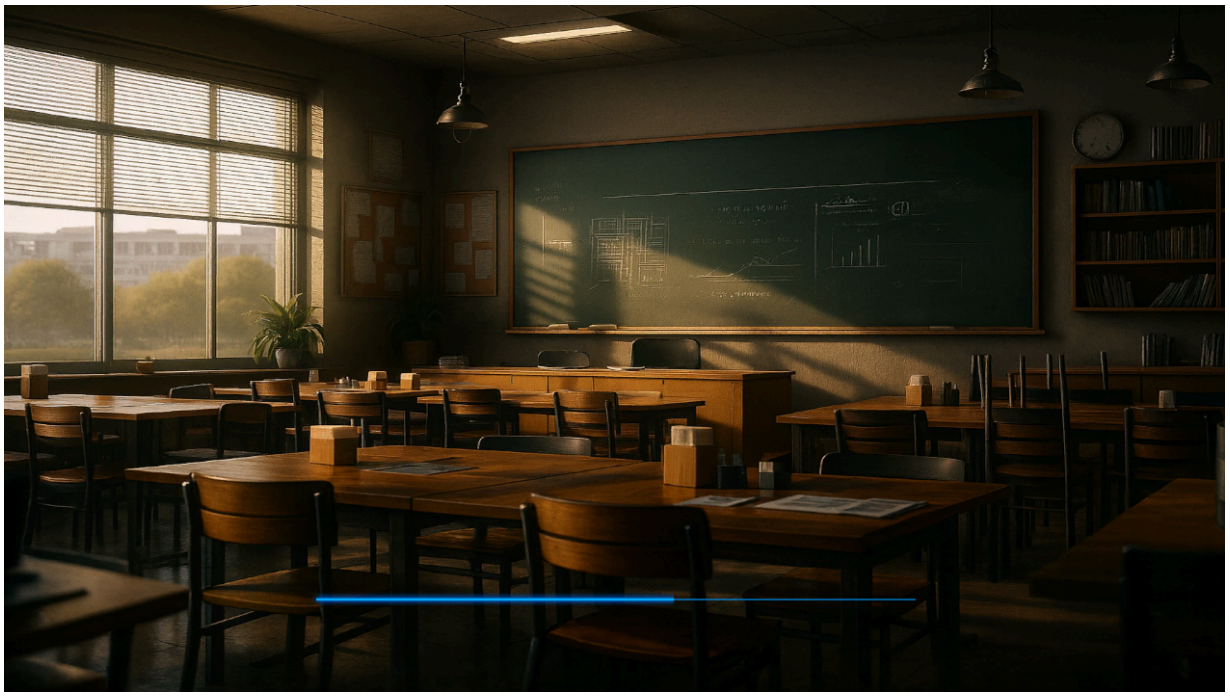

Рис. 3.2. Приклад екрана завантаження

Джерело: [створено автором]

3.3.3. Ігровий режим та службові класи.

У BP_RunnerGameMode зібрані стани «Ready/Playing/Paused/GameOver», підрахунок очок, запуск таймерів підсилень, а також криві складності, що керують зростанням швидкості та щільності загроз. PlayerController обробляє глобальні команди (пауза, перехід між екранами), а GameInstance тримає спільні параметри - обраного героя, мову та профіль якості. Взаємодія між режимом і HUD побудована на Event Dispatchers, тому інтерфейс миттєво реагує на зміну станів гри.

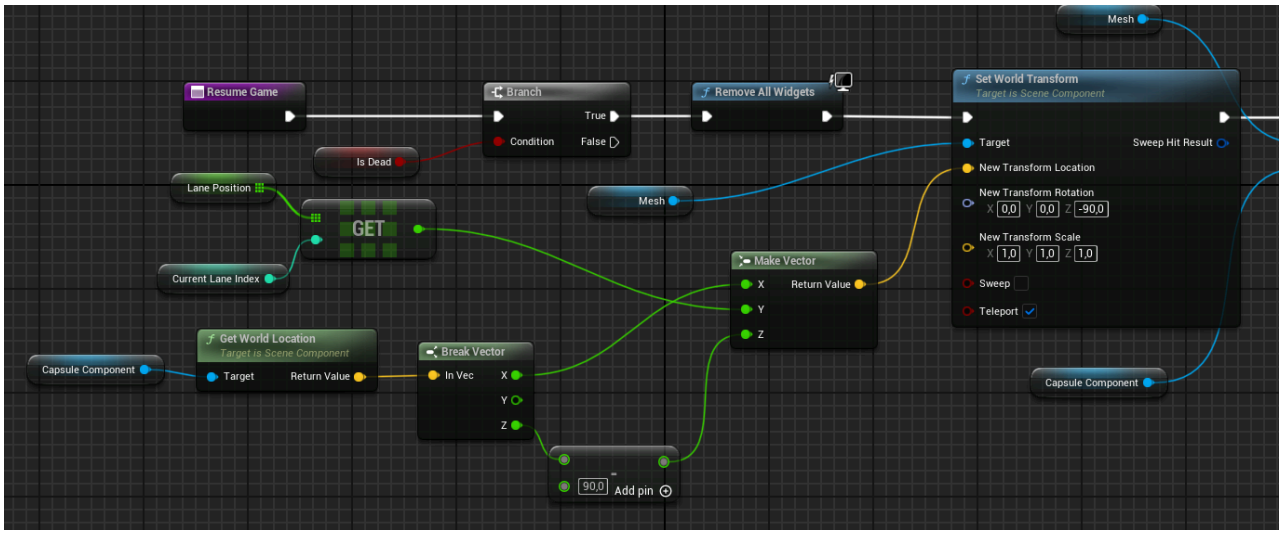

Рис. 3.3. Початок функції продовження гри

Джерело: [створено автором]

На рисунку 3.4 створюється набір точок появи бонусів. Спочатку проходять масив існуючих позицій, після чого в циклі генеруються додаткові точки для кожної з них. Для кожної точки створюється Arrow Component, який слугує маркером у редакторі. Йому задається колір та параметри позиції через Make Vector і Make Transform.

Отримані координати додаються в окремий масив Spawn Points, який надалі використовується для появи бонусів під час гри. Така логіка дозволяє швидко та автоматично формувати потрібну кількість spawn-точок без ручного розміщення.

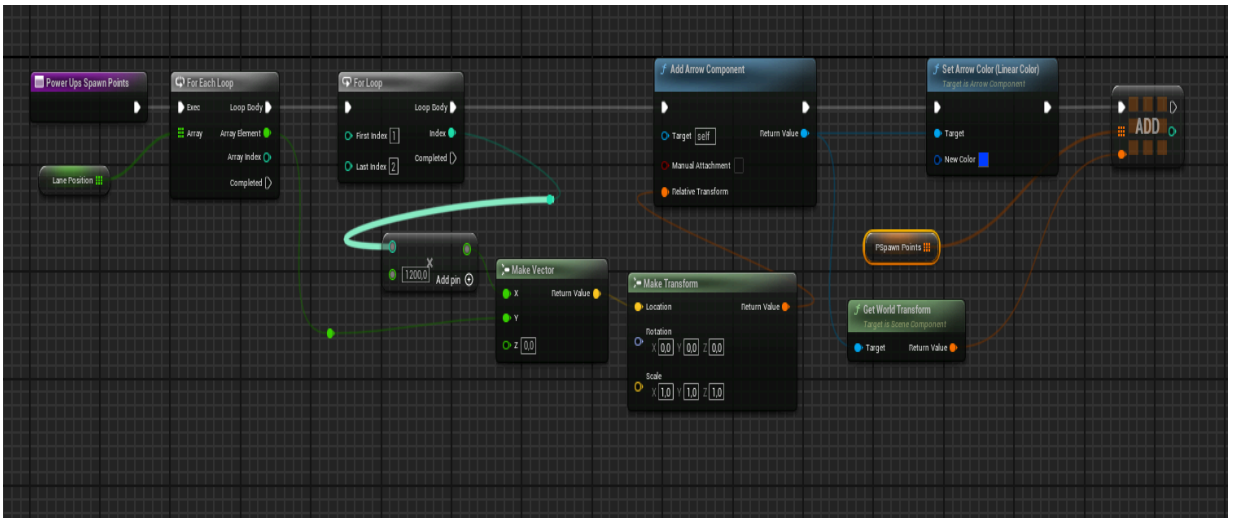

Рис. 3.4. Функція появи бонусів

Джерело: [створено автором]

3.3.4. Використання Blueprints

Використання Blueprint у Unreal Engine є важливим елементом сучасного процесу розробки завдяки їхній наочності, зручності та високій швидкості роботи. Візуальна система скриптів дозволяє розробникам створювати логіку без необхідності писати код, що значно прискорює прототипування та тестування ігрових механік. Це особливо корисно на ранніх етапах проєкту, коли потрібно швидко вносити зміни, експериментувати з поведінкою об’єктів або перевіряти різні підходи до реалізації функціоналу.

Крім того, Blueprint забезпечують просту інтеграцію з мешами, матеріалами, анімаціями та іншими компонентами рушія, що робить їх універсальним інструментом як для програмістів, так і для дизайнерів. Завдяки візуальній природі системи легше аналізувати структуру логіки, шукати помилки та підтримувати проєкт у довгостроковій перспективі. У поєднанні з можливістю оптимізації критичних частин через C++, Blueprints створюють гнучкий і ефективний робочий процес, що дозволяє швидко отримувати робочі результати та адаптувати гру під потреби проєкту.

Даний Blueprint BP_RunnerCharacter – подія “Change Lanes” відповідає за зміну смуги руху персонажа під час гри. Коли гравець натискає клавішу вліво або вправо, система визначає поточну позицію персонажа, розраховує нову координату смуги та плавно переміщує його за допомогою вузлів Lerp і Set Actor Location. Додатково перевіряється стан змінної Is Switching Lanes, щоб уникнути повторного перемикання під час руху.

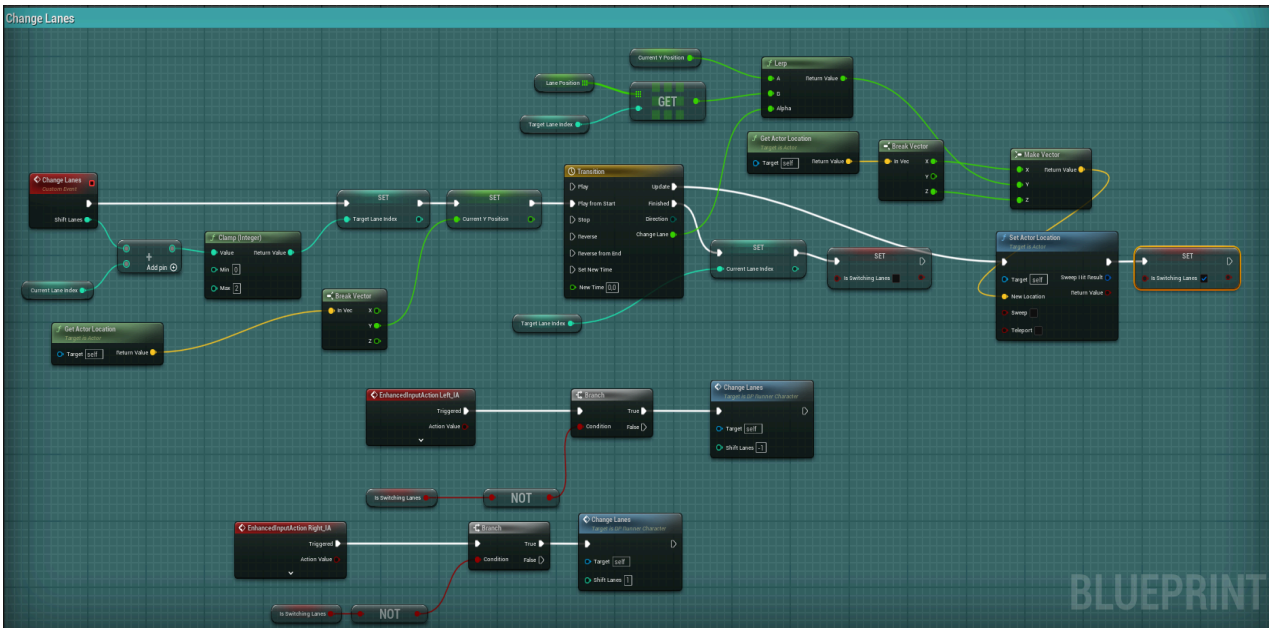

Рис. 3.5. Blueprint події “Change Lanes”
Джерело: [створено автором]

Даний Blueprint BP_RunnerCharacter – функція “Add Infinite Run” реалізує безперервний рух персонажа вперед. Подія Event Tick викликається кожен кадр, отримує поточний кут повороту за допомогою вузла Get Control Rotation, формує напрямок руху (Get Forward Vector) і додає постійне прискорення вперед через вузол Add Movement Input. Це забезпечує нескінченний біг персонажа без необхідності додаткового керування з боку гравця.

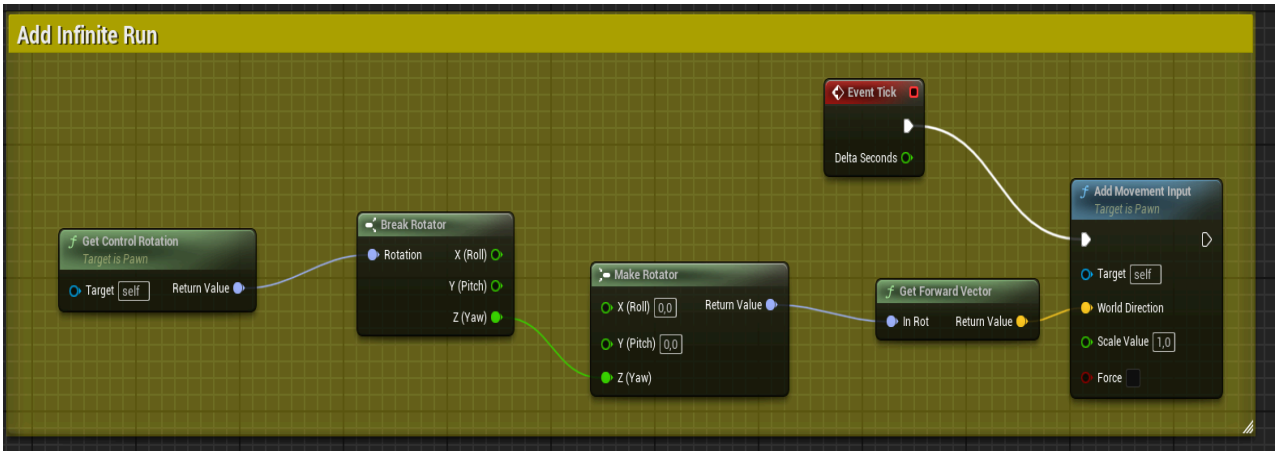

Рис. 3.6. Blueprint функції “Add Infinite Run”
Джерело: [створено автором]

Цей Blueprint BP_RunnerCharacter – подія “Basic Projectile Spawn” відповідає за створення снаряда при натисканні кнопки атаки. Подія Enhanced InputAction IA_Fire активується, коли гравець здійснює дію пострілу. За допомогою вузла Get World Transform визначається поточне положення точки спавну (Projectile Spawn Location), після чого викликається вузол SpawnActor BP_Projectile, який створює снаряд у заданій позиції з параметром «Always Spawn, Ignore Collisions». Це забезпечує точне та безперебійне створення пострілів під час гри [16].

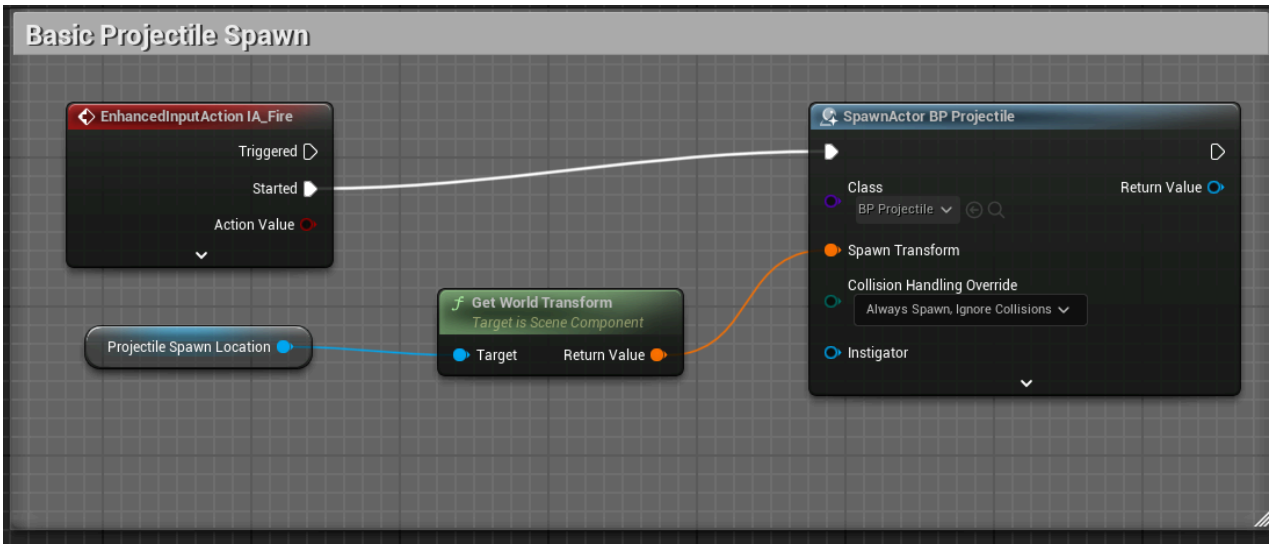

Рис. 3.7. Blueprint події “Basic Projectile Spawn”
Джерело: [створено автором]

Цей Blueprint BP_RunnerCharacter – функція “Update Coins” реалізує систему збереження та оновлення кількості монет, зібраних гравцем. Після виклику функції Update Coins відбувається звернення до поточного режиму гри через Get Game Mode і перевірка, чи існує збереження за допомогою Does Save Game Exist. Якщо збереження знайдено, дані завантажуються з файлу через Load Game from Slot і оновлюються поточним значенням монет. У випадку відсутності збереження створюється новий об’єкт через Create Save Game Object, після чого кількість монет записується у змінну CoinValue та зберігається командою Save Game to Slot.

Таким чином забезпечується стабільне збереження прогресу гравця між сеансами гри, що є важливою частиною ігрового процесу та системи прогресу [28].

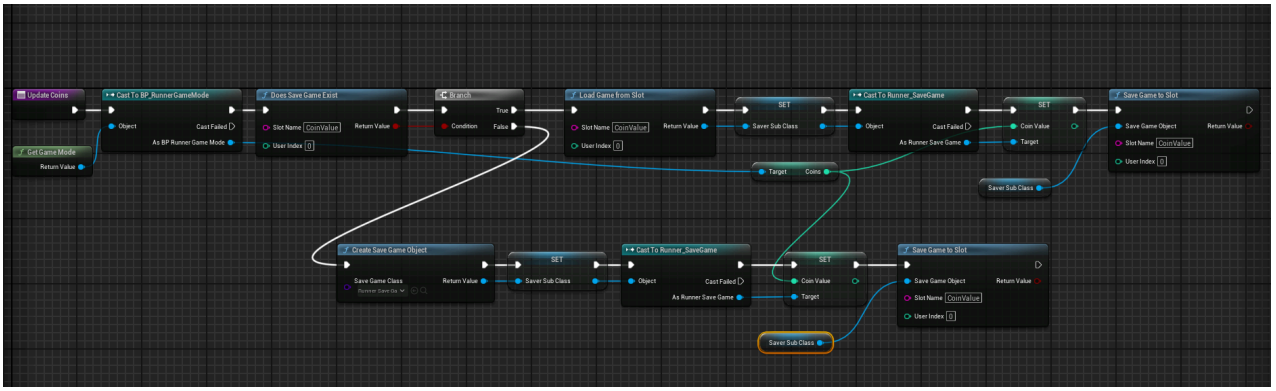

Рис. 3.8. Blueprint функції “Update Coins”
Джерело: [створено автором]

Цей Blueprint BP_RunnerCharacter – подія “Add Input Mapping” ініціалізує систему обробки введення на початку гри. Подія Event BeginPlay виконується під час запуску рівня, встановлюючи початкові значення змінних (Is Dead, Montage Index) та створює головний інтерфейс гри (Create Gameplay UI Widget), який додається на екран через вузол Add to Viewport.

Далі отримується контролер гравця (Get Controller), який перетворюється на тип Player Controller і перевіряється на валідність. Якщо все виконано успішно, за допомогою Enhanced Input Local Player Subsystem додається контекст керування (Add Mapping Context), який визначає дії гравця - рух, стрибок, стрільбу тощо.

Цей Blueprint відповідає за початкове налаштування управління та відображення інтерфейсу після запуску гри.

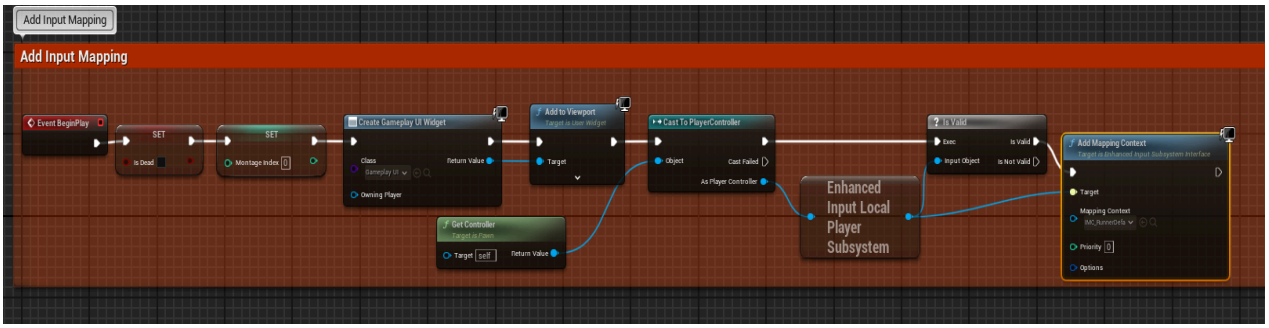

Рис. 3.9. Blueprint події “Add Input Mapping”

Джерело: [створено автором]

Цей Blueprint BP_Floor - функція “Spawn Obstacles” відповідає за випадкове створення перешкод на ігровій доріжці. Після виклику функції Spawn Obstacles система вибирає одну з доступних точок спавну (Spawn Points) за допомогою генерації випадкового числа (Random Integer in Range). Потім цикл For Loop проходить усі точки, і через вузол Switch on Int визначається тип перешкоди, що буде створено - стіна (BP Obstacles Wall) або блокер (BP Obstacles Blocker).

Вузол Add Child Actor Component додає вибраний тип перешкоди до поточного об’єкта підлоги (BP_Floor), використовуючи трансформацію позиції через Make Relative Transform. Завдяки цьому механізму кожна секція траси отримує унікальний набір перешкод, що підвищує варіативність ігрового процесу.

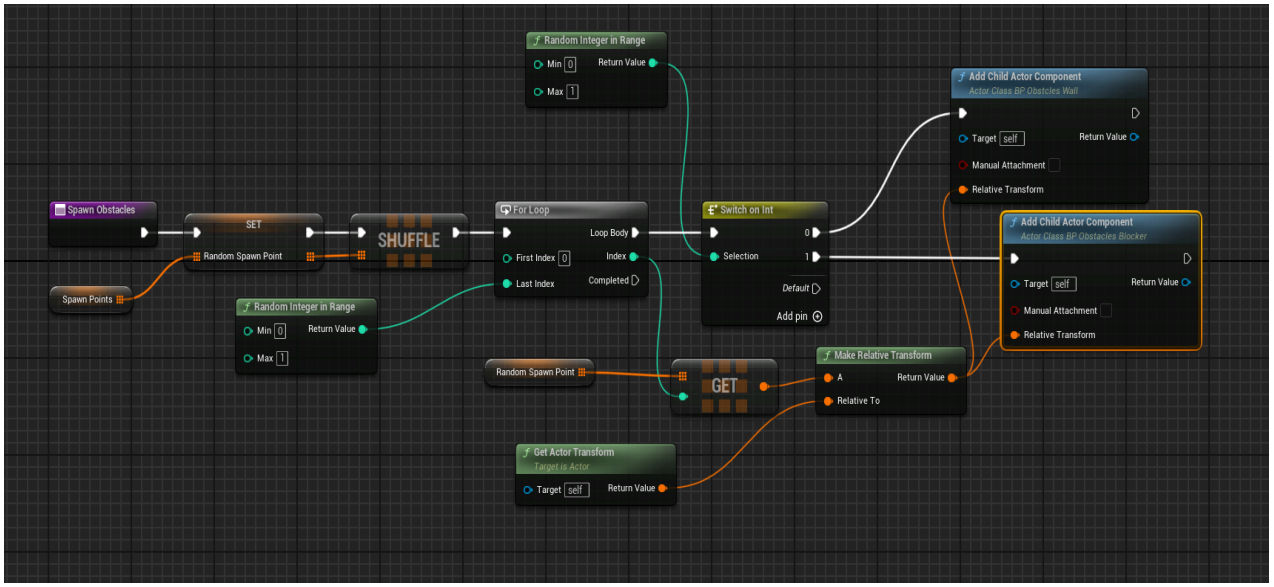

Рис. 3.10. Blueprint функції “Spawn Obstacles”

Джерело: [створено автором]

Даний Blueprint **BP_Floor - функція “Setup Spawn Points”**створює та налаштовує точки спавну для майбутнього розміщення об’єктів, таких як перешкоди чи бонуси. Під час виконання функції Setup Spawn Points відбувається проходження по всіх позиціях смуг (Lane Positions) за допомогою циклів For Each Loop та For Loop. Для кожної смуги обчислюється координата за віссю X із заданим кроком (1200 одиниць), після чого створюється вектор позиції (Make Vector) і трансформація (Make Transform).

Далі до об’єкта додається стрілка (Add Arrow Component) для позначення точки спавну на рівні, а її координати отримуються через Get World Transform і записуються в масив Spawn Points за допомогою вузла Add. Це забезпечує автоматичне створення набору точок, у яких надалі відбуватиметься генерація ігрових елементів.[1]

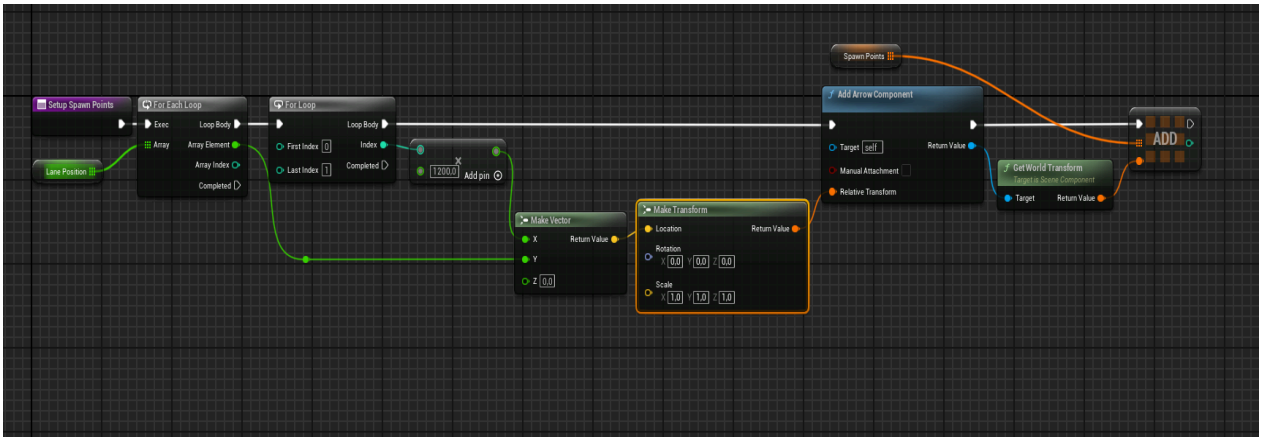

Рис. 3.11. Blueprint функції “Setup Spawn Points”

Джерело: [створено автором]

Цей Blueprint BP_Floor – функція “Power Ups Spawn Points” створює точки спавну для бонусів (підсилень) на ігровій трасі. Під час виконання функції Power Ups Spawn Points виконується цикл For Each Loop, який проходить усі доступні позиції смуг (Lane Positions). Для кожної позиції визначаються координати з кроком 1200 одиниць, формується вектор (Make Vector) і трансформація (Make Transform) для розташування об’єкта.

Далі за допомогою вузла Add Arrow Component створюється стрілка, яка позначає місце появи бонусу, а її колір змінюється через Set Arrow Color - синій колір вказує, що ця точка призначена саме для бонусних об’єктів. Отримана позиція зберігається в масиві Spawn Points для подальшого використання при генерації підсилень під час гри.

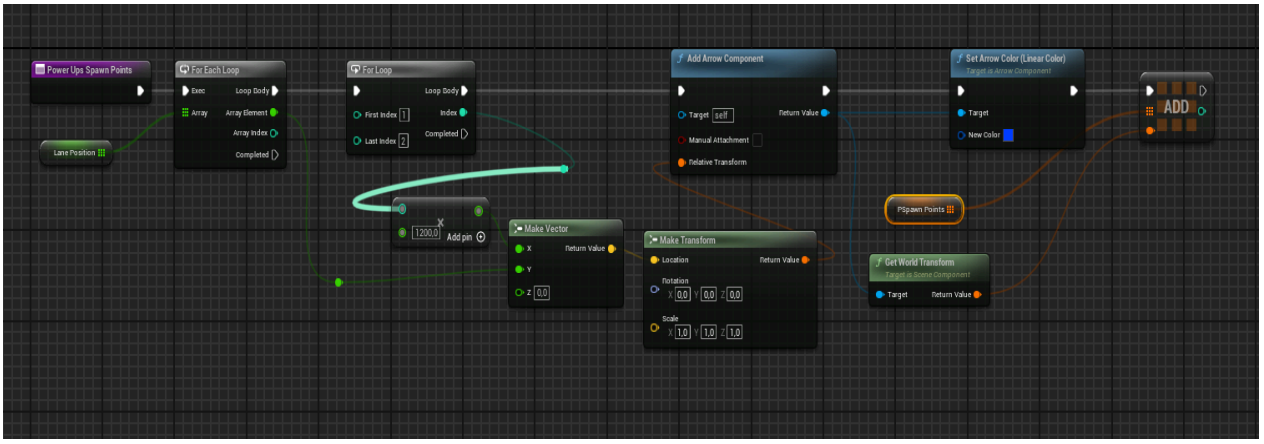

Рис. 3.12. Blueprint функції “Power Ups Spawn Points”

Джерело: [створено автором]

3.3.4. Персонаж і керування (Enhanced Input)

Героя реалізовано як Character Blueprint із lane-based логікою: три смуги руху, коротка інтерполяція між смугами через Timeline, стрибок і ковзання. Єдині дії вводу (MoveLeft/Right, Jump, Slide, Fire, Pause) я зробив у два контексти: на мобільних - жести (свайпи/тап), на ПК - клавіатура/миша. Це дало одну логіку Blueprints для обох платформ і стабільний UX. Колізії та ураження обробляються подіями OnComponentHit/BeginOverlap з урахуванням активного щита, що гарантує передбачувану поведінку в будь-яких FPS.

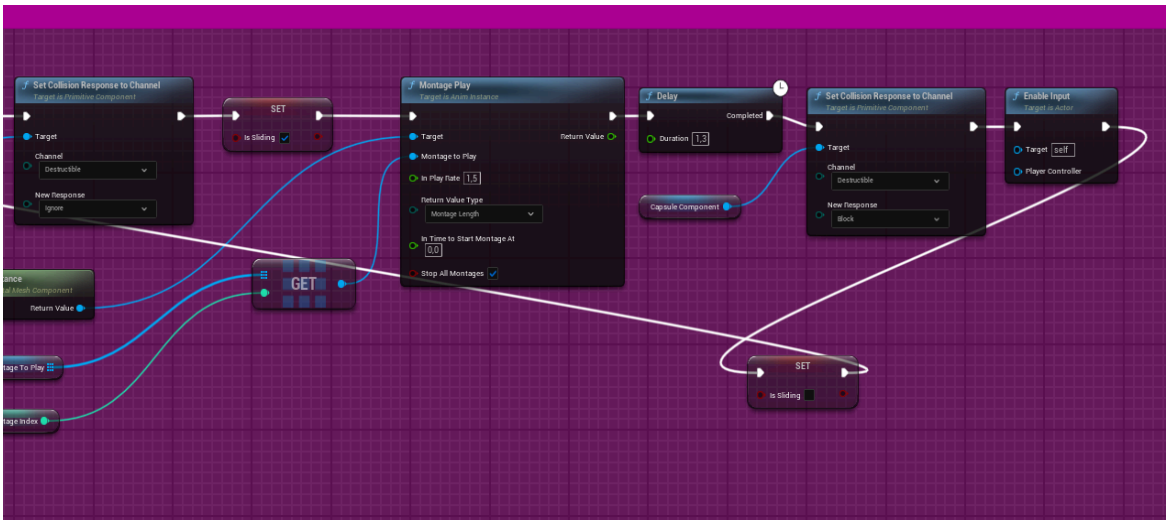

Рис. 3.13. Завершення функції слайду персонажа

Джерело: [створено автором]

3.3.5. Простір і генерація траси.

Нескінченний трек зібраний із модулів BP_Floor, кожен має маркери Entry/Exit для безшовного стикування. Попереду завжди підтримується черга з кількох секцій, а позаду відпрацьовані модулі потрапляють у BP_ObstacleDestroyer і повертаються в пул - завдяки цьому під час сесії немає дорогих Spawn/Destroy. Варіативність забезпечує набір «патернів» секцій (безпечні, «перешкодні», «ворожі», змішані та ризикові), що комбінуються залежно від кривої складності.

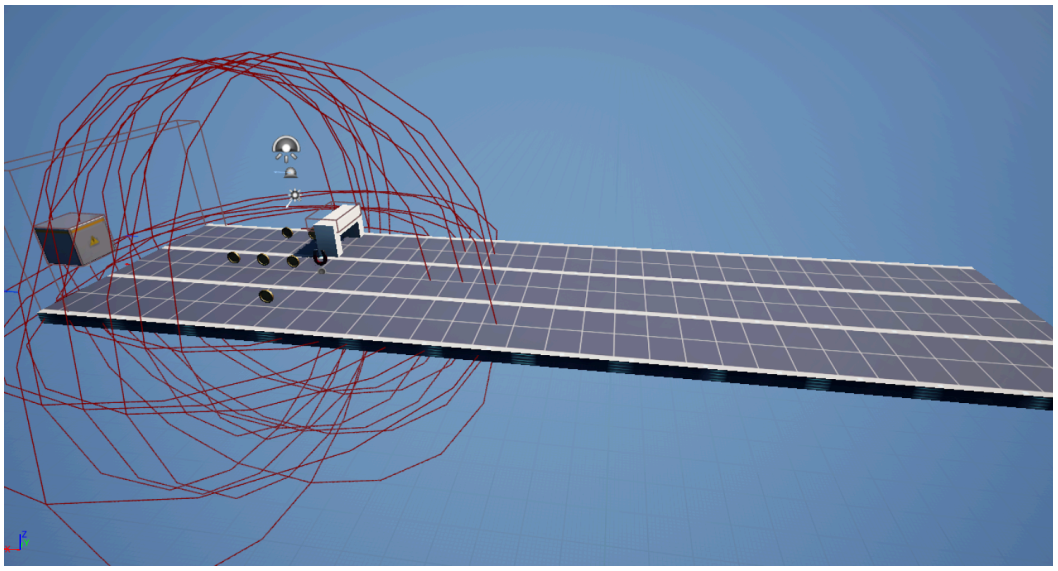

Рис. 3.14. Початковий вигляд траси

Джерело: (створено автором)

3.3.6. Перешкоди та система кидання

У грі реалізовано один тип перешкод - статичні бар'єри з простими хітбоксами, які розміщуються на дорозі та вимагають від гравця швидкої реакції для ухилення.

Рис. 3.15. вигляд однієї із перешкод

Джерело: [створено автором]

Основна взаємодія з перешкодами відбувається через систему колізій, що визначає зіткнення персонажа з об'єктами. Постріл реалізований у BP_Projectile та працює у фіксованому напрямку руху персонажа, без наведення чи корекції траєкторії. Параметри швидкості снаряда, часу його життя та кулдауну підібрані таким чином, щоб підтримувати динамічний темп гри й зберігати баланс складності. Подія зіткнення викликає сигнал у GameMode для нарахування очок та візуального фідбеку під час проходження [17].

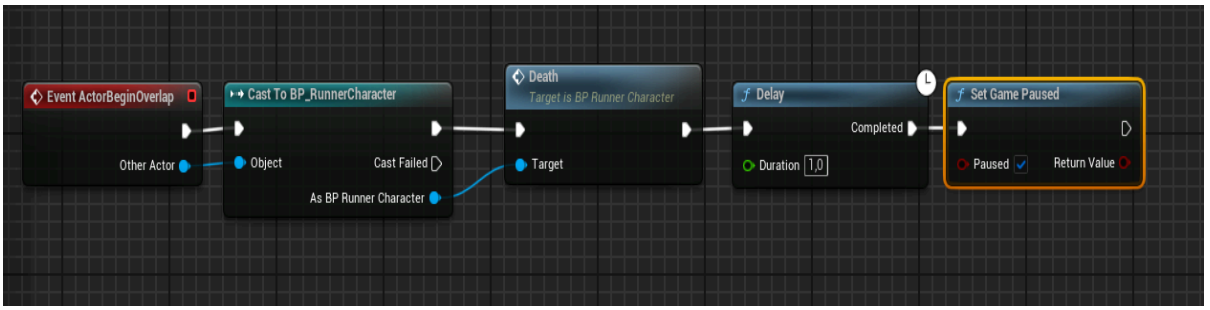

Рис. 3. 16. функція зіткнення персонажа з перешкодою

Джерело: [створено автором]

3.3.7. Підсилення, колекційні об’єкти та прогрес.

Монети (BP_Coin) мають легку анімацію обертання й дають миттєвий фідбек у HUD. Підсилення включають BP_PowerUp_Magnet із радіусом притягування монет. У Runner_SaveGame я зберігаю рекорд, сумарні монети, вибраного персонажа, гучності, чутливість і профіль якості - усе це однаково працює на Android і Windows.

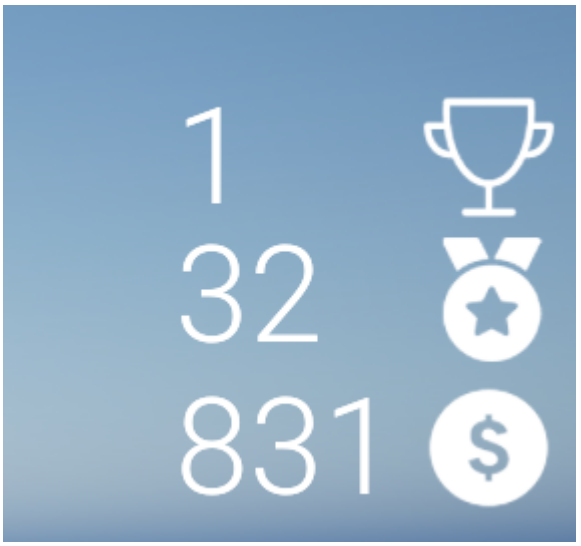

Рис. 3. 17. Інтерфейс прогресу в грі

Джерело: [створено автором]

3.3.8. Інтерфейс і навігація (UMG).

Інтерфейс створено як окремі Widget Blueprints: W_MainMenu, W_CS, W_Settings, W_Pause, W_Results і W_HUD. Менеджер UI у PlayerController показує/ховає екрани, забезпечує плавні переходи й застосовує налаштування без перезапуску. HUD показує очки, дистанцію, монети, статус активних ефектів і короткі контекстні підказки для перших ігор. Я використав anchors, DPI Curve та Safe Zone, тож UI однаково читабельний на різних екранних щільностях; на мобільних збільшені зони торкання, на ПК активні гарячі клавіші.

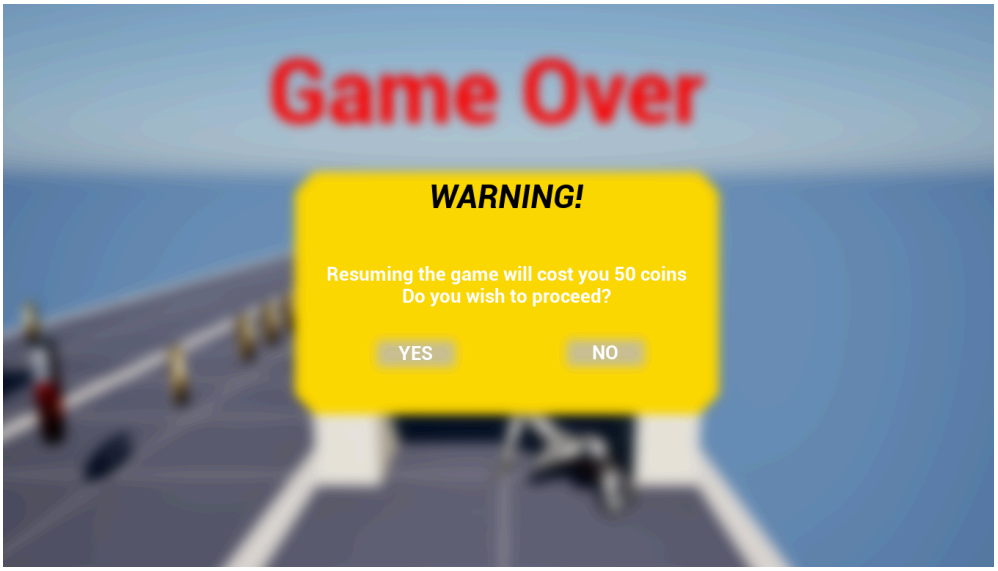

Рис. 3. 18. Інтерфейс попередження про продовження гри

Джерело: [створено автором]

На рисунку 3.11 реалізована логіка кнопок «Resume» та «Main Menu» у меню паузи. Після створення віджету встановлюється фокус і вмикається курсор миші, щоб гравець міг взаємодіяти з інтерфейсом. Кнопка Resume перевіряє, чи гра перебуває в паузі. Якщо так, пауза вимикається, режим вводу повертається у Game Only, курсор приховується, а меню видаляється з екрану. Кнопка Main Menu також прибирає віджет і виконує перехід до головного меню

через Open Level. Таким чином, ця логіка забезпечує коректне продовження гри або вихід у головне меню під час паузи.

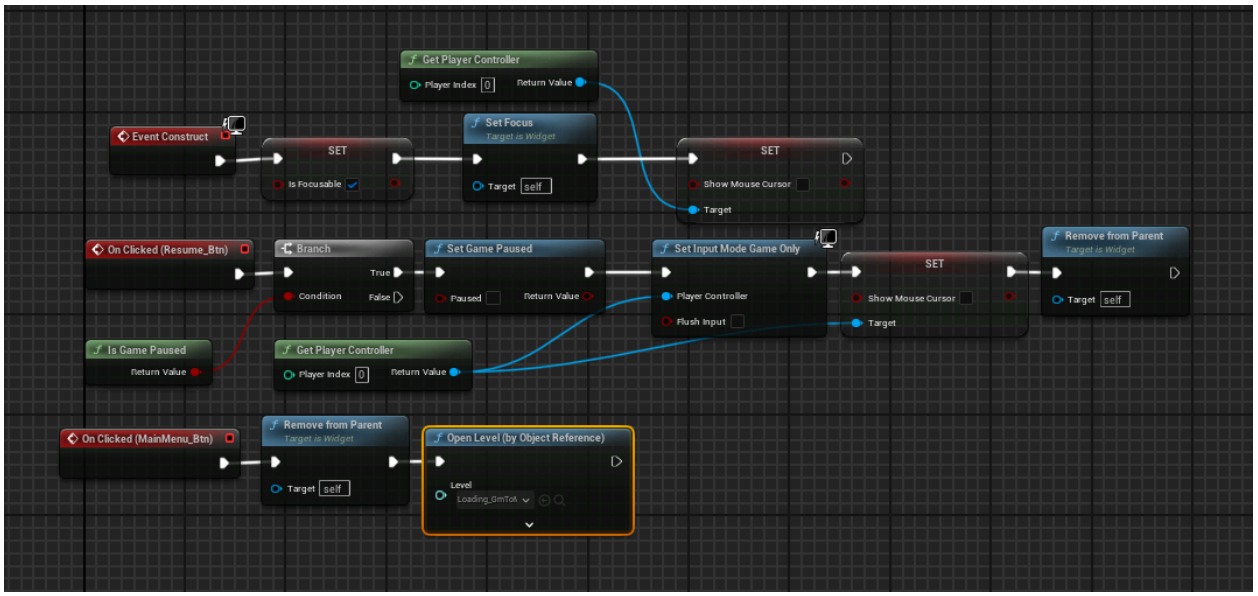

Рис. 3. 19. функція продовження гри в меню паузи

Джерело: [створено автором]

3.3.9. Моделювання середовища, колізії та фізика

Інструмент Landscape/Terrain я використав для створення простого фону навколо траси, щоб надати глибину зображенню без складних моделей. Завдяки системі LOD ландшафт не впливає на продуктивність гри. За допомогою вбудованих Modeling Tools я швидко створював прості форми перешкод і налаштовував колізії для руху персонажа по смугах. Це дозволило швидко перевіряти зміни без використання сторонніх програм для моделювання [10].

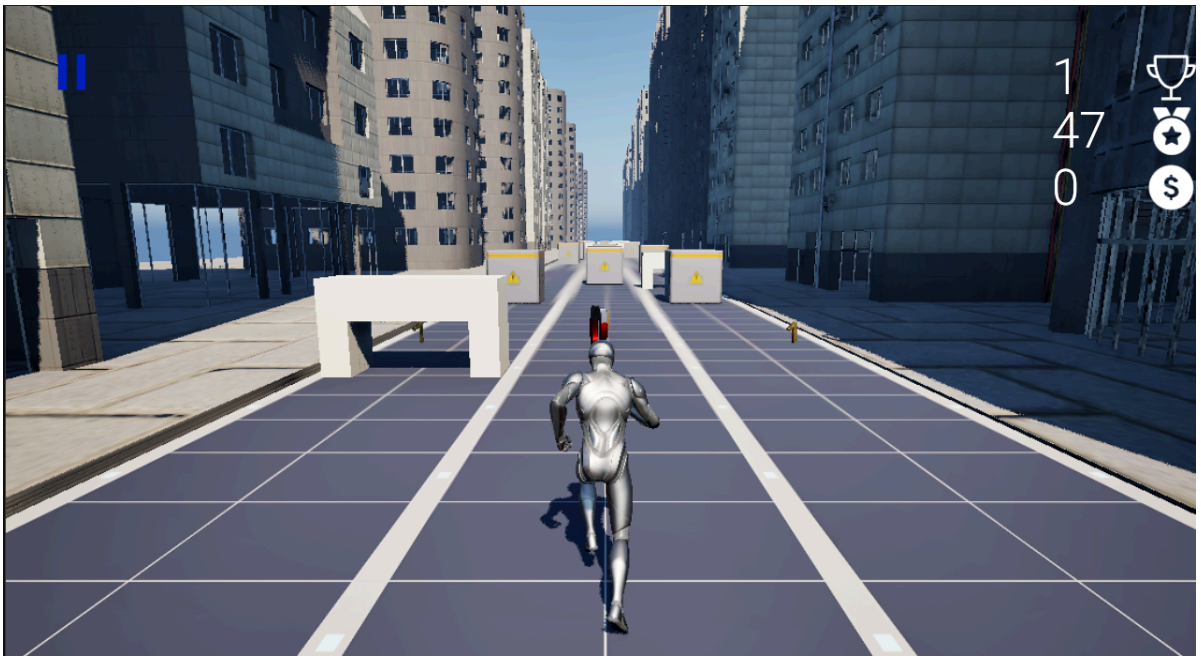

Рис. 3. 20. Приклад базової карти

Джерело: [створено автором]

Я створив окремі канали взаємодії для персонажа, перешкод, снарядів і монет, щоб кожен об’єкт реагував тільки на потрібні події. Наприклад, монети не блокують рух, а просто активують подію підбору. Фізичну симуляцію я не використовував, щоб гра працювала швидше. Усі зіткнення побудовані на простих тригерах і точних хітбоксах, що добре підходить для динамічного геймплею раннера.

3.3.10. Оптимізація та mesh instancing.

Для повторюваних невеликих об’єктів і декору я використовував Instanced Static Mesh (HISM) - це дозволяє відображати багато копій одного об’єкта з мінімальним навантаженням на систему. Завдяки цьому кількість викликів рендеру (draw calls) значно зменшилася.

Також я застосував LOD-профілі для будівель, стиснення текстур (ASTC/ETC2 для Android і BC для Windows) та обмежив кількість унікальних

матеріалів. Це допомогло підтримувати стабільну частоту кадрів - 50–60 FPS на мобільних пристроях і 60 FPS на ПК.

Для пошуку проблем із продуктивністю я користувався профайлерами Stat Unit, GPU Visualizer. Після аналізу я спрощував матеріали, зменшував кількість ефектів і замінював часті створення об’єктів на пулінг, щоб гра працювала плавніше.

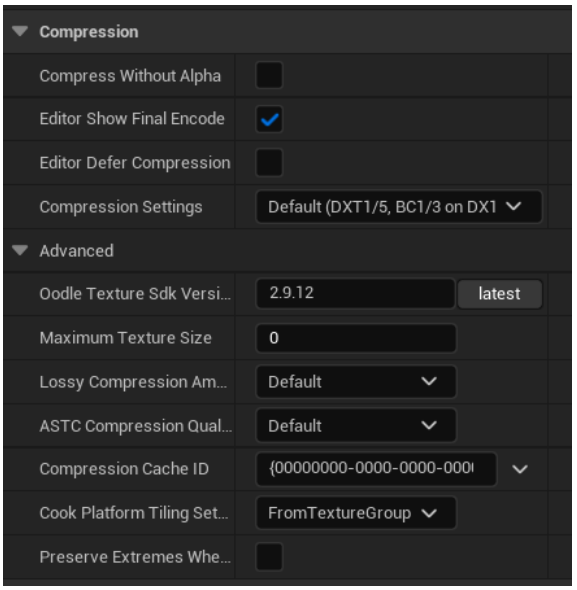

Рис. 3. 21. Приклад оптимізації текстур

Джерело: [створено автором]

На рисунку 3.14 наведено діагностичний оверлей продуктивності Unreal Engine 5, який відображає реальні показники роботи гри. Перш за все зазначається кількість кадрів за секунду (FPS). Параметри Frame Time, Game, Draw та RHIT показують час, необхідний для обробки одного кадру, логіки гри, підготовки рендера та виконання графічних команд відповідно - підвищені значення цих параметрів означають перевантаження системи. GPU Time демонструє навантаження на відеокарту, а показники Mem та vMem відображають споживання оперативної та відеопам’яті. Крім того, параметри Draws та Prims показують кількість викликів рендера та полігонів, що

впливають на продуктивність. Дані метрики допомагають зрозуміти, що саме варто оптимізувати для покращення FPS.

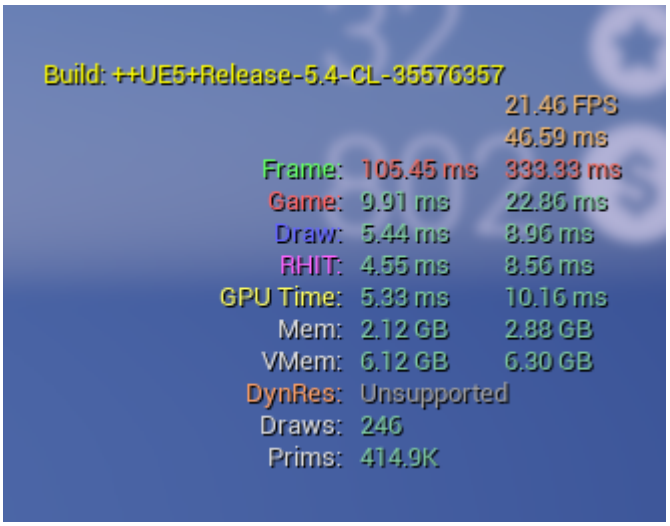

Рис. 3. 22. Приклад характеристики гри

Джерело: [створено автором]

3.3.11. Loading/Sequencer і збірка.

Короткі екрани завантаження показуються під час підготовки рівня Runner_Map, коли гра завантажує потрібні ресурси та оновлює об’єкти в пам’яті. Для вступних і фінальних сцен я використав прості кадри Sequencer без складних ефектів, щоб не знижувати продуктивність.

Гра збирається звичайним способом: для Android - у форматі arm64 з Vulkan і стисканням текстур ASTC/ETC2, а для Windows - у x64 з DX12. Обраний профіль якості (високий, середній або низький) задається в меню Settings, зберігається в Runner_SaveGame і автоматично застосовується під час запуску гри.

3.3.12. Вибір персонажа

Система вибору персонажа реалізована у Blueprint-класі BP_CS_Characters, який відповідає за збереження вибраного героя та його передачу у головний рівень гри. Після вибору персонажа через головне меню гравець активує функцію Save Character, що перевіряє наявність файлу збереження (Does Save Game Exist) у слоті PlayerCharacter. Якщо файл уже існує, дані завантажуються за допомогою вузла Load Game from Slot і оновлюються новим значенням вибраного персонажа; якщо ні - створюється новий об’єкт збереження (Create Save Game Object).

Отриманий об’єкт зберігається через Save Game to Slot, після чого гра автоматично відкриває рівень Runner_Map, уже з обраним персонажем. Для цього у класі Runner_SaveGame передбачена змінна PlayerCharacter, яка зчитується під час запуску карти, щоб визначити, який герой буде використаний у грі [15].

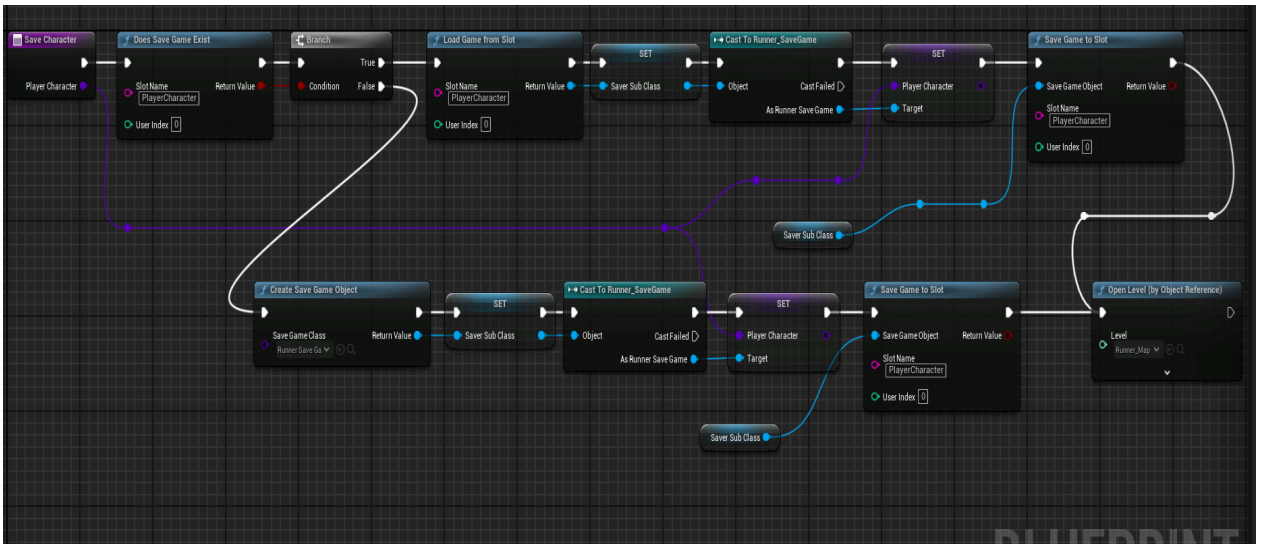

Рис. 3. 23. Функція збереження персонажа

Джерело: [створено автором]

Таким чином, вибір персонажа зберігаються локально у файлі SaveGame, що забезпечує збереження вибору між сеансами гри. Система є простою,

надійною та легко розширюваною: щоб додати нового героя, достатньо внести його у список вибору меню та передати його ім’я до змінної PlayerCharacter.

Рис. 3. 24. Механіка вибору персонажа

Джерело: [створено автором]

Висновки до розділу 3

У третьому розділі було детально розглянуто технічну складову проєкту, що охоплює засоби розробки, вимоги до програмного та апаратного забезпечення, а також особливості програмної реалізації основних елементів гри. У межах дослідження обґрунтовано вибір рушія Unreal Engine 5.4 як оптимального середовища для створення кросплатформної гри жанру runner-shooter, що дозволяє реалізовувати повний цикл розробки - від прототипу до збірки - без залучення сторонніх інструментів і мов програмування. Використання систем Blueprints, UMG, Enhanced Input забезпечило швидке створення ігрової логіки, інтерфейсу, ефектів і стабільну роботу гри як на ПК, так і на мобільних пристроях.

Визначено технічні та програмні вимоги до системи розробки й цільових платформ. Сформульовані мінімальні та рекомендовані характеристики обладнання гарантують стабільну роботу рушія й готової гри на різних пристроях. Особливу увагу приділено підтримці Android-платформ із Vulkan-рендером і сучасних Windows-конфігурацій із DX12. Це створює передумови для масштабування проєкту та майбутнього розширення його функціональності.

Найбільший обсяг роботи припав на опис програмної реалізації. У межах цього підрозділу створено впорядковану структуру контенту в директорії Runner, що забезпечує логічний поділ між компонентами гри: персонажем, трасою, перешкодами, бонусами, інтерфейсом і системою збережень. Детально реалізовано ігровий цикл, який охоплює основні карти - MainMenu, CS_Map, Loading та Runner_Map, - що формують повний користувацький потік від вибору персонажа до результатів сесії.

Описано архітектуру керування станами гри через BP_RunnerGameMode, що координує режими Ready, Playing, Paused і GameOver, а також пов’язану з ним логіку у PlayerController та HUD. У підсистемі керування персонажем реалізовано рух між смугами, стрибки, ковзання та стрільбу через систему Enhanced Input, що уніфікує керування для мобільних пристроїв і ПК. Генерація

нескінченної траси побудована на модульному принципі з використанням BP_Floor і BP_ObstacleDestroyer, що забезпечує безперервність ігрового процесу без перевантаження пам’яті.

У межах графічної частини реалізовано базову сценографію через Landscape і Modeling Tools, створено простий фон і оточення без втрати продуктивності. Колізійна система побудована на окремих каналах взаємодії для персонажа, перешкод і колекційних об’єктів, що гарантує коректну роботу зіткнень і подій. Задля підвищення ефективності рендерингу впроваджено Instanced Static Mesh (HISM), використано рівні деталізації (LOD) і компресію текстур для різних платформ, що дозволило досягти стабільних 50–60 FPS на мобільних пристроях і 60 FPS на ПК.

Також у грі реалізовано систему збору монет і збереження прогресу через Runner_SaveGame, а інтерфейс побудовано на базі UMG, із адаптацією до роздільностей і сенсорного керування. Окремо розроблено просту систему вибору персонажа (BP_CS_Characters), яка зберігає обраного героя між сеансами гри. Для етапів завантаження застосовано легкі сцени з Sequencer, що приховують ініціалізацію ресурсів та створюють плавний користувацький досвід.

Результати розробки засвідчили, що запропонована архітектура є стабільною, логічно організованою й масштабованою. Вона дозволяє легко додавати нові елементи геймплею, оптимізувати продуктивність і готувати гру до подальших етапів - локалізації, фінальної оптимізації та публікації у цифрових магазинах. Проведена реалізація підтвердила практичну ефективність обраних технічних рішень і продемонструвала можливість створення повноцінного ігрового продукту на основі візуального програмування без залучення коду C++.

РОЗДІЛ 4
ПІДГОТОВКА ГРИ ДО РЕЛІЗУ

4.1 Реліз на різних платформах

Публікація мобільної версії гри в Amazon Appstore передбачатиме проведення окремого циклу підготовки, оптимізації та тестування відповідно до вимог платформи Amazon. На початковому етапі в Unreal Engine у розділі Project Settings → Platforms → Android буде налаштовано параметри збірки відповідно до версій Android API, а також буде активовано підтримку архітектури arm64. Як основний рендерер планується використовувати Vulkan для забезпечення стабільної продуктивності на більшості пристроїв, доступних у магазині Amazon.

Після завершення технічної підготовки проєкт буде зібраний у формат .apk або .aab, а підписання застосунку виконуватиметься за допомогою власного keystore, сумісного з Android Studio. Попереднє тестування гри здійснюватиметься на реальних пристроях для перевірки стабільності керування, коректності інтерфейсу, частоти кадрів та загального користувацького досвіду.

Завантаження проєкту відбуватиметься через Amazon Developer Console, де буде сформована сторінка застосунку. До неї увійдуть основні матеріали: назва, опис, категорія, іконка, скріншоти, трейлер, вікові обмеження та інформація щодо конфіденційності та збору даних. Також буде налаштована сумісність із моделями пристроїв Fire та іншими Android-пристроями, які підтримує Amazon Appstore.

Після подання гри до публікації Amazon здійснюватиме автоматичну технічну перевірку, що включатиме контроль правильності підпису, аналіз наявності критичних помилок та оцінку відповідності застосунку рекомендаціям платформи. У разі успішного проходження перевірок гра буде передана на модерацію, яка може тривати від кількох годин до кількох днів.

Після схвалення модераторами гра стане доступною для завантаження в Amazon Appstore, а розробник отримуватиме можливість відстежувати статистику встановлень, оцінки користувачів, технічні звіти та інші дані через Amazon Developer Console.

4.2 Тестування та налагодження

У межах цього процесу перевірялась коректність роботи основних систем - руху персонажа, логіки перешкод і бонусів, системи рахунку, збереження прогресу та інтерфейсу. Тестування проводилося як на ПК, так і на мобільних пристроях, щоб виявити відмінності у поведінці елементів та FPS. Для аналізу продуктивності використовувались вбудовані профайлери Unreal Engine - Stat Unit, Stat GPU. За результатами перевірок виконувалося спрощення матеріалів, зменшення кількості активних ефектів і усунення дрібних логічних помилок.

4.3 Балансування геймплею та оптимізація інтерфейсу (UX/UI)

За допомогою кривих складності (Curve Float) у GameMode поступово збільшувалась швидкість бігу персонажа та частота появи перешкод. Тестування балансу проводилось кількома користувачами, щоб визначити оптимальну тривалість сесії, середню дистанцію та комфортний темп гри. Після декількох ітерацій було досягнуто стабільного рівня складності, який підходить як для новачків, так і для досвідчених гравців.

Меню гри, HUD були адаптовані під різні роздільності екранів - від мобільних HD-дисплеїв до широкоформатних моніторів ПК. Було проведено уніфікацію кольорової палітри, збільшено контрастність текстів.

4.4 Маркетингова підготовка

Буде створено набір візуальних матеріалів для сторінок у Amazon AppStore: логотип, іконку, скріншоти і банери. Для публікації буде підготовлено тексти описів - короткий (до 80 символів) і повний (до 4000 символів), які описують ігровий процес, жанр та ключові особливості. Також розроблено політику конфіденційності, яка відповідає вимогам Amazon AppStore.

Ефективна стратегія просування є важливою складовою життєвого циклу ігрового проєкту, оскільки саме вона забезпечує залучення аудиторії, формує впізнаваність бренду та підвищує комерційний потенціал гри. Для інді-проєктів, таких як Runner-Shooter, маркетингова кампанія має бути гнучкою, візуально привабливою й максимально орієнтованою на соціальні платформи, де сконцентрована цільова аудиторія мобільних гравців.

Основними каналами просування гри визначено соціальні мережі (Instagram, TikTok, Facebook), платформи для інді-розробників (itch.io, Game Jolt), а також ігрові сервіси з великою користувацькою базою, такі як Amazon AppStore. У межах цієї стратегії просування передбачено кілька ключових етапів.

Етап 1. Тизер-кампанія

Перший етап просування - створення короткого тизеру, який показує головну ідею гри без розкриття всіх деталей. Тизер тривалістю 15–20 секунд має передавати динаміку геймплею, стиль візуалізації та головну особливість гри - поєднання елементів раннера й шутера. Його можна поширювати через TikTok, Instagram Reels, YouTube Shorts, що дозволить швидко охопити мобільну аудиторію. На цьому ж етапі створюється базовий слоган (наприклад, “Run. Aim. Survive.”), який стане частиною майбутнього бренду гри.

Етап 2. Демоверсія

Наступним кроком є публікація демо-версії на платформах itch.io або Steam Next Fest. Це дозволяє отримати перший зворотний зв’язок від гравців, протестувати стабільність і виявити технічні недоліки. Демоверсія має включати одну карту, базовий набір перешкод і бонусів, а також систему підрахунку очок. Зібрані відгуки можна використати для покращення балансу, UI та керування.

Етап 3. Трейлер

На основі оновленої версії гри створюється офіційний трейлер, який демонструє основні механіки, персонажів, підсилення та ефекти. Оптимальна тривалість - 45–60 секунд. Трейлер супроводжується енергійним саундтреком і показує найдинамічніші моменти геймплею. Він публікується на сторінках гри в Steam, Google Play і соціальних мережах, а також використовується як рекламний матеріал під час подання на фестивалі інді-ігор.

Етап 4. Сторінка в Amazon AppStore

Для публікації гри створюється сторінка продукту в Amazon AppStore, яка містить назву гри, короткий опис (до 80 символів), повний опис (до 4000 символів), скріншоти, трейлер, категорію, рейтинг і політику конфіденційності. Особливу увагу приділено вибору іконки, яка має бути чіткою, контрастною й асоціюватися з динамікою гри. Також готується заставка (feature graphic) у форматі 1024×500 пікселів, яка використовується як банер на сторінці гри.

Формування бренду гри

Для створення впізнаваного образу гри розроблено елементи **візуальної айдентики**:

- **Логотип** - мінімалістичний текстовий знак із динамічними лініями, який передає рух і швидкість.
- **Іконка** - зображення головного персонажа або моменту стрибка, виконане у яскравих контрастних кольорах.

- **Постер** - композиція, що поєднує персонажа, трасу та слоган, може використовуватись у соціальних мережах або як головне промо-зображення.

Для збереження єдиного стилю використовується узгоджена кольорова палітра (синьо-помаранчева гама) та шрифт із чіткими геометричними формами, що підкреслює жанр гри.

Приклад застосування маркетингової стратегії

1. **Передреліз (Pre-launch):** публікація тизеру, створення сторінки гри в itch.io і коротких роликів у TikTok.
2. **Реліз:** публікація гри в Amazon AppStore, запуск трейлера, збір відгуків, перше оновлення [27].
3. **Після Реліз (Post-launch):** регулярні пости, оновлення контенту, публікація статистики (“досягнуто 10 000 гравців”), інтерактивні опитування, додавання рейтингових таблиць.

Розроблена маркетингова стратегія передбачає послідовний розвиток гри від тизеру до активного ком’юніті після релізу. Використання безкоштовних і доступних каналів просування дозволяє ефективно охопити цільову аудиторію без великих фінансових витрат. У майбутньому реалізація цього плану дасть змогу не лише збільшити впізнаваність гри Runner-Shooter, але й закріпити її позиції серед інді-проєктів у своєму жанрі.

Фінальним кроком стане підготовка збірок для публікації. Для Android буде створено підписаний .aab-пакет із використанням власного .keystore, який буде протестовано перед випуском у продакшн. Для Windows-версії буде виконано збірку у форматі Shipping і підготовлено інсталяційний пакет для публікації. Після перевірки обох збірок буде сформовано релізні версії з єдиною базою контенту та конфігураціями, що забезпечують стабільну роботу гри на різних пристроях.

Висновки до розділу 4

У межах підготовки гри до релізу було проведено комплексну роботу, спрямовану на підвищення якості ігрового процесу, стабільності та зручності користування. У ході тестування перевірено коректність роботи основних систем: руху персонажа, колізій, збору бонусів, інтерфейсу, а також логіки підрахунку очок і переходу між станами гри. Аналіз проводився як у середовищі редактора, так і на готових збірках для мобільних пристроїв та ПК. За допомогою інструментів Stat Unit, Stat GPU було виявлено кілька технічних моментів, що впливали на стабільність FPS, після чого виконано низку дрібних оптимізацій — спрощення матеріалів, корекцію колізій і покращення логіки обробки подій.

Під час балансування геймплею вдалося досягти стабільної динаміки, яка відповідає жанру runner-shooter: поступове зростання швидкості руху персонажа, збільшення кількості перешкод і поступове ускладнення траси створюють відчуття прогресу й виклику. Застосування кривих складності (Curve Float) дозволило зробити цей процес плавним і гнучким, а проведені ітерації тестів допомогли знайти комфортне співвідношення між швидкістю, складністю та тривалістю сесії.

Окрема увага приділялася користувацькому інтерфейсу - оновлені елементи HUD, меню та панелі управління стали більш адаптивними й зручними. Кольорова гама була уніфікована, а кнопки й тексти зроблені контрастнішими для покращення читабельності, що особливо важливо для мобільних пристроїв. Удосконалення структури інтерфейсу дозволило зробити взаємодію з грою інтуїтивною, а візуальні ефекти при натисканні й переходах додають динамічності та приємних відчуттів під час гри.

Водночас частина робіт, зокрема фінальна оптимізація, локалізація інтерфейсу та підготовка релізних збірок для публікації, запланована на подальші етапи. Вони передбачають впровадження системи багатомовності, остаточне налаштування продуктивності для мобільних пристроїв, стискання контенту та формування релізних пакетів для публікації в Amazon AppStore. Це

дозволить зробити продукт доступним для ширшої аудиторії, забезпечити кросплатформну стабільність і відповідність технічним вимогам маркетплейсів.

Загалом виконана робота на етапі підготовки до релізу дозволила підвищити якість геймплею, збалансувати складність і забезпечити комфортну взаємодію користувача з інтерфейсом. Подальші кроки будуть спрямовані на вдосконалення технічної частини, підвищення продуктивності та офіційне розповсюдження гри через цифрові платформи, що стане логічним завершенням життєвого циклу проєкту та переведе його в етап пострелізної підтримки.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи, перш за все було виконано комплексне дослідження, присвячене управлінню проєктом розробки мобільної гри жанру runner-shooter на рушії Unreal Engine 5.4. У ході дослідження сформовано теоретичні, організаційні та практичні засади створення інтерактивного ігрового продукту, що поєднує особливості мобільних і комп'ютерних платформ. Отримані результати підтверджують доцільність поєднання сучасних методологій проєктного менеджменту з інструментами геймдизайну, візуального програмування та технічної оптимізації.

У першому розділі розглянуто теоретичні основи управління ігровими проєктами. Доведено, що розробка ігор потребує поєднання класичних принципів проєктного менеджменту (планування, управління ресурсами, контролю якості) з гнучкими підходами Agile, Scrum та Kanban, які передбачають короткі ітерації, постійний зворотний зв'язок і можливість швидкого вдосконалення продукту. Проведено аналіз сучасних рушіїв (Unity, Unreal Engine, Godot, CryEngine), у результаті якого обґрунтовано вибір Unreal Engine 5.4 як оптимальної технологічної платформи для створення кросплатформної 3D-гри. Рушій забезпечує високоякісну графіку, систему візуального програмування Blueprints, ефективні інструменти роботи з UI (UMG), освітлення (Lumen) та оптимізації продуктивності. Також було досліджено ринок ігор-аналогів (Temple Run 2, Subway Surfers, Minion Rush, Into the Dead), що дозволило визначити ключові риси успішних runner-проєктів - просте керування, високу динаміку, короткі ігрові сесії та систему досягнень. Це дало підстави сформувати концепцію власного продукту як поєднання класичного раннера з бойовими елементами, бонусами та поступовим ускладненням геймплею.

Другий розділ було присвячено проектуванню архітектури та визначенню технічної основи гри. У результаті проведено аналіз предметної області та розроблено модульну архітектуру Unreal Engine 5.4, яка включає підсистеми Control, Movement, Combat, Scene Management, Resources, Stats, Core, UI, Saving

та Inputs. Для кожної з них створено окремі Blueprint-класи: BP_RunnerGameMode (логіка станів гри), CharacterBPs (поведінка персонажів), BP_Floor (генерація сегментів траси), BP_ObstacleDestroyer (видалення застарілих об’єктів), BP_Projectile (механіка кидання), BP_PowerUp_Magnet (система підсилень), Runner_SaveGame (збереження прогресу) тощо. Всі елементи об’єднано у впорядковану структуру контенту (Animations, Blueprints, Characters, Maps, Meshes, Textures), що спрощує масштабування та командну роботу.

Створено повний користувацький інтерфейс - головне меню, карту вибору персонажа, налаштування, екран паузи, результати гри та HUD. Навігація адаптована під мобільні пристрої та ПК, а система вводу Enhanced Input уніфікує різні типи керування. Реалізовано генерацію траси, перешкод, бонусів і підсилень із використанням object pooling, що підвищує продуктивність. Для збереження продуктивності застосовано LOD і контроль draw calls. Отримана архітектура є гнучкою, масштабованою та придатною до розширення - зокрема, для додавання нових типів рівнів, персонажів або ворогів у майбутньому.

У третьому розділі описано засоби розробки, технічні вимоги та програмну реалізацію гри. Для створення використовувалися стандартні інструменти Unreal Engine 5.4 - Blueprints, UMG, Modeling Tools, Landscape, а також редактор матеріалів. Гра протестована для роботи на Android (arm64, Vulkan, ASTC/ETC2) і Windows (x64, DX12). Проведено аналіз необхідних технічних ресурсів і забезпечено стабільну роботу на мобільних пристроях середнього класу. Реалізовано основні геймплейні механіки: рух персонажа між трьома смугами, стрибки, ухилення, збір бонусів і взаємодію з перешкодами. Інтерфейс побудовано на UMG із підтримкою різних роздільностей і режимів керування. Проведено базову оптимізацію за допомогою профайлерів Stat Unit, GPU Visualizer.

Під час підготовки гри до релізу виконано тестування функціональних систем, балансування складності, полірування інтерфейсу та UX. Визначено

етапи подальшої роботи - локалізація, фінальна оптимізація, створення сторінок у Amazon AppStore та Itch.io, а також підготовка збірок для публікації. Це дозволить завершити повний цикл управління проєктом — від ініціації до випуску продукту.

У результаті виконання роботи сформовано повноцінну систему управління розробкою ігрового проєкту, реалізовано всі ключові технічні компоненти, розроблено структурований ігровий прототип, готовий до тестування, подальшої оптимізації та релізу. Практична частина підтвердила ефективність використання Blueprints як основного інструменту візуального програмування, що дозволяє реалізовувати складну логіку без залучення коду C++. Створена архітектура довела свою стабільність, масштабованість і можливість кросплатформного розгортання.

Підсумовуючи, можна зазначити, що поставлені в роботі завдання виконано повністю. Розроблена гра демонструє практичну реалізацію теоретичних принципів управління проєктами в геймдеві, підтверджує доцільність застосування сучасних інструментів Unreal Engine та показує, як завдяки структурованому підходу до планування, архітектури та тестування можна створити конкурентоспроможний ігровий продукт. Проведена робота має прикладне значення для студентів, інді-розробників і фахівців у сфері управління IT-проєктами, адже демонструє взаємозв’язок між теорією проєктного менеджменту й практичною реалізацією геймдев-проєкту від ідеї до готового продукту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Unreal Engine 5 docs - Blueprints Visual Scripting [Електронний ресурс] URL:https://surl.li/mjnnrf (дата звернення 02.07.2025)
2. Widget Blueprints - Unreal Engine [Електронний ресурс] URL: https://surl.li/txpjae (дата звернення 02.07.2025)
3. Game Feel - A game designer's guide to virtual sensation by Steve Swink - 376 pages - 19 nov 2008 (дата звернення 03.03.2025)
4. Jira Atlassian [Електронний ресурс] URL: https://jira.atlassian.com (дата звернення 12.08.2025)
5. UE5 - Animation Blueprints / Character Movement [Електронний ресурс] URL: https://surl.lu/dzosoi (дата звернення 03.08.2025)
6. Unreal Engine asset Store. [Електронний ресурс] URL: https://www.fab.com/ (дата звернення 03.08.2025)
7. Temple Run [Електронний ресурс] URL: https://surl.li/mdhwgl (дата звернення 02.08.2025)
8. Subway Surfers [Електронний ресурс] URL: https://surl.li/iqbjvw (дата звернення 02.08.2025)
9. Jetpack Joyride[Електронний ресурс] URL: https://surl.li/bthknr (дата звернення 02.08.2025)
10. Unreal Engine assets of buildings URL: https://kitbash3d.com/collections/kits (дата звернення 02.08.2025)
11. Unreal Engine Blue book asset. [Електронний ресурс] URL:https://surl.li/khzszc (дата звернення 12.07.2025)
12. Content Examples Sample Project [Електронний ресурс] URL: https://surl.li/oipcmr (дата звернення 10.07.2025)
13. Landscape Brushes in Unreal Engine. [Електронний ресурс] URL: https://surl.lu/vshjnt (дата звернення 12.07.2025)

14. Landscape Manage Mode in Unreal Engine [Електронний ресурс] URL: https://surl.li/oewxuo (дата звернення 12.07.2025)
15. Saving and loading in Unreal Engine [Електронний ресурс] URL: https://surl.li/iudnuj (дата звернення 12.07.2025)
16. Enhanced Input In Unreal Engine. [Електронний ресурс] URL:https://surl.li/qzdkpk (дата звернення 12.07.2025)
17. Blueprints Visual Scripting for Unreal Engine 5: Unleash the true power of Blueprints to create impressive games and applications in UE5 [Електронний ресурс] URL: https://surl.li/amkoaw (дата звернення 12.08.2025)
18. Gameplay Framework - Unreal Engine [Електронний ресурс] URL: https://surl.li/latfof (дата звернення 12.08.2025)
19. Android Development Requirements [Електронний ресурс] URL:https://surl.li/qrdort (дата звернення 12.08.2025)
20. Scripting API Animator. [Електронний ресурс] URL: https://surl.li/pzsexc (дата звернення 12.08.2025)
21. Actors [Електронний ресурс] URL: https://surl.li/nbieec (дата звернення 12.08.2025)
22. Google Play Asset Delivery Reference [Електронний ресурс] URL:https://surl.li/dkgrif (дата звернення 12.08.2025)
23. Packaging IOS Projects [Електронний ресурс] URL: https://surl.lt/grgsmc (дата звернення 12.07.2025)
24. Animation Editors [Електронний ресурс] URL: https://surl.li/rkktuz (дата звернення 12.08.2025)
25. Skeleton Editor [Електронний ресурс] URL: https://surl.lu/lllaqp (дата звернення 15.07.2025)
26. UMG UI Designer Quick Start Guide [Електронний ресурс] URL: https://surl.li/fqjcys (дата звернення 16.06.2025)
27. Creating a Mobile Project [Електронний ресурс] URL: https://surl.li/udyjnf (дата звернення 12.08.2025)

28. Sounds and music. Sound Effects. [Електронний ресурс] URL: https://surl.li/spszvo (дата звернення 11.09.2025)
29. GitHub project. [Електронний ресурс] URL: https://github.com/Max355/Git_Runner (дата звернення 01.09.205)