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 у жанрі Roguelite RPG.____
_____Балансування, анімація, наратив»_____
(тема)

Виконав:

студент _4_ курсу, групи _____ПЗПІ-20-5_____

_____ Вожова М.В _____
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного____
_____забезпечення_____
(код і повна назва спеціальності)

Тип програми ____освітньо-професійна_____

Освітня програма ___Програмна інженерія_____

_____
(повна назва освітньої програми)

Керівник __старший викладач Новіков Ю.С.____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ _____Дудар З.В._____
(підпис) (прізвище, ініціали)

2024 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)

«___»_____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентці _____Вожовій Маргариті Володимирівні_____
(прізвище, ім'я, по батькові)

1. Тема роботи __«Ігровий програмний застосунок у жанрі Roguelite RPG._____ Балансування, анімація, наратив»_____
Затверджена наказом університету від _20_ __травня_____ 2024 р. №__471Ст____
2. Термін подання студентом роботи до екзаменаційної комісії 06 _червня_2024 р.
3. Вихідні дані до роботи _____Розробити ігровий програмний застосунок у жанрі Roguelite RPG, а саме баланс, анімації, та наратив за допомогою ігрового рушію Unreal Engine 5 та мови програмування C++
4. Перелік питань, що потрібно опрацювати у роботі _вступ, аналізи предметної__ галузі, формулювання вимог до програмної системи, архітектура та проєктування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, впровадження програмного забезпечення, висновки, перелік джерел посилання, додатки_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	08.04.2024	*виконано*
2	Створення специфікації ПЗ	11.04.2024	*виконано*
3	Проектування ПЗ	15.04.2024	*виконано*
4	Розробка ПЗ	15.05.2024	*виконано*
5	Тестування ПЗ	17.05.2024	*виконано*
6	Оформлення пояснювальної записки	20.05.2024	*виконано*
7	Підготовка презентації та доповіді	21.05.2024	*виконано*
8	Попередній захист	22.05.2024	*виконано*
9	Нормоконтроль, рецензування	25.05.2024	*виконано*
10	Здача роботи у електронний архів	01.06.202	*виконано*
11	Допуск до захисту у зав. кафедри	03.06.2024	*виконано*

Дата видачі завдання 8 квітня 2024 р.

Студент (ка)_____ Вожова М.В.
(підпис)

Керівник роботи _____ старший викладач Новіков Ю.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, стор. 66, рис. 40, джерел. 9, табл. 4.

АНІМАЦІЯ, СИСТЕМА АТРИБУТІВ, ROGUELIKE, RPG, ПРОГРАМНИЙ ЗАСТОСУНОК, C++, UNREAL ENGINE

Об’єктом розробки є система атрибутів, система візуальних ефектів та анімації, які є частиною комплексної роботи з розробки ігрового застосунку у жанрі Roguelite RPG.

Метою розробки є оптимізована система атрибутів і система візуальних ефектів та анімацій, що вписуються у концепт ігрового жанра.

Метод рішення – середовище розробки Visual Studio 22, мова програмування C++ та Blueprint, ігровий рушій Unreal Engine 5.3.1, середовище 3D моделювання та анімації Blender.

У результаті створено систему атрибутів та візуальні ефекти з анімаціями, що є частиною комплексної роботи з розробки програмного застосунку в жанрі Roguelite RPG.

ANIMATION, ATTRIBUTE SYSTEM, ROGUELIKE, RPG, APPLICATION SOFTWARE, C++, UNREAL ENGINE

The object of development is a system of attributes, a system of visual effects and animations, which are part of a comprehensive work on the development of a game application in the genre of Roguelite RPG.

The goal of the development is an optimized system of attributes and a system of visual effects and animations that fit into the concept of the game genre.

The solution method is the Visual Studio 22 development environment, the C++ and Blueprint programming languages, the Unreal Engine 5.3.1 game engine, and the Blender 3D modeling and animation environment.

The result is a system of attributes and visual effects with animations that are part of a comprehensive development effort for a software application in the Roguelite RPG genre.

Я, Вожова Маргарита Володимирівна, студентка гр. ПЗПІ-20-5, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Ігровий програмний застосунок у жанрі Roguelite RPG. Балансування, анімація, наратив», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлена з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

Вступ 8
1 Аналіз предметної галузі 9
1.1 Аналіз предметної галузі 9
1.2 Аналіз конкурентів 10
1.3 Постановка задачі 13
2Формування вимог до програмної системи 15
2.1 Функціональні вимоги до ігрового застосунку 15
2.2 Нефункціональні вимоги до ігрового застосунку 16
2.3 Вимоги до середовищ розробки 16
3 Архітектура та проєктування програмного забезпечення 17
3.1 UML проєктування програмного забезпечення 17
3.2 Вибір архітектури та рушія 19
3.3 Огляд ігрового циклу 20
3.4 Огляд алгоритму розрахунку урону 22
3.5 Створення прототипів UI/UX 25
4 Опис прийнятих програмних рішень 29
4.1 Балансування і система атрибутів 29
4.2 Система анімацій 35
5 Тестування програмного забезпечення 38
6 Впровадження програмного забезпечення 41
6.1 Наукове впровадження проекту 41
6.2 Практичне впровадження проекту 41
Висновки 43
Перелік джерел посилання 45
ДОДАТОК А Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ 46
ДОДАТОК Б Слайди презентації 47
ДОДАТОК В Гейм дизайн документ 56
ДОДАТОК Г Тези доповіді для науково-практичної інтернет-конференції 58

ДОДАТОК Ґ Тези доповіді для науково-практичної інтернет-виставки 62
ДОДАТОК Д Приклад програмного коду 64

ВСТУП

Завдяки постійному розвитку відеоігор кожного року з’являються нові. Жанр Roguelite, який поєднує елементи стратегії та екшену, у той самий час надає гравцям багато годин контенту, дозволяючи їм безліч разів перегравати гру і не набридати. Багато ігор досягають цього завдяки використанню елементів RPG і систем атрибутів і прокачок, які додають варіативності грі та дозволяють грати зацікавленим сторонам різного рівня досвіду.

Основне завдання цього дипломного проекту це створення системи атрибутів, а також візуальних ефектів та анімацій при чаклунстві заклять. Системи, котрі розробляються, повинні співпадати з новітніми стандартами ігрової індустрії для жанру Rogue-lite. Вони повинні бути програмно-незалежними, оскільки система, котра розробляється, буде частиною комплексної роботи з розробки ігрового програмного забезпечення в жанрі Roguelite. Розробка гри повинна продемонструвати ефективність використання системи атрибутів та заклять з візуальними ефектами та анімаціями, як частини системи жанру.

Сфера комп’ютерних ігор та розваг, у якій можуть зацікавитися як розробники, так і гравці, буде сферою використання результатів роботи. Ця комплексна робота дозволяє використовувати ці системи на різних платформах, а доступ до них може бути безкоштовним або платним для гравців, що сприяє розвитку проекту.

Відеогра у жанрі Roguelite RPG, створена завдяки ретельній розробці, матиме кілька локацій і дозволить гравцю використовувати різні комбінації системи атрибутів, що робить кожну нову спробу унікальною.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

В останні роки піджанр roguelite привернув увагу до жанру Roguelike[1], який характеризується процесом створення підземель, вічною смертю та фокусом на майстерності через повторювані проходження. Roguelite зберігає основні риси свого батьківського жанру, але додає нові елементи, щоб зробити його ще більш привабливим.

Roguelite відрізняється від свого попередника тим, що містить компоненти, які допомагають подолати розчарування, яке часто пов’язане з перманентною смертю. Навіть поетапні постійні оновлення дають відчуття прогресу, яке виходить за рамки окремих запусків. Ці оновлення можуть включати підвищення характеристик персонажа, додавання нових здібностей або розширення набору предметів, які зустрічаються під час майбутніх проходжень. Повторні спроби стимулюються постійною системою прогресування.

Крім того, Roguelite збільшує різноманітність стилів гри. Деякі Roguelite зберігають стратегічні компоненти своїх предків, але інші мають більш орієнтований на дію підхід, який включає бій у реальному часі та швидкий рух. Це поєднання різних стилів гри спрямоване на більш широку аудиторію гравців, які, можливо, не насолоджувалися повільним темпом традиційного Roguelike.

Наратив у Roguelite є ще однією деталлю, яку потрібно враховувати. Roguelite можуть мати багату історію, яка органічно розгортається в кількох проходженнях, на відміну від традиційних Roguelike, де сюжет часто відходить на другий план. Деталі навколишнього середовища, взаємодія персонажів і маленькі підказки, розкидані по процедурно створеному світі, можуть скласти велику історію. Цей фрагментарний підхід до оповіді заохочується кілька разів, і гравці отримують винагороди за більше розуміння історії гри з кожною спробою.

1.2 Аналіз конкурентів

«Enter the Gungeon»[3] - одна з ігор, які можна розглянути. Гра відтворює неймовірне підземелля, де ви досліджуєте лабіринт підземного світу, наповнений ворогами та кулями, маючи в руках арсенал зброї, що постійно розширюється. (див. рис. 2.1)

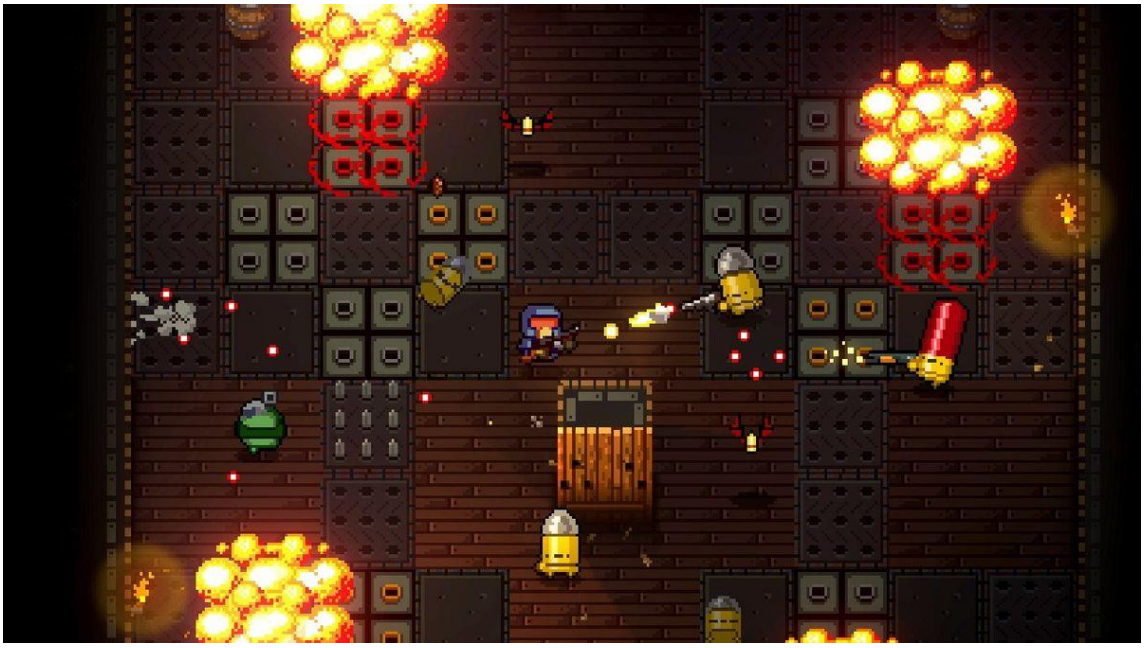


Рисунок 2.1 – Скріншот з гри «Enter the Gungeon» (2016 р.)

Переваги:

- «Enter the Gungeon» — це швидкий і безперервний екшн, який перенесе вас у центр подій із гострими перестрілками. Щоб вижити, вам знадобляться гострі навички ухилення та швидкі рефлекси..

- Купа зброї, оскільки гра пропонує широкий вибір різноманітних і чудових видів зброї, від традиційних пістолетів до дивних істот, які стріляють кулями, що відскакують, або вибуховими вівцями.

- Завдяки тому, що гра може містити процедурно створені рівні та елементи в кожному проходженні, досвід залишається новим і складним.

-Кооперативна гра з іншими гравцями.

Недоліки:

- Важко почати гру, оскільки вона відома як складна, через ворогів і систему постійної смерті, яка вимагає від гравця починати з нуля одразу після поразки.

- Повторюваний ігровий цикл, адже для деяких гравців основний ігровий цикл, пов’язаний з битвою в кімнатах і збиранням здобичі, може повторюватися.

«Hades»[4] це є рольова екшн-гра поєднує в собі захоплюючі битви та насичену історію, орієнтовану на персонажів. Гравці грають Загрея, сина грецького бога Аіда, який намагається вибратися з підземного світу на гору Олімп. (див. малюнок 2.2)

Рисунок 2.2 – Скріншот з гри «Hades» (2020 р.)

Переваги:

- Переконливий сюжет, який гра демонструє через те, як персонажі взаємодіють один з одним і з навколишнім середовищем. Структура roguelike дозволяє історіям розгортатися в кількох проходженнях, винагороджуючи працю.

- У грі використовується яскравий художній стиль, натхненний грецьким дизайном та архітектурою, і неймовірний саундтрек, який динамічно змінюється під час проходження гри.

- Висока можливість грати більше завдяки поєднанню генеративно-створених рівнів, постійних оновлень і обширної розповіді заохочує гравців повертатися за новими.

Мінуси:

- Цикл основної гри повторюється, хоч історія і розгортається протягом кількох ігор, але гравці вважають, що основний цикл гри включає повзання в підземеллях і бій, тому й повторюється

- «Hades» пропонує складний виклик, який може збентежити деяких, особливо тих, які не знайомі з жанром Roguelike.

«DoomRL»[5] - це безкоштовна Roguelike гра, яка поєднує в собі елементи класичного шутера Doom з Roguelite жанром. Гравці виживають у випадково згенерованому адському середовищі, збираючи зброю, бойові предмети та здібності, щоб боротися зі злом. Гра акцентується на стратегічному плануванні, вмінні уникати пасток та ворожих атак, і надає гравцям інтенсивний геймплей у стилі Roguelite. (див. малюнок 2.3)

Рисунок 2.3 - Скріншот з гри «DoomRL»(2013 р.)

Переваги:

- Динамічний геймплей дозволяє швидко насолоджуватись грою
- Кожна гра унікальна, завдяки різноманітності персонажів, зброї та рівнів.

Мінуси:

- Піксельна графка для деяких користувачів може виглядати неактуальною .
- Складність гри може надавати сумніви навіть досвідченим користувачам, не кажучи вже про новачків.

Підсумовуючи вище проаналізоване, можна зробити висновок, що деякі конкуренти можуть зосереджуватись на глибині і стратегії, забезпечуючи гравцям складні механіки або системи управління персонажами. Інші можуть ставити акцент на швидкості та динаміці, пропонуючи експресивний та захоплюючий геймплей.

1.3 Постановка задачі

У ході створення ігрового застосунку у жанрі Roguelite RPG[2], а саме його геймплейної та візуальної частин, у які входять системи атрибутів та анімація заклять потрібно врахувати аспекти, указані нижче:

Система атрибутів:

- атрибути можуть бути відображеними перед гравцем в залежності від їхніх характеристик;
- сила закляття повинна залежати від кількості атрибутів;

Балансування атрибутів:

- рівень вірогідності отримання гравцем атрибутів повинен залежати від їх кількості у грі.

Візуальні ефекти:

- ефекти не повинні сповільнювати швидкість генерації картинки;
- повинні бути простими та легко розпізнавальними.

Врахування цих аспектів є критичним для успішного створення геймплею ігрового застосунку у жанрі Roguelite RPG. Ці інтегровані системи атрибутів та візуальних ефектів не лише додають глибину і складність грі, а й впливають на загальний враження від ігрового досвіду.

Правильно збалансована система атрибутів дозволяє гравцеві відчути справедливість у розвитку свого персонажа. Вона стимулює гравців до дослідження та досягнення нових цілей, надаючи їм відчуття досягнення та прогресу. Відповідно, геймплей стає більш захоплюючим та цікавим, що збільшує ймовірність залучення гравців на тривалий час.

Покращені візуальні ефекти додають естетичне задоволення гравцям, роблячи гру більш привабливою та захоплюючою. Простота та легка розпізнаваність ефектів дозволяють гравцям швидко оцінити ситуацію в грі та приймати швидкі та зважені рішення.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Функціональні вимоги до ігрового застосунку

З розгляду до постановки задачі до цього, з врахуванням переваг та недоліків розглянутих аналогів раніше, для компоненту ігрового застосунку, шо розроблюватиметься, ставляться наступні функціональні вимоги:

Система атрибутів:

- більше атрибутів – сильніші закляття;
- чим більше одного атрибуту – тим рідше віз зустрічається;
- першорядні атрибути впливають на гру більше ніж другорядні;
- всі механіки бою відображаються в атрибутах;
- не всі атрибути можуть побачити гравець;
- атрибути впливають на швидкість перезарядки, витратність та інші аспекти закляття, що закладені в нього.

Анімації:

- анімації не повинні заважати геймплею;
- анімації можна скасувати;
- анімації не виглядають надто різкими/смиканими.

Візуальні ефекти:

- не повинні впливати на геймплей;
- повинні бути легко розпізнаваними та унікальними;
- схожі ефекти повинні робити схожі речі.

Отже, для досягнення цих вимог необхідно розробити систему атрибутів, що відповідає вищеописаним принципам, створити анімації, які дотримуються наведених вимог, і впровадити візуальні ефекти, які ніяк не впливають на геймплей, але додають до нього глибину та естетичний сприйняття.

2.2 Нефункціональні вимоги до ігрового застосунку

До нефункціональних вимог частини ігрового застосунку можна віднести:

- візуальні ефекти повинні співпадати за кольором та тривалістю дії;
- об’єкти та закляття, належні до одного атрибуту, повинні мати колір цього атрибуту;
- анімації повинні бути плавними;
- таймінги анімацій повинні співпадти по часу чаклування заклять.

Дотримання цих нефункціональних вимог допоможе забезпечити гармонійний та задоволений геймплей, що сприятиме позитивному враженню від ігрового досвіду.

2.3 Вимоги до середовищ розробки

Для створення гри буде використаний ігровий рушій Unreal Engine 5.3.1, як найбільш просунутий движок з широким інструментарієм найновіших функцій, які полегшують задачу створення ігрової анімації, у той самий час надаючи змогу створювати якісну графіку.

Unreal Engine 5 використовує мову програмування C++ як стабільну та оптимізовану для гри, у цей час підтримуючи інтеграцію з Visual Studio 22 для легкого розширення наявного функціоналу, так і для зміни частин вихідного коду ігрового рушія.

При створенні локацій, 3D об'єкти будуть використані із ассетів в Unreal Engine Marketplace та окремі елементи будуть розроблятися у програмі Blender.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування програмного забезпечення

Процесі розробки ігрового застосунку почався з використання UML-проектування. На першому етапі була створена діаграма Use Case, яка показує функціонал гри та усі потенційні взаємодії елементів. На рисунку 3.1 показано загальну діаграму використання для гри у жанрі Roguelite RPG.

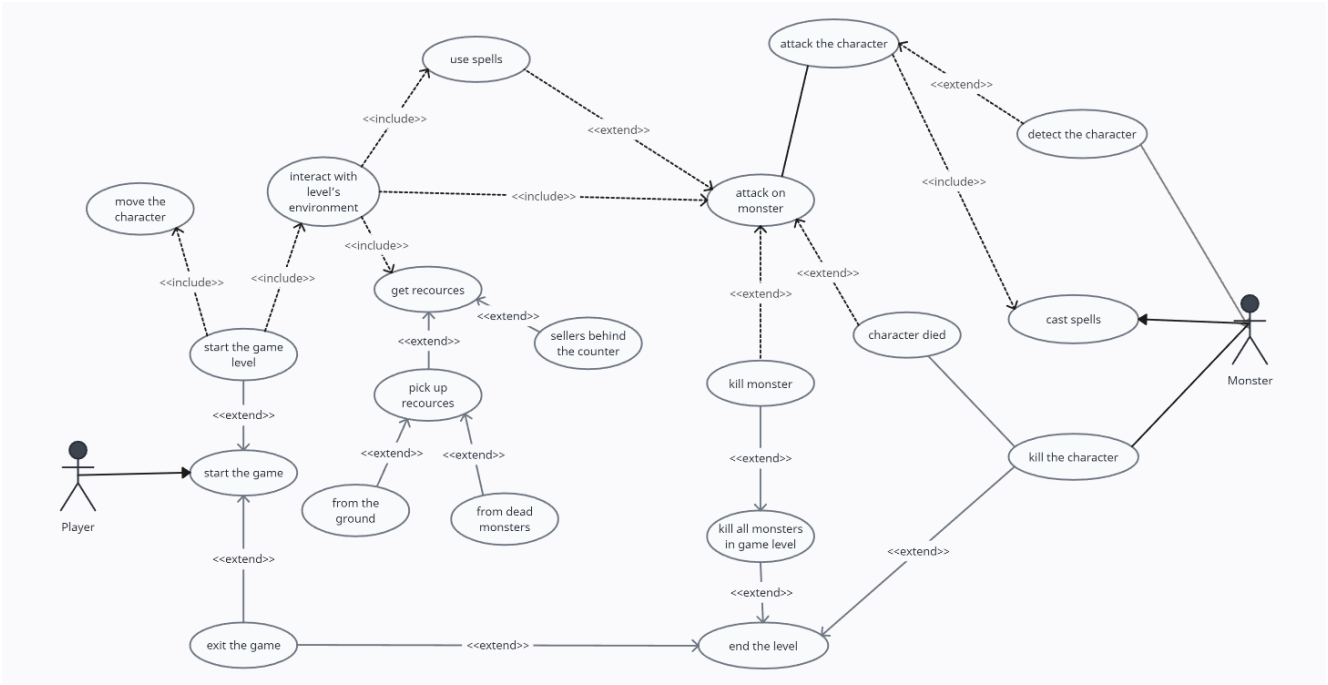


Рисунок 3.1 – Use Case діаграма основних функцій ігрового застосунку

Переглянемо більш детально деякі елементи діаграми, пов’язані з системою заклять та те, як вони впливають на атрибути.

Є два типи акторів : Монстр та Гравець.

Гравець – користувач ігрового застосунку, головний герой у самій грі

Монстр – це один з типів ворогів у грі, та не являється ігровим персонажем

- атакувати персонажа (attack the character) – монстр атакує гравця використовуючи закляття (cast spells), котрі супроводжуються візуальними ефектами та анімаціями, які впливають на його атрибути, або на атрибути самого монстра. Таким чином можна, наприклад, наносити шкоду ворогу, або ненароком гравцю

- вбити персонажа (kill the character) –монстр вбиває гравця, через що шкала здоров'я гравця анулюється, та атрибути здоров'я дорівнюють 0. Смерть гравця супроводжується візуальним ефектом миготіючого почервоніння

Розглянемо Гравця (Player) з точки зору атрибутів та анімації:

- почати гру (start the game) – гравець розпочинає гру, починає взаємодію з ігровим середовищем і виконуєв ньому різні дії.

- почати рівень гри (start the game level) – гравець починає битися з монстрами і отримує необхідні ресурси задля свого ігрового прогресу(досвід, атрибути, закляття)

- взаємодіяти з навколишнім середовищем (interact with level's environment) – гравець може знаходити предмети, бонуси, зброю, що надають змогу зростати навикам гравця та отримувати закляття.

- отримувати ресурси(get resources) – гравець може отримувати різними шляхами ресурси (здоров'я, досвід, зброя), такими як: шляхом підбору ресурсів з мертвих монстрів чи просто знахідок, або купляти їх у продавців за прилавком.

Таким чином, вище було розглянуто варіанти взаємодії для Монстру та для Гравця з боку використання атрибутів та візуальних ефектів.

Гравець і Монстр можуть взаємодіяти використовуючи закляття, щоб змінити свої атрибути. Основною метою обох є зниження здоров'я, щоб перемогти опонента.

3.2 Вибір архітектури та рушія

Сьогодні існує велика кількість ігрових рушіїв, які можна використовувати для створення високоякісних ігор. Але наразі три рушія Unreal Engine, Unity та Godot є найпопулярнішими рішеннями. У кожного є переваги та недоліки. Один проєкт буде ефективніше використовувати один рушій, а іншому буде набагато легше виконувати завдання з іншим.

Unreal Engine, на відміну від двох інших варіантів, полегшує створення оточення та надає можливість змінювати архітектуру рушія на низькому рівні, а також надає значно більше функціональності, пов’язаної з графікою та анімаціями. Програмування C++ дозволяє досягти більшої оптимізації. Це також стосується шейдерів, оскільки Unreal Engine надає можливість редагувати їх за допомогою мови HLSL, яка використовується для низькорівневого програмування шейдерів. У даний час як Godot, так і Unity програмують ігри виключно за допомогою C# або будь-якої іншої скриптової мови, що заважає оптимізації ігор.

Унікальна технологія візуалізації Unreal Engine [5] є однією з найважливіших переваг програми. Неймовірно деталізовані та захоплюючі середовища можна створювати за допомогою функцій, таких як Lumen [6], динамічні системи освітлення, і Nanite [7],віртуалізація геометрії. Даний момент дуже важливий для жанру Roguelite RPG, де гравці потрапляють у фантастичні світи з різноманітними локаціями. Завдяки здатності Unreal Engine 5 відтворювати ці світи з такою точністю, досвід гравця може значно покращитися, надаючи відчуття дива.

Unreal Engine 5 професійно створює динамічний світ за допомогою візуальних елементів. Ніагарська система візуальних ефектів дає змогу створювати захоплюючі анімації, гарні високоякісні реалістичні шейдери для відображення різних об’єктів навколишнього середовища та дивовижні здібності персонажів

Розглянемо різні види архітектури і визначимо найбільш підходящий для проєкту, який зараз розробляємо. Архітектура ігрових застосунків будується на трьох основних принципах:

- Компонентна архітектура, допомагає розбивати логічні частини на компоненти. Таким шляхом можно розбивати атрибути, анімації ефектів на компоненти, які можна комбінувати з іншими компонентами та проводити тестування . Усе це полегшує процес розробки ігрового застосунку

Завдяки обраній архітектурі зручно створювати новий функціонал, не заважаючи іншим членам команди розробляти застосунок.

3.3 Огляд ігрового циклу

Ігровий цикл – один з основних етапів розробки ігрового застосунку (англ. "game loop"). Він описує основні механіки гри та шлях отримання досвіду гравцем під час гри. Принцип циклу полягає в тому, що кожна дія призводить до наступної, що створює послідовність. У цьому результаті послідовність повертається до первісної дії – це основна суть формування геймплею. Деякі дії у циклі також можуть додавати або забирати ресурси гравця (валюта, енергія, час, ресурси).

Тепер подивимось на ігровий цикл розроблюваного ігрового застосунку у жанрі Roguelike (див. рис. 3.2). Так як цикл замкнут, на рисунку ми позначимо початок ігрової сесії з події "початок нового рівня" (start new level). Далі можна побачити підцикл, позначений як проходження рівня (level_walkthrough). Це є основною частиною ігрового циклу в жанрі Roguelike, де гравець взаємодіє з рівнями, досліджує їх, збирає ресурси та бореться з ворогами.

Після завершення проходження рівня (level_walkthrough) гравець має можливість вибрати наступний крок: або перейти до наступного рівня, або завершити гру. Якщо гравець обирає перехід до наступного рівня, цикл повторюється з події "початок нового рівня" (start new level), і гравець відправляється в новий пригодницький шлях.

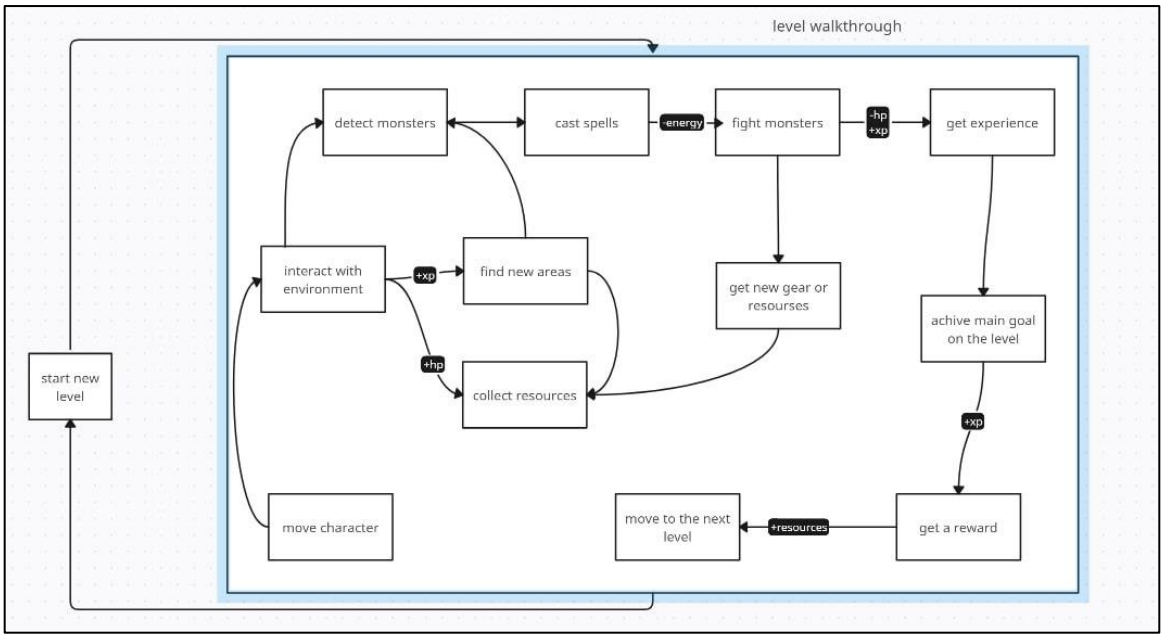


Рисунок 3.2 – Ігровий цикл розроблюваного застосунку

Основний функціонал гри наведено у середині циклу, основні дії, які відбуваютьсяв циклі:

- рух персонажа;
- взаємодія з навколишнім середовищем;
- відкриття нових локацій;
- виявлення супротивників;
- використання атрибутів;
- отримання нових предметів;

Підбираючи деякі предмети в ігровому середовищі, гравець може активувати ефекти, збільшити характеристики певних атрибутів, наприклад максимальне здоров'я, ману, спритність, силу, або ж отримати тимчасовий ефект, який робить те саме. Така система додає глибину геймплею і створює стратегічні можливості для гравця. Ось детальний огляд атрибутів у грі:

Здоров'я : Визначає загальну витривалість гравця. Коли здоров'я падає до нуля, гравець гине. Підвищення цього атрибуту збільшує шанси на виживання в поєдинках.

Мана : Використовується для активації спеціальних здібностей, заклинань або умінь. Це необхідно для виконання магічних дій. Чим більше мани, тим більше можливостей для використання магії.

Спритність : Впливає на швидкість руху гравця, його шанси ухилитися від атак та реакцію на загрози. Висока спритність дозволяє гравцеві швидше реагувати на небезпеку та ефективніше уникати ударів.

Сила : Визначає фізичну потужність гравця. Вона впливає на урон, який гравець може завдати в бою, а також на його здатність носити важке обладнання чи піднімати великі предмети.

Інтелект : Відображає кмітливість та здібності гравця до розумової діяльності. Вищий рівень інтелекту може збільшити ефективність магічних атак, збільшити обсяг мани та покращити можливості використання складних стратегій.

Вдача : Цей атрибут впливає на шанси гравця знаходити рідкісні або цінні предмети, уникати нещасних подій або успішно виконувати дії з випадковим результатом.

Гравець може підвищувати ці атрибути, збираючи предмети, які надають бонуси до певних характеристик. Також деякі дії в грі або досягнення можуть призводити до тимчасових змін у цих атрибутах. Наприклад, використання магічного еліксиру може тимчасово збільшити максимальне здоров'я або ману гравця, а знаходження бонусного предмета може надати тимчасовий буст до сили чи спритності.

3.4 Огляд алгоритму розрахунку урону

Цей алгоритм(рис.3.3.) розроблено для обчислення пошкодження, завданого гравцям у грі. Він дозволяє враховувати різні фактори, такі як шанс ухилення від удару та збільшення пошкодження від заклять. За основу алгоритм бере такі компоненти: отримати атрибути цілі, отримати атрибути джерела, прорахувати

урон, отримати тип урона, отримати опір в залежності від типа урону, перемножити, додати урон.

Роздивимось алгоритм детальніше:

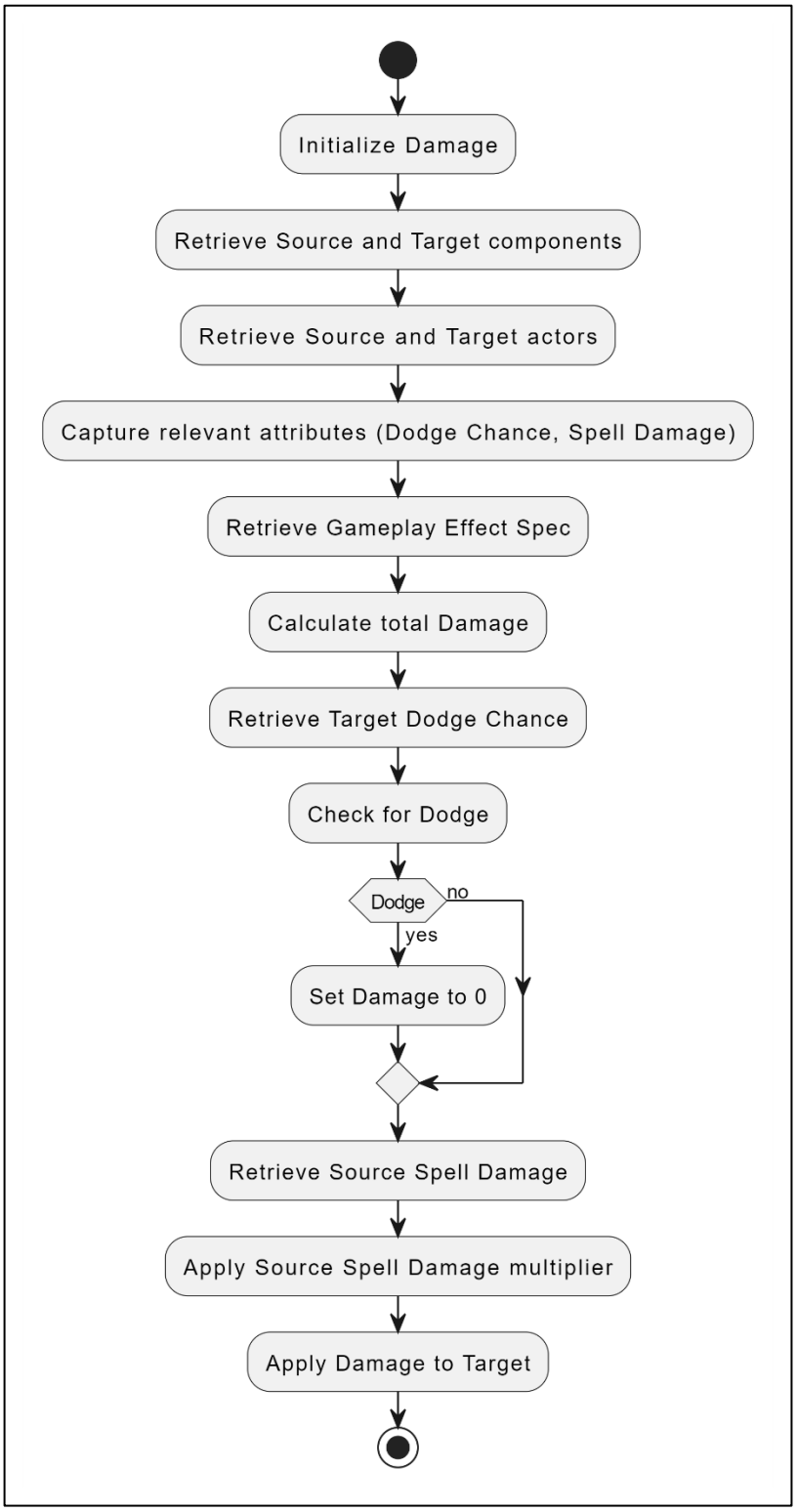


Рисунок 3.3 - Алгоритм розрахунку урону

Основні функції:

- оцінка пошкодження : алгоритм розраховує загальну суму пошкодження, враховуючи різні типи пошкодження, які можуть бути застосовані гравцеві.
- врахування шансу ухилитися: алгоритм визначає шанс цільового об'єкта ухилитися від удару, використовуючи атрибут "Шанс Ухилитися" зі збірника атрибутів цільового об'єкта. Він обчислює, чи вдалося цільовому об'єкту ухилитися від пошкодження, використовуючи випадкову величину.
- застосування модифікаторів пошкодження: алгоритм враховує будь-які додаткові модифікації пошкодження, наприклад, збільшення або зменшення від магічних ефектів.

Ключові компоненти:

- обчислення пошкодження: алгоритм складає загальну суму пошкодження, яка включає в себе різні типи пошкодження, представлені різними тегами.
- шанс ухилитися: використовуючи атрибут "шанс ухилитися" цільового об'єкта, алгоритм визначає можливість ухилення від пошкодження.
- модифікація пошкодження: застосовуються будь-які додаткові модифікації до пошкодження, такі як збільшення чи зменшення від магічних ефектів.

Даний алгоритм гарантує реалістичне та стратегічне обчислення отриманого урону для гравців у грі. Він дозволяє враховувати різноманітні фактори, що впливають на пошкодження, забезпечуючи захопливий ігровий досвід завдяки ключовим компонентам алгоритму, котрі включають обчислення пошкодження, шанс ухилитися та модифікацію пошкодження. Такий підхід до розрахунку урону додає глибини та стратегічності у геймплей, забезпечуючи задоволення від ігрового процесу.

3.5 Створення прототипів UI/UX

Для покращення інтерфейсу існуючих аналогів було введено низку змін, щоб підвищити досвід гри та привернути більшу увагу до світу гри.

Гравцю потрібна повна інформація про атрибути гри та їх спричинений ефект на закляття, тому необхідно передати майже всю інформацію про активні атрибути, які доступні гравцю. (рис.3.4)

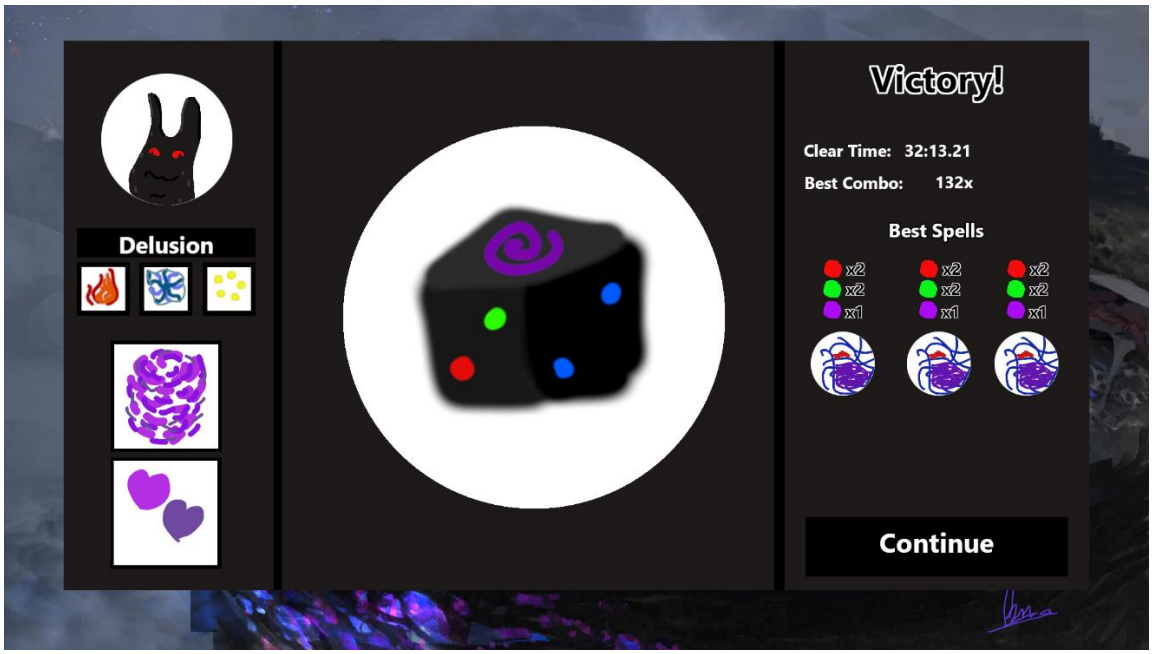


Рисунок 3.4 – Прототип UI під час перемоги

Рисунок 3.5 - Шкала основних життєвих атрибутів головного героя

У життєвих атрибутів три основні елементи: шкала здоров’я (червона стрічка), шкала мани (синя) та шкала героїчної здібності (бірюзова сфера) є кількома основними атрибутами, які представлені на рисунку 4.5. Кожен з атрибутів містить інформацію про гравця, котрий може продовжувати гру, відмовлятися від своїх героїчних здібностей, обережно зіграти, щоб зберегти здоров’я, чи використовувати додаткові закляття, оскільки мана нещодавно поповнилася.

З правого боку рисунку 3.5 можна побачити ключові атрибути, котрі сприяють на закляття.

Таким чином, ми маємо прототип, який містить усю інформацію, яку гравець повинен знати, щоб зробити важливі рішення щодо перемоги над ворогами, просування в сюжеті та загального прогресу в грі.

3.6 Створення наративу та сторітелінгу

У процесі архітектури та проєктування даного ігрового програмного застсоунку важливим аспектом є створення захоплюючого наративу та сторітелінгу. Наратив є ключовим елементом, що забезпечує емоційний зв'язок гравця з ігровим світом та його персонажами, а сторітелінг визначає, як саме ця історія буде розкрита протягом гри. У даній грі використовується динамічний та інтерактивний тип сторітелінгу, що дозволяє гравцям безпосередньо впливати на розвиток подій та еволюцію персонажа.

В основі гри лежить історія Денні, мисливця за надприродним, який у пошуках демонічної істоти потрапляє в «Щасливе Казино». У цьому місці, де він очікував знайти свою здобич, Денні зустрічає демона «Шифтера часу», який раптово змінює все довкола. Казино перетворюється на чарівний сюрреалістичний світ, населений дивовижними істотами, а сам Денні стає живою кубикоподібною істотою з унікальними здібностями.

Динамічний сторітелінг дозволяє інтегрувати елементи сюжету безпосередньо в ігровий процес. Кожен рівень гри є частиною сюрреалістичного світу, який постійно змінюється під впливом магічних сил, що додає елемент непередбачуваності і випробувань. Ці зміни не лише естетичні, але й функціональні – вони впливають на ігрову механіку та виклики, з якими стикається гравець. Наприклад, різні секції казино можуть змінюватися від рівня до рівня, створюючи нові пастки та можливості для дослідження.

Гравці керують Денні, використовуючи його нові здібності для боротьби з ворогами та вирішення загадок, що виникають на шляху. Кожна перемога та кожна поразка відкривають нові аспекти сюжету, дозволяючи гравцеві глибше зануритися в світ гри. Інтерактивний сторітелінг забезпечує, що гравці не просто спостерігають за розвитком подій, але активно беруть у них участь, приймаючи рішення, які впливають на хід гри.

Зокрема, здібності Денні, як кубикоподібної істоти, є центральним елементом ігрової механіки і одночасно ключовим сюжетним компонентом. Кожна нова здатність, яку він здобуває, не лише дозволяє долати нові перешкоди, але й розкриває більше інформації про його трансформацію та природу магічного казино. Цей підхід до сторітелінгу гарантує, що гра завжди залишається свіжою та цікавою, а гравці постійно отримують нові сюрпризи та виклики.

Наратив також розвивається через інтерактивні діалоги з іншими персонажами та зустрічі з дивовижними істотами, які населяють казино. Кожен з цих персонажів має свою власну історію та мотиви, що додає глибини основному сюжету. Гравці можуть дізнаватися більше про світ гри та його жителів, досліджуючи різні куточки казино, виконуючи побічні завдання та розгадуючи таємниці.

Додатково, в грі використовуються елементи випадково генерованих подій та локацій, що характерно для роглайк жанру. Це означає, що кожне нове проходження гри буде відрізнятися від попереднього, створюючи унікальний досвід для кожного гравця. Наприклад, кімнати казино можуть змінювати своє розташування, створюючи нові виклики та можливості для гравців. Цей аспект

додає грі повторюваності та довговічності, заохочуючи гравців повертатися до гри знову і знову.

Сторітелінг також підсилюється за допомогою візуальних та звукових ефектів, що створюють унікальну атмосферу магічного казино. Дизайн локацій, персонажів та звуковий супровід ретельно розроблені, щоб передати чарівність та загадковість світу гри. Кожна деталь, від звуків кроків на килимі казино до мерехтливих вогників на слот-машинах, сприяє створенню захоплюючого ігрового досвіду.

Таким чином, наратив та сторітелінг у цій грі не лише створюють багатошаровий сюжет, але й впливають на ігровий процес, додаючи йому глибини та інтриги. Динамічний та інтерактивний підхід дозволяє гравцям зануритися у світ гри, відчуваючи себе невід'ємною частиною цього чарівного, сюрреалістичного всесвіту. Кожна дія гравця, кожне прийняте рішення формує унікальний ігровий досвід, роблячи кожне проходження гри неповторним. Це забезпечує тривале залучення гравців та незабутній досвід, який залишає слід у їхній пам'яті.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Балансування і система атрибутів

Балансування включає в себе багато елементів системи, але дві основні категорії з яких можна отримати усі інші включають в себе закляття та атрибути. Атрибути поділяються на первинні, вторинні та життєво важливі. Другі дві категорії випливають з першої, тобто їхні значення залежать від первинних атрибутів, так, наприклад, здоров’я залежить від сили, а швидкість від гнучкості. Всього існує 16 атрибутів, які диктують правила гри і від яких напряму залежить геймплей. Залежно від атрибутів змінюються сили заклять, таким чином, наприклад інтелект впливає на посилення шкоди заклять.

Гравцю доступна лише інформація про теперішню кількість здоров’я та витривалості. Окрім цього, гравець може побачити скільки здоров’я залишилося у ворогів, якщо вони отримають шкоду. Важливим є і той факт, що деякі атрибути мають свої максимальні значення в залежності від рівня гри, наприклад здоров’я, або витривалість, відображаються у таблиці 4.1.

Таблиця 4.1 – Залежність максимальної значимості атрибутів від рівня

Назва атрибуту	Рівень 1	Рівень 2	Рівень 3	Рівень 4	Рівень 5	Рівень 6
Здоров’я	150	155	160	170	180	200
Витривалість	3	3	3	3	4	4
Сила атаки	10	15	20	25	30	35
Швидкість пересування	400	425	450	475	500	515

Деякі атрибути не мають максимальних значень, натомість вони обмежені у тому, як швидко зростає залежний від них атрибут, таким чином використовується функція натурального логарифму для зростання магічної шкоди залежно від інтелекту, таким чином за завеликої кількості інтелекту магічна шкода перестане зростати так само швидко (рис. 4.1). Це саме стосується й деяких інших атрибутів.

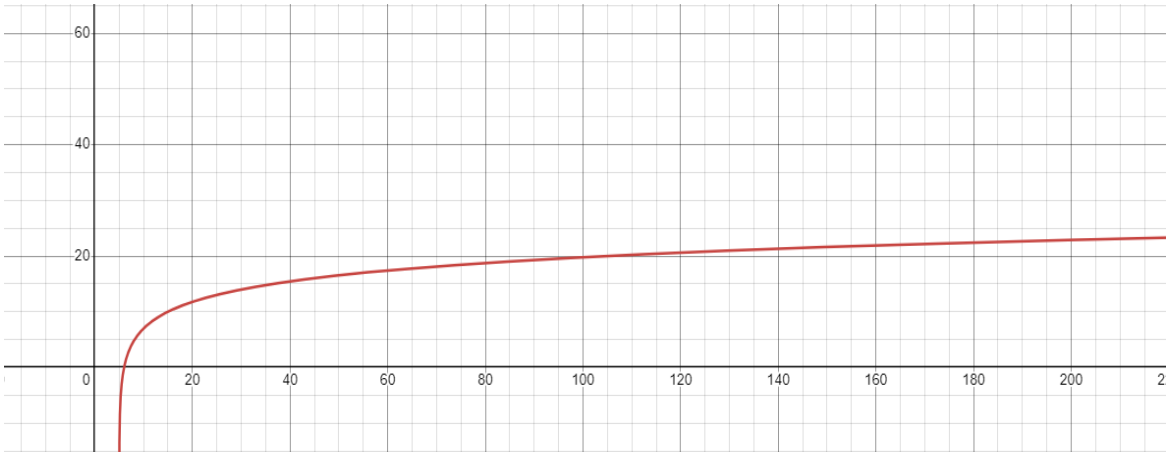


Рисунок 4.1 – Залежність магічної шкоди (вісь Y) від інтелекту (вісь X)

Окрім цього застосовуються вбудовані функції UE5 для створення потрібних залежностей, за допомогою кубічних рівнянь (рис 4.2 та 4.3).

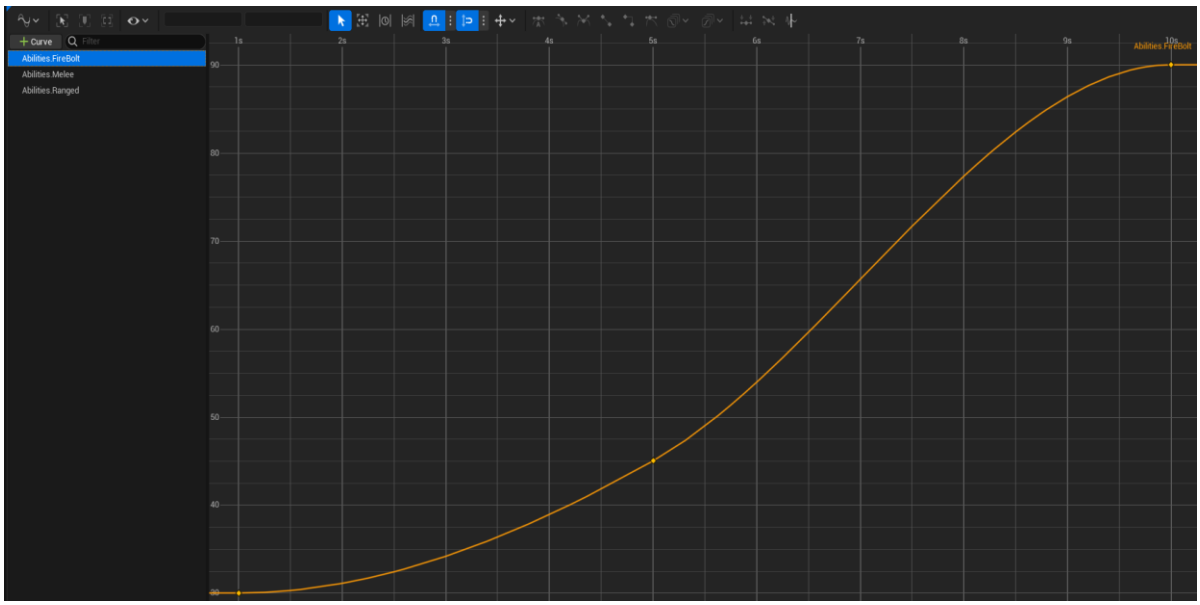


Рисунок 4.2 – Граф шкоди здібності Firebolt залежно від атрибуту тьми.

Використання логарифмічних і кубічних функцій забезпечує плавне та контрольоване зростання атрибутів, дозволяючи балансувальникам тонко налаштовувати ігрові механіки. Це також сприяє створенню складних та стратегічно насичених систем розвитку, де гравці повинні уважно підходити до

вибору напрямків розвитку своїх персонажів. Наприклад, вирішуючи, чи інвестувати більше в інтелект для підвищення магічної шкоди, гравець мусить зважати на те, що після певної точки ефективність цього показника буде знижуватися, і можливо, варто звернути увагу на інші атрибути.

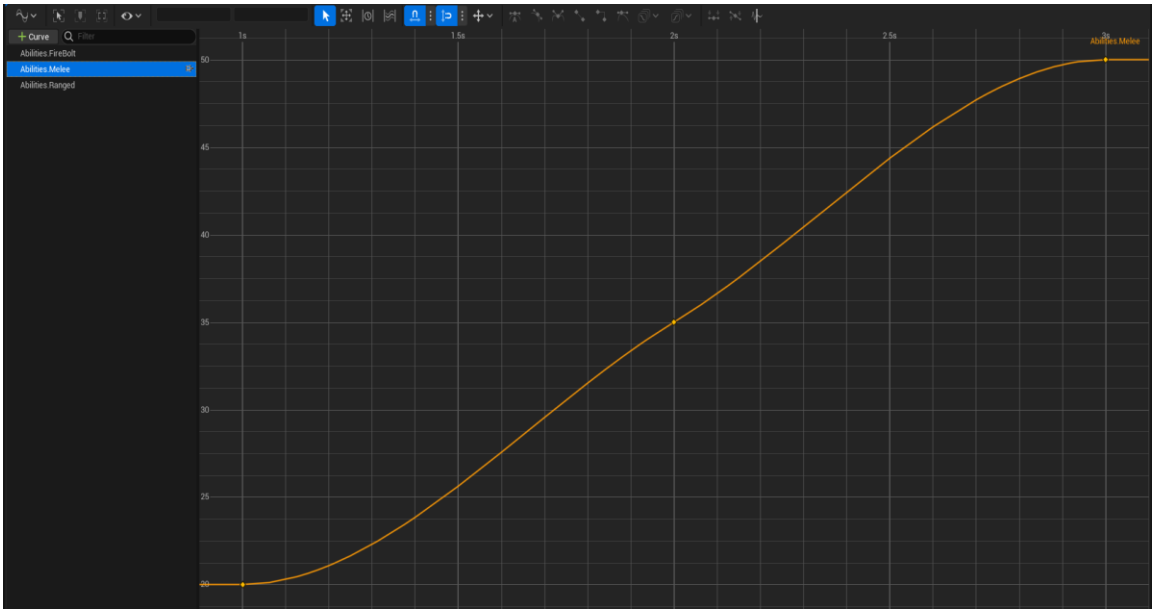


Рисунок 4.3 – Граф шкоди здібності Melee залежно від атрибуту сили

Усі атрибути та здібності напряму залежать від установлених у процесі балансування таблиць, які під час гри застосовуються напряму завдяки вбудованій системі UE5. Таким чином для гравця можна встановити стартові здібності та набір атрибутів, що він отримує, не змінюючи існуючий код. Це забезпечує гнучкість і легкість налаштування персонажів, дозволяючи розробникам швидко вносити зміни та тестувати різні сценарії. Наприклад, здоров’я персонажа може автоматично коригуватися залежно від рівня складності або конкретної місії, що додає додатковий рівень занурення в гру. Витривалість, у свою чергу, впливає на здатність персонажа виконувати різні дії, такі як біг або атака, що підвищує стратегічну складову гри. Сила атаки визначає, наскільки ефективно персонаж може завдавати шкоди противникам, враховуючи різні типи зброї та вміння. Швидкість пересування впливає на мобільність і можливість уникати небезпек.

Завдяки такому підходу, гравці отримують багатогранний і динамічний ігровий процес, де кожне рішення має значення і впливає на результат. (рис. 4.4)

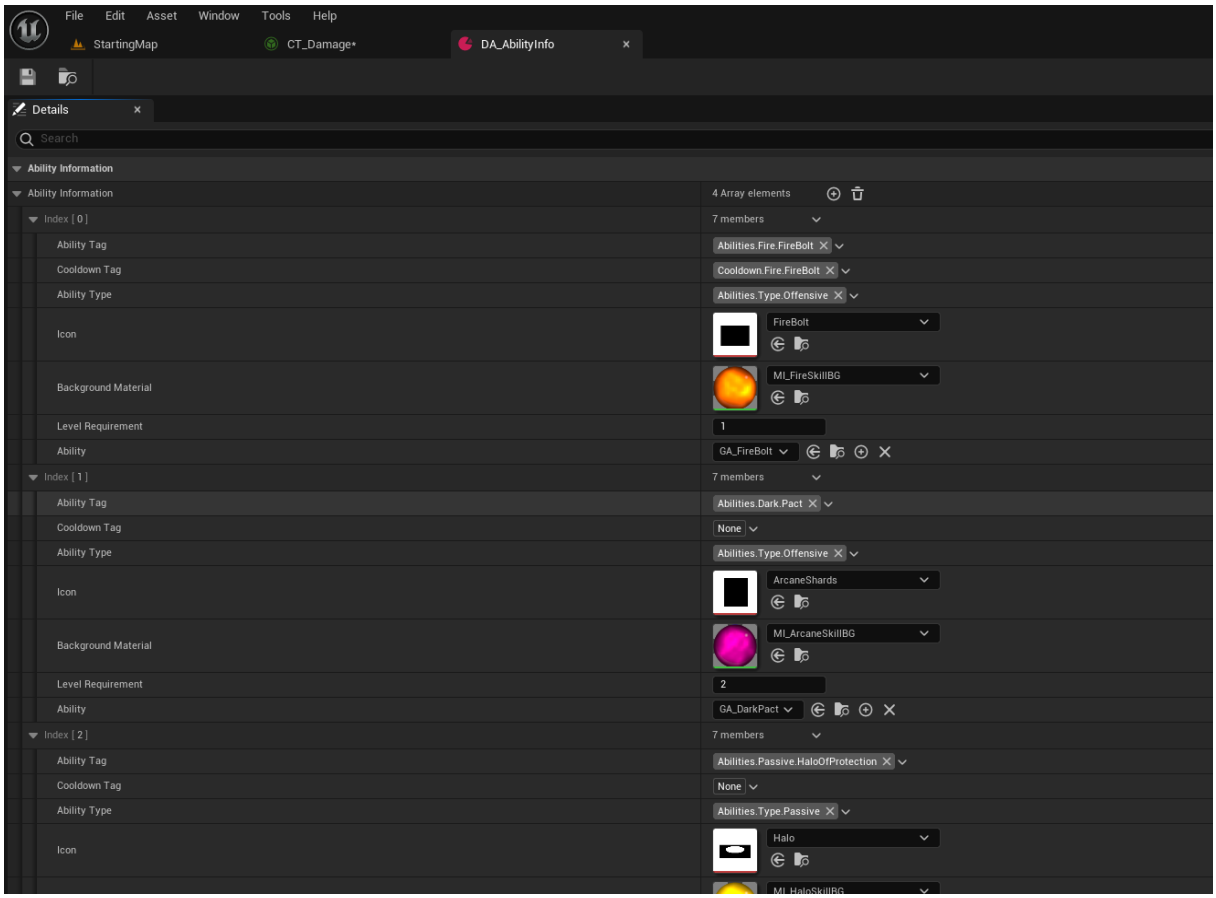


Рисунок 4.4 – Стартові атрибути та здібності гравця

Усі атрибути, що можна знайти у грі, та їх залежності від інших також прописані у вигляді таблиць та формул, що широко використовується для легшої маніпуляції у першу чергу балансувальниками, а також спрощує процес змін у ході самої гри. Цей підхід дозволяє розробникам і геймдизайнерам легко відслідковувати і коригувати різні параметри персонажів і механіки гри без необхідності глибокого втручання у код. Завдяки цьому можна швидко реагувати на зворотний зв'язок від гравців і здійснювати балансування, забезпечуючи оптимальний ігровий досвід.(рис.4.5)

Таблиці і формули дозволяють автоматизувати багато процесів, таких як підвищення рівня персонажа, розподіл очок характеристик, розрахунок ефектів від

використання різних предметів або здібностей. Це не тільки підвищує ефективність розробки, але й забезпечує прозорість і передбачуваність ігрових механік. Крім того, використання таких таблиць і формул сприяє легшій інтеграції нових елементів у гру, адже нові атрибути чи здібності можна просто додати до існуючих таблиць, не порушуючи загальної логіки гри.

Наприклад, якщо в гру додається новий вид зброї або обладунків, балансувальники можуть швидко внести відповідні зміни у таблиці, що визначають їх вплив на атрибути персонажа, такі як сила атаки чи рівень захисту. Також це дозволяє динамічно змінювати характеристики у відповідь на різні ігрові події або досягнення гравців, додаючи ще більше глибини і варіативності у геймплей. Завдяки цьому підходу, гра може постійно розвиватися і вдосконалюватися, залишаючись цікавою і захопливою для гравців протягом тривалого часу.

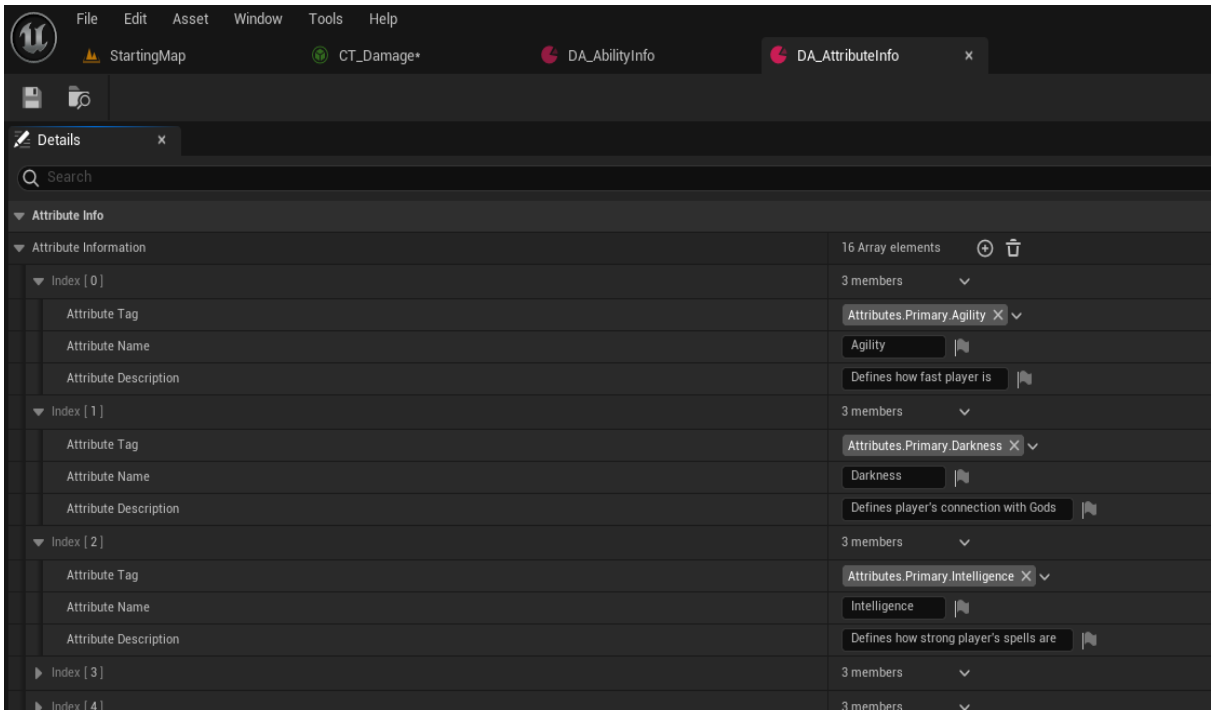


Рисунок 4.5 – Атрибути та інформація про них

Для більш складних прорахунків застосовуються спеціальні класи розрахунків. Це робиться для кращої читабельності алгоритму і розуміння ідеї того,

що відбувається у розрахунках, тобто з якими атрибутами виходить взаємодія та з якими акторами (рис. 4.6).

```
float UMMC_MaxHealth::CalculateBaseMagnitude_Implementation(const FGameplayEffectSpec&
Spec) const
{
    const FGameplayTagContainer* SourceTags =
Spec.CapturedSourceTags.GetAggregatedTags();
    const FGameplayTagContainer* TargetTags =
Spec.CapturedTargetTags.GetAggregatedTags();

    FAggregatorEvaluateParameters EvaluationParameters;
    EvaluationParameters.SourceTags = SourceTags;
    EvaluationParameters.TargetTags = TargetTags;

    float Strength = 0.f;
    GetCapturedAttributeMagnitude(StrengthDef, Spec, EvaluationParameters, Vigor);
    Vigor = FMath::Max<float>(Strength, 0.f);

    int32 PlayerLevel = 1;
    if (Spec.GetContext().GetSourceObject()->Implements<UCombatInterface>())
    {
        PlayerLevel =
ICombatInterface::Execute_GetPlayerLevel(Spec.GetContext().GetSourceObject());
    }

    return 80.f + 2.5f * Strength + 10.f * PlayerLevel;
}
```

Рисунок 4.6 – Ілюстрація фрагменту коду розрахунку максимального здоров’я

І окрім цього усі початкові атрибути та здібності напряму залежать від початкового класу гравця чи ворога, таким чином набір здібностей надається на початку гри відповідно до дата ассету, в якому можна знайти усю необхідну інформацію про класи персонажів та задати їхні початкові значення, які можуть бути також виражені у таблицях (рис. 4.7). Усього існує три класи: Воїн, Ренжер та Маг.

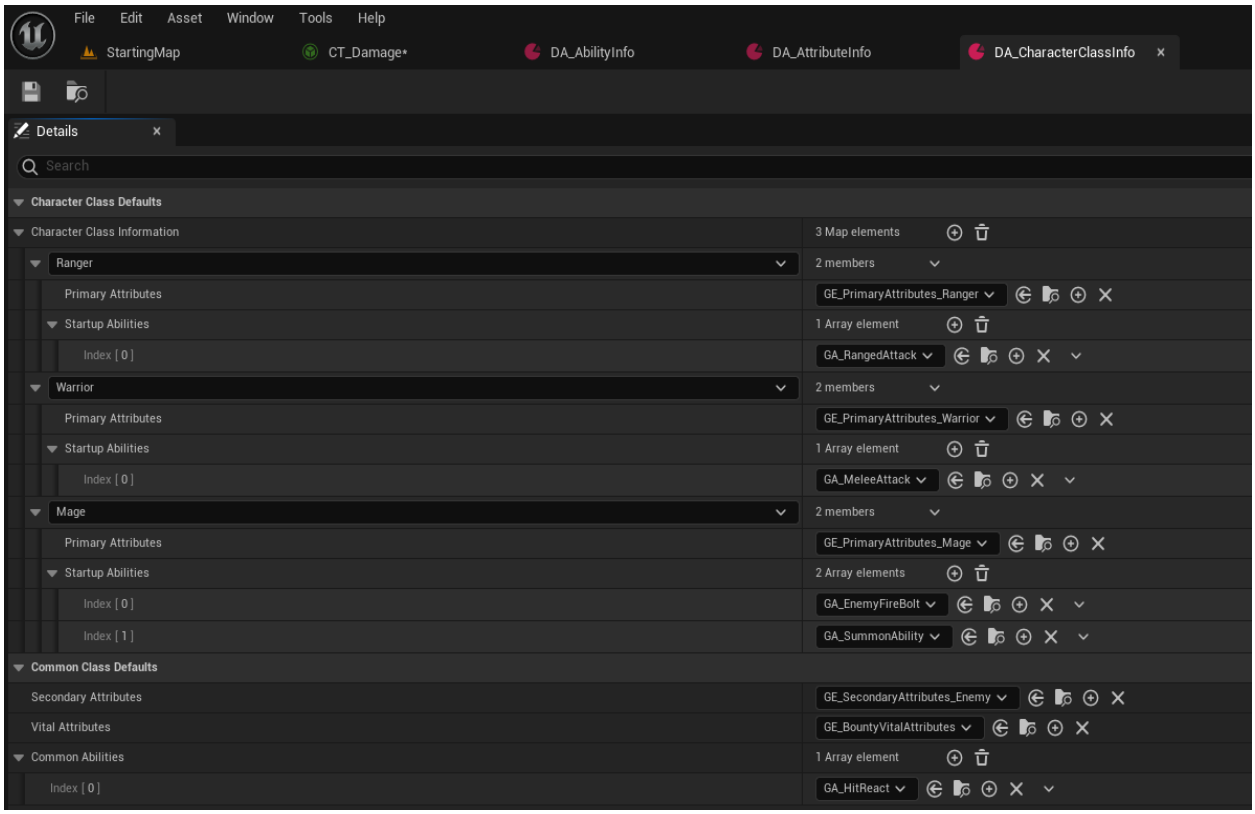


Рисунок 4.7 – Дата ассет початкових здібностей та атрибутів згідно з класами

Таким чином виглядає система атрибутів і відбувається балансування, прямо у редакторі, навіть під час самої гри для швидких змін, що дають змогу зрозуміти у якому напрямку потрібно робити нововведення. Це дає неймовірні можливості для подальшого розширення системи, без значних змін складності самої системи.

4.2 Система анімацій

Система анімацій є ще одною потужною частиною, що дозволяє легко розширювати кількість персонажів і здібності, не роблячи складнішим заміни існуючих анімацій чи додання нових (рис 4.8).

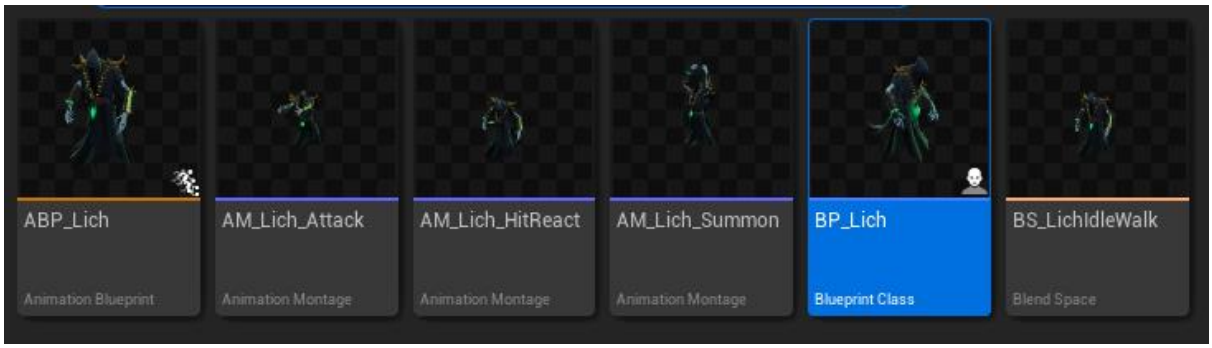


Рисунок 4.8 – Список анімацій для ворога Ліча

Також, ми не прив’язуємо здібність до анімації. Натомість, кожен персонаж має список тегів для монтажу та закріплених анімацій за цим монтажем (рис 4.9).

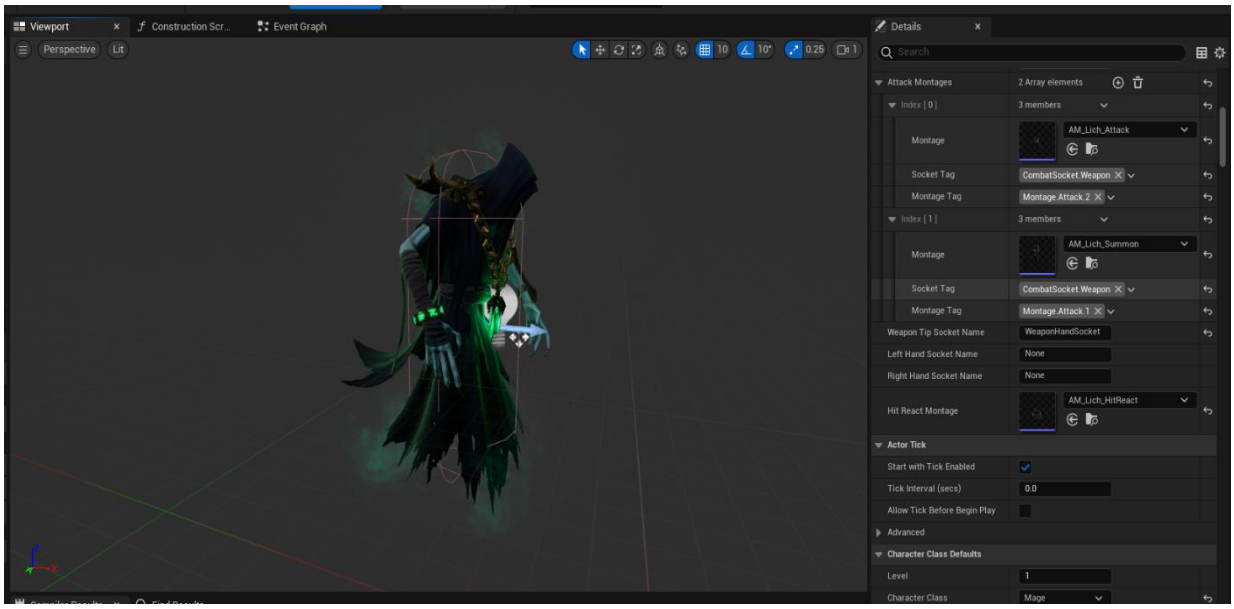


Рисунок 4.9 – Блюпринт ворогу Ліча з монтажами та анімаціями.

Закляття у свою чергу викликають необхідний монтаж згідно за тегом закріпленим за цим закляттям. Таким чином усі однакові закляття використовують один і той самий тег задля виклику схожих анімацій.

Анімації у свою чергу використовують систему оповіщень у який саме момент анімації має бути викликана здібність (рис. 4.10 та 4.11).

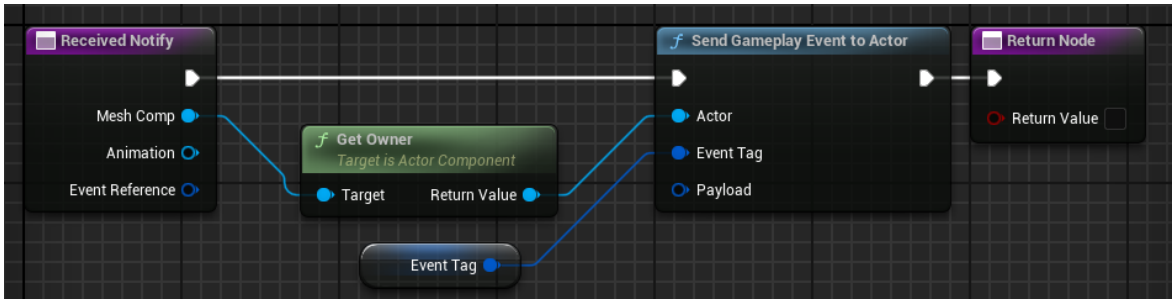


Рисунок 4.10 – Блюпринт оповіщення

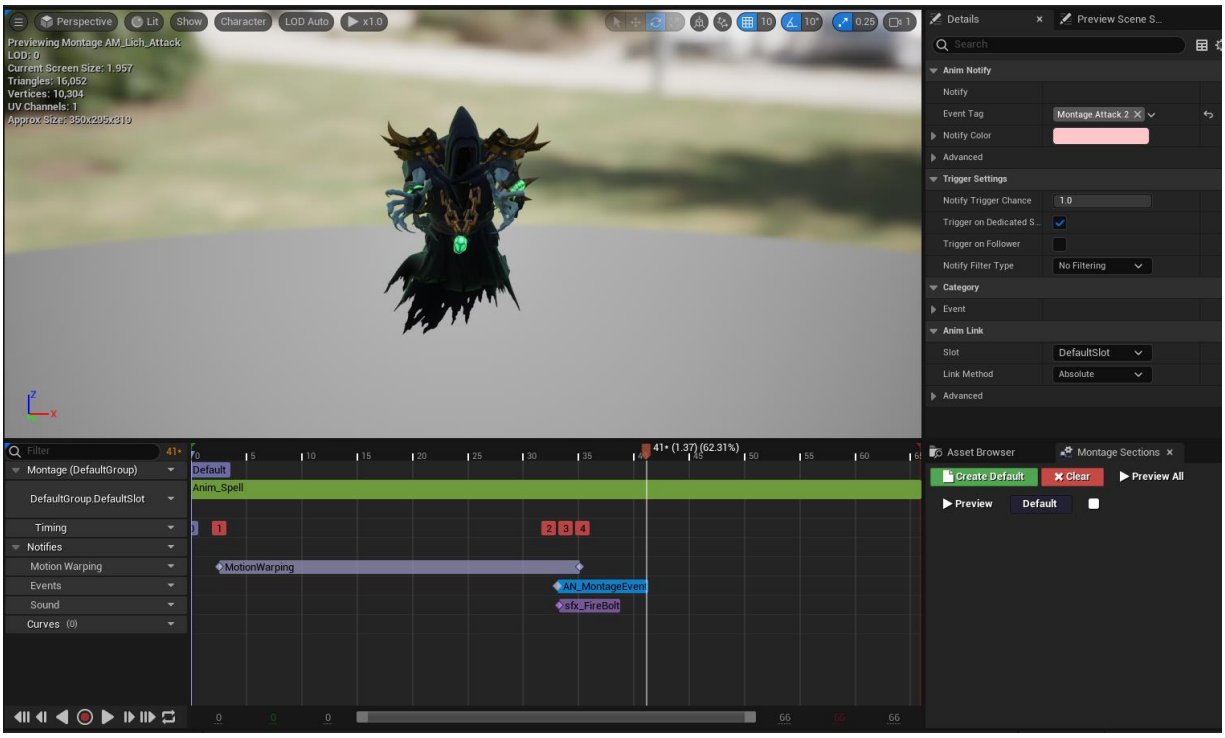


Рисунок 4.11 – Анімація Ліча та виклик оповіщення про застосуванні здібності.

Таким чином можна побачити яку саме здібність викличе ворог, не прив’язуючись до конкретного ворога, а залежачи від тегу монтажу, тобто вороги з однаковим скелетом будуть робити одну й ту саму дію, їм лише потрібно зазначити цю анімацію у них в блюпринті.

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування програмного забезпечення є невід'ємною частиною процесу розробки будь-якого програмного продукту, і ігрові застосунки не є виключенням. Зважаючи на складність сучасних ігор, які включають в себе різноманітні аспекти, від графіки і звуку до складних алгоритмів штучного інтелекту та мережевої взаємодії, процес тестування стає ще більш критичним. У цьому розділі будуть розглянуті такі аспекти тестування ігрових застосунків, як мануальне тестування та метод чорної і білої скриньки, їхню роль у забезпеченні якості гри, та інструменти, які використовуються для цього. Приклади тестових випадків, розроблених під час створення програмної системи, були наведені у таблиці 5.1 - 5.3.

Основна мета тестування ігрових застосунків полягає у виявленні та виправленні помилок, що можуть вплинути на функціональність, продуктивність та загальний користувацький досвід. Якісне тестування допомагає забезпечити стабільність гри, мінімізувати ймовірність крашів та багів, а також покращити загальну продуктивність ігрового продукту. Крім того, тестування дозволяє виявити проблеми, пов'язані з ігровим балансом, що є ключовим для забезпечення справедливості та задоволення гравців.

У процесі тестування програмного забезпечення ігрового застосунку кожен участник командного проекту тестував ту частину, яка була темою його кваліфікаційної роботи. Це дозволило глибоко зануритися в специфіку різних аспектів гри і забезпечити високу якість кінцевого продукту. Такий підхід не тільки підвищував мотивацію та зацікавленість учасників, але й забезпечував ефективність і точність процесу тестування.

Одним із ключових елементів гри є анімація, яка відповідає за візуальне сприйняття персонажів, оточення та всіх рухів у грі. Цей аспект і буде розглянуто у даному розділі. Спочатку будуть протестовані сінхронізації анімацій на предмет того, як анімації взаємодіють одна з одною. Та буде перевірка впливу анімацій на

продуктивність гри. Велика кількість складних анімацій може викликати затримки або зниження частоти кадрів, що негативно впливає на ігровий досвід.

Таблиця 5.1 – Тестовий випадок застосування анімації створення ворогів перед застосуванням здібності

Назва	Закликаю / не закликаю
Короткий опис	Ворог Lich застосовує анімацію створення ворогів перед застосуванням здібності Firebolt
Компонент програми	Геймплей. Здібності
Серйозність	S3 Значний
Пріоритет	P2 Середній
Кроки відтворення	Створити ворога Lich. Очікувати застосування здібності створення ворогів. Очікувати застосування здібності Firebolt
Фактичний результат	Ворог виконує однакову анімацію для однієї й тієї самої здібності
Очікуваний результат	Ворог виконує анімацію створення при застосуванні Firebolt

Таблиця 5.2 – Тестовий випадок фаєрбол без цілі

Назва	Фаєрбол без цілі
Короткий опис	Фаєрбол зависає на місці і залишається кружляти поряд з точкою
Компонент програми	Геймплей. Закляття
Серйозність	S3 Значний
Пріоритет	P2 Середній
Кроки відтворення	Запустити багато фаєрболів у режимі Homing. Після смерті ворога спостерігаємо баг
Фактичний результат	Фаєрбол залишається на точці біля зазначеної завчасно цілі і не зникає
Очікуваний результат	Фаєрбол не маючи цілі зникає через деякий короткий проміжок часу

Таблиця 5.3 – Ігровий застосунок перестає працювати, якщо у гравця 4 стартових здібності

Назва	Magic 4+
Короткий опис	Краш гри, якщо у гравця більше 4 стартових здібностей.
Компонент програми	Геймплей. UI. Здібності
Серйозність	S2 Критичний
Пріоритет	P1 Високий
Кроки відтворення	У файлі DA_AbilityInfo створити більше 4х записів здібностей. Почати гру. Відкрити меню прокачки заклять і прокачати єдине розблоковане закляття. Якщо баг не спрацював, то закрити меню і відкрити знову.
Фактичний результат	Гра автоматично закривається
Очікуваний результат	Ми бачимо прокачаний спел і можемо відкривати нові і переміщувати їх до слотів застосування

Результати тестування показують, що ігровий застосунок має кілька критичних і значних помилок, які можуть значно вплинути на гравців. Перше тестування виявило помилку в анімації здібностей ворога, що може викликати

непорозуміння і знизити загальний рівень задоволення від гри. Другий випадок демонструє проблему з закляттям, яке зависає без цілі, що може порушити баланс гри. Третій випадок виявив серйозну помилку, яка призводить до крашу гри при певних умовах, що критично впливає на ігровий процес і потребує негайного виправлення.

Таким чином, ці тестові випадки показують важливість ретельного тестування всіх аспектів гри, від анімації до функціональності та балансування. Виявлення і виправлення таких помилок є критично важливим для забезпечення стабільності, якості та привабливості ігрового продукту для кінцевих користувачів.

6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

6.1 Наукове впровадження проекту

На основі теми кваліфікаційної роботи була підготовлена наукова стаття у форматі тез для XXVIII Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті», який відбувся з 16 по 18 квітня 2024 року онлайн на базі Харківського Національного Університету Радіоелектроніки.

Наукова стаття «ВПЛИВ ГЕНЕРАЦІЇ РІВНІВ НА ГЕЙМПЛЕЙ ІГОР У ЖАНРІ ROGUELIKE» була опублікована в збірнику матеріалів конференції №6. [8].

У даній роботі тема була присвячена удосконаленню існуючого алгоритму підвищення складності генерації компонентів рівня.. У цьому тезісі також було розглянуто, як кількість кімнат певного типу та різні налаштування можуть впливати на складність ігор у жанрі Roguelike, а також наведено приклади того, як випадковість може змінити хід гри та сприяти або перешкоджати рівновазі.

6.2 Практичне впровадження проекту

На основі матеріалів кваліфікаційної роботи була взята участь у виставці технічної творчості молоді 2024 року, яка проходила онлайн на базі Харківського Національного Університету Радіоелектроніки з 16 по 18 квітня 2024 року.[9]

Під час виставки було представлено і захищено проект, основа мета котрого – проходження максимально можливої кількості хвиль ворогів, між якими гравець може покращувати свої атрибути та здібності або відкривати нові. Після поразки гравець повертається сильнішим, з поліпшеними здібностями, і може пройти гру знову. Вороги стають сильнішими з кожним рівнем. Декілька гравців можуть одночасно проходити один і той самий рівень.

Переваги розробки: кросплатформеність, простий і зрозумілий інтерфейс, легке впровадження нових значень дизайнерами за допомогою таблиць, підтримка мультиплеєрної гри.

ВИСНОВКИ

Під час виконання складової частини наданої комплексної роботи на тему «Ігровий програмний застосунок у жанрі Roguelite RPG: балансування, анімація, нарратив» було проведено аналіз предметної галузі, котрий визначив основних конкурентів у цій галузі, а також ігрові застосунки, які мають особливі механіки та відрізняються за жанром. У процесі проектування цього ігрового додатку враховувалися як переваги, так і недоліки кожного конкурента.

Після цього робота включала постановку завдань, яка включала створення атрибутів, їх вплив на закляття та супровідні ефекти. Було вирішено, які закляття мають використовуватися, щоб доповнити ігровий світ і створити анімації, котрі наповнять гру різнобарвними кольорами. Детальне проектування інтерфейсу користувача (UI/UX) було наступним етапом роботи. Було розроблено прототипи інтерфейсу гри та детально розглянуто основні інтерфейси для геймплею.

Розробка компоненту проекту - системи атрибутів, заслуговує особливої уваги. Були створені візуальні ефекти та компоненти для Unreal Engine, які відносяться до використання заклять. Були розроблені атрибути, які можуть впливати на хід гри, послаблюючи або покращуючи її бойову частину.

У процесі розробки ми також приділили значну увагу мануальному тестуванню, щоб забезпечити високу якість гри. Було проведено кілька тестових випадків, які допомогли ідентифікувати та виправити критичні помилки. Зокрема, ми тестували анімацію створення ворогів перед застосуванням здібності, виявивши, що ворог виконує однакову анімацію для різних здібностей. Інший тестовий випадок включав перевірку поведінки фаєрболів без цілі, де фаєрбол зависав на місці після смерті ворога. Ще один важливий тест виявив, що ігровий застосунок переставав працювати, якщо у гравця було більше чотирьох стартових здібностей. Усі ці тестові випадки дозволили вчасно виявити і виправити проблеми, що могли негативно вплинути на користувацький досвід.

Таким чином, кожен етап проектування та розробки був ретельно продуманий і протестований, що дозволило створити якісний ігровий продукт, який відрізняється стабільністю, збалансованістю та привабливістю для користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Joshua Bycer. Game Design Deep Dive: Roguelikes / Publisher: CRC Press, 2021h – 120c.(дата звернення: 11.05.2024).

2. Sebastian Deterding, José Zagal. Role-Playing Game Studies : Transmedia Foundations / Publisher: Routledge, 2018p – 494c(дата звернення: 11.05.2024).

3. Enter the Gungeon. URL: https://enterthegungeon.com/ (дата звернення: 13.05.2024).

4. Supergiant’s Games Hades URL: https://www.supergiantgames.com/games/hades/ (дата звернення 12.05.2024)

5. DRL (formerly DoomRL) URL: https://drl.chaosforge.org/ (дата звернення 12.05.2024)

6. Lumen. Unreal Engine 5 URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/lumen-global-illumination-and-reflections-in-unreal-engine?application_version=5.0 (дата звернення 10.05.2024)

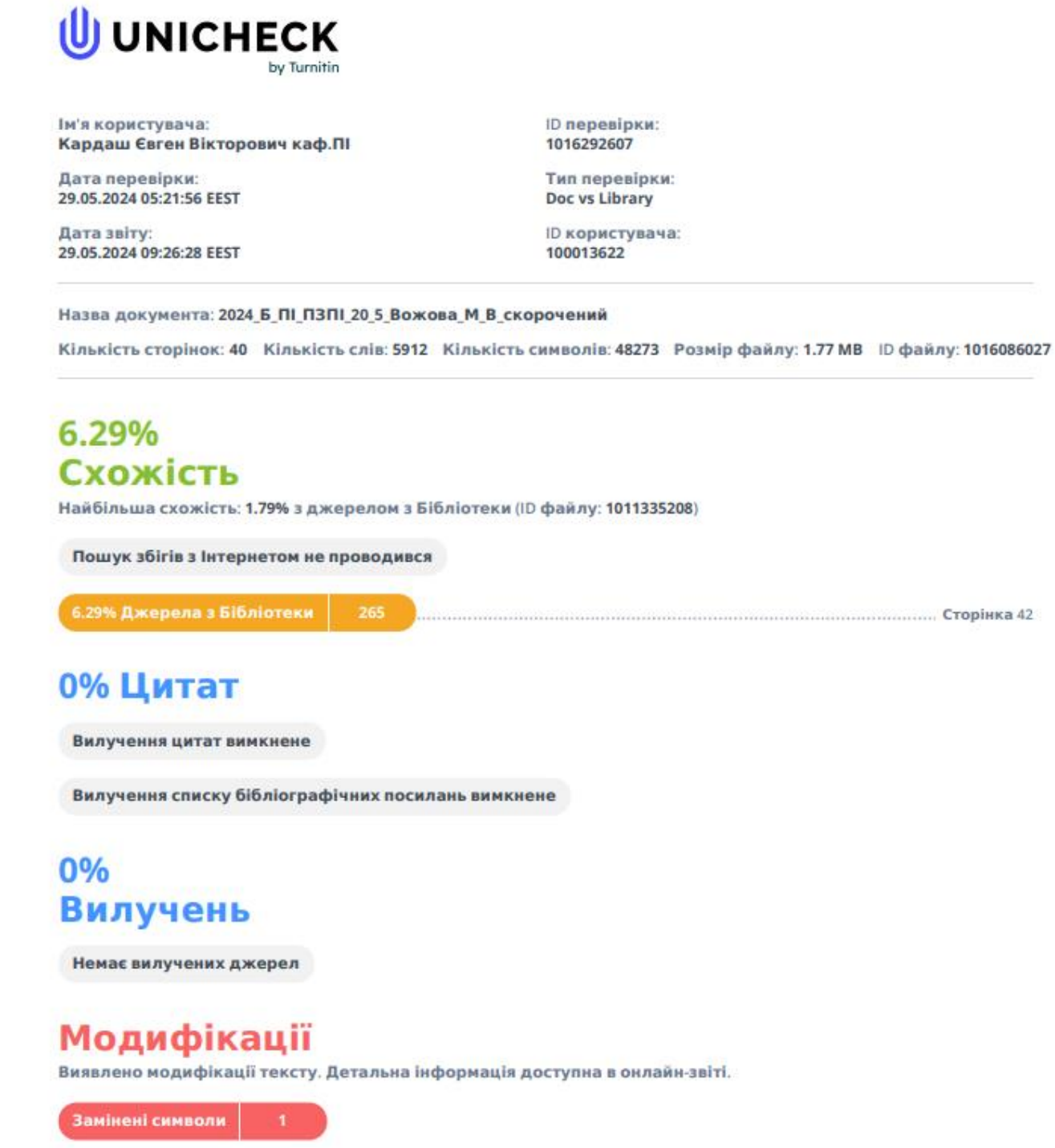
7. Nanite. Unreal Engine 5 URL: https://www.unrealengine.com/en-US/blog/understanding-nanite---unreal-engine-5-s-new-virtualized-geometry-system (дата звернення 10.05.2024)

8. Т. 6. Конференція «Комп'ютерної інженерії та захисту інформації». URL: https://openarchive.nure.ua/handle/document/26351 (дата звернення: 18.05.2024).

9. Каталог виставки технічної творчості молоді [Електронний ресурс]. URL: https://openarchive.nure.ua/handle/document/26355 (дата звернення: 18.05.2024).

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Кардаш Євген Вікторович каф.ПІ

ID перевірки:
1016292607

Дата перевірки:
29.05.2024 05:21:56 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
29.05.2024 09:26:28 EEST

ID користувача:
100013622

Назва документа: 2024_Б_ПІ_ПЗПІ_20_5_Вожова_М_В_скорочений

Кількість сторінок: 40 Кількість слів: 5912 Кількість символів: 48273 Розмір файлу: 1.77 MB ID файлу: 1016086027

6.29%
Схожість

Найбільша схожість: 1.79% з джерелом з Бібліотеки (ID файлу: 1011335208)

Пошук збігів з Інтернетом не проводився

6.29% Джерела з Бібліотеки 265 ..... Сторінка 42

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 1

ДОДАТОК Б

Слайди презентації

Рисунок Б.1 – Слайд 1

Рисунок Б.2 – Слайд 2

Рисунок Б.3 – Слайд 3

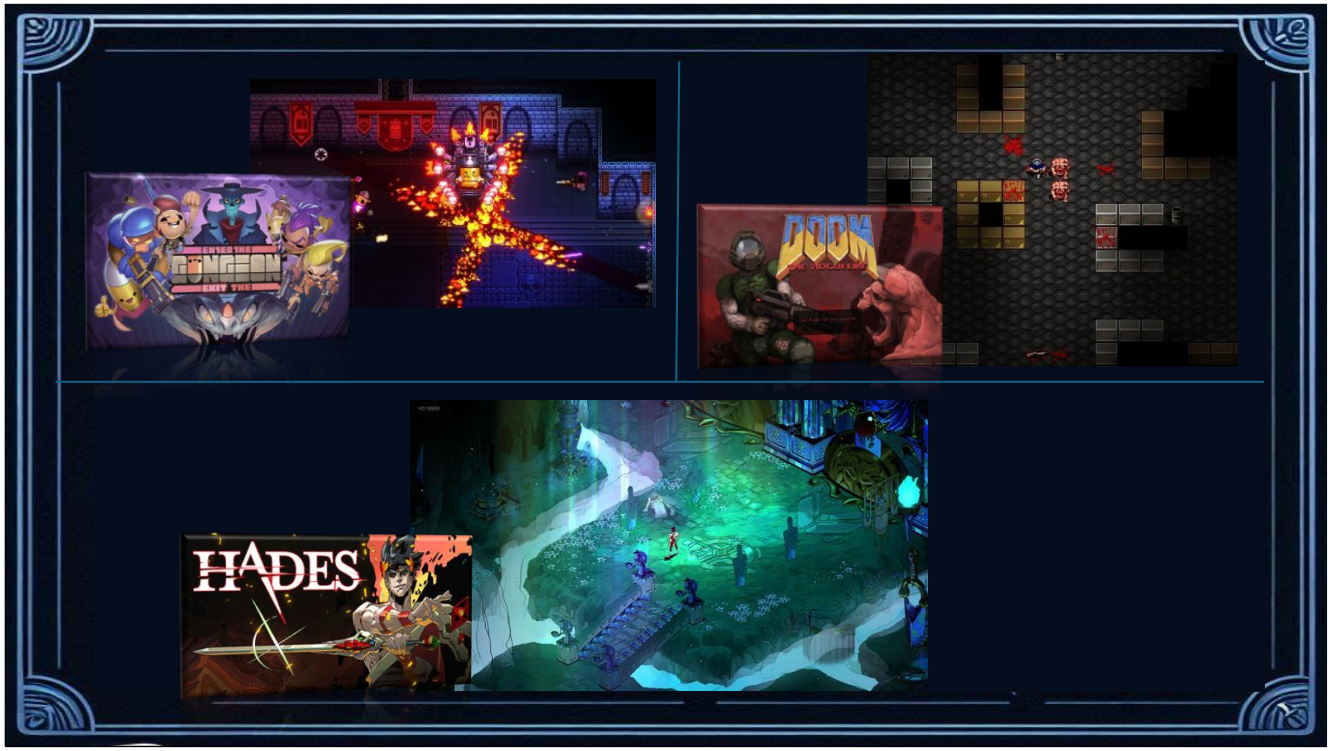


Рисунок Б.4 – Слайд 4

Рисунок Б.5 – Слайд 5

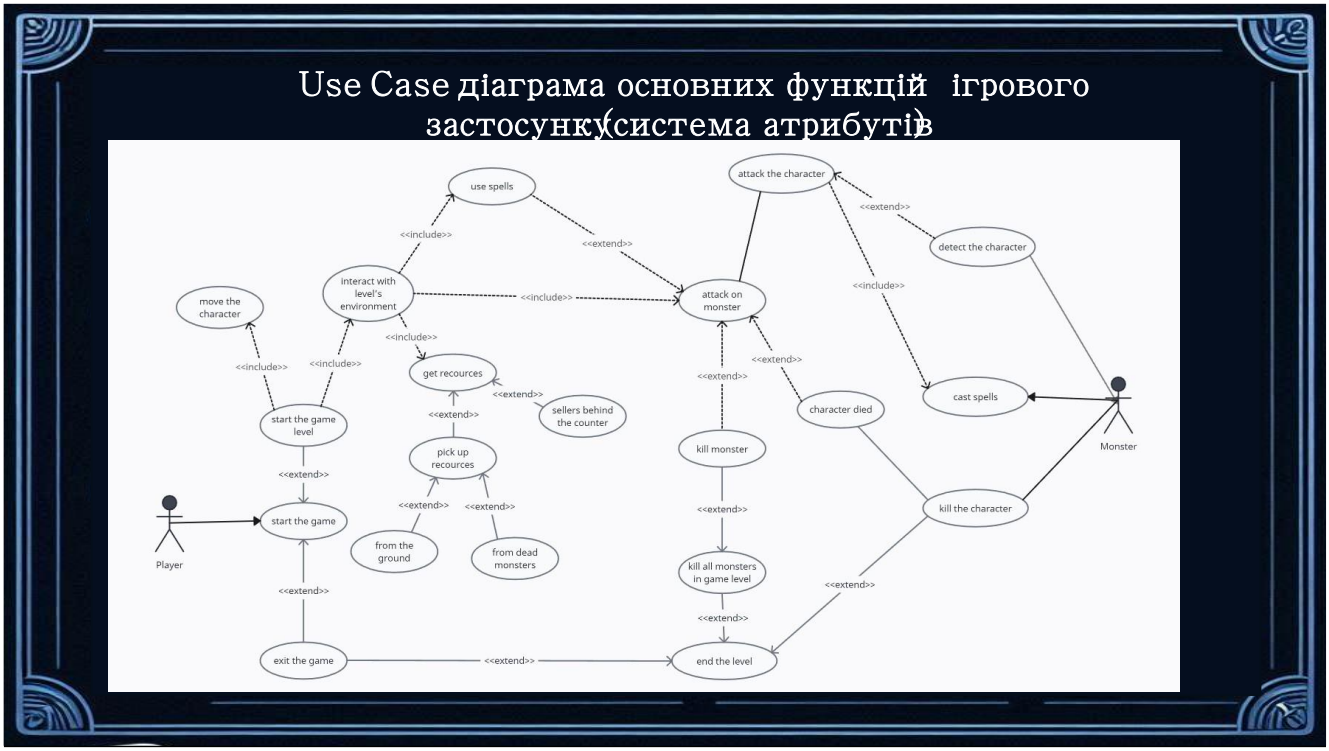


Рисунок Б.6 – Слайд 6

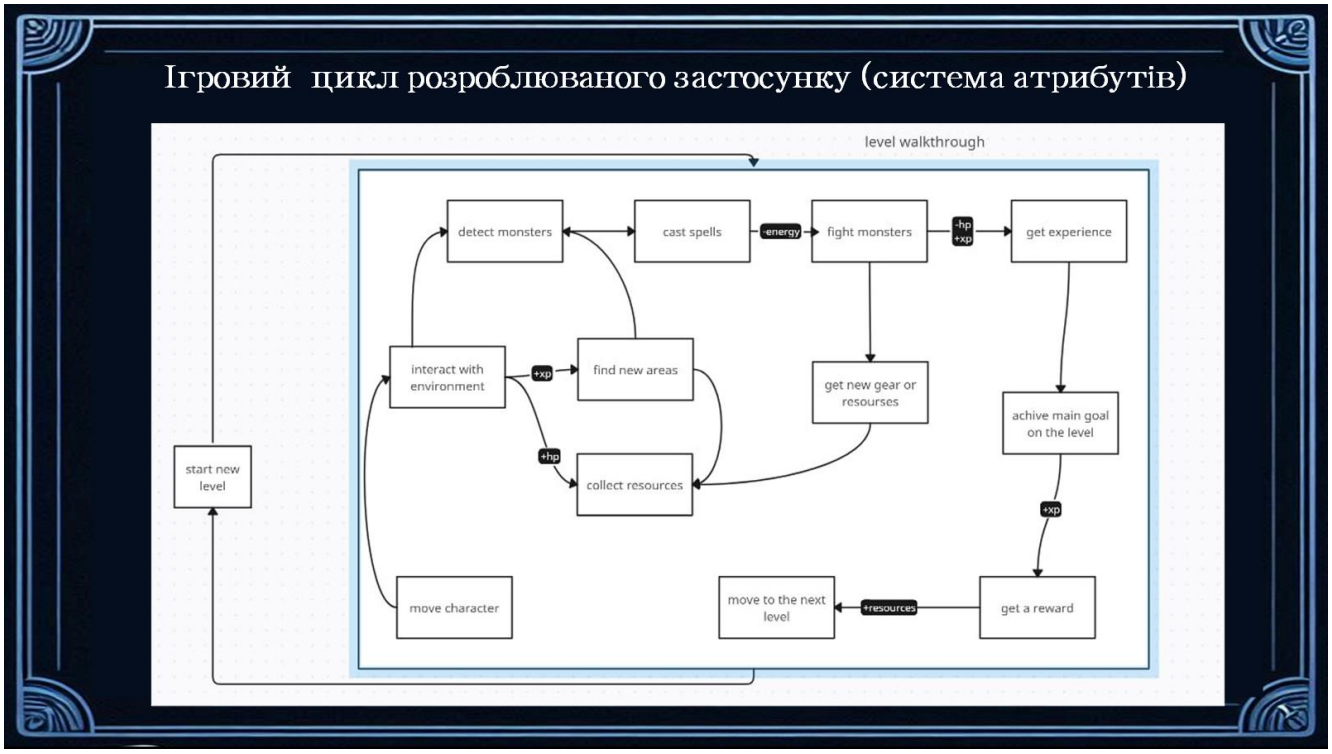


Рисунок Б.7 – Слайд 7

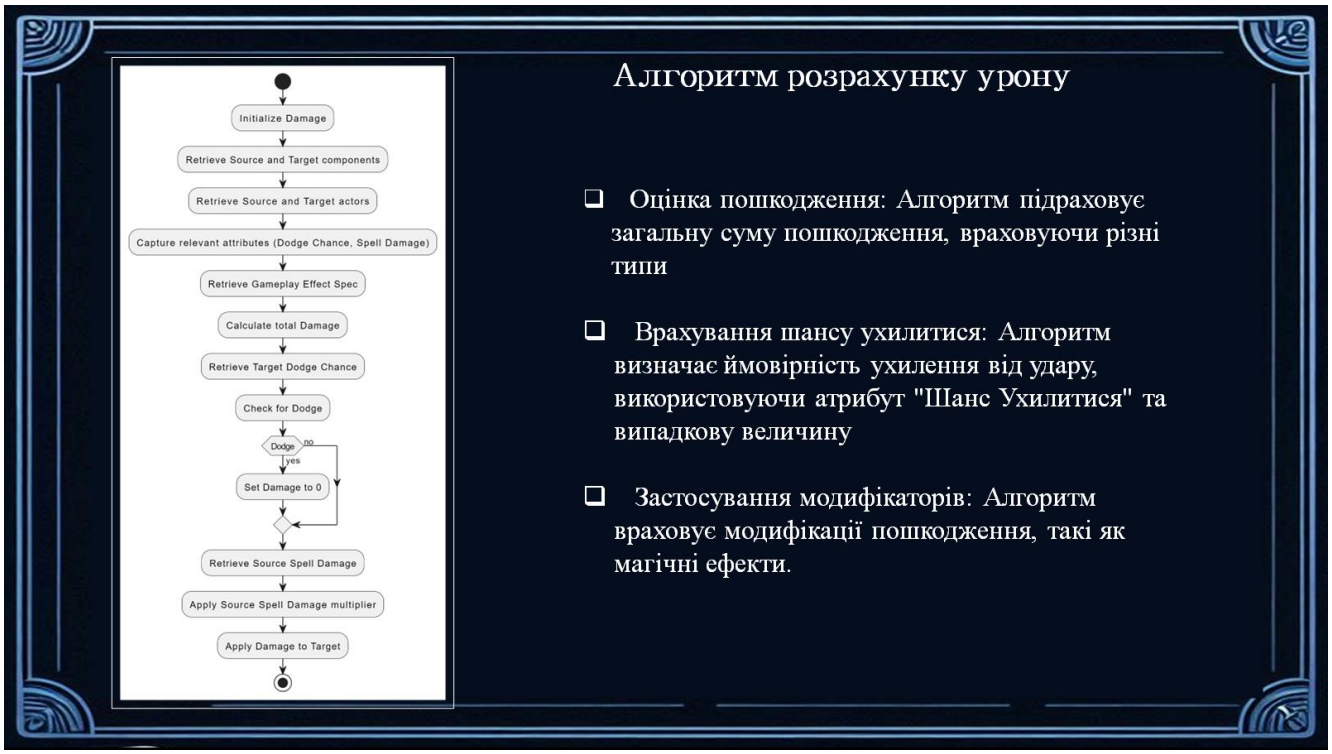


Рисунок Б.8– Слайд 8

Рисунок Б.9 – Слайд 9

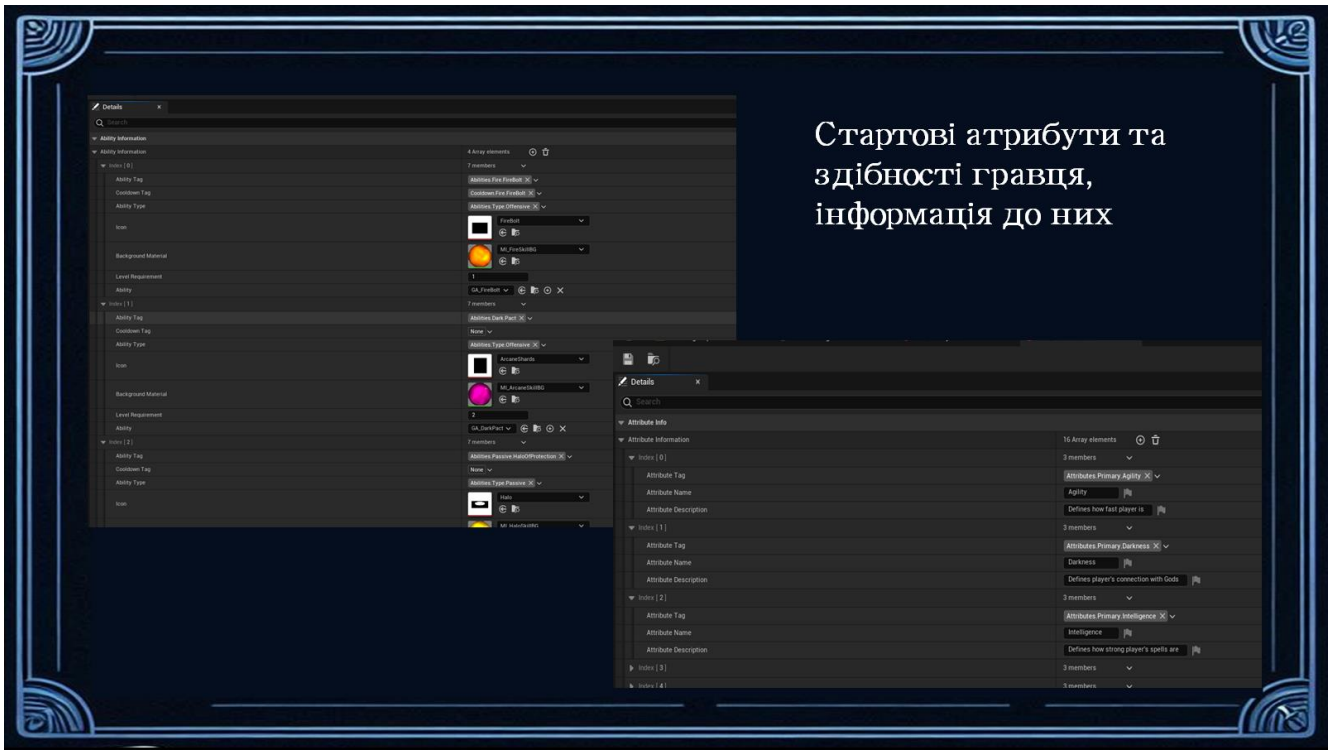


Рисунок Б.10 – Слайд 10

Рисунок Б.11 – Слайд 11

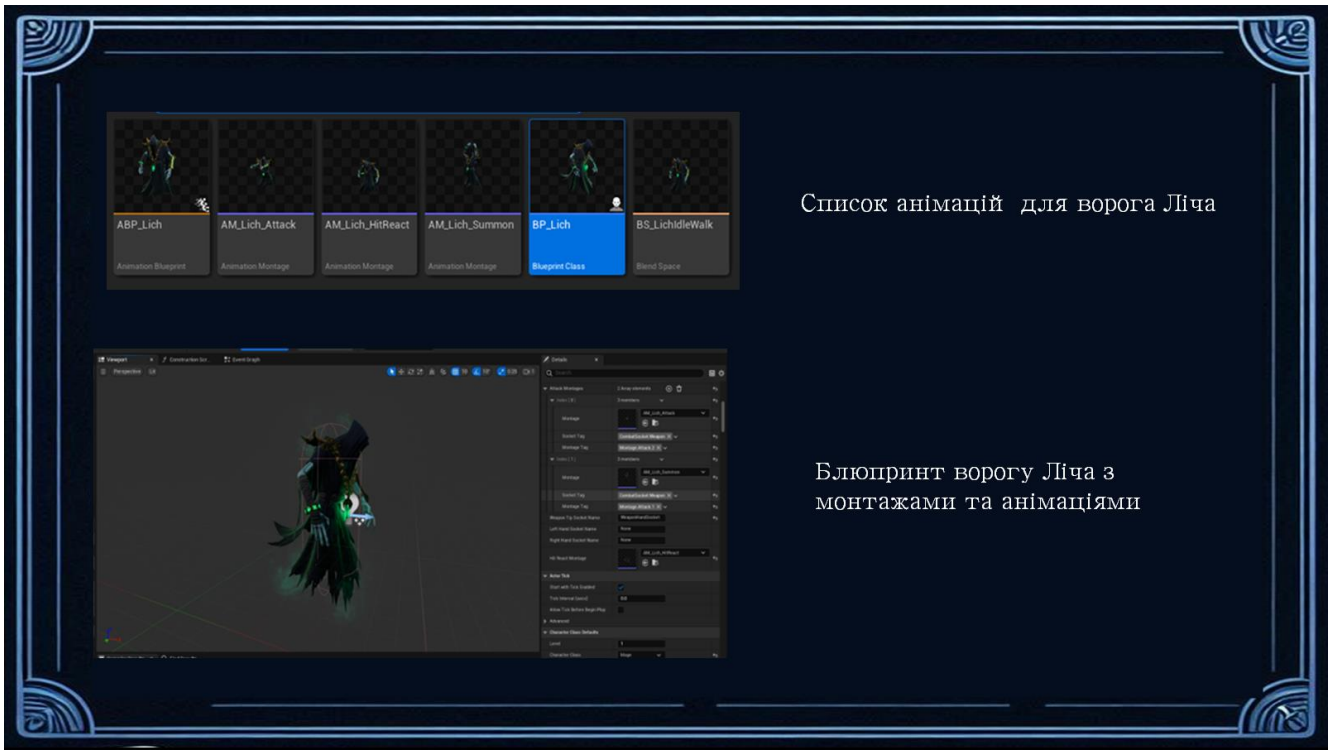


Рисунок Б.12 – Слайд 12

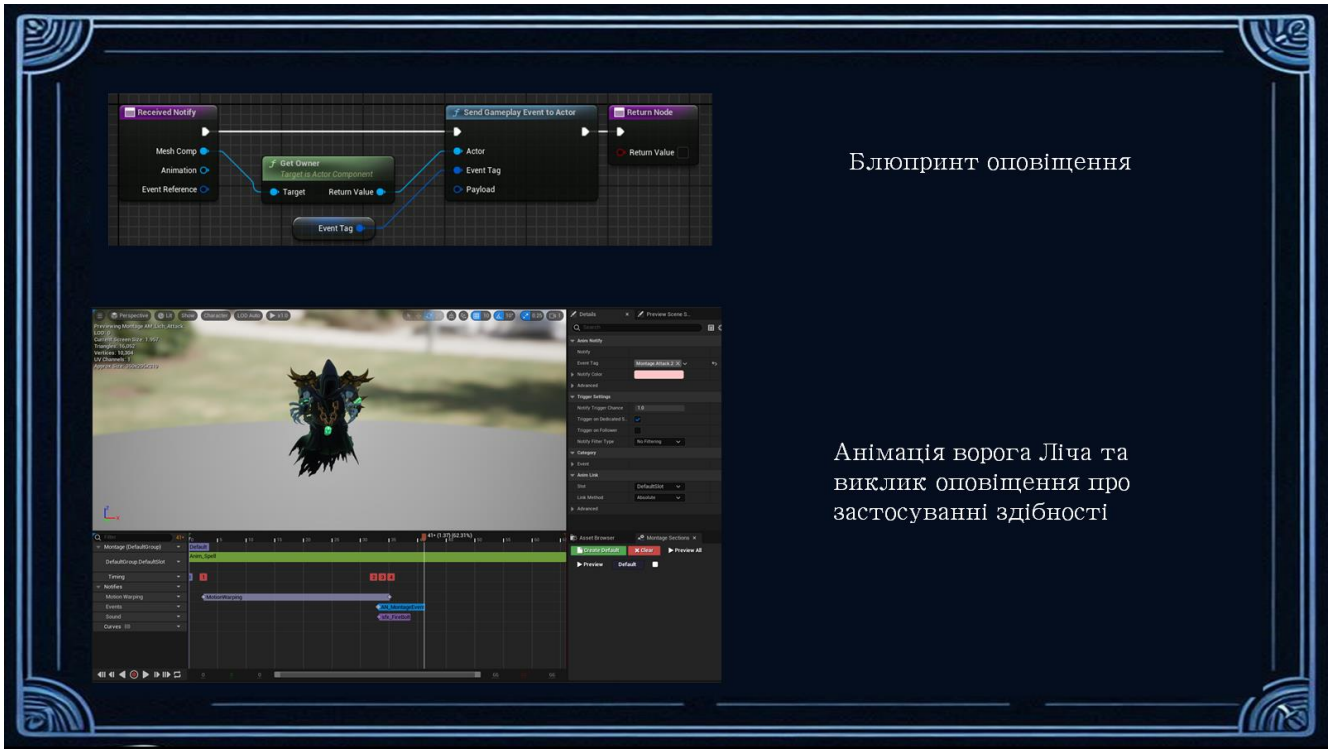


Рисунок Б.13– Слайд 13

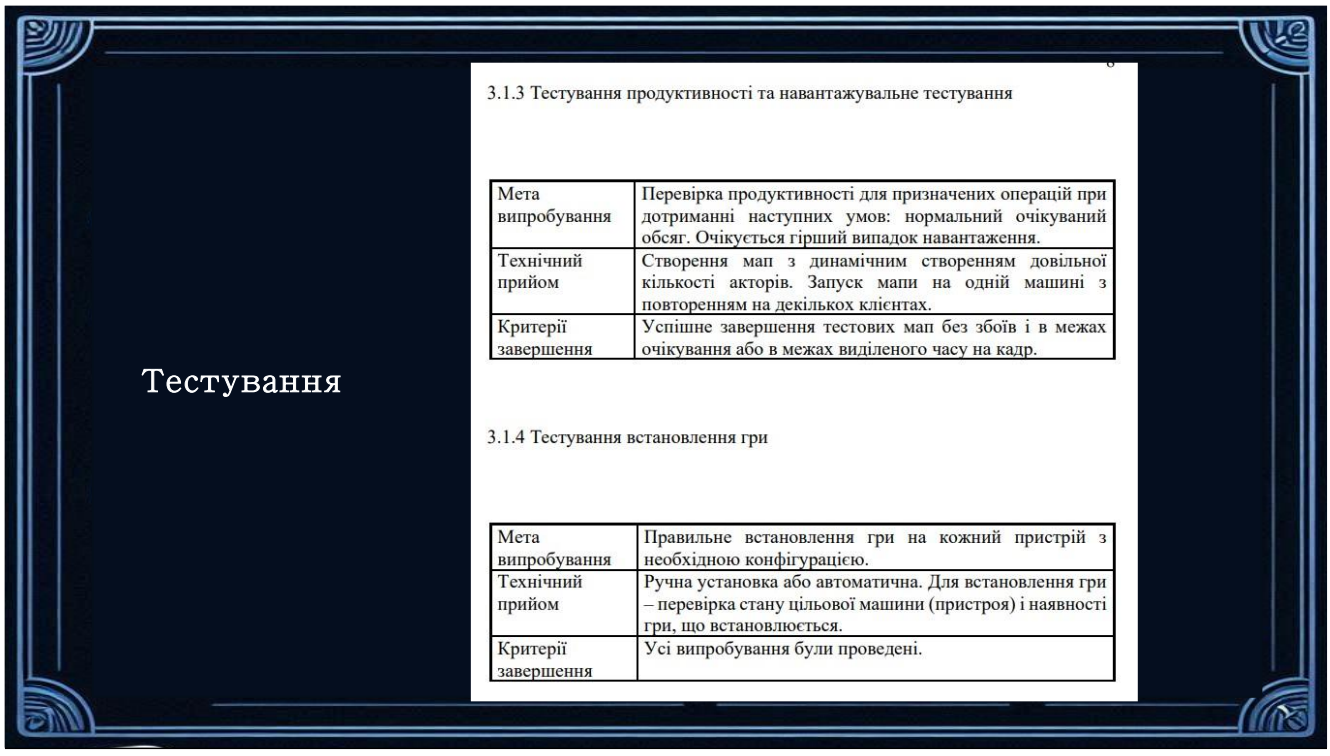
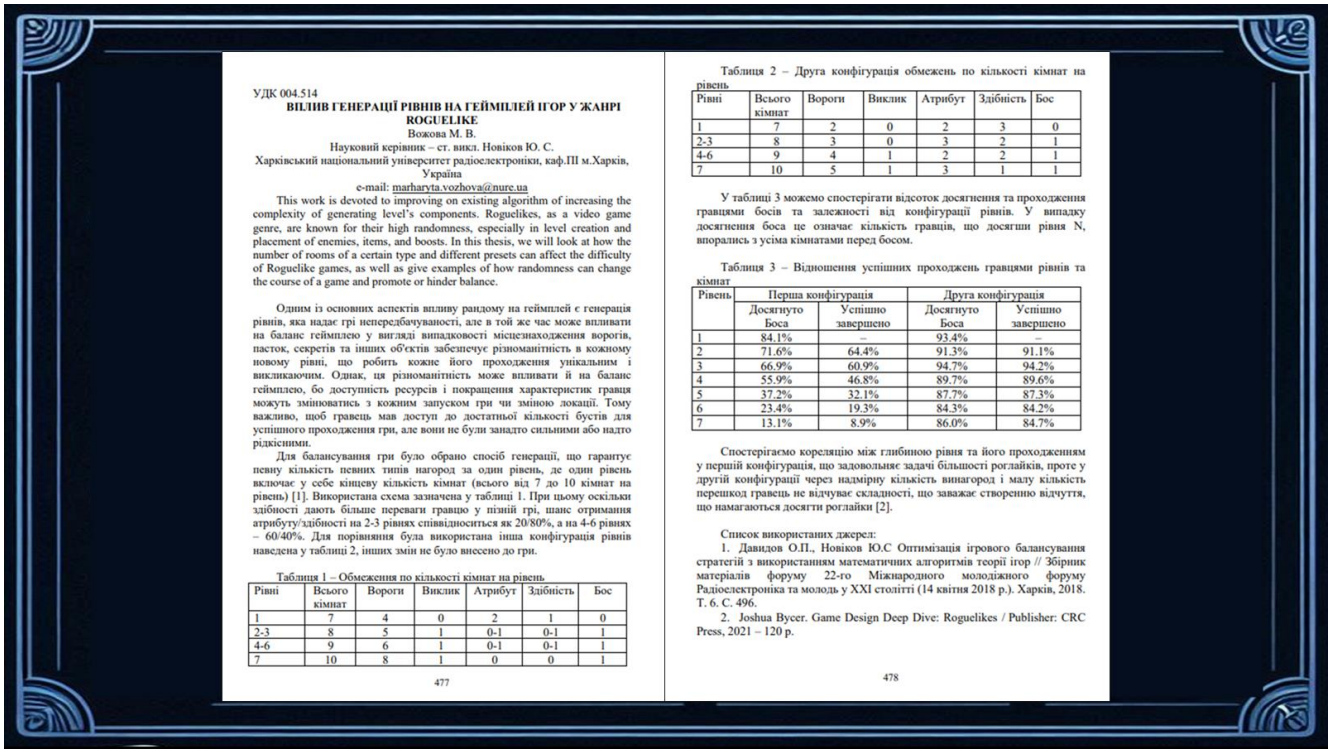


Рисунок Б.14– Слайд 14



УДК 004.514

ВПЛИВ ГЕНЕРАЦІЇ РІВНІВ НА ГЕЙМПЛЕЙ ІГОР У ЖАНРІ ROGUELIKE

Вожова М. В.
Науковий керівник – ст. викл. Новіков Ю. С.
Харківський національний університет радіоелектроніки, каф.ПІ м.Харків, Україна
e-mail: marharyta.vozhova@nure.ua

This work is devoted to improving on existing algorithm of increasing the complexity of generating level's components. Roguelikes, as a video game genre, are known for their high randomness, especially in level creation and placement of enemies, items, and boosts. In this thesis, we will look at how the number of rooms of a certain type and different presets can affect the difficulty of Roguelike games, as well as give examples of how randomness can change the course of a game and promote or hinder balance.

Одним із основних аспектів впливу рандому на геймплей є генерація рівнів, яка надає грі непередбачуваності, але в той же час може впливати на баланс геймплею у вигляді випадковості місцезнаходження ворогів, пасток, секретів та інших об'єктів забезпечує різноманітність в кожному новому рівні, що робить кожне його проходження унікальним і викликаючим. Однак, ця різноманітність може впливати й на баланс геймплею, бо доступність ресурсів і покращення характеристик гравця можуть змінюватись з кожним запуском гри чи зміною локації. Тому важливо, щоб гравець мав доступ до достатньої кількості бустів для успішного проходження гри, але вони не були занадто сильними або надто рідкісними.

Для балансування гри було обрано спосіб генерації, що гарантує певну кількість певних типів нагород за один рівень, де один рівень включає у себе кінцеву кількість кімнат (всього від 7 до 10 кімнат на рівень) [1]. Використана схема зазначена у таблиці 1. При цьому оскільки здібності дають більше переваги гравцю у пізній грі, шанс отримання атрибуту/здібності на 2-3 рівнях співвідноситься як 20/80%, а на 4-6 рівнях – 60/40%. Для порівняння була використана інша конфігурація рівнів наведена у таблиці 2, інших змін не було внесено до гри.

Таблиця 1 – Обмеження по кількості кімнат на рівень

Рівні	Всього кімнат	Вороги	Виклик	Атрибут	Здібність	Бос
1	7	4	0	2	1	0
2-3	8	5	1	0-1	0-1	1
4-6	9	6	1	0-1	0-1	1
7	10	8	1	0	0	1

477

Таблиця 2 – Друга конфігурація обмежень по кількості кімнат на рівень

Рівні	Всього кімнат	Вороги	Виклик	Атрибут	Здібність	Бос
1	7	2	0	2	3	0
2-3	8	3	0	3	2	1
4-6	9	4	1	2	2	1
7	10	5	1	3	1	1

У таблиці 3 можемо спостерігати відсоток досягнення та проходження гравцями босів та залежності від конфігурації рівнів. У випадку досягнення боса це означає кількість гравців, що досягши рівня N, впоралась з усіма кімнатами перед босом.

Таблиця 3 – Відношення успішних проходжень гравцями рівнів та кімнат

Рівень	Перша конфігурація		Друга конфігурація	
	Досягнуто Боса	Успішно завершено	Досягнуто Боса	Успішно завершено
1	84.1%	–	93.4%	–
2	71.6%	64.4%	91.3%	91.1%
3	66.9%	60.9%	94.7%	94.2%
4	55.9%	46.8%	89.7%	89.6%
5	37.2%	32.1%	87.7%	87.3%
6	23.4%	19.3%	84.3%	84.2%
7	13.1%	8.9%	86.0%	84.7%

Спостерігаємо кореляцію між глибиною рівня та його проходженням у першій конфігурація, що задовольняє задачі більшості роглайків, проте у другій конфігурації через надмірну кількість винагород і малу кількість перешкод гравець не відчуває складності, що заважає створенню відчуття, що намагаються досягти роглайки [2].

Список використаних джерел:

1. Давидов О.П., Новіков Ю.С Оптимізація ігрового балансування стратегій з використанням математичних алгоритмів теорії ігор // Збірник матеріалів форуму 22-го Міжнародного молодіжного форуму Радіоелектроніка та молодь у XXI столітті (14 квітня 2018 р.). Харків, 2018. Т. 6. С. 496.
2. Joshua Bycer. Game Design Deep Dive: Roguelikes / Publisher: CRC Press, 2021 – 120 p.

478

Рисунок Б.15 – Слайд 15

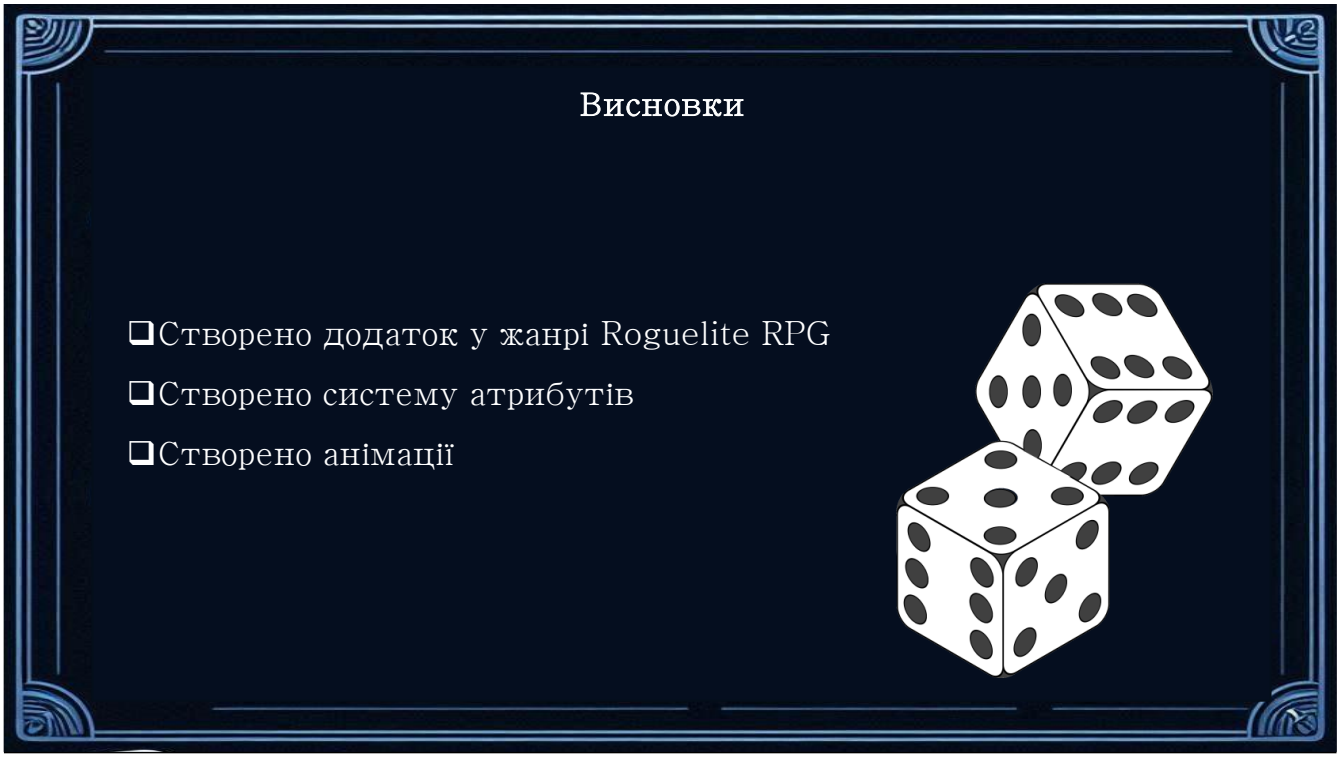


Рисунок Б.16– Слайд 16

Рисунок Б.17 – Слайд 17

ДОДАТОК В

Геймдизайн-документ

Chapter 2. Game Description

2.1. Game Process Description

2.1.1. Objective

The main objective is to kill the final boss of the game with every god and perk combination available. Player can do so by going through levels of enemies until they reach the final boss level.

Global objectives include unlocking new gods and their perks and spells for future runs. During the run main objectives are destroying enemies, finding spells, dicey-upgrades, special currencies e.g. gold, saving characters and completing quests.

2.1.2. Game Flow Structure

Game has non-linear story. It doesn't mean it has different start or ending, but in-between may vary depending on triggered events. For example, if you find one character before freeing another, first will not mention the second one in dialogue. Game loop consists of multiple levels. Player always starts in Level 0, a.k.a. Lobby. It is non-combat room full of NPCs where player can modify his pre-run God Ability and Perks. Also they can unlock new Spells and Gods here. When player starts the run appears in Level 1. This level consists of combos of rooms, specifically Combat Room, Mini-Boss Room, Boss Room, Quest Room, Shop Room, Chill Room. It repeats for the next levels, 4 levels in total representing unique locations. Every Level ends with Boss Room. When player dies within any level he returns to Level 0 with some resources retained. Estimated Gameplay Duration 20-40 minutes, where every level consists of 8-10 rooms with some exceptions. Gameplay Duration by room is depicted on the table, where level are divided into parts due to rising difficulty.

Table 3. Gameplay Duration by Level

	Combat Room	Mini-Boss Room	Boss Room	Quest Room	Shop Room	Chill Room
Level 0	N/A	N/A	N/A	10-30s	N/A	10-30s
Level 1, Pt. 1	10-20s	N/A	N/A	15s	15s	10-30s
Level 1, Pt. 2	15-25s	20-30s	40-60s	15-20s	15s	10-30s
Level 2, Pt. 1	10-25s	25-35s	N/A	15-20s	15-20s	10-30s
Level 2, Pt. 2	17-30s	25-35s	50-70s	15-25s	15-20s	10-30s
Level 3, Pt. 1	15-25s	30-40s	N/A	15-20s	15-20s	10-30s
Level 3, Pt. 2	20-30s	35-45s	60-90s	15-20s	15-20s	10-30s
Level 4, Pt. 1	10-35s	40-45s	N/A	15-20s	15-20s	10-30s
Level 4, Pt. 2	N/A	N/A	90-120s	N/A	N/A	N/A

Рисунок В.1 – Опис ігрового проєкту з геймдизайн-документу

2.2.5. Status Effects

Status Effect is a temporary modification to the character, it may be both positive and negative, applied to enemies and player himself.

Bountiful: Drops additional gold on death, but be quick, it can expire!
Drunk: Move and attack slower.
Hex: Can't attack or use spells.
Immortal: Can't be slowed or damaged.

2.2.6. Passive Abilities

Passive Ability is a persistant status effect, usually more complex.

Thorny Faces: After you take damage, deal 30 damage to all enemies near you.
Golden Bubble: Absorb any instance of damage. Restores in 20 seconds.
Lucky Three: After you cast three spells ***Invoke***

2.2.7. Some Keywords

Anvil: Anvil falls on your opponents, dealing 100 damage in small area.
Bountiful: Drops additional gold on death.
Burn: Lose Dice Pact after use.
Choose One: Give a choice of three items. Player can select one of them.
Critical: Deal +150% damage.
Endless: Does not expire after exiting combat.
Gamble: Randomly choose between Positive and Negative effect.
Haste: Move and Attack 30% faster.
Hex: Can't attack or use spells.
Immortal: Can't be slowed or damaged.
Invoke: Cast random spell.
Refresh: Replenish your God's Call Charge or Spell Charge instantly.
Wind: Projectile that slows attack speed and move speed of your enemies by 30%.
Wither: Take 30 damage per second for 6 seconds.

2.2.8. Dice Rolls

Dice Roll is inspired by dash abilities in different games, however in game it looks like dice *rolls* at certain speed for defined distance. Speed and Distance may vary depending on status effects and passive abilities. Player has 3 charges to roll, they replenish after some time that player doesn't roll. Apart from this, player can cast *dice abilities* by rolling.

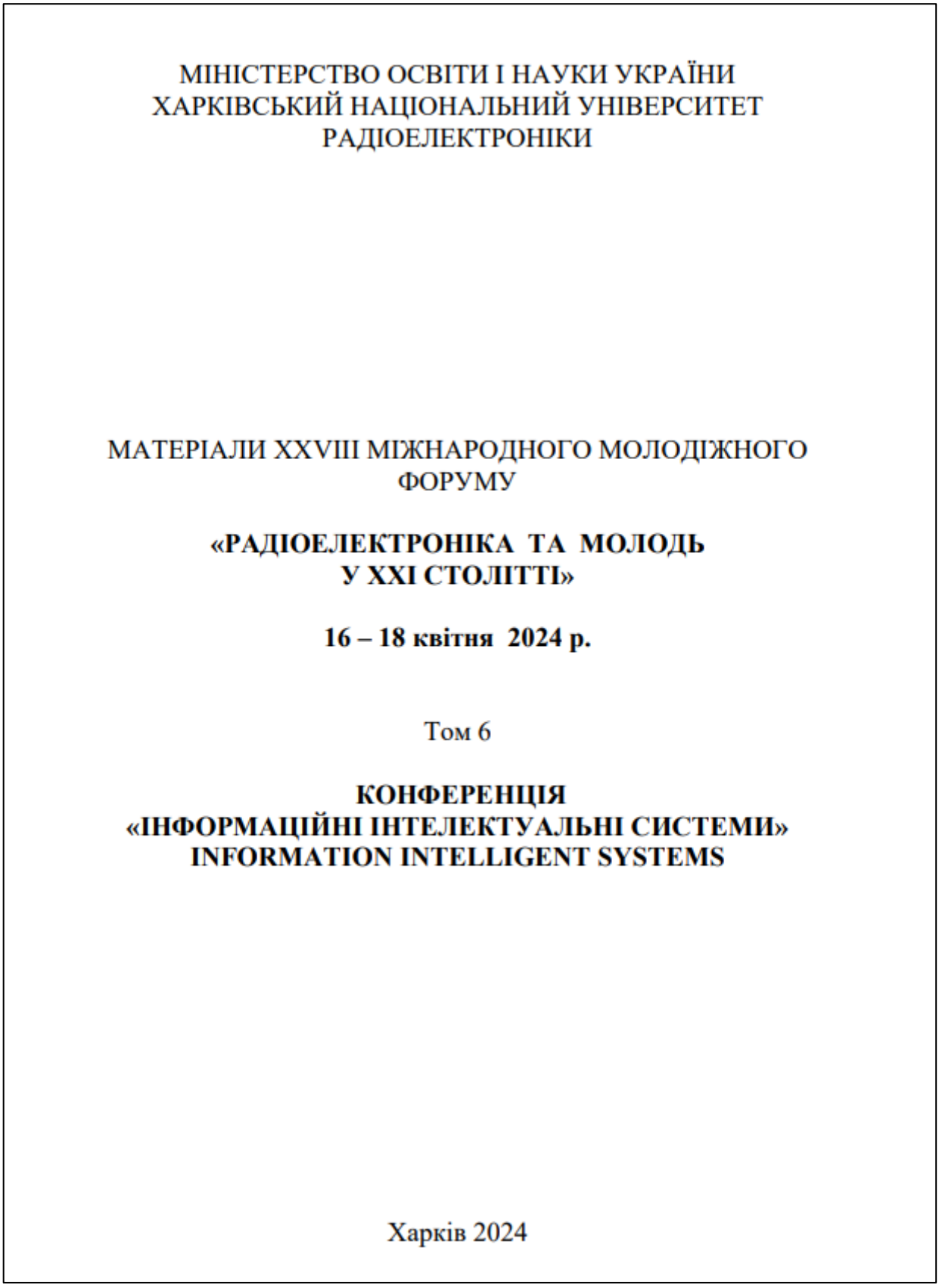
2.2.9. Dice Ability

Player casts dice ability by rolling 3 times in a row. The ability is determined based on combination of elements that were on dice' faces where player landed. If player lands on the face he has already

Рисунок В.2 – Опис основних ігрових механік

ДОДАТОК Г

Тези доповіді для науково-практичної інтернет-конференції



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
РАДІОЕЛЕКТРОНІКИ

МАТЕРІАЛИ XXVIII МІЖНАРОДНОГО МОЛОДІЖНОГО ФОРУМУ

«РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ»

16 – 18 квітня 2024 р.

Том 6

КОНФЕРЕНЦІЯ
«ІНФОРМАЦІЙНІ ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ»
INFORMATION INTELLIGENT SYSTEMS

Харків 2024

Рисунок Г.1 – Обкладинка збірника

АЛФАВІТНИЙ ПОКАЖЧИК

A

Avrunin O., 77

B

Bilokon V. A., 94

G

Gorishnia K. O., 402
Grebennik, 574
Grebennik I. V., 719

H

Hadzhyiev E. R., 834

I

Ivashyn S. S., 71

K

Khodyka O. S., 816
Khovrat A.V., 295
Kiprich Ivan, 510
Klymenko D. A., 191
Kobziev V. G., 295, 402
Kravets N., 448
Kupriianov S., 77

P

Pekaruk I. O., 719

R

Reshetnik V. M., 816
Ruzhitskyi S. V., 574
Ryabova N. V., 94

S

Savanevych V. E., 834
Seliutin D. A., 257
Shekhovtsova V. I., 191
Shergin V.L., 71
Smelyakov Kirill, 510

V

Vashchenko M., 448

Y

Yashyna O. S., 257

А

Абросімов Є. О., 35
Аврунін О. Г., 121
Агатін Є. Л., 359
Адамов О. С., 83
Алєксєєв Д. Д., 466
Андрєєв В. Р., 212
Андрєєв І. Г., 428
Антонов В. А., 723
Ареф'єв О. О., 513
Артеменко А. Д., 875
Артюхов М. А., 802
Афонькін Д. Д., 142
Ахтирський О. Ю., 792

Б

Бабій Д. В., 689
Бакала Ю. О., 776
Балюк І. В., 650
Барна К. М., 888
Батраченко В. О., 699
Башкіров М. О., 625
Беберіна К. О., 212
Бедрата Р. Р., 909
Безгодков С. Р., 732
Безкоровайний В. В., 748, 814
Безугла Г. Є., 875, 877, 880, 888
Безуглий Н. С., 663
Белименко В. С., 673
Бєліков Д. Ю., 468
Бєлінський Г. А., 484
Бзот С. В., 691
Бізюк А. В., 931
Біла Д. С., 916
Білий М. Д., 472
Білова Т. Г., 571, 593, 671, 723, 818
Білогур М. М., 778
Білоконь Б. О., 56
Бірюкова Ю. І., 597
Бовдуй Р.В., 40
Богун В. М., 562
Бодянський Є. В., 47, 128, 130
Бойко О. В., 97, 102
Бондаренко А. А., 504
Бондаренко Є. О., 532
Бондаренко К. О., 708
Борисенко А. Е., 390
Ботуз В. В., 116
Бочаров В. О., 320
Бочаров Г. І., 144
Брандт Н. М., 147
Бронов І. В., 557
Брухтій С. С., 918
Бугай Д. Ю., 314
Бурика О. О., 751
Бурим М. В., 806
Бурцева А.Д., 37
Бухало В. О., 375
Бухановський В. О., 61

В

Валенда Н. А., 314, 355
Валентій О. М., 646
Варданян К. А., 760
Варламов М. Д., 675
Васильцова Н. В., 155, 194, 219, 231, 241
Васильченко В. В., 812
Веретельніков Д. М., 149
Вечур О. В., 124, 490
Виноградов М. Ю., 766
Винокур О. О., 867
Вионг Куок За Бао, 661
Височин А. О., 800
Вишняк М. Ю., 610, 721, 736
Власенко Л. А., 466
Вовк О. В., 914
Вожова М. В., 477
Волоховський В. Є., 118
Воронова Д. С., 99
Ворочек О. Г., 320, 516, 656

Г

Гавриш Д. Л., 332
Гаденко В. Ю., 28
Галуза О. А., 421
Галюк Д. Ю., 669
Гвоздьов Р. Ю., 430
Гімонов С. В., 152
Гладкий Д. П., 693
Гладченко О. О., 353
Глусенко А.С., 303
Гмиря І. О., 387
Говдерчак А. П., 773
Голобородько Б. Ю., 408
Головін М. С., 153
Головянко М. В., 99
Голуб Д. К., 610
Голян В. В., 562
Голян Н. В., 522, 560
Горбань І. Ю., 155
Гордієнко А. О., 628
Горенський Г. Г., 136
Горішня К. О., 416
Горюнова М. С., 591
Границя А. В., 440
Гребеннік І. В., 615, 635
Гребенюк М. О., 890
Гребєнік О. А., 430
Греков О. О., 47
Гречка А. О., 432

953

Рисунок Г.2 – Алфавітний покажчик збірника

УДК 004.514

ВПЛИВ ГЕНЕРАЦІЇ РІВНІВ НА ГЕЙМПЛЕЙ ІГОР У ЖАНРІ ROGUELIKE

Вожова М. В.
Науковий керівник – ст. викл. Новіков Ю. С.
Харківський національний університет радіоелектроніки, каф.ПІ м.Харків, Україна
e-mail: marharyta.vozhova@nure.ua

This work is devoted to improving on existing algorithm of increasing the complexity of generating level's components. Roguelikes, as a video game genre, are known for their high randomness, especially in level creation and placement of enemies, items, and boosts. In this thesis, we will look at how the number of rooms of a certain type and different presets can affect the difficulty of Roguelike games, as well as give examples of how randomness can change the course of a game and promote or hinder balance.

Одним із основних аспектів впливу рандому на геймплей є генерація рівнів, яка надає грі непередбачуваності, але в той же час може впливати на баланс геймплею у вигляді випадковості місцезнаходження ворогів, пасток, секретів та інших об'єктів забезпечує різноманітність в кожному новому рівні, що робить кожне його проходження унікальним і викликаючим. Однак, ця різноманітність може впливати й на баланс геймплею, бо доступність ресурсів і покращення характеристик гравця можуть змінюватись з кожним запуском гри чи зміною локації. Тому важливо, щоб гравець мав доступ до достатньої кількості бустів для успішного проходження гри, але вони не були занадто сильними або надто рідкісними.

Для балансування гри було обрано спосіб генерації, що гарантує певну кількість певних типів нагород за один рівень, де один рівень включає у себе кінцеву кількість кімнат (всього від 7 до 10 кімнат на рівень) [1]. Використана схема зазначена у таблиці 1. При цьому оскільки здібності дають більше переваги гравцю у пізній грі, шанс отримання атрибуту/здібності на 2-3 рівнях співвідноситься як 20/80%, а на 4-6 рівнях – 60/40%. Для порівняння була використана інша конфігурація рівнів наведена у таблиці 2, інших змін не було внесено до гри.

Таблиця 1 – Обмеження по кількості кімнат на рівень

Рівні	Всього кімнат	Вороги	Виклик	Атрибут	Здібність	Бос
1	7	4	0	2	1	0
2-3	8	5	1	0-1	0-1	1
4-6	9	6	1	0-1	0-1	1
7	10	8	1	0	0	1

Рисунок Г.3 – Матеріал власних тез (сторінка 1)

Таблиця 2 – Друга конфігурація обмежень по кількості кімнат на рівень

Рівні	Всього кімнат	Вороги	Виклик	Атрибут	Здібність	Бос
1	7	2	0	2	3	0
2-3	8	3	0	3	2	1
4-6	9	4	1	2	2	1
7	10	5	1	3	1	1

У таблиці 3 можемо спостерігати відсоток досягнення та проходження гравцями босів та залежності від конфігурації рівнів. У випадку досягнення боса це означає кількість гравців, що досягши рівня N, впорались з усіма кімнатами перед босом.

Таблиця 3 – Відношення успішних проходжень гравцями рівнів та кімнат

Рівень	Перша конфігурація		Друга конфігурація	
	Досягнуто Боса	Успішно завершено	Досягнуто Боса	Успішно завершено
1	84.1%	–	93.4%	–
2	71.6%	64.4%	91.3%	91.1%
3	66.9%	60.9%	94.7%	94.2%
4	55.9%	46.8%	89.7%	89.6%
5	37.2%	32.1%	87.7%	87.3%
6	23.4%	19.3%	84.3%	84.2%
7	13.1%	8.9%	86.0%	84.7%

Спостерігаємо кореляцію між глибиною рівня та його проходженням у першій конфігурація, що задовольняє задачі більшості роглайків, проте у другій конфігурації через надмірну кількість винагород і малу кількість перешкод гравець не відчуває складності, що заважає створенню відчуття, що намагаються досягти роглайки [2].

Список використаних джерел:

1. Давидов О.П., Новіков Ю.С Оптимізація ігрового балансування стратегій з використанням математичних алгоритмів теорії ігор // Збірник матеріалів форуму 22-го Міжнародного молодіжного форуму Радіоелектроніка та молодь у XXI столітті (14 квітня 2018 р.). Харків, 2018. Т. 6. С. 496.

2. Joshua Bycer. Game Design Deep Dive: Roguelikes / Publisher: CRC Press, 2021 – 120 p.

Рисунок Г.4 – Матеріал власних тез (сторінка 2)

ДОДАТОК Ґ

Тези доповіді для науково-практичної інтернет-виставки

Рисунок Ґ.1 – Обкладинка каталогу виставки

10. Ігровий програмний застосунок у жанрі Roguelite RPG. Bounty Chase

Автори: ***Донець Дмитро Сергійович,*** ст. гр. ПЗПІ-20-2, ***Вожова Маргарита Володимирівна,*** ст. гр. ПЗПІ-20-5, ХНУРЕ.
Науковий керівник: Новіков Юрій Сергійович, старший викладач. каф. ПІ, ХНУРЕ.

Ігровий програмний застосунок у жанрі Roguelite RPG “Bounty Chase” створений для проведення вільного часу.

Основа мета гри – проходження якомога більшої кількості хвиль ворогів, між якими гравець може покращувати свої атрибути та здібності, або відкривати нові для нього. Окрім цього гравець після поразки повертається сильнішим, маючи сильніші здібності і може проходити гру знову за бажанням. Вороги посилюються з рівнем. Декілька гравців можуть проходити разом один і той самий рівень одночасно.

Переваги розробки: кросплатформеність, простий та зрозумілий інтерфейс, легке впровадження дизайнерами нових значень за допомогою таблиць, підтримка мультиплеєрної гри.

Рисунок Ґ.2 – Опис роботи для презентації

ДОДАТОК Д

Приклад програмного коду

Алгоритм розрахунку урону

```
#include "AbilitySystem/ExecCalc/ExecCalc_Damage.h"

#include "AbilitySystemComponent.h"
#include "BountyGameplayTags.h"
#include "BountyAbilityTypes.h"
#include "AbilitySystem/BountyAbilitySystemLibrary.h"
#include "AbilitySystem/BountyAttributeSet.h"

struct BountyDamageStatics
{
  DECLARE_ATTRIBUTE_CAPTUREDEF(DodgeChance);
  DECLARE_ATTRIBUTE_CAPTUREDEF(SpellDamage);

  BountyDamageStatics()
  {
    DEFINE_ATTRIBUTE_CAPTUREDEF(UBountyAttributeSet, DodgeChance, Target,
false);
    DEFINE_ATTRIBUTE_CAPTUREDEF(UBountyAttributeSet, SpellDamage, Source,
false);
  }
};

static const BountyDamageStatics& DamageStatics()
{
  static BountyDamageStatics DStatics;
  return DStatics;
}

UExecCalc_Damage::UExecCalc_Damage()
{
  RelevantAttributesToCapture.Add(DamageStatics().DodgeChanceDef);
  RelevantAttributesToCapture.Add(DamageStatics().SpellDamageDef);
}
```

```
void UExecCalc_Damage::Execute_Implementation(const
FGameplayEffectCustomExecutionParameters& ExecutionParams,
  FGameplayEffectCustomExecutionOutput& OutExecutionOutput) const
{
  const UAbilitySystemComponent* SourceASC =
ExecutionParams.GetSourceAbilitySystemComponent();
  const UAbilitySystemComponent* TargetASC =
ExecutionParams.GetTargetAbilitySystemComponent();

  const AActor* SourceAvatar = SourceASC ? SourceASC->GetAvatarActor() :
nullptr;
  const AActor* TargetAvatar = TargetASC ? TargetASC->GetAvatarActor() :
nullptr;

  const FGameplayEffectSpec& Spec = ExecutionParams.GetOwningSpec();

  const FGameplayTagContainer* SourceTags =
Spec.CapturedSourceTags.GetAggregatedTags();
  const FGameplayTagContainer* TargetTags =
Spec.CapturedTargetTags.GetAggregatedTags();
  FAggregatorEvaluateParameters EvaluationParameters;
  EvaluationParameters.SourceTags = SourceTags;
  EvaluationParameters.TargetTags = TargetTags;

  float Damage = 0.f;
  for (FGameplayTag DamageTypeTag : FBountyGameplayTags::Get().DamageTypes)
  {
    const float DamageTypeValue =
Spec.GetSetByCallerMagnitude(DamageTypeTag, false);
    Damage += DamageTypeValue;
  }

  float TargetDodgeChance = 0.f;

ExecutionParams.AttemptCalculateCapturedAttributeMagnitude(DamageStatics().
DodgeChanceDef, EvaluationParameters, TargetDodgeChance);
  TargetDodgeChance = FMath::Max<float>(TargetDodgeChance, 0.f);
```

```
  const bool bDodged = FMath::RandRange(1, 100) < TargetDodgeChance;

  FGameplayEffectContextHandle EffectContextHandle = Spec.GetContext();

  UBountyAbilitySystemLibrary::SetIsDodgedHit(EffectContextHandle,
bDodged);

  Damage = bDodged ? 0 : Damage;

  float SourceSpellDamage = 0.f;

ExecutionParams.AttemptCalculateCapturedAttributeMagnitude(DamageStatics().
SpellDamageDef, EvaluationParameters, SourceSpellDamage);
  SourceSpellDamage = FMath::Max<float>(SourceSpellDamage, 0.f);

  Damage *= 1 + SourceSpellDamage;

  const FGameplayModifierEvaluatedData
EvaluatedData(UBountyAttributeSet::GetIncomingDamageAttribute(),
EGameplayModOp::Additive, Damage);
  OutExecutionOutput.AddOutputModifier(EvaluatedData);
}
```