Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)
Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Ігровий програмний застосунок в жанрі Action RPG, піджанр Souls-like. Бойова підсистема з елементами розвитку персонажу.

(тема)

Виконав:
здобувач 4 року навчання,
групи ПЗПІ-21-6

Артур ТКАЧОВ
(власне ім'я, прізвище)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. Ольга КАЛИНИЧЕНКО
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри _____ Кирило СМЕЛЯКОВ
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)
Тип програми освітньо-професійна
Освітня програма Програмна Інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Ткачову Артуру Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Ігровий програмний застосунок в жанрі Action RPG, піджанр Souls-like. Бойова підсистема з елементами розвитку персонажу.
Затверджена наказом по університету від 19.05. 2025р. № 397 Ст
2. Термін подання студентом роботи до екзаменаційної комісії
3. Вихідні дані до роботи Розробити ігровий застосунок в жанрі Souls-like, котрий міститиме в собі 3 підігри, які можна обрати через головне меню, мовами програмування C# та HLSL використовуючи ігровий двигун Unity3D.
4. Перелік питань, що потрібно опрацювати в роботі
Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	20.05.2025	*виконано*
2	Створення специфікації ПЗ	22.05.2025	*виконано*
3	Проектування ПЗ	24.05.2025	*виконано*
4	Розробка ПЗ	28.05.2025	*виконано*
5	Тестування ПЗ	30.05.2025	*виконано*
6	Оформлення пояснювальної записки	05.06.2025	*виконано*
7	Підготовка презентації та доповіді	06.06.2025	*виконано*
8	Попередній захист	09.06.2025	*виконано*
9	Нормоконтроль, рецензування	06.06.2025	*виконано*
10	Здача роботи у електронний архів	10.06.2025	*виконано*
11	Допуск до захисту у зав. кафедри	11.06.2025	*виконано*

Дата видачі завдання 19 травня 2025р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Ольга КАЛИНИЧЕНКО
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра, 63 с., 10 рис., 4 формули, 3 додатка, 15 джерел.

БОЙОВА СИСТЕМА, ІГРОВИЙ ПРОГРАМНИЙ ЗАСТОСУНОК, РОЗВИТОК ПЕРСОНАЖА, ACTION RPG, BLUEPRINT VISUAL SCRIPTING, SOULS-LIKE, UNREAL ENGINE 5

Об'єкт розробки – ігровий програмний застосунок в жанрі Action RPG (Souls-like).

Мета розробки – створення ігрового програмного застосунку, яке поєднує динамічний бій у стилі «Souls-like»[1] з системою розвитку персонажа.

Метод рішення – середовище розробки Unreal Engine 5, використання Blueprint Visual Scripting для реалізації механік.

У результаті розробки створено ігровий програмний застосунок, що включає графічний користувацький інтерфейс, бойову систему, комбо-атаки з залежністю від типу зброї, управління витривалістю (витрата на атаки/блок, повільне відновлення), механіки ухилення, парирування, критичних ударів, покращення базових характеристик (сила, інтелект, спритність, витривалість), систему екіпірування з унікальними ефектами перстнів.

ABSTRACT

ACTION RPG, BLUEPRINT VISUAL SCRIPTING, CHARACTER DEVELOPMENT, COMBAT SYSTEM, GAME SOFTWARE, SOULS-LIKE, UNREAL ENGINE 5

The object of development is a game software application in the genre of Action RPG (Souls-like).

The purpose of the development is to create a game software application that combines dynamic combat in the style of “Souls-like” with a character development system.

The method of solution is the Unreal Engine 5 development environment, using Blueprint Visual Scripting to implement the mechanics.

As a result of the development, a gaming software application was created that includes GUI, a combat system, combo attacks depending on the type of weapon, endurance management (attack/block cost, slow recovery), mechanics of evasion, parrying, critical hits, improving basic characteristics (strength, intelligence, agility, endurance), and an equipment system with unique ring effects.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

UE5 – Unreal Engine 5

BVS – Blueprint Visual Scripting

ПКМ – права кнопка миші

ЛКМ – ліва кнопка миші

СКМ – середня кнопка миші

ШІ – штучний інтелект

ПК – персональний комп’ютер

UMG – Unreal Motion Graphics UI Designer

HUD – heads-up display

DLC – downloadable content

EQS – Environment Query System

ВСТУП

У сучасному ігровому світі жанр Action RPG продовжує набувати популярності завдяки поєднанню глибоких бойових механік із системами розвитку персонажа. Водночас постійне зростання вимог гравців до якості та гнучкості цих механік підкреслює важливість досліджень і розробок у цій галузі. Актуальність роботи полягає у необхідності створити інноваційні рішення для підвищення доступності й водночас збереження складності Souls-like проєктів, що сприятиме не тільки розширенню аудиторії, але й розвитку технологічних підходів до балансування, адаптивного інтерфейсу та швидкого прототипування бойових систем.

Особливої уваги заслуговує піджанр Souls-like, відзначений високою вимогливістю до майстерності гравця та стратегічною глибиною кожної сутички. З огляду на зростаючий інтерес до ігор із ретельно відточеною бойовою системою та можливістю вільного налаштування персонажа, розробка «Catalous» є актуальною для науково-практичної спільноти й індустрії розваг.

Серед найбільш відомих представників жанру слід зазначити «Dark Souls» (FromSoftware, 2011), який заклав основи витривалості як ключового ресурсу, та «Elden Ring» (FromSoftware, 2022), що вдосконалив систему відкритого світу і розвиток стратегій поліпшення персонажа. Аналіз цих аналогів показує, що успіх проєкту залежить від чіткого балансу між викликом і відчуттям прогресу, а також від швидкого прототипування бойових параметрів для адаптації під потреби різних гравців.

Метою кваліфікаційної роботи є проєктування й реалізація бойової підсистеми та механік розвитку персонажа в ігровому застосунку «Catalous», які забезпечують: точний бій із керуванням витривалістю, стратегічний вибір між атакою, блоком, парируванням чи ухиленням, а також лінійне підвищення характеристик із можливістю перерозподілу очок розвитку біля точок збереження. Це дозволить

гравцям не лише випробувати власні навички, а й вільно змінювати стратегію без втрати накопичених досягнень.

Для досягнення поставленої мети обрано Unreal Engine 5[2] із вбудованою системою BVS, яка забезпечує ефективне прототипування та інтеграцію всіх елементів системи – від анімацій до штучного інтелекту ворогів. Використання UE5 гарантує сучасні можливості оптимізації та масштабованості проєкту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Сучасні «Souls-like» ігри, що панують на ринку (від класики «Dark Souls» до нових хітів на кшталт «Lies of P» чи «Lords of the Fallen»), будують свій успіх на фундаменті постійного виклику та невблаганності ігрового світу. Ключові стовпи жанру – вороги, що відроджуються після кожного збереження (або відпочинку біля точки збереження), та лімітовані, часто незворотні рішення щодо розвитку персонажа (розподіл очок характеристик) – створюють напружену атмосферу, але й стають бар'єром для багатьох потенційних гравців. Вони вимагають високої концентрації, терпіння та готовності до постійного повторення ділянок, що може викликати значне стресове навантаження.

Даний проєкт ставить за мету зберегти саму суть та складність жанру – глибокий бій, складних ворогів, потребу в майстерності та вивченні закономірностей – але при цьому свідомо м'якшає деякі з його найжорсткіших аспектів, запроваджуючи інноваційні механіки, спрямовані на зниження стресу без втрати глибини:

– Гнучкий та безпечний прогрес. На відміну від більшості конкурентів, де перерозподіл характеристик є рідкісним, дорогим або пов'язаним зі значними обмеженнями (як у «Dark Souls III» або «Lies of P», де це потребує рідкісних предметів чи доступне лише на певних етапах), Catalous інтегрує цю можливість безпосередньо в точки збереження, завдяки чому гравець отримує небувалу свободу експериментувати з різними характеристиками персонажа: від важкого мечника до спритного стрільця. Якщо обрана комбінація характеристик виявляється неефективною проти конкретного боса чи на новій локації, можна швидко та без будь-яких негативних наслідків (втрати ресурсів, необхідності шукати рідкісні предмети) повернути ці очки назад і спробувати нову стратегію. Це усуває тривогу за

помилковий вибір, що є значною стресовою складовою класичних Souls-like, і заохочує креативність.

– Стратегічне Відродження. Одна з найбільш виснажливих механік жанру – постійне відродження всіх звичайних ворогів при кожному збереженні/відпочинку – в Catalous замінена на більш стратегічну систему, де вороги відроджуються не після кожного відпочинку, а лише після повного завершення рівня, що веде за собою обмеженість ресурсів, бо гравець не може нескінченно перемагати звичайних ворогів на одній ділянці для отримання досвіду або предметів. Кожен ворог, як і кожна скриня, стає цінним ресурсом, що змушує гравця ретельно планувати витрати своїх припасів не лише на боса, а й на проходження основної частини рівня.

Ці дві фундаментальні механіки роблять Catalous доступнішою для новачків, які можуть зосередитись на освоєнні бою та дослідженні світу, не відчуваючи постійного тиску необоротності рішень чи нескінченного напливу ворогів. Водночас, вони чітко відрізняють гру від основних конкурентів («Dark Souls», «Elden Ring», «Lies of P»), де перерозподіл характеристик завжди є обмеженим, а вороги завжди чекають гравця на своїх місцях після майже будь-якої спроби відпочити.

Серед ключових конкурентів – ігри з відкритим світом («Elden Ring») та лінійні проєкти («Lies of P»). Цільова аудиторія – прихильники «Souls-like», які шукають новий досвід у жанрі та користувачі, які цікавляться RPG, але лякаються надмірної складності класичних представників жанру. Використання UE5 забезпечує високу якість графіки та оптимізацію навіть на середньостатистичному ПК. BVS дозволяє швидко створювати прототипи без великих витрат на розробку.

Стратегія монетизації орієнтована на лояльність спільноти та якісний додатковий контент. Монетизація буде включати в себе виключно продаж базової гри на платформі «Steam» та DLC (додатковий контент), що будуть включати нові рівні, боси, види зброї. Це забезпечить тривалий інтерес до гри після її випуску.

Використання потужного рушія Unreal Engine 5 забезпечує високу якість графіки (Lumen, Nanite) та відмінну оптимізацію, дозволяючи грі комфортно працювати навіть на середньостатистичних ПК. Для швидкої ітерації та прототипування механік використовується BVS (Blueprint Visual Scripting) у UE5, що дозволяє розробникам швидко втілювати ідеї без великих витрат часу на написання коду, прискорюючи процес розробки та тестування. Ця технологічна база робить Catalous легко масштабованою. Наприклад, можна без великих труднощів додавати нові типи точок збереження/відпочинку з унікальними функціями (наприклад, тимчасовий бонус до характеристик), створювати нові рівні з різними правилами відродження ворогів чи унікальними механіками інтегруючи це в існуючу систему, легко розширювати арсенал зброї та інших предметів для підтримки різноманітності ігрового процесу.

Як висновок, Catalous пропонує особливий підхід до жанру «Souls-like», де ключові механіки (перерозподіл характеристик, обмежене відродження ворогів) покращують ігровий досвід. Гнучкість покращення залучає широку аудиторію, а відмова від постійної появи ворогів робить гру доступнішою без втрати складності. Використання UE5 дозволяє легко масштабувати систему – наприклад, додавати нові типи точок збереження або рівні з унікальними правилами. Ці особливості, поєднані з класичною «Souls-like» атмосферою, позиціонують Catalous як проєкт, здатний зайняти своє місце серед інших представників жанру.

1.2 Виявлення та вирішення проблем

Сучасні ігри в жанрі «Souls-like» традиційно відзначаються високою вимогливістю до гравця та жорсткими наслідками ухвалених рішень. Однією з найбільш відчутних проблем є незворотність ранніх виборів у розподілі характеристик: якщо гравець помилився при інвестуванні очок розвитку, йому доводиться або перезапускати проходження, або дочекатися моменту, коли в грі з’явиться можливість перерозподілу очок. Такий підхід створює психологічний бар’єр

і приводить до страху помилки, що особливо відлякує новачків жанру, які не готові витрачати години на виправлення разових промахів.

У «Catalous» ця проблема вирішена через впровадження механіки перерозподілу очок розвитку біля точок збереження. Кожен гравець може у будь-який момент переглянути та відкоригувати інвестиції у статистики персонажа без необхідності починати рівень наново. Це знижує емоційний тиск і стимулює експериментування з різними комбінаціями здібностей, оскільки будь-яку помилку можна виправити буквально за кілька секунд. Завдяки такому підходу геймер отримує свободу пошуку власного стилю проходження, що додає довговічності та різноманітності ігровому процесу.

Ще однією характерною рисою «Souls-like» ігор є повторне відродження ворогів після завантаження збереження, що часто перетворює просування рівнями на рутину без смислового навантаження. В «Catalous» супротивники не відроджуються після відвідування контрольної точки, а з’являються лише після повного завершення рівня. Така механіка зроблена для того, щоб гравці могли концентруватися на тактичних рішеннях і стратегічному плануванні, а не на нескінченних бійках з одними і тими самими ворогами. Для новачків це означає менше стресу і більшу мотивацію рухатися вперед, а для досвідчених – збереження гостроти сутичок і можливість обирати оптимальну тактику без зайвих перешкод.

Водночас «Catalous» не знижує загальний рівень виклику жанру, а навпаки — підсилює його завдяки поєднанню свободи налаштування персонажа та збереженню класичних елементів Souls-like. Навички гравця залишаються вирішальними, адже успіх у битві залежить не лише від статистик, але й від чіткого відчуття моменту ударів, блоків і ухилень. Можливість швидкого перерозподілу очок та відсутність безкінечного відродження ворогів створюють баланс між доступністю для новачків і збереженням глибини для ветеранів.

Таким чином, «Catalous» перетворює проблему «страху помилки» на перевагу, дозволяючи гравцям творчо досліджувати механіки бою та системи розвитку.

Відсутність кари за експерименти з характеристиками та оптимізовані рівні забезпечують плавний темп проходження, а добре налаштована бойова система гарантує відчуття досягнення та майстерності.

1.3 Порівняння з конкурентами

Аналіз лідерів жанру "Souls-like"[3][4] виявив критично важливі уроки для проектування бою в "Catalous". Розглянемо сильні та слабкі сторони глибше та визначимо додаткові нюанси:

а) «Dark Souls». Плюси:

1) Система витривалості. Це не просто ліміт дій – це основа стратегічного бою. Кожен удар, блок, ухилення чи біг має чітку вартість. Це змушує гравця постійно оцінювати ризик/винагороду, планувати послідовності атак і керувати агресивністю. Витривалість стає ключовим ресурсом, що визначає темп сутички.
2) Якісна візуальна та аудіо-зворотна реакція. Успішне парирування супроводжується характерним ляскання м та вібрацією геймпада, критичний удар – специфічною анімацією та звуком. Блок ворожої атаки викликає видимий ефект і відчутний відкид. Це формує інтуїтивне розуміння ефективності дій без необхідності постійно стежити за цифрами*, підсилюючи задоволення від майстерності.
3) Баланс ризик/винагорода через затримки атак. Анімації ворогів мають чіткі "вікна" підготовки (wind-up), активного удару та відновлення (recovery). Це дозволяє гравцеві вивчити їх поведінку, передбачати атаки та знаходити безпечні моменти для контратаки чи лікування. Ризик полягає в невірній оцінці цих вікон, а винагорода – у влучній контратакі чи безпечному лікуванні.

б) «Dark Souls». Мінуси:

1) Жорсткість перерозподілу очок. Неможливість або зайва складність зміни характеристик обмежує експериментування та дещо карає за неідеальний вибір на ранніх етапах. Необхідність знаходити рідкісні одноразові предмети або обмежена кількість можливостей створює психологічний бар'єр.
2) Відсутність інструментів швидкого прототипування балансу: Технічна застарілість рушія та інструментарію ускладнювала процес ітеративної настройки бою для розробників*. Зміна параметрів атаки, витривалості чи анімації часто вимагала значних зусиль, уповільнюючи балансувальні процеси та обмежуючи гнучкість у пошуку ідеального "feeling".
3) "Заторможеність" персонажа через анімації. Персонаж гравця має повноцінні анімації з прив'язкою до кадрів. Це означає, що після натискання кнопки атаки чи використання зілля, дія має завершити свою анімацію, перш ніж гравець зможе виконати іншу дію (наприклад, ухилитися). Це додає ваги та стратегії, але може викликати відчуття втрати контролю, особливо для новачків. Надмірна довжина деяких анімацій відновлення після дії стає покаранням за помилку.

в) «Sekiro: Shadows Die Twice». Плюси:

1) Акцент на парирування та пошкодження (Posture). Бій будується навколо ритмічного парирування, що вимагає високої точності та реакції. Успішні парирування не лише блокує шкоду, але й заповнює особливу шкалу ворога. Заповнена шкала відкриває можливість для смертельної контратаки (Deathblow). Це створює інтенсивний, схожий на танець досвід бою, де майстерність винагороджується швидкою перемогою.
2) Мінімалістична система розвитку. Відсутність класичних RPG-характеристик (сила, спритність) та рівнів персонажа зосереджує увагу гравця на вдосконаленні власних навичок та отриманні нових бойових мистецтв (Combat Arts).

г) «Sekiro: Shadows Die Twice». Мінуси:

1) Обмежена мотивація до довготривалого покращення. Відсутність традиційного рівня персонажа та відносно невеликий вибір бойових мистецтв/протезів порівняно зі зброєю/магією в інших Souls-like може зменшити мотивацію до повторного проходження. Основний прогрес – це навички гравця.
2) Надмірна гнучкість відновлення. Система відновлення здоров'я (Gourd Seeds) та витривалості робить небезпечні ситуації менш загрозливими. Гравець має багато засобів лікування, а миттєве відновлення витривалості після успішного парирування або блоку суттєво знижує покарання за помилки в управлінні ресурсом і може зменшити напругу.
3) Обмежена варіативність. Спеціалізація на мечі та парируванні обмежує можливість експериментувати з принципово різними стилями гри (наприклад, магія чи дальній бій), що є ключовим для багатьох шанувальників жанру. Гра пропонує глибину в межах одного стилю.
4) Висока база складності. Акцент на ідеальному почутті парирувань створює дуже високу планку вхідної складності. Гравцям, які не можуть опанувати цей ключовий механік, буде надзвичайно важко прогресувати.

На основі цього аналізу визначаються наступні ключові завдання для "Catalous":

а) Глибоке та гнучке налаштування характеристик:

1) Проблема: успадкувати глибину системи характеристик «Dark Souls», але усунути жорсткість та незворотність. Забезпечити ефективне налаштування вартості очок, коефіцієнтів зростання ефективності кожної характеристики (напр., скільки атаки дає 1 пункт сили, як швидко зростає витрата витривалості на атаки зі збільшенням спритності).
2) Рішення: "криві" зростання ефективності (лінійні, квадратичні, логарифмічні) для кожної характеристики та її впливу на різні аспекти (фізична/магічна атака, здоров'я, витривалість, швидкість, шанс

критичного удару тощо). Використання механіки вільного перерозподілу біля точок збереження (як описано раніше) дозволить гравцям експериментувати, а розробникам – сміливіше налаштовувати баланс, знаючи, що помилковий вибір гравця не є фатальним.

б) Золота середина у системі витривалості:

1) Проблема: знайти баланс між стратегічною глибиною системи витривалості «Dark Souls» (де кожна дія важлива) та потенційною надмірною суворістю або, навпаки, надто великою поблажливістю, як у «Sekiro». Уникнути ситуацій, коли витривалості занадто багато (зменшує стратегічність) або занадто мало (робить бій надто важким).
2) Рішення: система, що дозволяє динамічно регулювати базовий запас витривалості та швидкість її відновлення, вартість різних дій (легкі/важкі атаки, блок, ухилення, біг), залежно від використовуваної зброї, обладунків чи навичок. Штрафи за повну втрату витривалості (напр., тимчасове зниження швидкості атаки чи руху).

в) Кристально чітка візуальна та аудіо зворотна реакція:

1) Проблема: забезпечити гравця миттєвим і безпомилковим розумінням стану його витривалості, успішності його дій (влучання, блок, парирування) та дій ворога. Це критично для відчуття контролю, особливо в інтенсивному бою.
2) Рішення: простий та зрозумілий HUD, миттєва анімація витрати/відновлення витривалості (напр., швидке зменшення/наповнення). Яскраві спалахи, чіткі унікальні звуки для успішного парирування/блоку. Різні ефекти/звуки для легких/важких ударів, критичних ударів.

1.4 Постановка задачі

У межах реалізації бойової підсистеми ігрового застосунку «Catalous» було поставлено завдання чітко визначити та розділити ключові компоненти, які забезпечують відчутний і водночас збалансований бій. Першим кроком буде опрацювання механіки точного нанесення ударів і контролю витривалості: необхідно буде розробити алгоритми розрахунку витрат витривалості на кожну дію, враховуючи застосовані покращення й характер дії (атаки чи ухилення тощо). При цьому параметри відновлення витривалості мають коригуватися залежно від зовнішніх факторів – наприклад, після успішного парирування чи тривалого утримання захисної позиції – щоб зберегти баланс між викликом та справедливістю сутичок.

Другим завданням буде впровадження стратегічної глибини вибору між атакою, блоком, парируванням і ухиленням. Це означає, що кожна з цих дій отримує унікальні характеристики впливу на бойовий процес: блок знижує шкоду, але поступово вичерпує витривалість; парирування відкриває вікно для потужного контратаки, проте вимагає суворого відчуття моменту атаки; ухилення дозволяє уникнути удару, але на короткий час залишає відкритим простір для контрзаходів ворога. Для реалізації цієї глибини буде передбачено поділ логіки бойового циклу на окремі підсистеми, які взаємодіють через події: підсистема гравця, підсистема анімації, підсистема обробки витривалості та підсистема атаки та отримання ушкоджень.

Третім напрямом роботи буде побудова прогресу персонажа через лінійне підвищення статистик із можливістю перерозподілу очок розвитку без ризику хибного інвестування. Для цього буде розроблено модуль управління характеристиками, який підраховує поточний рівень, доступні для інвестицій очок та їхню вартість із урахуванням зростання складності рівнів. Зміна вартості очок прив’язана до кривих зростання, що адаптуються до рівня персонажа: вищий рівень робить подальший прогрес дорожчим, але відкриває доступ до більшої потужності персонажа.

Фінальним етапом інтеграції буде поєднання всіх компонентів у єдиному середовищі Unreal Engine 5 за допомогою системи BVS. Завдяки цьому створюється можливість швидкого прототипування та тонкого налаштування кожного параметра без необхідності перезбірки проєкту: розробник може в реальному часі регулювати криві відновлення витривалості, коефіцієнти витрат очок розвитку, налаштування компонентів та змінних анімацій у зручному інтерфейсі. Такий підхід гарантує технічну стабільність ігрового процесу, підтримку потрібного рівня стабільності гри та відчуття досягнення, коли успіх у бою залежить від власної майстерності гравця.

1.4.1 Цільова аудиторія

До цільової аудиторії[5] ігрового застосунку «Catalous» належать:

- гравці, які віддають перевагу високому рівню виклику та відчутній складності бою;
- прихильники жанру «Action RPG» і піджанру «Souls-like», що цінують точність керування витривалістю та стратегічну глибину сутичок;
- користувачі, які прагнуть вдосконалювати власні навички й відчувати прогрес завдяки вправності, а не лише статистикам персонажа;
- люди віком від 16 до 35 років із інтересом до фентезі-тематики та естетики темних світів;
- гравці на платформах PC, зацікавлені в технічно стабільному й оптимізованому досвіді;
- досвідчені користувачі, знайомі з принципами «Souls-like», а також ті, хто бажає розпочати знайомство з жанром через інтуїтивно зрозумілу лінійну систему розвитку.

Таке поєднання характеристик аудиторії забезпечує як залучення фанатів важких ігор із репутацією, так і відкриває двері для новачків, котрі шукають насичений ігровий процес із відчуттям досягнення через власні вміння.

1.4.2 Монетизація

Монетизація «Catalous» базується на моделі преміального продажу базової версії гри з подальшим випуском необов’язкових доповнень. Початкова вартість гри у цифрових крамницях (Steam, Epic Games Store, PlayStation Store, Xbox Marketplace) встановлюється на рівні середньоринкового сегмента Action RPG, що дозволяє покрити початкові витрати на розробку та маркетинг, але при цьому залучає широку аудиторію.

Головним джерелом доходу стануть сюжетні та ігрові доповнення (DLC), що розширюють світ «Catalous» новими локаціями, босами та механіками. Кожне доповнення випускається із завершеною сюжетною лінією й унікальними випробуваннями Souls-like, що гарантує гравцям нові стимули до повернення в гру та сприяє зростанню часу життя проєкту. Продаж таких DLC попередньо анонсується в офіційних каналах спільноти, що підвищує рівень передзамовлень.

Крім прямих продажів, передбачено проведення сезонних розпродажів і знижок на набори DLC, які стимулюють приплив нових гравців і повернення ветеранів. Вартість кожного пакета налаштована так, щоб утримувати відчуття цінності придбань, водночас не перетворювати монетизацію на нав’язливу систему. Такий підхід забезпечує стабільний фінансовий потік, дозволяючи продовжувати підтримку та розвиток «Catalous» без шкоди для основного ігрового досвіду.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

У «Catalous» базова технологічна платформа вибрана з урахуванням потреб сучасного Action RPG: для розробки використовується Unreal Engine 5, що забезпечує високу якість графіки, гнучкі можливості налаштування бойової логіки через BVS-панель і просте масштабування під різні конфігурації ПК. Підтримка Windows 10 (64-bit) і новіших версій гарантує сумісність із більшістю ігрових систем, а паралельна розробка для консолей може бути розглянута на наступних етапах.

Інтерфейс «Catalous» побудований таким чином, щоб гравець завжди мав під рукою всю потрібну інформацію. У верхньому куті екрана відображається запас витривалості, зміна якого анімована у реальному часі відповідно до дій персонажа. Поруч розташовано індикатор здоров’я та поточний стан активної зброї, у нижній частині екрана — піктограми зілля та кількість накопиченої валюти. Система керування реалізована під клавіатуру і мишу.

Щоб уникнути зайвого навантаження на гравця і не відволікати від проходження, механіка перерозподілу очок розвитку активується лише біля спеціальних точок збереження. Після взаємодії з об’єктом збереження гравець може миттєво відкоригувати розподіл характеристик, налаштувавши персонажа під свій стиль бою. Супротивники, переможені до збереження, не відроджуються при завантаженні, а з’являються знову лише після того, як гравець повністю пройде рівень, що дозволяє сконцентруватися на основних викликах гри без рутини нескінченних битв.

Збереження прогресу організовано через локальні слоти з використанням контрольних сум для запобігання корупції даних. Кожен слот зберігає інформацію про позицію гравця на рівні, поточні характеристики персонажа, стан інвентаря та історію битв із босами. Такий підхід підвищує надійність системи збережень і мінімізує ризик втрати важливого прогресу.

Серед мінімальних апаратних вимог – сучасний чотирьохядерний процесор (Intel Core i5-6600K або AMD Ryzen 5 1400), 8 ГБ оперативної пам’яті, відеокарта рівня NVIDIA GeForce GTX 1060 6 GB або AMD Radeon RX 580 8 GB, підтримка DirectX 12 і не менше 30 ГБ вільного місця на жорсткому диску для швидкого завантаження текстур та рівнів.

Додатково, перед початком розробки ігрового застосунку в жанрі souls-like були визначені основні функції зі сторони гравця. Після детального аналізу була створена Use-case діаграма (див. рис. 2.1).

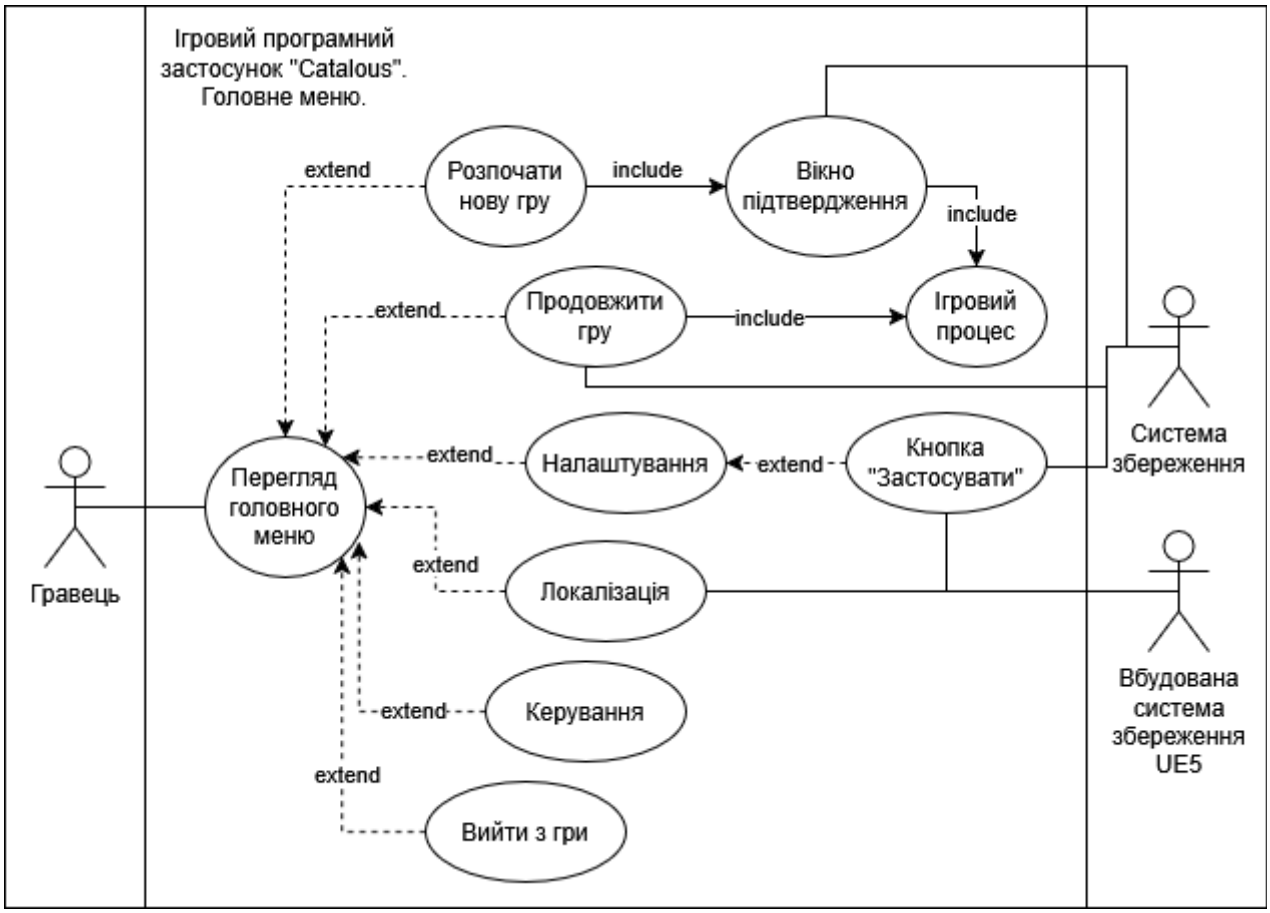


Рисунок 2.1 – Use-case діаграма головного меню (рисунок виконаний самостійно)

Гравець виступає центральним користувачем ігрового застосунку «Catalous», тісно взаємодіючі не тільки з ігровим світом, а й із внутрішньою системою збереження

прогресу. Весь шлях користувача починається з головного меню (див. розділ А.1 додатку А), яке виступає своєрідним порталом у гру та визначає подальші дії.

Перш за все, гравець може обрати опцію «Нова гра»: після підтвердження з’являється діалогове вікно, що повідомляє про видалення попереднього прогресу, і лише за згодою користувача система збереження очищує всі дані та створює перший запис для нового проходження.

Якщо ж гравець бажає продовжити вже розпочате проходження, потрібно вибрати пункт «Продовжити гру». У цьому випадку система збереження автоматично завантажує найсвіжіший слот із попередніми досягненнями: позицією на карті, рівнем персонажа, інвентарем та налаштованими навичками. Завдяки цьому інтерфейсу швидкого доступу користувач може повернутися до гри за лічені секунди без додаткових підтверджень.

Окремий блок налаштувань дозволяє гравцеві детально керувати параметрами звуку та зображення: регулювати гучність музики і звукових ефектів, змінювати розмір і співвідношення екрану, увімкнути або вимкнути вертикальну синхронізацію, а також встановити загальний рівень графічної якості та окремі компоненти, зокрема тіні, деталізацію текстур і якість ефектів.

Меню локалізації містить перемикач між англійською та українською мовами інтерфейсу. Кожна зміна відразу записується через UE5 Localization Manager, тому після перезапуску гра знову виводитиме текст і підказки у вибраній мові.

Крім взаємодії з головним меню, було деталізовано поведінку гравця під час роботи з точками збереження в ігровому світі. Саме тут користувач може не лише відпочити й відновити здоров’я, а й скористатися меню покращення, телепортації або швидкого виходу на хаб. Детальна схема всіх можливих дій гравця винесена в Use-case діаграму (див. рис. 2.2), яка відображає ролі та взаємодію системи збереження з іншими підсистемами проєкту.

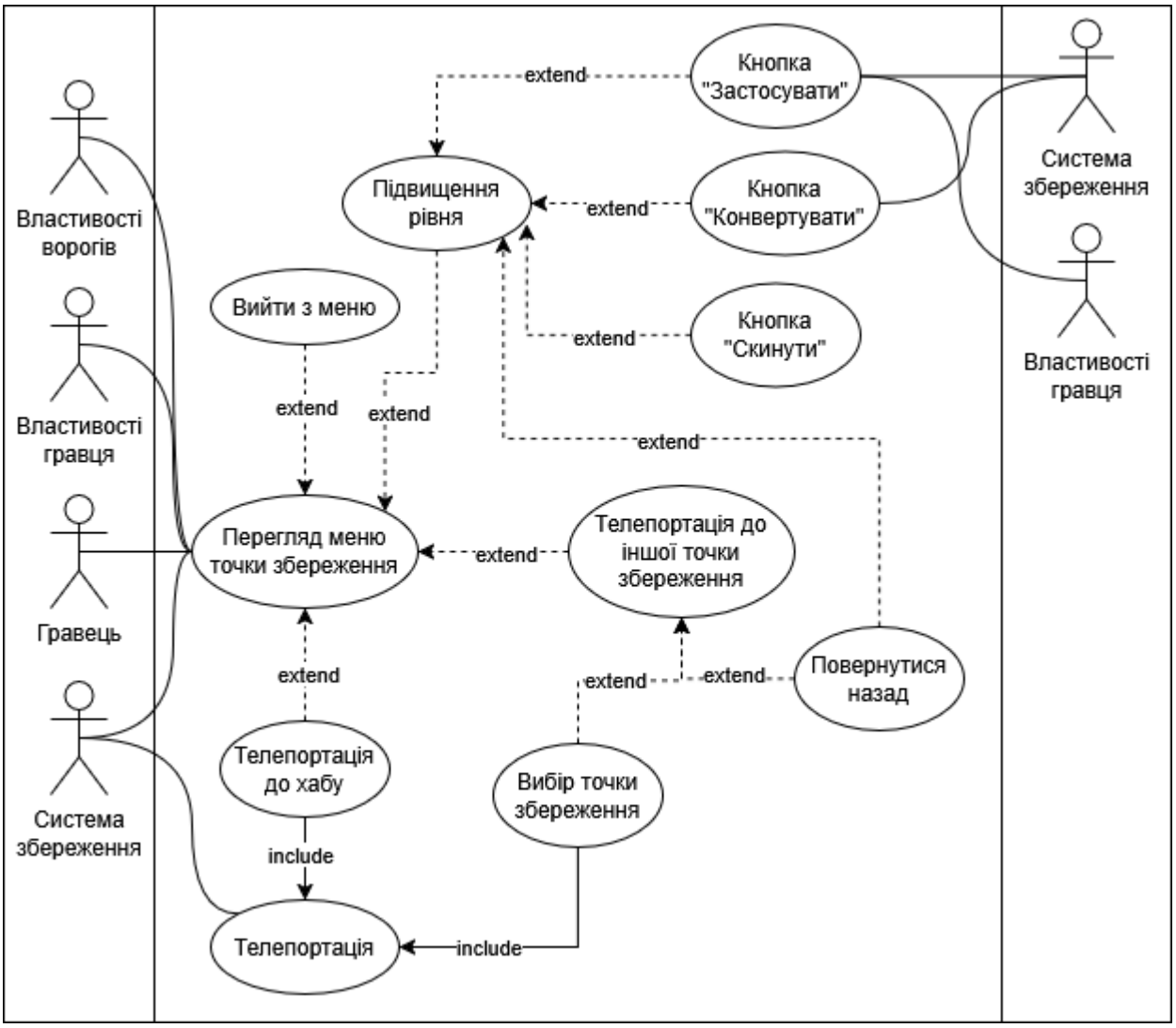


Рисунок 2.2 – Use-case діаграма меню точки збереження (рисунок виконаний самостійно)

У точках збереження відкривається спеціальне меню (див. розділ А.9 додатку А), яке забезпечує гравцю доступ до всіх ключових інструментів керування прогресом. При вході в це меню користувач може обрати повернення до ігрового світу без жодних змін або перейти до доступних підменю, що дозволяють коригувати розвиток персонажа та навігацію по рівнях. Кожна дія в інтерфейсі супроводжується оновленням системи збережень, що гарантує надійність фіксації всіх змін.

Меню підвищення рівня активується в тому ж вікні, де користувач бачить поточні характеристики та доступну валюту. За допомогою кнопки «Конвертувати»

вся накопичена валюта перетворюється на очки покращення, після чого відразу відбувається автоматичний запис у файл збереження. Далі гравець отримує можливість розподілити ці очки між наданими статистиками, обираючи ті атрибути, які найкраще відповідають його стилю проходження. Після внесення змін кнопка «Застосувати» підтверджує їх і оновлює параметри персонажа, а «Скинути» повертає початковий стан точок покращення, дозволяючи переглянути альтернативні варіанти розвитку. Повернення до головного меню точки збереження здійснюється через клавішу «Назад», без втрати вже зафіксованих даних.

Меню телепортації (див. розділ А.10 додатку А) надає список доступних локацій-контрольних точок у вигляді умовних мініатюр. Вибір будь-якої з цих мініатюр одразу змінює місцезнаходження персонажа в ігровому світі, після чого система збережень фіксує нову позицію. Якщо гравець вирішує залишити меню телепортації без переміщення, натискання «Назад» повертає його до головного екрану точки збереження.

Окремою опцією у головному вікні є телепортація до хабу — головної точки збереження, через яку проходять усі гравці на початку своєї подорожі. Використання цієї функції миттєво переносить персонажа до зони хабу та викликає оновлення збережень, що гарантує коректність усіх параметрів після зміни локації. Така архітектура меню точки збереження забезпечує швидкий і зручний доступ до механік відновлення, покращень та навігації, зберігаючи при цьому послідовність і безпеку ігрового процесу.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Були визначені основні функції зі сторони гравця. Після детального аналізу[6][7] була створена «State» діаграма (див. рис. 3.1), діаграма систем, що використовують актори (див. рис. 3.2) та діаграма взаємодії цих систем між собою.

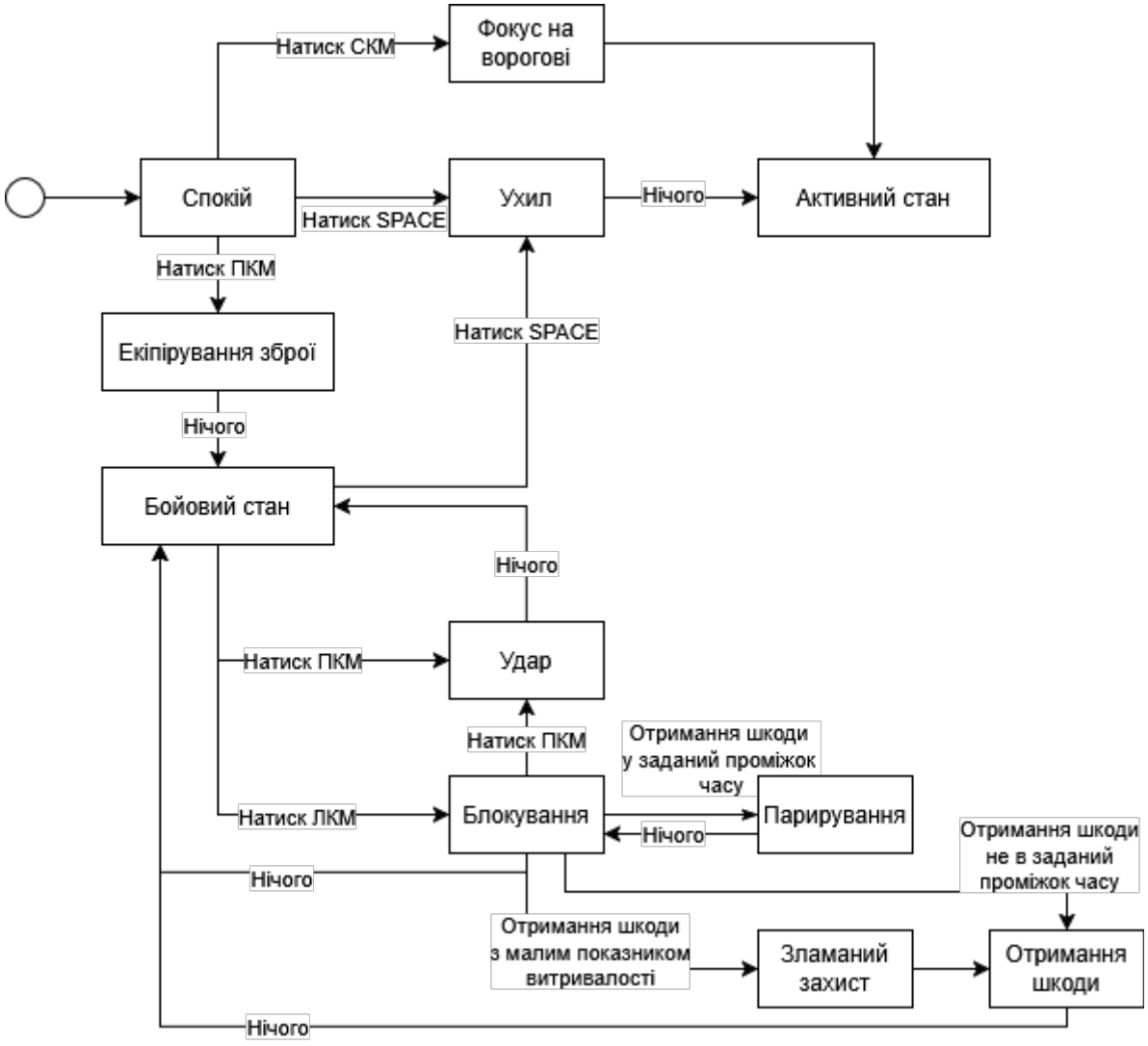


Рисунок 3.1 – State діаграма бойової системи (рисунок виконаний самостійно)

Під час гри користувач має здатність перейти у бойовий стан (див. рис. 3.1), в котрому він здатен:

– ухилятися (доступно і без бойового стану);

– атакувати;
– блокувати;
– парирувати, якщо блокування було активоване у заданий проміжок часу (шкода не отримується);
– захист може бути зламаним, якщо під час блокування була отримана шкода без достатньої кількості витривалості (шкода отримується);
– захист із достатньою кількістю витривалості призведе до отримання меншої кількості шкоди та зменшення витривалості.

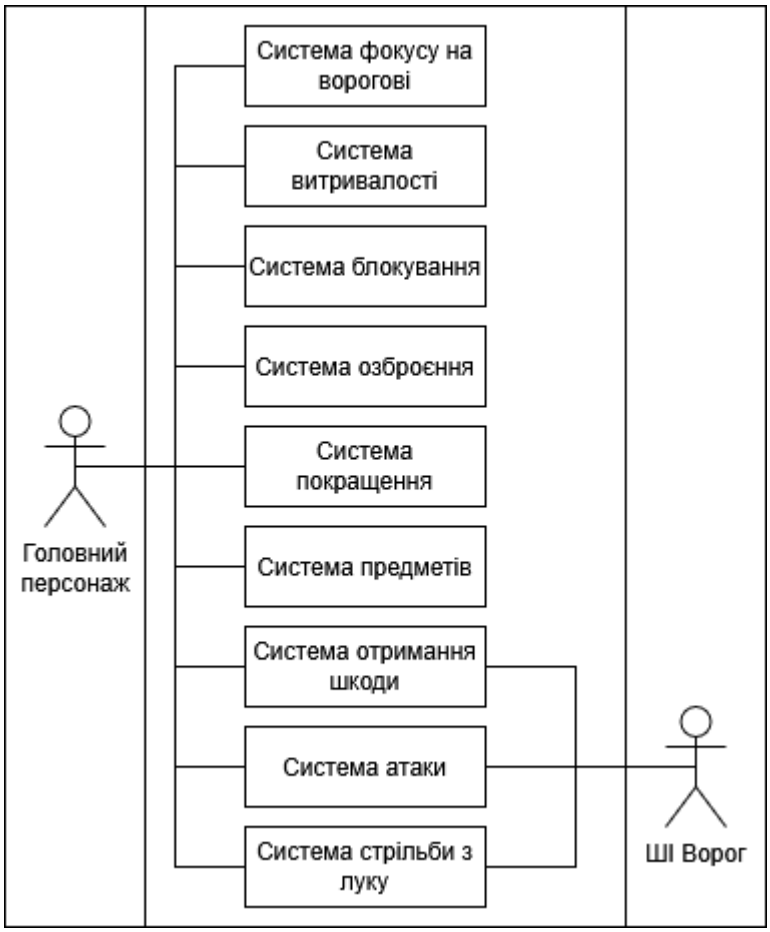


Рисунок 3.2 – Діаграма компонентів персонажів (рисунок виконаний самостійно)

При розробці персонажів було створено низку взаємопов’язаних компонентів (див. рис. 3.2), що забезпечують гнучкість і глибину бойової системи. Поєднання механік фокусу, витривалості та блокування дозволяє гравцеві точно контролювати

позицію та стани персонажа: система фокусу на ворогові активується натисканням правої кнопки миші й утримує камеру на цілі, що допомагає зосередитися на супротивнику в найінтенсивніші моменти бою. Паралельно динамічна система витривалості регулює витрати витривалості під час атак, ухилень та блоків, а статичні параметри витривалості дають уявлення про максимальний запас і швидкість відновлення ресурсу після коротких пауз.

Удосконалена система блокування та отримання шкоди працює в тандемі: при активації блокування (СКМ) враховується не тільки поточний стан витривалості, але й кут атаки ворога, що дозволяє реалізувати поступове зменшення шкоди або зовсім її відхилення у разі успішного парирування. Детальна модель отримання ушкоджень аналізує комбінації дій — блок, ухилення, парирування — і коригує величину заподіяних пошкоджень з урахуванням броні, використаних перстнів та інших бонусів, що надають обладунки чи навички.

Система атаки та стрільби з лука виводить бойову взаємодію на новий рівень: завдяки модульній архітектурі підсистеми атаки персонаж може виконувати ланцюжки з кількох ударів із різних видів зброї, враховуючи швидкість анімації та час відновлення після кожного руху. Стрілянина з лука реалізована окремим компонентом, що вимірює тривалість натягнення тятиви й автоматично масштабує завдану шкоду залежно від сили пострілу. Обидві системи — і ближнього бою, і стрільби — використовуються як головним персонажем, так і штучним інтелектом ворогів, що гарантує збалансованість та передбачуваність сутичок.

Механіки інвентаря та озброєння інтегровані в єдину систему управління ресурсами: гравець може носити з собою кілька видів зброї, боєприпаси, перстні та витратні предмети, перемикаючись між ними в реальному часі. Компонент підбору предметів автоматично обробляє події взаємодії з об’єктами оточення – підбирання ліків, зброї чи матеріалів – і додає їх до інвентарю з урахуванням обмежень ваги чи кількості.

Система покращення (див. рис. 3.3) забезпечує лінійний розвиток персонажа: після конвертації валюти в очки покращення гравець розподіляє їх між базовими характеристиками, користуючись меню біля точки збереження. Модуль управління характеристиками стежить за поточним рівнем, перераховує вартість наступного очка розвитку та дозволяє виконати скидання розподілу без шкоди для балансу.

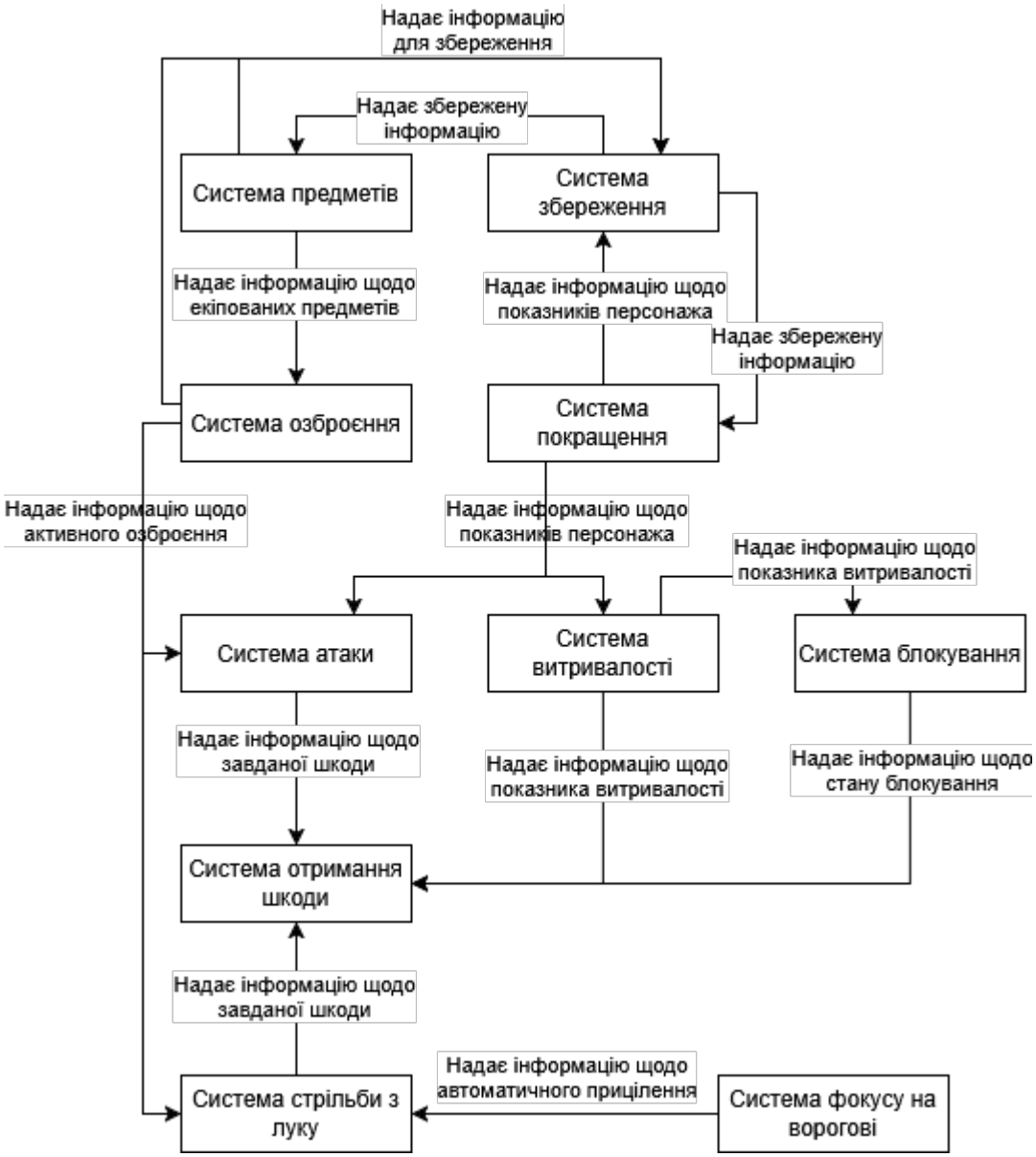


Рисунок 3.3 – Діаграма зв’язку компонентів (рисунок виконаний самостійно)

Усі компоненти взаємодіють через інтерфейси UE5, що дає змогу швидко прототипувати й налаштовувати їх через BVS-панель без перезбирання проєкту.

При розробці компонентів між ними були створені наступні зв’язки (див. рис. 3.3):

– система збереження надає збережену інформацію системі предметів та покращення відповідно для ініціалізації актуального озброєння гравця та встановлення покращених характеристик персонажа;
– система предметів пов’язана з системою озброєння, надаючи їй інформацію щодо обраного гравцем озброєння (назва, ідентифікатор, текстури тощо) та з системою збереження, надаючи актуальні предмети гравця для запису у файл збереження;
– система покращення пов’язана з системою збереження, надаючи їй інформацію про актуальні характеристики персонажа для запису у файл збереження. Пов’язана з системою атаки, надаючи інформацію щодо показників персонажа, котрі випливають на силу атаки персонажа. Також пов’язана з системою витривалості, надаючи їй актуальний показник витривалості персонажа;
– система озброєння пов’язана з системою збереження, надаючи актуальне озброєння гравця для запису у файл збереження. Пов’язана з системою атаки, надаючи інформацію щодо поточного озброєння гравця. Також пов’язана з системою стрільби з луку, надаючи інформацію щодо поточного озброєння гравця, включаючи боєприпаси;
– система атаки пов’язана з системою отримання шкоди, надаючи інформацію щодо завдання шкоди персонажу (кількість шкоди, стан персонажа тощо);
– система витривалості пов’язана з системою блокування, надаючи інформацію щодо поточного значення витривалості персонажа для визначення чи здатен гравець використовувати блокування. Також пов’язана з системою отримання шкоди, надаючи інформацію щодо поточного значення витривалості для обчислення результату гравцем отримання шкоди;

– система блокування пов’язана з системою отримання шкоди для надання інформації щодо стану блокування гравця в момент часу;
– система стрільби з луку пов’язана з системою отримання шкоди, надаючи інформацію щодо завданої шкоди, враховуючи час натягнення тятиви луку;
– система фокусу на ворогові пов’язана з системою стрільби з луку, визначаючи тип прицілення гравця (автоматичний/ручний).

3.2 Проєктування архітектури ПЗ

Архітектура «Catalous» базується на модульному підході, при якому ядро гри реалізовано на Unreal Engine 5. UE5 відповідає за генерування графіки, фізичні обчислення та відтворення анімацій; усі основні підсистеми – від системи зіткнень до симуляції частинок – інтегровані в єдине середовище, що гарантує узгодженість та синхронність станів персонажа, ворогів та оточення.

Для побудови користувацького інтерфейсу використовується UMG, що дозволяє створювати гнучкі елементи HUD, меню точок збереження, вікна поліпшень і телепортації. Взаємодія користувача з інтерфейсом обробляється через події (event dispatchers), які активують відповідні методи в модулях збереження, розвитку персонажа та навігації. Завдяки BVS-панелі ці події можна прототипувати в реальному часі, змінюючи параметри без перезбирання проєкту та негайно бачити результат у грі.

Спілкування між підсистемами відбувається через інтерфейси (Interface), що задають контракт на надсилання подій і отримання результатів. Наприклад, під час взаємодії з точкою збереження система інтерфейсу «ICheckpoint» передає дані про поточний стан гравця до «SaveManager», який формує структуру «SaveData» і записує її в локальний файл із контролем контрольної суми.

Крім того, користувачеві надається доступ до розділу графічних налаштувань, де можна змінювати якість текстур, розмір буферів освітлення та рівень деталізації тіней, що дозволяє адаптувати «Catalous» під широкий спектр обладнання.

Завдяки такій архітектурі забезпечується швидке й безпечне прототипування нових механік, просте обслуговування коду та масштабування проєкту. Додавання нових механік або правок у бойову систему, AI чи систему розвитку персонажа не вимагає глибоких інтеграційних змін: достатньо реалізувати новий клас із відповідним інтерфейсом і налаштувати його поведінку через BVS, що значно скорочує час розробки та знижує ризики регресії.

3.3 Проєктування структури зберігання даних

Під час розробки системи зберігання, було створено 3 різні типи зберігання, а саме: збереження даних гравця, збереження користувацьких налаштувань, збереження графічних налаштувань, збереження налаштувань керування, збереження локалізації. Після детального аналізу були створені діаграми (див. рис. 3.4).

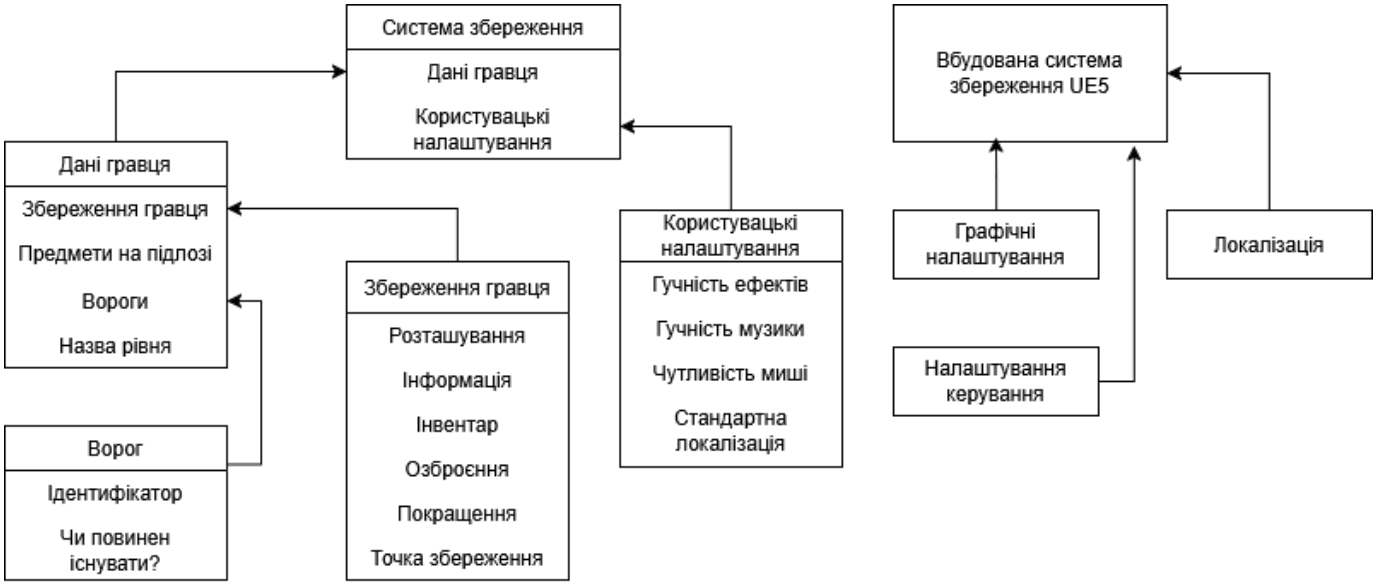


Рисунок 3.4 – Діаграма системи збереження (рисунок виконаний самостійно)

Вбудована система збереження UE5 [8][9] забирає на себе роботу щодо збереження графічних налаштувань, налаштування керування та активної локалізації користувача. Тоді як окремо побудована система збереження зберігає наступні дані:

- дані гравця, що містять елементи наведені на рисунку 3.4, де «Збереження гравця» це структура, що зберігає розташування гравця (локація, ротація, розмір), інформація (назва, швидкість бігу, швидкість ходьби, здоров'я, витривалість тощо), інвентар (зброя, розхідні предмети, боєприпаси, перстні – всі є відповідними структурами), озброєння (слоти для зброї, боєприпасів, перстнів та розхідних предметів, їх відповідні активні індекси та їх відповідні текстури), покращення (рівень, очки покращення, валюта, сила, спритність, інтелект, витривалість) та точка збереження (розташування точки збереження, розташування точки відродження, назва та ідентифікатор);
- користувацькі налаштування, що містять елементи наведені на рисунку 3.4.

3.4 Приклади найцікавіших алгоритмів та методів

3.4.1 Алгоритм трасування зброї

Оскільки алгоритм був розроблений з використанням BVS, замість коду будуть наведені діаграми (див. рис. 3.5).

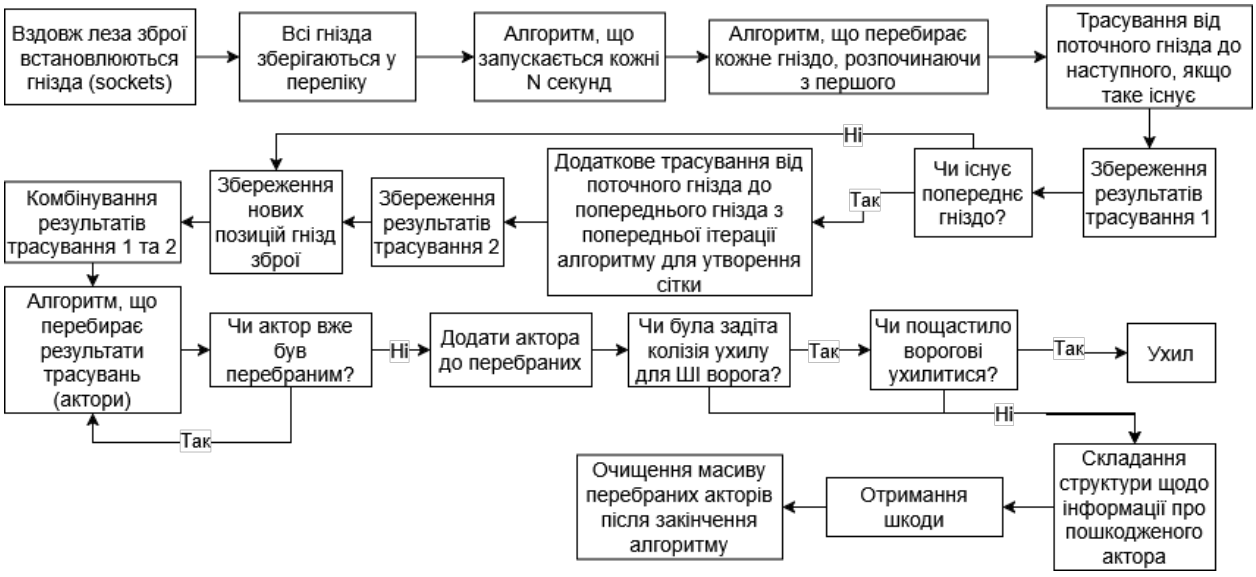


Рисунок 3.5 – Діаграма алгоритму трасування зброї (рисунок виконаний самостійно)

Після виконання алгоритму наведеного на рисунку 3.5, утворюється трасування зброї у формі сітки.

Додатково, «структура щодо інформації про пошкодженого актора» складається з наступних елементів:

– актор, що задав шкоди;
– оброблене значення шкоди (враховуючи характеристики персонажа та зброї);
– «чисте» значення шкоди;
– логічне значення чи може атаки бути парируваною;
– логічне значення чи може атака бути заблокованою;
– логічне значення чи може атака завдати шкоди крізь ухиляння;
– напрямок, з котрого була виконана атака;
– логічне значення чи був атакований актор у стані блокування;
– логічне значення чи був атакований актор у стані ухилу;
– логічне значення чи був атакований актор у стані парирування.

3.4.2 Метод обчислення кількості необхідної валюти для наступного рівня

Для обчислення кількості необхідної валюти використовуються дві формули. Якщо рівень персонажа нижчий за 15, то використовується формула 3.1.

$$floor(CL * SCOP * 0.832) + SCOP \quad (3.1)$$

де *CL* – поточний рівень гравця;

SCOP – початкова ціна за рівень;

floor – функція, що округлює число в меншу сторону.

Якщо рівень гравця вищий або дорівнює 15, то використовується більш складна формула 3.2.

$$floor(CL^3 * 0.02 + CL^2 * 2.06 + CL * 85.6 - 895) \quad (3.2)$$

де *CL* – поточний рівень гравця;

floor – функція, що округлює число в меншу сторону.

Таким чином, запропоновані формули забезпечують плавне й поступове ускладнення процесу покращення персонажа: на початкових рівнях (нижче 15) вартість підвищення визначається простою лінійною залежністю (формула 3.1), що дозволяє гравцям відчути швидкий старт і швидке зростання потужності. Водночас для рівнів 15 і вище застосовується складна кубічна формула (3.2), у якій внесок кожного наступного рівня зростає значно швидше завдяки додаванню степеневих коефіцієнтів та констант.

У практичному сенсі це означає: чим вищий поточний рівень гравця, тим більше ресурсів (валюти чи очок покращення) знадобиться для чергового підвищення. На ранніх етапах гравці отримують відчуття швидкого прогресу — кожен рівень коштує відносно небагато, що мотивує до проходження та знайомства зі світом. Але згодом, як тільки рівень досягає 15, вартість прокачування починає зростати за крутою кубічною кривою. Це дозволяє ефективно балансувати ігровий процес: досягти надзвичайно високих характеристик стає надзвичайно витратним із точки зору ігрових ресурсів і часу.

Такий підхід запобігає персонажам з надмірною силою на ранніх стадіях, а також дає змогу розробникам точно регулювати темп прогресу. Завдяки лінійній формулі на початку гравці не відчувають надмірної складності, а кубічна функція вищих рівнів гарантує, що створити «занадто сильного» персонажа без значних зусиль практично неможливо. В результаті ігровий процес залишається збалансованим та цікавим протягом усього проходження, а кожне нове підвищення рівня відчувається гідною нагородою за попередні досягнення.

3.4.3 Метод виконання атаки ШІ-ворога

Оскільки алгоритм був розроблений з використанням BVS, замість коду будуть наведені діаграми. Додатково, структура «ADRPD» (Attack-Delay-Rest-Probability-Distance), що визначає атаку ШІ-ворога складається з наступних елементів:

- перелік анімацій та затримок всередині відповідних анімацій;
- відпочинок після виконання всього переліку атак;
- ймовірність виконання відповідного переліку атак;
- дистанція проведення відповідного переліку атак, котра обчислюється та множиться на вище згадану ймовірність (формула 3.3).

$$bool\_with\_weight\left(P * \left(1 - \left|\frac{D}{DT} - 1\right|\right)\right) \tag{3.3}$$

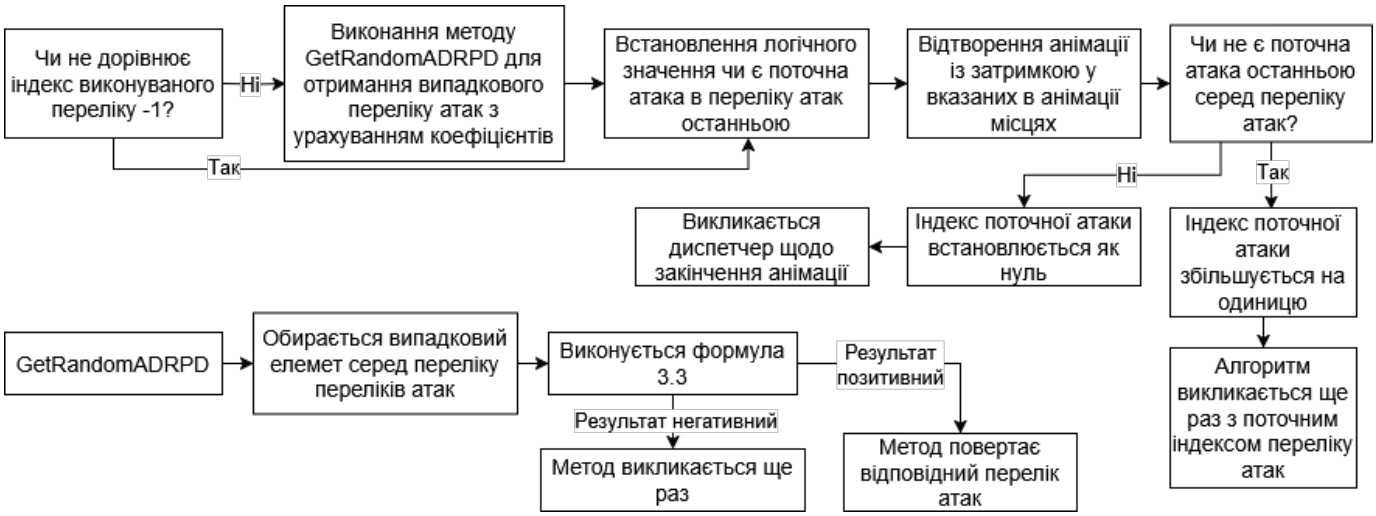
де *bool_with_weight*[10] – вбудований метод UE5, що повертає випадкове логічне значення в залежності від переданого значення (від 0 до 1 – від 0% до 100% відповідно);

P – ймовірність виконання переліку атак;

D – бажана дистанція до гравця;

DT – дистанція до гравця.

У контексті ADRPD такий підхід дозволяє створити адаптивний режим бою: штучний інтелект перевагу віддає атакам, для яких він знаходиться на «правильній» відстані, але іноді може обрати не найоптимальніший варіант, підтримуючи елемент непередбачуваності. Це також сприяє тому, що комбінація ймовірності та дистанційного фактору породжує збалансовану та живу поведінку ворогів: гравець не може точно передбачити кожний хід ШІ, але відчує, що атаки вибираються розумно і відповідають ситуації на полі бою.



3.6 – Діаграма алгоритму атаки ШІ-ворога (рисунок виконаний самостійно)

На рисунку 3.6 представлено детальну схему логіки атак ШІ-ворога, а також алгоритм методу «GetRandomADRP», який відповідає за вибір випадкового переліку анімацій атак з урахуванням імовірностей та відстані до гравця. Спочатку перевіряється чи не дорівнює індекс поточної атаки значенню –1 (стан, коли атака ще не обрана). Якщо індекс рівний –1, викликається метод «GetRandomADRP», інакше логіка переходить безпосередньо до відтворення анімації.

Метод «GetRandomADRP» розпочинає з випадкового вибору елемента серед доступних переліків атак, кожен з яких оцінений за допомогою формули 3.3. Ця формула враховує коефіцієнт відстані між ШІ та гравцем, базову імовірність атаки й поточний стан ресурсу стаміни, що дозволяє динамічно налаштовувати вагу кожного варіанту. Якщо результат формули позитивний, метод повертає відповідний перелік атак як об’єкт для виконання; якщо ж ні – алгоритм знову викликає себе того ж самого списку, доки не буде обрано придатний варіант.

Після отримання індексу атаки відбувається встановлення логічного прапорця “поточна атака активна”, і відразу запускається відтворення анімації з урахуванням штучної “затримки” в точках, визначених у секвенції анімацій. Цей затриманий

проміжок вводить елемент непередбачуваності під час атаки, ускладнюючи гравцеві завдання парирування. По завершенні анімації система перевіряє, чи була це остання атака в поточному переліку: якщо так – індекс оновлюється і цикл починається знову, якщо ні – індекс збільшується на одиницю і процес повторюється.

Кожен крок взаємодіє з підсистемою подій UE5: при старті та завершенні анімації викликаються відповідні диспетчери, які взаємодіють з іншими компонентами логіки бою про зміну стану ШІ. У сукупності ця схема гарантує безперервний пошук «найбільш правильного» способу нападу, адаптованого до поточної ситуації, та забезпечує належний рівень виклику для гравця.

3.5 Створення UI/UX

Весь користувацький інтерфейс "Catalous" створено за допомогою інструментарію UMG, вбудованого в Unreal Engine 5. Цей підхід дозволив розробникам створити гнучкі та модульні елементи інтерфейсу, мінімізувавши необхідність у написанні складних C++ класів. Інтерфейс відрізняється адаптивністю: використані шрифти та розмітка автоматично коректно масштабуються під різні роздільні здатності екранів. Це забезпечує зручність використання як на широкоформатних моніторах, так і на екранах з меншою площею, наприклад, ноутбуках.

З метою забезпечити доступність для глобальної аудиторії, всі текстові елементи UI повністю локалізовано українською та англійською мовами. Для ефективного управління перекладами було впроваджено власний менеджер рядків. Він значно зменшує дублювання текстових ресурсів і надає технічну можливість легко додати підтримку нової мови в будь-який момент розробки або після виходу гри.

Головне меню гри інтуїтивно згрупувало ключові функції: запуск нової гри, завантаження збережень та вихід з програми. Його візуальна подача підкріплена плавними анімаціями появи та зникнення UI-елементів, що створює приємне перше враження. Меню паузи, яке активується клавішею «Escape», служить центральним

вузлом для швидкого доступу до системних функцій. Воно надає безпосередній перехід до збереження прогресу, перегляду інвентарю та екіпірування персонажа, налаштувань гри та повернення на головний екран.

Меню інвентарю та екіпірування реалізовано через динамічні списки UMG. Ця система автоматично адаптує розмір та розкладку інвентарних осередків під колекцію предметів гравця. Віджет постійно синхронізований з ігровою системою інвентарю – під час підбору чи використання предметів, або при зміні зброї, його вміст миттєво оновлюється без затримок.

Особливу увагу приділено дизайну меню точки збереження. Воно інтегрує дві ключові функції у формі вкладених екранних форм. Перша форма – меню підвищення рівня – дає гравцеві свободу конвертувати ігрову валюту в очки покращення, розподіляти їх між характеристиками та експериментувати з покращенням персонажа, маючи можливість скасувати зміни без перезавантаження локації. Друга форма – меню телепортації – наочно відображає доступні для переходу локації через мініатюрні зображення-прев'ю. Такий комплексний підхід до проектування UI дозволив створити для "Catalous" зручний, динамічний та легко локалізований інтерфейс, що відповідає сучасним стандартам користувацького досвіду.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Створення системи задання та отримання шкоди

Система завдання та отримання шкоди працює через два компоненти: «Система атаки» та «Система отримання шкоди» (див. рис. 3.4). Оскільки головний метод системи атаки був розглянутий (див. рис. 3.6), розглянемо головні методи системи отримання шкоди, а саме:

– сценарій шкоди. Визначає сценарій за котрим була отримана шкода власником системи (отримання шкоди, блокування, парирування, ухил). Потребує інформацію щодо завданої шкоди (див. пункт 3.4.1);
– отримання шкоди. Викликає диспетчери в залежності від сценарію шкоди, обчислюючи нове значення здоров'я власника системи. Потребує інформацію щодо завданої шкоди;
– відновлення здоров'я. Відновлює здоров'я власнику системи. Потребує кількість здоров'я для відновлення (число/відсоток).

Оскільки алгоритм був розроблений з використанням BVS, замість коду будуть наведені діаграми. Таким чином, на рисунку 4.1 можна побачити логіку методу обчислення сценарію шкоди. Окрім того, обчислення напряму завдання шкоди виконується за формулою 4.1.

$$v1 = Normalize2D(AL - TACL), v2 = Normalize2D\big(RightVector(TACR)\big) \quad (5.1)$$
$$v3 = Normalize2D\big(ForwardVector(TACR)\big), v4 = atan2(dot(v1, v2), dot(v1, v3))$$

де *AL* – локація актора, котрий завдає шкоди;

TACL, TACR – локація та ротація камери актора, котрому завдається шкоди;

Normalize2D[11] – вбудована функція UE5, що нормалізує вектор;

RightVector[12]*, ForwardVector*[13] – повертають вектори об'єкту;

dot – вбудована функція UE5, що виконує скалярний добуток А та Б;

atan2 – вбудована функція UE5, що повертає арктангенс А/Б.

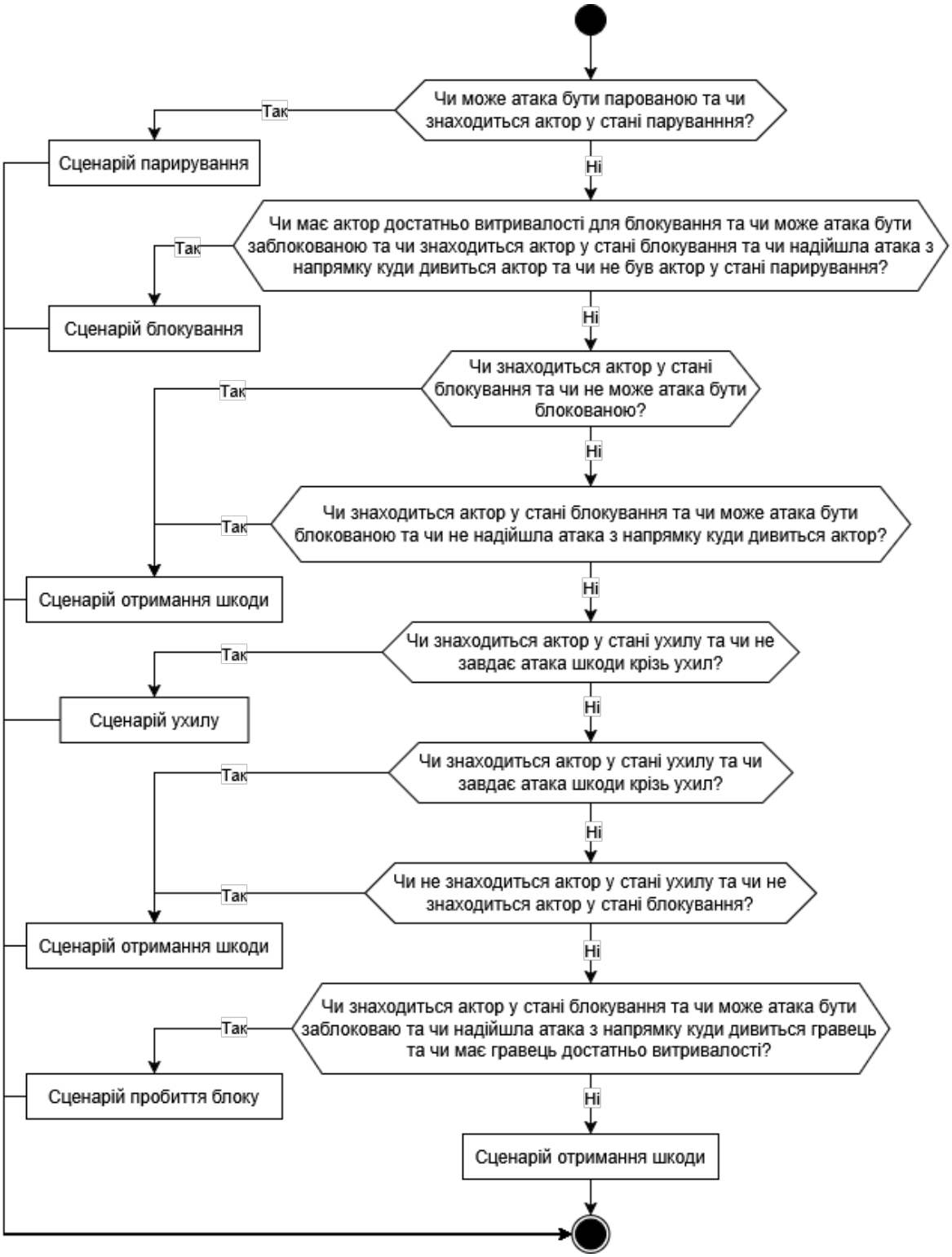


Рисунок 4.1 – Діаграма методу визначення сценарію шкоди (рисунок виконаний самостійно)

Подібна система точно оброблює сценарії завдання/отримання шкоди.

4.2 Створення системи збереження

Оскільки система збереження вже була частково розглянута (див. рис. 3.4 та 3.5), то розглянемо опис методів, котрими володіє ця система:

- під час ініціалізації, система завантажує збереження гравця та його налаштувань у синхронному режимі;
- метод зчитування збереження виконується шляхом використання вбудованих функцій UE5, зберігаючи значення зчитаного збереження. Має можливість виконуватися асинхронно;
- метод запису збереження виконується шляхом використання вбудованих функцій UE5, зберігаючи час проведення операції для подальшого сповіщення гравця;
- метод збереження предметів на рівні та ворогів виконується шляхом додавання всіх акторів до словнику, де значення – логічне значення чи повинен актор існувати;
- відновлення ворогів виконується при взаємодії з точкою збереження та змушує ворогів повернутися на їх початкову позицію, відновити здоров’я, використовуючи система отримання шкоди (див. пункт 4.1) та приховати користувацький інтерфейс;
- під час ініціалізації головного персонажа, системи предметів, озброєння та покращення (див. рис. 3.4) використовують збережене значення для імпортування відповідних значень;
- під час збереження в точці збереження, системи предметів, озброєння та покращення експортують свої значення до системи збереження.

4.3 Створення шаблону атак ворогів

Оскільки логіка атаки ШІ-ворогів вже була детально розглянута (див. рис. 3.7), розглянемо дерево поведінки одного з ворогів, що працює завдяки дереву поведінки (див. рис. 4.2).

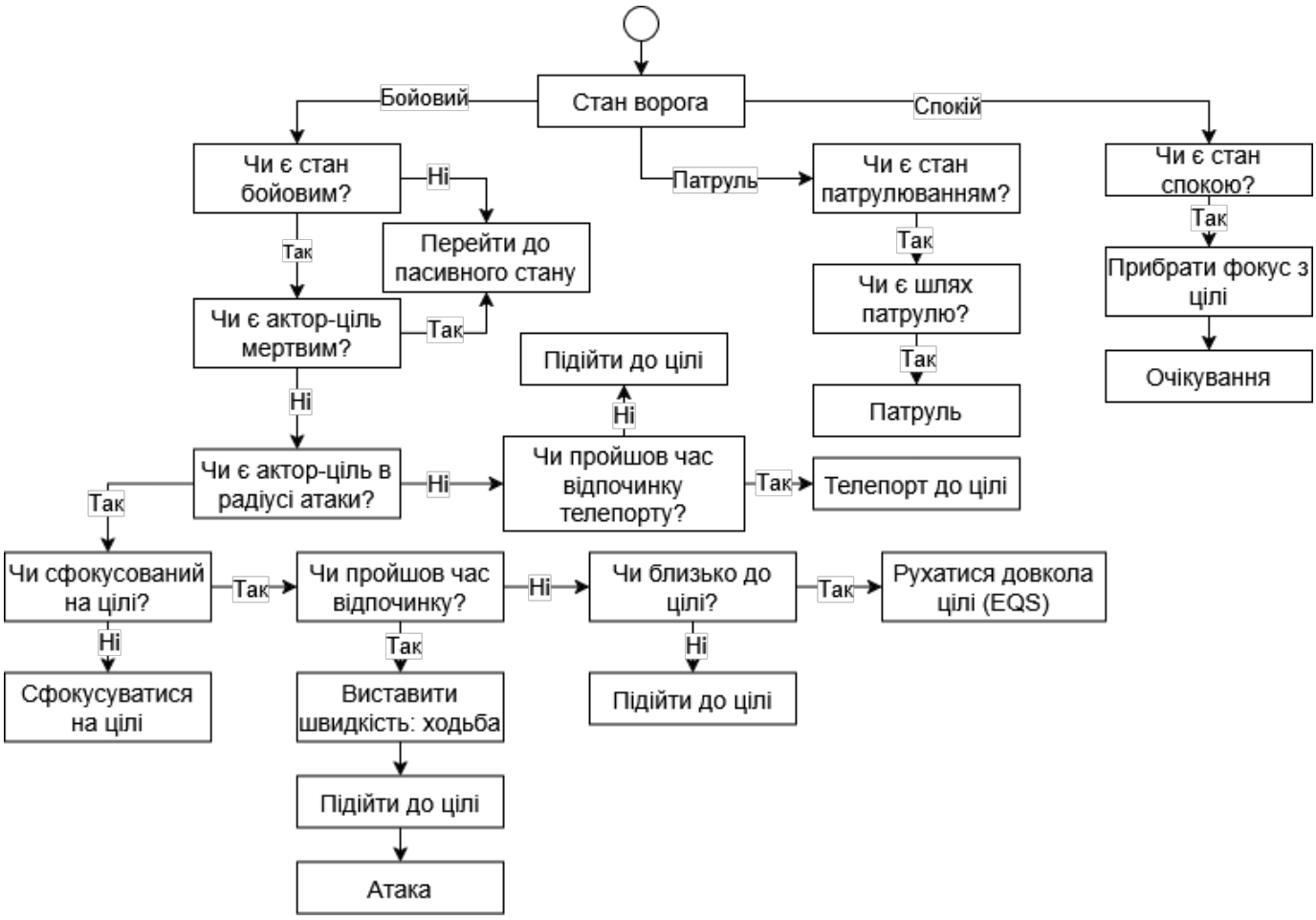


Рисунок 4.2 – Діаграма дерева поведінки одного з ворогів

Згадана в рисунку 4.2 EQS[14] це система, що відповідає за визначення «правильної» позиції для розташування ШІ-ворога відповідно до поточного розташування у світі гравця.

4.4 Створення системи покращення персонажа

Система покращення персонажа дозволяє користувачу покращувати силу, спритність, витривалість або ж інтелект персонажа. Для поліпшення персонажа використовуються очки покращення (див. пункт 3.4.2). Характеристики персонажа мають наступний вплив на ігровий процес:

- підвищення сили сприяє підвищенню здоров'я та показника сили персонажа. Кожна зброя може мати «посилювачі», ефект котрих прямо пропорційно залежить від показників персонажа. Посилювачі можуть бути наступними: E, D, C, B, A, S, з котрих E – найслабший посилювач та S – найсильніший. Таким чином, якщо меч із показником шкоди 50 матиме посилювач на силу A, то показник сили персонажа буде впливати на показник шкоди меча, посилюючи його від 50 до більшого значення, залежно від показника сили персонажа. Окрім того, зброя може мати обмеження – мінімальні характеристики для використання повного потенціалу зброї. Якщо зброя з показником шкоди 50 має обмеження в 10 очок сили, але персонаж має меншу кількість, зброя буде завдавати 25 шкоди;
- підвищення витривалості збільшує витривалість персонажа та показник витривалості. Від цього показнику залежить наскільки довго персонаж буде здатен виконувати дії під час бійки (атака, блокування, біг тощо) не знаходячи себе в ситуації, коли закінчується витривалість. Не має впливу на «посилювачі» або обмеження;
- підвищення спритності підвищує показник спритності персонажа. Впливає на «посилювачі» зброї та її обмеження;
- підвищення інтелекту підвищує показник інтелекту персонажа. Впливає на «посилювачі» зброї та її обмеження.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Тестування програмного забезпечення

Тестування ігрового застосунку «Catalous» було зосереджене на комплексному перевірянні всіх ключових механік у реальних ігрових умовах, проте не передбачало використання сторонніх фреймворків чи розширених методів автоматизації.

Перший етап полягав у проходженні тестових сценаріїв безпосередньо в UE5, де кожна нова версія збірки запускалася в режимі Play-In-Editor для перевірки базових функцій: від коректності витрат і відновлення витривалості до обробки взаємодії з точками збереження. Результати тестування фіксувалися в нотатках, де зазначалися параметри дії, очікуваний результат і фактичний стан після виконання дії.

Другий етап охоплював інтеграційні перевірки: під час ігрових сесій із різною зброєю та стилями бою рівні запускалися в різних комбінаціях, щоб упевнитися в стабільності взаємодії бойової логіки, AI ворогів та UI-інтерфейсу.

Особлива увага приділялася сценаріям з інтенсивними серіями атак та ухилень, адже саме в таких умовах виявляються найчастіші артефакти анімацій та колізії влучання. Виявлені помилки коригувалися в Blueprint-скриптах та через BVS-панель одразу перевірялися заново.

У майбутньому планується інтегрувати базові юніт-тести для ключових алгоритмів бойової системи та автоматизовані перевірки критичних сценаріїв через CI/CD-конвеєр UE5, щоб прискорити виявлення регресій і знизити залежність від ручного проходження.

5.2 Ручне тестування

У процесі тестування ігрового застосунку «Catalous» більшість перевірок виконувалася вручну без залучення сторонніх фреймворків. Кожну нову збірку запускали в режимі «Play-In-Editor», де імітували типові сценарії бою: зміни параметрів витривалості, різні комбінації атак та парирування, стрільбу з лука,

телепортацію, збереження, а також взаємодію з точками збереження. У ході таких сесій усі функції – від системи завдання та отримання шкоди до механіки перерозподілу очок розвитку – піддавалися багаторазовому випробуванню з різним озброєнням, підсиленнями та різними типами ворогів, що давало змогу виявити як очевидні, так і нетривіальні помилки в грі.

Усі виявлені неполадки досліджувалися крок за кроком із використанням вбудованого інструменту «Blueprint Debugger». Завдяки цьому інструменту вдалося простежити хід виконання скриптів, виявити некоректні значення змінних і виправити несправності ще на етапі розробки. Особливу увагу приділяли складним ланцюжкам подій у бою, де навіть невелика розбіжність могла призвести до логічних колізій.

Крім функціональних перевірок, було проведено серію стрес-тестів із максимальною кількістю одночасних ефектів частинок та активним AI-керованим ворогом. Усі знайдені дефекти класифікувалися за ступенем критичності та терміновістю виправлення.

Дрібні UI-артефакти та несуттєві затримки в анімаціях було виправлено впродовж робочої сесії, тоді як більш складні помилки з логікою зіткнень та розрахунку шкоди потребували глибшого аналізу й кількох ітерацій усунення.

Такий підхід забезпечив поступове зростання якості проєкту і дозволив випустити фінальну версію з мінімальною кількістю критичних помилок.

Незважаючи на відсутність повної автоматизації, описана стратегія ручного та інтеграційного тестування дала змогу забезпечити високу стабільність основних механік. Водночас у подальших етапах планується доповнити процес автоматичними юніт-тестами ключових алгоритмів бойової системи та налаштувати CI/CD-конвеєр для швидкого виявлення регресій. Це дозволить зменшити навантаження на тестувальників і підвищити ефективність перевірок.

ВИСНОВКИ

У межах реалізації бойової підсистеми ігрового застосунку «Catalous» були чітко визначені та розділені ключові компоненти, які забезпечили відчутний і водночас збалансований бій. Були опрацьовані механіки точного нанесення ударів і контролю витривалості: розроблені алгоритми розрахунку витрат витривалості на кожну дію, враховуючи застосовані покращення й характер дії (атаки чи ухилення тощо). При цьому параметри відновлення витривалості коригуються залежно від зовнішніх факторів – наприклад, після успішного парирування чи тривалого утримання захисної позиції – щоб зберегти баланс між викликом та справедливістю сутичок.

Було впроваджено стратегічну глибину вибору між атакою, блоком, парируванням і ухиленням. Кожна з цих дій отримала унікальні характеристики впливу на бойовий процес: блок знижує шкоду, але поступово вичерпує витривалість; парирування відкриває вікно для потужного контратаки, проте вимагає суворого відчуття моменту атаки; ухилення дозволяє уникнути удару, але на короткий час залишає відкритим простір для контрзаходів ворога. Для реалізації цієї глибини було передбачено поділ логіки бойового циклу на окремі підсистеми, які взаємодіють через події: підсистема гравця, підсистема анімації, підсистема обробки витривалості та підсистема атаки та отримання ушкоджень.

Будо реалізовано прогрес персонажу через лінійне підвищення статистик із можливістю перерозподілу очок розвитку без ризику хибного інвестування. Для цього було розроблено модуль управління характеристиками, який підраховує поточний рівень, доступні для інвестицій очок та їхню вартість із урахуванням зростання складності рівнів. Зміна вартості очок прив’язана до кривих зростання, що адаптуються до рівня персонажа: вищий рівень робить подальший прогрес дорожчим, але відкриває доступ до більшої потужності персонажа.

Фінальним етапом інтеграції стало поєднання всіх компонентів у єдиному середовищі Unreal Engine 5 за допомогою системи BVS. Завдяки цьому створюється можливість швидкого прототипування та тонкого налаштування кожного параметра без необхідності перезбірки проєкту: розробник може в реальному часі регулювати криві відновлення витривалості, коефіцієнти витрат очок розвитку, налаштування компонентів та змінних анімацій у зручному інтерфейсі. Такий підхід гарантує технічну стабільність ігрового процесу, підтримку потрібного рівня стабільності гри та відчуття досягнення, коли успіх у бою залежить від власної майстерності гравця.

Мета розробки – створення ігрового програмного застосунку, яке поєднує динамічний бій у стилі «Souls-like» з системою розвитку персонажа – була досягнута.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Guzsvinecz T. The Soulsification of Video Games // Multimedia Tools and Applications. 2024. Vol. 84, No. 15. P. 14827–14853.

2. Unreal Engine [Електронний ресурс] // Unreal Engine 5.6 Documentation – 2025 – URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-6-documentation (дата звернення: 07.06.2025)

3. IGN [Електронний ресурс] // Dark Souls Review – 2011 – URL: https://www.ign.com/articles/2011/09/30/dark-souls-review (дата звернення: 07.06.2025)

4. IGN [Електронний ресурс] // Sekiro: Shadows Die Twice Review – 2019 – URL: https://www.ign.com/articles/2019/03/21/sekiro-shadows-die-twice-review (дата звернення: 07.06.2025)

5. Verified Market Reports [Електронний ресурс] // Soulslike Game Market Insights – 2025 – URL: https://www.verifiedmarketreports.com/product/soulslike-game-market/ (дата звернення: 08.06.2025)

6. Guzsvinecz T. The correlation between positive reviews, playtime, design and game mechanics in souls-like role-playing video games // Multimedia Tools and Applications. 2022. Vol. 82. P. 4641–4670.

7. Nguyen T. How to create a good Souls game: Case: Elden Ring : бакалаврська робота / LAB University of Applied Sciences. Vaasa, 2024.

8. Unreal Engine [Електронний ресурс] // Set Current Culture – 2025 – URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/BlueprintAPI/Utilities/Internationalization/SetCurrentCulture (дата звернення: 08.06.2025)

9. Unreal Engine [Електронний ресурс] // Apply Settings – 2025 – URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/BlueprintAPI/Settings/ApplySettings (дата звернення: 08.06.2025)

10. Unreal Engine [Електронний ресурс] // Random Bool With Weight – 2025 – URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/BlueprintAPI/Math/Random/RandomBoolwithWeight (дата звернення: 09.06.2025)

11. Unreal Engine [Електронний ресурс] // Normalize2D – 2025 – URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/BlueprintAPI/Math/Vector2D/Normalize2D (дата звернення: 12.06.2025)

12. Unreal Engine [Електронний ресурс] // Get Right Vector – 2025 – URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/BlueprintAPI/Math/Vector/GetRightVector (дата звернення: 12.06.2025)

13. Unreal Engine [Електронний ресурс] // Get Forward Vector – 2025 – URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/BlueprintAPI/Math/Vector/GetForwardVector (дата звернення: 12.06.2025)

14. Unreal Engine [Електронний ресурс] // Environment Query System – 2025 – URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/environment-query-system-in-unreal-engine (дата звернення: 13.06.2025)

15. Github – NureTkachovArtur/2025_B_PI_PZPI-21-1_Tkachov_A_O. Guthub. URL: https://github.com/NureTkachovArtur/2025_B_PI_PZPI-21-1_Tkachov_A_O (дата звернення: 14.06.2025)