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

# КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка сайд-скролер проєкту із застосування ігрового рушію Unreal Engine»

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

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

_____ Микола МАРКОВЕЦЬ
*(підпис)*

Виконав: здобувач вищої освіти групи ПД-42

Микола МАРКОВЕЦЬ

Керівник: Максим КУКЛІНСЬКИЙ
*канд. техн. наук*

Рецензент: _____

**Київ 2026**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

**Навчально-науковий інститут інформаційних технологій**

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

**ЗАТВЕРДЖУЮ**
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2026 р.

# ЗАВДАННЯ
# НА КВАЛІФІКАЦІЙНУ РОБОТУ

Марковцю Миколі Миколайовичу

1. Тема кваліфікаційної роботи: «Розробка сайд-скролер проєкту із застосування ігрового рушію Unreal Engine»
керівник кваліфікаційної роботи канд. техн. наук Максим КУКЛІНСЬКИЙ, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 року № 51.

2. Строк подання кваліфікаційної роботи «29» травня 2026 року.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про розробку 2D-платформерів, принципи проєктування ігрових механік, методи організації ігрового процесу та ігрового циклу, технічна документація Unreal Engine 4, матеріали аналізу існуючих програмних продуктів та сучасних ігрових рушіїв.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної галузі 2D side-scroller ігор, ролі візуального оформлення, анімацій, VFX та порівняння з іграми-аналогами.
2. Огляд засобів реалізації «Blade of Destiny»: Unreal Engine 4, Blueprint, Paper2D, Adobe Photoshop та Git.

3. Проєктування і розробка візуальної частини гри: персонажа, ворогів, оточення, рівнів, спрайтів, анімацій та ефектів атак.
4. Реалізація, оптимізація та тестування гри: організація проєкту в Unreal Engine 4, інтеграція ассетів, налаштування механік, перевірка працездатності та балансування геймплею.

5. Перелік графічного матеріалу: *презентація*
1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма прецедентів.
5. Алгоритм роботи застосунку.
6. Діаграма класів.
7. Екранні форми.
8. Апробація результатів дослідження

6. Дата видачі завдання «23» лютого 2026 року.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Підбір та аналіз науково-технічної літератури з розробки 2D-платформерів | 23.02-28.02.2026 | |
| 2 | Аналіз та дослідження існуючих ігор-аналогів | 01.03-07.03.2026 | |
| 3 | Огляд ігрових механік, алгоритмів та технологій розробки side-scroller проєктів | 08.03-15.03.2026 | |
| 4 | Проєктування 2D-платформера «Blade of Destiny» | 16.03-27.03.2026 | |
| 5 | Програмна реалізація 2D-платформера засобами Unreal Engine 4 | 28.03-17.04.2026 | |
| 6 | Тестування програмного продукту | 18.04-27.04.2026 | |
| 7 | Оформлення роботи: вступ, висновки, реферат | 28.04-04.05.2026 | |
| 8 | Розробка демонстраційних матеріалів | 05.05-10.05.2026 | |
| 9 | Попередній захист роботи | 11.05-22.05.2026 | |

Здобувач вищої освіти _____ Микола МАРКОВЕЦЬ
*(підпис)*

Керівник
кваліфікаційної роботи _____ Максим КУКЛІНСЬКИЙ
*(підпис)*

# РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: _51_ стор., _2_ табл., __11__ рис., _26_ джерел.

*Мета роботи* — створення візуальної складової 2D side-scroller гри «Blade of Destiny» засобами Unreal Engine 4, включаючи розробку та інтеграцію графічних матеріалів, анімацій і візуальних ефектів в ігровий процес.

*Об'єкт дослідження* — процес розробки 2D side-scroller гри із застосуванням ігрового рушія Unreal Engine 4.

*Предмет дослідження* — методи та засоби створення, підготовки, оптимізації та інтеграції 2D-графіки, спрайтів, анімацій і візуальних ефектів у грі «Blade of Destiny».

*Короткий зміст роботи*: у роботі проаналізовано особливості жанру 2D side-scroller, роль візуального оформлення у 2D-іграх та існуючі аналоги, зокрема Hollow Knight, Celeste та Magic Rampage. Розглянуто засоби реалізації проєкту: Unreal Engine 4, Blueprint, Paper2D, Adobe Photoshop та Git. Розроблено візуальну частину гри «Blade of Destiny», зокрема графіку головного персонажа, ворогів, елементів оточення, рівнів, спрайтів, анімацій та візуальних ефектів для атак і ударів. Підготовлено ассети до імпорту в Unreal Engine 4, виконано їх інтеграцію через Paper2D, налаштовано flipbook-анімації та базову взаємодію графічних елементів з ігровими механіками. Проведено тестування ігрового процесу, перевірено роботу анімацій, колізій, атак, відображення спрайтів, меню налаштувань та зручність керування персонажем.

*Сферою використання результатів роботи* є подальша розробка та вдосконалення 2D side-scroller гри «Blade of Destiny», зокрема розширення рівнів, додавання нових ворогів, анімацій, візуальних ефектів і геймплейних механік.

КЛЮЧОВІ СЛОВА: 2D SIDE-SCROLLER, BLADE OF DESTINY, UNREAL ENGINE 4, BLUEPRINT, PAPER2D, ADOBE PHOTOSHOP, СПРАЙТИ, АНІМАЦІЇ, VFX, ІГРОВІ МЕХАНІКИ.

**ЗМІСТ**

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Side-scroller — жанр гри, у якому події відображаються збоку, а персонаж переважно рухається горизонтально рівнем.

UE4 — ігровий рушій Unreal Engine 4.

Blueprint — система візуального програмування в Unreal Engine 4, яка дозволяє створювати ігрову логіку за допомогою вузлів і зв’язків між ними.

Paper2D — інструмент Unreal Engine 4 для роботи з 2D-графікою, спрайтами, flipbook-анімаціями, тайлами та 2D-колізіями.

Спрайт — двовимірне зображення, яке використовується для відображення персонажів, ворогів, предметів, ефектів або елементів оточення.

Ассет — графічний, звуковий або інший ресурс, який використовується в ігровому проєкті.

VFX(Visual Effects) — візуальні ефекти, які використовуються для підсилення ігрових дій, наприклад ударів, атак, спалахів або зіткнень.

Flipbook-анімація — анімація, створена з послідовності окремих кадрів-спрайтів, які швидко змінюються один за одним.

Колізія — область зіткнення об’єкта, яка визначає взаємодію персонажа з платформами, ворогами або іншими елементами рівня.

Тайл — невеликий графічний елемент, з якого складаються частини рівня, наприклад земля, стіни, платформи або декоративні фрагменти.

Idle — стан або анімація спокою персонажа, коли він не виконує активних дій.

# ВСТУП

*Актуальність*: сучасна індустрія комп’ютерних ігор активно розвивається, а 2D-проєкти жанру side-scroller залишаються популярними завдяки простому ігровому процесу, динамічному проходженню рівнів та можливості створення виразного візуального стилю. У таких іграх важливу роль відіграє не лише механіка руху чи бойова система, а й графічне оформлення. Саме воно формує атмосферу, допомагає гравцеві орієнтуватися в ігровому просторі та робить проєкт впізнаваним [1].

Актуальність теми зумовлена необхідністю розробки цілісної візуальної частини гри «Blade of Destiny» з використанням Unreal Engine 4. Для якісної реалізації 2D side-scroller проєкту потрібно не лише створити персонажів, ворогів, оточення та ефекти, а й правильно підготувати ассети, інтегрувати їх у рушій, налаштувати анімації та забезпечити їхню коректну взаємодію з ігровими механіками [2].

Особливість цієї роботи полягає в тому, що основний акцент зроблено не на створенні окремої механіки, а на поєднанні візуальних матеріалів із роботою гри. Для 2D side-scroller проєкту важливо, щоб персонаж, вороги, рівні, анімації та ефекти не існували окремо один від одного. Вони мають працювати разом: рух персонажа повинен підтримуватися анімацією, атака — візуальним ефектом, а зіткнення з ворогом — зрозумілою реакцією на екрані.

*Об’єктом дослідження* є процес розробки 2D side-scroller гри із застосуванням ігрового рушія Unreal Engine 4.

*Предметом дослідження* є методи та засоби створення, підготовки, оптимізації та інтеграції 2D-графіки, анімацій, спрайтів і візуальних ефектів у проєкті «Blade of Destiny».

*Метою роботи* є забезпечення цілісного графічного стилю, коректної інтеграції ассетів, анімацій і візуальних ефектів у ігровий процес.

Для досягнення поставленої мети необхідно виконати такі завдання:

1. проаналізувати особливості жанру side-scroller;

2. дослідити аналоги 2D-платформерів;
3. обґрунтувати вибір засобів реалізації;
4. створити графічні матеріали для персонажів, ворогів та оточення;
5. підготувати ассети до імпорту в Unreal Engine 4;
6. інтегрувати спрайти, анімації та VFX;
7. провести тестування і балансування ігрового процесу.

*Методи дослідження*: у роботі використано метод аналізу предметної галузі для визначення особливостей 2D side-scroller ігор, порівняльний аналіз існуючих аналогів, методи проєктування ігрових об’єктів, практичні методи створення 2D-графіки в Adobe Photoshop, методи інтеграції ассетів у Unreal Engine 4 за допомогою Paper2D, а також методи тестування для перевірки коректності відображення графіки, анімацій, ефектів і базових механік гри.

*Наукова новизна роботи* полягає у поєднанні методів створення 2D-графіки, підготовки ассетів, інтеграції анімацій та візуальних ефектів у межах єдиного side-scroller проєкту на базі Unreal Engine 4. У роботі запропоновано підхід до формування візуальної цілісності гри «Blade of Destiny», що передбачає узгодження персонажів, ворогів, оточення, текстур, анімацій і VFX із загальною атмосферою проєкту.

*Практична значущість* полягає у створенні візуальної основи 2D-гри «Blade of Destiny», яка може бути використана для подальшого розвитку проєкту, розширення рівнів, додавання нових персонажів, ворогів, анімацій та ігрових механік. Розроблені графічні матеріали, спрайти, анімації та візуальні ефекти покращують читабельність ігрової сцени та підвищують якість взаємодії користувача з ігровим середовищем.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ОСОБЛИВОСТЕЙ 2D SIDE-SCROLLER ІГОР

## 1.1 Характеристика жанру side-scroller та його особливості

Жанр side-scroller належить до класичних напрямів у розробці 2D-ігор. Його головна особливість полягає в тому, що гравець бачить події збоку, а персонаж рухається в межах двовимірного простору. Такий формат добре підходить для ігор, де основна увага зосереджена на проходженні рівнів, стрибках, боях і взаємодії з об’єктами середовища [3].

У side-scroller іграх камера зазвичай не змінює ракурс, тому гравець постійно спостерігає за персонажем з одного боку. Це робить ігровий процес зрозумілим: користувач бачить, куди рухається герой, де розташовані платформи, вороги та перешкоди. Завдяки такому підходу рівень можна будувати послідовно, поступово додаючи нові небезпеки, складніші ділянки та бойові ситуації [5, 7].

Основою такого жанру є рух персонажа рівнем. Гравець переміщується вперед, перестрибує перешкоди, уникає небезпечних зон і взаємодіє з ворогами. У грі «Blade of Destiny» це реалізується через керування самураєм, який проходить локації, атакує мечем і реагує на зіткнення з противниками. Саме тому для цього проєкту важливими є не лише самі механіки, а й те, наскільки зрозуміло вони показані на екрані.

Ще однією характерною рисою side-scroller ігор є поступове ускладнення. Спочатку гравець звикає до руху, стрибка та атаки, а далі стикається з більш складними ділянками рівня. Це можуть бути вороги біля платформ, небезпечні зони, обмежений простір для маневру або потреба швидко змінювати дію. Такий підхід допомагає не перевантажувати користувача на початку гри й поступово вводити його в основні правила проходження [2, 5].

Для «Blade of Destiny» жанр side-scroller обрано через його зручність для бойового 2D-проєкту. Образ самурая добре поєднується з видом збоку, оскільки

гравець чітко бачить напрям руху, положення меча та момент атаки. Бокова перспектива також дозволяє краще показати ворогів, платформи, ефекти ударів і загальну побудову рівня.

## 1.2 Роль візуального оформлення у 2D-іграх

Візуальне оформлення у 2D-іграх напряму впливає на те, як гравець сприймає проєкт. Ще до того, як користувач повністю розуміє механіки, він бачить персонажа, фон, кольори, анімації та загальний настрій гри. Саме тому графіка у 2D-проєктах не може бути випадковою. Вона має одразу показувати, у якому світі відбуваються події та яку атмосферу автори хочуть передати [12, 15].

У side-scroller іграх візуальна частина повинна допомагати проходженню. Гравець має швидко розуміти, де розташована платформа, звідки може з’явитися небезпека, який об’єкт є ворогом і коли відбувається атака. У матеріалах з геймдизайну платформерів окремо наголошується, що рівні мають будуватися навколо руху персонажа, перешкод і реакції гравця, тому графіка повинна підтримувати ці дії, а не заважати їм [11].

Для 2D side-scroller гри важливо, щоб гравець швидко зчитував ситуацію на екрані. На відміну від тривимірних ігор, де користувач може змінювати ракурс камери, у 2D-проєктах більшість подій відбувається в одній площині. Через це всі важливі об’єкти мають бути помітними одразу: герой, вороги, платформи, небезпечні зони та ефекти атаки. Якщо ці елементи погано відрізняються між собою, гравець може помилятися не через складність механіки, а через недостатню зрозумілість сцени [13, 15].

Також значення має зв’язок між графікою та діями персонажа. Коли гравець натискає кнопку руху, стрибка або атаки, він очікує побачити швидку й зрозумілу реакцію на екрані. Саме тому візуальне оформлення не можна розглядати окремо від геймплею. Спрайти, анімації та ефекти повинні підтверджувати дію гравця: показувати рух, момент удару, отримання шкоди або зіткнення з ворогом [10].

Під час роботи над «Blade of Destiny» враховано, що головний персонаж повинен добре виділятися на фоні локації. Це особливо важливо для гри з бойовими сценами, де гравець має швидко бачити самурая, напрям його руху та момент удару. Якщо персонаж або ворог зливається з фоном, гравець може помилитися не через складність гри, а через погану видимість сцени.

Окреме значення мають анімації. Вони показують, що саме робить персонаж у конкретний момент: стоїть, біжить, стрибає, атакує або отримує шкоду. Без анімацій герой виглядав би статично, а його дії були б менш зрозумілими. У «Blade of Destiny» анімації потрібні не лише для естетики, а й для зворотного зв'язку, щоб гравець бачив результат своєї дії.

Візуальні ефекти також виконують практичну роль. Удар мечем, спалах, слід від атаки або реакція на зіткнення допомагають гравцеві побачити результат своєї дії. У джерелах про візуальний зворотний зв'язок зазначається, що такі реакції допомагають користувачу краще розуміти наслідки власних дій у грі [6, 13]. Тому в «Blade of Destiny» ефекти атак і ударів мають підсилювати бойові сцени та робити їх більш зрозумілими.

Також важливо, щоб усі графічні елементи виглядали як частини одного світу. Персонаж, вороги, фон, платформи й ефекти повинні мати спільний стиль. Якщо вони відрізняються занадто сильно, гра сприймається як набір окремих зображень, а не як цілісний проєкт. Для «Blade of Destiny» це особливо важливо, бо атмосфера гри будується навколо образу самурая, небезпечних локацій і постійного руху вперед, оскільки візуальне оформлення у 2D-іграх впливає не тільки на зовнішній вигляд, а й на зручність проходження.

У 2D-грі гравець не має можливості змінити кут огляду або наблизити камеру так, як це часто відбувається у 3D-проєктах. Через це вся важлива інформація повинна бути зрозумілою з одного ракурсу. Якщо платформа, ворог або ефект атаки погано відрізняються від фону, гравець може сприймати ситуацію неправильно. У такому випадку складність виникає не через задум гри, а через недостатньо вдале візуальне рішення.

Для «Blade of Destiny» це має практичне значення, оскільки гра поєднує платформерне проходження та бойові сцени. Під час руху рівнем гравець одночасно стежить за положенням персонажа, платформами, ворогами й моментом атаки. Тому візуальна частина повинна не тільки прикрашати гру, а й допомагати швидко приймати рішення під час проходження.

## 1.3 Аналіз існуючих аналогів

Аналіз існуючих аналогів потрібний для того, щоб краще визначити, які рішення вже використовуються в 2D-платформерах і side-scroller іграх. У межах цієї роботи аналоги розглядаються не тільки як приклади готових ігор, а і як джерело практичних рішень щодо побудови рівнів, оформлення персонажів, створення ворогів, використання анімацій та візуальних ефектів. Такий підхід дозволяє зрозуміти, які елементи добре працюють у подібних проєктах, а які можуть заважати проходженню або перевантажувати сцену.

Для гри «Blade of Destiny» важливо було проаналізувати саме ті проєкти, у яких помітну роль відіграють рух персонажа, читабельність рівня, бойові дії або візуальна атмосфера. Через це для порівняння обрано Hollow Knight, Celeste та Magic Rampage(рис 1.1). Ці ігри відрізняються між собою за стилем і темпом проходження, але кожна з них має окремі рішення, корисні для розробки власного 2D side-scroller проєкту.


Рис. 1.1 Логотипи ігор-аналогів

Hollow Knight розглядається як приклад гри з виразною атмосферою, темним візуальним стилем і цілісним оформленням світу. У цьому проєкті важливим є не лише сам геймплей, а й те, як локації, персонажі, вороги та анімації працюють разом. Для «Blade of Destiny» цей приклад є корисним через підхід до створення настрою гри та узгодження всіх візуальних елементів [23].

Celeste використано для аналізу читабельності сцени та точності керування. У цій грі графіка не перевантажена деталями, а рівні побудовані так, щоб гравець швидко розумів, куди рухатися і як подолати перешкоду. Для «Blade of Destiny» це важливо, оскільки side-scroller гра з платформами й боями потребує зрозумілої сцени, де персонаж, вороги, платформи та небезпечні зони не зливаються між собою [24].

Magic Rampage обрано як приклад поєднання платформера з активною бойовою системою. У цій грі велике значення мають атаки, зброя, зіткнення з ворогами та візуальні ефекти. Для «Blade of Destiny», де головний персонаж є самураєм і використовує меч, такий аналог допомагає краще оцінити роль анімацій атак, ефектів ударів і реакції ворогів на пошкодження [25].

Під час аналізу увагу приділено кільком основним критеріям: жанру, візуальному стилю, побудові рівнів, бойовій системі, читабельності сцени, анімаціям та візуальним ефектам. Окремо враховано, як ці елементи можуть бути застосовані у «Blade of Destiny». Проєкт не копіює готові рішення аналогів, але використовує їх як орієнтир для вибору власного підходу до графіки, анімацій, бойових сцен і загального стилю гри.

Аналіз аналогів також допомагає чіткіше визначити напрям розробки. Для «Blade of Destiny» основним завданням є створення 2D side-scroller гри з виразним головним персонажем, зрозумілими ворогами, атмосферними локаціями та помітними ефектами атак. Саме тому порівняння з іншими іграми дає можливість краще обґрунтувати вибір візуального стилю, структури рівнів і способу подання бойових дій.

### 1.3.1 Hollow Knight

Hollow Knight — 2D-екшн-пригодницька гра, у якій гравець досліджує великий підземний світ із різними локаціями, ворогами та босами. Розробники описують гру як подорож забутим королівством, наповненим таємницями, небезпечними істотами й атмосферними місцями. Саме атмосфера є однією з найсильніших сторін цього проєкту [23].

Візуально Hollow Knight вирізняється темною палітрою, плавними анімаціями та добре продуманим дизайном персонажів(див рис. 1.2). Головний герой має простий, але дуже впізнаваний вигляд. Вороги й локації також виконані в одному стилі, тому світ гри сприймається цілісно. Мною встановлено, що для 2D-гри важливо не перевантажувати персонажа деталями, якщо його силует і так добре читається на фоні.

Рис. 1.2 Демонстрація візуального оформлення Hollow Knight

Для «Blade of Destiny» корисним є підхід Hollow Knight до створення атмосфери. Графіка в такій грі не просто прикрашає рівні, а допомагає передати настрій світу. У моєму проєкті це можна застосувати через виразний образ самурая, темніші або контрастніші локації, помітні атаки та єдиний стиль ворогів і оточення.

### 1.3.2 Celeste

У грі Celeste головний акцент зроблено на точному керуванні, стрибках і проходженні складних ділянок рівня. Офіційний опис, на сторінці у Steam, підкреслює, що це “super-tight, hand-crafted platformer”, тобто платформер із дуже точним керуванням і вручну створеними випробуваннями. У грі є велика кількість екранів із платформерними задачами, секретами та поступовим ускладненням [24].

Візуальний стиль Celeste простіше, ніж у Hollow Knight, але це не мінус, а навпаки: гравець одразу бачить, куди можна стрибнути, де знаходиться небезпека і як рухатися далі. Це демонструє, що 2D-графіка не обов’язково має бути дуже деталізованою. Важливіше, щоб вона була зрозумілою та не заважала проходженню. Нижче наведено простоту візуального стилю гри.

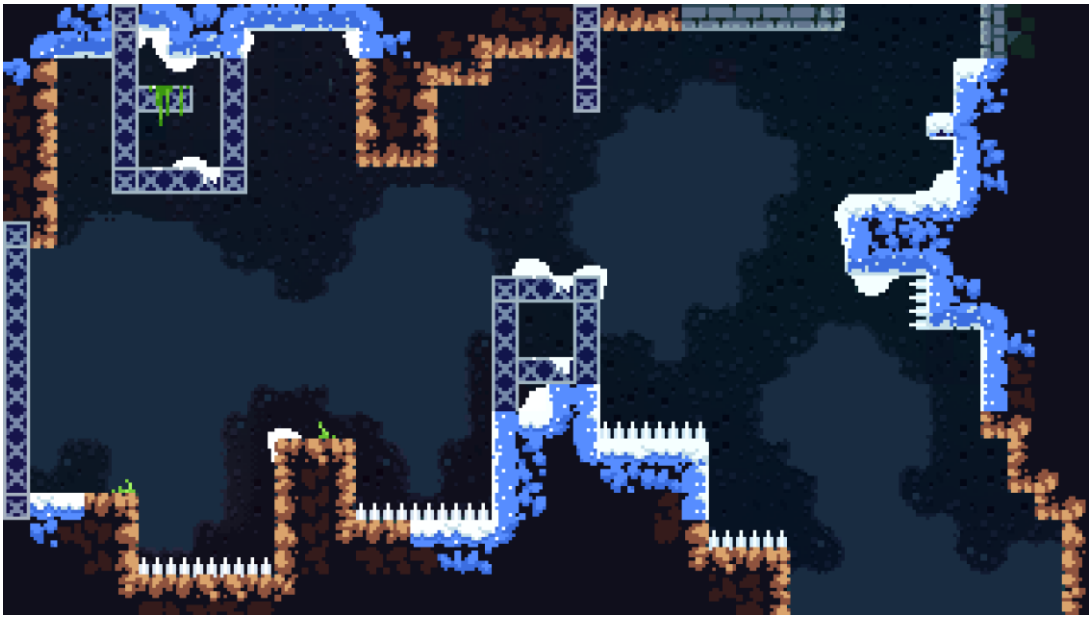


Рис. 1.3 Celeste – приклад простого візуального стилю

Для «Blade of Destiny» цей приклад є важливим через підхід до читабельності сцени. Адже в іграх цього жанру гравець має швидко розуміти ситуацію на екрані. Тому під час створення графіки я стежив за тим, щоб фон не заважав бачити персонажа, ворогів і межі платформ. Це напряму впливає на комфорт проходження.

### 1.3.3 Magic Rampage

Окрім комп’ютерних ігор цього жанру, проаналізовано також мобільні ігри. Однією з найвідоміших є Magic Rampage — 2D-екшн-RPG-платформер, який поєднує платформерний ігровий процес із рольовими елементами. У грі є проходження підземель, вороги, секретні зони, битви з босами, зброя та кастомізація персонажа. Офіційні сторінки гри описують її як швидкий платформер з елементами RPG та бойовим геймплеєм [25].

На відміну від Celeste, у Magic Rampage більший акцент зроблено саме на боях і розвитку персонажа. Гравець не лише проходить рівні, а й використовує різну зброю, стикається з ворогами та шукає приховані місця. Через це візуальні ефекти ударів, атак і взаємодії з ворогами мають велике значення.

Рис. 1.4 Демонстрація процесу бою у Magic Rampage

Для «Blade of Destiny» цей аналог корисний тим, що він показує поєднання платформера з бойовою системою. Оскільки в моєму проєкті головний персонаж — самурай, важливо якісно реалізувати удари, зіткнення з ворогами та реакцію на отримання шкоди. Саме тому анімації атак і VFX мають бути не другорядними елементами, а частиною самого геймплею.

### 1.3.4 Порівняльний аналіз аналогів

Після аналізу аналогів можна зробити висновок, що кожна з розглянутих ігор має власні сильні сторони. Hollow Knight добре демонструє важливість атмосфери, стилю та цілісного світу. Celeste показує, наскільки важливими є зрозумілість сцени та точність керування. Magic Rampage є прикладом поєднання платформера з активними боями, зброєю та візуальними ефектами.

Для «Blade of Destiny» доцільно врахувати ці підходи, але не копіювати їх напряму. Гра має власну основу: персонажа-самурая, бойові сцени, небезпечні локації та стилізовану 2D-графіку. Я визначив, що для цього проєкту найважливішими є три напрями: зрозуміле відображення персонажа на сцені, єдиний стиль оточення та помітні ефекти атак.

Таблиця 1.1

**Порівняння ігор-аналогів**

| **Технічні характеристики** | **Hollow Knight** | **Magic Rampage** | **Celeste** | **Blade of Destiny** |
|---|---|---|---|---|
| Бойова система | + | + | - | + |
| Дослідження світу | + | - | - | + |
| Магазин | лише покращення | скіни | - | покращення, скіни |
| Прогресія персонажа | + | - | - | + |
| Покращення після рівня | + | - | - | + |
| Рівень складності | складний | нормальний | легкий | нормальний, складний |
| Лінійний світ | - | + | + | + |

Отже, аналіз аналогів показав, що якісний 2D side-scroller має поєднувати зрозумілий геймплей, виразну графіку, якісні анімації та помітні бойові ефекти. Для «Blade of Destiny» важливо не лише створити окремі графічні матеріали, а й зробити так, щоб вони працювали разом у межах єдиного ігрового світу.

## 1.4 Визначення вимог до гри Blade of Destiny

Перед подальшою розробкою «Blade of Destiny» потрібно було визначити вимоги до гри. Це дало змогу зрозуміти, які можливості має підтримувати проєкт, як повинна виглядати ігрова сцена та які елементи потребують окремої перевірки під час тестування. Для 2D side-scroller гри важливо враховувати не лише самі механіки, а й те, як вони сприймаються гравцем: чи добре видно персонажа, чи зрозуміло працює атака, чи не заважає фон проходженню рівня.

*Функціональні вимоги* описують основні дії, які гравець може виконувати під час проходження. У грі реалізовано керування головним персонажем, рух у межах рівня, стрибок, атаку мечем, взаємодію з ворогами, отримання шкоди та проходження локації. Також передбачено використання анімацій для різних станів персонажа, зокрема спокою, бігу, стрибка, атаки, отримання пошкодження та смерті. Ці механіки є базовими для гри обраного жанру, тому вони повинні працювати зрозуміло й не створювати відчуття випадковості [1, 6].

До функціональних вимог також належить наявність зрозумілого зворотного зв’язку після дій гравця. Коли гравець натискає кнопку атаки, гра має не лише виконати перевірку зіткнення, а й показати відповідну анімацію та ефект удару. Коли персонаж отримує шкоду, це повинно відображатися через зміну стану або реакцію героя. Завдяки цьому користувач краще розуміє, які дії спрацювали та який результат вони дали.

Окремо враховано вимоги до бойової взаємодії. Оскільки головний герой є самураєм, атака мечем виступає однією з основних дій. Під час її реалізації важливо не лише запустити механіку нанесення шкоди, а й показати гравцеві сам момент удару. Для цього використано анімацію атаки та візуальний ефект траєкторії меча. Якщо удар не має помітного візуального підтвердження, гравець може не зрозуміти, чи дія виконалася правильно.

*Нефункціональні вимоги* стосуються загальної якості гри. Проєкт має запускатися стабільно, мати зрозуміле керування та не перевантажувати гравця зайвими деталями на екрані. У side-scroller грі користувач постійно стежить за

рухом персонажа, платформами, ворогами та небезпеками, тому сцена повинна залишатися читабельною. Якщо важливі об’єкти зливаються з фоном або атака виглядає непомітно, складність гри починає залежати не від геймплею, а від проблем із відображенням.

Окрему увагу потрібно приділити зручності налаштувань гри. Оскільки проєкт містить меню з параметрами екрана, мови, V-Sync та якості текстур, ці елементи мають бути простими для сприйняття. Користувач не повинен витрачати багато часу на пошук потрібного параметра. Меню налаштувань у такому проєкті виконує допоміжну роль, але воно також впливає на загальне враження від гри.

*Вимоги до графічного оформлення* є одними з основних для цієї роботи. Персонаж, вороги, фони, платформи, декоративні об’єкти та ефекти мають поєднуватися між собою. Головний герой повинен добре виділятися на фоні локації, вороги мають швидко розпізнаватися як небезпека, а платформи — чітко читатися як поверхні для руху. Для цього під час створення графіки враховано контраст, силуети об’єктів і загальний стиль сцени.

*До вимог щодо анімацій* належить інтуїтивно зрозуміле відображення станів персонажа та ворогів. Анімація не повинна бути просто прикрасою: вона показує гравцеві, що саме відбувається в конкретний момент. Наприклад, біг має відповідати швидкості руху, стрибок — положенню персонажа в повітрі, а атака — моменту удару мечем. Для ворогів також потрібні прості й зрозумілі стани: очікування, рух, атака, отримання шкоди або зникнення після поразки [12, 16,].

*Вимоги до продуктивності* пов’язані з тим, що гра створюється на базі Unreal Engine 4 і використовує велику кількість 2D-ассетів. Графічні матеріали потрібно готувати так, щоб вони нормально виглядали в ігровій сцені й не створювали зайвого навантаження. Для цього застосовано відповідні розміри зображень, формат PNG із прозорістю, впорядковану структуру файлів і повторне використання частини елементів оточення [18, 19].

Також у грі передбачено базові налаштування відображення, зокрема зміну роздільної здатності, режиму екрана, вертикальної синхронізації, мови інтерфейсу та якості текстур. Повзунок якості текстур не перемикає окремі набори

низькоякісних ассетів, оскільки такі дублікати не створювалися. Він використовується як загальне налаштування відображення сцени, при якому основні графічні елементи залишаються тими самими.

## 1.5 Постановка завдання на розробку проєкту

Завданням роботи є створення 2D side-scroller гри «Blade of Destiny» на базі Unreal Engine 4 з акцентом на візуальну частину, анімації, бойові ефекти та інтеграцію графічних матеріалів у рушій. Гравець керує самураєм, який проходить небезпечні локації, долає перешкоди, взаємодіє з ворогами та використовує меч у бойових ситуаціях. Гра повинна мати зрозумілий ігровий процес, читабельну сцену та єдиний стиль оформлення [1, 3, 4].

Таблиця 1.2

**Етапи виконання завдання для проєкту «Blade of Destiny»**

| Етап роботи | Що виконано в межах проєкту | Значення для гри |
|---|---|---|
| Визначення ідеї гри | Обрано формат 2D side-scroller гри з персонажем-самураєм, який проходить небезпечні локації | Дає основу для сюжету, стилю та подальшої побудови рівнів |
| Формування візуального стилю | Визначено загальний настрій гри, характер оточення, вигляд персонажа, ворогів і бойових ефектів | Допомагає зробити гру впізнаваною та візуально цілісною |
| Створення головного персонажа | Підготовлено образ самурая, його силует, основні пози та графіку для анімацій | Гравець отримує зрозумілого центрального героя, за яким легко стежити на екрані |
| Розробка ворогів | Створено образи супротивників, які відрізняються від героя та сприймаються як загроза | Підсилює бойову частину гри та робить небезпеку на рівні помітною |
| Підготовка оточення | Створено фони, платформи, тайли, декоративні елементи лісу та печер | Формує ігровий світ і допомагає будувати рівні |
| Налаштування механік | Реалізовано рух, стрибок, атаку, отримання шкоди та взаємодію з ворогами | Забезпечує базовий ігровий процес |

Продовження таблиці 1.2

**Етапи виконання завдання для проєкту «Blade of Destiny»**

| Етап роботи | Що виконано в межах проєкту | Значення для гри |
|---|---|---|
| Додавання бойових ефектів | Підготовлено ефекти траєкторії удару, спалахи та реакції на зіткнення | Робить атаки помітними та зрозумілими для гравця |
| Перевірка результату | Проведено тестування керування, анімацій, колізій, атак і загального проходження | Дозволяє виявити помилки та покращити зручність гри |
| Підготовка ассетів | Графіку збережено у потрібному форматі, перевірено розміри, прозорий фон і структуру файлів | Спрощує імпорт матеріалів в Unreal Engine 4 |
| Інтеграція в рушій | Спрайти перенесено в Unreal Engine 4, налаштовано Paper2D та flipbook-анімації | Графічні матеріали стають частиною ігрової сцени |

Подані в таблиці етапи показують, що розробка «Blade of Destiny» складається не лише зі створення графіки. Кожен графічний матеріал проходить кілька кроків: спочатку формується ідея, потім створюється зображення, після цього воно готується до імпорту, переноситься в рушій і перевіряється у сцені. Такий підхід дозволяє оцінювати не тільки зовнішній вигляд ассетів, а й те, наскільки правильно вони працюють у грі.

Важливо, що етапи не завжди виконуються строго один раз. Після інтеграції або тестування окремі елементи можуть потребувати доопрацювання: зміни розміру, положення, контрасту, тривалості анімації або сили візуального ефекту. Це нормальний процес для 2D-гри, оскільки частина проблем стає помітною лише після перевірки матеріалів у реальній ігровій сцені.

Першим завданням стало визначення загальної концепції гри. Потрібно було сформувати образ головного героя, атмосферу світу, характер локацій і загальний напрям візуального стилю. Для цього обрано тему самурая, який рухається через ворожий світ. Такий образ добре підходить для side-scroller гри, оскільки дозволяє чітко показати рух персонажа, напрям атаки, зіткнення з ворогами та поступове проходження рівнів.

Наступним завданням стала підготовка графічних матеріалів. До них належать спрайти головного героя, ворогів, елементи оточення, фони, платформи, тайли та ефекти для атак. Під час створення таких матеріалів потрібно було враховувати не тільки зовнішній вигляд, а й подальше використання в рушії. Наприклад, персонаж має бути придатним для створення анімацій, а платформи — зручними для налаштування колізій.

Окрему увагу приділено анімаціям. Для головного персонажа необхідно підготувати стани спокою, бігу, стрибка, атаки, отримання шкоди та смерті. Ці стани пов’язані з базовими механіками гри, тому вони мають запускатися в потрібний момент і зрозуміло показувати дію героя. Якщо анімація атаки не збігається з моментом нанесення шкоди, бій виглядає неточно, навіть якщо механіка працює технічно правильно [18, 19].

Ще одним завданням стала інтеграція ассетів у Unreal Engine 4. Після підготовки графіки матеріали потрібно імпортувати в проєкт, налаштувати спрайти, створити flipbook-анімації через Paper2D, перевірити масштаб і положення об’єктів у сцені. На цьому етапі важливо, щоб зображення не просто відображалися в рушії, а правильно працювали разом із рухом персонажа, атакою, зіткненнями та переходами між станами[20].

Завершальним етапом є тестування та балансування. Потрібно перевірити, чи зручно керувати персонажем, чи правильно працюють колізії, чи не виникають проблеми зі спрайтами, чи помітні атаки та чи не перевантажена сцена деталями. У результаті гра має сприйматися не як набір окремих механік, а як цілісний 2D-проєкт із зрозумілим рухом, боями й візуальним стилем.

# 2 ОГЛЯД ЗАСОБІВ РЕАЛІЗАЦІЇ 2D-ПЛАТФОРМЕРА «BLADE OF DESTINY»

## 2.1 Ігровий рушій Unreal Engine 4

Під час розробки 2D side-scroller гри важливо обрати середовище, у якому можна зручно працювати з рівнями, графікою, анімаціями, об’єктами сцени та логікою взаємодії. Для таких проєктів можуть використовуватися різні рушії, зокрема Unity, Godot, GameMaker Studio та Unreal Engine 4. Усі вони підтримують створення 2D-ігор, але відрізняються підходом до роботи з логікою, структурою проєкту та інтеграцією графічних матеріалів.

Unity має розвинені інструменти для 2D- та 3D-розробки. У ньому можна працювати зі спрайтами, фізикою, анімаціями та готовими компонентами. Однак для цього проєкту основний акцент зроблено на візуальній частині та швидкій перевірці механік у редакторі. Через це важливішою стала можливість використовувати візуальне налаштування логіки та швидко бачити результат змін у сцені.

Godot є зручним рушієм для 2D-ігор і має просту структуру сцен. Його часто використовують для невеликих та середніх проєктів. Водночас для «Blade of Destiny» потрібне середовище, у якому можна поєднати редактор рівнів, роботу з ассетами, візуальне програмування та інтеграцію анімацій у межах одного знайомого робочого процесу. Через це Godot не став основним варіантом для реалізації гри.

GameMaker Studio добре підходить для простіших 2D-проєктів, аркад і платформерів. Його сильна сторона — швидке створення базової логіки й рівнів. Проте «Blade of Destiny» містить багато графічних матеріалів, анімацій, ефектів, ворогів і елементів оточення. Для такого проєкту важливо було мати більше контролю над організацією ассетів, налаштуванням сцени та зв’язком між графікою і механіками [26].

Для реалізації «Blade of Destiny» обрано Unreal Engine 4. У цьому рушії є підтримка 2D-графіки через Paper2D, система візуального програмування Blueprint, редактор рівнів, засоби роботи з ассетами, анімаціями та ігровими об’єктами. Unreal Engine 4 дозволяє імпортувати графічні матеріали, налаштовувати спрайти, створювати flipbook-анімації, розміщувати об’єкти сцени та перевіряти їхню роботу в ігровому режимі [16].

Unreal Engine 4 також зручний тим, що дозволяє швидко переходити від редагування сцени до її перевірки в ігровому режимі. Для «Blade of Destiny» це особливо важливо, оскільки значна частина роботи пов’язана з візуальними матеріалами. Після зміни спрайта, анімації, колізії або положення об’єкта можна одразу перевірити, як цей елемент виглядає під час руху персонажа, атаки або зіткнення з ворогом.

Для цієї роботи важливо, що Unreal Engine 4 дає можливість поєднати різні частини розробки в одному середовищі. Наприклад, створений у Photoshop спрайт можна імпортувати в рушій, налаштувати через Paper2D, додати до персонажа, пов’язати з Blueprint-логікою та одразу перевірити в тестовій сцені. Такий процес зручний для гри, де графіка, анімації та механіки тісно пов’язані між собою.

Окремою перевагою є можливість швидкого тестування сцени. Після зміни параметрів руху, колізій, анімацій або положення об’єктів рівень можна одразу запустити й перевірити, як усе працює під час гри. Це має значення для side-scroller проєкту, де навіть невелике зміщення спрайта, неправильна колізія або невчасний запуск анімації одразу впливають на сприйняття проходження [16, 18].

## 2.2 Візуальне програмування Blueprint

Blueprint — система візуального програмування в Unreal Engine 4, яка дозволяє створювати логіку гри за допомогою вузлів (нодів), подій і зв’язків між ними. Її перевагою є те, що логіка відображається у вигляді зрозумілої схеми. Це спрощує пошук помилок і дозволяє швидше змінювати окремі елементи механіки. Якщо потрібно змінити швидкість персонажа, умову запуску анімації або реакцію

ворога на удар, це можна зробити без перебудови всієї системи. Blueprint є зручним інструментом для ігрових проєктів, оскільки більшість дій у грі пов’язані з подіями: натисканням клавіш, зіткненнями, переходами між станами персонажа або запуском анімацій. У такій системі легше простежити, що саме відбувається в конкретний момент гри та які елементи впливають один на одного [7, 8].

У проєкті «Blade of Destiny» Blueprint використовується для налаштування основної поведінки ігрових об’єктів. За його допомогою створено логіку руху персонажа, атаки, отримання шкоди, переходів між анімаціями та взаємодії з ворогами. Наприклад, коли гравець натискає кнопку атаки, через Blueprint можна запустити відповідну анімацію, додати візуальний ефект удару та перевірити зіткнення з ворогом. Окремо Blueprint важливий для поєднання механіки з візуальною частиною гри: якщо персонаж рухається, має запускатися відповідна анімація; якщо виконується удар, гравець повинен бачити сам рух атаки та ефект зіткнення; якщо герой отримує шкоду, це також має відображатися через зміну стану або анімації [17]. Я встановив, що використання Blueprint спрощує тестування ігрових механік, оскільки після зміни параметра або зв’язку між вузлами можна одразу запустити сцену й перевірити результат. Такий підхід дозволяє поєднати рух персонажа, атаки, анімації, взаємодію з ворогами та візуальні ефекти в одній зрозумілій системі .

## 2.3 Плагін Paper2D для роботи з 2D-графікою

Paper2D — це інструмент Unreal Engine 4, який використовується для роботи з 2D-графікою. Він дозволяє імпортувати спрайти, налаштовувати їх у рушії, створювати 2D-анімації та використовувати графічні елементи безпосередньо в ігровій сцені. Для проєкту «Blade of Destiny» Paper2D є важливим, оскільки гра створюється у форматі 2D side-scroller, де основними візуальними елементами є персонажі, вороги, платформи, фони, тайли та ефекти [18].

Основою роботи в Paper2D є спрайти. Через них у грі відображаються головний персонаж, вороги, предмети, елементи оточення та частини рівня. Після

імпорту зображення в Unreal Engine 4 його можна налаштувати як окремий спрайт або використати для створення набору кадрів. Це зручно для гри «Blade of Destiny», оскільки підготовлена в Adobe Photoshop графіка може бути перенесена в рушій і далі використана для побудови рівнів та налаштування персонажів [19].

Для створення анімацій у Paper2D використовуються flipbook-анімації. Вони складаються з послідовності окремих кадрів, які швидко змінюються один за одним і створюють ефект руху. У проєкті такі анімації застосовуються для відображення станів персонажа: спокою, бігу, стрибка, атаки, отримання шкоди або смерті. Також цей підхід може використовуватися для ворогів і візуальних ефектів, наприклад для ударів, спалахів або інших бойових дій.

Окрему роль у Paper2D відіграють 2D-колізії та робота з тайлами. Колізії потрібні для того, щоб персонаж міг стояти на платформах, стикатися з ворогами, отримувати шкоду або взаємодіяти з об’єктами рівня. Тайли дозволяють складати локації з окремих графічних частин, що зручно для побудови платформ, землі, стін і декоративних елементів. Мною встановлено, що Paper2D у «Blade of Destiny» допомагає не лише перенести графіку в рушій, а й зробити її частиною ігрового процесу [20].

## 2.4 Adobe Photoshop як інструмент створення графічних матеріалів

Adobe Photoshop використовується у проєкті «Blade of Destiny» як основний інструмент для створення та редагування 2D-графіки. За його допомогою готуються зображення персонажів, ворогів, текстур, фонових елементів, деталей оточення та візуальних ефектів для атак і ударів. Для 2D side-scroller гри це важливо, оскільки більша частина ігрового світу передається саме через графічні матеріали [21].

Під час створення головного персонажа важливо враховувати не тільки його зовнішній вигляд, а й те, як він буде виглядати безпосередньо в грі. У «Blade of Destiny» головним героєм є самурай, тому його силует, одяг, зброя та пози повинні бути помітними на фоні локації. Також у Photoshop можна підготувати окремі

кадри для майбутніх анімацій, щоб рух, атака або отримання шкоди виглядали зрозуміло й послідовно.

Окремо створюються графічні матеріали для ворогів та оточення. Вороги мають відрізнятися від фону й одразу сприйматися як небезпечні об’єкти. Елементи рівня, як-от платформи, земля, каміння, дерева, стіни або декоративні частини сцени, повинні поєднуватися між собою за стилем і кольорами. Важливо зберегти єдиний візуальний стиль, щоб персонажі, вороги та локації виглядали як частини одного ігрового світу.

Adobe Photoshop також використовується для створення VFX: візуальних ефектів атак, ударів, спалахів або зіткнень. Такі ефекти потрібні для того, щоб гравець бачив результат своєї дії. Наприклад, під час удару мечем має бути помітний момент атаки, а при зіткненні з ворогом — реакція на удар. Після підготовки графічні матеріали експортуються у відповідному форматі та переносяться в Unreal Engine 4 для подальшої роботи зі спрайтами й анімаціями[20, 21].

## 2.5 Система контролю версій Git

Git — це система контролю версій, яка використовується для збереження змін у проєкті та відстеження історії його розробки. У процесі створення гри постійно змінюються графічні матеріали, рівні, налаштування об’єктів, анімації та інші файли. Git дозволяє фіксувати ці зміни й за потреби повернутися до попередньої версії проєкту [22].

Для «Blade of Destiny» Git є корисним також через командний характер розробки. Коли над проєктом працює кілька учасників, важливо мати спільне місце для збереження файлів і контролю оновлень. Для цього може використовуватися віддалений репозиторій на GitHub або GitLab, де зберігається актуальна версія проєкту та історія внесених змін.

Оскільки гра містить багато графічних ассетів, анімацій і файлів Unreal Engine 4, контроль версій допомагає уникнути плутанини між різними версіями

матеріалів. Мною враховано, що під час роботи з такими файлами важливо правильно організовувати структуру проєкту, підписувати зміни та не змішувати тестові матеріали з основними. Це спрощує командну роботу й зменшує ризик втрати важливих файлів.

## 2.6 Обґрунтування вибору засобів реалізації

Для створення «Blade of Destiny» використано Unreal Engine 4, Blueprint, Paper2D, Adobe Photoshop та Git. Кожен із цих інструментів виконує окрему роль у процесі розробки. Unreal Engine 4 є основним середовищем, Blueprint відповідає за візуальне налаштування логіки, Paper2D використовується для роботи зі спрайтами та 2D-анімаціями, Adobe Photoshop застосовується для створення графічних матеріалів, а Git допомагає зберігати зміни та контролювати версії проєкту.

Вибір Unreal Engine 4 пов’язаний із тим, що рушій дозволяє працювати з рівнями, персонажами, ворогами, анімаціями та ефектами в одному середовищі. Для «Blade of Destiny» це важливо, оскільки гра має не тільки набір окремих зображень, а й повноцінну ігрову сцену з рухом, атаками, зіткненнями та перевіркою реакцій. У такому проєкті графічний матеріал швидко переходить від етапу створення до етапу тестування [16].

Blueprint обрано через зручність роботи з ігровими подіями. Рух персонажа, атака, отримання шкоди, запуск анімації та реакція ворога — усе це логічно описується через події та зв’язки між ними. Візуальна структура Blueprint допомагає швидше знаходити помилки й змінювати окремі параметри без повної перебудови механіки. Наприклад, можна змінити швидкість руху, умову запуску атаки або реакцію на зіткнення з ворогом [17].

Paper2D використовується для перенесення 2D-графіки в Unreal Engine 4. Через нього налаштовуються спрайти, flipbook-анімації, тайли та 2D-колізії. Для гри обраного жанру це один із ключових інструментів, оскільки персонажі, вороги, рівні та бойові ефекти існують саме як 2D-елементи. Paper2D дозволяє не просто імпортувати зображення, а перетворити їх на частину ігрової сцени [18-20].

Adobe Photoshop використано для створення та редагування графічних матеріалів. У ньому підготовлено персонажів, ворогів, елементи оточення, фони, текстури та ефекти атак. Photoshop зручний для роботи з шарами, прозорістю та окремими кадрами анімацій. Для «Blade of Destiny» це важливо, бо більша частина стилю гри формується саме через 2D-зображення [21].

Git застосовано для організації роботи з файлами проєкту. Під час розробки змінювалися графічні матеріали, анімації, рівні, Blueprint-класи та налаштування рушія. Контроль версій дає можливість зберігати історію змін, повертатися до попередніх варіантів і підтримувати порядок у командній роботі. Для гри з великою кількістю ассетів це зменшує ризик втрати або випадкового перезапису важливих файлів [22].

Обраний набір засобів добре підходить для проєкту з акцентом на візуальну частину. Спочатку графіка створюється в Adobe Photoshop, потім переноситься в Unreal Engine 4, налаштовується через Paper2D, пов’язується з логікою Blueprint і перевіряється в ігровій сцені. Такий ланцюжок роботи дозволяє послідовно переходити від створення ассетів до їхнього практичного використання в грі.

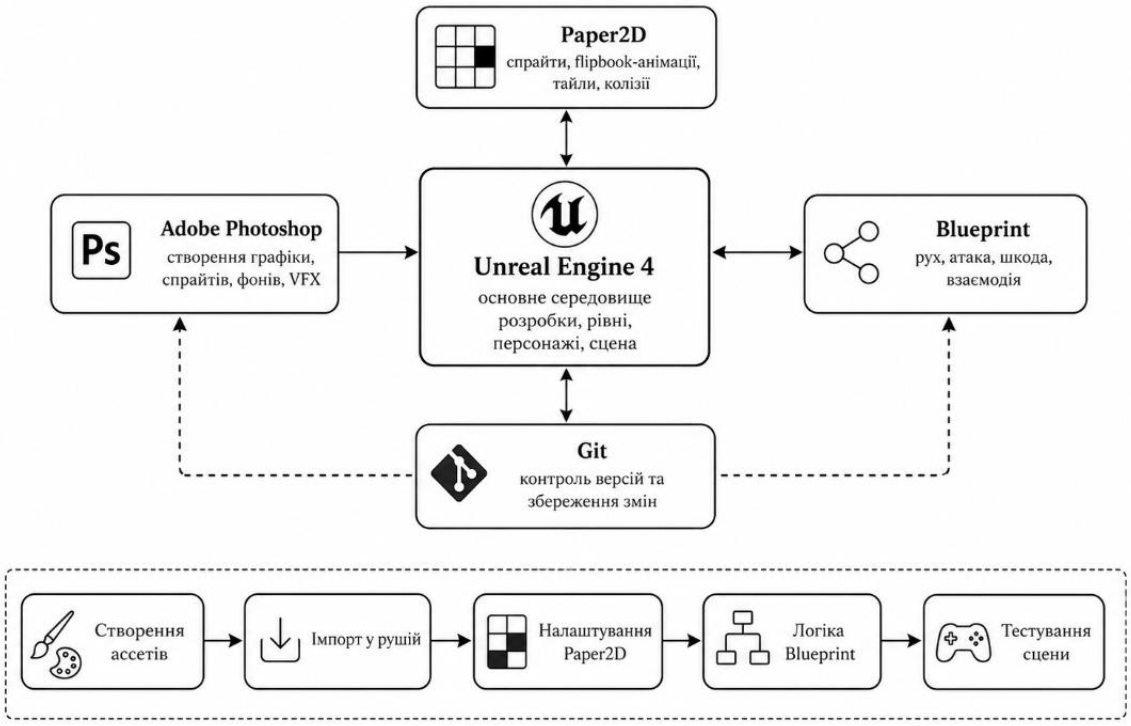



Рис. 2.1 Схема взаємодії засобів реалізації у проєкті «Blade of Destiny»

# 3 ПРОЄКТУВАННЯ ТА РОЗРОБКА ВІЗУАЛЬНОЇ ЧАСТИНИ ГРИ «BLADE OF DESTINY»

У цьому розділі розглянуто процес проєктування та створення візуальної частини гри «Blade of Destiny». Основну увагу приділено загальній концепції гри, формуванню стилю, створенню головного персонажа, ворогів та підготовці графічних матеріалів для подальшої інтеграції в Unreal Engine 4. Для цього проєкту візуальна складова є важливою не лише як елемент оформлення, а і як частина ігрового процесу, що допомагає гравцеві краще сприймати сцену, рух персонажа та бойові дії [12, 15].

Розробка гри складається з кількох послідовних етапів, які пов’язані між собою. Спочатку формується ідея проєкту, після чого визначається структура рівнів, стиль персонажів, основні механіки та спосіб взаємодії гравця з ігровим світом. Далі створені матеріали переходять до етапу реалізації, тестування та доопрацювання.

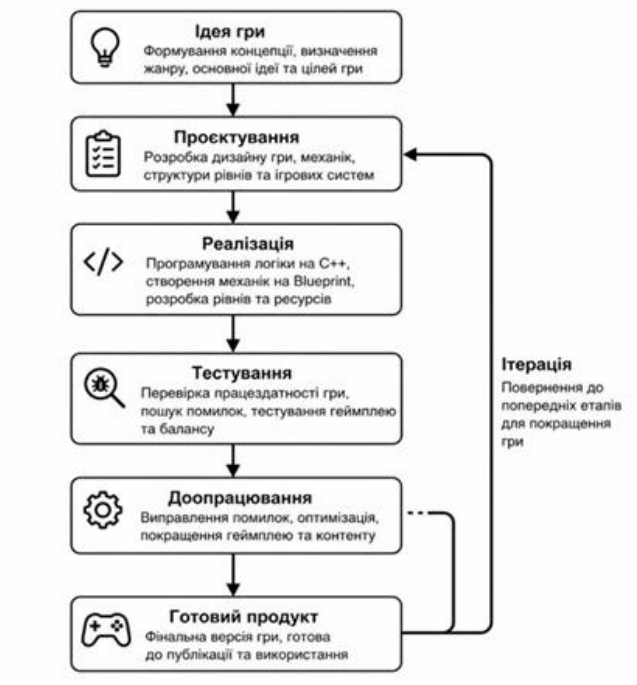



Рис. 3.1 Етапи розробки гри «Blade of Destiny»

Як показано на рисунку 3.1, процес розробки гри не є повністю лінійним. Після тестування окремі елементи можуть повертатися на етап доопрацювання: це стосується графіки, анімацій, рівнів, бойових ефектів і загального балансу гри. Такий підхід дозволяє поступово покращувати проєкт і перевіряти, як створені візуальні матеріали працюють безпосередньо в ігровій сцені.

## 3.1 Загальна концепція гри Blade of Destiny

«Blade of Destiny» — це 2D side-scroller гра, у центрі якої знаходиться самурай, що проходить небезпечний шлях у пошуках власного призначення. Гравець керує героєм, рухається локаціями, долає перешкоди, вступає в бій із ворогами та поступово просувається вперед. Гра існує як пригодницький бойовий проєкт, де основну роль відіграють рух, атака мечем, ворожі істоти та атмосфера небезпечного світу.

Сюжетна основа пов’язана з образом самотнього воїна, який втратив рідні землі й продовжує шлях через ворожі території. Такий образ добре працює в межах side-scroller жанру, оскільки гравець бачить персонажа збоку, чітко спостерігає за його рухом і розуміє напрям дії. Самурай як головний герой також природно пояснює наявність ближнього бою, ударів мечем, бойових анімацій і візуальних ефектів атак.

Світ гри має передавати відчуття небезпеки. У ньому присутні темні ліси, кам’янисті ділянки, печери, монстри та інші загрози. Локації не виконують лише декоративну роль: вони створюють настрій і водночас впливають на проходження. Наприклад, платформи визначають шлях персонажа, вороги змушують реагувати на небезпеку, а фон підтримує загальну атмосферу без зайвого відволікання.

Під час розробки концепції обрано підхід, у якому сюжет, графіка й механіки пов’язані між собою. Якщо герой є самураєм, то його силует, зброя, анімації та атаки мають одразу підтримувати цей образ. Якщо світ подається як небезпечний, то вороги, фони й ефекти повинні підсилювати це відчуття. Через це візуальна

частина не створювалася окремо від геймплею, а підбиралася під конкретні дії персонажа й структуру рівнів.

У грі передбачено базові механіки, характерні для side-scroller проєкту: рух вліво та вправо, стрибок, атака, отримання шкоди, взаємодія з ворогами та проходження рівня. Ці дії прості для розуміння, але потребують точного візуального супроводу. Коли персонаж атакує, гравець має бачити момент удару; коли герой отримує шкоду, повинна з’являтися зрозуміла реакція; коли персонаж рухається, його анімація має відповідати швидкості переміщення.

Загальна концепція гри також впливає на темп проходження. Оскільки головний герой є воїном ближнього бою, геймплей не може будуватися лише на стрибках між платформами. Важливо, щоб гравець регулярно стикався з ворогами, використовував атаку мечем і бачив результат своїх дій. Через це бойові елементи, анімації та ефекти ударів стають частиною загального сприйняття гри, а не окремим декоративним доповненням.

Окрему увагу приділено тому, щоб концепція була зрозумілою вже з перших сцен. Гравець має одразу бачити, хто є головним героєм, у якому середовищі він перебуває і які об’єкти становлять небезпеку. Для цього використано образ самурая, ворожих істот, темне оточення та бойові ефекти. Візуальна частина в такому випадку не пояснює сюжет текстом, а передає його через вигляд персонажа, локацій і ворогів.

## 3.2 Формування візуального стилю гри

Візуальний стиль «Blade of Destiny» сформовано навколо образу самурая, небезпечного світу та бойових сцен. Для гри обрано стилізацію, яка не перевантажує сцену деталями, але дозволяє передати атмосферу подорожі через ворожі локації. У 2D side-scroller грі важливо, щоб гравець швидко розумів, де знаходиться персонаж, які об’єкти є небезпечними та куди потрібно рухатися [12].

Під час роботи над стилем враховано різницю між головними та другорядними елементами сцени. Персонаж і вороги мають бути помітними,

платформи — зрозумілими для пересування, а фон — підтримувати атмосферу, не забираючи на себе всю увагу. Якщо задній план занадто яскравий або деталізований, він може заважати побачити ворога чи межі платформи. Через це для фонів доцільно використовувати стриманіші відтінки, а важливі об’єкти робити контрастнішими [12].

Кольорова палітра гри побудована на природних і приглушених відтінках. Для лісових і печерних локацій підходять темно-зелені, коричневі, сірі та синюваті кольори. Водночас ефекти ударів, спалахи та траєкторії атак можуть бути світлішими або контрастнішими. Це допомагає швидко виділяти бойові дії на фоні рівня. У результаті гравець не витрачає зайвий час на пошук важливих подій на екрані [15].

Під час формування стилю також враховано різницю між фоновими та активними об’єктами. Фон має створювати настрій, але не повинен заважати гравцеві бачити персонажа й ворогів. Активні елементи, навпаки, повинні мати чіткіші межі та помітніший силует. Такий поділ допомагає уникнути ситуацій, коли декоративні об’єкти сприймаються як платформи або важливі елементи рівня губляться серед деталей.

Важливим рішенням стало збереження одного рівня стилізації для персонажа, ворогів і оточення. Якщо головний герой виконаний у одному стилі, а вороги або фон виглядають зовсім інакше, сцена починає сприйматися розірвано. Тому під час створення графіки враховувалися форма об’єктів, товщина контурів, деталізація та загальний настрій локації. Це стосується не лише великих об’єктів, а й дрібних деталей: платформ, тайлів, прапорів, каміння, стін і декоративних елементів.

Стиль ворогів підібрано так, щоб вони відрізнялися від головного героя, але залишалися частиною того самого світу. Самурай має виглядати впорядковано й зрозуміло, тоді як вороги можуть мати більш агресивні форми. При цьому їх не варто робити надто деталізованими, бо під час бою гравець має швидко впізнавати супротивника й оцінювати небезпеку.

Окремо враховано візуальний стиль атак. Удар мечем не повинен губитися серед інших елементів сцени. Для цього використано ефекти траєкторії удару, короткі спалахи та реакції на зіткнення. Вони не мають тривати занадто довго та перекривати персонажа, але повинні достатньо чітко показувати момент атаки. Такий підхід робить бойові сцени більш зрозумілими без додаткових пояснень [6, 13].

### 3.3 Розробка 2D-графіки персонажів

Розробку головного персонажа розпочато з визначення його ролі у грі. У «Blade of Destiny» гравець керує самураєм, тому зовнішній вигляд героя має одразу показувати, що це воїн ближнього бою. Для персонажа обрано виразний силует, зброю та характерну поставу, щоб він залишався помітним під час руху, стрибків і бойових сцен.

Під час створення графіки враховано, що персонаж у грі постійно змінює стан. Він стоїть, біжить, стрибає, атакує, отримує шкоду або гине, тому його не можна розглядати лише як одне статичне зображення. Поза героя, розміщення меча, нахил корпусу та положення ніг підготовлено з урахуванням подальшої анімації. Для атаки передбачено кілька послідовних кадрів: підготовка до удару, сам удар і повернення до звичайного стану.

Окрему увагу приділено читабельності рухів. Якщо атака не має зрозумілого замаху, гравець гірше відчуває момент удару. Якщо біг виглядає занадто різко, персонаж сприймається неприродно. Тому кадри анімації підготовлено так, щоб кожен наступний рух логічно продовжував попередній, а дія персонажа легко сприймалася під час проходження рівня.

Під час роботи над героєм збережено баланс між деталізацією та простотою. Надто дрібні елементи одягу або спорядження майже не видно в реальному розмірі гри, особливо під час руху. Через це акцент зроблено на силуеті, мечі та основних деталях образу, які гравець може швидко помітити на екрані. Такий підхід

допомагає краще бачити персонажа на фоні локації та зручніше використовувати його в анімаціях.

Після підготовки графічних матеріалів персонажа їх перенесено в Unreal Engine 4. У рушії вони використовуються як спрайти, на основі яких створено анімації руху, атаки та інших станів. На цьому етапі перевірено, як герой виглядає безпосередньо в ігровій сцені: чи не зливається з фоном, чи правильно читається атака, чи не заважає розмір спрайта проходженню рівня [6, 19].

Під час підготовки персонажа враховано, що його силует має залишатися зрозумілим у різних станах. У стані спокою гравець бачить базовий образ героя, під час бігу — напрям руху, а під час атаки — положення меча та момент удару. Якщо в різних кадрах персонаж виглядає занадто по-різному, анімація може сприйматися як набір окремих зображень. Тому для героя важливо зберегти однакову основу форми, навіть коли змінюється поза або дія [20].

### 3.4 Розробка графіки ворогів

Ворогів у «Blade of Destiny» розроблено так, щоб вони відрізнялися від головного героя не лише зовнішнім виглядом, а й загальним сприйняттям. Самурай є керованим персонажем, тому його образ залишається впорядкованим і зрозумілим. Вороги, навпаки, мають створювати відчуття небезпеки. Для них обрано стиль монстрів і ворожих істот, які належать до світу, через який проходить герой.

Під час створення ворогів враховано, що гравець повинен швидко розуміти: перед ним не частина фону, а небезпечний об’єкт. Для цього використано виразні форми, окремий силует і помітні деталі. Якщо ворог знаходиться на темному фоні лісу або печери, його контур має залишатися читабельним. Це впливає на геймплей, оскільки гравець має вчасно зреагувати: атакувати, відійти або уникнути зіткнення.

Для ворогів також підготовлено анімаційні стани. Навіть проста істота повинна мати очікування, рух, атаку, отримання шкоди та зникнення або смерть. Без таких станів бій виглядає слабшим, а взаємодія з ворогом сприймається менш

зрозуміло. Наприклад, після удару мечем супротивник має показати реакцію: зміну кадру, короткий відступ або інший візуальний сигнал.

Мною враховано, що дизайн ворогів не повинен бути складнішим за дизайн головного героя. Надмірна деталізація може забирати зайву увагу або порушувати загальний стиль сцени. Тому графіку ворогів створено так, щоб вона підтримувала атмосферу небезпечного світу, але залишалася простою для сприйняття під час руху, атаки та зіткнення з персонажем [13, 15].

## 3.5 Створення елементів оточення та рівнів

Під час створення оточення для «Blade of Destiny» увагу приділено не лише зовнішньому вигляду локацій, а й тому, як вони впливають на проходження. У грі присутні платформи, фони, декоративні об’єкти, елементи лісу, печерні ділянки, кам’яні поверхні та тайли. Кожен із цих елементів виконує свою роль: одні формують шлях гравця, інші підтримують атмосферу, а частина використовується для візуального наповнення сцени.

Рівні створюються у форматі side-scroller, тому гравець рухається переважно в одному напрямку. Через це структура локації має бути зрозумілою: платформи показують шлях, перешкоди створюють виклик, а вороги додають небезпеку. Під час побудови рівнів важливо, щоб декоративні елементи не сприймалися як активні об’єкти. Якщо дерево, камінь або фрагмент фону виглядає занадто схожим на платформу, гравець може неправильно оцінити сцену.

Для створення рівнів використано тайловий підхід. Окремі графічні фрагменти застосовуються для побудови землі, стін, платформ, переходів і декоративних частин. Це дозволяє збирати локацію з повторюваних елементів, не створюючи кожну ділянку з нуля. При правильному використанні тайли не виглядають одноманітно, оскільки їх можна комбінувати з фоновими об’єктами, декоративними деталями та зміною висоти платформ.

Використання тайлів спрощує побудову рівнів, але потребує уважності до повторюваності елементів. Якщо однакові фрагменти занадто часто розміщуються

поруч, локація може виглядати штучно. Щоб уникнути цього, тайли доцільно комбінувати з декоративними об’єктами, зміною висоти платформ, фоновими деталями та різними переходами між частинами рівня. Це дозволяє зберегти єдиний стиль, але зробити сцену менш одноманітною.

Фони виконують роль атмосферного шару. Вони задають настрій локації, але не повинні конкурувати з персонажем і ворогами. Для цього задній план зроблено менш активним, ніж основні об’єкти сцени. У лісових або печерних ділянках фон може містити дерева, камені, темні проходи або елементи руїн, але важливі частини рівня мають залишатися помітними.

Платформи й поверхні, по яких рухається персонаж, потребують окремої уваги. Їхній вигляд має одразу показувати, що це активна частина рівня. Гравець повинен розуміти, де можна стояти, куди можна стрибнути і яка ділянка є небезпечною. Через це платформи мають бути візуально чіткішими за фон і мати зрозумілу форму.

Декоративні об’єкти використовуються для того, щоб рівні не виглядали порожніми. Це можуть бути прапори, каміння, елементи стін, зруйновані частини споруд, рослинність або інші деталі. Їх кількість потрібно контролювати, бо надмірна кількість дрібних об’єктів ускладнює сприйняття сцени. У грі з бойовими ситуаціями гравець має насамперед бачити персонажа, ворогів і напрям руху [5, 18, 19].

## 3.6 Підготовка ассетів до імпорту в Unreal Engine 4

Перед перенесенням графіки в Unreal Engine 4 ассети підготовлено у форматі, зручному для подальшої роботи зі спрайтами. Основні графічні матеріали збережено у форматі PNG, оскільки він підтримує прозорий фон і добре підходить для 2D-об’єктів. Це особливо важливо для персонажів, ворогів, ефектів ударів та окремих елементів оточення, які не повинні мати зайвого фону навколо себе.

Під час підготовки спрайтів враховано їхній розмір і майбутнє використання в грі. Занадто великі зображення можуть перевантажувати проєкт, а занадто малі — втрачати якість під час масштабування. Тому графічні матеріали підготовлено так, щоб вони відповідали розміру сцени та добре виглядали в реальному ігровому масштабі.

Підготовка ассетів також включає перевірку зайвого прозорого простору навколо спрайта. Якщо його занадто багато, об’єкт може неправильно розміщуватися в сцені або створювати незручності під час налаштування колізій. Для персонажа й ворогів це особливо помітно, оскільки їхнє положення має збігатися з платформами та зоною взаємодії. Тому перед імпортом потрібно перевіряти не тільки саме зображення, а й межі кадру.

Для анімацій персонажів та ворогів кадри розділено на окремі зображення або підготовлено у вигляді послідовності. Це спростило створення flipbook-анімацій у Paper2D. Наприклад, для атаки потрібно кілька кадрів, які показують початок руху, момент удару та повернення персонажа до звичайного стану. Якщо кадри підготовлено нерівномірно або вони мають різні відступи, анімація може виглядати різкою, тому розміщення персонажа в кожному кадрі перевірено окремо [19, 21].

Також виконано базову оптимізацію ассетів. З графічних файлів прибрано зайві частини, перевірено прозорість, назви матеріалів і структуру папок. Така підготовка допомагає уникнути плутанини під час імпорту та подальшої роботи.

## 3.7 Інтеграція графічних матеріалів у рушій

Після підготовки графічні матеріали було перенесено в Unreal Engine 4. Імпортовані зображення організовано у відповідних папках, щоб окремо зберігати персонажа, ворогів, фони, елементи рівнів, ефекти та анімаційні кадри. Така структура проєкту спрощує пошук потрібних матеріалів і зменшує ризик випадкового змішування тестових та основних ассетів.

У Paper2D імпортовані зображення налаштовано як спрайти. Для окремих об’єктів перевірено масштаб, положення точки прив’язки та відображення в сцені. Це важливо для правильного розміщення персонажа на платформі, налаштування ворогів і використання елементів рівня. Якщо точка прив’язки встановлена неправильно, об’єкт може зміщуватися під час анімації або некоректно розташовуватися у сцені.

Після імпорту спрайтів важливо перевірити їхній масштаб відносно персонажа, ворогів і платформ. Зображення може виглядати якісно окремо, але в сцені бути занадто великим або малим. Через це під час інтеграції необхідно перевіряти, як об’єкти співвідносяться між собою: чи не перекриває ворог платформу, чи не виглядає персонаж занадто дрібним, чи достатньо помітні ефекти атаки.

Для анімованих об’єктів створено flipbook-анімації. Послідовність кадрів налаштовано так, щоб рух персонажа, атака, отримання шкоди або смерть виглядали зрозуміло. Нижче, на рисунку 3.2, наведений приклад такої анімації.

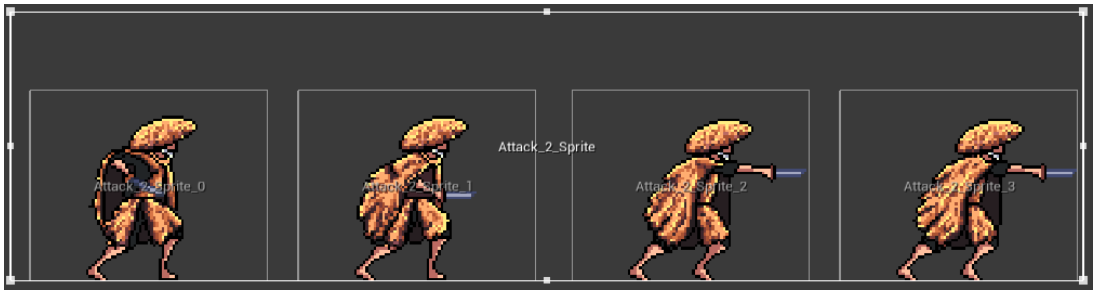



Рис 3.2 Flipbook-анімація атаки

Після створення анімацій їх перевірено безпосередньо в ігровій сцені, оскільки в редакторі окрема анімація може виглядати правильно, але під час руху персонажа або бою потребувати корекції.

Окрему увагу приділено колізіям. Для платформ і об’єктів, з якими взаємодіє персонаж, налаштовано межі зіткнення. Це дозволяє герою стояти на поверхнях, не провалюватися крізь рівень і коректно реагувати на ворогів або перешкоди. Мною враховано, що інтеграція графіки в рушій не завершується самим імпортом файлів: кожен з ассетів має правильно працювати в умовах реального проходження рівня [18, 19].

### 3.8 Створення та інтеграція анімацій персонажів

Для головного персонажа «Blade of Destiny» підготовлено набір основних анімаційних станів. До них належать idle, run, jump, attack, hit і death. Кожна з цих анімацій виконує окрему роль у грі й допомагає гравцеві зрозуміти, що саме відбувається з персонажем у конкретний момент.

Під час створення анімацій важливо враховувати не лише окремі кадри, а й момент переходу між станами. Наприклад, персонаж може стояти, потім почати біг, після цього виконати атаку або стрибок. Якщо перехід між цими діями різкий, рух виглядає неприродно. Тому анімації мають не просто відтворюватися окремо, а правильно змінювати одна одну відповідно до дій гравця.

Анімація idle використовується тоді, коли персонаж стоїть без руху. Вона потрібна для того, щоб герой не виглядав статичним. Run показує переміщення персонажа рівнем і має відповідати швидкості руху, інакше виникає відчуття ковзання. Для jump важливо показати відрив від поверхні та положення героя в повітрі, щоб гравець краще відчував стрибок.

Бойові анімації мають особливе значення. Attack показує рух меча та момент удару, тому кадри цієї анімації підготовлено так, щоб дія була помітною навіть під час швидкого бою. Hit використовується для реакції на отримання шкоди, а death позначає завершення стану персонажа після втрати здоров’я. Такі анімації роблять

взаємодію з ворогами зрозумілішою та допомагають краще сприймати наслідки бойових дій [20].

Після підготовки кадрів анімації інтегровано в Unreal Engine 4 через Paper2D. Для цього створено flipbook-анімації та налаштовано їхній запуск залежно від стану персонажа. Наприклад, під час руху активується run, під час атаки — attack, а після отримання шкоди — hit.

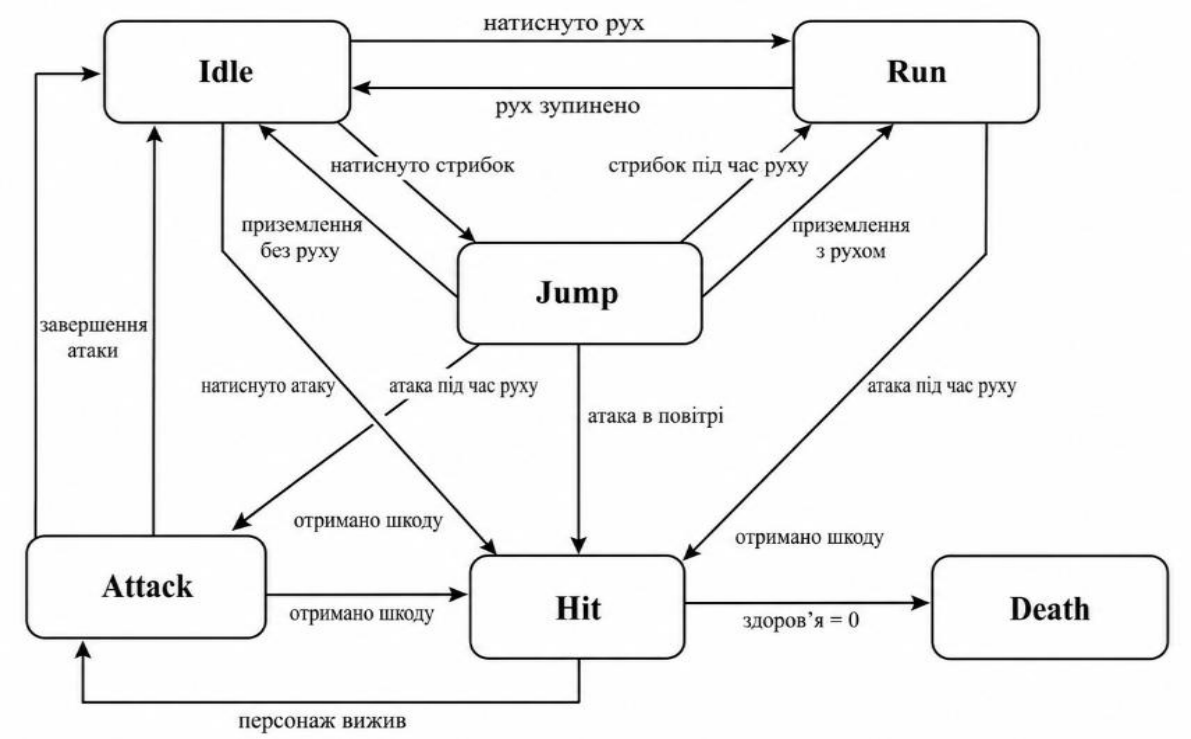



Рис. 3.2 Основні анімації персонажа у грі «Blade of Destiny»

## 3.9 Створення візуальних ефектів для атак та ударів

Візуальні ефекти у «Blade of Destiny» створено для того, щоб бойові дії краще сприймалися під час гри. Оскільки головний персонаж використовує меч, момент удару має бути помітним не лише через анімацію руки чи зброї, а й через додатковий графічний ефект. Для цього підготовлено слеші, короткі спалахи та ефекти зіткнення, які з'являються під час атаки або контакту з ворогом.

Слеш(візуальний ефект від удару мечем) використовується для показу напрямку удару мечем. Він допомагає гравцеві швидше зрозуміти, у який бік

виконано атаку та яку зону вона охоплює. Спалахи й ефекти зіткнення додаються в момент контакту з ворогом, щоб дія не виглядала порожньою. У багатьох 2D-іграх такий візуальний відгук використовується для того, щоб гравець одразу бачив результат натискання кнопки атаки.

Ефект удару не повинен бути занадто довгим. Якщо він залишається на екрані після завершення атаки, гравець може неправильно оцінити момент нанесення шкоди. Для бойової сцени важливо, щоб VFX з'являвся саме в момент удару й швидко зникав після нього. Так ефект підсилює дію, але не заважає подальшому руху персонажа або сприйняттю ворога.

Під час створення VFX враховано загальний стиль гри. Ефекти зроблено помітнішими за фон, але без надмірної деталізації. Якщо ефект занадто великий або триває довго, він може перекривати персонажа, ворога чи платформу. Через це для атак використано короткі візуальні елементи, які швидко з'являються і не заважають подальшому руху персонажа.

Візуальні ефекти пов'язані з ігровими механіками. Коли персонаж атакує, запускається відповідна анімація і з'являється ефект удару(рис 3.3). Якщо ворог отримує пошкодження, гравець бачить реакцію через зміну стану, короткий спалах або інший візуальний сигнал. Завдяки цьому бойові сцени у «Blade of Destiny» виглядають зрозуміліше, а удари мечем мають більш відчутний результат [6, 13,20].

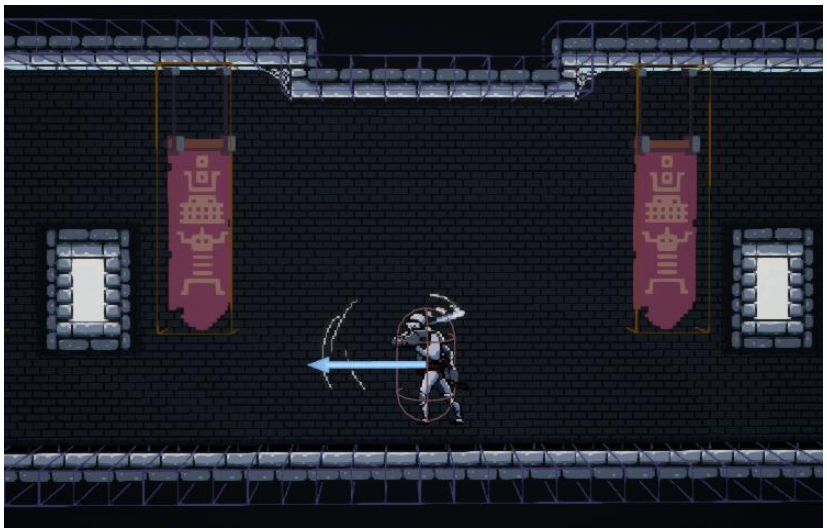


Рис. 3.3 Узгодження образу ворога з оточенням та відображення візуального ефекту траєкторії удару мечем

# 4 РЕАЛІЗАЦІЯ, ОПТИМІЗАЦІЯ ТА ТЕСТУВАННЯ ГРИ

## 4.1 Організація проєкту в Unreal Engine 4

Проєкт «Blade of Destiny» в Unreal Engine 4 організовано за окремими групами матеріалів. Такий порядок потрібний для зручної роботи з великою кількістю файлів: персонажами, ворогами, рівнями, текстурами, анімаціями, ефектами та Blueprint-класами. Якщо всі матеріали зберігати в одній папці, пошук потрібного спрайта або анімації займає більше часу, особливо під час тестування рівнів [16, 22].

У структурі проєкту передбачено окремі папки для головного персонажа, ворогів, локацій, фонів, тайлів, візуальних ефектів та анімацій. Наприклад, матеріали персонажа не змішуються з графікою ворогів, а ефекти атак зберігаються окремо від елементів оточення. Такий поділ допомагає швидше замінити потрібний ассет, додати нову анімацію або перевірити окремий елемент сцени.

Під час організації проєкту важливо також дотримуватися зрозумілої системи назв. Назва файлу має показувати, до якого об’єкта він належить і яку функцію виконує. Наприклад, для персонажа доцільно використовувати назви, пов’язані зі станами руху або атаки, а для ефектів — із дією, яку вони підсилюють. Це полегшує пошук матеріалів і зменшує ризик випадково використати не той спрайт або анімацію.

Blueprint-класи також розміщено окремо від графічних матеріалів. Вони відповідають за поведінку персонажа, ворогів та об’єктів рівня. Через них налаштовано рух, атаку, отримання шкоди, запуск анімацій та реакцію на зіткнення. Така структура дозволяє не змішувати логіку гри з візуальними ресурсами [17].

Під час організації проєкту враховано, що «Blade of Destiny» містить багато 2D-ассетів. Для них важливо мати зрозумілі назви та логічне розміщення в папках.

Це спрощує подальшу роботу в Paper2D, де спрайти, flipbook-анімації та елементи рівнів використовуються разом у сцені.

## 4.2 Реалізація базових ігрових механік

У «Blade of Destiny» гравець керує самураєм, який переміщується рівнем, стрибає між платформами, атакує ворогів і реагує на небезпеки. Для side-scroller гри ці механіки є основою проходження, тому під час реалізації важливо було зробити керування зрозумілим і достатньо швидким. Рух персонажа налаштовано вліво та вправо, а швидкість підібрано так, щоб герой не виглядав повільним, але водночас не втрачав керованість під час бою або стрибків.

Стрибок реалізовано як окрему дію, яка використовується для подолання перешкод і переходу між платформами. Під час тестування перевірялася висота стрибка, точність приземлення та поведінка персонажа біля країв платформи. Для такої гри важливо, щоб гравець не відчував, що персонаж «ковзає» або погано реагує на натискання клавіш, тому рух і стрибок налаштовувалися разом із перевіркою рівнів [6, 17].

Бойова частина побудована навколо атаки мечем. Після натискання кнопки атаки запускається відповідна анімація, а в момент удару з’являється візуальний ефект траєкторії меча. Для взаємодії з ворогами налаштовано перевірку зіткнення, нанесення шкоди та реакцію супротивника. Якщо ворог торкається персонажа або атакує його, герой переходить у стан отримання шкоди.

Проходження рівня реалізовано як послідовний рух локацією. Гравець зустрічає ворогів, долає перешкоди, використовує стрибки й атаки для просування вперед. Під час тестування перевірялося, чи не заважають платформи руху, чи помітні вороги на фоні, чи правильно спрацьовують зіткнення та чи не виникають ситуації, у яких гравець не розуміє, що потрібно робити далі. Нижче, на рисунку 4.1, наведено узагальнюючу схему базових механік гри.

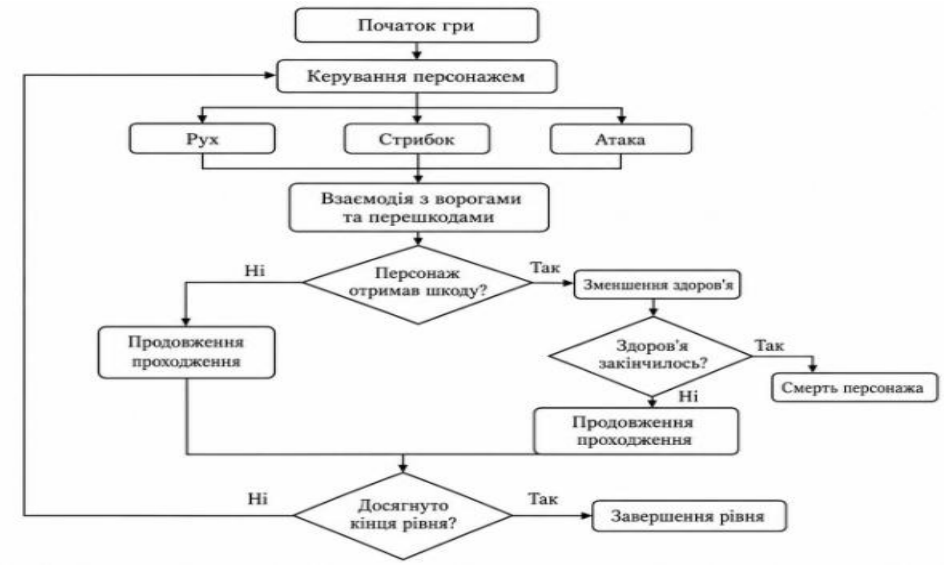



Рис. 4.1 Загальна схема базових ігрових механік

## 4.3 Інтеграція візуальної частини з ігровими механіками

Кожна основна дія персонажа у грі має своє візуальне відображення. Коли герой стоїть без руху, використовується анімація спокою; під час переміщення запускається біг; при стрибку змінюється положення персонажа в повітрі. Це потрібно не лише для зовнішнього вигляду, а й для зрозумілого сприйняття стану героя під час проходження [6].

Для бойових дій візуальна частина має особливе значення. Атака мечем супроводжується анімацією удару та ефектом траєкторії меча. Якщо удар досягає ворога, на екрані має з’явитися реакція: зміна стану супротивника, короткий ефект або анімація отримання шкоди. Так гравець бачить зв’язок між натисканням кнопки, рухом персонажа й результатом дії [13, 17].

Окремо налаштовано переходи між станами персонажа. Після зупинки руху герой повертається до стану спокою, після атаки — до попередньої дії, а після отримання шкоди або продовжує гру, або переходить до стану смерті, якщо здоров’я закінчується. У 2D side-scroller грі такі переходи добре помітні, тому для них перевірялися затримки, порядок запуску анімацій і правильність відображення кадрів.

Під час налаштування бойових дій важливо синхронізувати три елементи: механіку атаки, анімацію персонажа та візуальний ефект. Якщо шкода наноситься раніше, ніж гравець бачить удар, дія сприймається неточно. Якщо ефект з’являється із запізненням, атака виглядає слабшою. Тому перевірка бойових сцен має виконуватися не лише за технічною працездатністю, а й за тим, наскільки природно виглядає послідовність дій.

Для перевірки роботи графіки разом із механіками використовувалися тестові сцени в Unreal Engine 4. У них перевірялося, чи не перекриває ефект удару персонажа, чи не зливається ворог із фоном, чи правильно запускаються анімації руху, атаки та отримання шкоди. Особливої уваги потребували бойові моменти, де одночасно працюють переміщення, зіткнення, анімація та VFX [20].

### 4.4 Оптимізація графіки під Unreal Engine 4

Щоб сцена стабільно працювала в Unreal Engine 4, графічні матеріали підготовлено з урахуванням їхнього розміру, формату та подальшого використання в рушії. Для більшості 2D-ассетів використано формат PNG, оскільки він підтримує прозорість і підходить для персонажів, ворогів, ефектів, тайлів та окремих елементів оточення. Перед імпортом перевірено, чи немає зайвого фону, неправильних країв або непотрібних частин зображення [19, 21].

Розміри текстур підібрано відповідно до масштабу гри. Занадто великі зображення збільшують обсяг проєкту, хоча в сцені можуть не давати помітної різниці в якості. Занадто малі ассети, навпаки, втрачають чіткість при збільшенні. Тому графіку підготовлено так, щоб вона нормально виглядала в реальному ігровому розмірі та не створювала зайвого навантаження [14, 16].

Для кадрів анімації перевірено однакове положення персонажа в межах зображення. Якщо в одному кадрі герой розташований вище, а в іншому нижче, під час відтворення анімація може смикатися. Через це перед створенням flipbook-анімацій спрайти впорядковано, зайві прозорі поля зменшено, а назви файлів зроблено зрозумілими для подальшої роботи.

Кількість декоративних елементів у сцені також потребувала контролю. Фони, тайли, платформи й ефекти використовуються так, щоб рівень не виглядав порожнім, але й не перевантажувався деталями. Частину графічних елементів застосовано повторно, особливо для платформ, стін, підлоги та фрагментів оточення. Це дало змогу не збільшувати кількість зайвих ресурсів у проєкті та зберегти єдиний стиль локацій.

Окремо в грі реалізовано меню налаштувань, у якому користувач може змінювати режим екрана, роздільну здатність, вертикальну синхронізацію, якість текстур та мову інтерфейсу. Для графіки передбачено один повзунок Texture Quality, який відповідає за загальний рівень якості відображення текстур у грі. При зменшенні цього параметра зображення може виглядати трохи простішим або менш чітким, однак структура рівнів, персонажі, вороги та основні елементи сцени залишаються тими самими.

Оскільки для гри не створювалися окремі набори текстур низької якості, оптимізація зосереджена на правильній підготовці основних ассетів. Це включає вибір відповідного розміру зображень, уникнення зайвого прозорого простору, повторне використання елементів оточення та контроль кількості декоративних об’єктів у сцені. Такий підхід дозволяє не ускладнювати структуру проєкту дублями файлів, але зберігати прийнятну якість відображення.

Важливо зазначити, що для проєкту не створювалися окремі набори текстур низької якості. Це рішення пов’язане з тим, що гра використовує 2D-ассети, а дублювання всіх спрайтів у кількох варіантах значно збільшило б обсяг проєкту та ускладнило б організацію файлів. Тому повзунок якості текстур, який показано на рис. 4.2, використано як базове графічне налаштування, яке впливає на загальне відображення сцени без заміни основних ассетів.

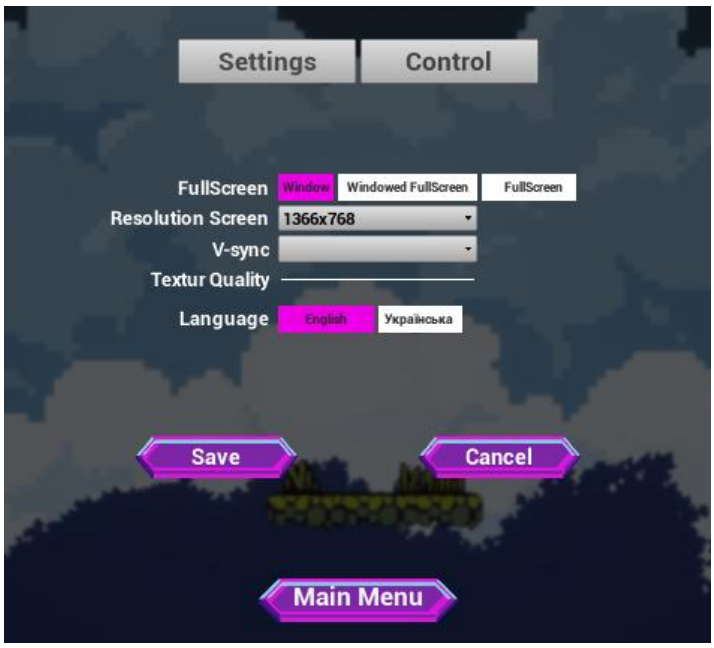



Рис. 4.2 Меню налаштування гри «Blade of Destiny»

## 4.5 Тестування ігрового процесу

Після налаштування основних механік гру перевірено в ігровому режимі Unreal Engine 4. Тестування проводилося не лише для пошуку технічних помилок, а й для перевірки загального відчуття від проходження. Для 2D side-scroller гри важливо, щоб персонаж реагував на дії гравця без помітної затримки, правильно взаємодіяв із платформами та ворогами, а бойові дії виглядали зрозуміло [1, 2].

Перевірку виконано безпосередньо під час проходження тестових ділянок рівня. Такий підхід дозволяє краще побачити помилки, які не завжди помітні в редакторі. Наприклад, спрайт може виглядати правильно в окремому перегляді, але зміщуватися під час анімації; колізія платформи може бути налаштована, але персонаж все одно зачіпає край або приземляється неточно; ефект удару може виглядати красиво, але перекривати ворога під час бою.

Окремо перевірено роботу анімацій персонажа. Аналізувалися стани спокою, бігу, стрибка, атаки, отримання шкоди та смерті. Під час тестування важливо було переконатися, що анімації запускаються в потрібний момент і не конфліктують між собою. Наприклад, після завершення атаки персонаж має повернутися до

звичайного стану, а під час руху анімація бігу повинна відповідати фактичній швидкості переміщення.

Колізії перевірялися на платформах, стінах, ворогах і небезпечних ділянках рівня. Персонаж не повинен провалюватися крізь поверхні, застрягати на краях або отримувати шкоду без реального контакту з ворогом. Для атак перевірено, чи правильно визначається зіткнення з ворогом, чи збігається момент нанесення шкоди з анімацією удару, чи з'являється візуальна реакція після атаки [17, 18].

Окремо перевірялися ситуації на межах платформ і біля ворогів. Такі місця часто виявляють неточності, які непомітні під час простого руху по рівній поверхні. Персонаж може правильно стояти на платформі, але некоректно реагувати на її край. Так само атака може працювати в простій ситуації, але потребувати уточнення, якщо ворог рухається або знаходиться близько до перешкоди. Тому тестування виконувалося в різних положеннях персонажа на рівні.

Правильність відображення спрайтів перевірялася для головного персонажа, ворогів, фону, платформ і візуальних ефектів. Під час руху спрайти не мають зміщуватися, втрачати чіткість або зливатися з фоном. Особливо це стосується бойових сцен, де на екрані одночасно є персонаж, ворог, платформа та ефект удару. Якщо ефект траєкторії меча перекриває занадто велику частину сцени або з'являється не в той момент, атака виглядає неправильно.

Також протестовано меню налаштувань гри. У ньому перевірено зміну режиму екрана, роздільної здатності, вертикальної синхронізації, мови інтерфейсу та параметра якості текстур. Оскільки в проєкті використовується один повзунок Texture Quality, тестування полягало в перевірці того, як сцена виглядає при різних значеннях цього параметра. Окремі текстури низької якості для гри не створювалися, тому при зменшенні налаштування не відбувається повна заміна ассетів [14, 16].

Після зміни роздільної здатності або режиму екрана перевірялося, чи не зміщується інтерфейс і чи залишаються кнопки доступними для користувача. Для меню налаштувань це важливо, оскільки навіть незначне зміщення елементів може ускладнити використання гри. Також перевірялося, чи залишається сцена

зрозумілою після зміни якості текстур, оскільки персонаж, вороги та платформи мають залишатися помітними незалежно від вибраного параметра.

Зручність керування перевірялася через повторне проходження одних і тих самих ділянок рівня. Це дозволило оцінити, чи комфортно персонаж рухається, чи достатньо швидко реагує на натискання клавіш, чи вистачає висоти стрибка для платформ і чи не є атака занадто повільною. Також перевірялося, чи зрозуміло гравцеві, коли персонаж отримує шкоду, коли ворог був уражений і коли потрібно змінити дію.

### 4.6 Балансування геймплею

Після тестування основних механік виконано налаштування параметрів, які впливають на складність і зручність проходження. Для «Blade of Destiny» баланс не зводиться лише до кількості ворогів або рівня шкоди. На проходження впливають швидкість персонажа, висота стрибка, розміщення платформ, відстань до ворогів, частота атак і те, наскільки добре гравець бачить небезпеку на екрані.

Швидкість руху персонажа підібрано так, щоб гравець міг нормально реагувати на перешкоди й ворогів. Надто повільний рух робить проходження затягнутим, а надто швидкий ускладнює точні стрибки та бій. Стрибок також перевірено в контексті рівня, а не окремо від нього. Герой має діставати до потрібних платформ, але не повинен перестрибувати складні ділянки без зусиль.

Поведінку ворогів налаштовано так, щоб вони створювали загрозу, але не блокували проходження повністю. Якщо ворог атакує занадто часто, гравець не має часу на реакцію. Якщо атаки відбуваються занадто рідко, бойова сцена втрачає напруження. Тому перевірялися відстань між персонажем і ворогом, момент початку атаки, реакція на удар і можливість гравця уникнути пошкодження [6, 13].

Під час балансування ворогів враховано не лише їхню силу, а й розміщення на рівні. Один і той самий ворог може сприйматися по-різному залежно від того, де саме він знаходиться: на відкритій ділянці, біля платформи, у вузькому проході або

після складного стрибка. Через це розміщення супротивників перевірялося разом із рухом персонажа, щоб бій не виглядав випадковим або надмірно незручним.

Окремо враховано бойову читабельність. Гравець повинен бачити момент удару мечем, реакцію ворога та отримання шкоди персонажем. Якщо анімація атаки занадто коротка або візуальний ефект з’являється із запізненням, бій сприймається неточно. Через це анімації, ефекти ударів і логіка нанесення шкоди перевірялися разом.

Розміщення платформ, ворогів і перешкод перевірялося з погляду поступового ускладнення. На початку рівня гравець має звикнути до керування, руху та атаки. Далі можна додавати складніші ситуації: ворога біля платформи, стрибок перед небезпечною ділянкою або необхідність атакувати після переміщення. Такий підхід робить проходження більш природним, оскільки гравець поступово стикається з новими комбінаціями дій[2, 5].

Баланс також перевірявся через повторне проходження рівня. Один і той самий фрагмент тестувався кілька разів: із повільним рухом, активними атаками, уникненням ворогів і різними варіантами стрибків. Це допомогло побачити, чи не виникають місця, де персонаж застрягає, не дістає до платформи або отримує шкоду занадто несподівано.

Під час балансування враховано не лише складність, а й темп проходження. Якщо рівень занадто довго не пропонує жодної дії, гравець може втратити інтерес. Якщо навпаки — одразу з’являється багато ворогів і перешкод, проходження стає різким і незручним. Тому рівень має чергувати спокійніші ділянки, прості стрибки, бойові ситуації та моменти з більшою концентрацією небезпек.

Після налаштування балансу рівень має сприйматися як послідовність зрозумілих ситуацій. Спочатку гравець звикає до руху, потім використовує стрибки, далі переходить до бойових дій і поєднує кілька механік одночасно. Така побудова дозволяє поступово підвищувати складність без різких переходів. Для «Blade of Destiny» це важливо, оскільки гра поєднує платформерне проходження з ближнім боєм, а отже гравець має одночасно контролювати рух персонажа й реагувати на ворогів.

# ВИСНОВКИ

У кваліфікаційній роботі розглянуто процес розробки 2D side-scroller гри «Blade of Destiny» із застосуванням ігрового рушія Unreal Engine 4. Основну увагу в роботі приділено створенню, підготовці та інтеграції візуальної частини гри: графіки головного персонажа, ворогів, елементів оточення, спрайтів, анімацій та візуальних ефектів для бойових дій. Проєкт орієнтований на створення зрозумілої, стилізованої та цілісної 2D-гри, у якій графічне оформлення не лише формує зовнішній вигляд, а й допомагає гравцеві краще сприймати ігровий процес.

У першому розділі проаналізовано предметну галузь та особливості 2D side-scroller ігор. Розглянуто характерні риси жанру: бокову перспективу, послідовне проходження рівнів, рух персонажа, стрибки, перешкоди, ворогів і бойові дії. Встановлено, що такий жанр добре підходить для гри «Blade of Destiny», оскільки дозволяє чітко показати шлях персонажа-самурая, його атаки, взаємодію з ворогами та поступове просування локаціями.

Окремо розглянуто роль візуального оформлення у 2D-іграх. Графіка в таких проєктах має не тільки естетичне значення, а й практичну функцію. Вона допомагає гравцеві швидко зрозуміти, де знаходиться персонаж, які об’єкти є небезпечними, куди можна рухатися та коли відбувається атака. Для «Blade of Destiny» це особливо важливо, оскільки гра містить бойові сцени, де гравець повинен добре бачити самурая, ворогів, платформи та ефекти ударів.

У межах аналізу аналогів розглянуто ігри Hollow Knight, Celeste та Magic Rampage. На прикладі Hollow Knight визначено значення атмосфери, цілісного стилю та впізнаваного образу персонажа. Celeste показує важливість читабельності сцени, точного керування та зрозумілої побудови рівнів. Magic Rampage є прикладом поєднання платформера з активною бойовою системою, зброєю, ворогами та візуальними ефектами. Аналіз цих проєктів допоміг визначити основні вимоги до «Blade of Destiny»: зрозумілий геймплей, помітний головний персонаж, єдиний стиль, читабельні рівні, якісні анімації та виразні ефекти атак.

У другому розділі розглянуто засоби реалізації, які використано під час створення гри. Основним середовищем розробки обрано Unreal Engine 4, оскільки він дозволяє працювати з рівнями, ігровими об’єктами, анімаціями, ассетами та тестуванням сцени в одному середовищі. Також розглянуто інші рушії, зокрема Unity, Godot і GameMaker Studio, однак для цього проєкту Unreal Engine 4 виявився зручнішим через підтримку Paper2D, Blueprint, редактор рівнів і можливість швидко перевіряти результат у сцені.

Для створення ігрової логіки використано Blueprint. Ця система візуального програмування дала змогу налаштовувати рух персонажа, атаку, отримання шкоди, взаємодію з ворогами та запуск анімацій без повного написання коду. Для роботи з 2D-графікою застосовано Paper2D, за допомогою якого імпортовано спрайти, створено flipbook-анімації, налаштовано 2D-колізії та підготовлено графічні елементи до використання в ігровій сцені.

Для створення та редагування графічних матеріалів використано Adobe Photoshop. У ньому підготовлено зображення персонажа, ворогів, фонів, текстур, елементів оточення та візуальних ефектів для атак і ударів. Для організації роботи з файлами та контролю версій застосовано Git, GitHub або GitLab. Такий набір засобів дав змогу послідовно виконувати роботу: створювати графіку, готувати її до імпорту, переносити в Unreal Engine 4, налаштовувати анімації та перевіряти результат у грі.

У третьому розділі описано проєктування та розробку візуальної частини «Blade of Destiny». Розробку побудовано навколо загальної концепції гри: самурай проходить небезпечний світ, зустрічає ворогів, долає перешкоди та шукає власне призначення. Така концепція вплинула на зовнішній вигляд персонажа, ворогів, локацій, анімацій та бойових ефектів.

Під час формування візуального стилю враховано необхідність збереження єдності між усіма елементами гри. Персонаж, вороги, фони, платформи, декоративні об’єкти та ефекти повинні виглядати як частини одного світу. Головний герой має добре виділятися на фоні рівня, вороги повинні сприйматися як загроза, а оточення має створювати атмосферу небезпечної подорожі, не

заважаючи проходженню. Для цього підібрано відповідну кольорову палітру, рівень деталізації, силуети та візуальні акценти.

Розроблено 2D-графіку головного персонажа та ворогів. Для персонажа-самурая акцент зроблено на впізнаваному силуеті, зброї, поставі та зручності подальшої анімації. Для ворогів підібрано окремий стиль, який відрізняє їх від головного героя та допомагає гравцеві швидко розпізнавати небезпеку. Також створено елементи оточення: фони, платформи, тайли, декоративні об’єкти, лісові та печерні елементи сцени.

Окрему увагу приділено підготовці ассетів до імпорту в Unreal Engine 4. Графічні матеріали збережено у відповідному форматі, перевірено розміри, прозорість, положення кадрів та структуру файлів. Після цього ассети інтегровано в рушій, налаштовано спрайти, створено flipbook-анімації та перевірено їхню роботу в ігровій сцені. Для персонажа підготовлено основні анімаційні стани: idle, run, jump, attack, hit і death. Також створено візуальні ефекти для атак, зокрема слід траєкторії удару мечем, спалахи та реакції на зіткнення.

У четвертому розділі описано практичну реалізацію, оптимізацію та тестування гри. Проєкт в Unreal Engine 4 організовано за окремими групами матеріалів: персонажі, вороги, рівні, текстури, анімації, ефекти та Blueprint-класи. Така структура дала змогу підтримувати порядок у файлах, швидше знаходити потрібні ассети та зручніше працювати з великою кількістю графічних матеріалів.

У грі реалізовано базові механіки, необхідні для 2D side-scroller проєкту: рух персонажа, стрибок, атаку мечем, взаємодію з ворогами, отримання шкоди та проходження рівня. Окремо налаштовано зв’язок між механіками та візуальною частиною. Під час руху запускається анімація бігу, під час атаки — анімація удару та ефект траєкторії меча, а при отриманні шкоди — відповідна реакція персонажа. Завдяки цьому дії героя мають зрозуміле візуальне підтвердження.

Оптимізацію графіки виконано з урахуванням особливостей Unreal Engine 4 та 2D-графіки. Для ассетів підібрано відповідні розміри, використано формат PNG із прозорістю, зменшено зайві прозорі поля, упорядковано кадри анімацій і структуру папок. У грі також передбачено меню налаштувань, де користувач може

змінювати роздільну здатність, режим екрана, V-Sync, мову інтерфейсу та якість текстур. Повзунок якості текстур не замінює ассети на окремі низькоякісні варіанти, але впливає на загальний рівень відображення сцени.

Проведено тестування ігрового процесу. Перевірено роботу анімацій, колізій, спрайтів, атак, отримання шкоди, меню налаштувань і загального проходження рівня. Особливу увагу приділено тому, щоб персонаж не провалювався крізь платформи, не застрягав на краях, а удари мечем збігалися з анімацією та візуальним ефектом. Також перевірено, чи достатньо помітні вороги, чи не зливаються вони з фоном і чи не перекривають ефекти важливі елементи сцени.

Під час балансування налаштовано швидкість персонажа, висоту стрибка, частоту атак ворогів, розміщення платформ і загальну складність проходження. Перевірка виконувалася через повторне проходження тестових ділянок рівня. Це дало змогу оцінити, чи зручно керувати персонажем, чи не є вороги надто складними або надто простими, чи зрозуміло гравцеві, куди рухатися далі та як реагувати на небезпеку.

У результаті виконано основні завдання кваліфікаційної роботи. Проаналізовано жанр side-scroller, досліджено аналоги, визначено вимоги до гри, обґрунтовано вибір інструментів, створено графічні матеріали, підготовлено ассети, інтегровано їх у Unreal Engine 4, налаштовано анімації, візуальні ефекти та базові механіки. Також проведено тестування і балансування, що дозволило перевірити коректність роботи гри в умовах реального проходження.

Практична цінність роботи полягає в тому, що створена візуальна основа може використовуватися для подальшого розвитку «Blade of Destiny». Проєкт можна розширювати новими рівнями, типами ворогів, анімаціями, ефектами, предметами та ігровими механіками. Підготовлена структура ассетів і використання Unreal Engine 4 дозволяють надалі доповнювати гру без повної перебудови вже створеної основи.

Подальший розвиток гри може передбачати розширення бойової системи, додавання нових комбінацій атак, покращення поведінки ворогів, створення нових локацій і вдосконалення системи налаштувань графіки. Також перспективним є

покращення візуальних ефектів, додавання нових анімаційних станів і розширення сюжетної частини. Це дозволить зробити «Blade of Destiny» більш повною, різноманітною та цікавою для гравця.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Марковець М.М., Куклінський М.В. Застосування засобів Unreal Engine в розробці сайд-скролер проєкту. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2025, С.404 - 407.

2. Марковець М.М., Куклінський М.В. Розробка візуального контенту та ігрових механік у сайд-скролер проєкті на Unreal Engine. Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 року, Київ, ДУІКТ. Збірник тез. – К.: ДУІКТ, 2026, С.142 - 144.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Schell J. The Art of Game Design: A Book of Lenses. 3rd ed. Boca Raton: CRC Press, 2019. 652 p.

2. Fullerton T. Game Design Workshop: A Playcentric Approach to Creating Innovative Games. 5th ed. Boca Raton: CRC Press, 2024. 586 p.

3. Rogers S. Level Up! The Guide to Great Video Game Design. 2nd ed. Chichester: Wiley, 2014. 560 p.

4. Nystrom R. Game Programming Patterns. Genever Benning, 2014. 354 p.

5. Totten C. W. An Architectural Approach to Level Design. 2nd ed. Boca Raton: CRC Press, 2019. 476 p.

6. Swink S. Game Feel: A Game Designer's Guide to Virtual Sensation. Burlington: Morgan Kaufmann, 2008. 358 p.

7. Novak J. Game Development Essentials: An Introduction. 3rd ed. Clifton Park: Cengage Learning, 2012. 512 p.

8. Romero M., Sewell B. Blueprints Visual Scripting for Unreal Engine 5. 3rd ed. Birmingham: Packt Publishing, 2022. 568 p.

9. Butler S., Oliver T. Game Development Patterns with Unreal Engine 5: Build Maintainable and Scalable Systems with C++ and Blueprint. Birmingham: Packt Publishing, 2024. 254 p.

10. Marques G., Sherry D., Pereira D., Fozi H. Elevating Game Experiences with Unreal Engine 5. 2nd ed. Birmingham: Packt Publishing, 2023. 760 p.

11. Doran J. P. Unreal Engine Game Development Cookbook. Birmingham: Packt Publishing, 2015. 326 p.

12. Okur M. The Art of Graphic Design in Video Games: Beyond the Visual. Amazonia Investiga. 2024. Vol. 13, no. 77. P. 102–113.

13. Caroux L., Pujol M. Player Enjoyment in Video Games: A Systematic Review and Meta-analysis of the Effects of Game Design Choices. Psychological Bulletin. 2023. Vol. 149, no. 5–6. P. 337–370.

14. Pasqualotto A., Altarelli I., De Angeli A., Menestrina Z., Bavelier D., Venuti P. Video Game Design for Learning to Learn. International Journal of Human–Computer Interaction. 2023. Vol. 39, no. 11. P. 2211–2228.

15. Mai L. C. E., Le N. T., Nguyen T. H. Analysing of Players' Perceptions on Game Aesthetics. Frontiers in Communication. 2025. Vol. 10. Article 1622613.

16. Epic Games. Unreal Engine 4.27 Documentation. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-4-27-documentation (date of access: 13.04.2026).

17. Epic Games. Blueprints Visual Scripting in Unreal Engine. URL: https://dev.epicgames.com/documentation/unreal-engine/blueprints-visual-scripting-in-unreal-engine (date of access: 15.04.2026).

18. Epic Games. Paper 2D in Unreal Engine 4.27 Documentation. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/paper-2d?application_version=4.27 (date of access: 19.04.2026).

19. Epic Games. Paper 2D Sprites in Unreal Engine 4.27 Documentation. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/paper-2d-sprites?application_version=4.27 (date of access: 19.04.2026).

20. Epic Games. Paper 2D Flipbooks in Unreal Engine. URL: https://dev.epicgames.com/documentation/unreal-engine/paper-2d-flipbooks-in-unreal-engine (date of access: 18.04.2026).

21. Adobe. Work with the Layers Panel in Photoshop. URL: https://helpx.adobe.com/photoshop/desktop/create-manage-layers/get-started-layers/work-with-the-layers-panel.html (date of access: 15.05.2026).

22. Chacon S., Straub B. Pro Git. 2nd ed. URL: https://git-scm.com/book/en/v2 (date of access: 19.05.2026).

23. Team Cherry. Hollow Knight — Official Website. URL: https://www.hollowknight.com (date of access: 10.05.2026).

24. Extremely OK Games. Celeste — Official Website. URL: https://www.celestegame.com (date of access: 10.05.2026).

25. Asantee Games. Magic Rampage — Official Website. URL: https://magicrampage.com (date of access: 10.05.2026).

26. ItProger. Ігровий движок Godot! Чи замінить він Unity та Unreal Engine? — URL: https://itproger.com/ua/news/igrovoy-dvizhok-godot-zamenit-li-on-unity-i-unreal-engine (date of access: 25.03.2026).

# ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

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


## Розробка сайд-скролер проєкту із застосування ігрового рушію Unreal Engine

Виконав студент 4 курсу
групи ПД-42
Марковець Микола Миколайович
Керівник роботи
канд.техн.наук Куклінський Максим Володимирович

Київ – 2026

## МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

**Мета роботи**: підвищення якості візуальної реалізації гри «Blade of Destiny» шляхом створення та інтеграції 2D-графіки, анімацій і VFX в Unreal Engine 4.

**Об'єкт дослідження:** процес розробки візуальної складової 2D side-scroller гри «Blade of Destiny».

**Предмет дослідження:** методи та засоби створення, оптимізації та інтеграції 2D-графіки, спрайтів, анімацій і VFX у Unreal Engine 4.

# ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Проаналізувати особливості жанру 2D side-scroller, визначити їхні ключові особливості та вплив на сучасну ігрову індустрію.
2. Проаналізувати візуальну складову ігор-аналогів: Hollow Knight, Celeste, Magic Rampage та визначити їх переваги та недоліки.
3. Визначити програмні засоби для реалізації гри (ігровий рушій, технології роботи з графікою всередині рушія(Paper2D, Blueprint), графічне середовище розробки).
4. Розробити графічні матеріали для персонажа, ворогів, оточення та рівнів.
5. Підготувати ассети до імпорту в Unreal Engine 4.
6. Інтегрувати спрайти, анімації та візуальні ефекти в рушій.
7. Провести тестування і балансування ігрового процесу.

3

# АНАЛІЗ АНАЛОГІВ

| Критерій аналізу | Hollow Knight | Celeste | Magic Rampage | Blade of Destiny |
|---|---|---|---|---|
| Розвиток персонажа | часткове (з сюжетом) | – | з проходженням рівнів | з проходженням рівнів + сюжетом |
| Різноманіття анімацій атак | + (лише ближні) | – | + (ближні, дальні) | + (ближні, дальні) |
| Персоналізація | – | – | скіни | скіни |
| Оточення та рівні | темні, атмосферні локації | рівні з акцентом на паркур | підземелля та бойові рівні | ліси, підземелля, платформи, замок |
| Приховані локації | + | – | – | + |
| Внутрішньоігровий магазин | покращення (через прихованого на локаціях торговця) | – | скіни | скіни, покращення |

4

# ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**Функціональні вимоги:**

1. Переміщення рівнем, стрибки та взаємодія з платформами.
2. Виконання атаки мечем і взаємодія з ворогами(візуальна складова).
3. Обробка отримання шкоди персонажем та ворогами(реакція на отримання шкоди);
4. Проходження рівнів і перехід між ігровими станами;
5. Запуск анімацій idle, run, jump, attack, hit, death;
6. Відображення VFX для атак, ударів і зіткнень;
7. Робота меню, налаштувань та базових елементів інтерфейсу.

**Нефункціональні вимоги:**

1. Стабільна робота(30, 60 FPS) гри, без критичних помилок під час тривалого ігрового процесу.
2. Класичне для жанру керування (WASD, shift).
3. Коректне відображення сутностей на сцені під час руху та бою, через обводку.
4. Єдиний візуальний стиль персонажів, ворогів і оточення.
5. Оптимізація 2d-ассетів для unreal engine 4; коректне відображення спрайтів, анімацій і VFX.
6. Підтримка різних графічних налаштувань гри(повзунок з відсотком якості редерингу текстур).

**Мінімальні системні вимоги:** ОС: Windows 10 Процесор: Intel Core i3 / AMD Ryzen 3 Оперативна пам'ять: 4 GB RAM Відеокарта: NVIDIA GeForce GTX 750 Ti / AMD Radeon RX 560 Місце на диску: 2 GB.
**Рекомендовані системні вимоги:** ОС: Windows 10 / 11 Процесор: Intel Core i5 / AMD Ryzen 5 Оперативна пам'ять: 8 GB RAM Відеокарта: NVIDIA GeForce GTX 1060 / AMD Radeon RX 570 Місце на диску: 4 GB.

5

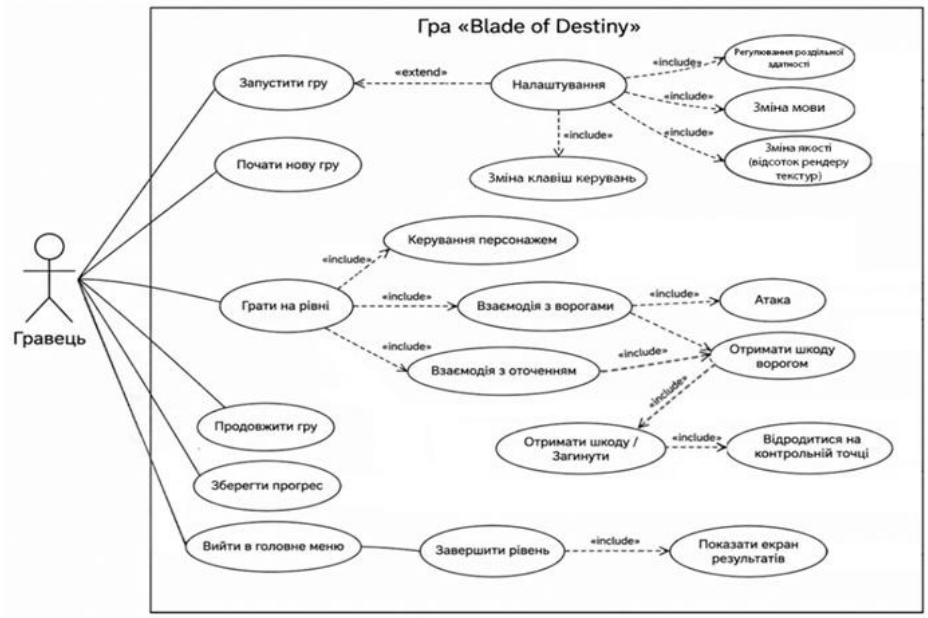
# ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ





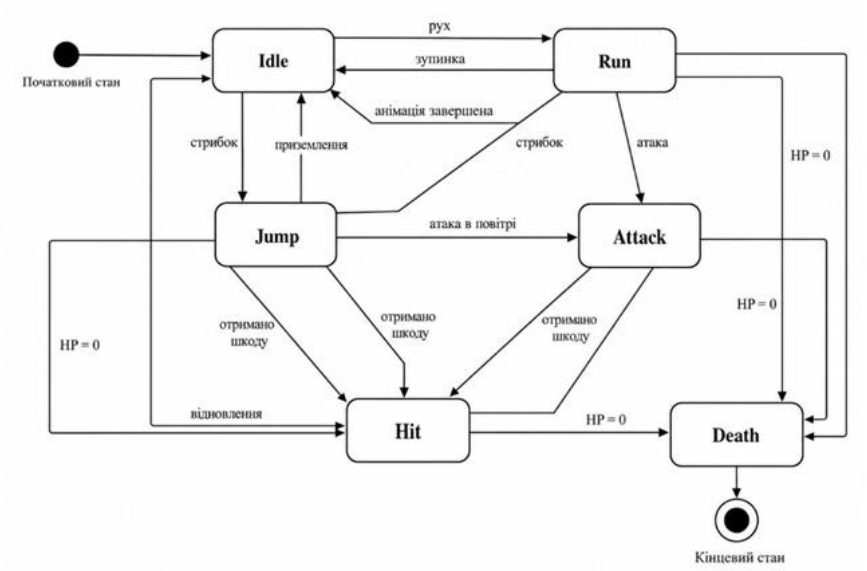

6

# ДІАГРАМА ПРЕЦЕДЕНТІВ




7

# ДІАГРАМА СТАНІВ АНІМАЦІЇ ПЕРСОНАЖА




8

# ЕКРАННІ ФОРМИ


«Головне меню гри “Blade of Destiny”»

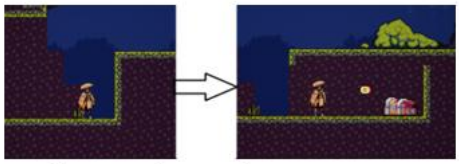

«Меню налаштувань»


«Магазин скінів»

9

# ЕКРАННІ ФОРМИ


«Початок гри/Перехід на новий рівень»



«Приховані локації з нагородами»

«Взаємодія з NPC: Діалог»

«Демонстрація атаки ворога та бойової взаємодії з персонажем»

10

# АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Марковець М.М., Куклінський М.В. Застосування засобів Unreal Engine в розробці сайд-скролер проєкту. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2025, С.404 - 407.

2. Марковець М.М., Куклінський М.В. Розробка візуального контенту та ігрових механік у сайд-скролер проєкті на Unreal Engine. Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 року, Київ, ДУІКТ. Збірник тез. – К.: ДУІКТ, 2026, С.142 - 144.

11

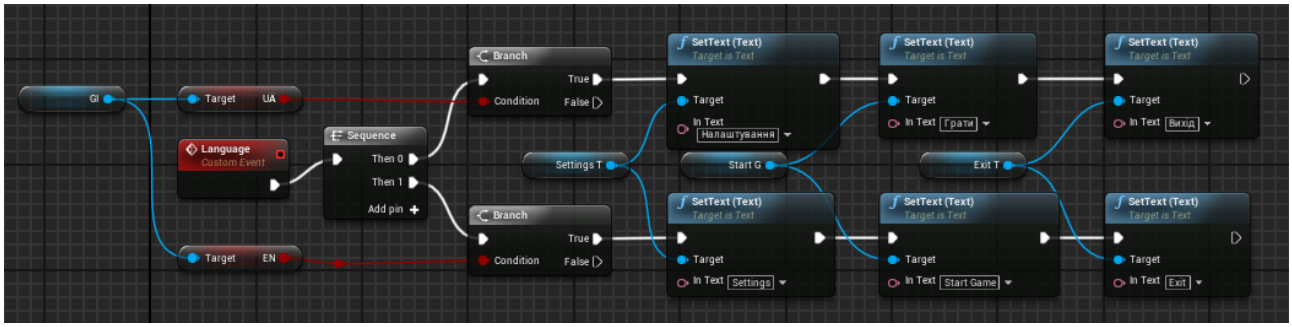
# ВИСНОВКИ

1. Досліджено роль візуального оформлення, анімацій та VFX у 2D-іграх. Встановлено, що графіка в такому проєкті впливає не лише на зовнішній вигляд, а й на читабельність сцени, сприйняття атак, руху та взаємодії з ворогами.
2. Проведено аналіз ігор-аналогів Hollow Knight, Celeste та Magic Rampage. На основі порівняння визначено ключові рішення для «Blade of Destiny».
3. Обґрунтовано вибір засобів реалізації Unreal Engine 4, Blueprint, Paper2D, Adobe Photoshop та Git. Ці інструменти забезпечили створення графіки, інтеграцію ассетів, налаштування анімацій, реалізацію логіки та контроль версій проєкту.
4. Розроблено візуальні матеріали для гри, а саме графіку персонажа, ворогів, оточення, рівнів, спрайти, анімації та ефекти атак.
5. Проведено тестування та балансування ігрового процесу. Перевірено роботу персонажа, ворогів, анімацій, колізій, VFX, меню налаштувань і бойових сцен, що дозволило покращити зручність проходження та загальну якість гри.

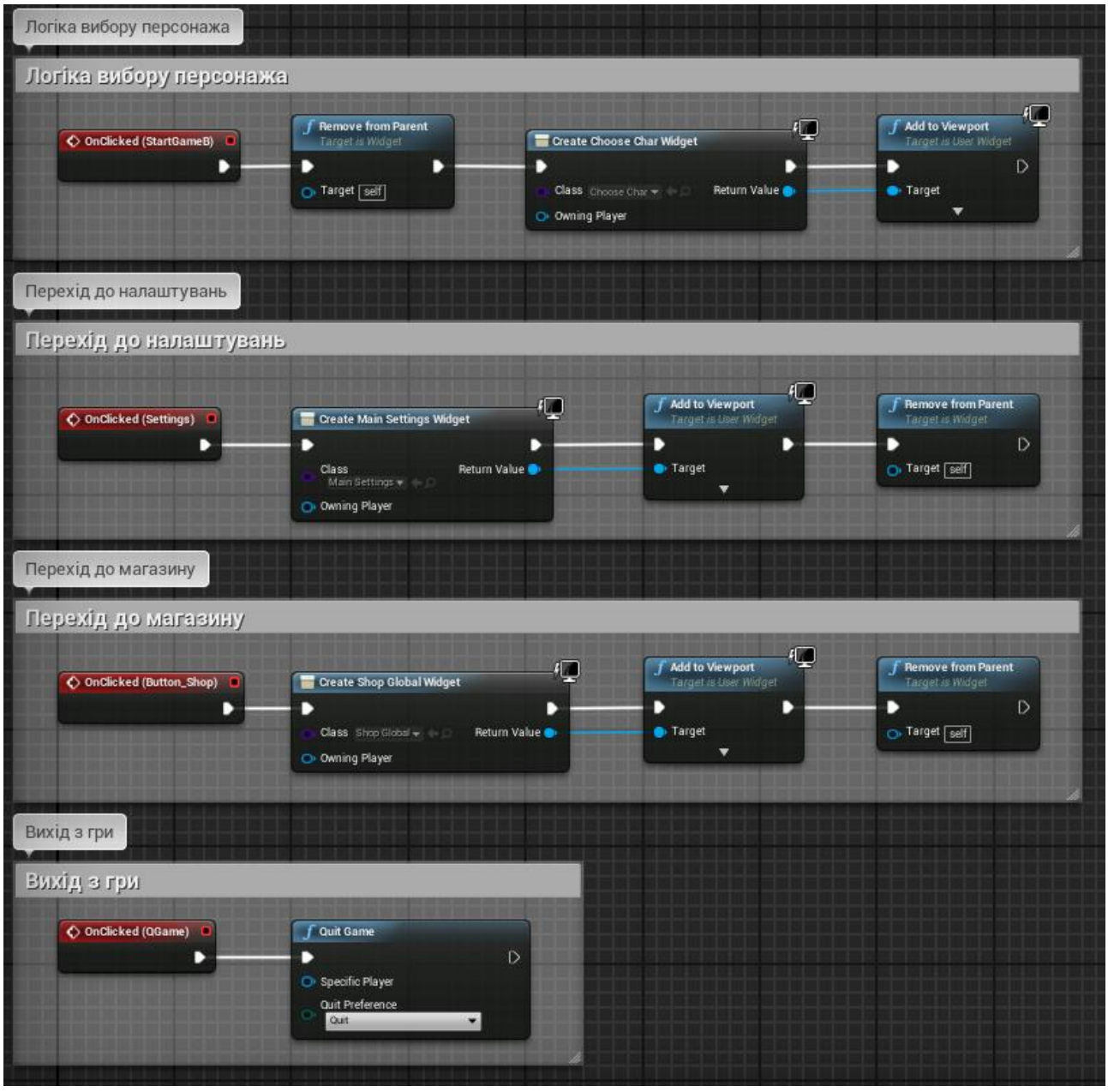
12

# ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

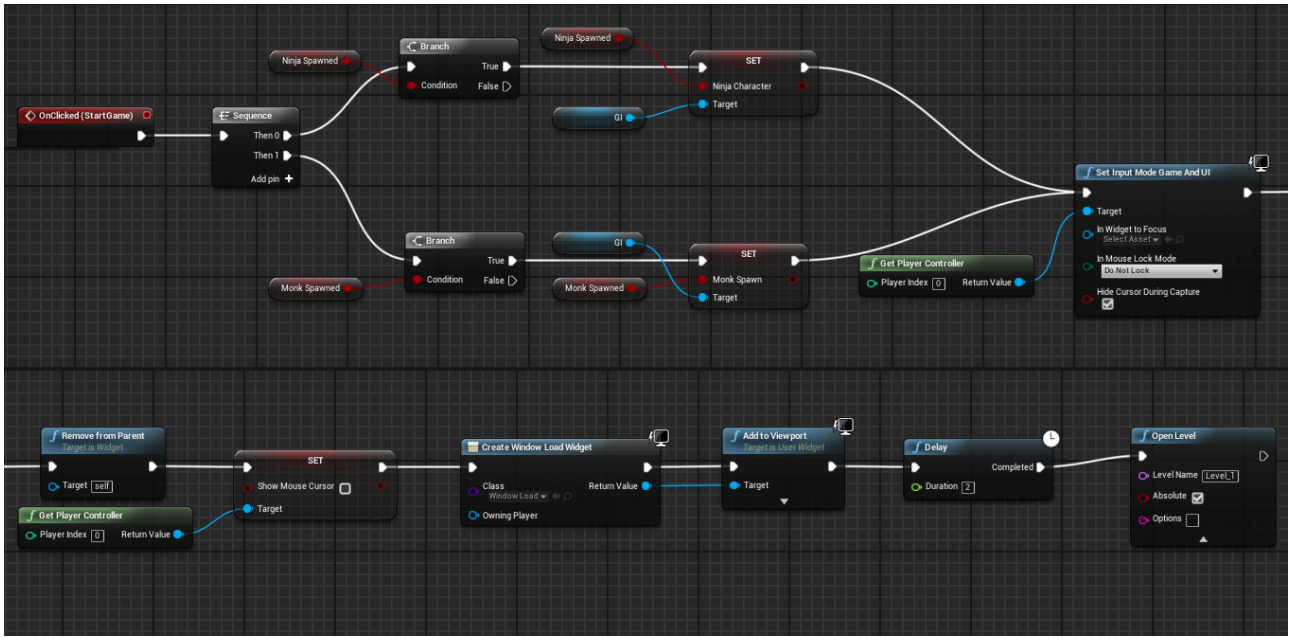
Англійська та українська локалізації у головному меню:



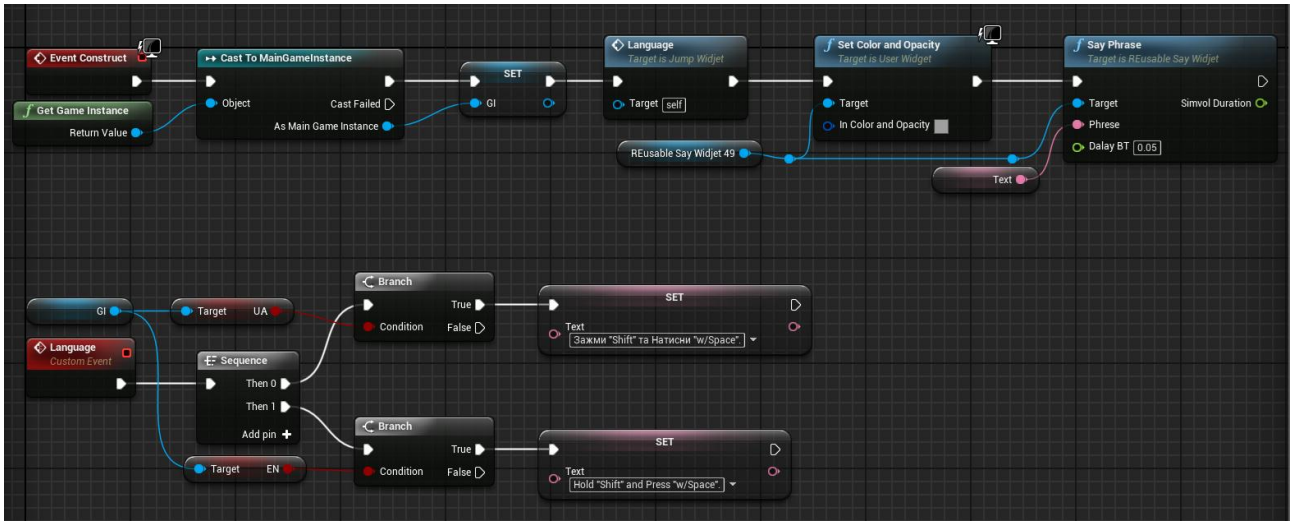

Функції головного меню:



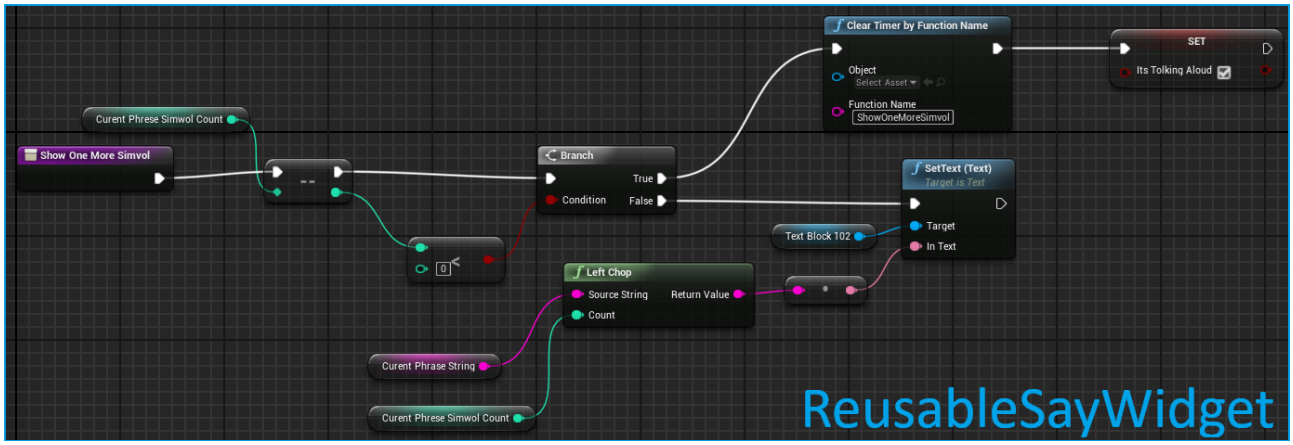
Обробник події вибору персонажу:



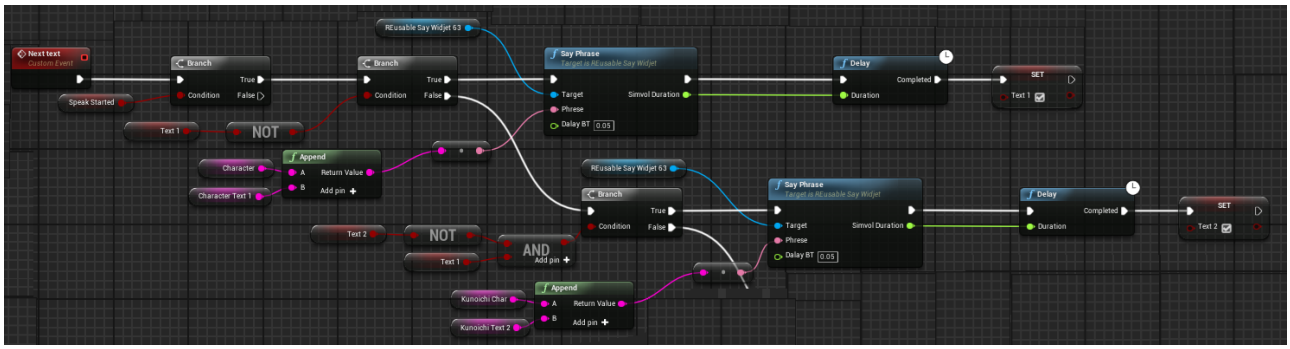

Відображення підказки під час ігрового процесу:



ReusableSayWidget використовується для виклику діалогів:




Приклад послідовного використання діалогу через ReusableSayWidget:



Виклик реплік через ReusableSayWidget: