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 RPG з елементами економічної стратегії та roguelike. Механіки генерації рівнів, механіки на рівні підземель
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-4

Долгий Андрій Іванович
(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 ст.викл. кафедри ПІ Новіков Ю.С.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(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
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Долгому Андрію Івановичу
(прізвище, ім’я, по батькові)

1. Тема роботи Ігровий програмний застосунок у жанрі RPG з елементами економічної стратегії та roguelike. Механіки генерації рівнів, механіки на рівні підземель
Затверджена наказом по університету від 20.05.2024р. № 471 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 06.06.2024
3. Вихідні дані до роботи Розробити ігровий програмний застосунок у жанрі RPG з елементами стратегії і roguelike. Створити механіки генерації рівнів та механіки на рівні підземель. Ігровий програмний застосунок має містити у собі два компоненти: компонент із стратегічною розбудовою власного поселення та програмний модуль із roguelike специфікою, такою як генерація рівнів, їх дослідження та пов’язані із цим механіки. Зазначені частини мають складатися у єдиний проєкт. При розробці використовуються мова програмування C#, рушій для розробок ігор Unity3D (із вбудованими модулями та можливостями), графічні матеріали створені власноруч або запозичені (з урахуванням авторського права), допоміжні засоби розробки та відповідні інструменти.
4. Перелік питань, що потрібно опрацювати в роботі
Вступ, аналіз предметної галузі, формування вимог до ігрового застосунку, архітектура та проєктування програмного модулю, опис прийнятих програмних рішень, тестування програмного модулю, впровадження програмного модулю, медійна діяльність, 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	15.04.2024	*виконано*
3	Проєктування ПЗ	25.04.2024	*виконано*
4	Розробка ПЗ	15.05.2024	*виконано*
5	Тестування ПЗ	19.05.2024	*виконано*
6	Оформлення пояснювальної записки	23.05.2024	*виконано*
7	Підготовка презентації та доповіді	24.05.2024	*виконано*
8	Попередній захист	25.05.2024	*виконано*
9	Нормоконтроль, рецензування	26.05.2024	*виконано*
10	Здача роботи у електронний архів	02.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	03.06.2024	*виконано*
12	Захист кваліфікаційної роботи	06.06.2024	*виконано*

Дата видачі завдання 08 квітня 2024р.

Студент (ка)_____ Долгий А. І.
(підпис)

Керівник роботи _____ ст.викл. кафедри ПІ Новіков Ю.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 197 стор., 169 рис., 1 табл., 32 джерела.

МУЛЬТИПЛАТФОРМОВА ГРА, БАЛАНСУВАННЯ ІГРОВОГО КОНТЕНТУ, БОЙОВА СИСТЕМА, ВОРОЖИЙ ШТУЧНИЙ ІНТЕЛЕКТ, ГРАФІЧНІ ЕФЕКТИ VFX, КРАЩІ ІГРОВІ ПРАКТИКИ, МЕДІЙНИЙ КОНТЕНТ, МОВА ПРОГРАМУВАННЯ C#, ПАТЕРНИ ПРОЄКТУВАННЯ, РУШІЙ ДЛЯ СТВОРЕННЯ ІГОР UNITY3D, СИСТЕМА КОНТРОЛЮ ВЕРСІЙ GIT, СТИХІЙНА СИСТЕМА, ІГРОВИЙ ПРОГРАМНИЙ ЗАСТОСУНОК, AI NAVIGATION, GITHUB, INPUT SYSTEM, ROGUELIKE, THE FATE OF THE LIAR, VFX GRAPH, VISUAL CODE

Метою кваліфікаційної роботи постає створення частини гри для комплексного проєкту «The fate of the liar». До згаданої половини продукту відноситься розробка roguelike складової, а саме ігровий процес дослідження підземель із відповідними можливостями.

Об’єктом кваліфікаційної роботи є відтворення ігрового процесу з відвідування підземель, а саме реалізація окремного програмного модулю, у рушії для створення ігор Unity3D, у якому відтворено функціональність із генерації рівнів, бойових, виживатських та дослідницьких можливостей, разом із пов’язаними механіками та системами.

Для втілення кваліфікаційної роботи було застосовано рушій для створення ігрових додатків – Unity3D. Для написання коду компонентів гри, використано мову програмування C#. Для поєднання розрізнених частин комплексного проєкту було застосовано систему контролю версій Git, зокрема сервіс GitHub, що в свою чергу спростило розробку у групі. У якості IDE для написання коду та його модифікацій використано Visual Code із відповідними, до потреб розробки, розширеннями, оскільки зазначений редактор коду є достатньо швидким, зручним у використанні та легко модифікуємим необхідними компонентами, що значно

прискорює процес розробки. Для створення візуальних ефектів (VFX) було використано VFX Graph – компонент рушія Unity3D, що надає глибокий та наочно зрозумілий інструментарій для реалізації гарних симуляцій частинок тощо. Для впровадження частини із переміщенням до штучного інтелекту ворогів, було використано пакет AI Navigation, для пошуку шляхів та побудови маршрутів для противників персонажу користувача. Задля впровадження більш глибокої та модифікуємої системи управління, було використано нове API рушія Unity3D, а саме пакет Input System. Для створення великих комплексних систем, підвищення модифікуємості та підтримуємості разом із чистотою коду, було застосовано кращі практики для реалізації відповідних компонентів у поєднанні з використанням патернів проєктування тощо. У якості графічних матеріалів для створення наповнення, було взято безкоштовні наробки з інтернету, що розповсюджуються за CC0 - правилом про авторські права, зроблені власноруч або створенні за допомогою штучного інтелекту, що також підпадає під згадане право власності тощо.

Результатом виконання кваліфікаційної роботи стала реалізація ігрового процесу з дослідження підземель у вигаданому світі гри «The fate of the liar», яка включає у себе створення комплексних систем:

- випадкова генерація рівнів;
- відтворення бойових механік, що включають стихійну систему, особливі вміння головного герою та базові можливості, такі як перекат, простий удар тощо;
- штучний інтелект ворогів, який базується на низці правил та складається з різних наборів вмінь, поведінок та особистих параметрів;
- дослідницькі можливості для різноманіття ігрового процесу випадково згенерованих підземель;
- супутні механіки, додаткові сервіси та допоміжні системи для покращення ігрового досвіду та процесу тощо.

У доповнення до створеної частини гри «The fate of the liar», було проведено медійну діяльність, для збору зацікавленої аудиторії для майбутнього випуску

поточного проєкту та наступних. Також розроблений програмний модуль було теоретично апробовано на Міжнародній науковій конференції та практично продемонстровано на виставці Міжнародного молодіжного форуму.

MULTIPLATFORM GAMEPLAY, GAME CONTENT BALANCE, COMBAT SYSTEM, ENEMY ARTIFICIAL INTELLIGENCE, GRAPHIC EFFECTS, VFX, GAMING BEST PRACTICES, MEDIA CONTENT, C# PROGRAMMING LANGUAGE, DESIGN PATTERNS, UNITY3D GAME ENGINE, GIT VERSION CONTROL SYSTEM, ORIGINAL SYSTEM, GAME SOFTWARE APPLICATION, AI NAVIGATION, GITHUB, INPUT SYSTEM, ROGUELIKE, THE FATE OF THE LIAR, VFX GRAPH, VISUAL CODE

The purpose of the certification work is to create a part of the game for the collective project "The fate of the liar". The mentioned half of the product includes the development of the roguelike component, namely the dungeon exploration gameplay with the corresponding features.

The object of the certification work is to reproduce the gameplay from visiting dungeons, namely the implementation of a separate software module in the Unity3D game engine, which reproduces the functionality of generating levels, combat, survival and research opportunities, together with related mechanics and systems.

To implement the certification work, an engine for creating game applications - Unity3D was used. The C# programming language was used to write the code of the game components. To combine the various parts of the collective project, the Git version control system was used, in particular the GitHub service, which in turn simplified the development in the group. As an IDE for writing code and its modifications, Visual Code was used with appropriate extensions for the needs of development, because the specified code editor is fast enough, convenient to use and easily modifiable with the necessary components, which significantly speeds up the development process. VFX Graph was used to create visual effects (VFX) - a component of the Unity3D engine, which provides a deep and intuitive toolkit for implementing good particle simulations, etc. To implement

the part with the movement to the artificial intelligence of the enemies, the AI Navigation package was used to find paths and build routes for the enemies of the user character. In order to implement a deeper and more modifiable control system, a new API of the Unity3D engine was used, namely the Input System package. To create large complex systems, improve modifiability and maintainability along with code cleanliness, best practices were applied to implement the respective components in combination with the use of design patterns, etc. As graphic materials for creating the content, free creations from the Internet were taken, distributed under CC0 - the copyright rule, made by own hands or created with the help of artificial intelligence, which also falls under the mentioned property rights, etc. In addition to the created part of the game "The fate of the liar", a media activity was carried out to gather an interested audience for the future release of the current project and the following ones.

The result of the attestation work was the implementation of the game process of exploring dungeons in the fictional world of the collective game "The fate of the liar", which includes the creation of complex systems:

- random generation of levels;
- reproduction of combat mechanics, including elemental system, special skills of the main character and basic abilities, such as roll, simple strike, etc.;
- artificial intelligence of enemies, which is based on a number of rules and consists of different sets of skills, behaviors and personal parameters;
- research opportunities for gameplay diversity of randomly generated dungeons;
- related mechanics, additional services and support systems to improve the gaming experience and process, etc.

In addition to the created part of the game "The fate of the liar", a media activity was carried out to gather an interested audience for the future release of the current project and the following ones. Also, the developed software module was theoretically tested at the International Scientific Conference and practically demonstrated at the International Youth Forum exhibition. Also, the developed software module was theoretically tested at the International Scientific Conference and practically demonstrated at the International Youth Forum exhibition.

Я, Долгий Андрій Іванович, студент гр. ПЗПІ-20-4, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Ігровий програмний застосунок у жанрі RPG з елементами економічної стратегії і roguelike. Механіки генерації рівнів, механіки на рівні підземель», 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 ознайомлений 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 11
1 Аналіз предметної галузі 15
1.1 Аналіз ринку 15
1.2 Аналіз проблематики та визначення рішення 23
1.3 Постановка задачі 24
2 Формування вимог до ігрового застосунку 26
2.1 Постановка мети 26
2.2 Загальний опис 26
2.3 Загальні обмеження 28
2.4 Припущення та залежності 29
3 Архітектура та проєктування програмного модулю 31
3.1 UML проєктування ПЗ 31
3.2 Проєктування архітектури програмного модулю 32
3.3 Проєктування інтерфейсу користувача (UI/UX) 39
4 Опис прийнятих програмних рішень 47
4.1 Вибір інструментарію для програмної реалізації програмного модулю 47
4.2 Опис програмної реалізації модулю roguelike 47
4.3 Розроблені варіації ворогів 61
4.4 Розроблені ефекти стихійної системи 65
4.5 Балансування наповнення програмного модулю roguelike 68
5 Тестування програмного модулю 75
5.1 Розробка мапи думок тестування 75
5.2 Розробка тестових випадків 79
6 Впровадження програмного модулю 95
6.1 Наукове впровадження проєкту 95
6.2 Практичне впровадження проєкту 96
7 Медійна діяльність 97
7.1 Постанова мети медійної активності 97
7.2 Реклама проєкту 98

7.3 Аналіз активності у YouTube 99
7.4 Аналіз активності у Instagram 101
7.5 Аналіз активності у Telegram 103
7.6 Аналіз активності у Facebook 107
7.7 Аналіз активності у X (Twitter) 109
7.8 Підсумки медійної діяльності 110
Висновки 114
Перелік джерел посилання 116
Додаток А. Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ 120
Додаток Б. Слайди презентації 121
Додаток В. Геймдизайн-документ 130
Додаток Г. Тест-план 157
Додаток Д. Приклад програмного коду 171
Д.1 Користувацький атрибут для полів, що є об’єктами спадкоємцями «ScriptableObject» 171
Д.2 Користувацький атрибут для полів, що є «серіалізованими інтерфейсами» 173
Додаток Е. Балансування характеристик гравця та ворогів 176
Додаток Ж. Стаття для Міжнародної науково-практичної конференції 178
Додаток И. Презентація проєкту на виставці Міжнародного молодіжного форуму 188
Додаток К. Каталог проєктів виставки Міжнародного молодіжного форуму 193
Додаток Л. Диплом за призове місце на виставці Міжнародного молодіжного форуму 197

ВСТУП

Актуальність. На сьогоднішній день існує багато ігор, різних сегментів (від інді-розробок, до дорогих великих ігор), у яких основний ігровий процес, або його частина, основана на жанрі roguelike та притаманним ньому аспектам, а саме будові віртуального світу за ідеєю випадково генеруємих рівнів та їх наповнення у поєднанні із застосуванням RPG–систем для урізноманітнення основного досвіду гри. Слід відзначити, що проєкти подібного направлення виходять достатньо часто, через власну специфіку відносно дешевої та простої розробки, завдяки чому ринок перенасичується одноманітними іграми. До того ж звернемо увагу, на те, що більшість зі зразків подібних проєктів мають лише roguelike частину та єдину схему побудови ігрового досвіду у останній. Процес проходження рівнів складається з простого повторення низки дій:

- пройти декілька кімнат (або альтернатив);
- зустріти кімнату із новим зразком озброєння;
- змінити за бажанням поточну зброю;
- повторювати поки не пройдете усі кімнати та переможете босів або загините, аби почати знов.

До подібного алгоритму дій, іноді додається подача історії світу чи сюжет або гравець отримує можливість збирати певні ресурси для власного удосконалення тощо. Слід зазначити, що існують більш індивідуальні елементи, що відрізняються, але вони є вторинними серед ігрових механік. У поєднані цього факту із перенасиченням roguelike сегменту ринку, постає необхідність у свіжому нововведені у жанрі. Саме тому, нова інтерпретація основ жанру є необхідною для поштовху прогресу відповідної сфери та має неабиякий потенціал зайняти вільне місце на ринку.

Окрім загального ігрового досвіду в roguelike іграх, уваги потребує й внутрішнє наповнення рівнів та процесу їх проходження. Як зазначалось вище, гравець, проходячи випадково згенеровані рівні, має досить маленьке різноманіття у досвіді. В основному, гра іноді підкидує головному герою випадкові предмети, у вигляді іншої зброї чи активної або пасивної можливості, що загалом змінює лише

цифри шкоди, запас здоров’я тощо та візуальне оформлення при застосуванні зброї, при цьому нічого не змінюючи у основі бойової системи. Це означає, що загалом, ігровий процес не міняється у продовж всієї подорожі, а успішність кожного проходження досить сильно залежить від вдачі, та того, які предмети з’являться, в той самий час, коли внесок розуміння механік гравцем відходить на другий план. Отже оновлення потребує також бойова система та основні елементи проходження рівнів, у грі з жанром roguelike.

Таким чином актуальною постає потреба у новому бачені як самих випадково згенерованих рівнів та відповідних аспектів жанру roguelike, так і їх наповнення у ігро-механічному сенсі, тобто переосмислення потребують бойова система, дослідницькі можливості та пов’язані із цим механіки.

Отже метою даної кваліфікаційної роботи є розробка програмного модулю, який відтворює ігровий досвід для roguelike частини гри «The fate of the liar», що включатиме в себе механіки та системи для створення процесу дослідження випадково згенерованих рівнів, подолання ворогів і босів, у поєднанні з елементами виживання та пошуку коштовностей тощо.

Поставлена мета роботи досягається шляхом розв’язання обумовлених нею задач, а саме:

- аналіз предметної галузі розробки;
- формування вимог до програмного модулю;
- проєктування архітектури та будови зазначеного компоненту гри;
- розробка програмного модулю за встановленими вимогами та архітектурними рішеннями;
- тестування розробленого компоненту;
- впровадження розробки з наукової та практичної сторін;
- медійна активність для привернення уваги до продукту, що розроблюється.

Об’єктом проєктної розробки постає програмний модуль, що є частиною гри «The fate of the liar» та містить roguelike компонент, із генерацією рівнів, механіками дослідження підземель та пов’язаними системами і можливостями.

Предметом кваліфікаційної роботи постають складові програмного модулю, що обговорюється, а саме система випадкової генерації рівнів – підземель, бойова складова разом зі штучним інтелектом ворогів, механіки дослідження створених локацій у поєднанні із пошуком скарбів та допоміжні механіки та системи, що супроводжують гравця у підземеллях, із метою покращити його ігровий досвід.

Новизною roguelike компоненту гри виступає переосмислення ваги системи із генерацією підземель, ставлячи її на один рівень з другою половиною проєкту – розбудовою власного поселення. Такі зміни дарують нові відчуття від гри, на тлі сучасних ігор із жанром roguelike. Разом із переосмисленням концепції випадкових підземель, у кваліфікаційній роботі є перероблену бойова система та пов’язаний з нею алгоритм дій гравця. Натомість випадковій генерації зброї, разом із активними та пасивними вміннями, що видаються гравцеві у інших іграх подібного жанру, що в свою чергу змінюють лише певні параметри персонажу та візуальне супроводження, через що, досвід від продукту повністю базується на вдачі; змінена бойова система робить гру більш базованою на тактиці та вміннях користувача, ніж на випадковостях. У новій бойовій системі, основним інструментом є стихійна система та RPG механіки, що створюють для гравця унікальний досвід під час дослідження підземель та водночас дарують свободу дій та можливість варіативної поведінки у сутичках із ворогами.

Теоретичне значення кваліфікаційної роботи полягає у здобутті знань та навичок, що необхідні для реалізації програмного модулю, що є частиною гри, та містить у собі механіки та системи притаманні roguelike жанру. Також, при створенні зазначеного компоненту ігрового застосунку, отримуються навички та вміння з аналізу відповідної галузі та наявних пропозицій ринку та виокремлення з них інформації, що буде підґрунтям та основою для подальшої розробки програмного модулю із додаванням актуальних та новітніх рішень до перевірених компонентів.

Практичним значенням кваліфікаційної роботи постає можливість подальшого впровадження розробленого ігрового застосунку (частиною якого є програмний модуль, що розроблюється) у комерційний простір, а саме публікація

у магазині ігор, на кшталт Steam, Epic Games, GOG або платформа Itch.io тощо. Завдяки подібному впровадженню, за розроблений продукт можна отримати, як грошову винагороду для розробників, так і ресурси та ґрунт для подальшої підтримки та вдосконалення гри або створення нового проєкту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз ринку

1.1.1 Аналіз жанру roguelike

1.1.1.1 Історії становлення жанру

Історія жанру roguelike, походить з кінця XX століття, а саме 1980-х років, коли у 1980 року, для UNIX подібних операційних систем, була розроблена гра «Rogue» [1], від якої й пішла назва усього жанру. Дослівно roguelike з англійської можна перекласти як «rogue подібна». Слід відмітити, що до гри «Rogue», виходили й інші проєкти, що можна віднести до roguelike жанру, такі як «Beneath Apple Manor» [2], «Sword of Fargoal» [3] або «Star Trek»[4], але саме зазначена гра є основою усього направлення.

Ігровий процес «Rogue», був заснований на концепції проходження згенерованого рівня з одним життям, через що після смерті генерувався новий рівень та все починалось з початку. Такий підхід до створення ігрового досвіду був досить успішним, завдяки великій реграбельності та певної унікальності кожного проходження.

Завдяки отриманій популярності жанру, стали з’являтися нові ігри послідовники, які вносили щось нове, однак зберігаючи основи ігор попередників, а саме парадигми із перманентною смертю та випадковій генерації рівнів. Слід зазначити, що перші ігри у жанрі roguelike, для бойової системи та супутніх механік, адаптовували такі системи настільних ігрових ігор, як GURPS (Generic Universal Role Playing System) [5], DnD (Dungeons&Dragons) [6] тощо. Виділяючи найбільш відомі та значущі ігри ранньої епохи жанру roguelike, можна побудувати певне дерево «успадкування», яке демонструє процес становлення жанру, починаючи з гри «Rogue» (див. рис. 1.1).

Популяризація ігор жанру roguelike, призвела до того, що у 2000-х роках, з’явилися піджанри, roguelike-like та roguelite, які відрізнялися від своєї основи тим, що містили лише певні аспекти, уникаючи решти, а в свою чергу відрізнялися між собою тим, що перший жанр все ще будується на ідеї штурму підземелля, а другий – представляє ігри, в якому є певна кількість аспектів властивим основоположному

жанру roguelike. Однак слід зазначити, що питання доречності виділяти зазначені піджанри є дискусійним, отже у цій роботі буде зазначатися саме жанр roguelike, як основний для них, та відзначатися, що застосовуються саме його елементи чи аспекти, а не підтипи тощо.

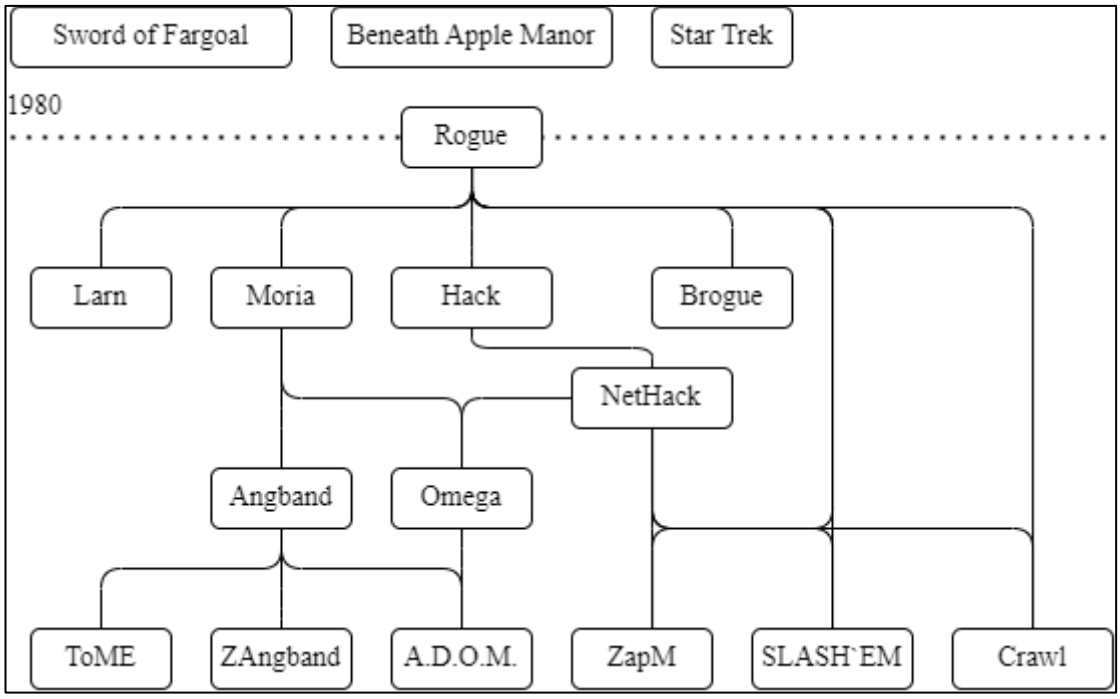


Рисунок 1.1 – Дерево «успадкування» ігор ранньої епохи жанру roguelike (розроблено автором)

Зазначений підхід, із виділенням лише частини аспектів, властивим roguelike жанру, став використовуватися у більших іграх, рівня ААА. Так ігри серії «Diablo» та «Bloodborne» отримали механіки генерації випадкових підземель, що створювало унікальний ігровий досвід, несхожий на той, що був у інших. Також, Demon Souls, включно з наступними іграми Хідетако Міядзакі випущеними компанією FromSoftware, взяли аспект перманентної смерті, із втратою усього прогресу, та переробили його на механіку повернення до найближчого безпечного місця, де зберігся користувач, із втратою зароблених ресурсів. Такий підхід до загибелі гравця, породив певного роду піджанр soulslike, якому властиві аспекти «Demon Souls» (однак слід зазначити, що зазначений підтип є дискусійною темою, та його існування не є загальноприйнятим). Також, аспект із випадковою генерацією підземель, був запозичений Маркусом Персоном та застосований у грі

«Minecraft». У подальшому подібним шляхом він був використаний у «Noita» та «Terraria» й подібних пісочницях.

Однак зазначені вище ігри, та застосовані у них аспекти roguelike жанру, не є собою розвитком жанру чи напрямку. Прямим продовжувачем ранніх roguelike ігор, вважається «Dwarf Fortress» [7, 8], що покращує кожен з початкових аспектів «Rogue», додає нові унікальні механіки та повністю повторює парадигму гри, із генерацією цілого світу та його знищення після смерті гравця тощо. Проте, слід зазначити, що через таке покращення початкових аспектів та їх розширення, є дискусійним питання, чи це розвиток жанру, чи черговий відхід від бази roguelike ігор.

Після «Dwarf Fortress», значною подією для жанру, є вихід у 2011 році «The Binding of Isaac» [9], гри що привнесла суттєві зміни до формули ігор відповідного жанру, додавши до випадкових підземель з кімнат, свободу переміщення, бою, та механіку із варіативної зброї, що тільки підсилює основний аспект roguelike жанру.

Після цього виходили подібні ігри: «Risk of Rain», «Hades», «Dead Cells», «Cult of the Lamb» тощо, які змінювали певні аспекти, привносили деякі нововведення та покращення до існуючих елементів, але не змінюючи основну формулу ігрового процесу, що призводить до перенасичення ринку одноманітними продуктами, які відмінні, здебільшого, лише обгорткою.

Слід зазначити, що існують рідкісні відхилення, від поточної будови ігор у roguelike жанрі, а саме проєкти на кшталт «Inscryption», «Balatro» тощо, однак вони містять значне додавання сторонніх жанрів до roguelike елементів, що можна охарактеризувати як гру, лише з елементами зазначеного типу. Отже, для огляду розвитку саме основного жанру roguelike, вказаними та подібними іграми можна знехтувати.

Враховуючи, що «The Binding of Isaac» вийшов більше ніж за 10 років ніж на момент написання поточної роботи, можна відмітити певну стагнацію жанру та його інтерпретацій на ринку ігрових застосунків.

1.1.1.2 Оприявлення аспектів жанру

Відповідно до гри «Rogue», та її наслідувачів, що є засновником жанру roguelike, можна виокремити основні ознаки:

- випадково згенеровані рівні та наповнення у них;
- перманентна смерть, що означає втрату усього прогресу, та наступна спроба з самого початку;
- свобода дій, наскільки це позволяють умови гри та випадково створеного світу.

Також можна виділити додаткові ознаки, що є набутими або не обов'язковими для жанру roguelike, а саме:

- дослідження ігрового світу;
- достатнє різноманіття елементів оточення, ворогів, зброї тощо;
- цікаві, неочевидні механіки та деталі.

До зазначених ознак, що визначаються представниками жанру, починаючи з самої першої гри, слід додати також перелік аспектів, що є супроводжуючими або являються наслідком одного чи сукупності елементів:

- достатньо висока складність у порівнянні із іншими продуктами на ринку;
- наявність бойової та дослідницьких систем та механік;
- вороги подібні до головного героя: мають власні характеристики, штучний інтелект, підкорюються тим же самим законам гри, що й персонаж, мають зброю та вміють захищатись й нападати, тощо.

Окрім описаних вище ознак, що виходять із наочного розбору та аналізу існуючих представників жанру roguelike, слід також звернути увагу, на «берлінську інтерпретацію» [10], яка є результатом обговорення теми канонів жанру, на Міжнародній конференції розробників roguelike ігор, від 16 вересня 2008 року. Так, «берлінська інтерпретація» визначає, що саме є roguelike жанр та його представники. Вона наголошує на тому, що це не сукупність ігор, що нагадують «Rogue», а саме жанр, і не що інше. Далі, визначаються первинні, вторинні та третинні, за важливістю, ознаки roguelike жанру:

а) первинні ознаки:

1) ігровий світ та його наповнення генеруються процедурно, заново з кожною новою грою;
2) персонаж має заплатити за помилки гравця, ціною життя або альтернативи, при цьому продовження гри не рекомендується і надається в основному для продовження перерваної гри (не через смерть чи подібний випадок);
3) усі сутності в ігровому вимірі стають у чергу нескінченного очікування поки не отримують свою власну та незалежну від інших чергу виконувати дію;
4) гравець може отримати доступ до усіх дій гри з будь-якого місця у ній, та не має бути ніяких штучних обмежень тощо щодо доступності дій в тій чи іншій ситуації у грі (хоч у боях, хоч поза ними);
5) прогрес проходження не переходить у лінійну прогресію, є повна свобода дій, що і коли робити;
6) гравець самостійно повинен дослідити та з'ясувати природу та принцип використання предметів у створеному світі;

б) вторинні ознаки:
1) гравець керує єдиним персонажем – головним героєм;
2) існує достатнє різноманіття типів монстрів та предметів, щоб дослідити його варто (понад 24 різновиди монстрів й предметів є гарним показником);
3) використання предметів є нетривіальне, гравець може робити певні речі, які можуть виглядати неочевидними для самої природи предмета;

в) третинні ознаки:
1) гра стає важчою досить швидко, через що, гравець навряд чи виграє, доки не набере достатньо досвіду;
2) природа ворогів подібна до персонажа, вони можуть мати озброєння, особисті параметри, штучний інтелект і підпорядковуються тим же самим правилам створеного світу, що й гравець;

3) гравець взаємодіє із ігровим оточенням через інтерфейс користувача базуючись на символах персонажів, що в свою чергу представляють собою артефакти інтерфейсу гри та сутності у грі;
4) ігровий процес передбачає вбивство ворогів, для того, аби стати могутнішим, і збір скарбів, щоб купити більш краще озброєння та повторити цикл.

Як можна побачити, більшість аспектів співпадають, однак, серед ознак, що наведені у «берлінській інтерпретації», присутні застарілі та дуже обмежуючі правила жанру roguelike. У випадку з ознаками, що вже не актуальні, такі як символьне представлення гри чи покрокова дія усього додатку, можна просто знехтувати, оскільки у реаліях сучасного ринку більшість не буде грати у подібне. Із аспектами, що вибудовують консервативні обмеження щодо певних моментів гри, таких як відсутність позациклового наповнення чи відсутність допомоги гравцю зі сторони розробника тощо, є причиною стагнації ігор зазначеного жанру та причиною відсутності різноманіття у користувацькому досвіді. Зазначена «берлінська інтерпретація», хоч і є загальноприйнятою, проте вона є застарілою, як з точки зору вимог, так і фактично за часовими рамками, оскільки була прийнята, ще у 2008 році. Тому при плануванні та реалізації програмного модулю, за автором залишається право адаптувати та ігнорувати певні описані ознаки жанру та створювати механіки з огляду на поточні або наявні потреби.

1.1.2 Аналіз існуючих аналогів

Як зазначалося у минулому підрозділі 1.1.1, наразі актуальними проєктами, у жанрі roguelike, постає низка проєктів: «Hades», «Dead Cells», «The Binding of Isaac», Cult of the Lamb», тощо. У свою чергу, «Balatro», «Inscryption» та подібні roguelike ігри не розглядаються як аналог, через значне відхилення від основи жанру. Відзначимо, що за рейтингом користувачів, найбільш оціненими постають «Hades», «The Binding of Isaac» та «Dead Cells» [11], отже у якості конкуруючих аналогів є доцільним розглянути саме ці три проєкти, що є найбільш оціненими користувачами. Проаналізуємо визначені аналоги для оприявлення слабких та

сильних сторін для концептуалізації рішення, щодо програмного модулю, який розроблюється у рамках кваліфікаційної роботи.

«Hades» [12] - гра у жанрі roguelike, що містить у собі як частину із випадково згенерованими підземеллями, так і загальний лінійний прогрес сюжету та прокачки озброєння. Містить бойову систему, що не будується на покроковій парадигмі, а продовжує нововведення «The Binding of Isaac» із боями у реальному часі.

Слід відзначити гарне поєднання механіки загального сюжету із позапідземельною локацією, у якій гравець може займатися іншими справами, що урізноманітнює ігровий досвід. Також бойова система має гарні рішення з точки зору варіативності предметів та візуально частини. Однак, у грі, через основоположну роль підземелля, виникає момент, коли сюжет встає на місці, через випадковість світу та спорядження, що так само генерується процедурно й перетворює забіги у гру з удачею, оскільки на великих складностях, які присутні у грі, багато чого вирішує саме озброєння гравця, а не його навички.

«The Binding of Isaac» [9] – основоположник сучасних ігор у жанрі roguelike, який не містить у собі, на відмінну від «Hades», щось окрім випадкових підземель, однак має таку саму, вільну від покроковісті бойову систему. Так само як і у попереднього аналога, генерація світу відбувається покімнатно, тобто рівні представляють з себе мережу з кімнат та дверей між ними.

Відзначимо, що гра має дуже глибоку систему із «синергіями» предметів, що урізноманітнюють бої. Також гарно себе показує система із «секретами», у грі є певні предмети та послідовності дій, що призводять до неочевидних результатів, таких як отримання нових, унікальних предметів або потрапляння на особливі версії рівнів тощо. Однак на певному етапі, гра стає повторюваною та одноманітною, через присутність випадковості у генерації предметів, через що багато спроб проходжень завершуються значно раніше ніж могли, саме завдяки процедурному озброєні персонажу, через що, вплив вдачі пересилює вправність гравця, що може зіпсувати враження від проєкту. Також, у грі немає ніякого ні альтернативного контенту для підземель, ні додаткових механік на самих рівнях,

що значно сильно фокусує гравця на єдиній активності – бійці та швидко його втомлює одноманіттям.

«Dead Cells» [13] – гра у жанрі roguelike, але із більш унікальною генерацією світу. У зазначеному проєкті, процедурне створення мапи, полягає не у розміщенні кімнат та побудови між ними шляхів із дверима, як у попередників, а у створенні цілих, єдиних локацій, із елементами жанру metroidvania. «Dead Cells» так само має активну бойову систему, як й попередні аналоги.

Слід відзначити, що підхід до генерації є певною мірою нововведенням у жанр, однак він також має свої обмеження, через що, через кілька проходжень, патерни будови кожної локації запам’ятовуються та становляться одноманітними. Також відзначимо, що у «Dead Cells» додано позапідземельне наповнення, а саме кожнодобові виклики, колекціонування різновидів озброєння та косметики, кімнати тренування тощо. Однак це так само має ті ж самі проблеми, що й в аналогів, а саме одноманітність озброєння на початку проходження, через брак розблокованих предметів, а потім предметів стає забагато та починається проблема вдалої генерації потрібного озброєння. Також із кожнодобовими викликами та ж сама ситуація із загальною генерацію світу, як і з звичайними локаціями – незмінність загального патерну.

1.1.3 Цільова аудиторія

Ігри roguelike жанру можуть бути будь-якої статі, оскільки зазначений тип проєктів не містить у собі жодних гендерних прив’язок, тощо, отже гру може захотіти спробувати як чоловік, так і жінки.

Орієнтовний вік цільової аудиторії полягає у діапазоні з 18 до 35 років. Такі рамки обумовленні тим, що roguelike гра матиме певні спірні теми, такі як вбивство ворогів, ношення героєм зброї, небезпечні дії тощо, отже будуть певні вікові обмеження до 18 років. У свою чергу, верхній поріг – 35 років, обумовлений зниженням зацікавленості іграми у більш старшої аудиторії. Однак сліз зауважити, що проєктом можуть зацікавитися люди і поза межами вказаного вікового діапазону, але їх буде менше ніж користувачів від 18 до 35 років.

Особливості жанру roguelike, неодмінно приверне користувачів, такі як реграбельність та перепроходження, цікава бойова система із можливістю певних випробувань щодо стилю ведення бою, азартне ставлення до кожної невдалої спроби, що змушує спробувати ще раз тощо.

1.2 Аналіз проблематики та визначення рішення

Як зазначалось вище, у попередніх підпункті 1.1.1.1, на поточний момент, у сегменті ігор roguelike жанру відбувається стагнація, оскільки суттєвих змін до формули ігрового досліду основи roguelike не вносилося суттєвих змін, а саме, до бойової системи та відношення до сенсу відповідної частини проєкту зокрема.

Так само, у попередньому підпункті 1.1.2, було розглянуто проблематику наявних на ринку та актуальних на поточний момент аналогів – ігор roguelike жанру. До проблем можна віднести одноманітність підходу до побудови бойової системи, що не змінюється від гри до гри зовсім, єдине, що їх різнить, це обгортка генеруємих речей та місцями додаткових до них механік, на кшталт «синергій» предметів з «The Binding of Isaac» тощо.

Зазначені проблему призводить до того, що гравцю, що грав у один подібний проєкт, майже нічого шукати у іграх цього жанру, оскільки він бачив більшу частину механік та систем, що повторюються майже без змін у кожному новому проєкту. Так нові ігри можуть затягнути, в основному, лише яскравим графічним виконанням або будь яким подібним способом, але після невеликого часу проведеному у грі, досвідчений користувач втрачає інтерес до проєкт, через його вторинність.

Через це, актуальним стає питання переосмислення наявної формули жанру, причому так, аби він залишався собою, без зайвої модифікації основної формули та відповідних аспектів roguelike. Привнесення новизни у жанр, грамотно апелюючи до встановлених головних принципів roguelike направлення, може створити неабиякий поштовх до подальшої еволюції жанру та відповідних проєктів зокрема.

Так розглядаючи питання одноманітності ваги roguelike частини у іграх, слід відзначити, що у всіх випадках, що розглядалися, згенеровані підземелля є чи не

єдиною активністю у проєкті, як у «The Binding of Isaac». Окремі випадки із додатковими механіками поза межами згенерованих рівнів, мають дуже обмежений функціонал, та втілюють єдину ціль, відображати прогрес гравця у «закритті» усього наявного контенту як у випадку з «Dead Cells», або додатково ще містять невеликі елементи сюжету, як у «Hades». Виходячи з описаного вище, є очевидним рішенням, розробити додаткову частину до roguelike модулю. Слід зауважити, що обидва компоненти єдиної гри мають бути рівноправними та взаємопов'язаними модулями проєкту, що означатиме, що по одинці вони не зможуть існувати.

У той самий час, розглядаючи питання вторинності та повторюваності бойових систем, що складаються з простого алгоритму дій: пройди декілька секцій із ворогами тощо, знайди новий предмет озброєння, такий як зброя, активне чи пасивне уміння (може залежати від концепції кожної окремої гри), після чого замінити ним існуючий чи просто пройти повз, повторювати до завершення підземелля або смерті. У таких системах, у певний момент часу, вдача, через випадкову генерацію усього на рівні, переважує внесок вмінь гравця, та знижає процент успішних заходів до підземель, не зважаючи на досвід та навички користувача. Отже, рішенням, яке переосмислить існуючу концепцію бойової системи у roguelike іграх, постане позбавлення відповідного модулю від фактору вдачі, прибравши елементи, що відповідали за генерацію випадкового озброєння та механізм подальшої заміни ним існуючого. Натомість гравець отримає оновлену бойову систему, із упором у тактику гри, та запланованого походу до підземелля, що означатиме, що гравець має планувати власноруч свій захід до roguelike рівня. При цьому, кожен елемент оновленої системи має бути самостійним, аби гравець міг сам визначати, як само йому грати.

1.3 Постановка задачі

У ході виконання кваліфікаційної роботи, необхідно реалізувати програмний модуль roguelike частини гри «The fate of the liar». Зазначений компонент ігрового застосунку, має відповідати основним аспектам жанру, не виходячи за його межі,

за винятком обумовлених елементів. Програмний модуль має містити реалізацію основних механік та систем, що необхідні для відповідності до roguelike жанру.

Так у кваліфікаційній роботі має бути: створені механіки процедурної генерації рівнів, впроваджена система управління персонажем та механіки дослідження кімнат у підземеллі, бойова система, штучний інтелект ворогів.

Також слід зазначити, що згідно визначеному рішенні актуальних проблем жанру roguelike (підрозділ 1.2), програмний модуль, що розробляється, має бути пов’язаний із іншою частиною гри «The fate of the liar» - розбудови власного поселення, та мати рівну вагу у ігромеханічному сенсі. У той же самий час, бойова система так само має бути перероблена, та позбавлена випадкового отримання спорядження із заміною замість актуального тощо. Натомість, гравець сам визначає, що він візьму в похід до підземелля, які навички обере. Також, персонаж отримає стихійну систему, яка завдяки взаємодії елементів, створюватиме тактичні задачі перед гравцем, завдяки тому, що вороги у підземеллі схильні до ефектів стихій тощо. Слід також зауважити, що зазначені елементи оновленої бойової системи є опціональними, тож гравець сам обирає спосіб проходження.

Описані механіки та системи програмного модулю roguelike частини мають бути реалізовані, щонайменше на базовому рівні, із підтримкою масштабованості наповнення та спроможності до подальших вдосконалень.

2 ФОРМУВАННЯ ВИМОГ ДО ІГРОВОГО ЗАСТОСУНКУ

2.1 Постановка мети

Метою виконання кваліфікаційної роботи є реалізація програмного модулю roguelike частини проєкту «The fate of the liar» у рушії для створення ігрових застосунків Unity3D. Зазначений компонент загального проєкту має вирішувати проблему стагнації та одноманітності сучасних ігор у roguelike жанрі.

Зазначений програмний модуль включатиме в себе механіки та системи задля відтворення процесу дослідження процедурно згенерованих підземель (включно із самою процедурною генерацією), подолання ворогів й босів (включно із штучним інтелектом для монстрів), у поєднанні з механіками виживання та пошуку скарбів у підземеллі тощо.

Слід відмітити, що бойова система, має бути перероблена, під потреби ринку, а саме із заміною процедурно генеруємої механіки появи нового озброєння на тактичну підготовку перед походом до підземелля, включно із стихійною системою гаджетів.

Також варто звернути увагу на те, що програмний модуль, що розроблюється, є рівноправною частиною «The fate of the liar», через що, потребує поєднання із компонентом проєкту, що включає в собі розбудову власного поселення.

2.2 Загальний опис

У програмному модулі roguelike частини, гравця зустрічає система завантаження рівня, яка відповідає за те, аби усі інші системи завантажились успішно, у правильній послідовності, а користувач, у цей же час, бачив заготовлене меню, а не рівень, об’єкти якого з’являються з повітря.

Під час завантаження, слід виділити систему генерації рівнів, яка процедурно створює мережу кімнат, що пов’язані між собою дверима. Кожна кімната має власний тип (із ворогами, пуста, зі скарбами, босом, рідкісними ресурсами, з екстреним виходом тощо). Кімнати так само наповнюються під час процесу генерації рівня.

Після завершення завантаження, гравець отримує доступ до системи переміщення, елементів простої бойової системи, системи спеціальних вмінь та стихійної системи із колесом вибору гаджетів.

Слід розглянути стихійну системи детальніше. Передбачено чотири елементарні гаджети, а саме вогняний, електричний, водяний та льодовий. Кожен з них має унікальну взаємодію один з іншим, ворогами тощо. Так, після застосування, вогонь спалахує та підпалює усе, до чого може дотягнутися та може розтопити лід або випарити воду. Електричний гаджет створює, при активації, поле електричних блискавок, що наносить шкоду всім, хто потрапив у поле, та може розширити зону своєї дії, взаємодіяючи із водою. Водяний застосунок не випаровується сам по собі, проте легко замерзає, проводить блискавки чи навпаки тушить вогонь, зникаючи після недовгої взаємодії із іншими гаджетами. Льодовий гаджет може потушити вогонь, при цьому створивши водяну калюжу або навпаки зробити з калюжі рідини льодову перешкоду. Усі реакції мають коректно спрацьовувати та впливати на ворогів тощо.

Під час переміщення по згенерованому рівні, гравець зустрічає ворогів та босів, які використовують систему штучного інтелекту. Зазначена система має підтримувати легку модифікуємість кожного з елементів поведінки та мати змогу підтримувати декілька можливих атак тощо.

Також, під час дослідження, гравцеві мають траплятися кімнати із скарбами та жилами рідкісних руд, добич з яких має бути, певною мірою, випадковою та підходящою до рівня персонажу. Зібрані ресурси повинні потрапляти до тимчасового інвентарю та відправлятись до постійного загального сховища, при успішному завершені рівня. Також слід відзначити, що з босів та ворогів, має існувати ймовірність отримувати певні ресурси, скарби тощо.

Після перемоги над босом, у кімнаті з’явиться портал, через який можна повернутися до поселення, захопивши із собою усі здобуті коштовності, або продовжити досліджувати процедурно згенерований рівень, повернувшись до виходу пізніше. Також має існувати певна ймовірність на появу виходу у вільній кімнаті на рівні.

2.3 Загальні обмеження

Програмний модуль , що розроблюється у рамках кваліфікаційної роботи, має відповідати певним обмеженням:

- компонент має бути працездатним, як частина повноцінної гри;
- під час завантаження рівня, система відповідає за те, щоб усі інші компоненти завантажились успішно, у відповідній послідовності тощо;
- користувач, під час завантаження підземелля, бачить відповідне вікно;
- система генерації рівнів процедурно відтворює мережу з кімнат, які пов’язані між собою переходами з дверима;
- кожна кімната отримує власний тип (пуста, із ворогами або босом, зі скарбами, рідкісними ресурсами, з екстреним виходом тощо);
- кімнати, наповнюються під час процесу генерації підземелля;
- після повного завантаження рівня, гравець отримує можливість переміщуватись, битися тощо;
- стихійна система передбачає чотири елементарні гаджети: вогняний, електричний, водяний та льодовий;
- кожен зі стихійних девайсів має унікальну взаємодію один з одним, із ворогами тощо;
- вороги та боси використовують систему штучного інтелекту;
- штучний інтелект має підтримувати легку модифікуємість кожного з етапу поведінки та мати можливість вміти декілька можливих атак одночасно;
- зібрані ресурси повинні потрапляти до тимчасового інвентарю та відправлятись до постійного загального сховища, при успішному завершені рівня;
- після поразки боса, у його кімнаті з’являється портал, через який можна повернутися до поселення або продовжити досліджувати процедурно згенероване підземелля, повернувшись до нього пізніше;
- програмний компонент може працювати незалежно від проблем у окремо взятій підсистемі тощо;

Оприявлення визначених вище обмежень, є основою для подальшого проєктування та розробки програмного модулю roguelike частини.

2.4 Припущення та залежності

При розробці, слід розглянити низку припущень та залежностей:

а) система завантаження рівня:
   1) припускається, що система завантаження рівня забезпечить успішне завантаження всіх інших систем у правильній послідовності;
   2) ця система має бути синхронізована з відображенням заготовленого меню, поки рівень не буде повністю завантажений;

б) система генерації рівнів:
   1) припускається, що система генерації рівнів процедурно створить мережу кімнат, що зв'язані дверима, з різними типами комірок;
   2) кімнати будуть наповнюватись відповідними об'єктами під час процесу генерації рівня;

в) системи ігрового процесу:
   1) після завершення завантаження, гравець матиме доступ до систем переміщення, бойової системи, спеціальних вмінь, стихійної системи з гаджетами та інтерфейсу;
   2) реалізація стихійної системи має враховувати взаємодію чотирьох елементарних гаджетів, кожен з яких має унікальні властивості та реакції;

г) штучний інтелект ворогів:
   1) вороги та боси будуть використовувати систему штучного інтелекту з можливістю легкої модифікації їхньої поведінки та реалізації черги з декількох атак;

д) ресурси та скарби:
   1) під час дослідження рівнів, гравець зустріне кімнати із скарбами та рідкісними рудами, які можуть бути зібрані та використані;

2) зібрані ресурси мають бути збережені до тимчасового інвентарю та потім відправлені до загального сховища після успішного завершення рівня;

е) поведінка ворогів та нагороди:

1) існує ймовірність отримання певних ресурсів та скарбів від ворогів та босів;
2) після перемоги над босом, гравцеві буде доступний портал для повернення до поселення зі здобутими нагородами або продовження дослідження рівнів.

Ці припущення та залежності слід враховувати при розробці програмного модуля roguelike частини проєкту, особливо щодо взаємодії різних систем та обробки різних сценаріїв гри.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО МОДУЛЮ

3.1 UML проєктування ПЗ

Програмний модуль roguelike частини, що розроблюється у рамках кваліфікаційної роботи, є окремим компонентом проєкту «The fate of the liar», тому при побудові Use Case діаграми (див. рис. 3.1) для гравця, розглядатиметься лише зазначений модуль.

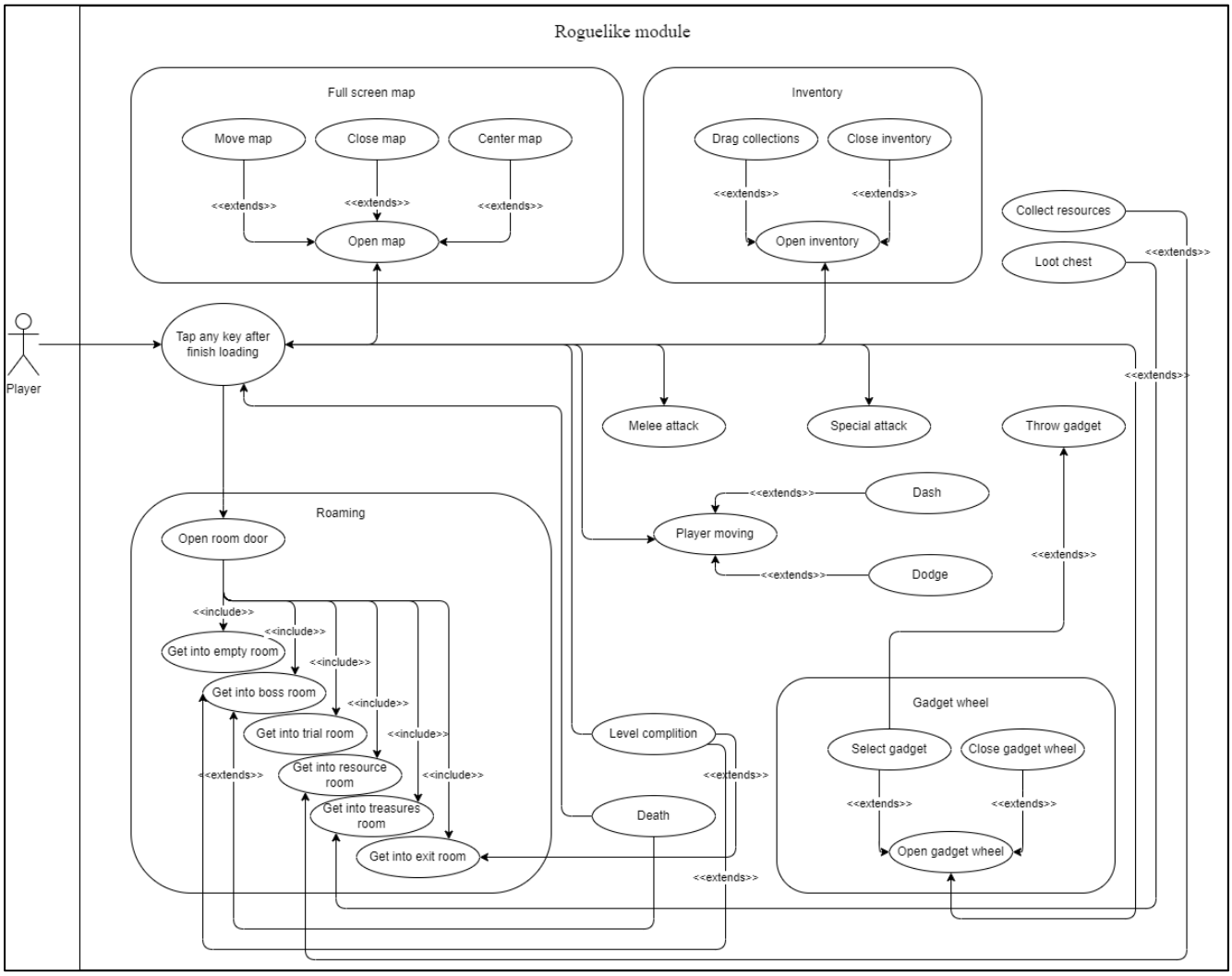


Рисунок 3.1 - Use Case діаграма можливостей гравця (розроблено автором)

Потенційні дії користувача було визначено й оприявлено згідно розділу 2. На діаграмі відображено, що гравець починає свою взаємодію з програмним модулем roguelike частини гри із завантаження рівня та натискання будь-якої кнопки, для того, аби потрапити на сцену. Можна побачити декілька груп дій, а саме мапу, інвентар, колесо гаджетів, та переміщення по рівню. Усі зазначені групи мають

додаткові дії, такі як потрапляння у один з доступних типів кімнат підземелля або виконання операції над інтерфейсом тощо. Окремо знаходять дії простого переміщення та додаткові - біг та перекат. Окремо також виділені три дії атаки, а саме проста атака, спеціальна атака та кидок гаджету. На останок є два осередка дій, а саме дії, що пов’язані із пошуком скарбів та ресурсів, та ті, що призводять до завершення поточного проходження – смерть та подія успішного завершення гри.

Спроєктувавши повний обсяг дій користувача (у середині модулю, що розробляється) та відношення між ними, стане значно простіше проєктувати наступні елементи системи. Також слід відзначити, що усі подальші рішення будуть спиратися на створену діаграму, що узагальнить вимоги та бачення даного компоненту проєкту.

3.2 Проєктування архітектури програмного модулю

3.2.1 Загальний ігровий цикл

Починати проєктування слід з загального бачення проєкту. Тож відповідно необхідно відобразити загальну схему роботи програмного модуля з roguelike частини (див. рис. 3.2).

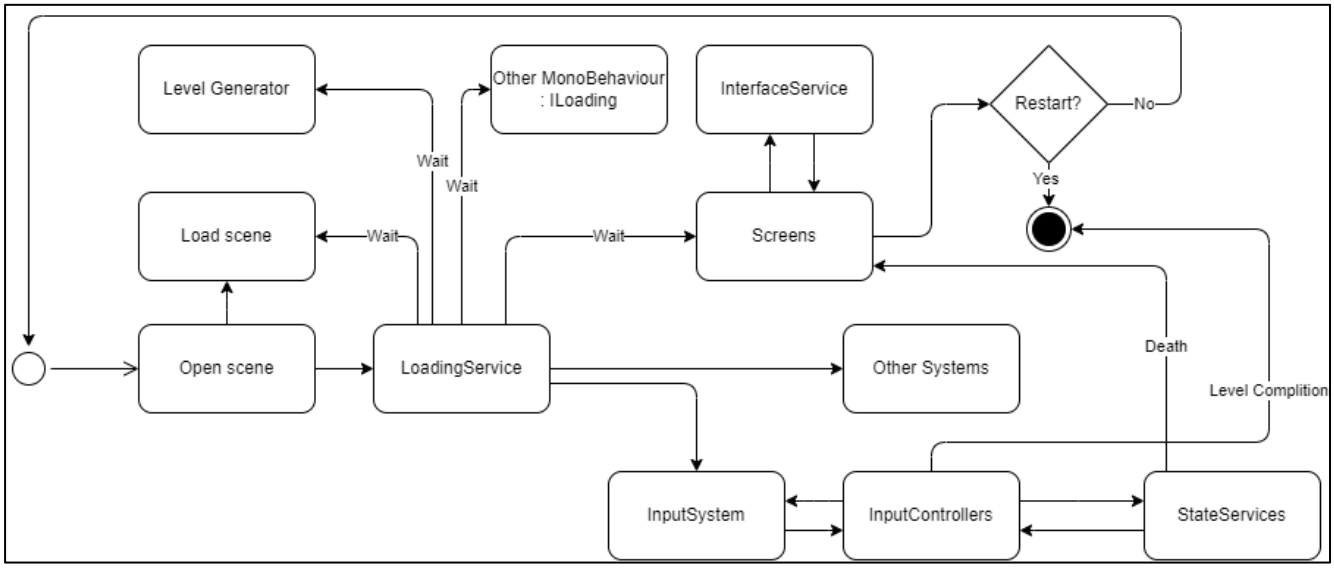


Рисунок 3.2 – Зображення схеми роботи загального ігрового циклу roguelike компоненту (розроблено автором)

На продемонстрованій схемі, зображена загальна схема ігрового циклу, що звертає увагу на початок завантаження сцени, як LoadingService блокує подальший прогрес, допоки не завантажить до кінця саму сцену, не згенерується рівень, не проініціалізуються усі інтерфейси та інші, менш значущі елементи, що успадкували MonoBehaviour та реалізують інтерфейс ILoading. Після повного завантаження усіх необхідних сервісів та систем починається штатна робота. На цьому етапі більшість зв’язків повністю прибираються, задля акцентування уваги на системі вводу, яка комунікує із відповідними контролерами, що у свою чергу відстежують та керують станами гри. Якщо стан змінився на відповідний до смерті гравця визивається відповідне вікно та у гравця питають про необхідність повтору спроби, після чого гра або завершається, або починається цикл заново. Так само, якщо гравець зайшов до вихідного порталу (через контролер введення), то цикл одразу завершується, оскільки немає причини для повтору, після чого гра переходить до наступної сцени та іншого ігрового циклу тощо.

3.2.2 Штучний інтелект ворогів

Однією з найважливіших ознак жанру roguelike, постають вороги, а саме головне їх штучний інтелект. Тож є важливим приділити цій темі достатньої уваги. На зображенні (див. рис. 3.3) продемонстрований алгоритм роботи штучного інтелекту.

Основою системи штучного інтелекту ворогів полягає у трьох інтерфейсах (IAttackBehaviour, IChaseBehaviour, IPatrolingBehaviour), які відповідають певним діям, а саме патрулювання, переслідування та атакуюча здібність. Потім зазначені інтерфейси реалізуються у абстрактний однойменних класах, після чого, їх можна використовувати для наслідування власних класів, тим самим створюючи нову унікальну поведінку для певної дії, що після, може бути додана до об’єкта ворога, після чого EnemyController, на моменті виклику метода Start(), збирає додані варіації поведінки. Слід відзначити, що EnemyController підтримую по одному поведінковому компоненту переслідування та патрулювання, одна може мати безліч атакуючих елементів.

Після ініціалізації контролеру, він переходить до циклічного методу Update(), у якому перевіряються три ситуації:

- ворог не бачить та не може атакувати гравця, тоді викликається поведінка патрулювання місцевості;
- якщо минула ситуація не була дійсною, то перевіряється, чи бачить монстр гравця, але не може його дістати власною атакою, якщо так, то він виконує поведінкову дію – переслідування, аби дістати гравця;
- останнім випадком є спроба дістати головного персонажа, якщо це вдається, викликається атака поточного поведінкового компоненту, після чого у будь-якому випадку цикл починається знов.

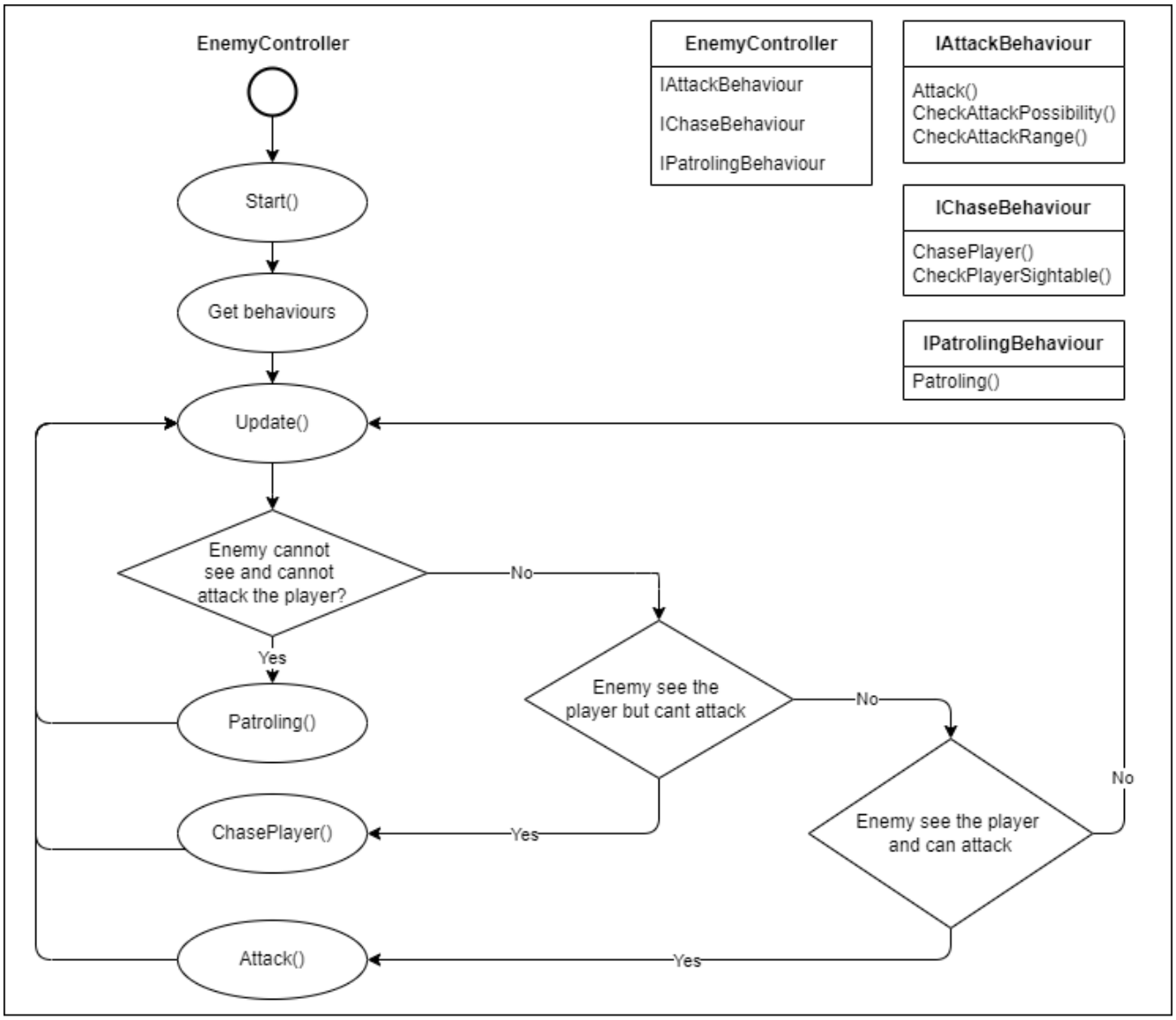


Рисунок 3.3 – Алгоритм роботи штучного інтелекту ворогів у програмному модулі roguelike (розроблено автором)

Слід відзначити, що подібна розроблена система є досить масштабованою, та дуже легко модифікуємою, через наявність абстракцій та спільних похідних інтерфейсів тощо. Відзначимо, що для пошуку шляхів, використовується офіційний компонент рушія Unity3D, а саме AI Navigation [14].

3.2.3 Генератор рівня

Ще одним, не менш важним аспектом жанру roguelike є випадкова генерація рівнів та їх наповнення. Тому є важливим момент із детальним аналізом метода генерації [15, 16, 17], що буде використано (див. рис. 3.4)

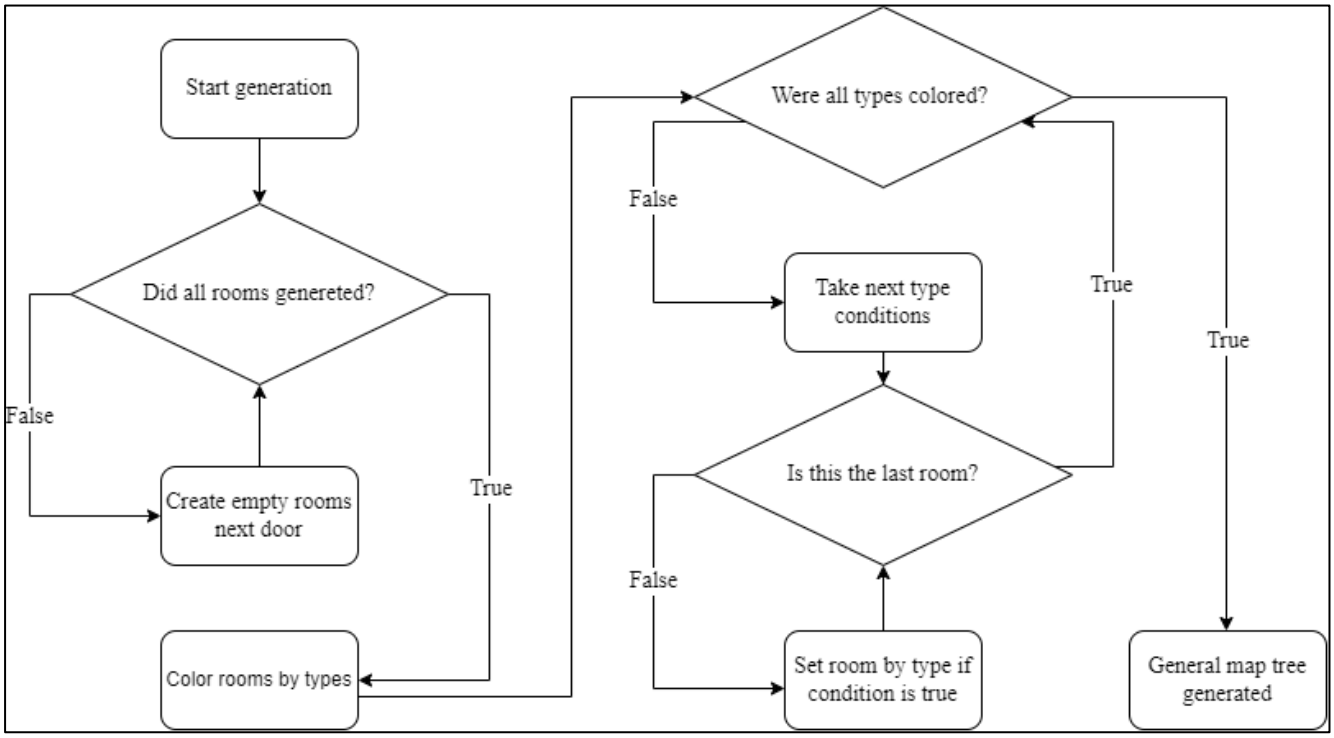


Рисунок 3.4 – Алгоритм роботи процедурної генерації рівнів підземелля (розроблено автором)

При генерації використовується алгоритм, що рекурсивно створює дерево з кімнат, певної довжини, починаючи зі стартової комірки. За таким підходом, середня часова складність становить O(n), тобто лінійна, через те, що на кожну кімнату витрачається єдина ітерація у середньому. Складність за пам’яттю також складає лише O(n), через те, що алгоритм зберігає лише відомість про дані комірок. Після створення комірок та маршрутів, алгоритм зафарбовує вільні місця

кімнатами, за відповідними правилами (додає комірки босів, скарбів, ресурсів, тощо). Складність другої частини алгоритму вже являє собою O(n*m), де n – загальна кількість кімнат, а m – число видів комірок, якими наповнюється мапа. Слід відзначити, що m є невеликим числом, тому O(n*m) є ближчою до лінійної складності ніж до квадратичної. Також у другій частині алгоритму додатково пам’ять не виділяється, оскільки використовується вже задані та записані раніше значення тощо. Водночас, побудова мапи із використанням дерева є доцільним підходом через зручну модель збереження кімнат та переходів між ними.

3.2.4 Система управління

Система управління є достатньо важливою темою, оскільки саме завдяки ній, користувач взаємодіє як з самою грою, так і світом усередині неї. Для її впровадження було використано оновлене API Input System [18], яке надає широкий функціонал з простим налаштовуванням дій по відношенню до фізичних натискань тощо (див. рис. 3.5).

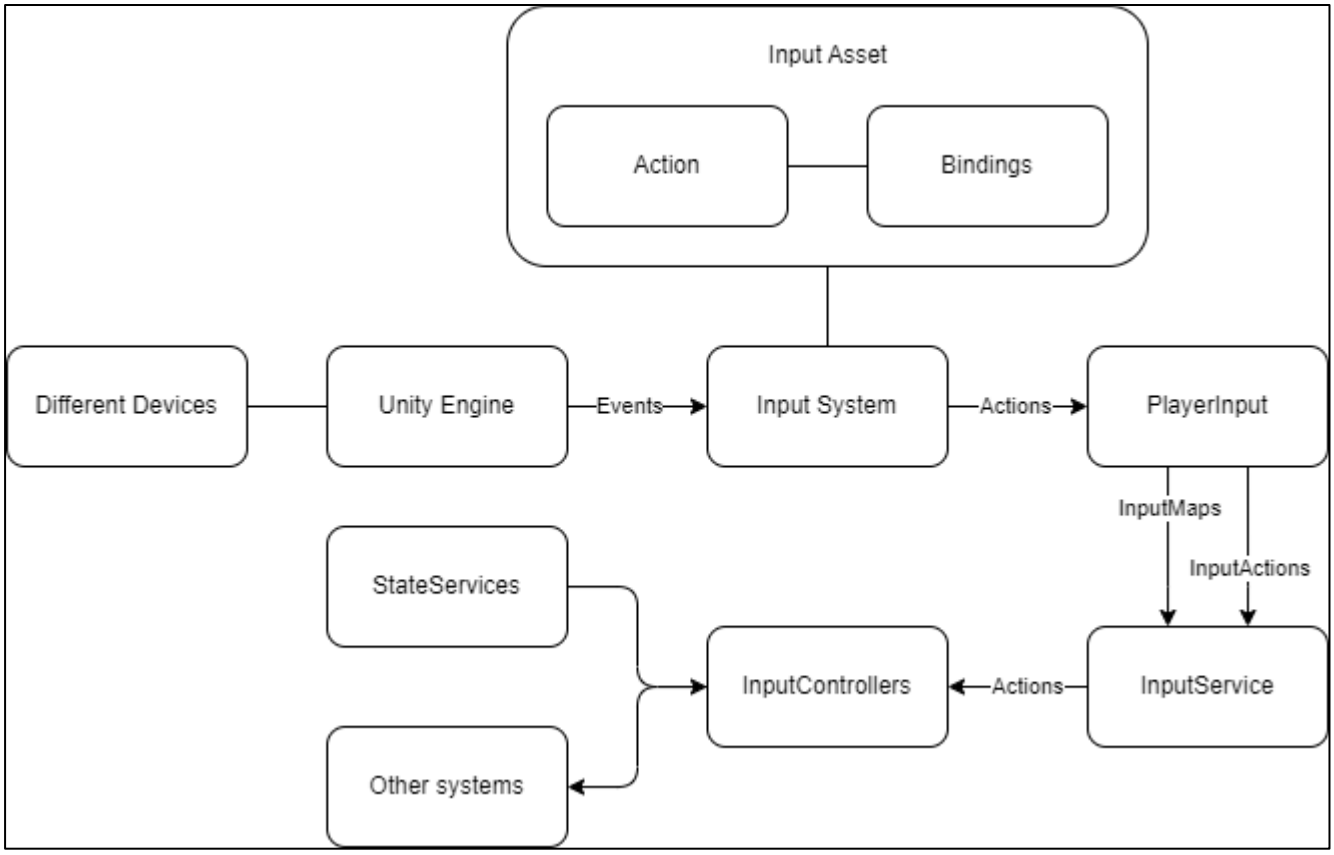


Рисунок 3.5 – Імплементація API InputSystem у власній системі управління (розроблено автором)

Обрана технологія надає можливість прив’язувати методи та логіку саме до моменту натискання чи іншої форми дії, замість перевірки раз у кадр (метод Update()), що призводило до проблем зі зчитуванням натискань, постійних перевірок та подібних рішень. Однак у новій системі обробки натискань, рушій створює зв’язки між дією фізичного пристрою, такого як геймпад, клавіатура, миш тощо, та абстрактним аналогом у грі. Через абстрактний компонент PlayerInput (який приймає будь який набір прив’язок), сервіс з контролю управління грою InputService збирає усі необхідні створені дії, та завдяки патерну проєктування одинак [19, 20], надає доступ до колекції дій з будь якого місця програми. Потім, місцеві контролери натискань прив’язуються до наданих подій сервісом. Останнім елементом ланцюжка є сервіси станів гри, гравця тощо, які визначають можливості певних дій, що відбуваються у контролерах.

3.2.5 Система контролю інтерфейсів

Система із управлінням та контролю інтерфейсів – це допоміжний абстрактний потужний сервіс, що надає функціонал із налаштування поведінки елементів. Зазначена система, контролює те, який інтерфейс відобразити, а який приховати, при кожній спробі показати чи прибрати один з них (див. рис. 3.6).

Дана система представлена єдиним публічним сервісом InterfaceService (створеним за патерном одинак) та абстрактним класом AbstractScreenHelper, який містить два методи: Show() та Hide(). Також система надає множення enum TransitionLaws, які визначають як один інтерфейс, що містить компонент, успадкований від AbstractScreenHelper, має перемінятися із іншим подібним об’єктом. Принцип роботи сервісу доволі простий, слід додати до всіх об’єктів – інтерфейсів компоненти, що успадковані від абстрактного класу, після чого зареєструвати їх попарно, додавши правило, за яким вони міняються. На вибір існує низка правил:

- при показі першого, другий вимикається в одну та обидві сторони;

- при показі першого, другий прибирається, але повернеться коли зникне перший, так само із варіаціями у обидва напрямки, та лише з першого до другого;
- показ першого поверх другого, так само із двома варіаціями направленості;
- показати один інтерфейс на рівні із другим, обостороннє за замовченням;
- відсутність правила, що означає вибір правила за замовченням та\або ігнорування у подальшому;
- відмова спробі показати перший разом із другим та навпаки.

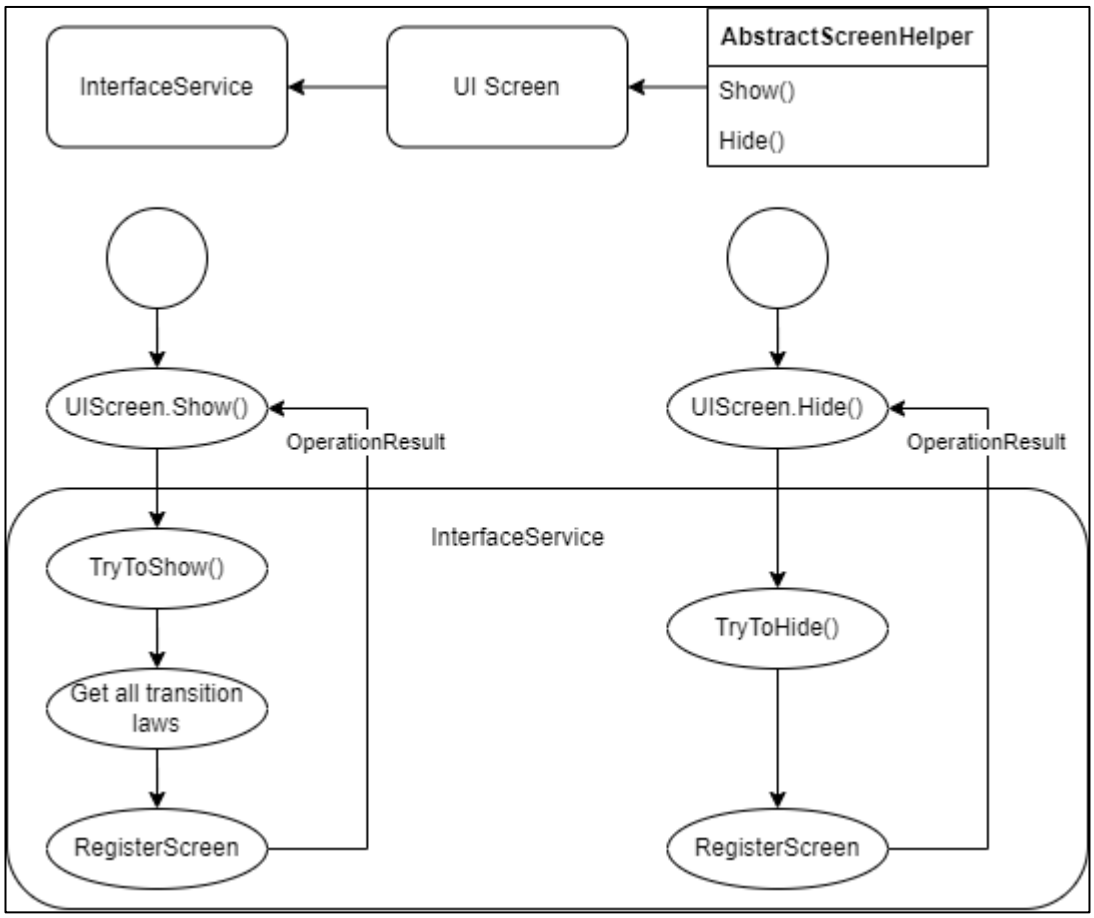


Рисунок 3.6 – Система контролю інтерфейсів (розроблено автором)

Після реєстрації усіх законів та екранів, слід показувати та призовувати інтерфейси через ті самі методи Hide() та Show(), що були успадковані від абстрактного класу AbstractScreenHelper, аби сервіс працював правильно. Слід зазначити, що сервіс як показує за правилами так і приховує за ними. Окрім цього, така обгортка набагато краща, за використання SetActive() методу рушія Unity3D

або його аналогів, через легку модифікуємість правил переходів та можливість розширювати та перевизначати функціонал у спадкоємцях.

3.3 Проєктування інтерфейсу користувача (UI/UX)

3.3.1 Екран завантаження рівня

Програмний модуль, що розроблюється у рамках кваліфікаційної роботи починається з екрану завантаження (див. рис. 3.7), який відображає стан завантаження рівня, у сукупності із ініціалізацією усіх необхідних систем. Кінцевим результатом цього екрану є інтерфейс (див. рис. 3.8) який більше не відображає проценти завантаження, а просить натиснути будь яку кнопку.

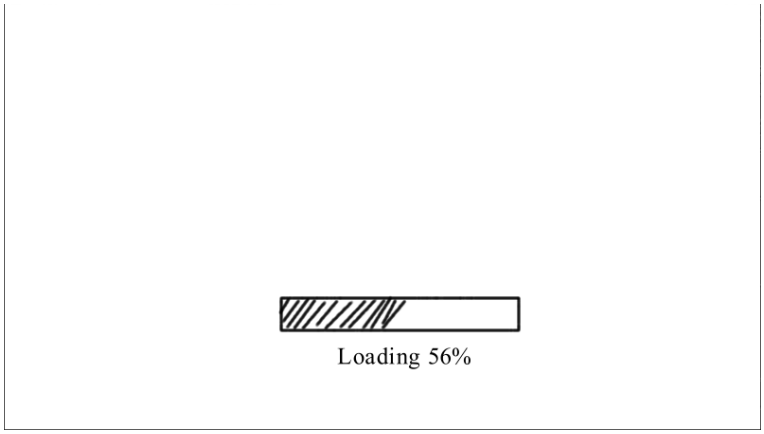


Рисунок 3.7 – Чернетка екрану завантаження у дії (розроблено автором)

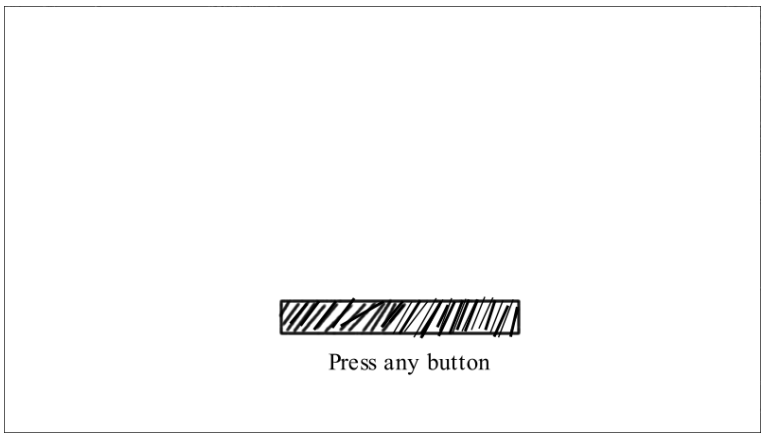


Рисунок 3.8 – Чернетка екрану завантаження у простої (розроблено автором)

Слід зазначити, що чернеткові варіанти інтерфейсу є лише приблизним баченням автора у формі швидкої замальовки, того, як має виглядити справжній.

3.3.2 Мінімапа та мапа

Мінімапа та повноекранна мапа, виконують загально спільну задачу, а саме, демонструють гравцеві відомості про поточно відкриті кімнати, та не перевірені проходи. Їх основна різниця полягає у тому, що мінімапа пасивно висить у правій частині екрану (див. рис. 3.9), демонструючи невелику область навколо поточної кімнати, а повноекранна мапа, у свою чергу, відкривається на увесь екран, та надає можливість гортати її, без обмежень (див. рис. 3.10).

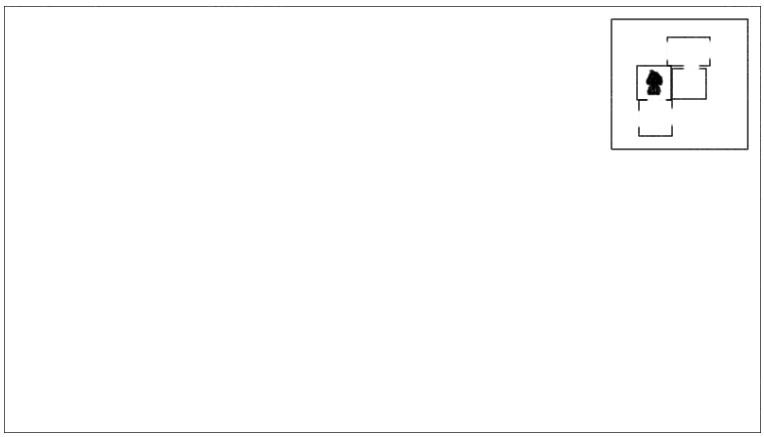


Рисунок 3.9 – Чернетка мінімапи (розроблено автором)

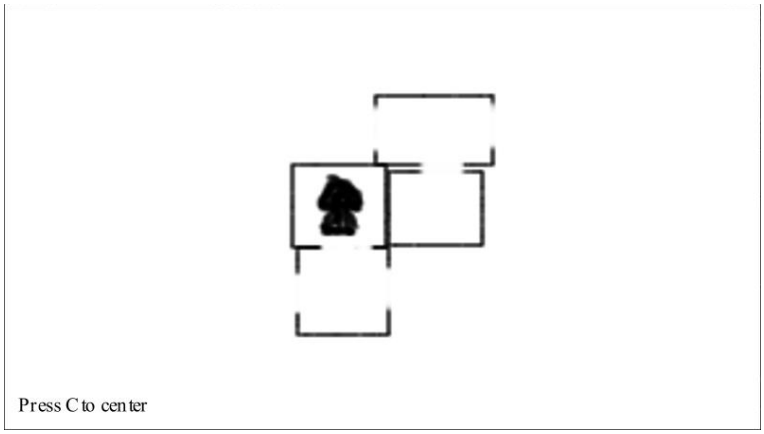


Рисунок 3.10 – Чернетка повноекранної мапи (розроблено автором)

Слід відмітити, що зазначені інтерфейси плануються бути замінниками один одному, що означатиме, що повноекранна мапа, при відкритті, приховує мінімапу, а при закритті, повертає її назад.

3.3.3 Колесо вибору гаджетів

Колесо гаджетів є дуже важливим елементом програмного модуля, що розробляється, оскільки саме через нього, можна обрати елементні гаджети з стихійно бойової системи. Колесо гаджетів є віджитом, що з'являється по натисканню кнопки, та зникає по відтисканню (див. рис. 3.11). До того ж, не менш важливим є момент вибори гаджета (див. рис. 3.12), оскільки у елементах інтерфейсу, пов'язаних з основними системами гри, мають бути наочно прості та зрозумілі для користувачів.

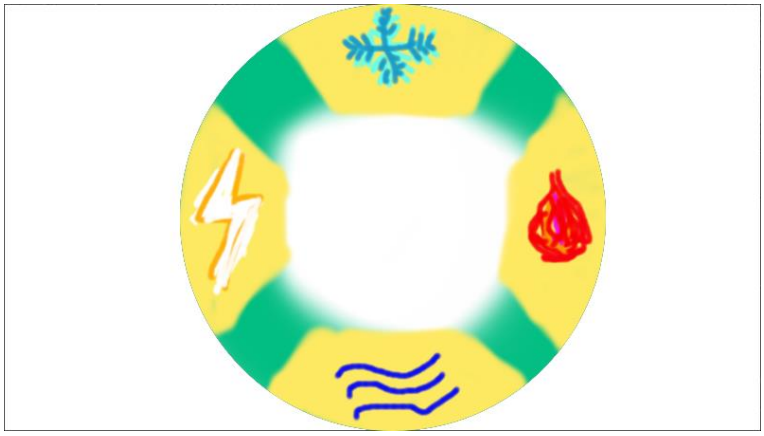


Рисунок 3.11 – Чернетка колеса гаджетів (розроблено автором)

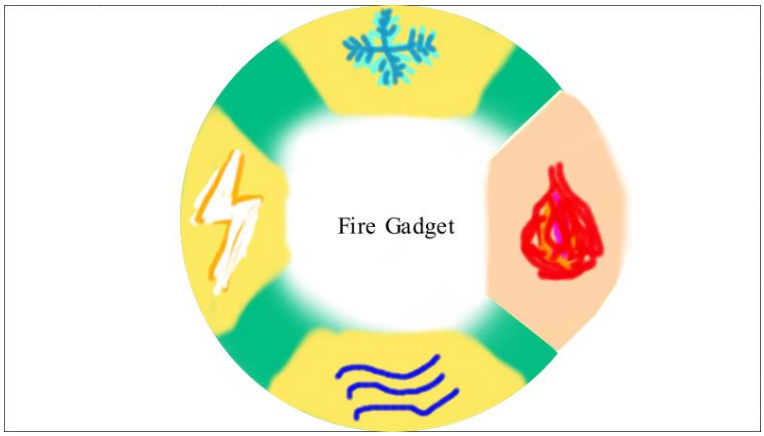


Рисунок 3.12 – Чернетка колеса гаджетів із обраним девайсом (розроблено автором)

Відзначимо, що на чернетках зображено колесо із чотирма «комірками», що відповідають стихійним девайсам: льоду, вогню, воді та електриці. При наведенні

на одну з комірок, планується її візуальне збільшення та появи текстової відомості про обраний гаджет тощо.

3.3.4 Показники здоров’я та розсуду

У іграх, де є суточки із ворогами, особливо у roguelike жанрі, важливо стежити за власним запасом здоров’я, та альтернативи психічної енергії, що використовується для виконання сильних прийомів. Такий елемент користувацького інтерфейсу прийнято розташовувати зліва зверху (див. рис. 3.13).

Рисунок 3.13 – Чернетка шкали запасу здоров’я та психічної енергії (розроблено автором)

Відзначимо, що «психічна енергія» - наративний елемент, що пов’язаний із поясненням механіки підземель та їх наповнення тощо. Фактично, зазначений елемент є альтернативою мани, духовної енергії Ци чи подібних аналогів у інших іграх.

3.3.5 Слоти зі спеціальним вміннями

Відображати гравцю наявні в нього вміння та спеціальні здібності (див. рис. 3.14), є важним моментом, оскільки відсутність такого нагадування призведе до загублення користувачем цієї механіки, через що, гравці зможуть активувати їх хіба на вдачу.

Рисунок 3.14 – Чернетка слотів для особливих вмінь (розроблено автором)

Зазначені комірки (слоти), наразі плануються як відображення інформації про стан обраних спеціальних вмінь, а саме їх доступність, час відновлення, собівартість у психічній енергії тощо. Однак слід зазначити, що змістовне наповнення комірок може бути динамічним.

3.3.6 Інформація про зібрані ресурси, отримання скарбу та підказки

Під час гри, дуже важливо отримувати своєчасний зворотній відгук від гри, у тому, чи іншому виді. Слід зауважити, що текстові повідомлення на екрані, про здобуття якихось ресурсів, рідких скарбів або підказка до дії, є не тільки корисним нагадуванням про щось, а й приємним елементом для гравця, пошуки скарбів або ресурсів якого окупилися (див. рис. 3.15)

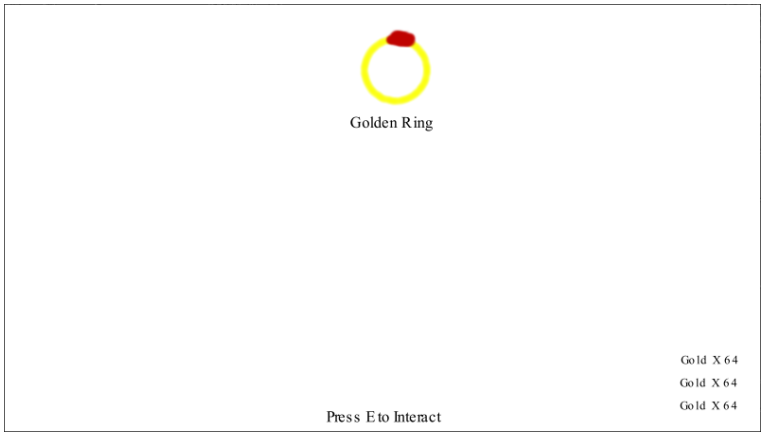


Рисунок 3.15 – Чернетка повідомлень про зібрані ресурси, отримання скарбу та підказки взаємодії (розроблено автором)

Зазначимо, що описанні повідомлення. Не є єдиним інтерфейсним елементом, а виступають сукупністю окремих змістовних елементів, а саме: повідомлення про отримання ресурсу чи знаходження скарба або підказка про можливість виконати певну дію. Усі ці елементи розробляються із власною поведінкою та постають самостійними сутностями, що відображаються гравцеві лише у відповідних ним умовам.

3.3.7 Екран про смерть персонажа

При смерті, гравець має отримати відповідне вікно, яке повідомить йому про цю прикрість, та надасть вибір, з можливістю продовжити гру почавши з початку (повністю перетворивши заново процедурне підземелля та оточення) або кнопку повернення до компоненту гри із розбудовою поселення (див. рис. 3.16).

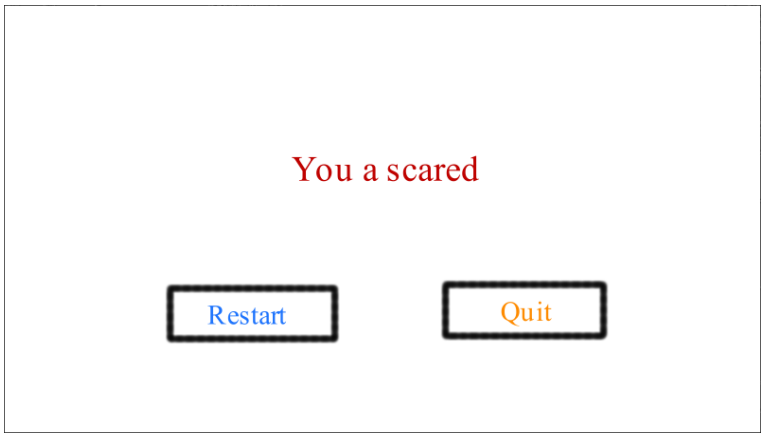


Рисунок 3.16 – Чернетка повідомлень про загибель персонажу (розроблено автором)

Екран смерті, має відображатись із повним прибиранням решти інтерфейсних елементів та мати напівпрозоре «зле» тло, що демонструє місце загибелі персонажа та причину цього. «Злим» тлом постає заливка червоного кольору чи його відтінків або відповідне до змісту зображення (це може змінюватись за обставин чи потреб), що підсвідоме натякає гравцю на поганий кінець.

3.3.8 Екран успішного завершення рівня

При успішному проходженні через портал, гравець має отримати відповідне віконце, яке повідомить йому про це, та надасть кнопку, з можливістю повернення до компоненту гри із розбудовою поселення, разом із усіма зібраними ресурсами (див. рис. 3.17).

Рисунок 3.17 – Чернетка повідомлень про успішну втечу персонажу (розроблено автором)

Екран успішного завершення вилазки до підземелля, аналогічно до посмертного, має відображатись із повним прибиранням решти інтерфейсних елементів, однак, натомість негативному - «злому», мати позитивне - «добре» тло, що демонструє місце, звідки персонаж увійшов до порталу. «Добрим» тлом постає заливка зеленого кольору чи його відтінків, на відміну від красного – «злого», або відповідне до позитивного посилу зображення, що підсвідоме натякає користувачу на хороший кінець.

3.3.9 Екран зібраних ресурсів та скарбів

При проходженні програмного модулю roguelike жанру, що розроблюється, гравець постійно збирає ресурси та рідкісні предмети, що зберігаються у тимчасовому інвентарі, який або зникне при смерті персонажа, або гравець забере його із собою, при проходженні через портал. Слід зазначити, що для цього інвентарю, необхідне графічне супроводження, а саме інтерфейс (див. рис. 3.18).

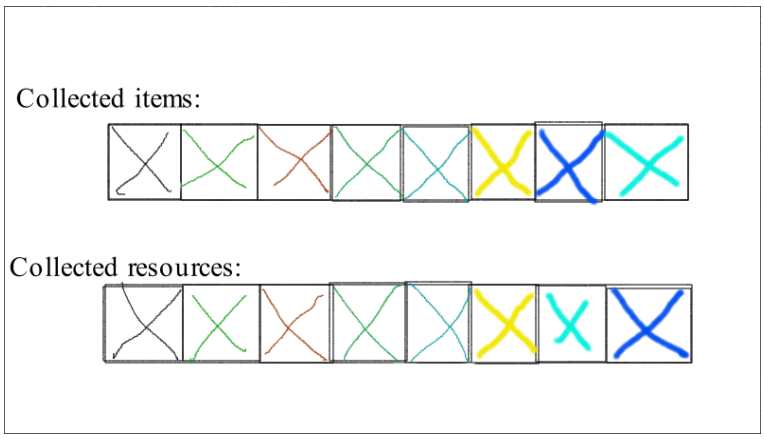


Рисунок 3.18 – Чернетка тимчасового інвентарю з речами (розроблено автором)

Зазначений інтерфейс інвентарю має відображати усі речі та ресурси, зібрані гравцем, розділяючи їх на відповідні дві групи, та відображати певні відомості про них, наприклад назву предмету або кількість матеріалів тощо. Тло екрану має бути напівпрозорим, із нейтральним забарвленням (сірого кольору, його відтінків або заповнене аналогічним зображенням – візерунком тощо), аби користувач постійно бачив власні обставини, та не відволікався під час важкої бойової ситуації тощо.

Слід зазначити, що тимчасовий інвентар – єдиний інтерфейс що може бути відображеним на тлі екрану завершення рівня, оскільки він містить інформацію про здобуття персонажу. Перегляд зазначеної інформації може бути корисним, адже гравець, під час повернення до компоненту гри з розбудовою поселення, матиме на увазі, які ресурси він тільки-но отримав й одразу придумати куди їх витратити тощо.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Вибір інструментарію для програмної реалізації програмного модулю

У якості рушія, для реалізації проєкту «The fate of the liar», зокрема програмного модулю roguelike, що виступає предметом поточної кваліфікаційної роботи, було обрано Unity3D, версії 2023.1.17f1, яка має довгу підтримку оновленнями. Відповідно до рушія, мовою програмування було обрано C#. Проєкт для розробки ігрового застосунку було створено за оновленим графічним конвеєром, а саме URP [21, 22], що покращує масштабованість проєктів, їх налаштованість та надає багато інших можливостей, щодо візуалізації та графічного наповнення продуктів.

Рушій для розробки ігор Unity3D має у собі велику кількість різноманітних вбудованих модулів, але найбільш важливими для розробки зазначеного компоненту ігрового застосунку, постають AI Navigation, що відповідає за побудову мапи маршрутів та пошуку шляхів за нею; VFX Graph [22, 23], для створення візуальних ефектів до стихійної системи; та нова Input System, що базується на подіях, замість попередника, що використовував значно менш масштабовану та зручну функціональність.

У якості інструмента для написання коду та його редагування тощо, було застосовано Visual Code від компанії Microsoft,у поєднанні із необхідними модулями розширення для роботи із рушієм Unity3D та пов’язаною з цим активністю.

Для кооперування з іншими членами команди, що також розробляться певні програмні модулі для гри «The fate of the liar», було застосовано систему контролю версій Git, зокрема сервіс GitHub.

4.2 Опис програмної реалізації модулю roguelike

4.2.1 Опис системи завантаження рівня

Процес завантаження рівня складається з декількох етапів. При переході до сцени з ігровим процесом програмного модулю roguelike, гравець бачить лише екран завантаження, який відображає поточний прогрес та повідомляє про останній

завантажений модуль системи (див. рис. 4.1). Одразу після відкриття сцени та її ініціалізації рушієм, сервіс «LoadingService», що відповідає за процес завантаження рівня, обирає випадкове зображення для відповідного екрану з наданої йому колекції (див. рис. 4.2). Слід відмітити, що оскільки цей процес відбувається всередині методу Start() (що є одним з методі конвеєру оброки рушія Unity3D, а саме класу MonoBehaviour) [24], зазначений процес буде відбуватися під час обробки першого кадру існування екрану, то ж користувач не помітить ніякої підміни зображення за замовчуванням, на отримане з колекції.

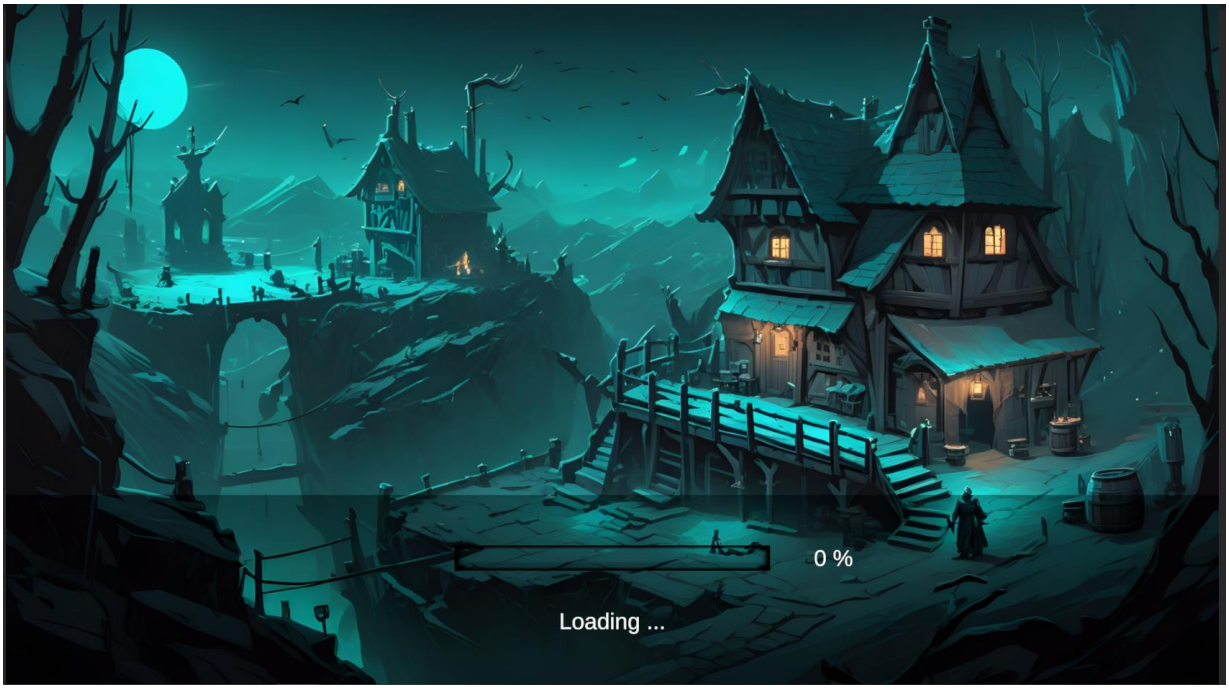


Рисунок 4.1 – Зображення початкового стану екрану завантаження (розроблено автором)

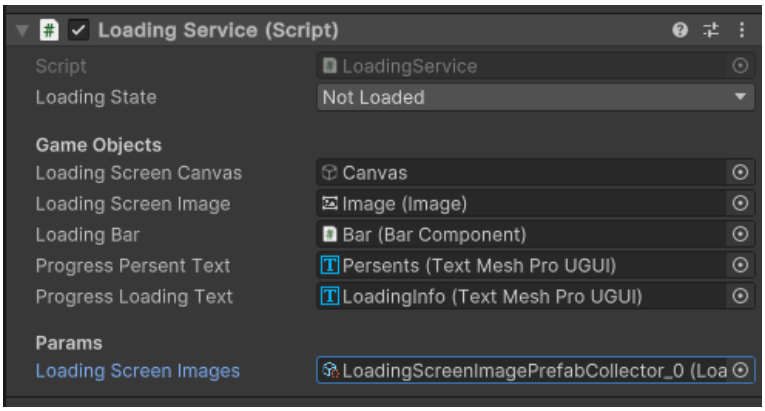


Рисунок 4.2 – Зображення параметрів сервісу завантаження, зокрема колекції заставок (розроблено автором)

Після ініціалізації екрану, починається процес завантаження, сервіс збирає усі об’єкти, що реалізовують інтерфейс «ILoading» та сортує їх, згідно встановленої черги завантаження (див. рис. 4.3). Після чого, починає запускати у «корутинах» [25] обробки методів Load(), зі згаданого інтерфейсу, для поточної черги елементів що завантажуються, після завершення чого, переходить до наступного прошарку. Увесь цей час, після завантаження кожного нового елементу, користувач бачить збільшення показника прогресу та змінення інформації, щодо останнього завантаженого елементу (див. рис. 4.4). Після завершення обробки останнього компоненту, гравець отримує відповідне повідомлення про можливість діяти, та відповідно отримує управління у свої руки (див. рис. 4.5)

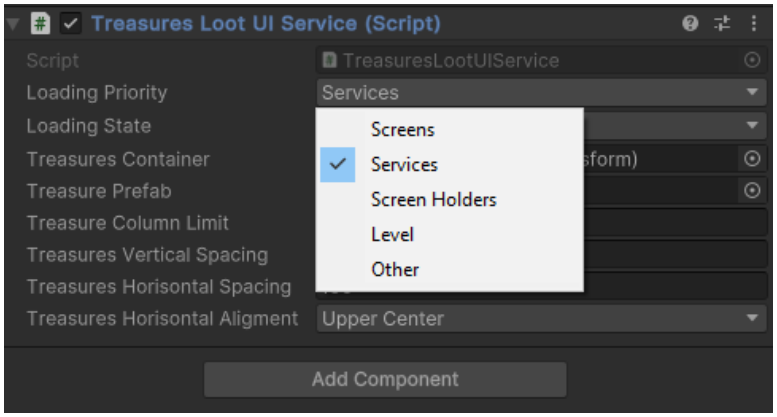


Рисунок 4.3 – Зображення шарів черг для завантаження (розроблено автором)

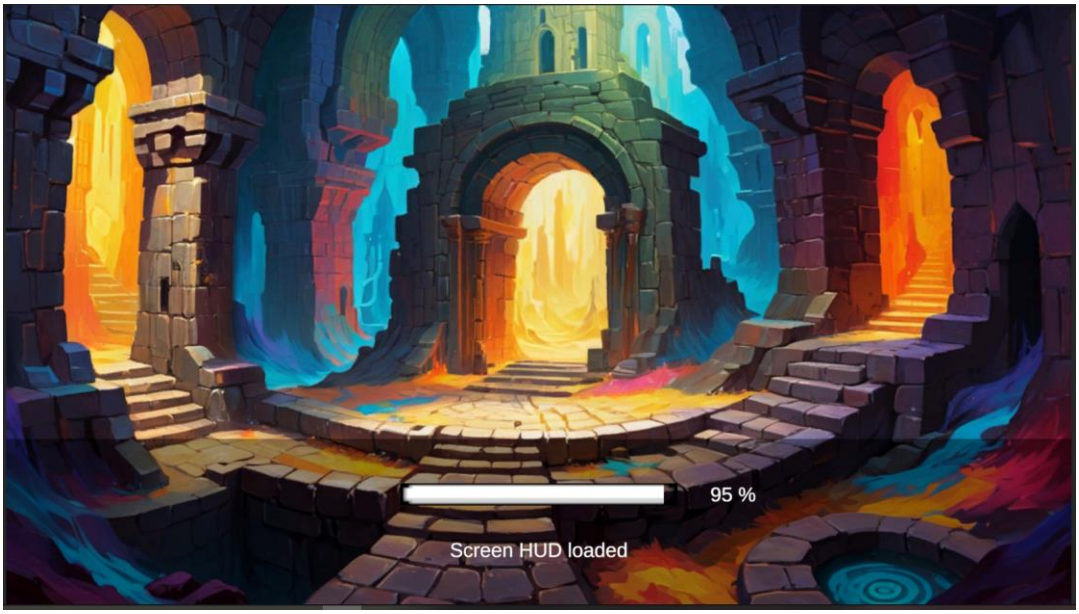


Рисунок 4.4 – Зображення шарів черг для завантаження (розроблено автором)

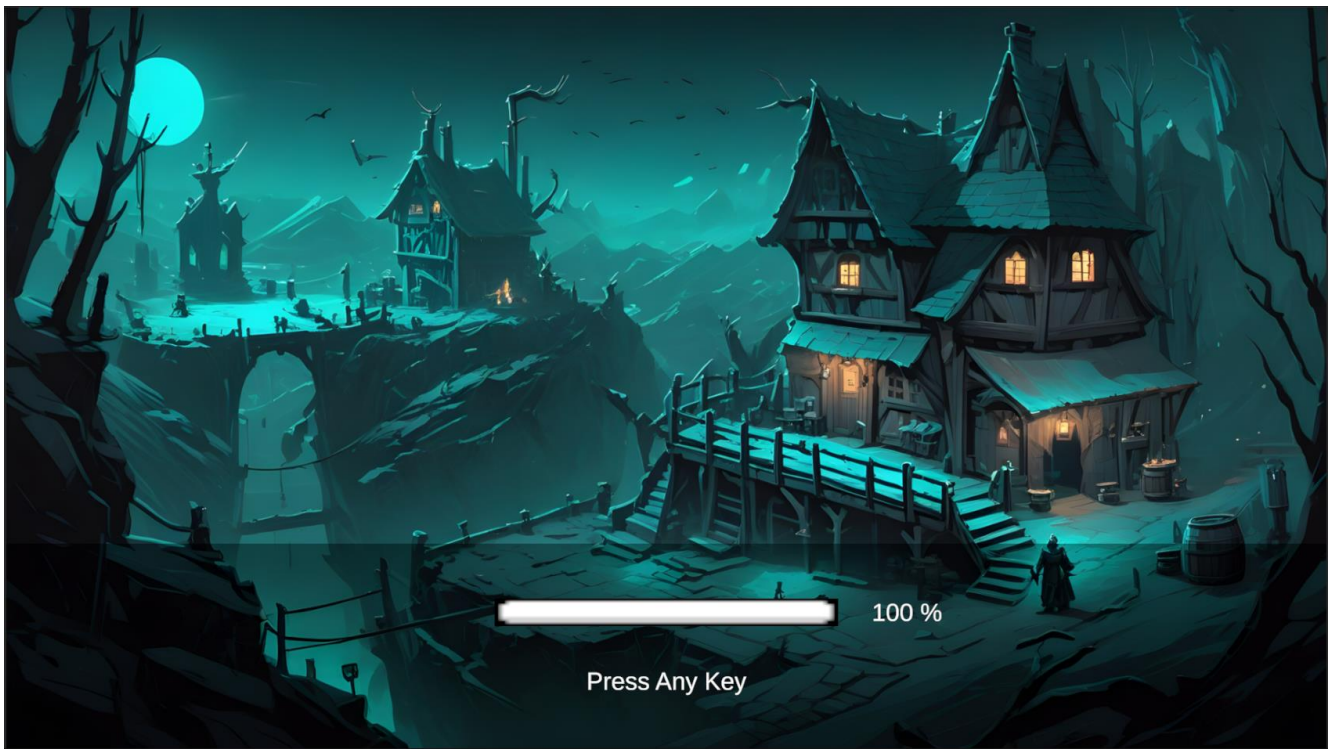


Рисунок 4.5 – Зображення екрану завантаження після кінцевої обробки (розроблено автором)

Результатом роботи зазначеної системи виступає правильно завантажені служби та компоненти на усіх рівнях, завдяки чому жодна з логічних чи апаратних зв’язок не була порушена, на відміну від підходу із використанням методів MonoBehaviour, таких як Start(), Update(), FixedUpdate() тощо. Останні мають неоднозначність з точки зору передбаченого результату, оскільки зазначенні події викликаються «одночасно», то ж якщо між декількома елементами системи є певна необхідність у послідовності ініціалізації, за стандартних методів, без купи запобіжних механізмів, система дуже часто буде завантажуватися із помилками через оприявлений фактор.

4.2.2 Опис системи процедурної генерації підземелля

Відзначимо, що загально, сам алгоритм генерації рівнів [26], є досимвольним втіленням спроєктованого компоненту, що розглядався у підрозділі 3.2.3, однак є важливими саме деталі його реалізації.

По-перше, слід зауважити, що до генерації послідовності кімнат, алгоритм якого проєктувався раніше, було додано ще два змістовних компоненти, а саме відтворювач рівня у просторі інтерфейсу, а саме мапи та мінімапи «Minimap» (див. рис. 4.6) та трьох вимірному просторі – самі кімнати із відповідним наповненням «LevelRoomSpawner» (див. рис. 4.7).

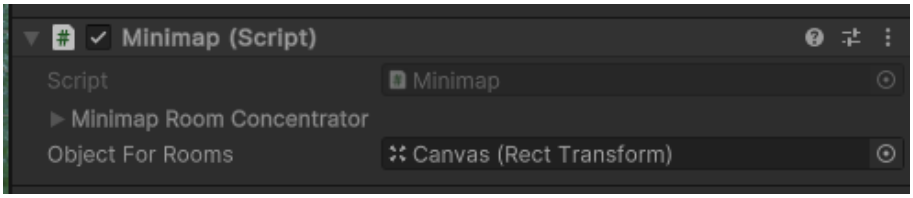


Рисунок 4.6 – Зображення класу «Minimap» у інспекторі рушія Unity3D (розроблено автором)

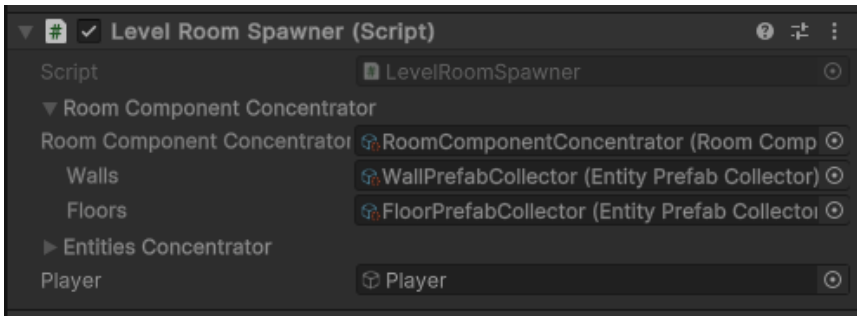


Рисунок 4.7 – Зображення класу «LevelRoomSpawner» у інспекторі рушія Unity3D (розроблено автором)

Відзначимо, що вказані поля зазначених класів: «MinimapRoomConcentrator» та «RoomCompoentConcentrator», є власними типами, що втілюють у собі об’єднання префабів, що використовуються кінцевими компонентами, та відповідний функціонал, що є допоміжним, для зручного використання продемонстрованих концентраторів. Також важливим є те, що для спрощення подальшої розробки, було розроблено власний користувацький атрибут «InlineScriptableObject» (див. додаток Д, зокрема розділ Д.1, із усіма підрозділами), завдяки якому, поля що були спадкоємцями «ScriptableObject», можуть бути переглянуті та відредаговані напряму в місці додавання. Саме тому, згадані вище поля «MinimapRoomConcentrator» та «RoomCompoentConcentrator» мають у інспекторі рушія відображення власних властивостей. Також підхід із

спадкоємцями «ScriptableObject» є дуже зручним за рахунок того, що можна створити безліч варіацій одного типу, та замінювати їх один на інший за потреби, не заносячи у колекцію усі потрібні елементи з початку тощо.

Важливим додатком до алгоритму генерації рівня постає сервіс «GenerationPipelineService», що так само є компонентом, який завантажується у власній черзі (див. рис. 4.8). Зазначений сервіс має на меті організувати послідовність генерації рівня, починаючи зі створення будови підземелля з кімнат, завершуючи створенням фізичних кімнат (див. рис. 4.9) та елементів інтерфейсу (див. рис. 4.10), за результатами генерації абстрактного рівня. Також варто відмітити, що зазначений сервіс містить поле з налаштуванням класу «Generator», який створює підземелля, опираючись саме на ці показники (див. рис. 4.11).

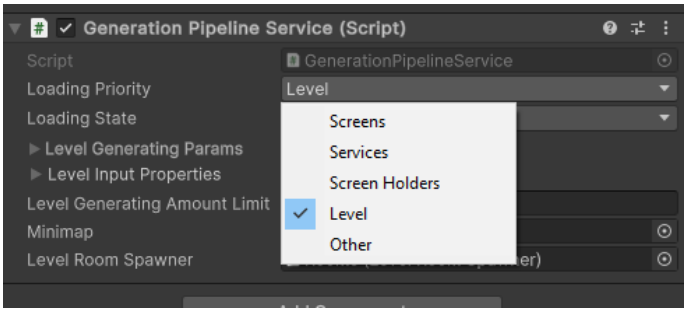


Рисунок 4.8 – Зображення класу «GenerationPipelineService» у інспекторі рушія Unity3D (розроблено автором)

Рисунок 4.9 – Зображення кімнат у ієрархії об’єктів(розроблено автором)

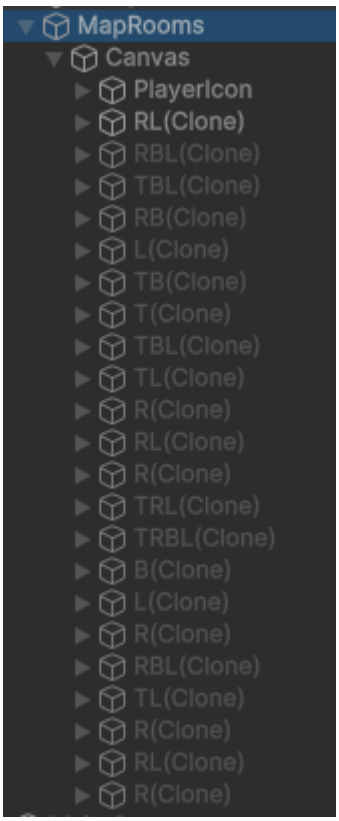


Рисунок 4.10 – Зображення комірок мапи у ієрархії об’єктів(розроблено автором)

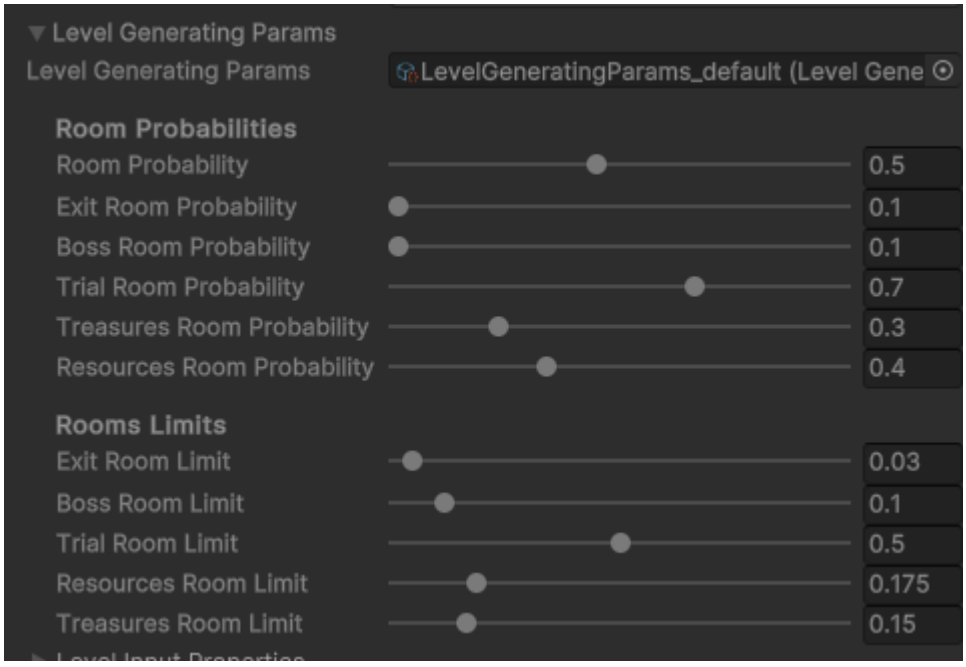


Рисунок 4.11 – Зображення налаштування параметрів генерації послідовності кімнат на рівні підземелля(розроблено автором)

Підсумовуючи стан розробленої системи процедурної генерації рівнів, остання є комплексним елементом програмного модулю, що розробляється у

рамках кваліфікаційної роботи бакалавра. Алгоритмічно, розроблений компонент є досить оптимізованим й швидким, аби генерувати найбільші рівні, які передбачаються взагалі, за менш ніж секунду (згідно підрозділу 3.2.3 з розбором архітектури та часових й пам’ятних складностей). З архітектурної точки зору, система генерації рівнів вийшло гарно розподілена та спроєктовано, аби кожен її компонент був легко модифікуємі та налаштовані під відповідні потреби.

4.2.3 Опис системи контролю інтерфейсів та ігрових дозволів

Важливою системою, програмного модулю roguelike, постає сервіс управління інтерфейсами. Загальну архітектуру компоненту було розглянуто та спроєктовано у підрозділі 3.2.5. Але варто розглянути саме результат розробки зазначеної системи, для визначення її характеристик з точки зору усього проєкту загалом, тощо.

Кожен окремий інтерфейс, що взаємодії із зазначеним сервісом має власний клас, що успадковує абстрактний тип «AbstractScreenHelper», що є класом, який завантажується відповідною системою, що розглядалась у підрозділі 4.2.1. Також, бажано реалізовувати інтерфейс «IPlayerUIPermissions», для урахування обмежень усіх відображених інтерфейсів на можливостях гравця, однак це не є обов’язковим фактором для використання загальної системи управління інтерфейсами – «InterfaceService». Кожен подібний клас, що управляє окремим екраном, має певний вигляд (див. рис. 4.12).

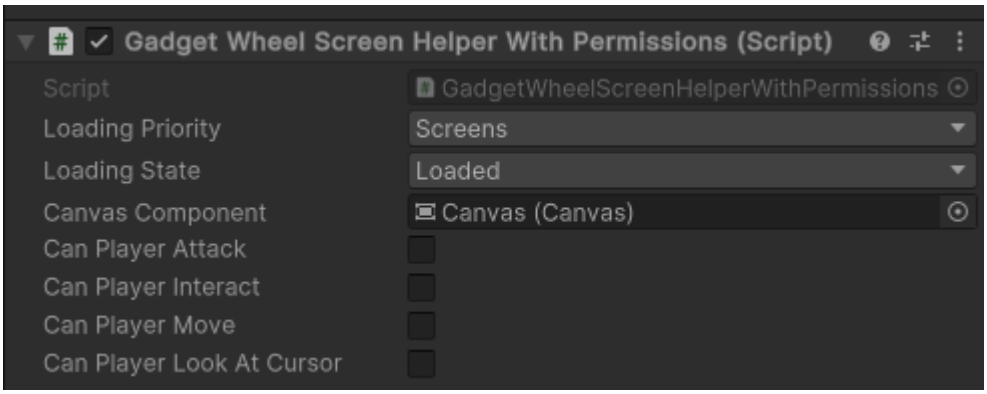


Рисунок 4.12 – Зображення типового класу, що успадкував «AbstractScreenHelper» та реалізував «IPlayerUIPermissions» (розроблено автором)

Слід зазначити, що компоненти системи з дозволами гравця та «InterfaceService» загалом, які розглядаються, підтримують більш складну ієрархію інтерфейсів, а ніж просто низка екранів з однаковими можливостями. Так у поточній конфігурації користувацького інтерфейсу, складну будову мають «HUD», що вміщає в себе три екрани, один з яких – «UIMessages», також вміщає три дочірніх інтерфейси (див. рис. 4.13). Для урахування подібної ієрархії, «InterfaceService» розшукує компоненти, що успадкувались від абстрактного типу «AbstractScreenHelper», у батьківських ігрових об’єктах, та підтримує приховування та показ інтерфейсів з урахуванням такої конфігурації. Однак у випадку з дозволами на дії гравця, спільним абстрактним родичом є інтерфейс «IPlayerUIPermissions», через що, у базовому наповненні рушія Unity3D, не можна створити поле, що буде видне та доступне до редагування у інспекторі. Для вирішення подібної проблеми, було створено користувацький атрибут «SerializeInterface» (див. додаток Д, а саме розділ Д.2). У підсумку, результат застосування описаного атрибуту має наступний вигляд вигляд (див. рис. 4.14), який демонструє втілення класу зі складною ієрархією та її підтримкою з боку системи дозволів гравця на дії.

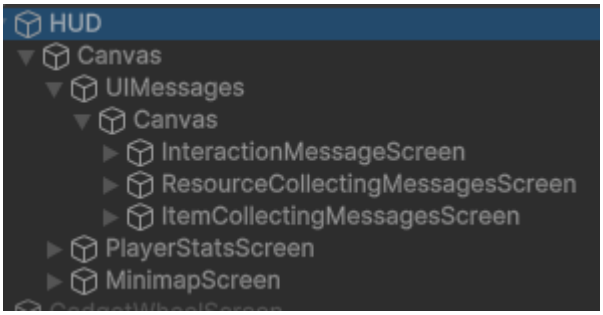


Рисунок 4.13 – Зображення складної ієрархії інтерфейсів (розроблено автором)

Розглядаючи систему управління інтерфейсами, варто звернути увагу на механізм роботи та підходу до встановлення правил переходів між екранами. Стосовно першого питання, воно розглядалось у розділі із проєктуванням архітектури (див. розділ 3.2.5). Стосовно другого, варто розглянути зображення (див. рис. 4.15). На зазначеному рисунку, відображено клас «InterfaceService» у інспекторі рушія Unity3D.

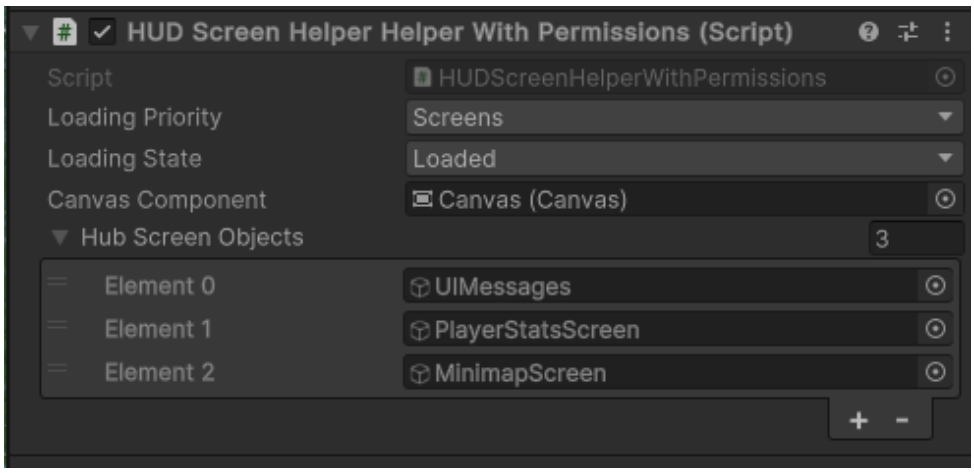


Рисунок 4.14 – Зображення класу із урахуванням складної ієрархії інтерфейсів (розроблено автором)

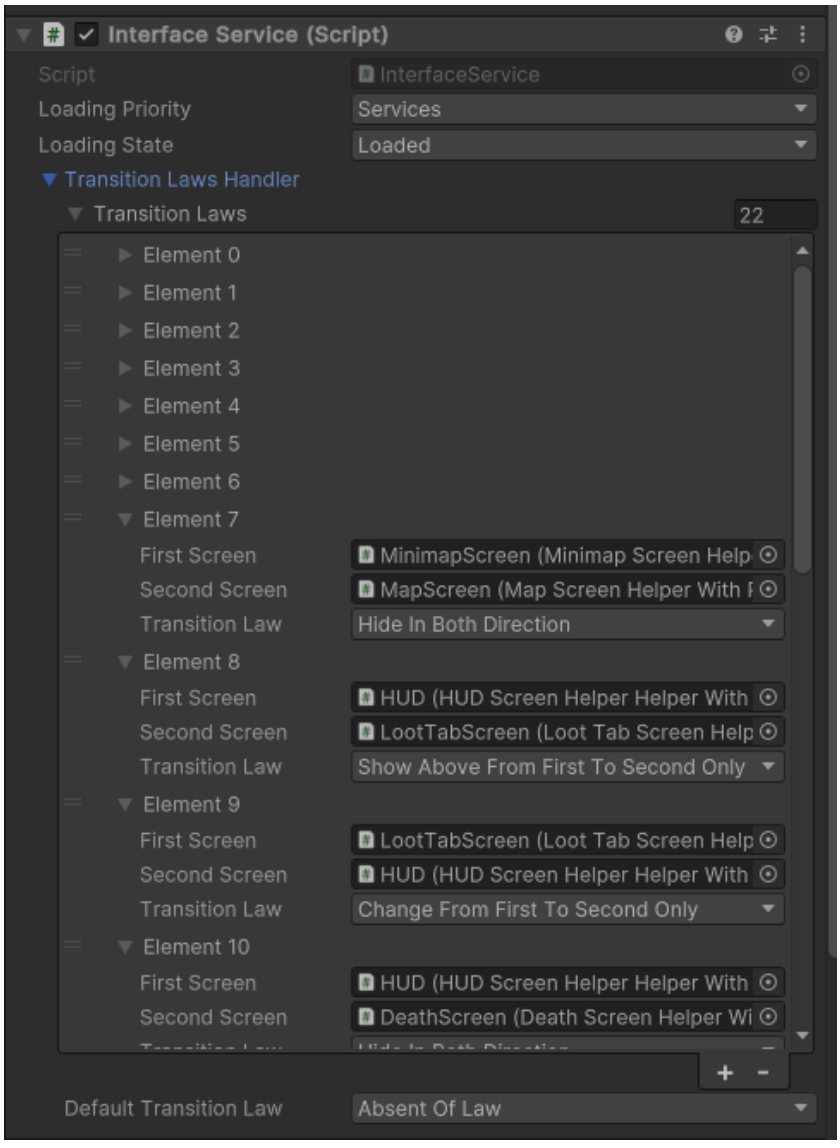


Рисунок 4.15 – Зображення класу «InterfaceService» у інспекторі рушія (розроблено автором)

На зображені продемонстровано, що «InterfaceService» містить у собі підклас «TransitionLawsHandler», який виконує дії зі зберігання у собі записаних правил переходів, які записані у вигляді сутностей із двома об’єктами екранів та значенням enum «TransitionLaws», що визначає собою закон переходу. Під час роботи сервісу, при спробі показати якийсь інтерфейс, спочатку система перевіряє можливість цієї дії за переліком поточно-відображених об’єктів, а після, за умови можливості дії, запитує список правил переходів між наразі ввімкнутими інтерфейсами та тим, що намагаються продемонструвати користувачеві. При відсутності правил між певними парами, підклас «TransitionLawsHandler» повертає закон за замовчування (поле «DefaultTransitionLaw»). Після цього, виконується перевірка усіх правил та результатом спроби продемонструвати екран, буде вердикт з обраних правил.

Також слід відзначити наявність допоміжного сервісу, до основного, «DungeonScreenHolders» (див. рис. 4.16), діяльність якого спрямована на наданні зручного доступу до кожного передбаченого інтерфейсу, саме тому зазначена підсистема реалізує патерн одинак, задля однакового доступу звідусіль.

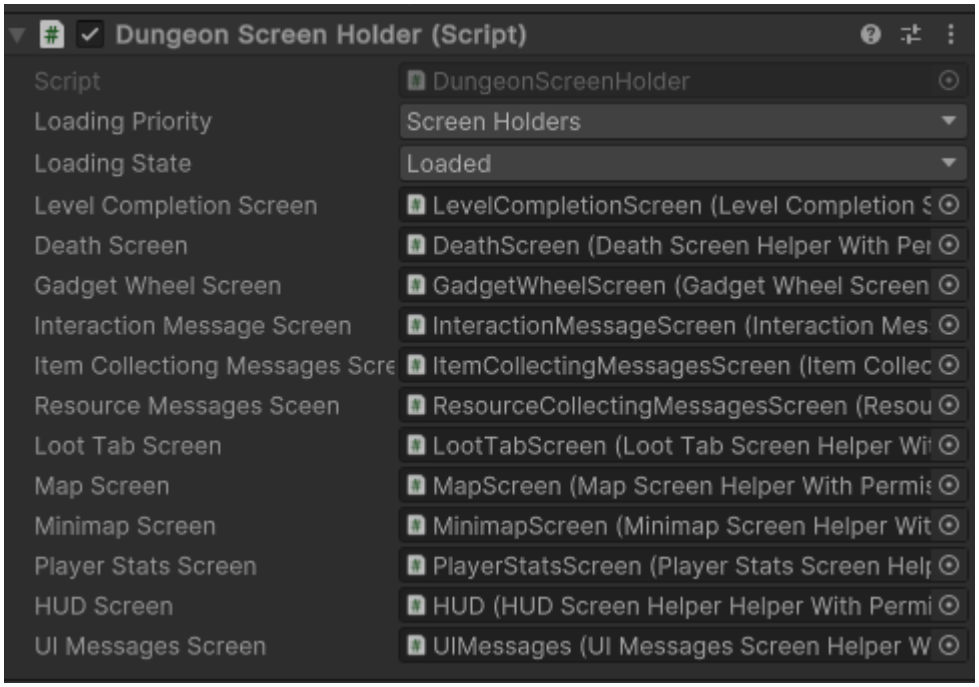


Рисунок 4.16 – Зображення класу «DungeonScreenHolder» у інспекторі рушія (розроблено автором)

Варто відзначити, що системи керуванням стану інтерфейсів та дозволів користувачів, що залежить від поточних екранів, є досить гнучкими, та легко масштабуються, завдяки абстракціям та достатньому рівні абстрагованості від сторонніх систем та повної самостійності як окремого програмного компоненту. Із використанням даних сервісів, достатньо вказати необхідні правили переходів екранів та після чого більше не перейматися через додаткові перевірки у коді, оскільки сервіс сам вирішить, чи можна показувати інтерфейс та поверне детальний результат спробу.

4.2.4 Опис системи управління

Варто розглянути також реалізацію системи управління. Як визначалось у розділі із проєктуванням (див. підрозділ 3.2.4), у якості інструменту зчитування введень з ігрових девайсів, було використано нову InputSystem, що базується на подіях та патерні «event bus» [27]. Зазначений модуль надає можливість створити власний ассет з налаштуванням прив’язки команд користувацьких девайсів, до ігрових подій, які розділені на «мапи» - групи дій, які можуть бути вимкненими, ввімкненими тощо (див. рис. 4.17).

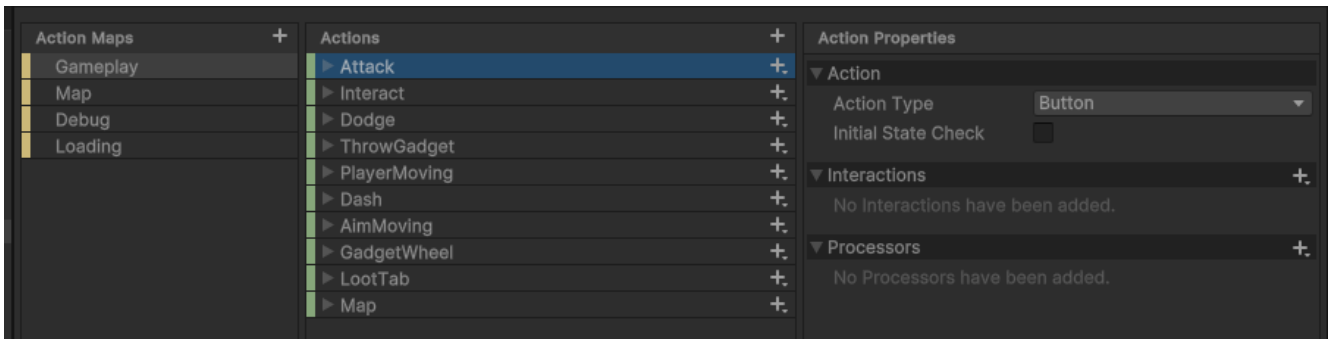


Рисунок 4.17 – Зображення ассету управління «RoguelikeInputSystem» (розроблено автором)

Для надання централізованого доступу до відповідних «мапів» та дій, було створено сервіс «RoguelikeInputSystemService», що вміщає у себе тримачі, що відповідають одному з «мапів» кожний. Додатково, було створено функціонал, що дозволяє блокувати певні дії, для випадків, коли певна подія є причиною якихось обмежень тощо. Відзначимо, що «RoguelikeInputSystemService» сервіс так само

реалізує патерн проєктування одинак. Загалом, підхід до створення описаного сервісу був обумовлений кращими практиками, з питання влаштування системи управління [28]. Також слід відмітити, що у якості інструменту, через який клас «RoguelikeInputSystemService» отримує доступ до створеної системи управління, було використано узагальнений компонент «PlayerInput», що обгартує ассет дій та прив’язок, завдяки чому, дуже легко змінювати різні системи управління, не втручаючись у вже написаний код сервісів тощо.

У доповнення до описаного сервісу, були створені самостійні контролери, що мали на меті надання функціоналу з управлінням певними елементами програмного модулю, такими як гравець, мапа, колесо гаджетів, вікно здобутих речей тощо. Зазначені обробники управління базуються на патерні «event bus», завдяки чому, додатково знижують навантаження на систему, оскільки кожна з подій викликається лише раз, коли було відправлено команду з користувацького девайсу. До таких контролерів відносяться класи: «GadgetWheelInputController», «MapInputController», «LootTabInputController» та «PlayerInputController».

Додатково слід відмітити, що було розроблено відокремлений сервіс, «RoomMovementService», що повністю втілює патерн «event bus», адже містить події, на які підписується безліч розрізнених компонентів, що мають виконати певну роботу у одну мить один з іншим, що у повній мірі демонструє корисність зазначеного патерну.

Результатом реалізації системи управління із використанням кращих практик, патернів проєктування та нової InputSystem, постав узагальнений компонент, що має розрізнені самостійні елементи, які залежать лише від загального доступу до самого управління, що робить розроблену систему легко масштабованою, практичною та зручною у використані, для створення нових контролерів введення чи покращення існуючих тощо.

4.2.5 Опис системи штучного інтелекту ворогів

Розглядаючи систему штучного інтелекту ворогів, слід почати із наведення зразку із налаштованим ворогом, що має певну поведінку, характеристики тощо

(див. рис. 4.18). Щодо загальної архітектури рішення будови поведінкових алгоритмів ворогів, слід звернутись до відповідного розділу, а саме підрозділу 3.2.2.

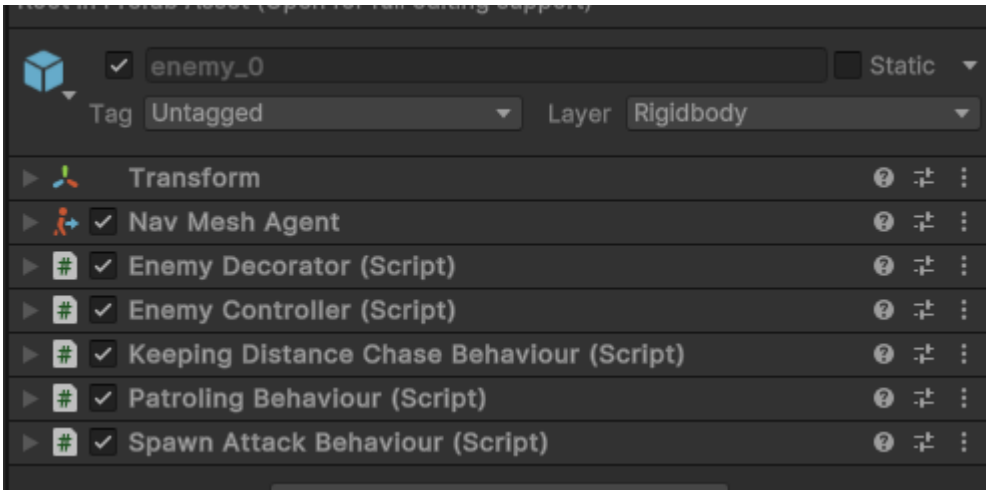


Рисунок 4.18 – Зображення ассету управління «RoguelikeInputSystem» (розроблено автором)

За наведеним зображенням, видно, що основою одиниці ворога є «EnemyController», який обробляє поведінкові патерни, що були підключені (такі класи як «KeepingDistanceChaseBehaviour» - поведінка при переслідуванні цілі, «PatrolingBehaviour» - принцип пошуку героя та «SpawnAttackBehaviour» - різновид атакуючої здібності), «NavMeshAgent», що є компонентом пакету AI Navigation, та відповідає за переміщення за навігаційним мешем та «EnemyDecorator», який визначає параметри ворогу з низки варіантів, що має поточний варіант префабу та екземпляр класу зокрема.

Розроблена система є достатньо легкою у застосуванні та подальшій модифікації та масштабуванні, оскільки, як зазначалось на етапі проєктування, усі поведінкові патерни, походять від трьох основних абстракцій, а саме переслідування гравця, його пошук та атакуюча дія.

Результатом розробки штучного інтелекту ворогів, постали наступні сім поведінкових патерни:

а) пошуковий шаблон;

б) наздоганяючий шаблон:

1) патерн тримання певної відстані між собою та персонажем;
2) патерн прямого наздоганяння цілі;

в) атакуючий шаблон

1) патерн атаки укусом персонажа;
2) патент атаки ударом;
3) патерн атаки із призовом маленьких прислужників;
4) патерн атаки зі жбурлянням снарядів у головного героя.

Завдяки замінності патернів однієї дії, потенційна варіативність навичків кожного окремого ворогу вражає, оскільки навіть із поточним, невеликим, переліком, можна створювати комбінації ворогів та їх вмінь тощо.

4.2.6 Інші систем та механіки

Розглядаючи решту механік, сервісів, систем та допоміжних конструкцій, що не були розглянуті у попередніх підрозділах, слід зауважити, що останні, є здебільшого елементами чи слідством описаних програмних рішень. У підрозділах 4.2.1 – 4.2.5 та 3.2.1 – 3.2.5, було розглянуто найбільш глобальні та важливі програмні компоненти та системи, з яких здебільшого складається програмний модуль roguelike, що виступає предметом кваліфікаційної роботи.

Подальший аналіз менших підсистем можна вважати надмірним та залишити на момент перегляду вихідного коду, чи розробки загалом.

4.3 Розроблені варіації ворогів

Під час створення програмного модулю roguelike, було створено п’ять префабів ворогів, а саме чотири звичайних монтра та один, більш здібний, бос. Слід зауважити, що бос має таку саму зовнішність, як звичайний огр, однак на відміну від нього, він має більший асортимент можливостей, кращі характеристики, цінніші винагороди за своє вбивство та просто більші розміри. Моделі ворогів наведені нижче (див. рис. 4.19 – 4.22), вороги знаходяться у Т-позі, для більш правильного наочного порівняння розмірів, зовнішності тощо.

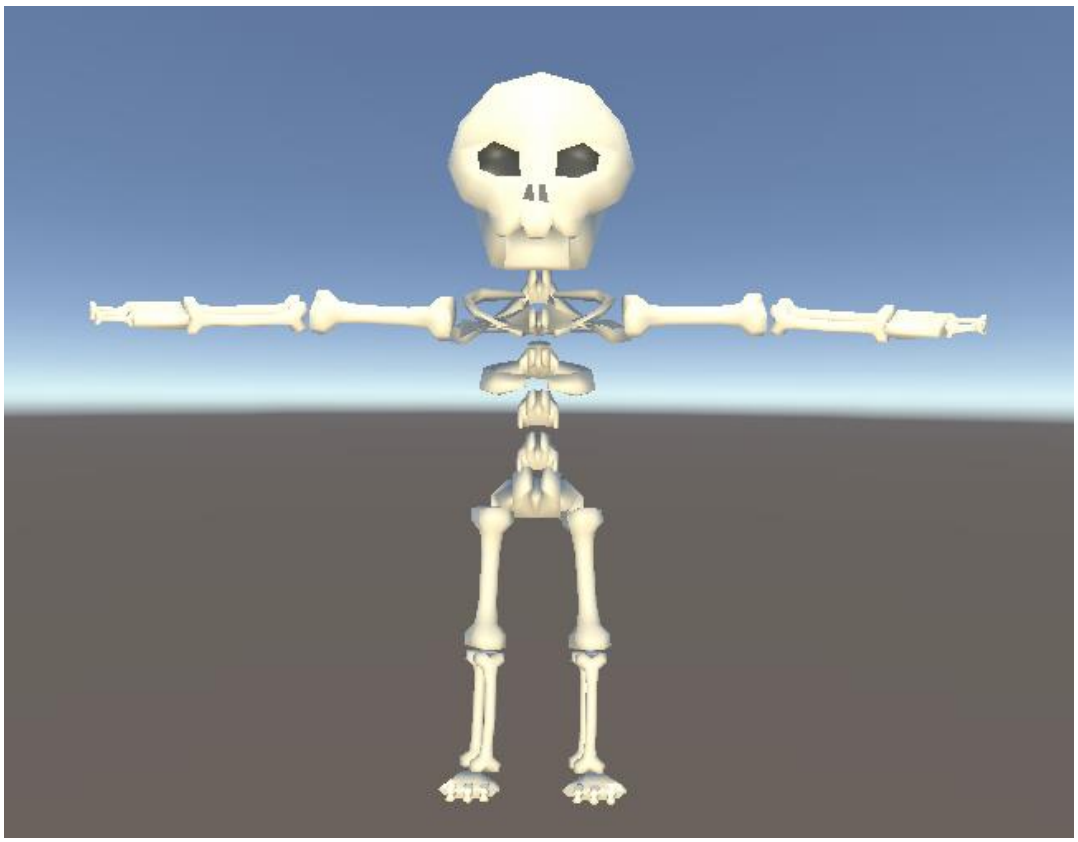


Рисунок 4.19 – Зображення скелету у Т-позі (розроблено автором)

Рисунок 4.20 – Зображення маленького демону у Т-позі (розроблено автором)

Рисунок 4.21 – Зображення огру у Т-позі (розроблено автором)

Рисунок 4.22 – Зображення розбійника у Т-позі (розроблено автором)

Розглядаючи префаби ворогів та боса, слід відзначити, які саме можливості та патерни поведінки відповідають кожному з них:

а) скелет:
   1) звичайний пошук гравця (PatrolingBehaviour);
   2) поведінка, за якою істота тримається на відстані від персонажу (KeepingDistanceChaseBehaviour);
   3) атака із киданням каміння (ThrowAttackBehaviour);

б) маленький демон:
   1) звичайна поведінка з патрулювання місцевості (PatrolingBehaviour);
   2) патерн переслідування головного герою до ближньої дистанції (ChaseBehaviour);
   3) атака укусом з близької відстані (BiteAttackBehaviour);

в) огр:
   1) звичайний патерн перевірки місцевості (PatrolingBehaviour);
   2) поведінка переслідування гравця у ближній дистанції (ChaseBehaviour);
   3) атака ударом у ближньому бої (PunchAttackBehaviour);
   4) атака жбурлянням каміння у гравця (ThrowAttackBehaviour);

г) розбійник:
   1) звичайна поведінка патрулювання початкової місцевості (PatrolingBehaviour);
   2) патерн з притримування певної дистанції між ним та гравцем (KeepingDistanceChaseBehaviour);
   3) здібність, яка призиває на поміч певних істот, які будуть воювати на його боці (SpawnAttackBehaviour);

д) бос огр:
   1) звичайна поведінка з пошуком гравця навколо початкової позиції (PatrolingBehaviour);
   2) патерн переслідування гравця у ближній дистанції (ChaseBehaviour);
   3) атака у ближньому бої (PunchAttackBehaviour);

4) атака кидком каміння у гравця (ThrowAttackBehaviour);
5) здібність, за допомогою якої, ворог призиває на свою сторону помічників (SpawnAttackBehaviour).

Розібравши перелік наявних патернів поведінок, різновидів ворогів та наборів їх вмінь, можна відзначити, що у певній мірі, відповідна вимога до жанру roguelike (згідно визначенню наведеному у підрозділі 1.1.1.2) щодо різноманіття ворогів та наповнення, є дотриманою. Це означає, що програмний модуль, що розробляється у рамках поточної кваліфікаційної роботи, має достатнє наповнення, як на поточний момент реалізації, та відповідає встановленим концептуальним вимогам до ігрового програмного застосунку, чим і постає поточний проєкт.

4.4 Розроблені ефекти стихійної системи

Відповідно до концептуального моделювання, що проводилось у розділі 1 та відповідно до геймдизайн документу (див. додаток В), у програмному модулі roguelike, передбачено стихійну систему, що складається з чотирьох основних елементів: електрична енергія, вогонь, лід, вода.

Відповідно до того, що програмний модуль, що розробляється, є грою та використовує можливості рушія Unity3D, відзначені вище стихії, повинні мати власні візуальні ефекти, з метою відрізнятись для гравця, та створювати необхідне враження відповідно. Для втілення поставленої задачі, було створено чотири візуальних ефекти (див. рис. 4.23 – 4.26), із використанням системи частинок («Particle system») [23, 24, 29], відповідно до обраних технологій, за допомогою модуля VFX Graph рушія Unity3D.

У доповнення до результату візуалізації ефектів, що були розроблені із використанням VFX Graph та Particle system, слід також розглянути саме візуальне програмування цих ефектів , на прикладі вогняного елементу (див. рис. 4.27).

На початку існування ефекту, система визначає кількість частинок, що також залежить від зовнішньої змінної «SpawnRate», що може бути змінена ззовні.

Після створення, VFX Graph переходить до етапу ініціалізації, у якому визначаються основні параметри частинок, такі як колір, положення швидкість та

напрям руху, часу існування тощо. Слід зауважити, що для визначення описаних параметрів, здебільшого, так само використовуються зовнішні змінні, що можуть бути відредаговані з програмного коду, тощо.

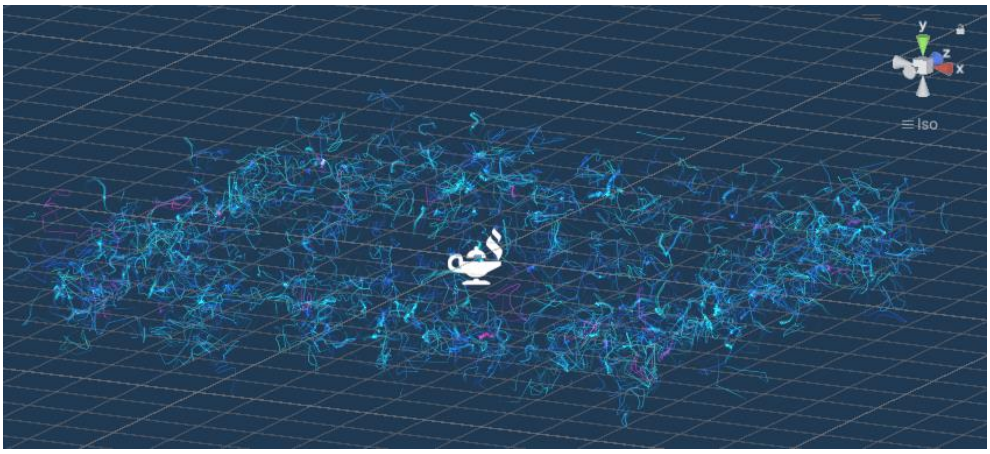


Рисунок 4.23 – Зображення VFX ефекту електричної енергії (розроблено автором)

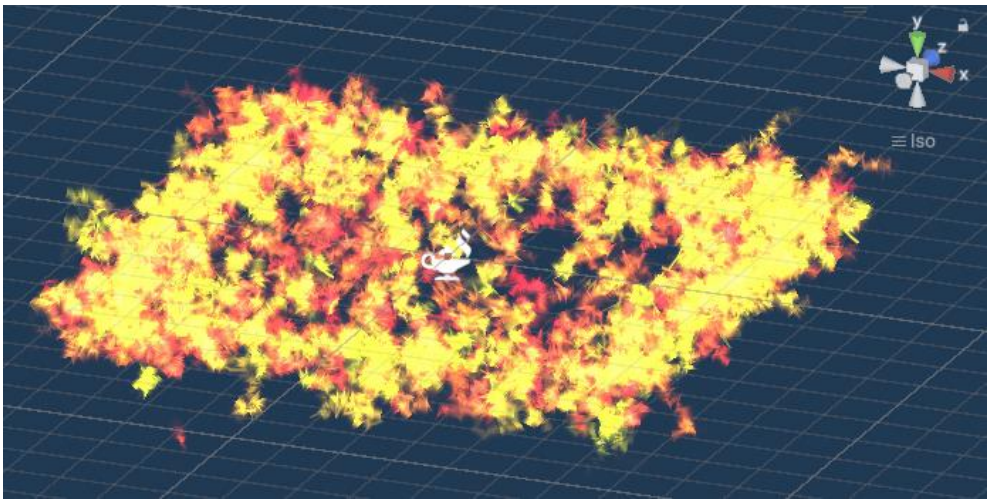


Рисунок 4.24 – Зображення VFX ефекту вогню(розроблено автором)

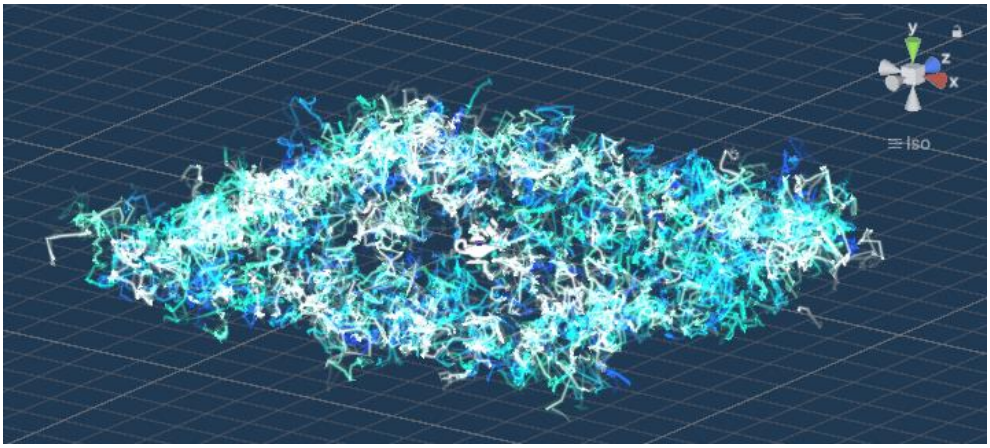


Рисунок 4.25 – Зображення VFX ефекту льодової аури (розроблено автором)

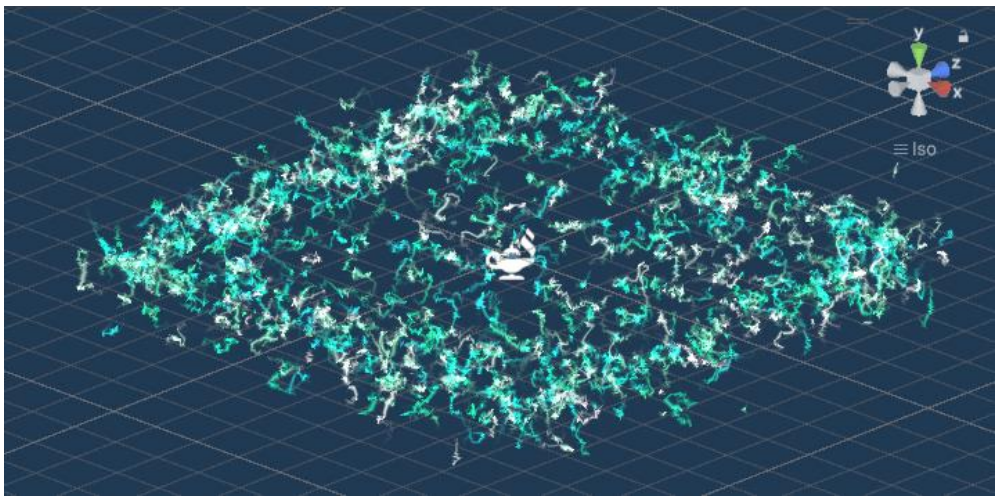


Рисунок 4.26 – Зображення VFX ефекту водяного елементу (розроблено автором)

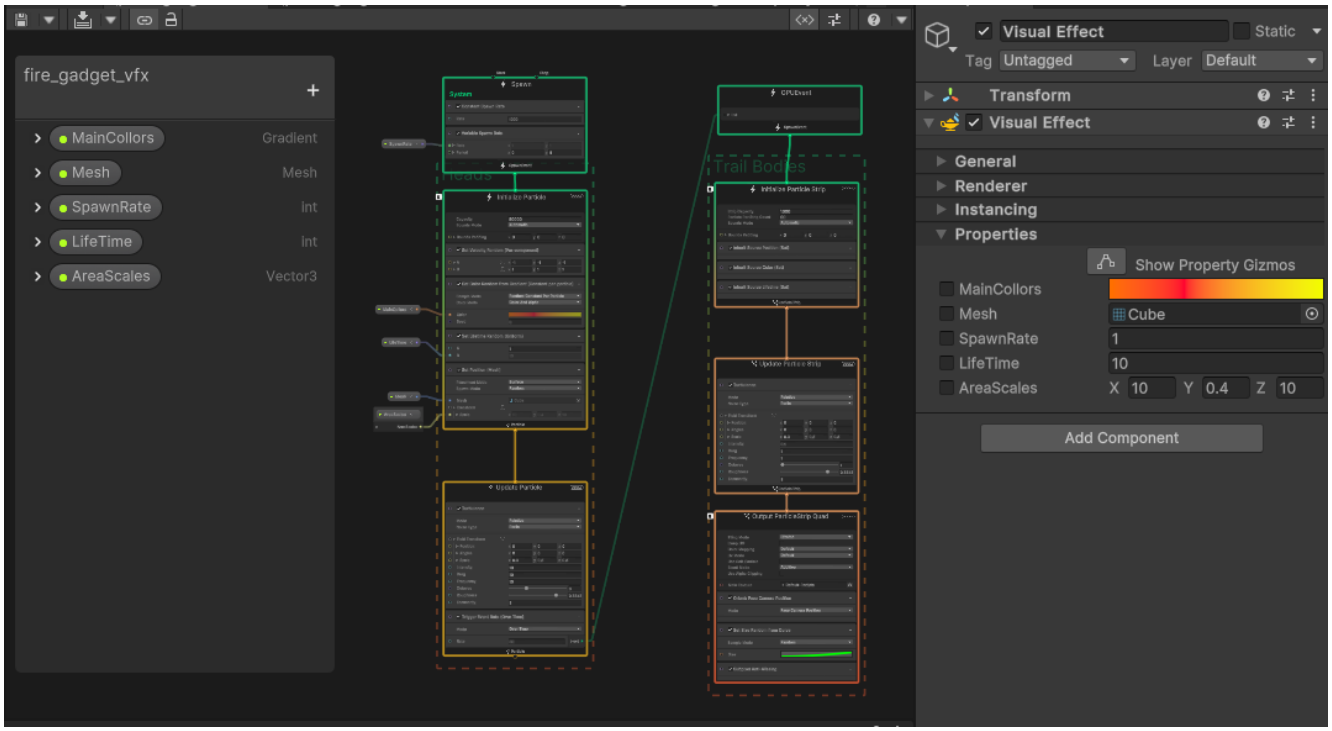


Рисунок 4.27 – Зображення VFX Graph-у вогняного елементу (розроблено автором)

Після етапу ініціалізації, визначається, яким чином буде змінюватися кожна окрема частинка на кожному кадрі. У демонстраційному ефектів, використовується компонент «Turbulence», що робить частинки хаотичними у своєму переміщені, створюючи ефект турбуленції, що й вказано у назві.

Далі, було додано компонент «GPU Event», який використовується для обробки взаємодій між частинками у реальному часі, з метою створення видимості

єдності усього ефекту, замість простого мерехтіння окремих кольористих точок на екрані гравця. Для впровадження зазначеної єдності, було додано компоненти «Initialize Particle Strip» та «Update Particle Strip» для того, аби вказати системі, що точки потрібно об’єднувати у стрічки. Відповідно до аналогічних вершин графу, без слова «Strip» у назві, що розглядалися вище, у зазначених фрагментах, визначається як ініціалізувати ефект та як його змінювати у кожному кадрі. Для завершення створення графічного ефекту, використовується вершина «Output Particle Strip Quad», що визначає як само візуалізовувати стрічки та самі частинки у екранному просторі гри

Результатом створення графічних ефектів до стихійної системи програмного модулю roguelike, проєкту «The fate of the liar» було реалізовано яскраві візуалізації чотирьох елементів (а саме лід, вогонь, вода та електрика), що відрізняються як за кольором, так і за загальним виглядом усього ефекту. Відзначимо, що завдяки зовнішнім змінним, дані компоненти системи є масштабованими та зручними у використанні.

4.5 Балансування наповнення програмного модулю roguelike

4.5.1 Модифікатори предметів та показники ворогів у залежності від рівня

Розглядаючи програмний модуль roguelike, а саме прийнятті рішення, щодо його реалізації, слід розглянути підхід до балансування наповнення цього компоненту ігрового застосунку, зокрема відношенню показників рівня гравця, до силових характеристик його ворогів та потенційного спорядження, що він носитиме.

Наперед слід відзначити, що рівень гравця, згідно загальній концепції гри та прийнятим рішенням, через одночасну прокачку відносин з багатьма расами (див. додаток В), рівень буде визначатись окремим від зазначеної системи взаємин. Прогрес гравця буде вираховуватись з середнього по усім доступним расам, що одночасно запобіжить занадто швидкому прогресу та залучить користувача досліджувати усе наявне наповнення гри. Притому зі збільшенням рівня, базові значення певних характеристик гравця будуть потроху зростати.

У свою чергу, загалом обладунки гравця дещо збільшують базовий показник характеристик, згідно розміру модифікатора, який базується на рівні предмету. Слід відзначити, що зазначений модифікатор є середнім значенням усього екіпірування.

Балансування зазначених предметів та характеристик відносно рівня наведено на рисунках нижче (див. рис.4.28 – 4.33).

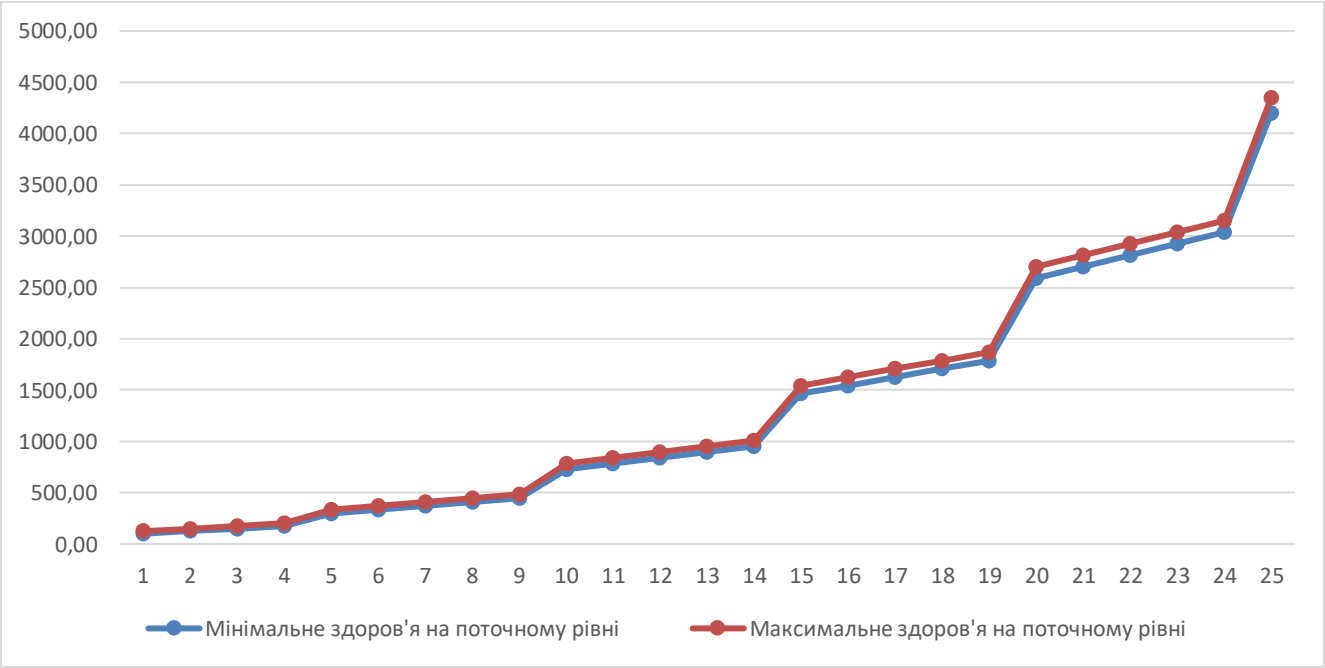


Рисунок 4.28 – Діаграма зміни здоров’я від рівня (розроблено автором)

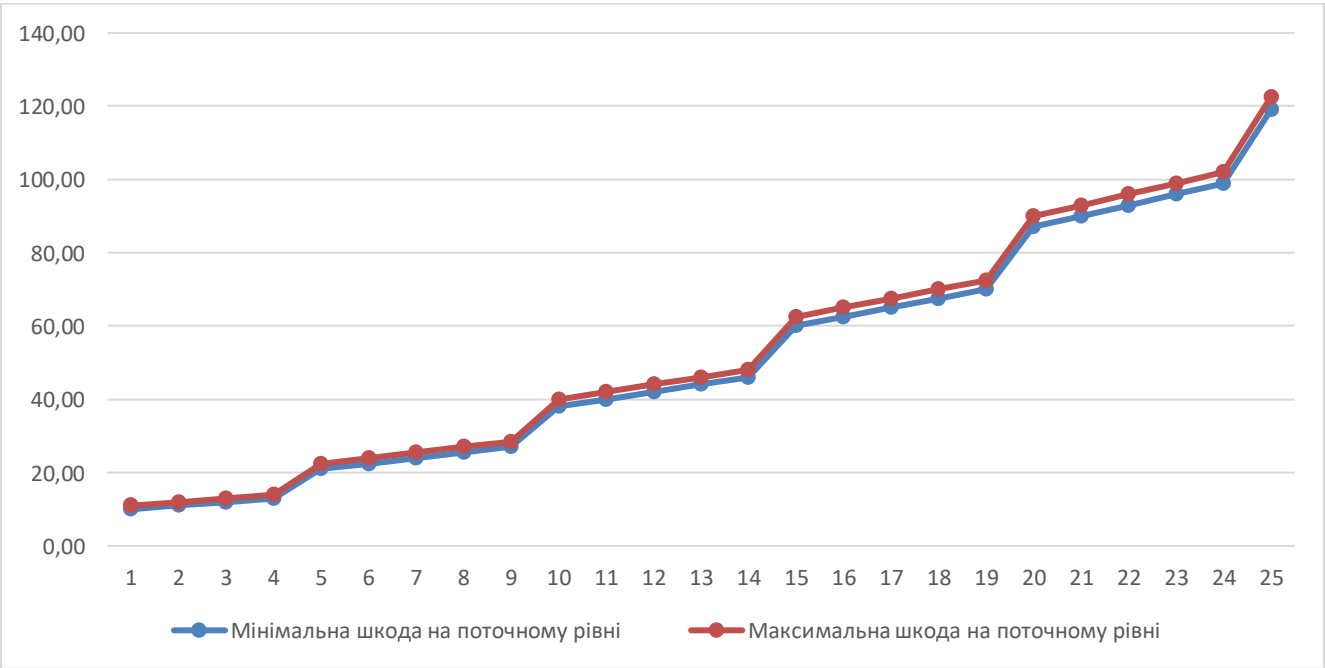


Рисунок 4.29 – Діаграма зміни шкоди від рівня (розроблено автором)

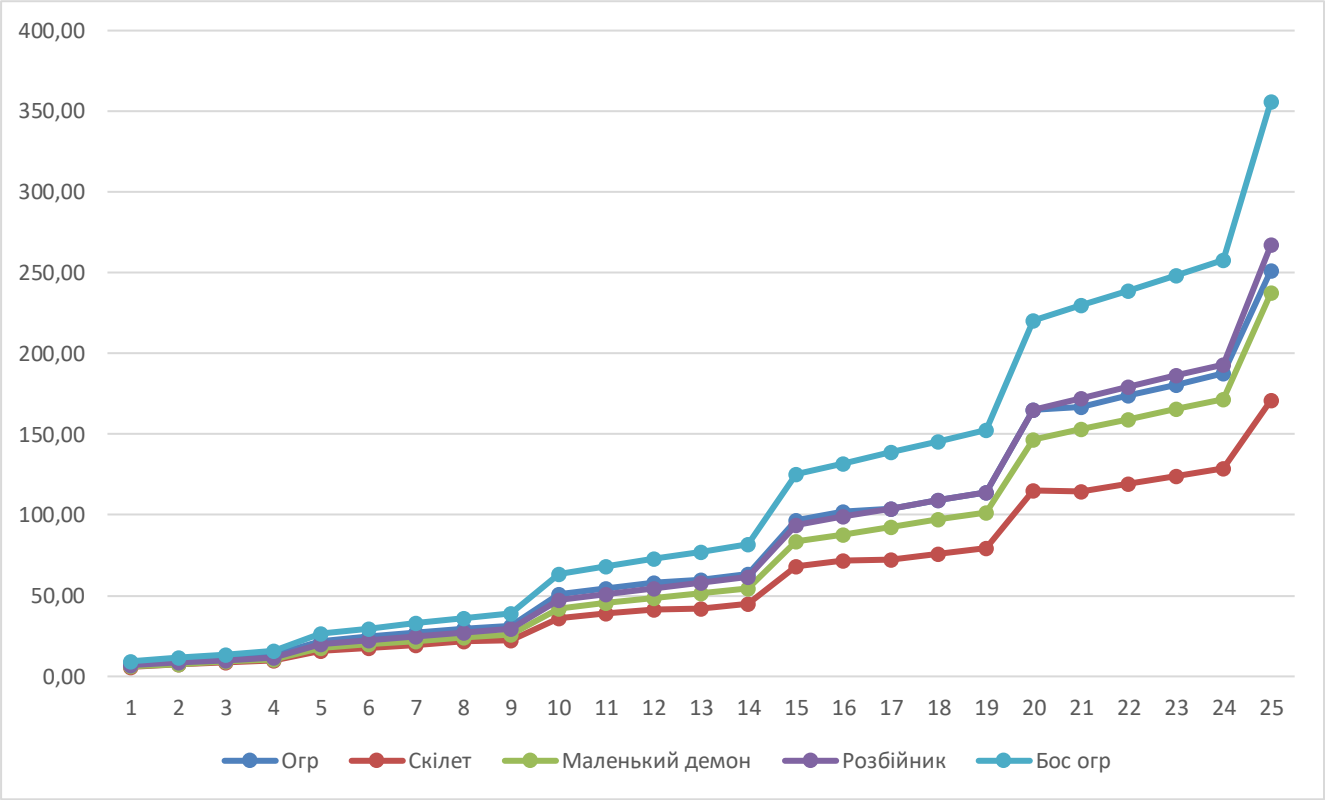


Рисунок 4.30 – Діаграма зміни шкоди ворогів від рівня (розроблено автором)

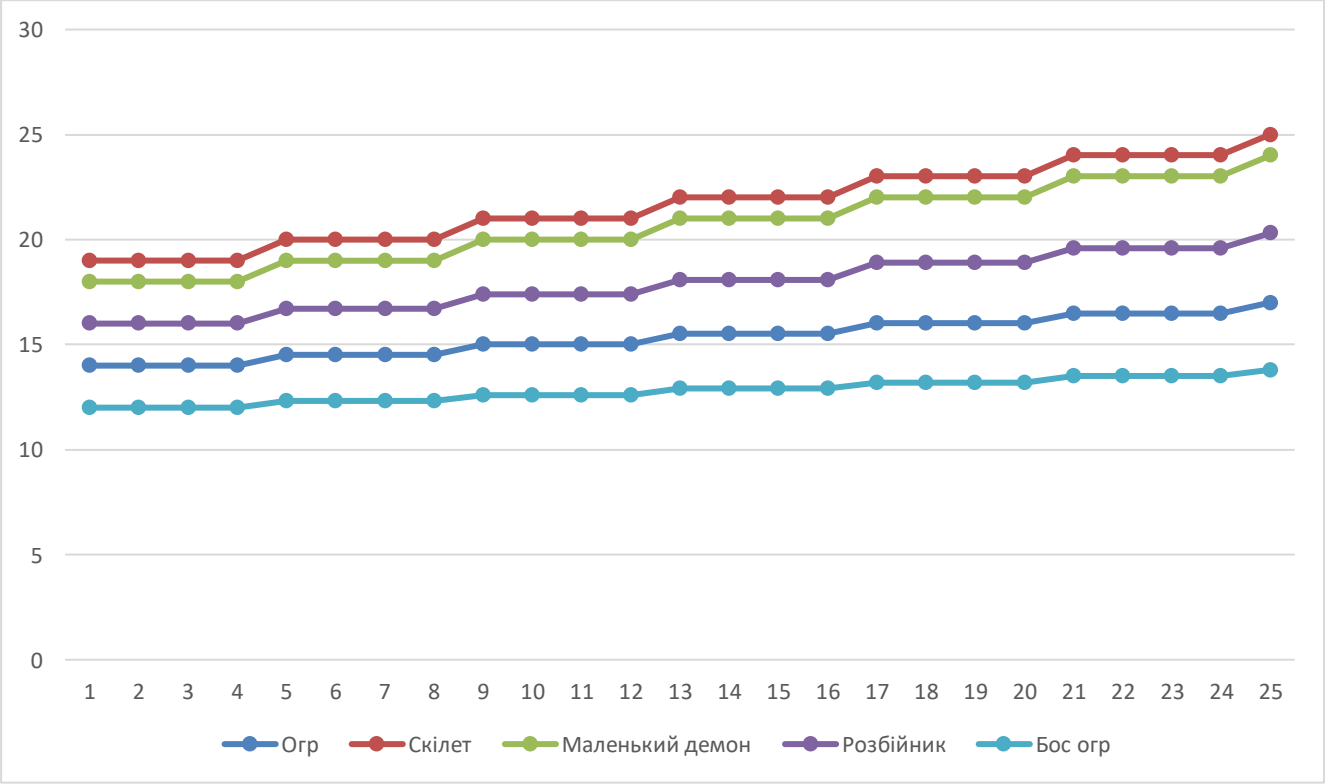


Рисунок 4.31 – Діаграма зміни кількості абстрактних атак ворогів, для вбивства гравця, від рівня (розроблено автором)

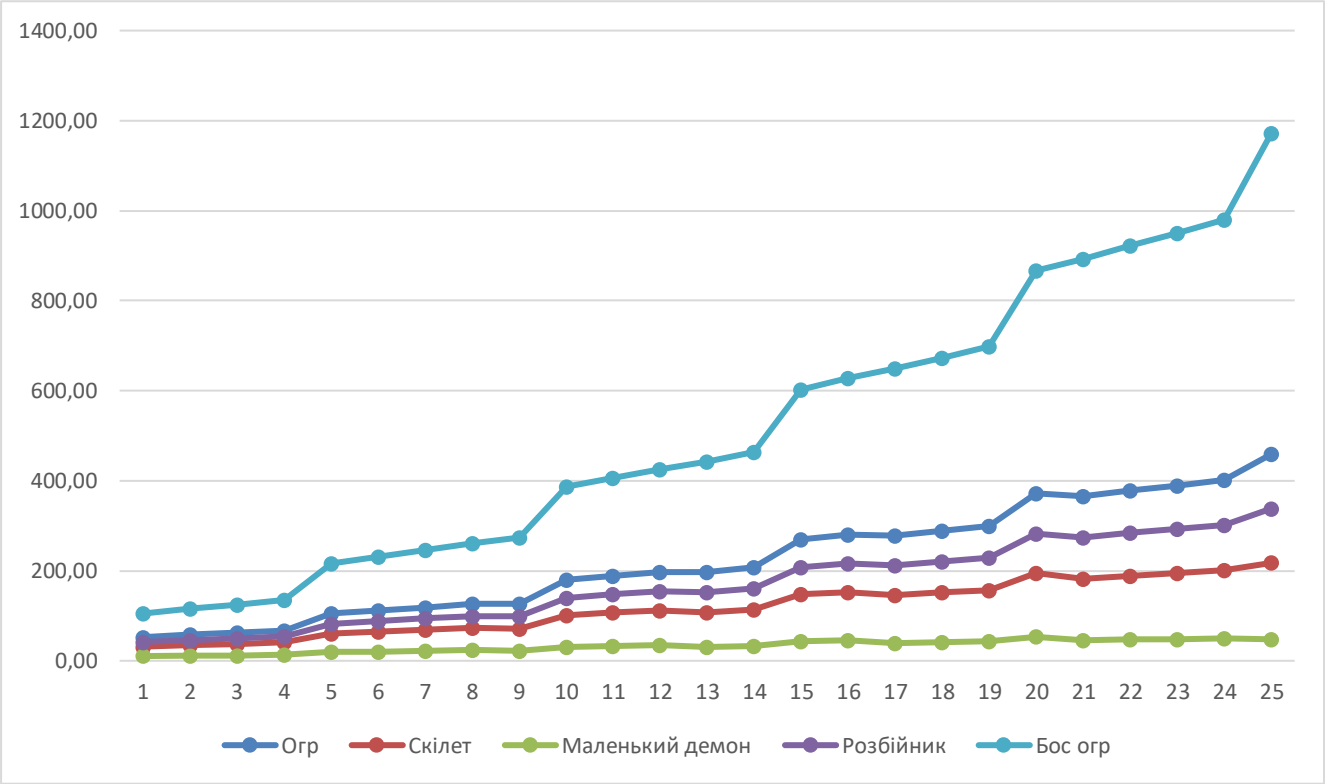


Рисунок 4.32 – Діаграма зміни запасу здоров'я ворогів від рівня (розроблено автором)

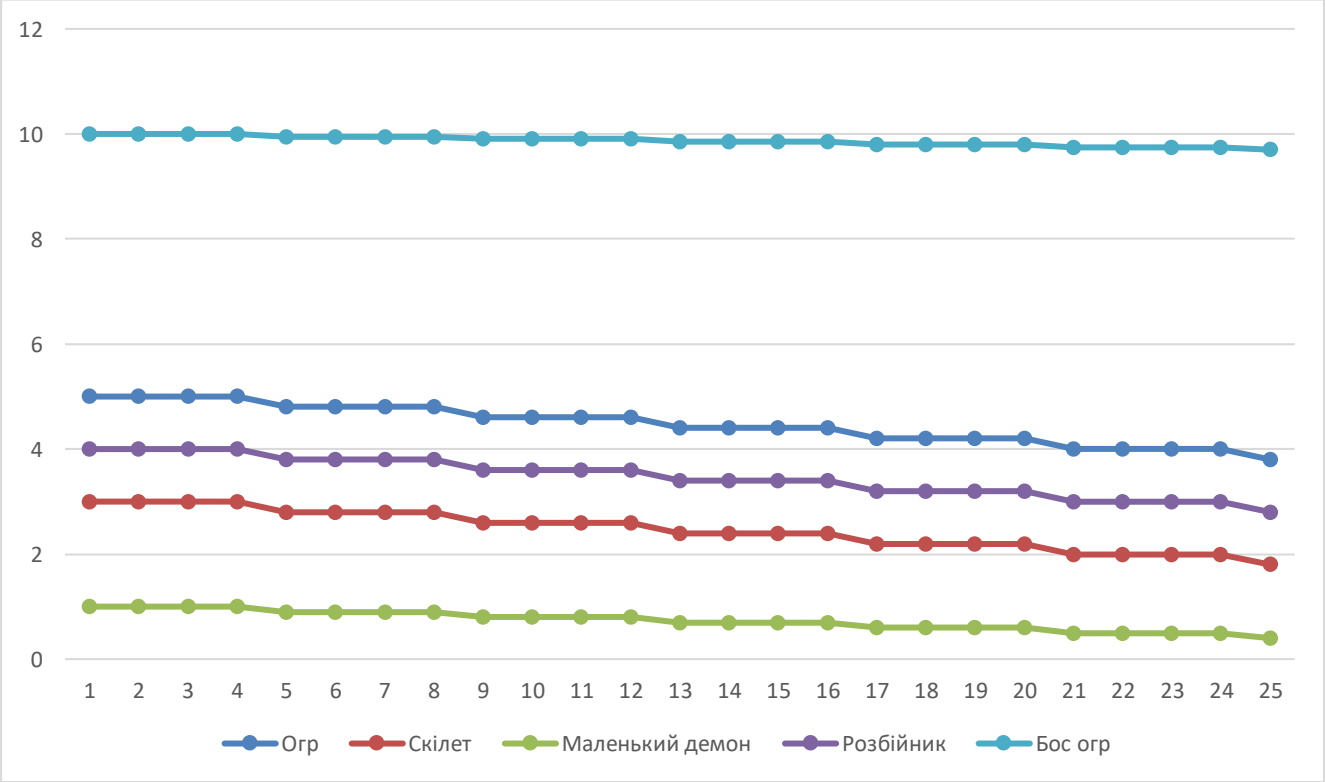


Рисунок 4.33 – Діаграма зміни кількості абстрактних атак гравця, необхіджних для вбивства ворога, від рівня (розроблено автором)

Слід відзначити, що кожні п’ять рівнів, гравець отримує покращення базових значень здоров’я та шкоди, оскільки саме тоді, відкриваються варіації більших підземель, із більшим вмістом ворогів відповідно. Також відмітимо, що показники ворогів базуються на значеннях гравця та теж враховують ускладнення підземель, тому мають певні зміни характеристик, так само, кожні п’ять рівнів. Повний документ балансування розташовано у додатках, (див додаток Е). Відмітимо, що решта характеристик персонажу гравця та ворогів, що не відповідають бойовому напрямку (останні будуть розглянуті далі), є тривіальними та не потребують детального балансування, оскільки впливають на результати лише деяких подій тощо.

4.5.2 Бойова система

Балансування вимагає також бойова система, а саме компоненти з атак ворогів, та стихійної системи, а саме параметрів відповідних елементів.

Розглядаючи розроблені атаки ворогів, а саме: простий удар, укус, дистанційна атака кидком каменю та призовну здібність, треба виокремити та збалансувати низку характеристик (див. рис. 4.34).

	Час удару (у секундах)	Час перерви (у секундах)	Дистанція для атаки (в умовних одиницях рушія)	Модифікатор шкоди	Відносний DPS єдиної атаки	Відносний DPS атаки при наявності інших бойових вмінь
Простий удар (Punch)	1	2	1,5	1,2	0,6	1,2
Укус (Bite)	1	1,2	0,7	1	0,83	1
Дистанційна атака кидком(Throw attack)	2	4	1000	0,8	0,2	0,4
Призовна атака (Summon attack)	4,2	40	100	0*	0*	0*

Рисунок 4.34 – Таблиця характеристик атакуючих здібностей ворогів (розроблено автором)

Такий розподіл характеристик обумовлений сенсом та смисловим представленням описаних атак. Згідно наведеній таблиці, укус має найбільш швидку атаку та середній розмір модифікатора шкоди, однак ворог для використання цієї здібності має бути максимально близько до гравця, що балансує зазначені позитивні фактори. Простий удар має більшу шкоду ніж укус у ситуації, коли у додаток є інші доступні вміння, але програє у випадку, в якому це єдиний атакуючий патерн. Зазначені слабкі та сильні сторони збалансовані більшою

дистанцією до гравця, що трохи збільшує виживаємість. Дистанційна атака має найнижчу, як загальну, так і у перерахуванні на секунди шкоду, однак це повністю компенсується можливістю атакувати ворога зі значної відстані. Призова здібність, у свою чергу, не має власної шкоди та повинна досить довго виконуватись, після чого перезаряджатись, у порівняні з рештою патернів, однак зазначена здібність має неабияку силу, а саме створювати собі помічників, що атакуватимуть замість відповідного ворога, що компенсує інші слабкі сторони вміння. Слід зазначити, що два останніх патерни мають певну логічну синергію, з поведінкою тримання відстані (KeepingDistanceChaseBehaviour), за рахунок чого, компенсуються слабкі сторони та підсилюються решта.

Окрім атак ворогів, варто розглянути також питання характеристик стихійної системи, як зазначалось вище (див. рис. 4.35).

	Вхідна шкода	Шкода у кожен тік гри	Час існування	Частота тіку стихії у грі	Загально нанесена шкода	Додатковий ефект
Вогонь	20	5	5	0,5	70	Підвищення вхідної шкоди
Вода	10	2	15	1	40	Підсилення шкоди інших стихій
Електрика	15	3	8	0,5	63	Періодичні переривання дій ворогів
Лід	5	1	10	0,2	55	Сповільнення ворогів

Рисунок 4.35 – Таблиця власних характеристик стихійної системи (розроблено автором)

Розглядаючи стихійні елементи, найсильнішим є вогонь, оскільки наносить найбільшу шкоду, однак швидше решти зникає, тож обмежує час на елементальні реакції та ефективність пасивного ефекту на ворогах. Вода є найслабшим елементом, як окремий гаджет, однак має найдовший час існування, що буде підтримувати стихійний ефект на ворогах та надавати додатковий час на реакції елементів. Електрика є контролюючим засобом, що наносить помірну шкоду ворогу та має шанс на переривання атак ворога, що допомагатиме гравцю під час битв, у свою чергу відносно короткий термін існування балансує цей елемент. Лід має найбільш усереднену шкоду, та корисний пасивний ефект, що сповільнює

ворогів, у той самий час має певні обмеження за часом, що робить цю стихію найбільш загальною та універсальною.

Також варто звернути увагу на балансування стихійних реакцій, що виникають при контакті двох різних гаджетів (див. рис. 4.36).

	Шкода від реакції				Ефекти від взаємодії			
	Вогонь	Вода	Електрика	Лід	Вогонь	Вода	Електрика	Лід
Вогонь	0	30	25	15	Оновлює час існування діючого гаджету	Прибирає обидві зони, завдаючі миттєвої шкоди	Прибирає зону вогню, завдаючі миттєвої шкоди	Розтоплює лід, створюючи воду, завдає миттєву шкоду
Вода	30	0	15	10	Прибирає обидві зони, завдаючі миттєвої шкоди	Оновлює час існування діючого гаджету	Розширює зону молній по воді, наносить миттєву та доаткову шкоду кожен тік	Вода моментально замерзає, перетворюючись у льодову зону, наносить миттєву шкоду
Електрика	25	15	0	0	Прибирає зону вогню, завдаючі миттєвої шкоди	Розширює зону молній по воді, наносить миттєву та доаткову шкоду кожен тік	Оновлює час існування діючого гаджету	Не взаємодіють
Лід	15	10	0	0	Розтоплює лід, створюючи воду, завдає миттєву шкоду	Вода моментально замерзає, перетворюючись у льодову зону, наносить миттєву шкоду	Не взаємодіють	Оновлює час існування діючого гаджету

Рисунок 4.36 – Таблиця характеристик стихійної системи від реакцій (розроблено автором)

За наведеними характеристиками, можна побачити, що вогонь знову таки має найбільшу шкоду, однак натомість прибирає обидва елемента, за винятком льоду, що стає водою. Вода ж, як і зазначалось, є допоміжним елементом, що реагує з усіма стихіями та виступає у ролі підтримки, що доповнює зазначений компонент бойової системи. Електрика є більш вузьконаправленною з точки зору стихійних взаємодій, оскільки ігнорує лід, однак натомість викликає вибух при взаємодії з вогнем, що наносить велику шкоду або підсилюється водою, що із урахуванням його пасивного ефекту балансує елемент. Лід виступає у ролі допоміжного інструменту, що наносить трохи шкоди, але натомість змінює умови бою, перетворюючись у воду при взаємодії з вогнем або навпаки морозить останню для поширення зони власного пасивного ефекту.

Підсумовуючи вище згадане, було описано балансні характеристики бойового компоненту ігрового застосунку, як з точки зору ворожих здібностей, так і зі стихійної системи. Описані основні характеристики, що впливають на баланс програмного модулю та відзначено усі сторони рівності кожного з елементів зазначених компонентів системи. Відмітимо, що решта бойової системи є більш опосередкованою й самодостатньою, тому загального балансування не вимагає.

5 ТЕСТУВАННЯ ПРОГРАМНОГО МОДУЛЮ

5.1 Розробка мапи думок тестування

Для тестування програмного модулю roguelike, що є частиною проєкту «The fate of the liar» та виступає у ролі кваліфікаційної роботи бакалавра, було створено тест-план документ, що включає у себе опис тестування усього продукту, з конкретизацією під зазначений компонент ігрового застосунку (див. додаток Г). Для формальної структуризації зазначеного документу та тестування загалом, було реалізовано діаграму «мапу думок» або Mind Map (оскільки діаграма є занадто великою, для підтримки читаності, вона була розділена на фрагменти, однак все ще є єдиним цілим) (див. рис. 5.1 – 5.6).

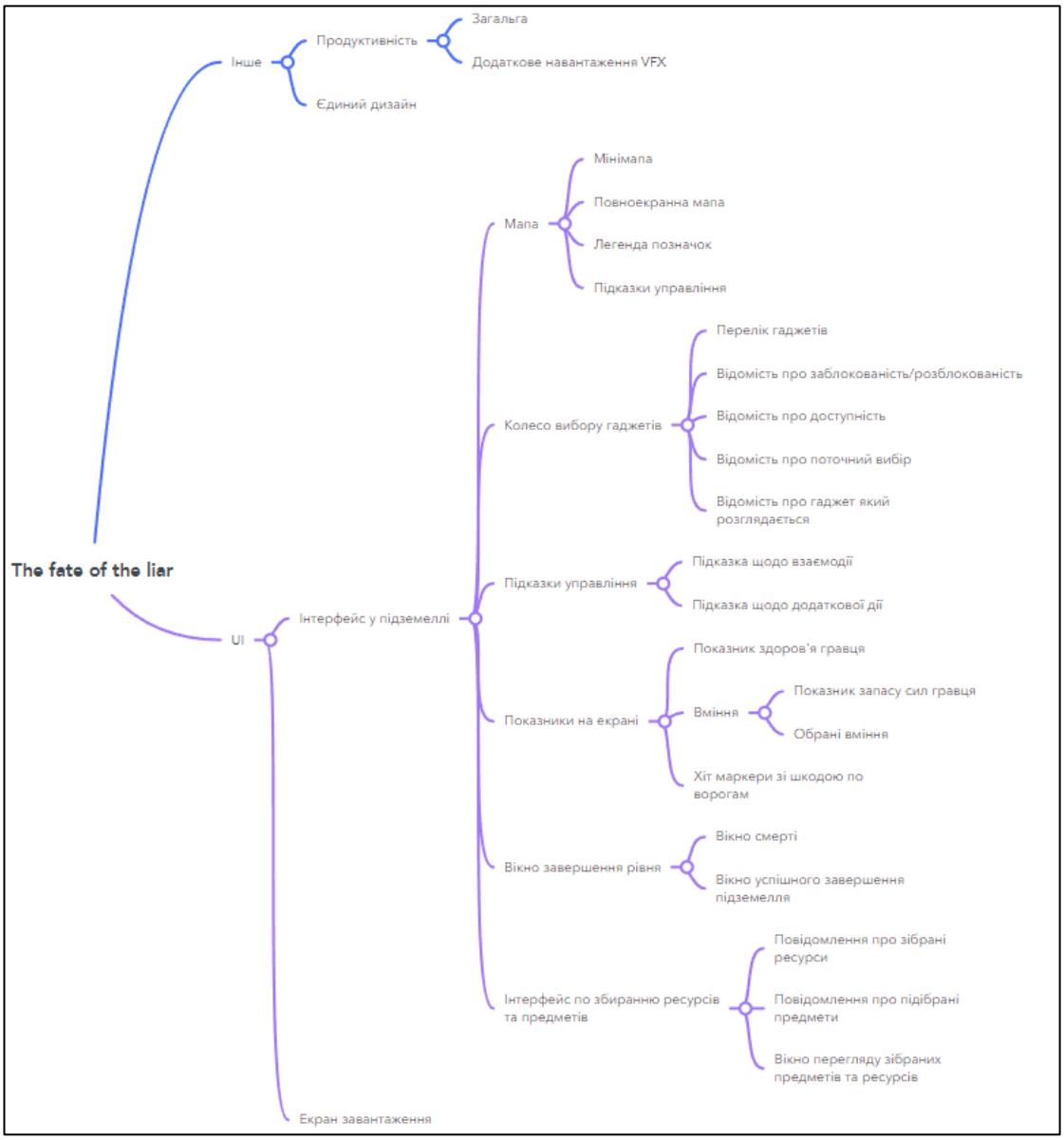


Рисунок 5.1 – Перший фрагмент мапи думок (розроблено автором)

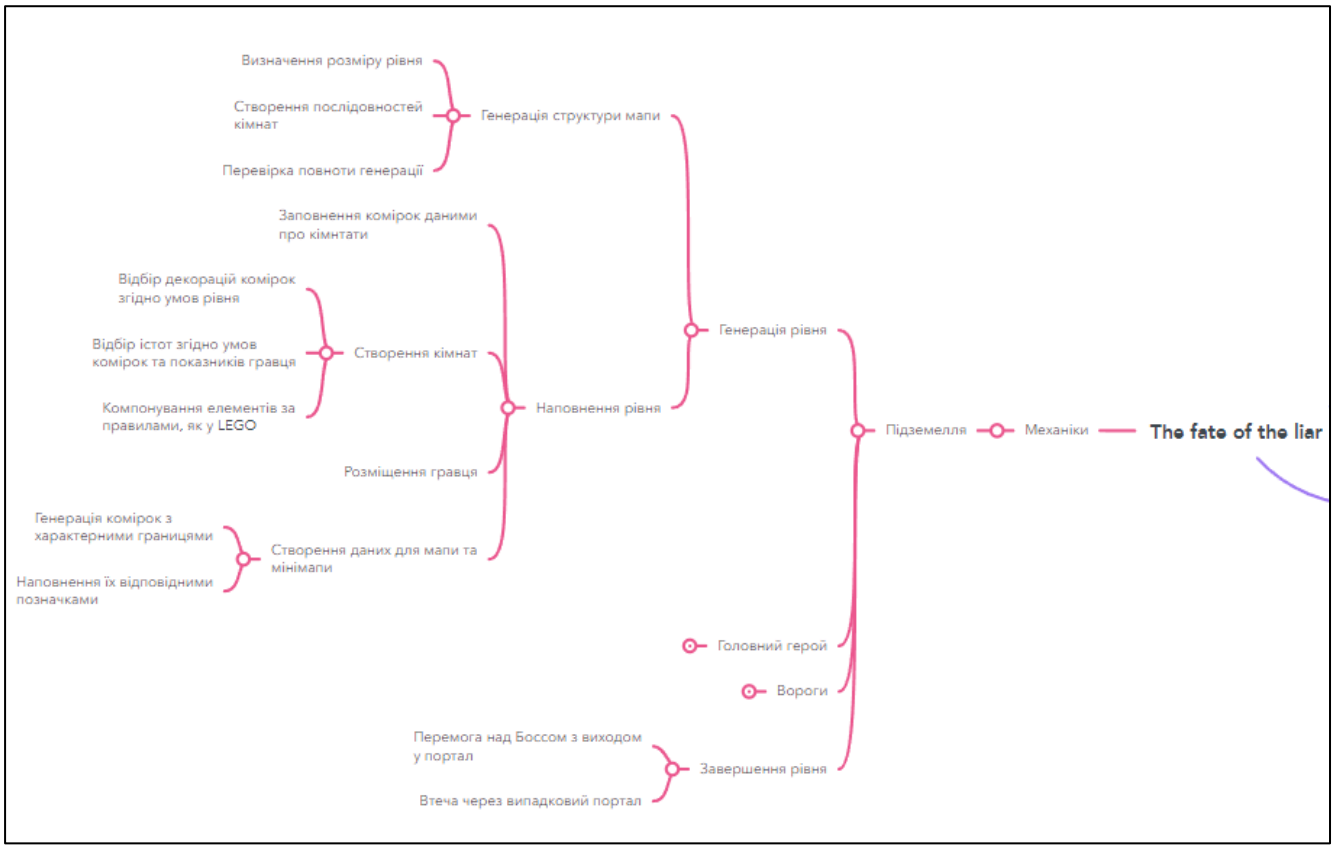


Рисунок 5.2 – Другий фрагмент мапи думок (розроблено автором)

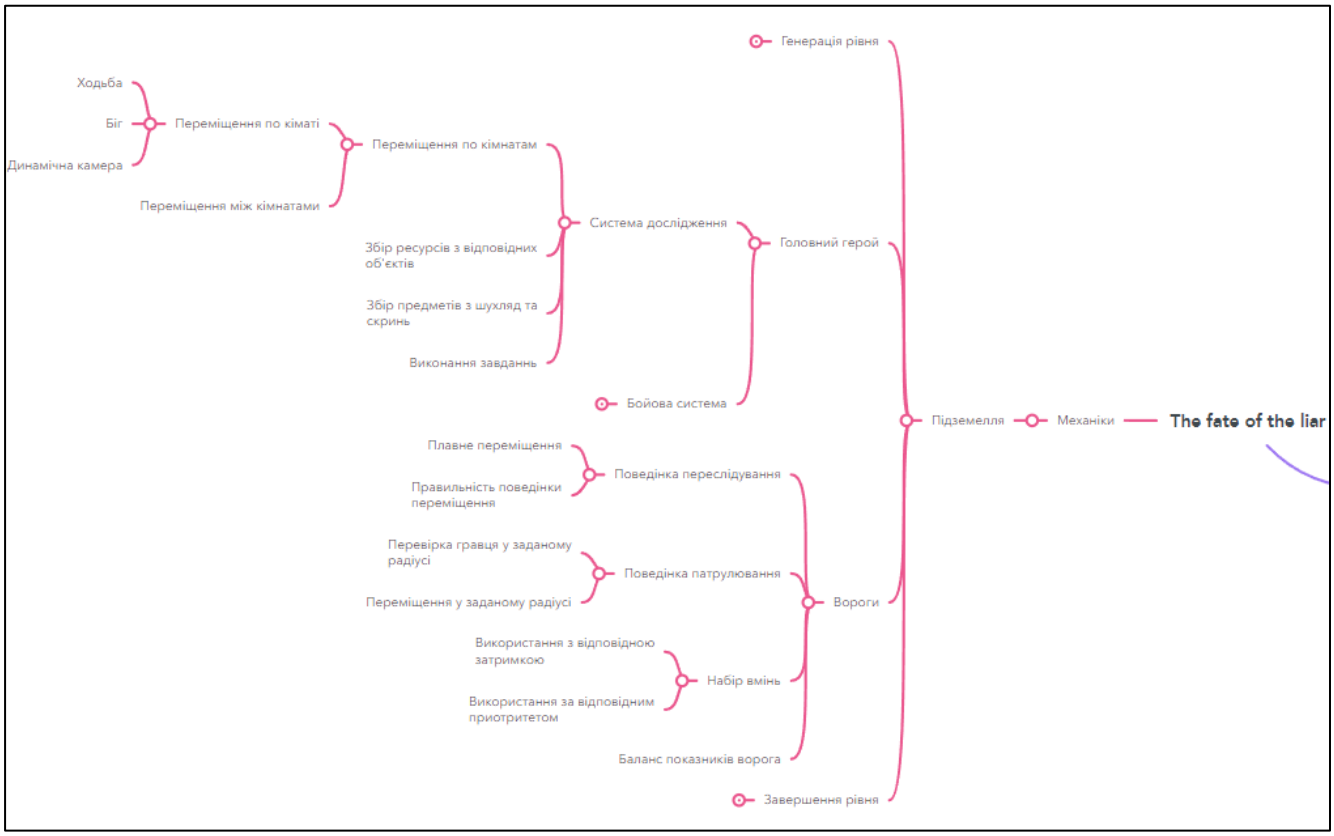


Рисунок 5.3 – Тертій фрагмент мапи думок (розроблено автором)

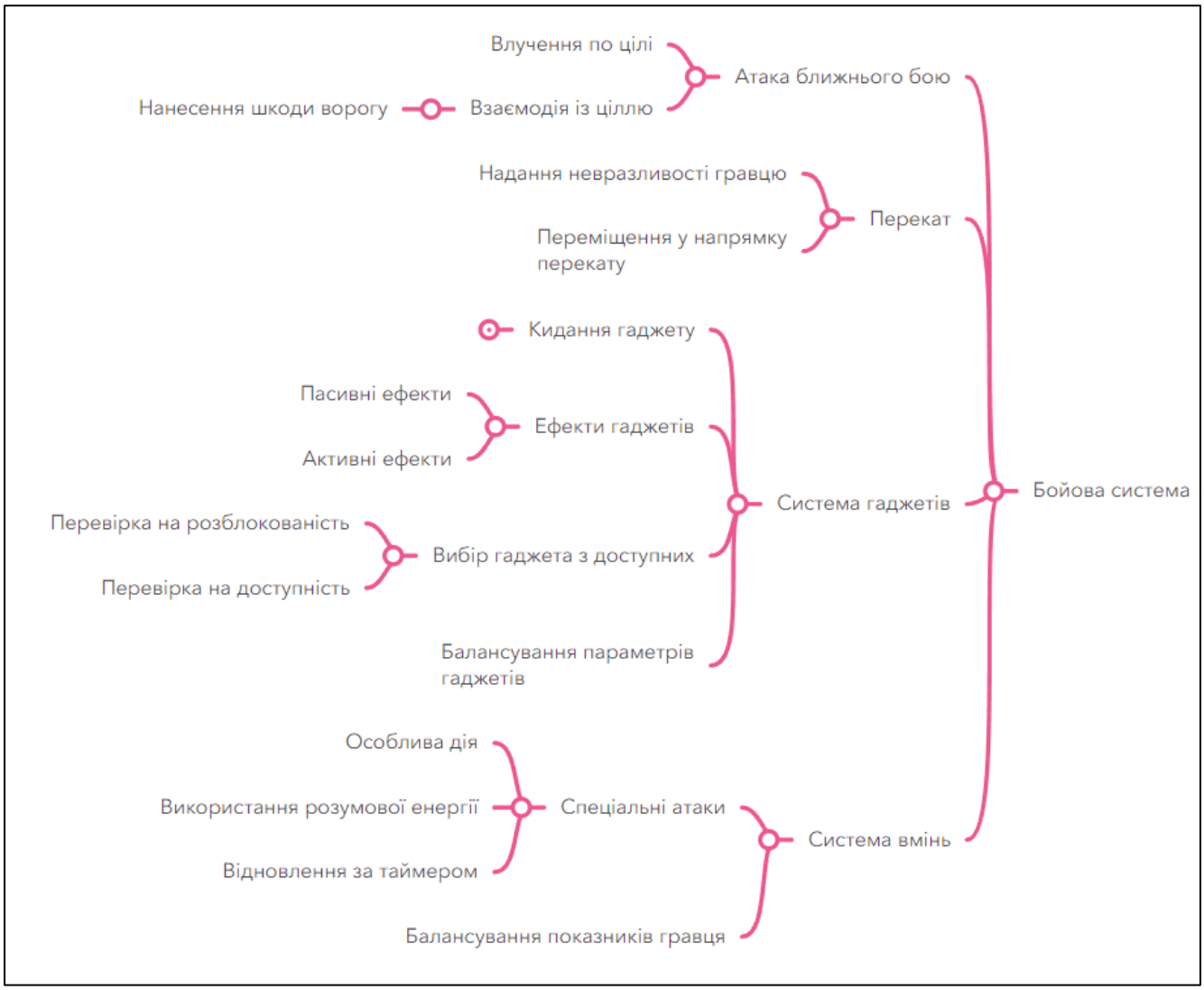


Рисунок 5.4 – Четвертий фрагмент мапи думок (розроблено автором)

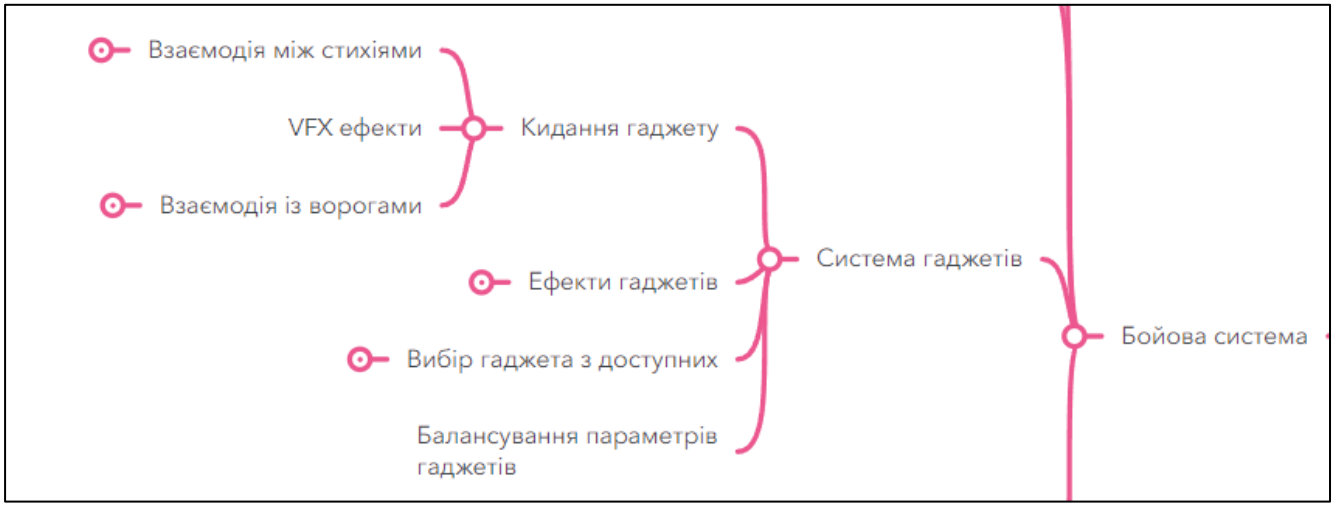


Рисунок 5.5 – П’ятий фрагмент мапи думок (розроблено автором)

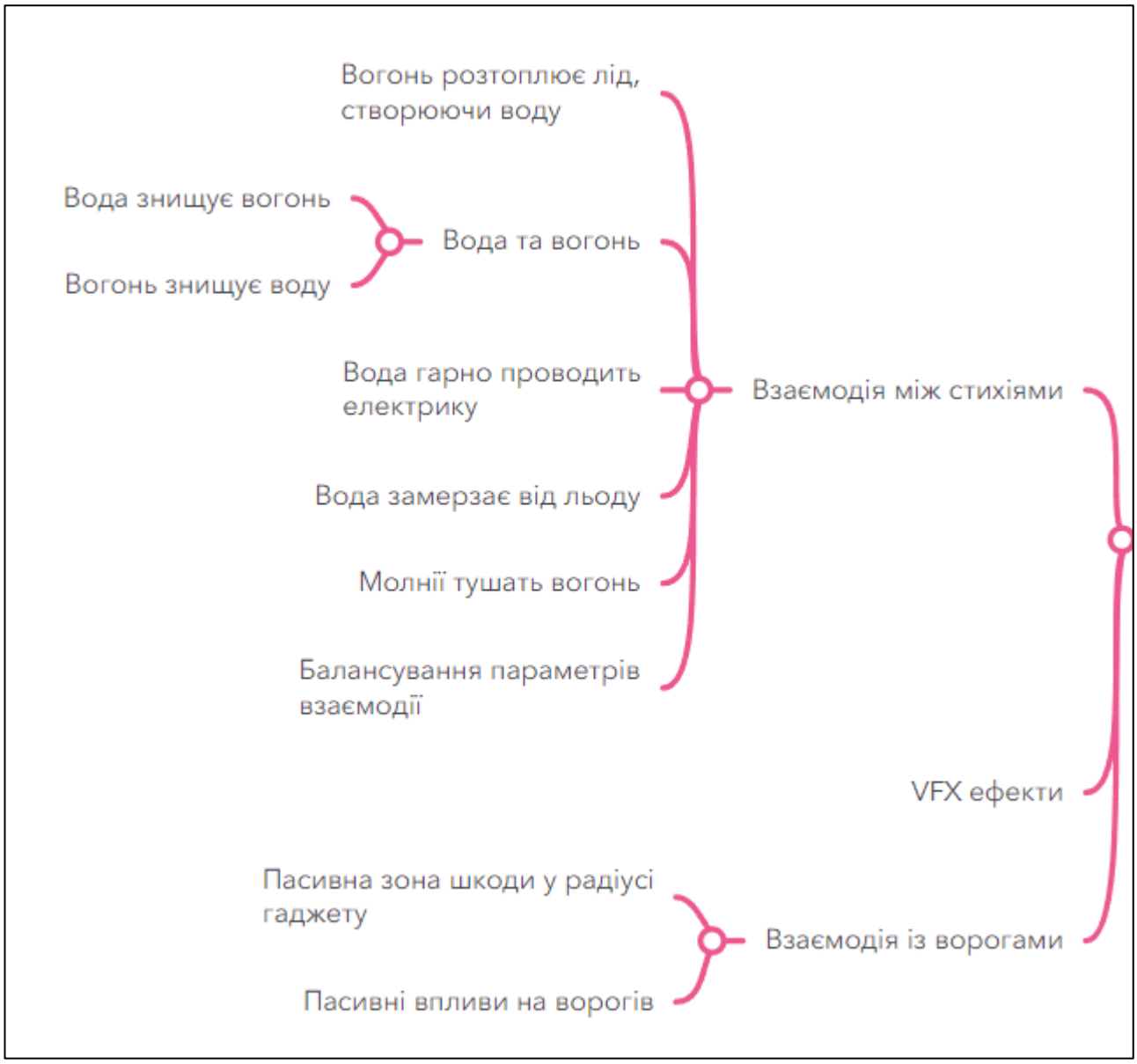


Рисунок 5.6 – Шостий фрагмент мапи думок (розроблено автором)

Створена діаграма «мапи думок» або Mind Map, спрямована на систематизацію та розділення програмного модулю roguelike та відповідних систем і механік на складові з чіткою ієрархію походження та залежності.

За даною діаграмою можна розділити напрямки тестування проєкту на основні категорії, а саме:

- основні механіки;
- інтерфейс користувача;
- інше, що включає у себе апаратні, дизайнові та подібні відокремленні елементи.

Слід також відзначити, що тестування, згідно розробленому тест-плану, включатиме у себе наступні типи перевірок:

а) Нефункціональне тестування:
   1) тестування продуктивності;
   2) тестування навантаженням;

б) Ручне тестування:
   1) тестування користувацького інтерфейсу
   2) дослідницьке тестування;
   3) тестування за сценаріями використання.

Результатом тестування має постати перелік знайдених помилок та невідповідностей проєкту, до спроєктованого рішення. Усі знайдені моменти мають бути вирішені та прибрані з фінальної версії проєкту «The fate of the liar» та програмного модулю roguelike, що являє собою предмет кваліфікаційної роботи.

5.2 Розробка тестових випадків

За створеним тест-планом та діаграмою «мапою думок» (Mind map), згідно обраним підходом до перевірки розробленого продукту, було проведено випробування програмного модулю roguelike, що виступає предметом кваліфікаційної роботи.

Результатом перевірки, став список з двадцяти п’яти помилок, що були відшукані у процесі тестування. Перелік усіх випадків наведений у таблиці нижче (див. табл. 5.1). Інформація щодо кожного артефакту розподіляється на:

- номер помилки;
- назва багу;
- короткий опис;
- компонент програми;
- серйозність, від S1 - блокуючої, до S5 - тривіальної;
- пріоритет, від P1 - високого, до P3 - низького;
- кроки відтворення;
- фактичний результат;

- очікуваний результат;
- стан багу.

Таблиця 5.1 – Список знайдених багів

Помилка №1	
Назва багу	Помилка генерації
Короткий опис	При генерації, було згенеровано менше кімнат, ніж визначалось на старті створення рівня
Компонент програми	Клас Generator
Серйозність	S4 Незначний (Minor)
Пріоритет	P3 Низький (Low)
Кроки відтворення	Запустити сцену RoguelikeLevel та, при відсутності помилки, натискати Backspace, до її виникнення, можливість повторної генерації рівня, доступно лише при запуску з рушія Unity3D
Фактичний результат	Створених кімнат менше ніж планувалось
Очікуваний результат	Створено рівень з правильною кількістю кімнат у ньому
Стан багу	Вирішено
Помилка №2	
Назва багу	Низька роздільна здатність зображення гаджета
Короткий опис	У колесі вибору гаджетів, «Ice Gadget» має низьку роздільну здатністю свого зображення
Компонент програми	Інтерфейс GadgetWheelScreen
Серйозність	S5 Тривіальний (Trivial)
Пріоритет	P3 Низький (Low)
Кроки відтворення	Запустити сцену RoguelikeLevel, та натиснути Q для відкриття колеса гаджетів
Фактичний результат	Зображення «Ice Gadget» має низьку роздільну здатність
Очікуваний результат	Всі зображення гаджетів мають однаково чіткі зображення

Продовження таблиці 5.1

Стан багу	Виправлено
Помилка №3	
Назва багу	Відновлення стану гаджета у інтерфейсі
Короткий опис	У колесі вибору гаджетів, при все ще активному гаджеті, при натисканні на довільне місце, активність іконки гаджета відновлюється до неактивного, хоча нічого не повинно відбутись
Компонент програми	Інтерфейс GadgetWheelScreen
Серйозність	S3 Значний (Major)
Пріоритет	P2 Середній (Medium)
Кроки відтворення	Запустити сцену RoguelikeLevel, та натиснути Q для відкриття колеса гаджетів, натиснути на один з доступних та натиснути на вільний простір поза кнопками, після чого при наведенні на будь який гаджет, буде відображення ніби ніякий гаджет не обрано
Фактичний результат	Відмітка активності гаджета на іконці зникає, хоча обрано все ще той самий девайс
Очікуваний результат	Нічого відбутись не повинно
Стан багу	Виправлено
Помилка №4	
Назва багу	Невідповідність гаджету у інтерфейсі
Короткий опис	Якщо змінити дані про обраний гаджет, у відповідному сервісі, не з кільця вибору гаджета, то обраний гаджет, згідно інтерфейсу не зміниться, але фактично, використовуватись буде вже правильний гаджет.
Компонент програми	Інтерфейс GadgetWheelScreen та сервіс GadgetService
Серйозність	S3 Значний (Major)
Пріоритет	P2 Середній (Medium)

Продовження таблиці 5.1

Кроки відтворення	Запустити сцену RoguelikeLevel, та натиснути Q для відкриття колеса гаджетів, обрати один з доступних гаджетів, після чого змінити поле CurrentGadgetIndex на інше значення, через інтерфейс рушія Unity3D, після чого перезавантажити рівень натискаючи Backspace (доступно лише при запуску з рушія Unity3D)
Фактичний результат	Обраний гаджет у інтерфейсі не змінився
Очікуваний результат	Обраний гаджет у інтерфейсі змінився
Стан багу	Виправлено
Помилка №5	
Назва багу	Постійно відкрите колесо гаджетів
Короткий опис	Якщо при відкритому вікні вибору гаджетів вивести з фокусу гру, після чого повернутись, то вікно все ще буде відображене, хоча відповідна клавіша не натиснута
Компонент програми	Інтерфейс GadgetWheelScreen та сервіс GadgetService
Серйозність	S5 Тривіальний (Trivial)
Пріоритет	P3 Низький (Low)
Кроки відтворення	Запустити сцену RoguelikeLevel, та натиснути Q для відкриття колеса гаджетів, натиснути на інше вікно у випадку віконного режиму гри, або натиснути win/alt+tab/win+tab тощо, для виведення гри з фокусу, після чого повернутись до гри
Фактичний результат	Після повернення фокусу, вікно гаджетів не закривається без повторного натискання Q
Очікуваний результат	Після повернення фокусу, вікно гаджетів само закриється
Стан багу	Виправлено
Помилка №6	

Продовження таблиці 5.1

Назва багу	Продовження переміщення гравця при відкритому вікні гаджетів
Короткий опис	При відкритті колеса вибору гаджетів під час переміщення, персонаж безкінечно продовжує переміщуватись, доки не закрити відповідний інтерфес
Компонент програми	Інтерфейс GadgetWheelScreen, сервіс GadgetService та клас PlayerInputController
Серйозність	S5 Тривіальний (Trivial)
Пріоритет	P3 Низький (Low)
Кроки відтворення	Запустити сцену RoguelikeLevel, почати йти у будь-якому напрямку та натиснути Q для відкриття колеса гаджетів, не відпускаючи кнопку переміщення
Фактичний результат	Персонаж продовжує переміщення за заданим напрямком
Очікуваний результат	Персонаж зупиниться
Стан багу	Виправлено
Помилка №7	
Назва багу	Текст за замовчуванням у вікні вибору гаджетів
Короткий опис	При відкритті вікна вибору гаджета, у полі для тексту, стосовно поточного обраного гаджету, відображається текст за замовчуванням, якого не має бути
Компонент програми	Інтерфейс GadgetWheelScreen, сервіс GadgetService
Серйозність	S5 Тривіальний (Trivial)
Пріоритет	P3 Низький (Low)
Кроки відтворення	Запустити сцену RoguelikeLevel, натиснути Q для відкриття колеса гаджетів
Фактичний результат	За замовчуванням буде текст «Gadget»
Очікуваний результат	Ніякого тексту не повинно бути без обраного гаджету
Стан багу	Виправлено

Продовження таблиці 5.1

Помилка №8	
Назва багу	Вибір заблокованого гаджета
Короткий опис	Не дивлячись на заблокованість гаджета, його можна обрати
Компонент програми	Інтерфейс GadgetWheelScreen, сервіс GadgetService
Серйозність	S3 Значний (Major)
Пріоритет	P1 Високий (High)
Кроки відтворення	Запустити сцену RoguelikeLevel, заблокувати гаджети у сервісі GadgetService, перезавантажити рівень та натиснути Q для відкриття колеса гаджетів та обрати гаджет, наведення на який, має сірий текст (заблокований)
Фактичний результат	Можна обрати навіть заблокований гаджет
Очікуваний результат	Неможливість та заборона обрати заблокований гаджет
Стан багу	Виправлено
Помилка №9	
Назва багу	Активність іконки вибору заблокованого гаджету
Короткий опис	Зображення гаджетів у відповідному вікні не відображають стан їх доступності
Компонент програми	Інтерфейс GadgetWheelScreen, сервіс GadgetService
Серйозність	S4 Незначний (Minor)
Пріоритет	P2 Середній (Medium)
Кроки відтворення	Запустити сцену RoguelikeLevel, заблокувати гаджети у сервісі GadgetService, натиснути Q для відкриття колеса гаджетів та навістись на гаджет, наведення на який, має сірий текст (заблокований).
Фактичний результат	Кнопка є активною та реагує на наведення й натискання
Очікуваний результат	Кнопка має бути неактивною

Продовження таблиці 5.1

Стан багу	Виправлено
Помилка №10	
Назва багу	Гаджет за замовчуванням
Короткий опис	За замовчуванням визначено обраний гаджет, хоча його не було обрано, та при його недоступності, все ще є можливість його використовувати, також хоч обрано початковий гаджет, це не відображається у колесі вибору гаджета
Компонент програми	Інтерфейс GadgetWheelScreen, сервіс GadgetService
Серйозність	S3 Значний (Major)
Пріоритет	P1 Високий (High)
Кроки відтворення	Запустити сцену RoguelikeLevel, (гаджет обрано за замовчуванням), при натисканні на Q, у колесі вибору гаджета не відображається поточний стан, а при натисканні на кнопку кидання пристрою, буде використано гаджет за замовчуванням, навіть якщо він заблокований
Фактичний результат	За замовчуванням обрано гаджет
Очікуваний результат	За замовчуванням нічого не обрано
Стан багу	Виправлено
Помилка №11	
Назва багу	Вибір гаджета поверх повноекранної мапи
Короткий опис	Можливість відкрити вікно вибору гаджетів при відкритій повноекранної мапі
Компонент програми	Інтерфейс GadgetWheelScreen, сервіс GadgetService, інтерфейс MapScreen, класи GadgetWheelInputController та MapInputController
Серйозність	S4 Незначний (Minor)

Продовження таблиці 5.1

Пріоритет	P2 Середній (Medium)
Кроки відтворення	Запустити сцену RoguelikeLevel, натиснути M для відкриття мапи на весь екран, натиснути Q для відкриття колеса вибору гаджетів
Фактичний результат	Відкриття вікна вибору гаджетів
Очікуваний результат	Неможливість відкриття вікна вибору гаджетів
Стан багу	Виправлено
Помилка №12	
Назва багу	Назва гаджета після наведення залишається
Короткий опис	Якщо навістись на гаджет але не обирати його, закрити колесо вибору, відвести курсор у інше місце та заново відкрити колесо, текст залишиться тим самим
Компонент програми	Інтерфейс GadgetWheelScreen, сервіс GadgetService
Серйозність	S5 Тривіальний (Trivial)
Пріоритет	P3 Низький (Low)
Кроки відтворення	Запустити сцену RoguelikeLevel, натиснути Q для відкриття колеса вибору гаджетів, навістись на гаджет, відпустити Q для закриття колеса, відвести курсор, натиснути Q
Фактичний результат	Текст залишився
Очікуваний результат	Текст відновився
Стан багу	Виправлено
Помилка №13	
Назва багу	Нескінченна анімація ходьби на місці
Короткий опис	При використанні перевороту та відкритті вікна вибору гаджета до його завершення, після того як персонаж завершить анімацію, запуститься стан, в якому він буде крокувати на місці

Продовження таблиці 5.1

Компонент програми	Клас PlayerInputController
Серйозність	S5 Тривіальний (Trivial)
Пріоритет	P3 Низький (Low)
Кроки відтворення	Запустити сцену RoguelikeLevel, натиснути Space для перекату, натиснути Q для відкриття вікна вибору гаджетів в момент анімації, дочекатись її завершення
Фактичний результат	Персонаж крокує на місці
Очікуваний результат	Персонаж стоїть на місці
Стан багу	Виправлено
Помилка №14	
Назва багу	Неправильна траєкторія польоту снаряда
Короткий опис	Снаряд атаки кидка летить точно у гравця, без врахування положення атакуючого та гравця відносно нього
Компонент програми	Класи EnemyController та ThrowAttackBehaviour
Серйозність	S5 Тривіальний (Trivial)
Пріоритет	P3 Низький (Low)
Кроки відтворення	Запустити сцену RoguelikeLevel, пройти до кімнати з ворогом, що може кидати каменюки, наприклад огр, після чого почати кружляти навколо нього, доки він не кине камінь, знаходячись в нього за спиною
Фактичний результат	Снаряд летить прямо у гравця, навіть крізь атакуючого
Очікуваний результат	Снаряд летить вперед від атакуючого
Стан багу	Виправлено
Помилка №15	
Назва багу	Переміщення хіт маркерів
Короткий опис	При нанесенні ушкоджень ворогам, гравцю демонструється хіт маркер із відображенням нанесеної шкоди, але маркер переміщається разом із цілю

Продовження таблиці 5.1

Компонент програми	Клас EnemyController та префаб HitMarker
Серйозність	S3 Значний (Major)
Пріоритет	P2 Середній (Medium)
Кроки відтворення	Запустити сцену RoguelikeLevel, пройти до кімнати з ворогом, після чого почати атакувати та кружляти навколо ворогів, поки хіт маркери не почнуть переміщуватись
Фактичний результат	Хіт маркери переміщуються відносно точки появи, за виключенням анімації
Очікуваний результат	Хіт маркери не переміщуються відносно точки появи, за виключенням анімації
Стан багу	Виправлено
Помилка №16	
Назва багу	Вихід за границі рівня
Короткий опис	Деякі вороги, з атаками ближнього бою, можуть виштовхнути героя через стіну тощо, за границі рівня
Компонент програми	Префаби гравця – Player, компонентів рівня та ворогів
Серйозність	S2 Критичний (Critical)
Пріоритет	P1 Високий (High)
Кроки відтворення	Запустити сцену RoguelikeLevel, пройти до кімнати з ворогом, після чого підійти до стіни, ведучи ворога за собою, поки він не проштовхне головного героя за стіну
Фактичний результат	Ворог проштовхнув гравця за границі рівня
Очікуваний результат	Гравець не може проштовхуватись крізь об’єкти
Стан багу	Виправлено
Помилка №17	
Назва багу	Камера гравця не обмежується кімнатою

Продовження таблиці 5.1

Короткий опис	Підійшовши до границі кімнати - стінки, можна зазирнути за її край
Компонент програми	Префаб гравця - Player
Серйозність	S3 Значний (Major)
Пріоритет	P2 Середній (Medium)
Кроки відтворення	Запустити сцену RoguelikeLevel та підійти до однієї зі стінок (особливо видно на прикладі нижньої частини кімнати)
Фактичний результат	Камера статична відносно персонажу та може вийти за стіни кімнати
Очікуваний результат	Камера динамічно переміщується за персонажем, залишаючись у кімнаті
Стан багу	Виправлено
Помилка №18	
Назва багу	Вирівнювання тексту у повідомленні про зібранні ресурси
Короткий опис	У повідомленні про зібранні ресурси, текст кількості та назви відображаються з автоматичним перенесенням, та може привести до не дуже зрозумілого повідомлення або вилазити за екран
Компонент програми	Префаб ResourceMessage
Серйозність	S5 Тривіальний (Trivial)
Пріоритет	P3 Низький (Low)
Кроки відтворення	Запустити сцену RoguelikeLevel та дійти до кімнати із ресурсами, зібрати ресурси з відповідного об’єкту, шляхом взаємодії з ним, після чого повідомлення буде отримано

Продовження таблиці 5.1

Фактичний результат	Текст може вилазити за границі екрану та погано розташовувати текст у своїх клітинах
Очікуваний результат	Текст повністю знаходиться у границях екрану та має зрозуміле перенесення
Стан багу	Виправлено
Помилка №19	
Назва багу	Помилка напрямку перекату
Короткий опис	При заданні напрямку перекату через положення курсору відносно гравця, певними напрямками, переворот робиться у помилкову сторону
Компонент програми	Префаб гравця – Player, зокрема клас PlayerInputController
Серйозність	S2 Критичний (Critical)
Пріоритет	P1 Високий (High)
Кроки відтворення	Запустити сцену RoguelikeLevel, та почати робити перевороти, змінюючи положення курсору відносно гравця, вказуючи йому нові напрямки
Фактичний результат	При заданні напрямку перекату через положення курсору відносно гравця, певними напрямками, переворот робиться у помилкову сторону
Очікуваний результат	За всіма напрямками перекат є однаковим
Стан багу	Виправлено
Помилка №20	
Назва багу	Помилкове вирівнювання зібраних ресурсів
Короткий опис	У вікні зібраних ресурсів на рівні, ресурси складаються у рядках, по десять комірок, однак, якщо у рядку лише 1 предмет, відступ комірки від лівої границі екрану відрізняється

Продовження таблиці 5.1

Компонент програми	Інтерфейс LootTabScreen
Серйозність	S5 Тривіальний (Trivial)
Пріоритет	P3 Низький (Low)
Кроки відтворення	Запустити сцену RoguelikeLevel, зібрати 1 або n*10 + 1 типів ресурсів або предметів для отримання неправильного вирівнювання
Фактичний результат	Якщо у рядку лише 1 предмет, відступ комірки від лівої границі екрану відрізняється
Очікуваний результат	Усі рядки мають однаковий відступ
Стан багу	Виправлено
Помилка №21	
Назва багу	Помилковий ліміт на відображення зібраних ресурсів
Короткий опис	Після збору 11 предмету або ресурсу, вікно зібраних ресурсів відображає нові типи зібраних ресурсів та предметів у наступному рядку, тобто 11 комірка є єдиною у 2 рядку
Компонент програми	Інтерфейс LootTabScreen
Серйозність	S2 Критичний (Critical)
Пріоритет	P1 Високий (High)
Кроки відтворення	Запустити сцену RoguelikeLevel, зібрати 12+ ресурсів або предметів
Фактичний результат	Правильно відображається лише перший рядок на 10 елементів: предметів або ресурсів, 11 елемент є помилковим і так далі
Очікуваний результат	Всі зібрані ресурси та предмети будуть відображені у рядках по 10 елементів
Стан багу	Виправлено
Помилка №22	

Продовження таблиці 5.1

Назва багу	Помилка мапи шляхів
Короткий опис	При рестарті рівня після смерті або примусового перезапуску рівня, при роботі у Unity3D, мапа шляхів не генерується для стартової кімнати, а залишається той, що був, та об'єднується з навігаційним мешем поточної кімнати, через що, штучний інтелект ворогів не може проходити через відповідні зони рівня, хоча вони й пусті (залишається лише на однин перезапуск рівня)
Компонент програми	Клас RoomMovementService
Серйозність	S3 Значний (Major)
Пріоритет	P1 Високий (High)
Кроки відтворення	Запустити сцену RoguelikeLevel, дійти до кімнати зі статичними об'єктами, після чого померти або перезапустити рівень через кнопку Backspace, знаходячись у рушії Unity3D
Фактичний результат	NavMesh з дірками від об'єктів залишився
Очікуваний результат	Було згенеровано новий, правильний, NavMesh
Стан багу	Виправлено
Помилка №23	
Назва багу	Помилковий напрямок переміщення ворога при переслідуванні
Короткий опис	При швидкому переміщенні кругами навколо ворога, він перестає встигати повертатись у сторону гравця та починає переміщатися спиною
Компонент програми	Класи EnemyController та ChaseBehaviour
Серйозність	S4 Незначний (Minor)
Пріоритет	P3 Низький (Low)

Продовження таблиці 5.1

Кроки відтворення	Запустити сцену RoguelikeLevel, дійти до кімнати із ворогами та почати бігати колами навколо ворога поки він не буде переслідувати героя спиною вперед
Фактичний результат	Вороги можуть переслідувати гравця спиною вперед
Очікуваний результат	Вороги ходять переважно передом
Стан багу	Виправлено
Помилка №24	
Назва багу	Баг з картою
Короткий опис	При повноекранному форматі карти, гравець може пересуватись, але при спробі перетягнути карту, разом з цим, гравець виконує атаку та кидає гаджет, хоча не повинен
Компонент програми	Інтерфейс MapScreen та клас PlayerInputController
Серйозність	S5 Тривіальний (Trivial)
Пріоритет	P3 Низький (Low)
Кроки відтворення	Запустити сцену RoguelikeLevel, натиснути M для відкриття мапи на весь екран, спробувати атакувати чи кинути гаджет
Фактичний результат	На описані дії виконалась атака та кинувся гаджет
Очікуваний результат	Описані дії нічого окрім переміщення карти не роблять
Стан багу	Виправлено
Помилка №25	
Назва багу	Збереження зібраних ресурсів після смерті
Короткий опис	Після смерті, при натисканні на кнопку рестарт, зібрані речі зберігаються, тобто можна безкінечно помирати, лише накопичуючи ресурси, у рамках одного підземелля
Компонент програми	Сервіс LootService та інтерфейс LootTabScreen
Серйозність	S3 Значний (Major)

Кінець таблиці 5.1

Пріоритет	P1 Високий (High)
Кроки відтворення	Запустити сцену RoguelikeLevel, пройти кілька кімнат, назбиравши ресурсів, померти, натиснути повтор спроби або перезапустити рівень за натисканням Backspace, після чого натиснути Tab для перегляду ресурсів, що збереглися
Фактичний результат	Після смерті зібрані ресурси залишились
Очікуваний результат	Після смерті зібрані ресурси видалились з інвентаря
Стан багу	Виправлено

Згідно отриманим результатам тестування, було виявлено та виправлено найбільші проблеми, що не відповідали встановленим вимогам з проєктування та концептуального моделювання програмного модулю roguelike, що виступає предметом кваліфікаційної роботи.

Слід відмітити, що при вирішенні окремих проблем, було створено допоміжні системи та запобігаючі механіки, що узагальнюють певні аспекти та прибирають можливість повторної появи подібних помилок, що покращує загальну структуру проєкту та спрощує подальшу розробку.

6 ВПРОВАДЖЕННЯ ПРОГРАМНОГО МОДУЛЮ

6.1 Наукове впровадження проєкту

Для апробації кваліфікаційної роботи було проведене наукове дослідження за її тематикою. У зазначеній науковій роботі, було розглянуто питання генерації рівнів або підземель, а саме розібрано узагальнені підходи до процедурного створення локацій у ігрових проєктах жанру roguelike та не тільки, у розрізі інді-розробки зокрема. За результатами такого аналізу, було наведено загальну специфікацію, що містить у собі усі підходи до генерації рівнів (що складаються з кімнат чи альтернатив тощо), які були розділені за принципом генерації та підходом до оптимізації ресурсів пристрою, на якому запущена гра. У роботі було розглянуто переваги та недоліки кожного із зазначених підходів за наративною складовою, продуктивною та візуальною.

Після розглядання узагальнення основних підходів до генерації рівнів або підземель, що складаються з кімнат чи їх замінників, було представлено власну розробку, що відповідала одному з описаних підходів до процедурного відтворення. Було наведено розбір алгоритму дії підходу, продемонстровано графічну схему роботи розробки та описані властивості зазначеної варіації.

Для підтвердження важливості дослідження, було описано та продемонстровано порівняння із аналоговими підходами, до генерації рівня у єдиних умовах - усі тести проводились на єдиній тестовій сцені із використанням згенерованого рівня на двадцять дев’ять кімнат. Реалізовано тестовий проєкт було на рушії Unity3D. Після цього було наведено аналіз даних, що були отримані, а саме поточного показника кадрової частоти, кількості трикутників у кадрі, обсягу вершин, час прорахунку кадру центральним процесором та тривалість побудови зображення, навантаження самого додатку на комп’ютер та кількість зайнятої оперативної пам’яті. За отриманими результатами відзначено оптимальність зазначеного підходу з точки зору оптимізації навантаження, продуктивності та наративної складової, у порівнянні з описаними аналогами.

Слід відзначити, що дослідження ґрунтується на низці інших наукових робіт, що вказані у зазначеній доповіді.

Описане наукове дослідження було опубліковано у форматі статті, на шістьох сторінках, у VI Міжнародній науково практичній конференції «Débats scientifiques et orientations prospectives du développement scientifique» (Наукові дискусії та перспективні напрямки розвитку науки), що проходила у Парижі першого березня 2024 року, під заголовком: «Оптимізація випадкової генерації roguelike рівнів», у дев’ятнадцятій секції, а саме «Génie informatique et logiciel» (комп'ютерна і програмна інженерія) (див. додаток Ж).

6.2 Практичне впровадження проєкту

У якості практичної апробації проєкту «The fate of the liar», зокрема програмного модулю roguelike, що розробляється у рамках кваліфікаційної роботи, було обрано участь у виставці, що відбувалась під час XXVIII Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті» у період з 16 по 18 квітня 2024 року, у секції «Ігрові технології».

На зазначеній виставці було показано проєкт у загальному плані, з точки зору його працездатності та можливостей. Під час проведення цього заходу, було продемонстровано журі презентаційні матеріали (див. додаток И) та надано відповіді на запитання суддів з комісії, щодо стану проєкту та прийнятих рішень тощо.

Результатом практичного впровадження ігрового застосунку «The fate of the liar», програмного модулю roguelike, що розробляється у рамках кваліфікаційної роботи, зокрема, постають каталог виставки із наявною відомістю про проєкт (див. додаток К) та диплом переможця для розробки у номінації «Ігрових технології», який відзначає певний рівень якості усього продукту (та окремого програмного модулю, що є відображенням кваліфікаційної роботи у тому ж числі), що було розроблено та продемонстровано (див. додаток Л).

7 МЕДІЙНА ДІЯЛЬНІСТЬ

7.1 Постанова мети медійної активності

Проєкт «The fate of the liar», що розроблюється, зокрема його програмний roguelike модуль (що виступає предметом кваліфікаційної роботи) є інді-розробкою. У доповненні, команда розробників, не має аудиторії, що зацікавиться розробкою, тому, при випуску гри на ринок, його оминуть через недостатній рівень зацікавленості. Також, фактор «інді», має на увазі відсутність можливості створювати коштовні рекламні активності й проводити подібні заохочувальні дії, що могли б збудувати початкову аудиторію навколо проєкту та підняти рівень інтересу до нього.

Таким чином, для пошуку зацікавленої аудиторії для проєкту «The fate of the liar», слід проводити медійну діяльність, що будуватиме бачення проєкта у охочих глядачів та збирати увагу потенційних гравців.

Подібна активність, окрім збору потенційної аудиторії, створює можливість, спілкуватися із кінцевим користувачем та отримувати цінні данні з їх відґуків на інформацію стосовно гри, що публікується. Завдяки подібному способу розмови із аудиторією, можна вчасно змінити курс розробки в загалом, прибрати найменш популярні рішення, додати щось, за проханням користувачів тощо, ще на ранніх етапах розробки, що приблизить проєкт до бажаного, аудиторією, виду, та позитивно вплине на інтерес до проєкту.

Таким чином, постає доречним, проведення медійної активності, що спрямована на заохочення кінцевої аудиторії до звернення уваги на проєкт, що розроблюється, та використання прямого спілкування із користувачами, задля покращення та відхилення у більш вигідну сторону розробки гри.

Інструментом медійної активності, було обрано низку соціальних мереж, а саме:

- відеохостинг YouTube – для публікації відеоконтенту, у першу чергу, та збір відповідної аудиторії на зазначеній платформі;
- кросплатформену соціальну мережу Telegram, для привернення уваги аудиторії молодого та середнього віку;

- соцмережу Instagram, у якості інструменту для привернення більш молодої аудиторії;
- платформу X (Twitter), як інструмент із привернення західної, англомовної, аудиторії;
- соціальну мережу Facebook, у якості платформи для збору більш старшої аудиторії.

Слід зазначити, що для привернення ширшої аудиторії, уся медійна активність, ведеться на англійській мові. Також, не менш важливим, для підтримки рівня зацікавленості користувачів, є підхід, із додаванням додаткового наповнення соцмереж, а саме контенту, що напряму не пов'язаний із тематикою проєкту, однак має певний зв'язок, наприклад кумедні меми або повчальні дописи, які розповідають про застосовані підходи до розробки, тощо. Додатково відзначимо, що кожна соцмережа має свої власні обмеження та напрямки бажаного наповнення та вимоги аудиторії, що вимагає відповідного балансування контенту між платформами.

7.2 Реклама проєкту

Як вже зазначалось, у попередньому розділі (7.1), гра «The fate of the liar» та програмний модуль roguelike зокрема, що розроблюється, належить до «інді» сектору, що має на увазі відсутність зацікавленої аудиторії у розробках команди та грошей на її «покупку». Тож було прийняте рішення провести рекламну акцію, серед подібних проєктів, завдяки чому, певна частка аудиторії кожної з розробок, має шанс на зацікавлення грою «The fate of the liar» та навпаки. До таких інді-проєктів увійшли наступні:

- «Bounty Chase» - гра у стилістиці roguelike RPG;
- «Icarus» - кооперативне космічне виживання від першої особи;
- «Tale of the Awakened» - гра у стилістиці магічне середньовіччя, «меджікпанку»;
- «Synthetic Supremacy» - 3D-шутер від третьої особи у стилі кіберпанку;

- «Blood Grace» - двовимірна пригодницька гра з елементами рольової гри у стилістиці темного фентезі;
- «Steam Mystery» - 3D пригодницький бойовик із елементами RPG, у стилістиці Вікторіанської епохи та стімпанк;
- «Dimension of Darkness» - ізометричний shoot'em-up з елементами roguelike.

Слід зазначити, що проєкти, які співпрацювали у рекламній активності, є такими самими інді-розробками, із малою аудиторією, проте подібна діяльність є шансом на здобуття нових зацікавлених користувачів та подальше просування проєкту в медійному просторі та загалом.

7.3 Аналіз активності у YouTube

Розглядаючи активність на платформі YouTube, слід зазначити, що на зазначену платформу, викладалися переважно відеоматеріали, через специфіку сервісу, однак, також були й текстові публікації, що розміщувалися у розділі спільноти.

На платформі YouTube, було опубліковано шість відеороликів, що сумарно здобули більше ста переглядів, а саме сто п’ятдесят п’ять (див. рис. 7.1). Найбільш популярним відеоматеріалом виявилась демонстрація ігрового процесу в підземеллях, а саме функціонал програмного модулю roguelike, що розроблюється у рамках кваліфікаційної роботи (див. рис. 7.2).

Content	Views	Watch time (hours)	Subscribers	Impressions	Impressions click-through rate
Total	**155**	**1.3**	**14**	**495**	**11.1%**
10:47 Roguelike part gameplay	37 23.9%	0.5 38.1%	0 0.0%	247	4.1%
2:39 UI Creating Timelapse	31 20.0%	0.2 12.6%	1 7.1%	27	37.0%
1:48 Hub building process	28 18.1%	0.2 15.2%	0 0.0%	56	23.2%
4:16 Level generation development by stages	21 13.6%	0.2 12.2%	1 7.1%	68	10.3%
3:37 Tavern interaction demonstrating	19 12.3%	0.2 11.8%	0 0.0%	63	14.3%
3:15 Hub interaction demo	19 12.3%	0.1 10.2%	0 0.0%	34	17.7%

Рисунок 7.1 – Загальна статистика платформи YouTube, щодо опублікованих матеріалів (розроблено автором)

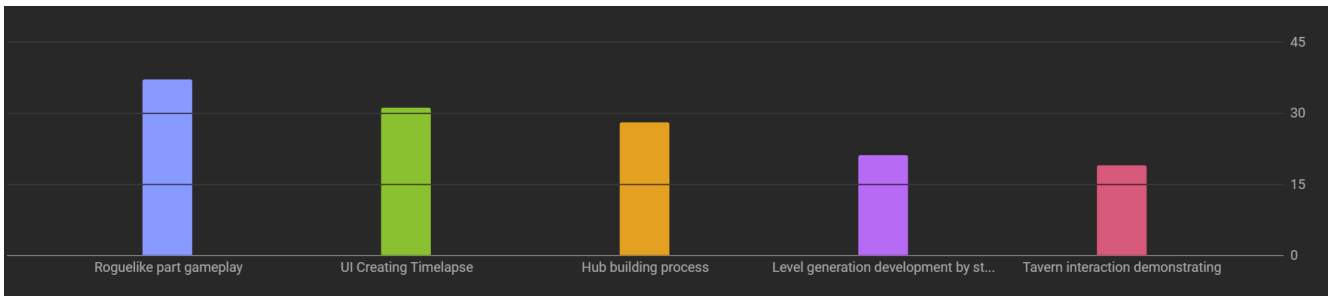


Рисунок 7.2 – Статистика платформи YouTube, щодо популярності опублікованих матеріалів (розроблено автором)

Слід відзначити, що на канал було підписано чотирнадцять користувачів (див. рис. 7.3), що відображає зацікавленість аудиторії проєктом, що розроблюється, також слід зазначити, що переважна більшість з переглядів, були зроблені саме підписниками, що відображено у відповідній статистиці (див. рис. 7.4).

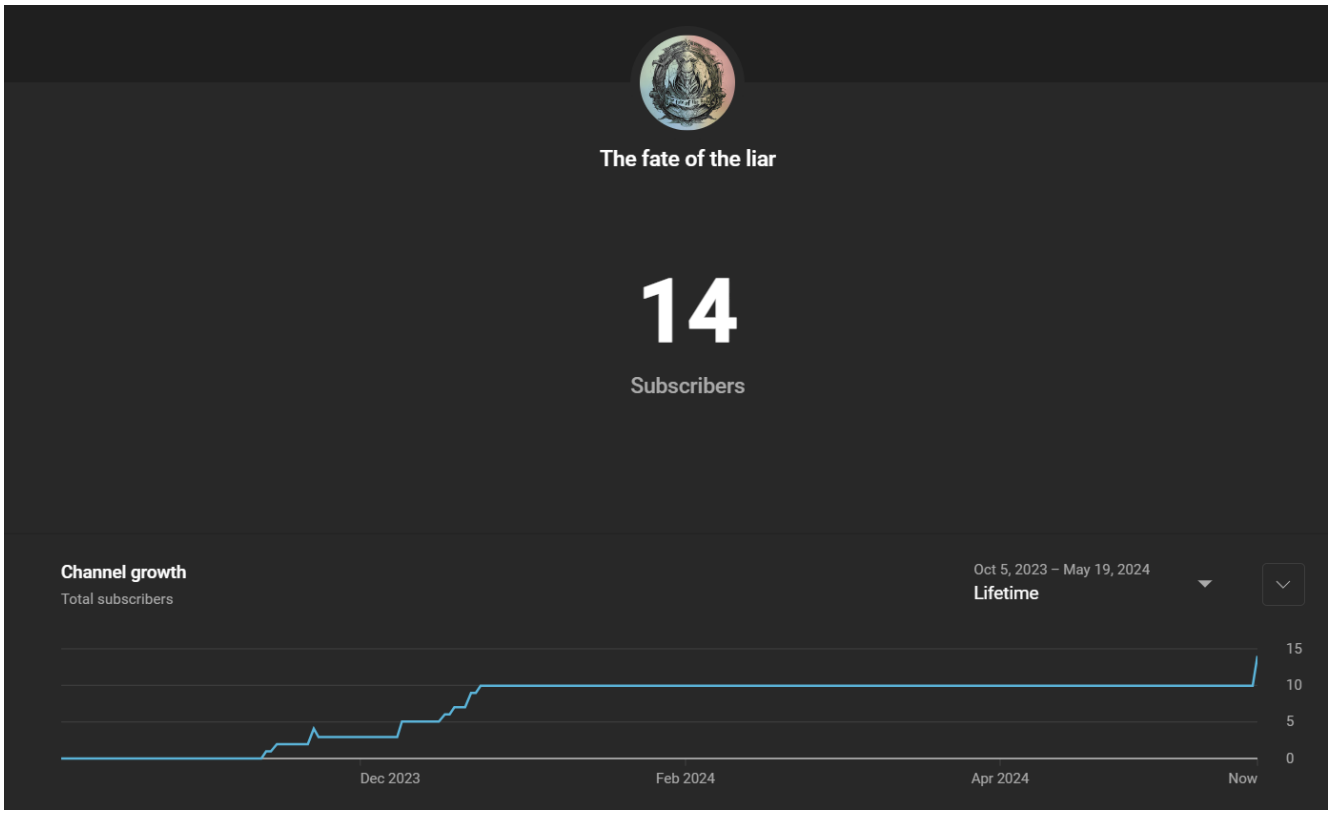
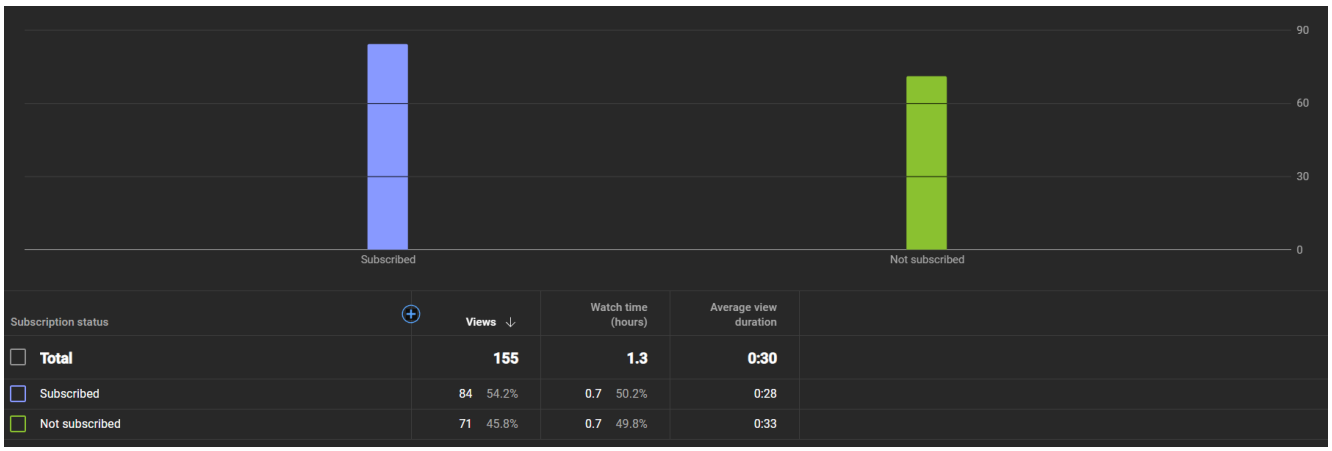


Рисунок 7.3 – Статистика платформи YouTube, щодо кількості аудиторії, що відображається за реальними датами підписок (розроблено автором)



Subscription status	Views	Watch time (hours)	Average view duration
Total	155	1.3	0:30
Subscribed	84 54.2%	0.7 50.2%	0:28
Not subscribed	71 45.8%	0.7 49.8%	0:33

Рисунок 7.4 – Статистика платформи YouTube, щодо переглядів опублікованих матеріалів за підпискою та без (розроблено автором)

Кількість вподобайок кожного з викладених відеоматеріалів (що згадувалися вище), відповідає у середньому розміру аудиторії, та відображає певну зацікавленість проєктом та вподобанням того, що робиться у процесі розробки гри.

Як зазначалось вище, окрім відеороликів, до платформи публікувався й текстовий контент, однак, через замалу аудиторію на поточний момент та відповідно до специфіки сервісу YouTube, вони не набули достатньої реакції від підписників. Це показує більшу користь саме від відеоконтенту на зазначеній платформі, при умові малої відомості серед користувачів та інді фактору зокрема.

7.4 Аналіз активності у Instagram

Специфікою соціальною мережі Instagram, постає спрямованість контенту, у графічний напрямок, однак на відміну від платформи YouTube, цільовим наповненням сервісу виступають зображення. У додачу до цього, у Instagram існують історії, що є окремим способом публікації. Останні є зручним способом для проведення певних оголошень для підписників, реклами та тимчасово актуальних публікацій. Саме із використанням історій, відбувався взаємний обмін аудиторіями із іншими інді-проєктами, які були описані у підрозділі 7.2.

Розглядаючи результати медійної активності у соціальній мережі, що обговорюється, варто відмітити, що за обумовлений час, було залучено увагу

більше двадцяти людей, а саме, вдалося отримати вісімнадцять підписників та двадцять шість осіб, що стежать за активністю каналу (див. рис. 7.5).

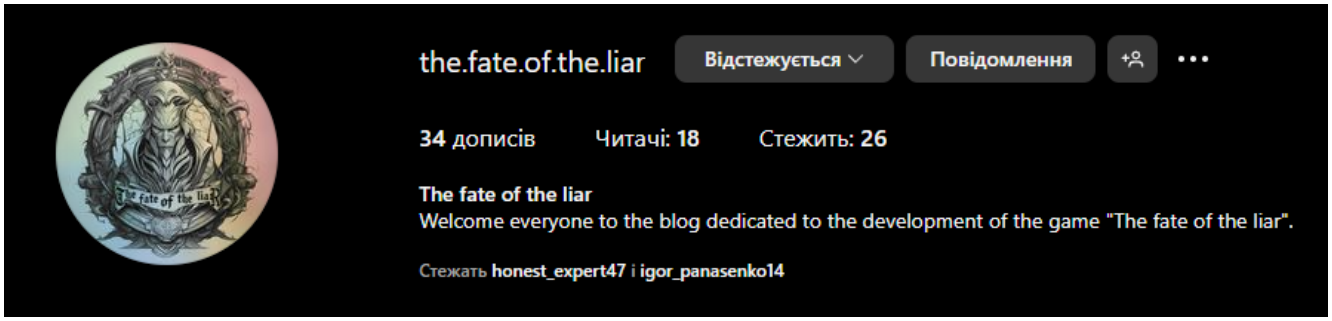


Рисунок 7.5 – Зображення основної інформації, щодо сторінки у соцмережі Instagram (розроблено автором)

За час проведення медійної активності, було зроблено тридцять чотири дописи та опубліковано вісім історій. Майже усі публікації мають, близьку до середньої, кількість вподобайок, а саме від дев’яти до дванадцяти. Розглядаючи питання популярності публікацій, слід відзначити найпопулярніший пост, який сподобався користувачам найбільше та отримав рекордні тринадцять вподобайок (див. рис. 7.6). Також, незважаючи на замалу аудиторію на поточний момент, зацікавлені користувачі залишали власні коментарі, у реакції на побачене наповнення (див. рис. 7.7).

Рисунок 7.6 – Зображення найпопулярнішого постів (розроблено автором)

Рисунок 7.7 – Зображення публікації разом із коментарем (розроблено автором)

Аналізуючи результати медійної активності у соціальній мережі Instagram, можна дійти до висновку (який будується на отриманому досвіді), що найбільш актуальним наповненням для цієї платформи постають саме кумедні та смішні меми, оскільки вони привертають увагу найбільше та допомагають отримати більш молоду аудиторію, що переважає у зазначеному сервісі.

7.5 Аналіз активності у Telegram

Соціальна мережа Telegram, надає інструментарій для створення будь якого контенту для власного каналу. Текстові повідомлення, відеоматеріали, зображення тощо, усе це є рівноправними за можливостями та значенням наповненням для медійного джерела, що привертає увагу цільової аудиторії до інді-розробки, що розроблюється. Таким чином, при розвитку зазначеної соціальної мережі, було використане поєднання типі публікацій, що згадуються вище. Слід зазначити наперед, що усі вони отримали однаково гарний відклик від аудиторії та не мали негативних відгуків.

Розглядаючи результати активності у соціальній мережі Telegram, було опубліковано сорок два пости, що містили інформативний контент:

- щодо процесу розробки та творчих ідей;
- щодо наповнення гри, опитування аудиторії;
- щодо їх бачення про поточний стан розвитку проєкту, наративних ідей тощо;
- щодо застосованих підходів при розробці;
- комедійний контент;
- рекламні публікації, що вміщали у себе, як просування власних каналів (YouTube, X (Twitter), Facebook, Instagram), так і дружніх інді-проєктів, що згадувалися у підрозділі 7.2.

Результатом такої активності, стало набуття аудиторії підписників, у розмірі сімнадцяти користувачів (див. рис. 7.8) та понад п’ятдесят загальних читачів, свідченням про яких, є кількість унікальних переглядів певних публікацій (див. рис. 7.9).

Рисунок 7.8 – Зображення Telegram каналу, зі статистикою про кількість підписників (розроблено автором)

Рисунок 7.9 – Зображення публікації разом із найбільшою кількістю унікальних переглядів (розроблено автором)

Слід відзначити, що аудиторія у зазначеному каналі є активною, про що свідчить наявність постійних емодзі-реакцій, майже під кожною публікацією. Також відмітимо пост, який мав найбільше охоплення зацікавленої аудиторії та відповідно отримав більше всього реакцій, а саме, публікацію, у якій було показано референсне зображення орка (що є представником однієї з п’яти рас, які будуть продемонстровані у грі «The fate of the liar») (додаток В) та розказано, про особливості цього народу, у наративному направленні (див. рис. 7.10).

Рисунок 7.10 – Зображення публікації із найбільшим охопленням цільової аудиторії (розроблено автором)

Слід відмітити, що при створені цього Telegram каналу, додатково було прив’язано групу для обговорення інформації, що публікується та мала можливість залишати коментарі, напряму під конкретними постами (див. рис. 7.11). Тож певні публікації також отримали відгуки та запитання від аудиторії, на які було надано відповіді тощо (див. рис. 7.12 – 7.13).

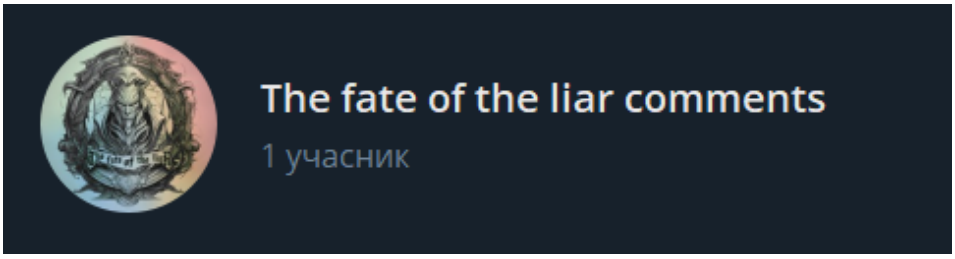


Рисунок 7.11 – Зображення додаткового каналу для обговорення та коментування опублікованого контенту в основному (розроблено автором)

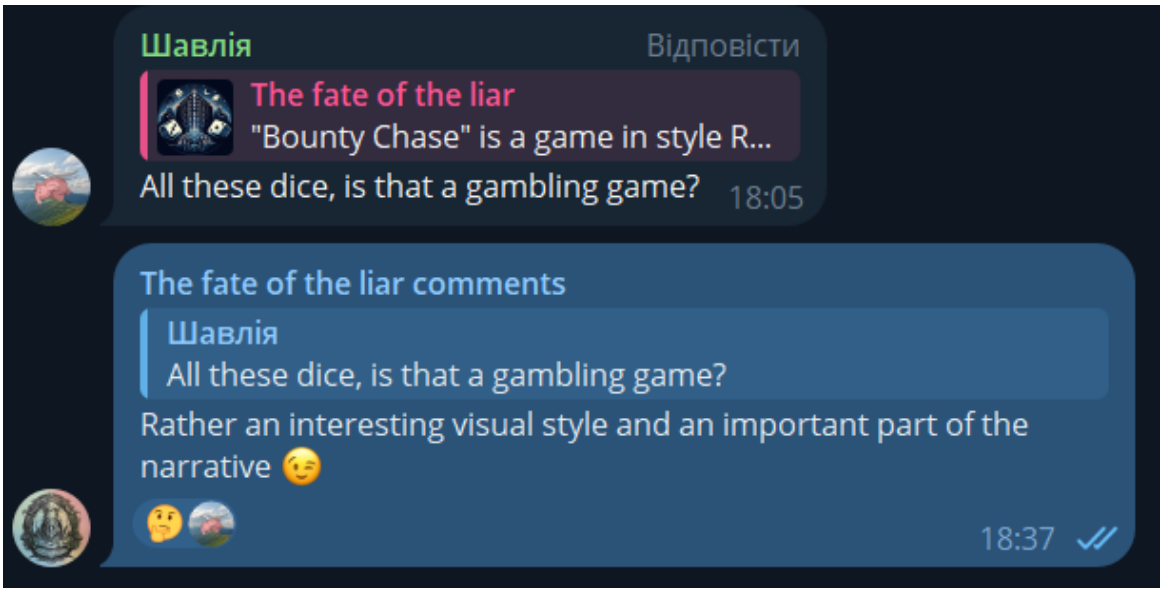


Рисунок 7.12 – Зображення питання у коментарях та відповіді на нього (розроблено автором)

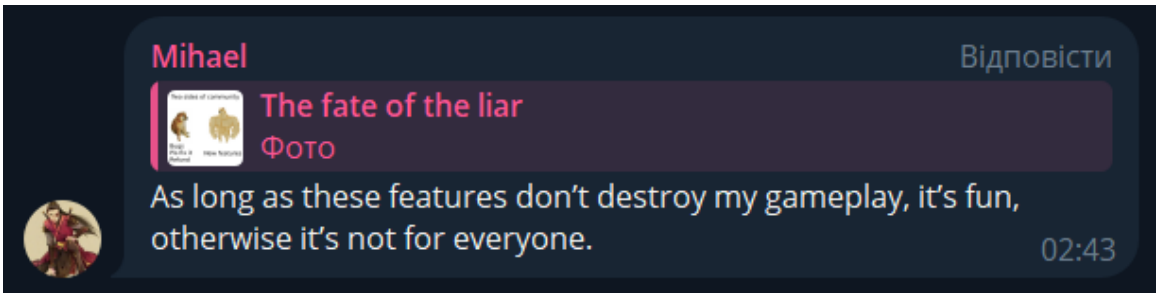


Рисунок 7.13 Зображення коментаря до публікації (розроблено автором)

Відзначимо, що аудиторія у Telegram виявилась більш активною та різнобічною, на що вказує гарне сприйняття усього наповнення каналу, що було створено, не зважаючи на більше розгалуження типів наповнення, у порівнянні з іншими соціальними мережами та сторінками у них, що велись під час зазначеної медійної активності.

7.6 Аналіз активності у Facebook

Розглядаючи активність соціальної мережі Facebook, варто відмітити, що за час проведення медійної діяльності, було здобуто шість постійних читачів, що підписалися на означену сторінку Facebook та понад чотирнадцять унікальних відвідувачів, що провзаємодіяли із наповненням каналу, а саме сорока двома дописами, що було опублікуовано (див. рис. 7.14).

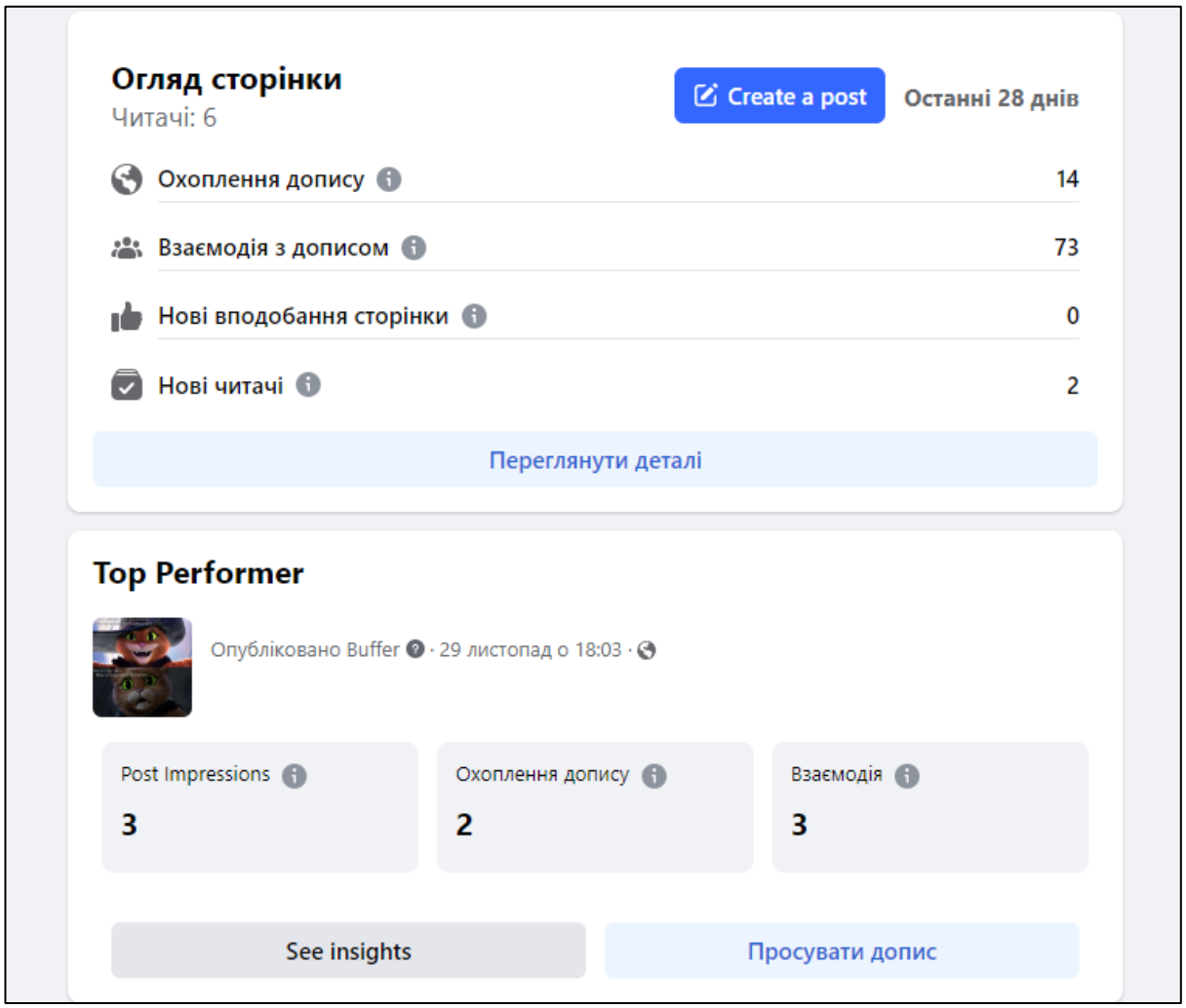


Рисунок 7.14 – Зображення статистики, щодо активності зі сторінкою Facebook (розроблено автором)

Найпопулярнішою публікацією став кумедний мем, що отримав найбільше вподобайок та переглядів, у поєднанні зі взаємодіями (див. рис. 7.15).

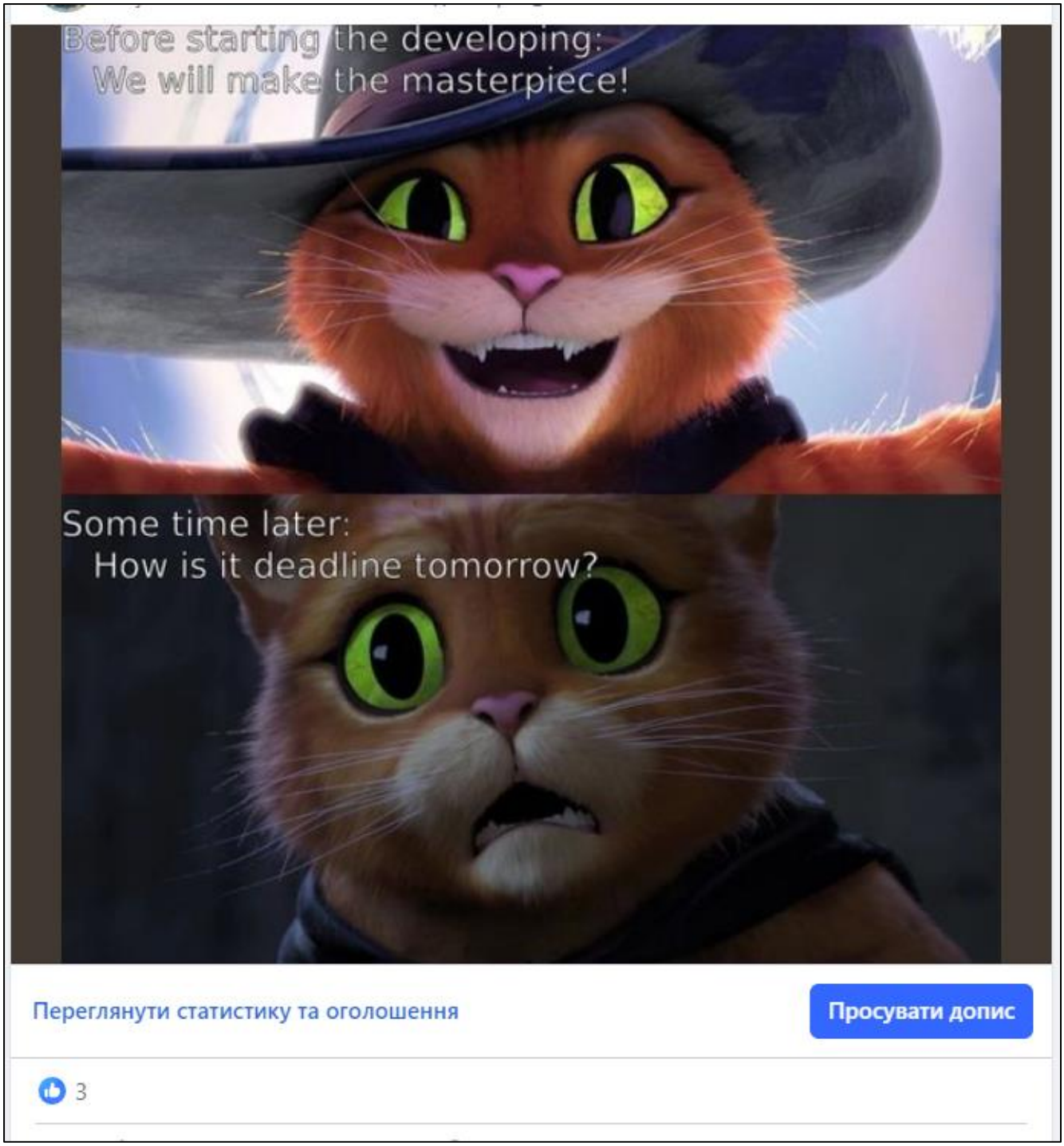


Рисунок 7.15 – Зображення найбільш популярного допису на сторінці Facebook (розроблено автором)

Варто звернути увагу на те, що через основну аудиторію Facebook, а саме те, що вони є більш старшими, охоплення, що отримала сторінка, є відносно меншим,

ніж у решти. Зазначений результат вказує на обмеження основної множини кінцевих користувачів певними віковими рамками, однак, це не відміняє те, що гра «The fate of the liar», потенційно, може мали гравців поза цільової аудиторії, за віковим фактором зокрема.

7.7 Аналіз активності у X (Twitter)

За час медійної діяльності у соціальній мережі X, раніше Twitter, було опубліковано сорок три дописи та отримано п’ять підписників (див. рис. 7.16). Разом із підписниками, сторінку відвідало понад п’ятдесят унікальних облікових записів (див. рис. 7.17).

Рисунок 7.16 – Статистика сторінки X (Twitter) із кількістю підписників (розроблено автором)

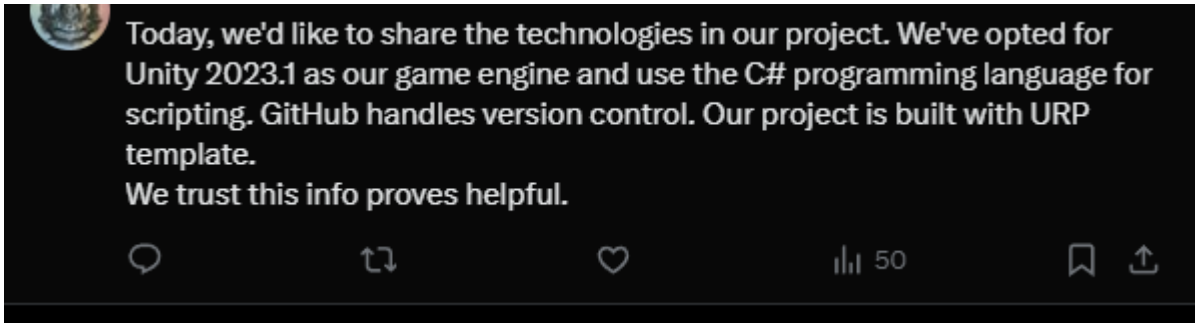


Рисунок 7.17 – Статистика сторінки X (Twitter) із кількістю унікальних переглядів допису (розроблено автором)

Варто відзначити допис, що більше всього сподобався аудиторії, а саме отримав найбільшу кількість вподобайок та унікальних переглядів одночасно,

серед інших. Такою публікацією постає допис зі звітністю, щодо розроблених на поточний момент функцій та можливостей ігрового застосунку «The fate of the liar» (див. рис. 7.18)

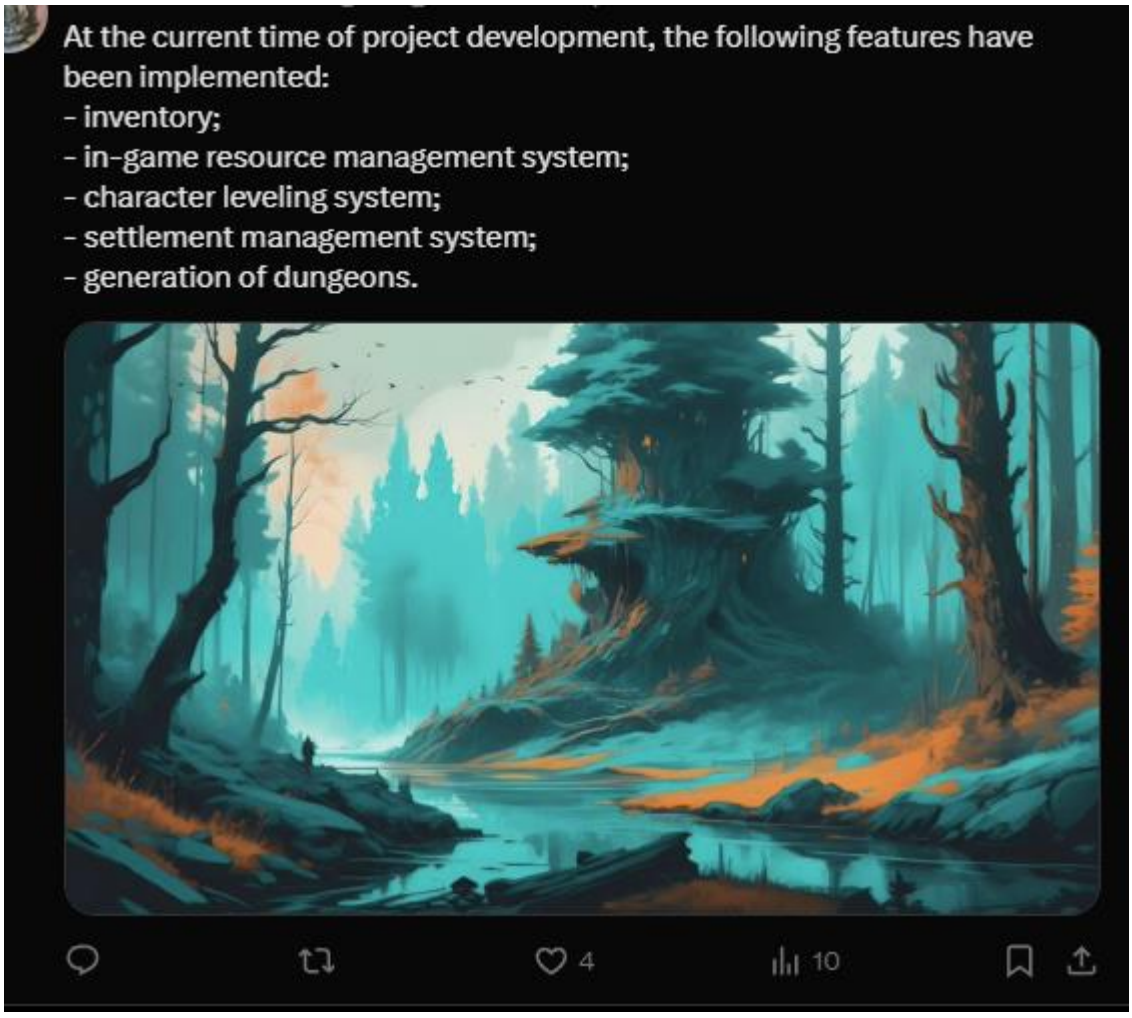


Рисунок 7.18 – Найпопулярніший допис на сторінці X (Twitter) за кількістю унікальних переглядів та вподобайок одночасно (розроблено автором)

Слід відзначити, хоч у результаті медійної діяльності у соціальній мережі X (Twitter) не було здобуто велику аудиторію підписників, однак натомість, сторінку відвідувало достатня кількість зацікавлених користувачів, аби вважати, що певна цільова аудиторія була зібрана.

7.8 Підсумки медійної діяльності

Відзначимо, що для створення візуального контенту, здебільшого, було використано сервіси зі штучним інтелектом із генерацією зображень, такі

«neural.love» [30], «Freepik» [31] тощо. У якості музичного супроводження у відеоматеріалах, було використано композиції, що розповсюджуються вільно, згідно авторському праву, та у описі опублікованих відео, уся необхідна інформація була вказана, аби не порушувати закон з доброчесного використання інтелектуальної власності. Також варто відзначити сервіс, що було використано для централізованої публікації багатьох постів у різні соціальні мережі, а саме платформу «Buffer» [32]

Результатом проведення медійної діяльності у п’яти соцмережах: YouTube, Instagram, Telegram, Facebook, X (Twitter), стала низка дописів та публікацій, що були розміщенні на відповідних платформах (див. рис. 7.19). Додатково слід відзначити, що до медійного впровадження можна віднести участь у виставці Міжнародного молодіжного форуму, що розглядалась у розділі 6, а саме підрозділі 6.2.

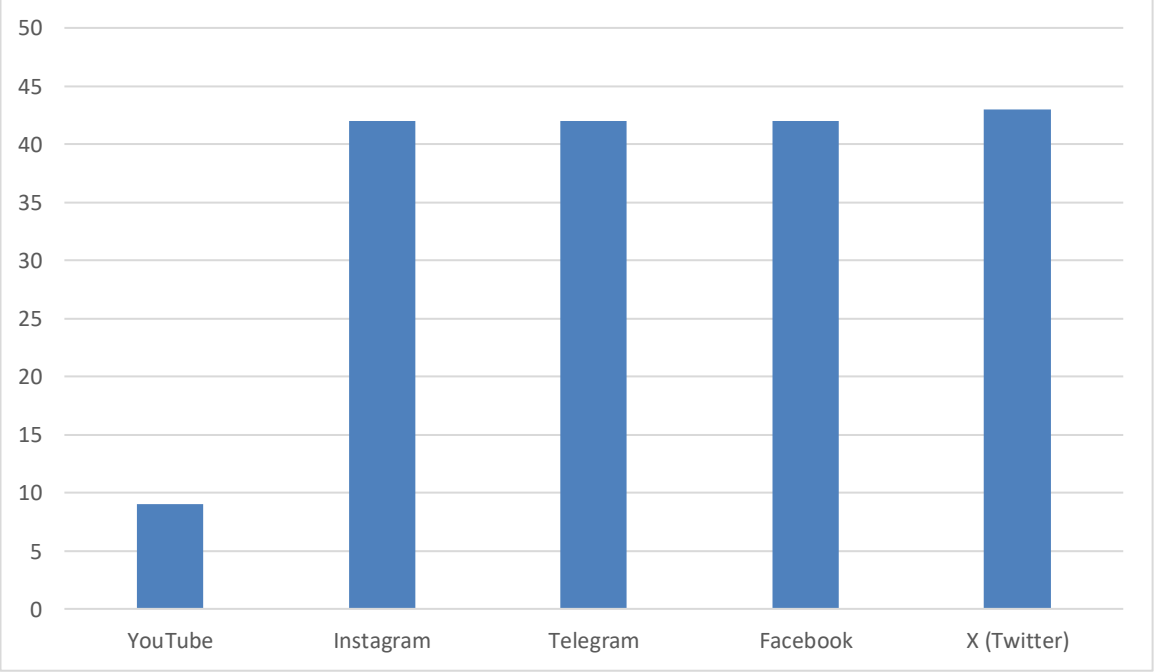


Рисунок 7.19 – Діаграма, що демонструє кількість публікацій у кожній з обраних соціальних мереж (розроблено автором)

Підсумовуючи аналіз медійної активності, що проводився у попередніх підпунктах: 7.3 - 7.5, слід розглянути діаграму, що відтворює результат порівняння зібраних аудиторій, за фактичною кількістю підписників, та загальної множини

унікальних відвідувачів (див. рис. 7.1). Слід зазначити, що кількість унікальних відвідувачів можна вирахувати зі статистики щодо переглядів за підпискою та без неї.

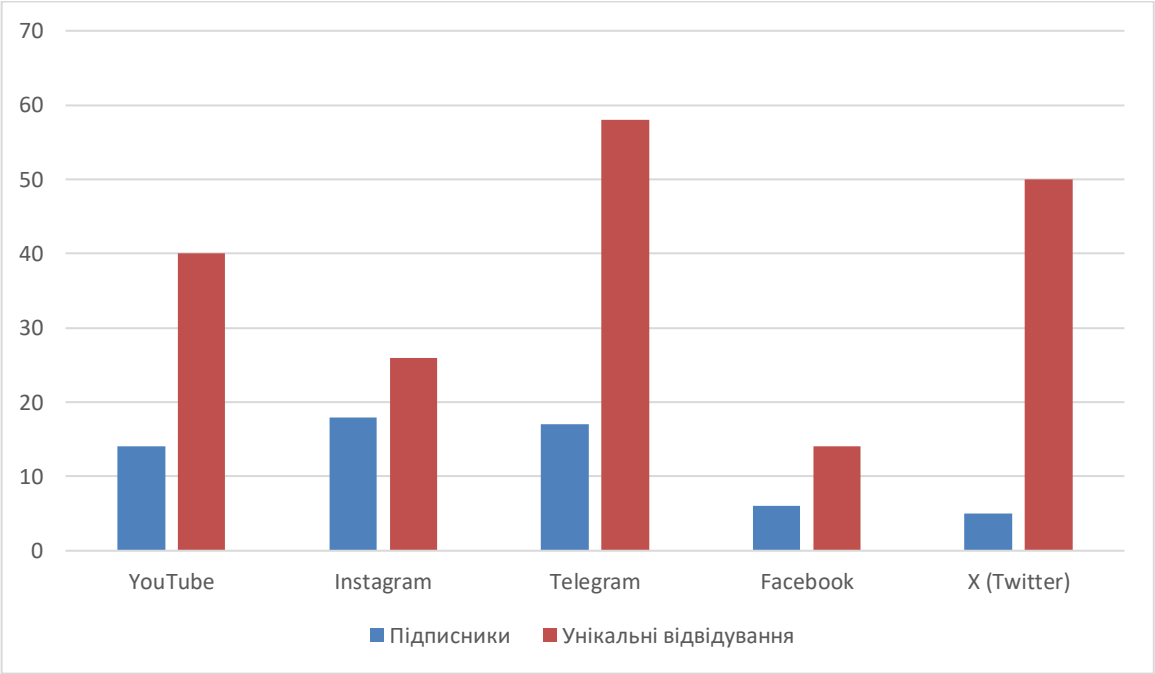


Рисунок 7.20 – Діаграма, що демонструє кількість підписників та унікальних відвідувачів у кожній з обраних соціальних мереж (розроблено автором)

З отриманої статистики можна зробити висновки, що по відношенню кількості дописів до розміру здобутої аудиторії лідирує платформа, YouTube, оскільки за значно менший обсяг публікацій, там вдалося зібрати аудиторію, що є максимально усередненої серед усіх інших мереж. Решта сервісів отримали приблизно рівні кількості дописів, тому їх доречно розглядати саме за кількістю здобутої аудиторії. Одразу слід виділити Telegram, що отримав найбільшу кількість унікальних відвідувань та майже максимально аудиторію з підписників, що демонструє влучність рішення, щодо використання цієї платформи. Instagram має значно нижчі показники з унікальним відвідуваннями сторінки, у порівнянні з попередником, однак відрізняється найбільшою здобутою аудиторією з читачів. Це демонструє те, що більшість користувачів, що відвідали сторінку, підписались. У свою чергу цей факт відображає гарне влучання у аудиторію та публікації, що були зроблені для цієї мережі. У свою чергу X (Twitter), хоч і отримав достатньо велике

охоплення аудиторії, що відвідало сторінку, однак серед підписників, лишилась дуже мала частка, що водночас є найнижчим результатом на тлі інших платформ. Це демонструє низьку зацікавленість користувачів у продемонстрованому наповнені сторінки та контенту інді-розробки «The fate of the liar» зокрема. Соціальна мережа Facebook, продемонструвала так само низький результат із привернення уваги аудиторії. Facebook трохи переганяє X (Twitter) за кількістю підписників, однак має найнижчу відвідуваність серед унікальних користувачів. Хоч зазначене відношення приблизно відповідає одному до двох, одна загальний показник демонструє майже повну відсутність зацікавленості аудиторії у зазначеній соціальній мережі до проєкту, що також вказує на низький подальший потенціал платформи.

Також слід відміти, що завдяки проведеній колаборації із іншими інді-проєктами у плані обміну аудиторіями у медійній діяльності (підрозділ 7.2), певна міграція підписників відбулась. Це вказує на успішність застосування даного підходу до збільшення основної аудиторії прихильників та поширення відомості гри у медіа просторі.

Згідно отриманим статистичним даним та зробленими за ними висновками, можна постановити, що найбільш актуальні соціальні мережі для медійної активності з просування проєктів «The fate of the liar» та збирання первинної аудиторії постають YouTube, Instagram та Telegram. У свою чергу X (Twitter) та Facebook, нажаль, мають значно нижчий потенціал до підбору прихильників та відрізняються за аудиторією та вимогами користувачів.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи бакалавра, було розроблено програмний модуль до гри «The fate of the liar», а саме компонент із випадковими підземеллями за roguelike жанром, включаючи усі супроводжуючі системи та механіки. Розробка поєднана із іншими частинами ігрового застосунку та є продовженням єдиного користувацького досвіду. Програмний модуль у повній мірі реалізує увесь функціонал, що був визначеним як необхідний, та включає у себе повноцінну систему генерації рівнів за roguelike зразком (з усіма відповідними аспектами жанру) та має потужні можливості до розширення та модифікації й вдосконалення у подальшому; для підземель було створено новітню бойову систему, що базується на RPG складовій – прокачці й уміннях, та стихійній механіці, що дозволяє гравцеві експериментувати та тактично планувати власне проходження рівнів; створено механіки дослідження підземель, із поєднанням з загальною ігровою системою; додана підтримка ряду геймпадів, що дозволятиме грати на консолях, завдяки чому можлива мультиплатформна підтримка у майбутньому; реалізовано додаткові системи та механіки для спрощення подальшої розробки, поєднання згаданих вище механік та покращення ігрового досвіду користувача.

Описаний вище функціонал був повністю розроблений на мові програмування C#, із використання рушія для створення ігор – Unity3D та вбудованих модулів, таких як VFX Graph для створення графічних ефектів, AI Navigation для впровадження штучного інтелекту для ворогів, нової системи введення Input System, що надає значно кращий API для створення управління персонажем тощо. Для створення якісного коду та підтримки його «чистоти», було застосовано патерни проєктування та кращі практики з розробки.

Для підтримки якості продукту було проведене тестування наявного функціонала згідно висунутим вимогам до розробки та потребам, що були визначенні на етапі планування. Усі знайдені помилки було усунуто та впроваджено покращення для запобігання подібних проблем у майбутньому.

Під час розробки гри «The fate of the liar», проводилась медійна активність у низці соціальних мереж, таких як Instagram, Telegram, Facebook, X (Twitter) та YouTube, у яких публікувалися пости, що стосувалися процесу реалізації ігрового застосунку та мали мету зібрати зацікавлену аудиторію. Результатом подібної активності постає збір потенційних користувачів продукта, що розроблюється, та, відповідно, це відображає певну зацікавленість та актуальність розробки на ринку.

Результат виконання кваліфікаційної роботи – програмний модуль із roguelike компонентом проєкту, було апробовано на Міжнародній науково-практичній конференції, із матеріалами щодо оптимізації однієї з основних механік та аспектів жанру roguelike – генерації рівнів, на основі розробленої системи у самій кваліфікаційні роботі. Повний ігровий застосунок «The fate of the liar» було апробовано на виставці Міжнародного молодіжного форуму, де було продемонстровано весь розроблений продукт загалом.

Перспективою розвитку проєкту та розробленого програмного модулю у якості кваліфікаційної роботи зокрема, є вихід гри у одному з сервісів, що надають відповідні послуги. Слід зазначити, що перед повноцінним релізом, варто зібрати відгуки зацікавленої аудиторії, наприклад через систему раннього доступу тощо, обробити їх та внести відповідні зміни до створеного продукту для відповідності до побажань цільової аудиторії тощо.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Epyx, Inc. Rogue [Електронний ресурс] / Epyx, Inc. – [Б. м.] : Pixel Games UK, 2020. – Режим доступу: https://store.steampowered.com/app/1443430/Rogue/ (дата звернення: 13.05.2024). – Назва з екрана.

2. Don Worth. Beneath Apple Manor [Електронний ресурс] / Don Worth. – The Software Factory. – [Б. м. : б. в.], 1978. – Режим доступу: https://archive.org/details/beneath-apple-manor-1978-version-fixed (дата звернення: 13.05.2024). – Назва з екрана.

3. Epyx, Inc. Sword of Fargoal [Електронний ресурс] / Epyx, Inc. – [Б. м.] : Pixel Games UK, 2022. – Режим доступу: https://store.steampowered.com/app/2231080/Sword_of_Fargoal/ (дата звернення: 13.05.2024). – Назва з екрана.

4. Poole S. Star Trek [Електронний ресурс] / Stephen Poole, Brian Mullin. – [Б м. : б. в.]. – Режим доступу: https://web.archive.org/web/20100616040240/http://www.gamespot.com/features/startrek_hs/ (дата звернення: 13.05.2024). – Назва з екрана.

5. Dungeons & Dragons | Official Home of the World's Greatest Roleplaying Game [Електронний ресурс] // D&D Official | Dungeons & Dragons. – Режим доступу: https://dnd.wizards.com (дата звернення: 13.05.2024). – Назва з екрана.

6. GURPS: Generic Universal RolePlaying System [Електронний ресурс] // Steve Jackson Games. – Режим доступу: https://www.sjgames.com/gurps/ (дата звернення: 13.05.2024). – Назва з екрана.

7. Tinner P. Dwarf Fortress Is Finally Getting Non-ASCII Art, And It Looks Way Better [Електронний ресурс] / Phillip Tinner // ScreenRant. – Режим доступу: https://screenrant.com/dwarf-fortress-graphics-art-tile-map-ascii-update/ (дата звернення: 13.05.2024). – Назва з екрана.

8. Bay 12 Games. Dwarf Fortress [Електронний ресурс] / Bay 12 Games. – [Б. м.] : Kitfox Games, 2022. – Режим доступу: https://store.steampowered.com/app/975370/Dwarf_Fortress/ (дата звернення: 13.05.2024). – Назва з екрана.

9. Edmund McMillen. The Binding of Isaac [Електронний ресурс] / Edmund McMillen, Florian Himsl. – [Б. м.] : Edmund McMillen, 2011. – Режим доступу: https://store.steampowered.com/app/113200/The_Binding_of_Isaac/ (дата звернення: 13.05.2024). – Назва з екрана.

10. Roguelikeness Factors [Електронний ресурс] // Temple of The Roguelike. – Режим доступу: https://blog.roguetemple.com/roguelike-definition/roguelikeness-factors/ (дата звернення: 13.05.2024). – Назва з екрана.

11. Rogue-Likes & Rogue-Lites [Електронний ресурс] // Welcome to Steam. – Режим доступу: https://store.steampowered.com/category/rogue_like_rogue_lite?flavor=contenthub_toprated (дата звернення: 13.05.2024). – Назва з екрана.

12. Supergiant Games. Hades [Електронний ресурс] / Supergiant Games. – [Б. м.] : Supergiant Games, 2020. – Режим доступу: https://store.steampowered.com/app/1145360/Hades/ (дата звернення: 13.05.2024). – Назва з екрана.

13. Motion Twin. Dead Cells [Електронний ресурс] / Motion Twin. – [Б. м.] : Motion Twin, 2018. – Режим доступу: https://store.steampowered.com/app/588650/Dead_Cells/ (дата звернення: 13.05.2024). – Назва з екрана.

14. AI Navigation | AI Navigation | 2.0.0 [Електронний ресурс] // Unity - Manual: Unity User Manual 2022.3 (LTS). – Режим доступу: https://docs.unity3d.com/Packages/com.unity.ai.navigation@2.0/manual/index.html (дата звернення: 14.05.2024). – Назва з екрана.

15. Cerny V. Rogue-Like Games as a Playground for Artificial Intelligence – Evolutionary Approach [Електронний ресурс] / Vojtech Cerny, Filip Dechterenko // Entertainment Computing - ICEC 2015. – Cham, 2015. – С. 261–271. – Режим доступу: https://doi.org/10.1007/978-3-319-24589-8_20 (дата звернення: 23.05.2024). – Назва з екрана.

16. Bacher T. Procedural Level Generation Algorithms / T. Bacher. – 2018.

17. Smith A. J. A Logical Approach to Building Dungeons: Answer Set Programming for Hierarchical Procedural Content Generation in Roguelike Games

[Електронний ресурс] / Anthony J. Smith, Joanna J. Bryson. – 2014. – Режим доступу: https://www.semanticscholar.org/paper/A-Logical-Approach-to-Building-Dungeons-:-Answer-in-Smith-Bryson/f6bd2ffa434f59100c37f3f5a85e6aacb91e09df#citing-papers (дата звернення: 23.05.2024). – Назва з екрана.

18. Input System | Input System | 1.8.2 [Електронний ресурс] // Unity - Manual: Unity User Manual 2022.3 (LTS). – Режим доступу: https://docs.unity3d.com/Packages/com.unity.inputsystem@1.8/manual/index.html (дата звернення: 14.05.2024). – Назва з екрана.

19. Singleton [Електронний ресурс] // Refactoring and Design Patterns. – Режим доступу: https://refactoring.guru/design-patterns/singleton (дата звернення: 14.05.2024). – Назва з екрана.

20. Martin R. Clean Code: A Handbook of Agile Software Craftsmanship / Robert Martin. – [Б. м.] : Pearson Education, Limited.

21. Universal Render Pipeline (URP) | Unity [Електронний ресурс] // Unity. – Режим доступу: https://unity.com/srp/universal-render-pipeline (дата звернення: 21.05.2024). – Назва з екрана.

22. Borromeo N. A. Hands-On Unity 2020 Game Development: Build, Customize, and Optimize Professional Games Using Unity 2020 and C# / Nicolas Alejandro Borromeo. – [Б. м.] : Packt Publishing, Limited, 2020. – 580 с.

23. Thorn A. Unity Animation Essentials / Alan Thorn. – [Б. м.] : Packt Publishing, Limited, 2015.

24. Unity - Manual: Order of execution for event functions [Електронний ресурс] // Unity - Manual: Unity User Manual 2022.3 (LTS). – Режим доступу: https://docs.unity3d.com/Manual/ExecutionOrder.html (дата звернення: 21.05.2024). – Назва з екрана.

25. Buttfield-Addison P. Unity Game Development Cookbook: Essentials for Every Game / Paris Buttfield-Addison, Jon Manning, Tim Nugent. – [Б. м.] : O'Reilly Media, 2019. – 406 с.

26. Долгий А. Оптимізація випадкової генерації roguelike рівнів [Електронний ресурс] / Андрій Долгий // Débats scientifiques et orientations

prospectives du développement scientifique / керівник Ю. Новіков. – [Б. м.], 2024. – Режим доступу: https://doi.org/10.36074/logos-01.03.2024.050 (дата звернення: 23.05.2024). – Назва з екрана.

27. Event Bus pattern [Електронний ресурс] // Manh Phan - Practice makes perfect!. – Режим доступу: https://ducmanhphan.github.io/2020-06-06-Event-Bus-pattern/ (дата звернення: 21.05.2024). – Назва з екрана.

28. 10 Unity Input System Best Practices [Електронний ресурс] // CLIMB. – Режим доступу: https://climbtheladder.com/10-unity-input-system-best-practices/ (дата звернення: 21.05.2024). – Назва з екрана.

29. Zhang B. Game special effect simulation based on particle system of Unity3D [Електронний ресурс] / Bingqing Zhang, Wenfeng Hu // 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), Wuhan, China, 24–26 трав. 2017 р. – [Б. м.], 2017. – Режим доступу: https://doi.org/10.1109/icis.2017.7960062 (дата звернення: 22.05.2024). – Назва з екрана.

30. Free AI Art Generator & AI Enhance | neural.love [Електронний ресурс] // Free AI Art Generator & AI Enhance | neural.love. – Режим доступу: https://neural.love (дата звернення: 21.05.2024). – Назва з екрана.

31. Freepik | Create great designs, faster [Електронний ресурс] // Freepik. – Режим доступу: https://www.freepik.com (дата звернення: 21.05.2024). – Назва з екрана.

32. Buffer [Електронний ресурс] // Buffer. – Режим доступу: https://buffer.com (дата звернення: 21.05.2024). – Назва з екрана.

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

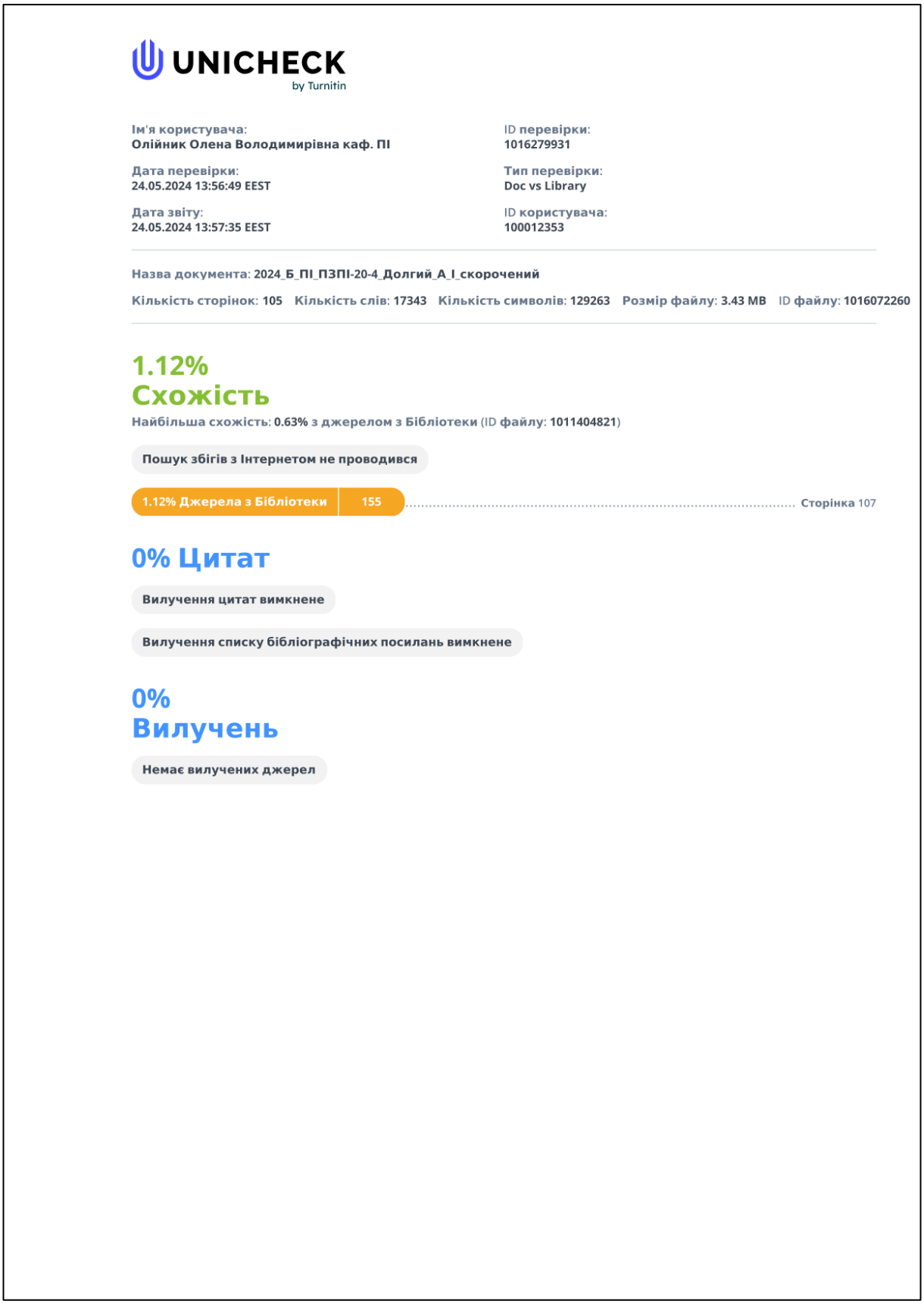


Рисунок А.1 – Результати перевірки на роботи на унікальність

ДОДАТОК Б

Слайди презентації

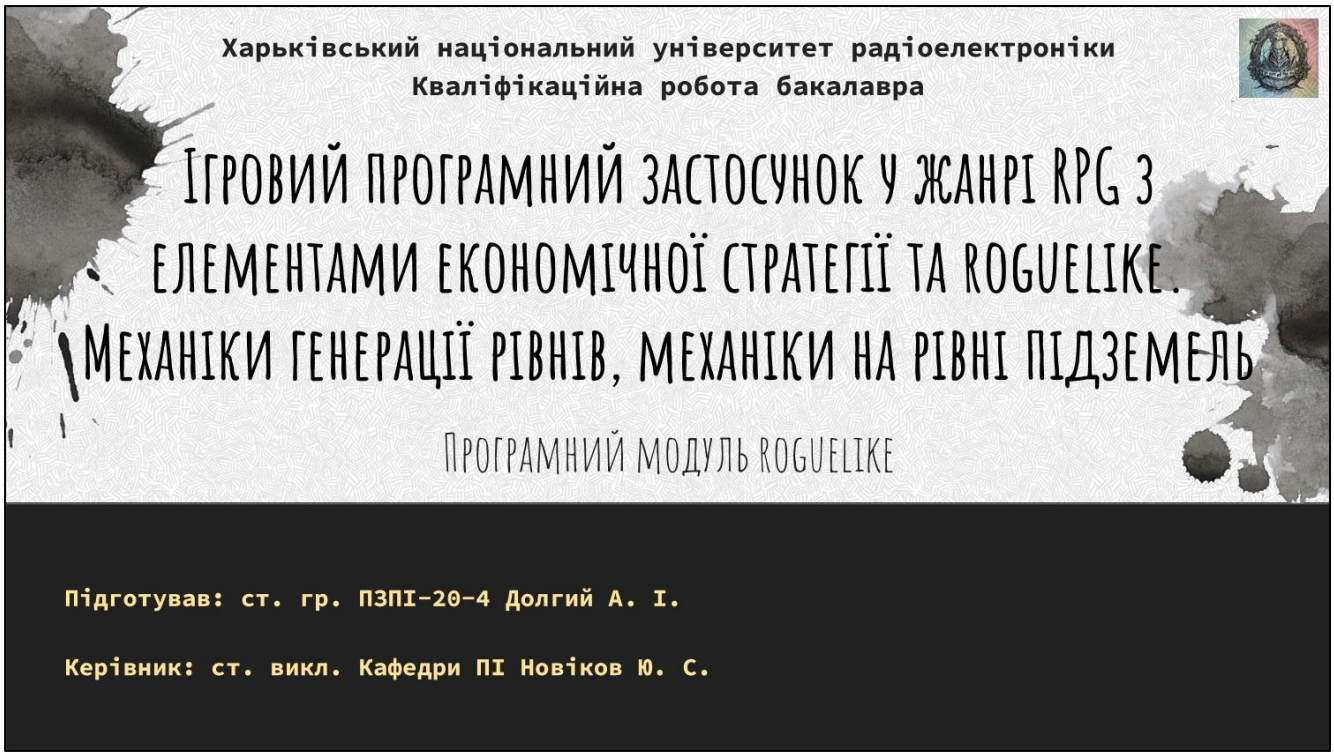


Рисунок Б.1 – Перший слайд презентації

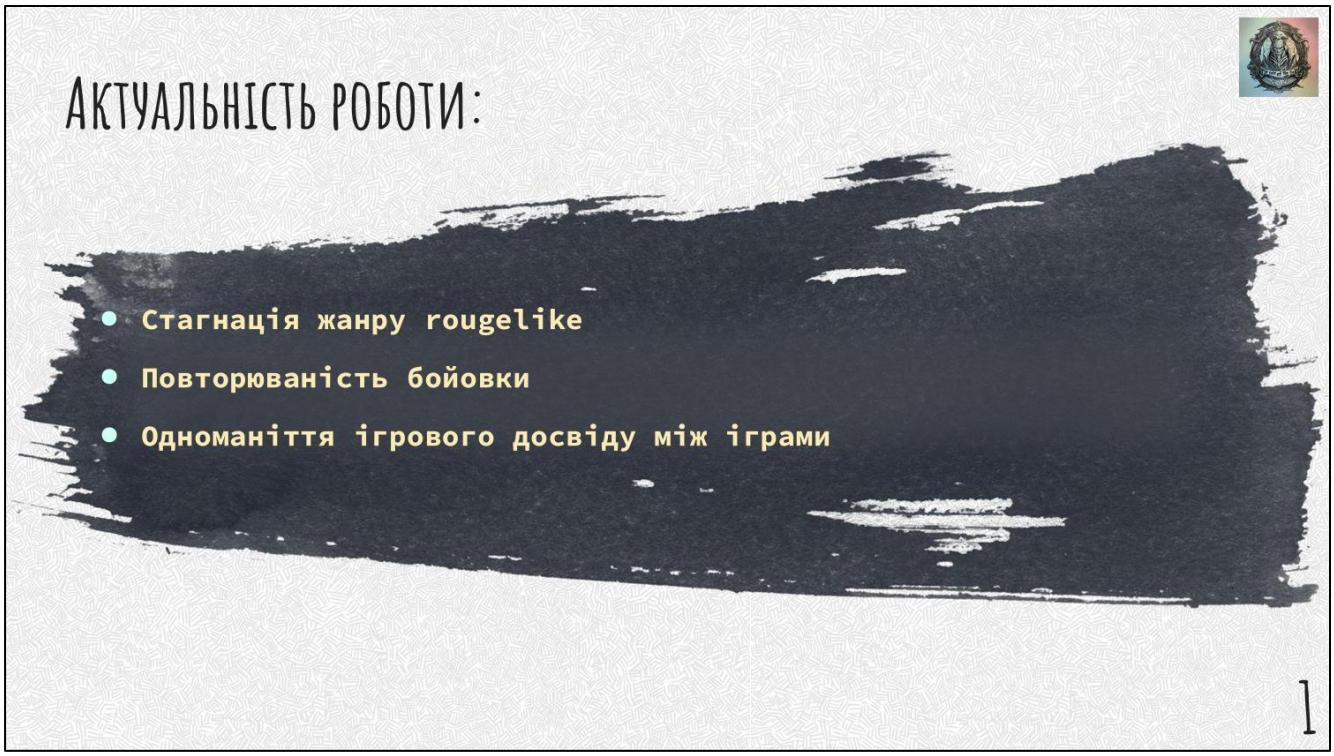


Рисунок Б.2 – Другий слайд презентації

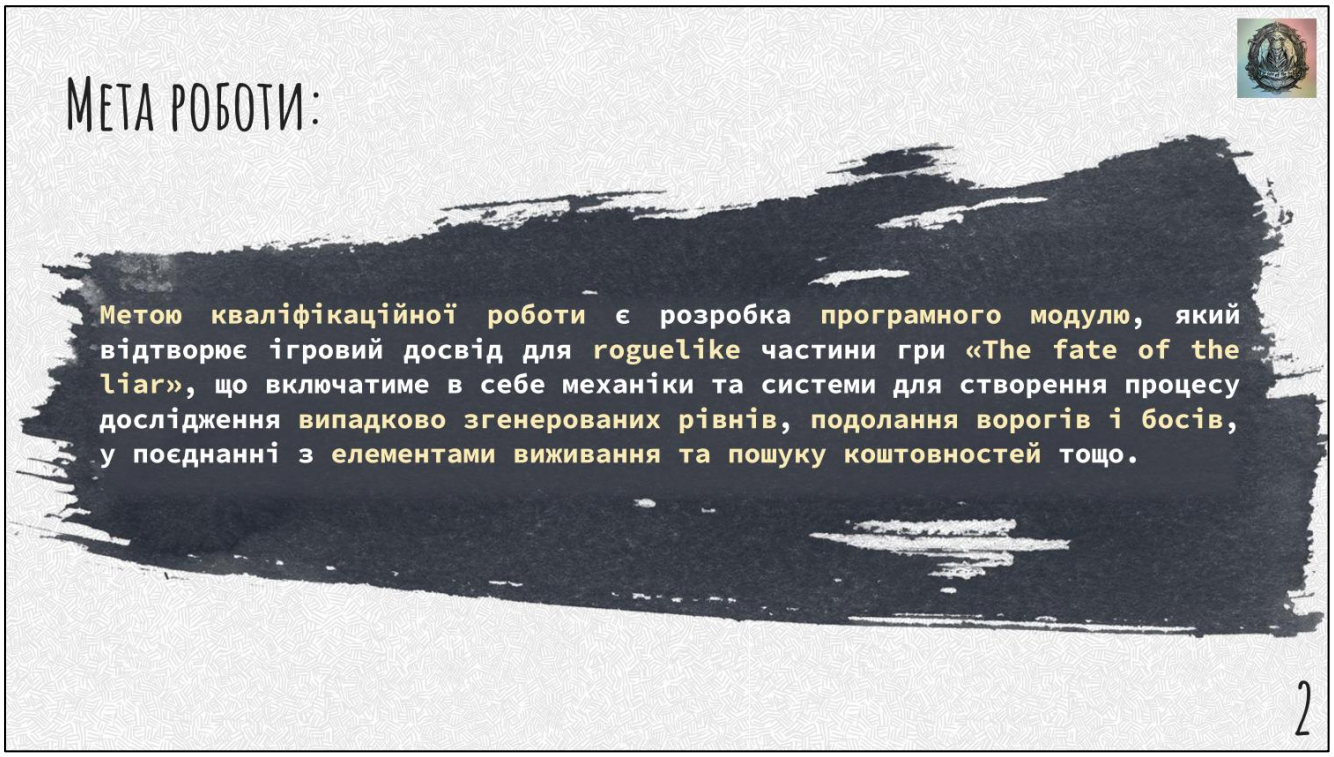


Рисунок Б.3 – Третій слайд презентації

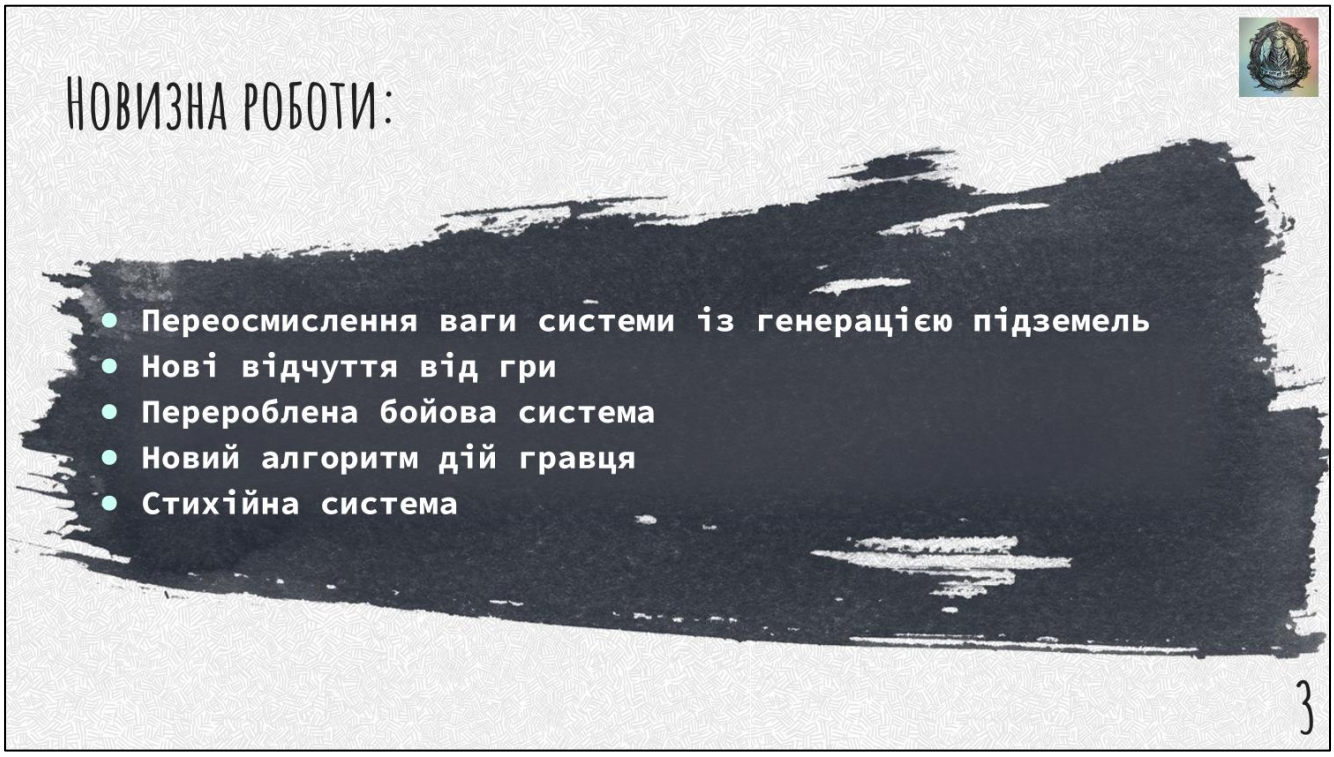


Рисунок Б.4 – Четвертий слайд презентації

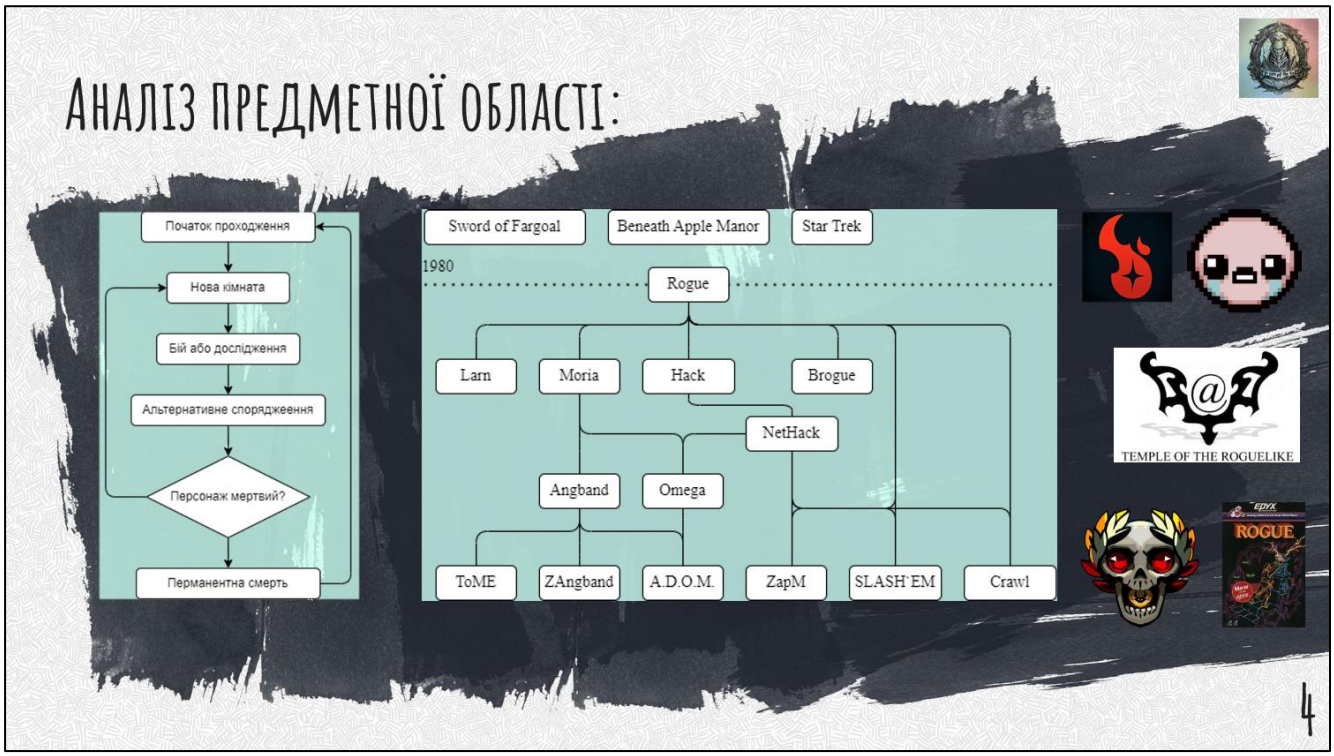


Рисунок Б.5 – П’ятий слайд презентації

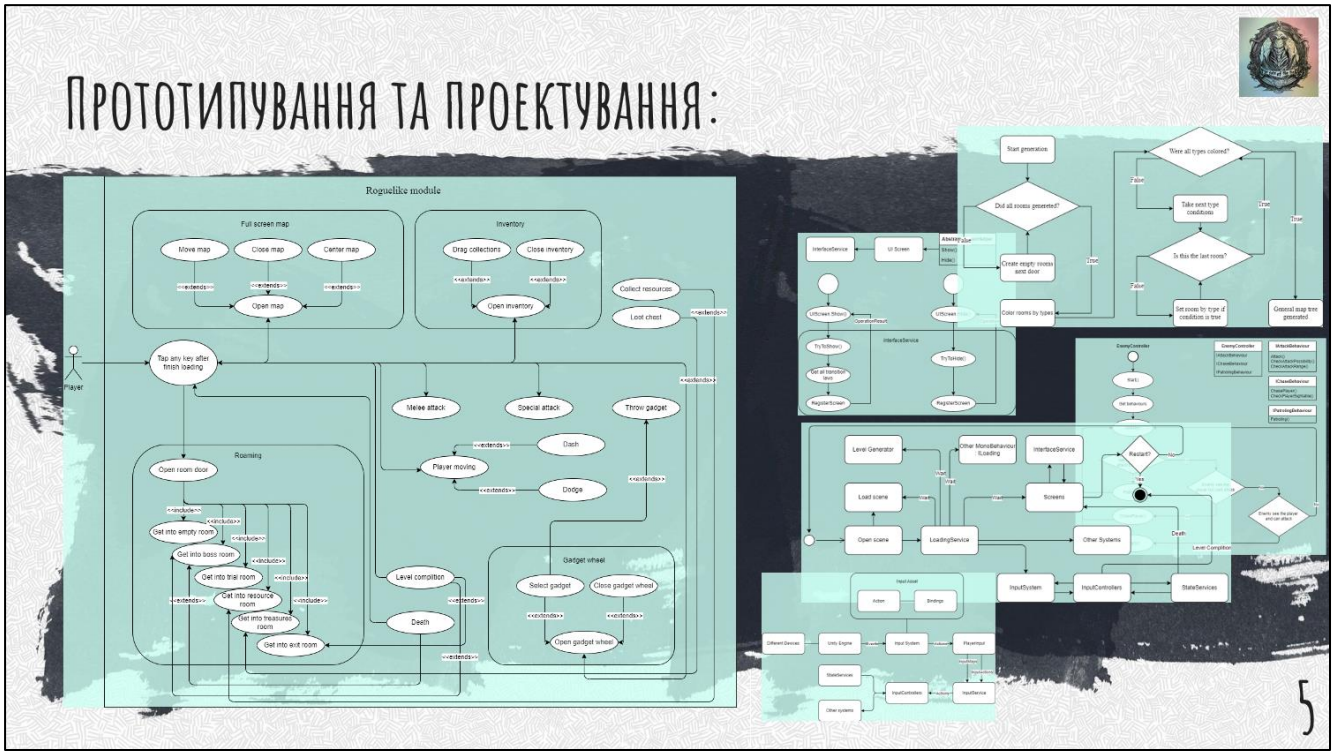


Рисунок Б.6 – Шостий слайд презентації

Рисунок Б.7 – Сьомий слайд презентації

Рисунок Б.8 – Восьмий слайд презентації

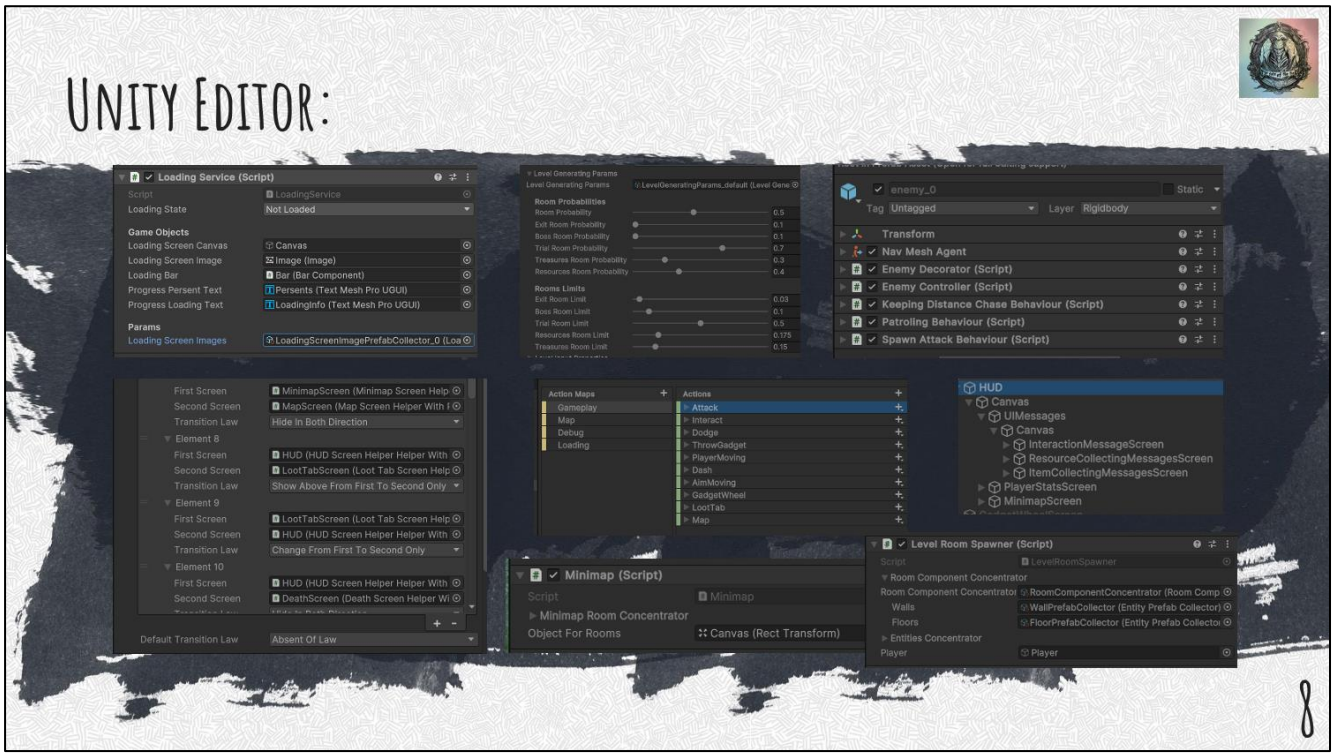


Рисунок Б.9 – Дев’ятий слайд презентації

Рисунок Б.10 – Десятий слайд презентації

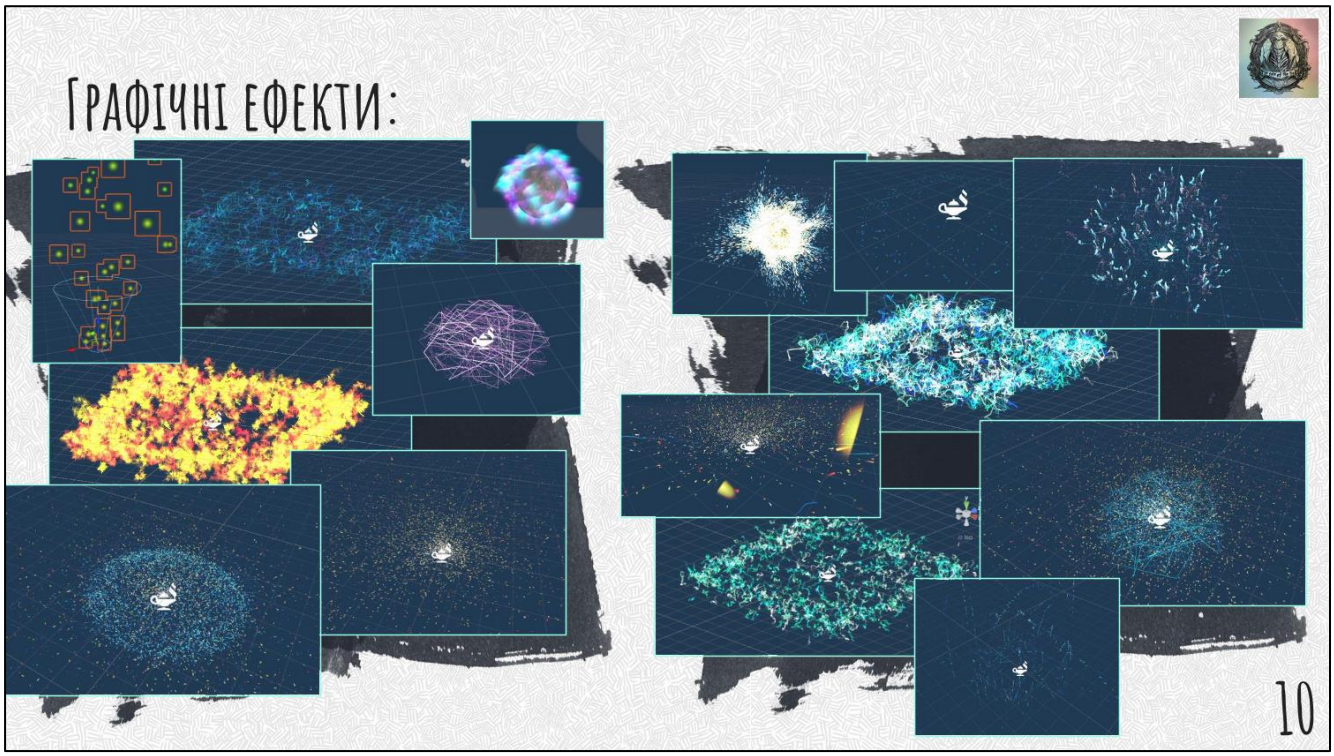


Рисунок Б.11 – Одинадцятий слайд презентації

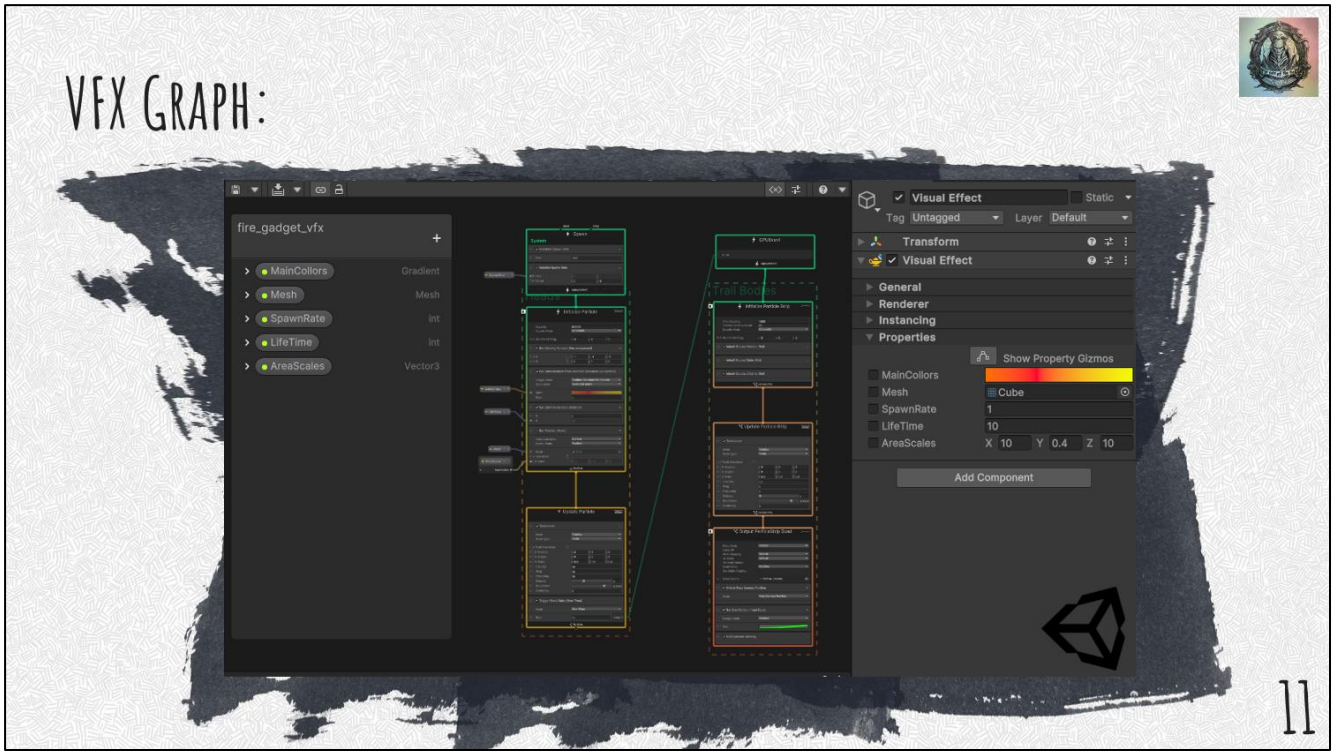


Рисунок Б.12 – Дванадцятий слайд презентації

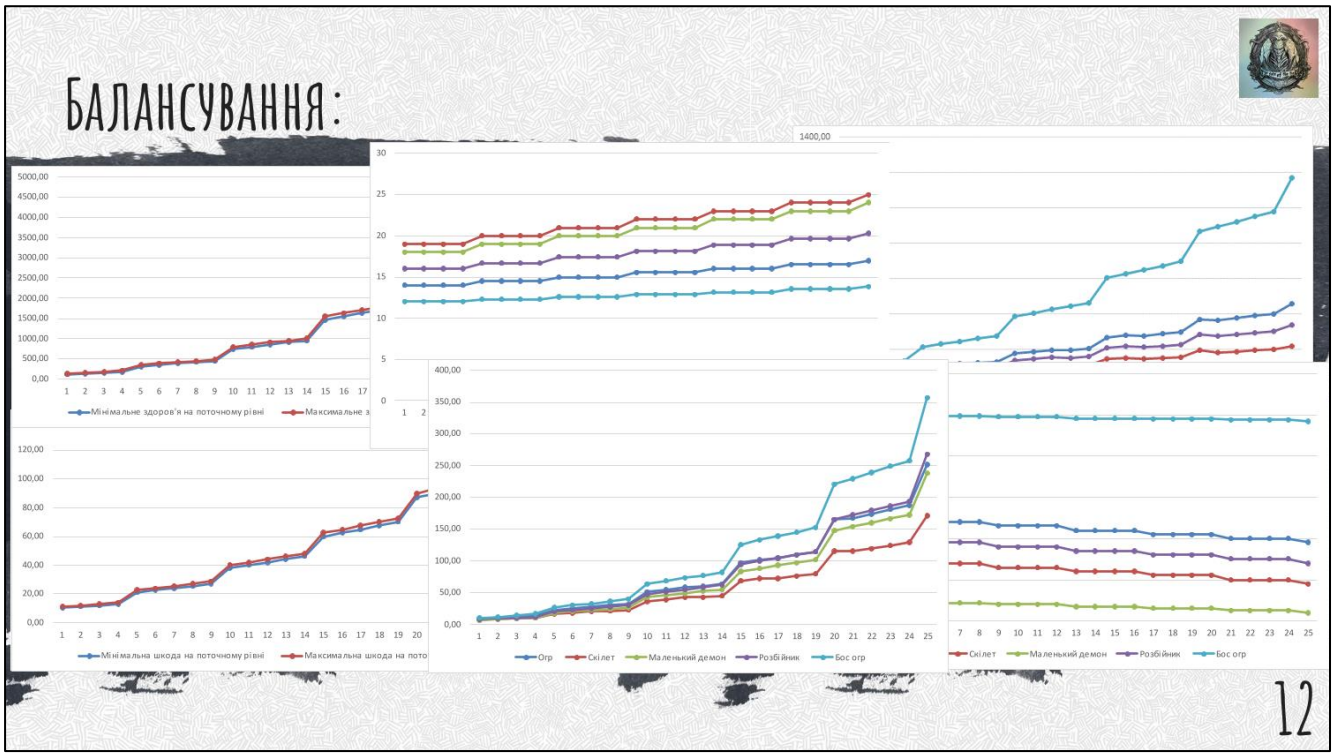


Рисунок Б.13 – Тринадцятий слайд презентації

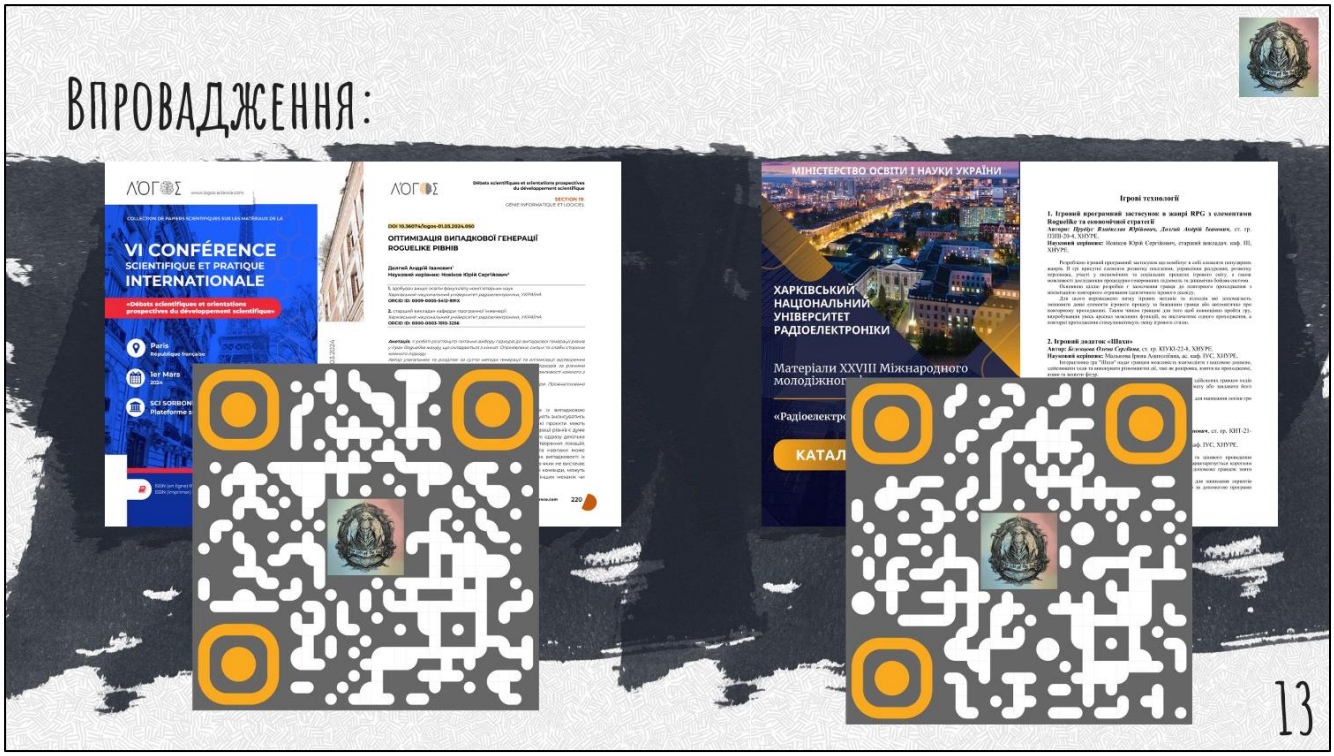


Рисунок Б.14 – Чотирнадцятий слайд презентації

Рисунок Б.15 – П’ятнадцятий слайд презентації

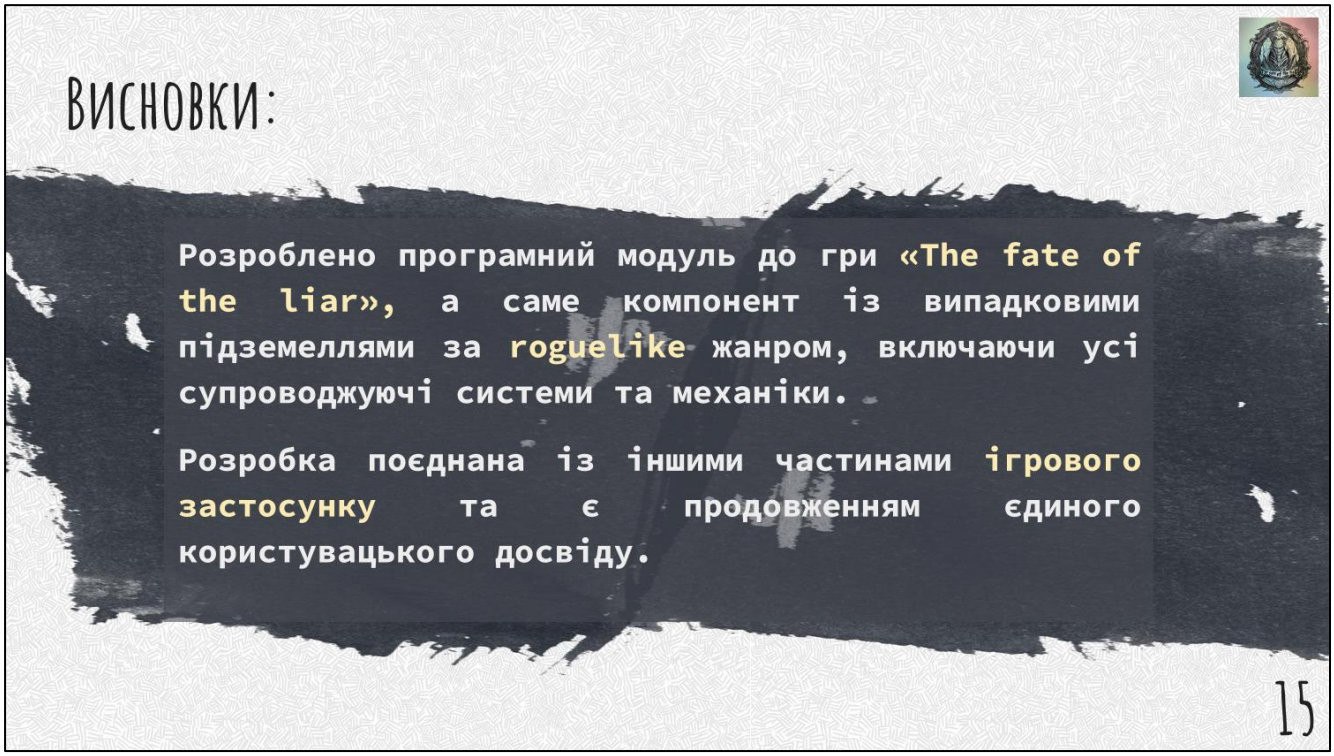


Рисунок Б.16 – Шістнадцятий слайд презентації

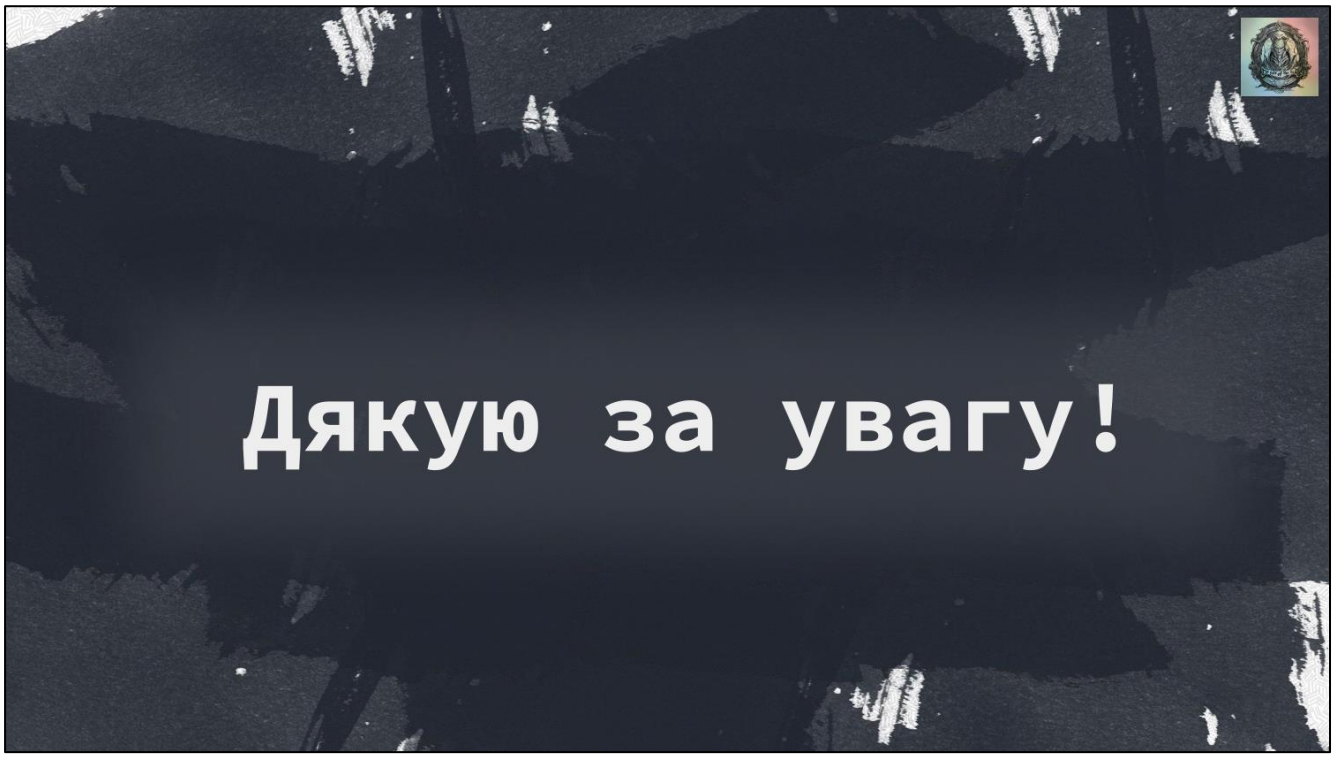


Рисунок Б.17 – Сімнадцятий слайд презентації

ДОДАТОК В

Геймдизайн-документ

THE FATE OF THE LIAR

(Геймдизайн-документ)

Зміст:

1. Тетра
2. Вступ
3. ЦА
4. USP
5. Час ігрової сессії
6. Технологічні характеристики
7. Описання гри
   7.1 Описання ігрового процесу
      7.1.1 Поселення
      7.1.2 Підземелля
   7.2 Механіки
      7.2.1 Механіки створення персонажу
      7.2.2 Механіки хабу
         7.2.2.1 Таверна
         7.2.2.2 Прилегла територія
         7.2.2.3 Народи
         7.2.2.4 Інші механіки
      7.2.3 Механіки “підземель”
         7.2.3.1 Механіки “сюжету”
         7.2.3.2 Механіки дослідження
         7.2.3.3 Механіки, що пов’язані із ворогами та зіткненнями із ними
      7.2.4 Механіки ігрового процесу

Рисунок В.1 – Перша сторінка геймдизайн-документу

7.2.4.1 Механіка карт Таро - пророцтв у “підземеллях”

7.2.4.2 Механіка розсудку

7.2.4.2.1 Механіка швидкого відновлення розсудку

7.2.4.2.2 Механіка потужних навиків

7.2.4.3 Механіки “реактивного” ігрового світу

7.2.4.3.1 Механіка знищення оточення

7.2.4.3.2 Механіка стихійних гаджетів та об’єктів на які вони впливають

7.2.4.3.3 Механіка спротиву ворогів до створених умов

7.2.5 Апаратні механіки

7.3 Референси

7.3.1 Референси UI

7.3.2 Раси

7.3.2.1 Ельфи

7.3.2.2 Гноми - дворфи

7.3.2.3 Орки

7.3.2.4 Низушки - напівролики

7.3.2.5 Люди

Рисунок В.2 – Друга сторінка геймдизайн-документу

1. ТЕТРА

Механіка: RPG з елементами Roguelike та економічної стратегії

Технологія: ПК

Історія: ГГ - простолюдин, обраний

Эстетика: темне фентезі, гротескність, lowpoly

2. ВСТУП

Великий острів на перетині гномових торгівельних шляхів населяють представники усіх можливих рас, стандартних і усім знайомих. Люди, ельфи, гноми, орки, та низушки. Але не такі вже й звичайні, як здавалося б на перший погляд… Низушки - ці малі та пруткі істоти - кровожерливі розбійники та вбивці. Орки - племена лицарів в блискучих латах що живуть за власним кодексом честі. Ельфи - кочовий лісний народ гуманістів, та геніїв технічного прогресу. Гноми - морські авантюристи та торговці, основа міського населення відомої частини світу. І наприкінці люди… Залишки найдавнішої раси, яка колись була майже володарем світу, але приречена втратити свій вплив, через прокляття, накладене на них могутніми чарівниками минулого. Це прокляття зобов’язує кожну людину завжди казати правду.

В цьому божевільному світі, на цьому старовинному острові, де у відносному спокої та рівновазі співіснують всі раси, один хлопець після втрати батька вирішує докорінно змінити своє життя. Отримавши у спадок невеличку таверну на перетині трактів, та величезну особливість, про яку він ще навіть не здогадується, хлопець поклявся, що його ім’я буде на вустах у всього острова. Але хто знає, яких йому сюрпризів підкине невблаганна доля!

3. ЦА

Основна цільова аудиторія проекту - це люди обох полів з переважанням чоловічої частини, від 18 до 35 років. Різні підходи для проходження гри та

Рисунок В.3 – Третя сторінка геймдизайн-документу

пов’язана з цим гнучкість та можливі повторні перепроходження гри, наряду з цікавою історією, моральними виборами прокачкою персонажа та міста, менеджментом ресурсів та можливістю відіграшу конкретної ролі на острові неодмінно приверне увагу користувачів.

4. USP

1) Ігрові механіки, що дають гравцеві можливість певним чином впливати на оточення, змінюючи довколишні умови;
2) Сукупність окремих ігрових механік, що урізноманітнюють кожну ситуацію, у якій перебуває гравець;
3) Впровадження сучасних технологій ШІ для створення ігрового контенту та підвищення реграбельності;
4) Впровадження механік, що симулюють міжособистісні та економічні стосунки для створення унікальності ігрового процесу на рівні історії;
5) Цікава комбінація механік RPG та економічних стратегій, що надає широкі можливості для експериментів та різноплановості ігрового процесу;
6) Моральні вибори, засновані на зловживанні гравцем здібностями персонажа та його взаємовідносинами з навколишнім світом;

5. ЧАС ІГРОВОЇ СЕССІЇ

Так як гра проектується для персональних комп’ютерів, то час ігрової сесії є досить великим, мінімальний поріг приблизно 60 хвилин. За цей час гравець має змогу пройти одне “підземелля” для отримання ресурсів та підтримки взаємовідносин з фракціями.

6. ТЕХНОЛОГІЧНІ ХАРАКТЕРИСТИКИ

Випуск гри планується на платформі персональних комп’ютерів, з наступними характеристиками (за основу взято характеристики аналогів).

Рисунок В.4 – Четверта сторінка геймдизайн-документу

Мінімальні:

– Потребує 64-бітних процесора та операційної системи
– ОС: Windows 7 or later
– Процесор: Intel Core i3-3240 (2 * 3400); AMD FX-4300 (4 * 3800)
– Оперативна пам’ять: 4 GB ОП
– Відеокарта: GeForce GTX 560 Ti (1024 VRAM); Radeon HD 7750 (1024 VRAM)
– Місце на диску: 4 GB доступного місця

Рекомендовані:

– Потребує 64-бітних процесора та операційної системи
– ОС: Windows 10
– Процесор: Intel Core i5-3470
– Оперативна пам’ять: 8 GB ОП
– Відеокарта: GeForce GTX 1050 (2048 VRAM); Radeon R9 380 (2048 VRAM)
– Місце на диску: 4 GB доступного місця

7. ОПИСАННЯ ГРИ

7.1 ОПИСАННЯ ІГРОВОГО ПРОЦЕСУ

Перед початком нової історії гравцеві пропонується створити нового персонажа та кастомізувати його характеристики, базуючись на власному розподілі очок або скориставшись більш просунутою системою формування персонажа.

Ця система пропонуватиме набір історій, пов’язаних з минулим персонажа, на основі яких базуватиметься розподіл очок характеристик. Обравши одну з запропонованих історій, гравець отримуватиме сформований розподіл характеристик, що напряму впливатиме на ігровий процес.

Рисунок В.5 – П’ята сторінка геймдизайн-документу

Сам же ігровий процес можна умовно розділити на дві частини управління посленням та дослідженням підземель.

7.1.1 ПОСЕЛЕННЯ

Головний герой стає на чолі невеликого поселення, що росте навколо напівзабутої таверни де буде відбуватись основний розвиток персонажу. Для цього існує декілька активностей.

По-перше - налагодження взаємозв'язків з різними народами, що дозволяє поглибитись в їх культуру та дізнатись більше про їх технології. Відповідно покращуючи відносини гравець відкриває все більше і більше доступних таємниць народів, таких як спорядження, нові будівлі, активні уміння, секрети та потужні особливі технології. Налагоджувати ці відносини гравець може виконуючи у підземеллях завдання, для кожного з народів.

По-друге - розбудова власного поселення, задля видобутку ресурсів, отримання доступу до тих чи інших елементів спорядження та для отримання постійних бонусів.

І наостанок - просуваючись по сюжету гравець отримуватиме вплив та впізнаваність у світі, що матиме вплив не лише на характеристики гравця, а й на ігровий процес. Також поряд з впізнаваністю гравець може зіткнутися з потужними прокляттями, якими Доля "нагороджує" гравця за негативний вплив на світ.

7.1.2 ПІДЗЕМЕЛЛЯ

У процесі просування грою, головний герой буде зустрічати усе більше персонажів, що будуть просити по допомогу у ділах, пов'язаних із віддаленими та небезпечними для звичайних людей місцями.

Окрім того, по ходу розвиття власного поселення герою буде все більше бракувати різноманітних ресурсів та рідких скарбів, що у великому об'ємі зберігаються у найнебезпечніших кутках континенту, які головний герой тільки може відвідати.

Рисунок В.6 – Шоста сторінка геймдизайн-документу

У підземеллях граця чекатиме важкі баталії, за які, головний герой отримає належне винагородження. Кожна битва буде унікальною, завдяки розгалуженій та варіативній прокаці вмінь протагоніста та системи “реактивного” світу, котрий реагує на певні дії користувача, та змінює умови оточення. Окрім “реактивності” оточення у підземеллях, самі підземелля, які досліджує герой, мають неповторюванну будову, яка змінюється кожного разу, як він відвідує їх.

Герой зустрінеться із різноманітними створіннями за формою та подобою, із особливими схемою поведінки та реагування на дії гравця та зміни оточення ним. До того ж, наповнення підземель може змінюватись через дії героя та його вибори за сюжетом.

У тому, щоб дослідження підземель героєм було успішне та він не загинув беславною смертю у всіми забутим куті континенту, йому буде допомагати стара карга віщунка, яка за допомогою зазирання тканину часу та миробудови допоможе герою досягнути бажаної цілі.

7.2 МЕХАНІКИ

7.2.1 МЕХАНІКИ СТВОРЕННЯ ПЕРСОНАЖУ

1) При вході в гру гравцеві пропонується почати нову історію, що веде за собою створення нового персонажу;
2) При створенні персонажу, йому надається ім’я, обирається стать та визначаються основні його характеристики;
3) Базовими характеристиками є сила, спритність, вдача, харизма та розсудливість. Гравцеві надається 10 очок характеристик, які він може розділити на свій розсуд, але не більше 5 очок на один показник;
4) Для полегшення розподілу очок, гравець зможе обрати одну з запропонованих карток з історією персонажу, які характеризуватимуть його. При виборі картки очки характеристик відповідним чином розподіляються. Цей розподіл, за потреби можна відкоригувати вручну;

Рисунок В.7 – Сьома сторінка геймдизайн-документу

5) Сила - це одна з базових характеристик персонажа, вона відповідає за кількість нанесеного урону супротивникам, а також за потужність деяких умінь;
6) Спритність - відповідає за швидкість переміщення, вірогідності критичних ударів та уникнення урону;
7) Вдача - впливає на перевірки удачі персонажа (шанс отримати + 1 до кидка кубика), а також на деякі вибори в діалогах та успішність брехні;
8) Харизма - основна характеристика при взаємодії з іншими персонажами, відповідає за приріст відносин з народами та за нагороди за виконання квестів;
9) Розсудливість - відповідає за максимальну кількість, втрати та поповнення розсудку при проходженні підземель;
10) Обрані характеристики залишаються незмінними протягом усієї гри, але можуть бути покращені за допомогою спорядження або пасивних навичок, причому при досягненні 5 очок, подальші просування не матимуть ефекту.

7.2.2 МЕХАНІКИ ХАБУ

7.2.2.1 ТАВЕРНА

1) Хаб представляє собою таверну та прилеглу до неї територію, яка буде забудовуватись гравцем;
2) Таверна відіграє роль головного меню хабу, що надає доступ до статистики розподілу ресурсів, ринку ресурсів, інвентарю та спорядження персонажу та найголовніше доступу до відслідковування відносин з народами та прокачки персонажу та поселення;
3) За менеджмент ресурсів відповідає окреме вікно, зі списком ресурсів, їх використанням та надходженням, детальним переглядом ресурсу, з аналізом каналів надходження та витрат, а також ринком, де можна налаштувати автоматичний збут або купівлю ресурсу;

Рисунок В.8 – Восьма сторінка геймдизайн-документу

4) Вікно інвентарю надає функціонал для менеджменту спорядження, де можна подивитись список наявного екіпірування, продати надлишки та спорядити персонажа у похід;
5) Індикатор відносин з народами, також присутній у таверні у вигляді 5 конечної зірки з індикаторами, що відображають рівень відносин з кожним народом, та відображення поточного народу з яким найкращі стосунки;
6) При натисканні на якийсь з описаних вище індикаторів, відкриватиметься дерево технологій окремого народу, з більш деталізованим індикатором прогресу відносин, що поділений на 25 зон, кожна з яких надає одне очко відносин, яке можна витратити на прокачку вмінь;
7) Дерево технологій включає в себе різного виду уміння: активні уміння 3 - покращення гаджетів та 6 звичайних умінь, будівлі - 9 од., цінності - 3 од. та секрети народу - 3 од, що розділені на три рівні доступу;
8) На першому рівні доступу знаходяться 3 будівлі, та 2 уміння, і він доступний одразу. Другий рівень займають 3 будівлі та 2 уміння, цей рівень потребує витратити 5 очок на дослідження цього народу. Останній рівень потребує вже 15 витрачених очок та містить 3 будівлі та 2 уміння. Також кожен рівень містить по одному потужному вмінню, цінності та секрету. При відкритті рівня, усі наявні на ньому уміння стають доступні для розблокування та коштують по одному очку;
9) Активні уміння - це прийоми, які може використати гравець під час проходження підземель, які надають бонуси чи проводять спеціальну атаку, мають час для перезарядки. Одночасно можуть бути екіпіровані у інвентарі у кількості до 3 одиниць;
10) Покращення гаджетів - це рідкісні техніки та технології, що характерні кожному з народів та пов’язані з різноманітними стихіями.

Рисунок В.9 – Дев’ята сторінка геймдизайн-документу

Використовуються також у підземеллях, несуть суттєвий вплив на ігрову ситуацію.

11) Будівлі є декількох видів: бонусні - надають постійні бонуси до характеристик персонажа; виробничі - майстерні, що переробляють базові матеріали в більш просунуті; магазини - продають спорядження, характерне народові. Після прокачки вміння, будівлю слід побудувати;
12) Цінності - показують перейняття способу життя того чи іншого народу гравцем. Цінності - це своєрідні умови, що не зобов'язують до виконання, але у разі виконання, надаватимуть бонуси. Може бути активною лише одна цінність, обирається у інвентарі;
13) Секрети - показник зближення гравця з народом, чим він ближчий до них, тим більше секретів знає. Секрет - це особливе знання народу, яке дає постійний невеликий бонус, але може бути розповсюджене, за допомогою дошки оголошень, що ввімкне альтернативний режим бонусу, що надасть збільшений бонус, але і накладе негативний ефект;

7.2.2.2 ПРИЛЕГЛА ТЕРИТОРІЯ

1) На території прилеглій до таверни, розташовані деякі фіксовані стаціонарні об'єкти а також площі для розміщення будівель;
2) Стаціонарні будівлі включають у себе палатку ворожки, дошку оголошень та ворота. Палатка ворожки є одним з ключових елементів, відповідає за просування у сюжеті, трактуванні карт та відродженні після смерті. Дошка оголошень містить списки завдань, та несе можливості до розповсюдження секретів. Ворота надають перехід до можливостей вибору підземель та відповідно їх проходження;
3) Для розміщення будівель, присутній розподіл на комірки для будівництва, які ділять локацію на зони, на яких розміщуються будівлі;
4) Для розміщення будівель, необхідно мати для неї місце та ресурси. Чим корисніша будівля, тим більше ресурсів витрачається на її будівництво;

Рисунок В.10 – Десята сторінка геймдизайн-документу

5) Загальна кількість комірок на локації значно менша за кількість можливих будівель, що зобов’язує гравця доцільніше використовувати площу та жертвувати тими чи іншими бонусами. Також, відповідно до цього, гравець може знести ту чи іншу будівлю, і вивільнити місце для нових;
6) Побудова будівлі потребує ресурсів та коштів, і найбільш важливе - дослідження відповідної технології. Також, деякі будівлі можуть бути покращені, що також потребує ресурсів, коштів та відповідних технологій, але надає новий функціонал;
7) Для розміщення будівель необхідно перейти у режим будівництва, за натисканням на відповідну кнопку, що показує на екрані, список доступних будівель, а також зони для розміщення будівель;
8) Після розміщення будівлі, з нею можна почати комунікацію, за натисканням на неї. При цьому з’являється модальне вікно відповідно до типу будівлі, з якою проводиться взаємодія.

7.2.2.3 НАРОДИ

1) Під народами мається на увазі 5 фракцій які панують у світі в якому проходять події, це люди, орки, ельфи, низушки та гноми. Представники цих народів видаватимуть завдання гравцеві, з ними ж будуть розвиватись взаємовідносини, від яких залежить розвиток гравця та поселення;
2) Люди - народ що втратив свою велич. З особливостей - орієнтованість на видобуток ресурсів, для цього є відповідні будівлі секрети та цінності. Спорядження має середні показники з ухилом у ближній бій. Щодо умінь, то це в основному активні бойові прийоми ближнього бою, потужні уміння пов’язані з дослідженням локацій (стихія - світло, гаджет - пасивне вміння, що дозволяє продивлятись умови наступної кімнати у “підземеллі”);

Рисунок В.11 – Одинадцята сторінка геймдизайн-документу

3) Орки - лицарський орден. Особливості - бойові бонуси та потужні магазини спорядження, а також складні але потужні цінності. Спорядження - повільний ближній бій та високий захист. Уміння - тимчасові бойові бонуси різного типу з руйнівними атаками в якості потужних умінь (стихія - вода, гаджет - невідома речовина що гарно проводить ток, тушить вогонь та легко замерзає)
4) Гноми - торговці мандрівники. Особливості - торгівля та виробництво, секрети з потужними бонусами та негативними ефектами. Низький захист але велика швидкість. Вміння - комплекс швидких прийомів фехтування які підсилюють один одне а потужні уміння - пов’язані з нафтою та вогнем (стихія - вогонь, гаджет - вогняна вода, що підпалює усе чого торкається);
5) Низушки - розбійники і пірати. Особливості - бонуси до базових характеристик, виробництво коштовних ресурсів, бонусне золото комбінований ближній та дальній бій. Вміння - комбінація ближніх та дальніх прийомів. Потужне уміння - використання різноманітних отрут (стихія - льод, гаджет - таємний лід, що заморожує будь яку рідину);
6) Ельфи - інженери і філософи. Ускладнені ланцюги виробництва, дороге високотехнологічне спорядження та бонуси до збору ресурсів. Тільки дальній бій. Уміння пов’язані з уроном по площі та уникненням урону. Потужні уміння експлуатують електрику (стихія - електрика, гаджет - електричний пристрій з полем блискавок)

7.2.2.4 ІНШІ МЕХАНІКИ

1) Завдання - основа розвитку та сюжету, їх гравець отримує на дошці оголошень. Нагородами за виконання квесту є ресурси, спорядження, відомості та стосунки з народами в тому числі від’ємні;

Рисунок В.12 – Дванадцята сторінка геймдизайн-документу

2) Завдання отримуються та приймаються до виконання на дошці оголошень. Задання не прийняте, буде знаходитись там допоки воно не прийметься або не відміниться грою внаслідок виборів гравця;
3) Важливою складовою також є показник відомості, що є трекером прогресу в контексті історії. Для відстеження гравцем цього показника, присутній індикатор у хабі. Цей індикатор розділений на 5 зон, які відповідають за статус прийняття світом головного героя. Кожна з цих зон має свої особливості, про які можна дізнатись у ворожки;
4) При досягненні максимального рівня відомості, на дошці стає доступним один з 5 фінальних квестів, в залежності від того з яким народом у гравця найвищий рівень відносин. Проходження цього завдання завершує гру, але може бути відкладене гравцем на невизначений термін;
5) Ще однією важливою механікою є трекер брехні. За лором гри, головний герой - єдина людина у світі яка може казати неправду. Ця особливість може бути використана для під час виконання завдань, та діалогів. Збрехавши гравець може полегшити собі життя в конкретний момент, але підніме рівень брехні що впливатиме на його долю;
6) Підвищення рівня брехні відслідковується грою. При підвищенні показника до однієї з 3 відміток на гравця накладають випадкове прокляття у вигляді карти таро, яка накладає обмеження. Негативні обмеження від карток сумуються, тобто при досягненні 3 відмітки у гравця буде аж 3 прокляття;
7) Показник брехні може бути знижений при достатній кількості чесних відповідей, при чому зменшивши цей показник нижче за відмітку, її прокляття знімається;
8) Слід звернути увагу також і на ресурси. Базовим ресурсом тут є золото, яке може бути використане як для конвертації в інші ресурси, так і для покупки спорядження і будівництва. На рівні з золотом є також 25 інших

Рисунок В.13 – Тринадцята сторінка геймдизайн-документу

ресурсів які також використовуються. для продажу, будівництва а також витрачаються на роботу деяких будівель;

7.2.3 МЕХАНІКИ “ПІДЗЕМЕЛЬ”

“Підземелля” - це абстрагована назва рівнів, на яких гравець зможе власноруч досліджувати локації, збирати необхідні йому ресурси, битися із ворогами, виконувати сюжетні та не зовсім доручення тощо. Зазначені рівні, будуть генеруватись динамічно та за певним алгоритмом, що має на меті створювати унікальні локації та умови для головного героя кожного разу, коли він опиняється у “підземеллях”. Загальна механіка “підземель” включає у себе реалізацію жанру **Roguelike**, що має на увазі підхід випадкового генерування рівнів.

Крім основної механіки, що включає в себе процес генерування випадкових рівнів та їх наповнення тощо, слід відзначити низку допоміжних механік, що будуть наповнювати створені підземелля та урізноманітнювати ігровий процес.

7.2.3.1 МЕХАНІКИ “СЮЖЕТУ”

Самі підземелля, можуть слугувати двом цілям, збору додаткових ресурсів, та проходження різних сюжетних доручень. Отже рівні, що спрямовані на наратив, окрім випадкових ділянок, повинні мати сегменти, що пов’язані із завданням та ціллю походу головного героя.

Окрім цього, на безсюжетних, тобто ресурсно орієнтованих рівнях, також будуть виникати певні ділянки, на яких випадкові персонажі, будуть давати невеликі опціональні доручення, розповідати лорні історії, давати підказки, продавати чи обмінювати речі та ресурси тощо. Однак такі кімнати будуть лише мати шанс на генерацію, та не будуть обов’язковими, у порівнянні із сюжетними на відповідних рівнях.

Рисунок В.14 – Чотирнадцята сторінка геймдизайн-документу

7.2.3.2 МЕХАНІКИ ДОСЛІДЖЕННЯ

Згенеровані локації, мають складатися з окремих підлокацій - “кімнат” або “пазлів” з яких будується увесь рівень. Ці кімнати мають поділятись на декілька типів, а саме:

1) кімнати із скарбами та пастками;
2) кімнати із ресурсними жилами, сховищами тощо;
3) “сюжетними” кімнатами;
4) кімнатами із ворогами та босами.

До того ж, у доповнення до RPG механік, кімнати матимуть “перевірки” показників персонажу на можливість виникнення певних ситуацій, що повинно збільшити рівень рольового занурення та покращити варіативність ігрового процесу. Так наприклад успішна перевірка спритності допоможе уникнути пастки у кімнаті зі скарбами, а у кімнаті із ресурсами, при провалі, навпаки призове ворогів, тощо. Зазначену механіку перевірок, планується реалізувати на основі аналітичної механіки у D&D.

7.2.3.3 МЕХАНІКИ, ЩО ПОВ’ЯЗАНІ ІЗ ВОРОГАМИ ТА ЗІТКНЕННЯМИ ІЗ НИМИ

Вороги плануються різноплановими створіннями, кожне з яких володіє своєю унікальною ціллю на полі бою. Хтось має атакувати здалеку, хтось витримувати всю шкоду для стримування наступу головного героя, хтось встановлює пастки тощо.

До того ж, Зазначена механіка тісно пов’язана із наступним пунктом та минулими. Так кількість та різновид противників залежить від прогресу сюжету та виборів гравця, так само і з прокачкою. Окрім цього, певні батальні сцени можна уникнути або навпаки спровокувати, все це буде залежати від результатів перевірок головного героя.

Рисунок В.15 – П’ятнадцята сторінка геймдизайн-документу

7.2.4 МЕХАНІКИ ІГРОВОГО ПРОЦЕСУ

Як вже зазначалося вище, ігровий досвід користувача поділений на два загальних напрямки, а саме:

- розбудова власного поселення;
- дослідження та проходження “підземель”.

Перший пункт вже було описано вище. Що стосується другого пункту, то частина із самими “підземеллями” була так само описана, отже час розглянути можливості саме гравця, на цих рівнях, та пов’язані із цим механіки.

7.2.4.1 МЕХАНІКА КАРТ ТАРО - ПРОРОЦТВ У “ПІДЗЕМЕЛЛЯХ”

При початку нової спроби пройти певне “підземелля”, гравцеві гра обере та продемонструє декілька карт Таро, ефект яких для гравця буде не відомий та про який він може тільки здогадуватись. Для кожної такої карти, буде певне випробовування, при виконанні якого, карта Таро розкриє свій сенс гравцеві, та вносить відповідні зміни до умов поточного “підземелля”. Зазначені ефекти можуть бути як позитивними, тобто якось допомагати головному герою, так і створювати йому перешкоди та заважаючи йому - негативні ефекти.

Зазначені карти, а саме їх дизайн мають бути створені за допомогою штучного інтелекту, а ефекти та випробування заздалегідь мають бути прописані та зафіксовані за певними картами, для збереження автентичності та наративної складової.

Наявність карт Таро додатково обґрунтована лором та наративним дизайном, оскільки все що відбувається у “підземеллях” - це лише передбачення карги.

7.2.4.2 МЕХАНІКА РОЗСУДКУ

Як вже відзначалось вище, з наративної сторони, проходження “підземель” відбувається у просторі передбачень каргою-гадалкою, що демонструє потенційне майбутнє, через що, гравець буде втрачати саме ментальне здоров’я - розсудок,

Рисунок В.16 – Шістнадцята сторінка геймдизайн-документу

замість звичайного. Сам розсудок буде втрачатися при отриманні шкоди головним героєм під час дослідження підземель та повертатись назад при нанесені шкоди ворогам. (Можливе додаткове ускладнення ігрового процесу, де з часом розсудок сам уходить, що сприяє гравця сприймати рішення швидше, та намагатися якнайшвидше покинути рівень, аби зберегти усі здобуті ресурси)

Окрім зазначеної концепції відтворення здоров'я гравця, слід відзначити декілька підмеханік.

7.2.4.2.1 МЕХАНІКА ШВИДКОГО ВІДНОВЛЕННЯ РОЗСУДКУ

У ситуаціях, при яких за маленький час гравець втрачає великий об'єм розсудку, він отримує можливість відновити його, виконавши невелике випробування за короткий термін, на кшталт "нанести комбо з 10 ударів за 15 секунд" тощо.

Зазначена механіка спрямована на можливість гравця виправити скоєну помилку, та націлена на створення нагороди для гравців, що гарно розібрались у механіках та самій грій в цілому.

7.2.4.2.2 МЕХАНІКА ПОТУЖНИХ НАВИКІВ

Ця механіка перетинається з системою навиків, а саме те, що у деревах здатностей, посеред інших пунктів, має бути особливі, надпотужні закляття, прийоми тощо. Ці навички спроможні повністю перевернути поле брані та змінити баланс сил у битві із ворогами. Однак ці прийоми вимагають для використання певного часу підготовки та витрачають велику кількість розсудку героя, що унеможливлює спам цими здатностями, та спонукає гравців залишати їх для особливих моментів, бо інакше головного героя можуть просто вбити вороги, бо в нього буде замало розсудку для боротьби.

Рисунок В.17 – Сімнадцята сторінка геймдизайн-документу

7.2.4.3 МЕХАНІКИ “РЕАКТИВНОГО” ІГРОВОГО СВІТУ

Під механіками “реактивного” ігрового світу мається на увазі системи, що змінюють умови оточення у “підземеллях”. Ці механіки слід розглянути кожну окремо, оскільки кожна з них, має свій напрямок для впливу на ігровий світ.

Додатково слід відмітити, що усі зміни, що гравень може внести у “підземелля”, потребують використання гаджетів, що головний герой розблоковує та вдосконалює при прокачці дерева здібностей.

7.2.4.3.1 МЕХАНІКА ЗНИЩЕННЯ ОТОЧЕННЯ

Першим напрямком, що має бути згадано - є об’єкти, що можна знищити для розблокування додаткових шляхів, створення нових особливих ділянок та умов на рівні. Так гравець може знищити крихкий пол на рівні та спробувати скинути у безодню ворогів, що стояли на цьому місці. Також герой може руйнувати декоративні об’єкти для пошуку додаткових ресурсів чи особливих предметів, що йому можуть знадобитись при розбудові поселення чи при проходженні “підземелля”. Додатково слід відмітити, що деякі проходи у опціональні кімнати спочатку заблоковані та приховані, але можуть бути знайдені, та відкриті, через руйнування відповідних перешкод.

7.2.4.3.2 МЕХАНІКА СТИХІЙНИХ ГАДЖЕТІВ ТА ОБ’ЄКТІВ НА ЯКІ ВОНИ ВПЛИВАЮТЬ

Цей напрямок зміни навколишнього світу схожий на той, що було описано вище, проте він спеціалізується не на руйнуванні, а на створенні перешкод. На рівнях передбачається різноманітні об’єкти, з якими можна стихійно провзаємодіяти.

До стихійних гаджетів можна віднести наступні:

1) склянка вогняної води, що спалахує та підпалює усе, до чого може дотягнутися та може розтопити лід;

Рисунок В.18 – Вісімнадцята сторінка геймдизайн-документу

2) громовий механізм, що створює при активації поле електричних блискавок, що наносить шкоду всім, хто потрапв у поле, та може розширити зону своєї дії, взаємодіячи із водою;
3) склянка невідомої речовини, що не випаровується сама по собі, але легко замерзає тане, проводе блискавки чи навпаки тушить вогонь, зникаючи після недовгої взаємодії із іншими гаджетами;
4) кубік магічного доісторичного льоду, що може затушити вогонь, при цьому створивши водяну калюжу або навпаки зробити з калюжи рідини льодову перешкоду, на якій усі вороги будуть падати та отримувати тимчасовий нокдаун.

Всі ці гаджети взаємодіють саме з рівнем, тимчасове змінюючи оточення на рівні.

7.2.4.3.3 МЕХАНІКА СПРОТИВУ ВОРОГІВ ДО СТВОРЕНИХ УМОВ

Дана підмеханіка поєднує зазначені можливості “реактивного” світу із інтелектом та здібностями ворогів. Цей пункт має на увазі те, що вориги мають бути не просто дурними бовдурами, що йдуть на пролом, а вони мають рухатись згідно власним особливостям та реагувати на створенні умови відповідно до них. Тобто ті, хто можуть літати, ніяк не реагують на створені провали у полі або на підлогу з льоду, а демони в свою чергу не реагують на вогняну перешкоду, големи не сприймають грім тощо.

Поєднання зазначених механік сприяє різноманітності та варіативності проходження та надає великий простір для експериментів гравцем.

7.2.5 АПАРАТНІ МЕХАНІКИ

До апаратних механік або особливостей, можна віднести підтримку технології Haptic Feedback та Adaptive Triggers для геймпадів Dualsence.

Застосування ШІ для реалізації певних елементів інших механік, що розглядалися вище тощо.

Рисунок В.19 – Дев’ятнадцята сторінка геймдизайн-документу

7.3 РЕФЕРЕНСИ

7.3.1 РЕФЕРЕНСИ UI

1) https://interfaceingame.com/games/hollow-knight/
2) https://www.gameuidatabase.com/gameData.php?id=1193
3) https://store.steampowered.com/app/1092790/Inscryption/
4) https://www.gameuidatabase.com/gameData.php?id=1496
5) https://interfaceingame.com/games/frostpunk/
6) https://interfaceingame.com/games/hades/
7) https://store.steampowered.com/app/512470/The_Pirate_Caribbean_Hunt/?l=ukrainian
8) https://www.gameuidatabase.com/gameData.php?id=1549
9) https://www.gameuidatabase.com/gameData.php?id=48

7.3.2 РАСИ

7.3.2.1 ЕЛЬФИ

Після поразки у війні, ельфи мали багато часу на те щоб побути на самоті та розвиватись духовно. Це дало змогу просунути свою давню релігію поклоніння Природі та заснувати декілька філософських течій на її основі. Центральною тезою їх світогляду є гуманізм, у найбільш широкому сенсі, найцінніше для них життя у будь якому її прояві. Нарівні з цим, в Спілці просуваються ідеї про зверхність знання, до якого призводить технічний прогрес, що призводить до розуміння природних процесів та як результат зближення з самою природою. Прихильники обох поглядів знаходяться у постійній дискусії щодо розбіжностей та протиріч двох панівних систем світогляду. У майбутньому це може стати причиною розколу в їх суспільстві.

Основа ельфійської діяльності - інженерія і алхімія та їх поєднання у різних пропорціях. Більшість їх виробів - з природних матеріалів, які можна зустріти в лісах, але за рахунок різноманітних технік та конструкторських підходів, вони можуть збудувати будь що, без використання цвяхів і інших кріплень. Деякі

Рисунок В.20 – Двадцята сторінка геймдизайн-документу

найбільш прогресивні представники ельфійського народу, в основному юнаки та юначки, просувають ідеї співпраці та виходу за межі лісів, що відкрило б дорогу до сучасних матеріалів не тільки лісного походження. Та ці ідеї не мають значної підтримки у більшості що є ще одним наріжним каменем розколу.

Може якось докрутити до ельфів, які відступили від любові природи, та стали безумними вченими. Наприклад додати якийсь стімпанк та об'єднання зовсім вже прогресивних та майже сучасних науковців, вікторіанську стилістику тощо. Або навіть додати нотки більш темного фентезі з наявністю темної магії чи навіть некромантії.

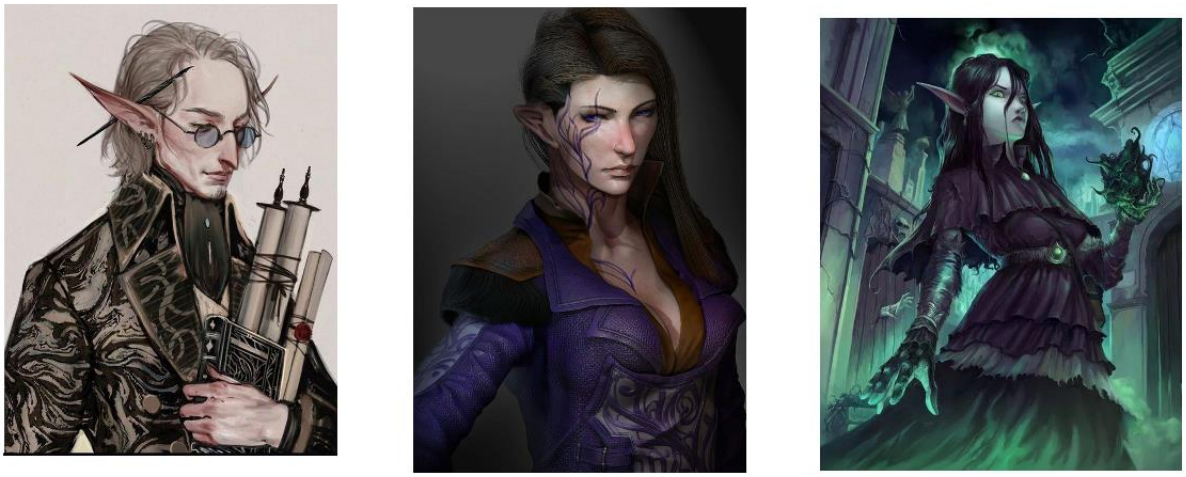


Рисунок В.21 – Двадцять перша сторінка геймдизайн-документу

7.3.2.2 ГНОМИ - ДВОРФИ

Небувало розвинута підприємницька жилка цього народу, ніколи не давала їм сидіти на місці. Гноми прагнули надприбутку, зменшення ризиків та пошуку нових джерел доходу, що значно полегшилось після створення ними системи колоній по всьому світу. Це також зробило їх потужною морською державою, яка щедро спонсує нові відправлення для дослідження нових земель. Головною їх особливістю є хитрість, за допомогою якої вони завжди вибивають собі кращі контракти, утворюють найвигідніші союзи, та разом зі своїм впливом навіть припинюють війни без битв. Хоча вони в цілому можуть обійтись і без цього, адже провідні оборонні ельфійські технологі, в купі з потужними укріпленнями та кращими найманцями, є найкращим аргументом перед боєм.

Можна зробити їх зфокусованими на багатстві, капіталістами-матеріалістами, що тримають увесь світовий ринок у власних лодонях тощо. Додати трохи елементів вікторіанського стилю, стімпанківських деталей та стильових образів для дослідницького та морського напрямку. Також можна затемнити лор тим, що на справді, саме двофи є кукловодами світу та великими політичними інтриганами.

Рисунок В.22 – Двадцять друга сторінка геймдизайн-документу

7.3.2.3 ОРКИ

Колись давно, орки були купкою диких гірських племен, котрі окрім грабунку сіл і торгових обозів, ніяк не комунікували з зовнішнім світом. Але все змінилося, коли Базур Топор отримав владу у своєму племені, і згуртував навколо нього інші. Тоді набіги стали більш систематичними, жорстокішими, та результативнішими, до тих пір, поки Базур не потрапив у полон до гномів, при штурмі їх приграничного замку. 4 роки полону значно змінили його, і після підписання миру і виплати значного викупу, Базур вийшов з за міських стін зовсім іншим орком, реформатором, який змінить орків на все життя.

Добившись від гномів пакту про ненапад, та дозволу селитись у долині річок, орки почали розводити коней, в гірських копальнях не замовкав гуркіт інструментів, а в кузнях стукіт молотів, а виняйняті у людей інструктори безперестанку тренували оркських наїздників - це сформувало основу оркської армії - важку броньовану кінноту. Адміністративні реформи призвели до створення союзу - конфедерації племен, та чітко розділили між ними землі а також місця у Народному Зборі, першому політичному органі орків. Але головним досягненням Базура було створення Кодексу, перші 8 статей якого були проголошені одразу після виходу ним з полону, і стали примірником лицарської поведінки, та

Рисунок В.23 – Двадцять третя сторінка геймдизайн-документу

згуртували орків як націю та змінили їх образ життя. Кодекс доповнювався новими поколіннями, що стало основою прецедентного права орків.

У своїй старості Базур все ж поквитався з гномами, активно підтримавши людей у їх війні і захопивши декілька прикордонних замків, але помер від лихоманки в одному з них, так і не дізнавшись, що війна буде програна, а люди - падуть. Діло Базура живе і понині, і хоча великих війн давно вже ніхто не вів, але його потомки потроху нарощують військову міць та залишаються серйозною стримуючою силою у світі.

Можна зробити з орків гігачадів гачістів, та перенести їх більш до краю континенту - берег, та зробити з них самураїв - тих, хто є дуже плилеглим, врівноваженим та стараним, що як раз дає контраст зі звичайними орками. Зробити з них воїнів честі, які будуть більш закритою расою, тому стиль їх буде відрізнятися. Стилістика, що буде їм притаманна - епоха Едо з японії, а поведінка буде відображати найсильніші якості лицарів з казок (п’ять чеснот з відьмака наприклад) та гармонію у душі, що відображається у їх поведінці. Можливо проповідують, щось схоже на буддизм. Майстри ближнього бою та меча(катани, утігатани, одаті тощо) та можливо підкорили багато тварин та кооперують разом з ними.

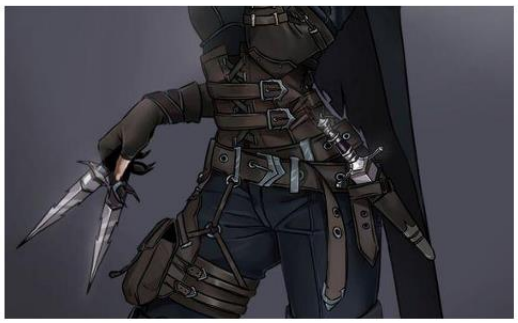
Рисунок В.24 – Двадцять четверта сторінка геймдизайн-документу

7.3.2.4 НИЗУШКИ - НАПІВРОСЛИКИ

Основним заняттям низушків-напівросликів є, як вже зазначалось, рекет, грабунок, піратство, та контрабанда. А останнім часом, вони почали проводити маленькі загарбницькі війни, як наприклад у *Kintondon-i*, що значно розлючує всіх навколо, адже гноми-дворфи поступово втратили будь який контроль над ними. А самі низушки, втратили будь яку повагу до навколишніх адже поваги достойні лише сильні.

Можна додати якийсь культ, тощо, зробити їх фанатиками, що роблять усі свої темні справи на честь забутого бога та своїх предків. Вони займаються пиратсвом, пиятством, розбоями. Живуть на віддалених островах, з досить суровим кліматом та не дуже плодороною землею. Мають стилістику вікінгів. Займаються контрабандою наркотиків з власної місцевості. Відрізняються високою кровожерлівістю та великими амбіціями. Незважаючи на власні габарити, є досить сильними та спритними створіннями, полюбляють використовувати важку двуручну зброю у відкритих боях, та не оминають шансів позбавитись ворогів з темряви, застосувавши скритну зброю.

Рисунок В.25 – Двадцять п’ята сторінка геймдизайн-документу



7.3.2.5 ЛЮДИ

Розпад Імперії ознаменувався періодом людської чесності, та фактичного зовнішнього управління гномами. Через повальну чесність, люди не можуть витримувати конкурентну боротьбу на ринку, заключати вигідні контракти та домовленості, та взагалі вигідно для себе, співпрацювати з кимось. Це швидко второпали гноми, які поставили людей в залежність від своєї торгівлі та зробили людей своїм сільськогосподарським придатком, при цьому навіть досить значно піднявши рівень життя звичайних громадян.

Рисунок В.26 – Двадцять шоста сторінка геймдизайн-документу

Сучасні люди, це в основному простий народ, що працює на ресурсних виробництвах: фермах, садах, у риболовних компаніях, лісозаготівлі, видобутку каміння чи корисних копалин. Більшість з цього звозиться до великих міст, де або продається на ринках або зберігається на складах, для відправки покупцям напряму. Це ставить людей у пряму залежність від гномів, які користуючись своїм положенням контролюють майже всі аспекти життя, але роблять це з розрахунком та максимально м’яко, тому не викликають особливих бунтів і повстань.

Рисунок В.27 – Двадцять сьому сторінка геймдизайн-документу

ДОДАТОК Г

Тест-план

The fate of the liar

«Ігровий програмний застосунок у жанрі RPG з елементами економічної стратегії та roguelike. Механіки генерації рівнів, механіки на рівні підземель»

План випробувань і тестування

(Test Plan)

Версія 1.0

(Version 1.0)

Рисунок Г.1 – Перша сторінка тест-плану

Ігровий програмний застосунок у жанрі RPG з елементами економічної стратегії та roguelike. Механіки генерації рівнів, механіки на рівні підземель	Версія: 1.0
Тест-план	Дата: 24.02.2024
Ідентифікатор документу: ТЕСТ-ПЛАН	

Історія змін (Revision History)

Дата	Версія	Опис	Автор
24.02.2024	1.0	Створення тест-плану	Долгий А. І. Прудіус В. Ю.

Рисунок Г.2 – Друга сторінка тест-плану

Зміст

1 Вступ (Introduction)
1.1 Мета (Purpose)
1.2 Довідкова інформація (Background)
1.3 Галузь застосування (Scope)
1.4 Визначення проєкту (Project Identification)
2 Вимоги до тестування (Requirements for Test)
3 Стратегії тестування (Test Strategy)
3.1 Типи тестування (Testing Types)
3.2 Тестування продуктивності
3.3 Тестування навантаженням
3.4 Тестування користувацького інтерфейсу
3.5 Дослідницьке тестування
3.6 Тестування за сценаріями використання
4 Ресурси (Resources)
4.1 Ролі (Roles)
5 Етапи проєкту (Project Milestones)
6 Кінцевий продукт (Deliverables)

Рисунок Г.3 – Третя сторінка тест-плану

1 Вступ (Introduction)

1.1 Мета (Purpose)

Цей документ має на меті визначення загальних цілей тестування, основних принципів та стратегії, які будуть використовуватися під час тестування програмного продукту. Мета цього документу полягає в забезпеченні якісного підходу до документування процесу тестування ігрового програмного забезпечення що має назву “Fate of the liar”, з використанням систематичного та комплексного підходу.

1.2 Довідкова інформація (Background)

До тестування приймаєтсья проєкт ігрового продукту “Fate of the liar”, що містить в собі низку механік з трьох популярних ігрових жанрів, рольових та стратегічних ігор а також Roguelike.

З механік рольових ігор присунтні розвиток персонажа, квестова система та система екіпірування, які є основними іструментами просування в грі.

Roguelike елементи гри дають можливість гравцеві вивчати світ та приймати участь у динамічних бійках за ресурси та нове спорядження, а також просуватись у історії.

Механіки стратегічних ігор представлені у вигляді управляння ресурсами та поселенням що доповнюють попередні механіки та надають їм додаткових сенсів.

Більш детально із інформацією про проєкт можна ознайомитись у геймдизайн-документі за посиланням: https://docs.google.com/document/d/1Y9_I0Mn20nhqRkoJ-gg1L4RRCzr9LS4yb1twfXA1SHc/edit?usp=sharing

1.3 Галузь застосування (Scope)

У цьому документі наведено основні положення щодо цілей, обсягів та підходів до тестування програмного продукту. Тестування проводитиметься для ігрового застосунку “The fate of the liar” версії 1.0.

Рисунок Г.4 – Четверта сторінка тест-плану

Будуть перевірені наступні компоненти системи:

а) користувацький інтерфейс;

б) коректність роботи впроваджених ігрових механік:

1) створення персонажа;
2) механік поселення;
3) механік розвитку персонажа;
4) механік процедурної - випадкової генерації рівнів;
5) механік дослідження локацій;
6) бойової системи;
7) ворожого штучного інтелекту;
8) стихійної системи;
9) впроваджених підсистем;
10) впроваджених сервісів.

1.4 Визначення проєкту (Project Identification)

У таблиці (див. табл. 1.1) наведено документацію та її готовність, для розробки плану тестування.

Таблиця 1.1 – Стан документації

Документ і версія	Створено	Переглянуто	Автор	Примітки
Геймдизайн-документ v1.2	13.09.2023	24.02.2024	Долгий А. І., Прудіус В. Ю.	
Таблиця контенту v1.1	12.11.2023	24.02.2024	Прудіус В. Ю.	
Балансування roguelike модулю v1.0	20.01.2024	24.02.2024	Долгий А. І.	

Рисунок Г.5 – П’ята сторінка тест-плану

2 Вимоги до тестування (Requirements for Test)

У наведеному нижче переліку виявлено ті елементи, які були визначені як цілі для тестування. У цьому списку детально наведено елементи системи що будуть протестовані:

а) створення персонажа;

б) управління поселенням;

в) управління ресурсами;

г) управління інвентарем;

д) розвиток умінь;

е) генерація рівнів;

ж) переміщення гравця по згенерованим локаціям:

1) переміщення гравця по кімнаті;

2) переміщення гравця між кімнат;

з) бойова система:

1) отримання шкоди;

2) ухилення та уникнення шкоди;

3) застосування звичайної атаки;

4) застосування спеціального вміння;

и) стихійна система:

1) застосування стихійних предметів;

2) взаємодія між стихіями;

3) взаємодія стихій із ворогами;

к) допоміжні механіки;

л) додаткові системи.

При тестуванні цих елементів основна увага приділяється коректності реалізації, відповідності вимогам та коректності відображення за допомогою графічного інтерфейсу користувача.

Рисунок Г.6 – Шоста сторінка тест-плану

3 Стратегії тестування (Test Strategy)

3.1 Типи тестування (Testing Types)

В процесі тестування будуть використовуватись наступні методи:

а) Нефункціональне тестування:
   1) тестування продуктивності;
   2) тестування навантаженням;

б) Ручне тестування:
   1) тестування користувацького інтерфейсу
   2) дослідницьке тестування;
   3) тестування за сценаріями використання.

3.2 Тестування продуктивності

Цілі:

– вимірювання продуктивності гри при різних умовах;
– оцінка частоти кадрів (FPS), часу завантаження та інших ключових показників;
– виявлення вузьких місць, що впливають на продуктивність;

Описання процесу:

– тестування продуктивності, виконання тестів для вимірювання FPS, часу завантаження та інших параметрів;
– тестування стійкості, запуск гри протягом тривалого часу для виявлення накопичувальних проблем;
– аналіз результатів, виявлення та аналіз вузьких місць продуктивності;

Критерії виходу:

– стабільна частота кадрів при різних умовах гри;
– відсутність значних затримок при завантаженні;
– виявлені та виправлені критичні проблеми продуктивності.

Рисунок Г.7 – Сьома сторінка тест-плану

3.3 Тестування навантаженням

Цілі:

– перевірка масштабованості гри під різними рівнями навантаження;
– оцінка продуктивності при максимальних навантаженнях, таких як велика кількість гравців або складні сцени;
– виявлення проблем з використанням системних ресурсів;

Описання процесу:

– тестування з інтенсивними обчисленнями, виконання сценаріїв, які вимагають великої кількості обчислень для оцінки продуктивності;
– тестування тривалості, тривалий запуск гри для виявлення накопичувальних проблем з ресурсами;

Критерії виходу:

– гра стабільно працює при різних рівнях навантаження;
– відсутність проблем з використанням пам'яті, процесором або іншими ресурсами;
– система ефективно масштабується під збільшення навантаження.

3.4 Тестування користувацького інтерфейсу

Цілі:

– перевірка зручності та інтуїтивності користувацького інтерфейсу;
– забезпечення відповідності інтерфейсу стандартам та очікуванням користувачів;
– виявлення проблем з навігацією та використанням інтерфейсу;

Описання процесу:

– перевірка відповідності інтерфейсу дизайновим специфікаціям;
– перевірка коректності роботи елементів інтерфейсу;
– тестування зручності;

Рисунок Г.8 – Восьма сторінка тест-плану

Критерії виходу:

- всі елементи інтерфейсу працюють коректно;
- інтерфейс зручний та інтуїтивно зрозумілий;
- виявлені проблеми виправлені.

3.5 Дослідницьке тестування

Цілі:

- виявлення невідомих проблем та дефектів;
- оцінка стабільності та функціональності системи в непередбачених умовах;
- поглиблене розуміння роботи системи через нестандартні підходи до тестування;

Описання процесу:

- планування дослідницьких тестів, визначення загальних напрямків для дослідження та цілей тестування;
- виконання дослідницьких тестів – проходження гри із використанням неочікуваних сценаріїв;
- документування результатів - запис виявлених проблем та поведінки системи під час тестування;
- аналіз результатів тестування та подальше покращення ігрового застосунку;

Критерії виходу:

- виявлені всі критичні проблеми, які можуть впливати на роботу гри при звичайному використанні;
- Протестовані усі заплановані сценарії гри;
- система стабільна в непередбачених умовах.

Рисунок Г.9 – Дев’ята сторінка тест-плану

3.6 Тестування за сценаріями використання

Цілі:

– перевірка коректності виконання основних функцій ігрового застосунку в типових сценаріях використання;
– оцінка зручності та зрозумілості взаємодії гравців із компонентами гри в реальних умовах;
– виявлення проблем, які можуть виникнути при виконанні звичайних завдань гравцями;

Описання процесу:

– визначення типових сценаріїв гри;
– виконання тестів за визначеними сценаріями;
– звітність, щодо виявлених проблем у заготовлених сценаріях;

Критерії виходу:

– всі основні, визначенні, сценарії використання успішно виконані без помилок;
– виявлені проблеми усунені;
– гра забезпечує зрозумілий досвід користувачам у типових сценаріях використання.

4 Ресурси (Resources)

4.1 Ролі (Roles)

Зазначена таблиця (див. табл. 4.1) показує припущення щодо кадрового забезпечення проєкту.

Таблиця 4.2 - Людський ресурс

Людський ресурс (Human Resources)		
Працівник	Рекомендований	Конкретні обов'язки

Рисунок Г.10 – Десята сторінка тест-плану

(Worker)	мінімальний обсяг осіб (повний робочий день). (Minimum Resources Recommended (number of full-time roles allocated).	або Коментарі (Specific Responsibilities or Comments)
Розробник (Developer)	2	- Розробка програмного забезпечення згідно з вимогами та технічним завданням. - Тестування розроблених компонентів та виправлення помилок.
Проєкт менеджер (PM)	1	- Планування та керування проєктом, включаючи розподіл завдань та контроль строків.
Контентмейкер (Content Creator)	1	- Розробка та створення різноманітного контенту для проєкту.
Художник (Artist)	1	- Створення візуальних елементів, таких як ілюстрації та анімація.
Тестувальник якості (QA)	1	- Виконання тестування програмного забезпечення для виявлення помилок та дефектів.

Рисунок Г.11 – Одинадцята сторінка тест-плану

UI/UX Designer	1	- Розробка дизайну інтерфейсу з урахуванням користувацького досвіду.

4.2 Система (System)

Нижче в таблиці (див. табл 4.2) представлені системні ресурси для тестування проєкту. Системи моделюватимуться з урахуванням виробничого середовища, обмеженого доступу і вимог до забезпечення платформи.

Таблиця 4.2 Системні ресурси

Системні ресурси (System Resources)	
Ресурси (Resource)	Назва (Name) / Тип (Type)
Клієнт випробувань ПК (Client Test PC's)	v. 1.0
Включає спеціальні вимоги до конфігурації (Include special configuration requirements)	Minimum Requirements: Потребує 64-бітних процесора та операційної системи ОС: Windows 7 or later Процесор: Intel Core i3-3240 (2 * 3400); AMD FX-4300 (4 * 3800) Оперативна пам’ять: 4GB ОП Відеокарта: GeForce GTX 560 Ti (1024 VRAM); Radeon HD 7750 (1024 VRAM) Місце на диску: 4GB доступного місця
	Recommended Requirements: Потребує 64-бітних процесора та операційної системи

Рисунок Г.12 – Дванадцята сторінка тест-плану

	ОС: Windows 10 Процесор: Intel Core i5-3470 Оперативна пам’ять: 8GB ОП Відеокарта: GeForce GTX 1050 (2048 VRAM); Radeon R9 380 (2048 VRAM) Місце на диску: 4GB доступного місця
Тестовий репозиторій (Test Repository)	-
Тестовий PC (Test Development PC's)	ОС: Windows 10 DirectX: 11 Процесор: Intel Core i7-9750H Оперативна пам’ять: 32GB Відеокарта: NVIDIA GeForce RTX 2060 Диск: NVMe SSD 1 TB
Link	-

5 Етапи проєкту (Project Milestones)

Тестування включатиме тестові заходи для кожного випробування, визначеного в попередніх розділах. Окремі етапи проєкту зазначені у таблиці нижче (див. табл. 5.1).

Таблиця 5.1 – Етапи проєкту

Цільове завдання (Milestone Task)	Обсяг робіт (Effort)	Дата початку (Start Date)	Дата закінчення (End Date)
План	2д	22.02.2024	24.02.2024

Рисунок Г.13 – Тринадцята сторінка тест-плану

випробувань (Plan Test)			
Тест – дизайн (Design Test)	5д	24.02.2024	29.02.2024
Реалізація випробувань (Implement Test)	7д	29.02.2024	07.03.2024
Виконання тесту (Execute Test)	7д	07.03.2024	14.03.2024
Оцінка випробувань (Evaluate Test)	7д	14.03.2024	21.03.2024

6 Кінцевий продукт (Deliverables)

У результаті виконання тестування розробленого проєкту, мають бути створені наступні документи:

– план тестування;
– тест-кейси;
– тестовий журнал;
– звітність зі знайденими багами.

Усі зазначенні елементи мають бути описані та задокументовані у відповідній кожному формі.

Рисунок Г.14 – Чотирнадцята сторінка тест-плану

ДОДАТОК Д

Приклад програмного коду

Д.1 Користувацький атрибут для полів, що є об’єктами спадкоємцями «ScriptableObject»

Д.1.1 Код класу «InlineScriptableObjectAttribute»

```
#if UNITY_EDITOR
using UnityEngine;

namespace Assets.Scripts.Helpers.InspectorExtensions.Attributes
{
    public class InlineScriptableObjectAttribute : PropertyAttribute
    {

    }
}
#endif
```

Д.1.2 Код класу «InlineScriptableObjectDrawer»

```
#if UNITY_EDITOR
using Assets.Scripts.Helpers.InspectorExtensions.Attributes;
using Assets.Scripts.Helpers.InspectorExtensions.Extensions;
using UnityEditor;
using UnityEngine;

namespace Assets.Scripts.Helpers.InspectorExtensions.Drawers
{

    [CustomPropertyDrawer(typeof(InlineScriptableObjectAttribute))]
    public class InlineScriptableObjectDrawer : PropertyDrawer
    {
        public override void OnGUI(Rect position, SerializedProperty
property, GUIContent label)
        {
            float foldoutHeight = GetPropertyHeight(property);

            Rect foldoutRect = new Rect(position.x, position.y,
position.width, foldoutHeight);

            position.y += foldoutHeight;

            property.isExpanded = EditorGUI.Foldout(foldoutRect,
property.isExpanded, label, true);

            if (property.isExpanded)
            {
```

```
                    EditorGUI.PropertyField(position,   property,   label,
true);

                    if (property.objectReferenceValue != null
                        &&          property.objectReferenceValue      is
ScriptableObject)
                    {
                        ScriptableObject        scriptableObject        =
(ScriptableObject)property.objectReferenceValue;
                        SerializedObject    serializedObject    =    new
SerializedObject(scriptableObject);
                        SerializedProperty             prop             =
serializedObject.GetIterator();
                        prop.NextVisible(true);

                        EditorGUI.indentLevel++;

                        position.y += EditorGUIUtility.singleLineHeight +
2;

                        while (prop.NextVisible(false))
                        {
                            float             propertyHeight             =
GetPropertyHeight(prop);

                            Rect  fieldPosition  =  new  Rect(position.x,
position.y, position.width, propertyHeight);

                            position.y += propertyHeight;

                            GUIContent        propLabel        =        new
GUIContent(prop.displayName);

                            EditorGUI.PropertyField(fieldPosition,   prop,
propLabel, true);
                        }

                        EditorGUI.indentLevel--;

                        serializedObject.ApplyModifiedProperties();
                    }
                }
            }

            protected  virtual  float  GetPropertyHeight(SerializedProperty
property)
            {
                float totalHeight = EditorGUIUtility.singleLineHeight + 2;

                HeaderAttribute                    header                    =
property.GetAttribute<HeaderAttribute>();
                if (header != null)
                {
                    totalHeight += EditorGUIUtility.singleLineHeight + 10;
                }

                return totalHeight;
            }
```

```
            public override float GetPropertyHeight(SerializedProperty
property, GUIContent label)
            {
                float totalHeight = EditorGUI.GetPropertyHeight(property,
label, true);

                if(property.isExpanded)
                {
                    totalHeight += GetPropertyHeight(property);
                }

                if (property.isExpanded
                    && property.objectReferenceValue != null
                    && property.objectReferenceValue is ScriptableObject)
                {
                    ScriptableObject          scriptableObject          =
(ScriptableObject)property.objectReferenceValue;
                    SerializedObject      serializedObject      =      new
SerializedObject(scriptableObject);
                    SerializedProperty               prop               =
serializedObject.GetIterator();
                    prop.NextVisible(true);

                    while (prop.NextVisible(false))
                    {
                        totalHeight += GetPropertyHeight(prop);
                    }
                }

                return totalHeight;
            }
        }
    }
    #endif
```

Д.2 Користувацький атрибут для полів, що є «серіалізованими інтерфейсами»

Д.2.1 Код класу «SerializeInterfaceAttribute»

```
    #if UNITY_EDITOR
    using System;
    using UnityEngine;

    namespace Assets.Scripts.Helpers.InspectorExtensions.Attributes
    {
        public class SerializeInterfaceAttribute : PropertyAttribute
        {
            public Type InterfaceType { get; private set; }
            /// <summary>
            ///     Requiring     implementation     of     the     <see
cref="T:SerializeInterfaceAttribute"/> interface.
```

```
            /// </summary>
            /// <param name="type">Interface type.</param>
            public SerializeInterfaceAttribute(Type type)
            {
                InterfaceType = type;
            }
        }
    }
    #endif
```

Д.2.2 Код класу «SerializeInterfaceDrawer»

```
    #if UNITY_EDITOR
    using UnityEngine;
    using UnityEditor;
    using Assets.Scripts.Helpers.InspectorExtensions.Attributes;

    namespace Assets.Scripts.Helpers.InspectorExtensions.Drawers
    {
        [CustomPropertyDrawer(typeof(SerializeInterfaceAttribute))]
        public class SerializeInterfaceDrawer : PropertyDrawer
        {
            public override void OnGUI(Rect position, SerializedProperty
property, GUIContent label)
            {
                if                (property.propertyType                !=
SerializedPropertyType.ObjectReference)
                {
                    Debug.LogError(string.Format("{0} - {1}: This drawer
must be used only on Object types",

property.serializedObject.targetObject.GetType().ToString(),
property.displayName));
                    return;
                }

                var constraint = attribute as SerializeInterfaceAttribute;

                Event evt = Event.current;
                if    (DragAndDrop.objectReferences.Length    >    0    &&
position.Contains(evt.mousePosition))
                {
                    var draggedObject = DragAndDrop.objectReferences[0] as
GameObject;

                    if (draggedObject == null || (draggedObject != null
                        &&
draggedObject.GetComponent(constraint.InterfaceType) == null))
                    {
                        DragAndDrop.visualMode                            =
DragAndDropVisualMode.Rejected;

                        if (evt.type == EventType.DragExited)
```

```
                                Debug.LogError(string.Format(
                                    "Object assigned to '{0}' must implement
interface '{1}'",
                                    property.name,
                                    constraint.InterfaceType));
                            }
                        }

                        property.objectReferenceValue = EditorGUI.ObjectField(
                            position,    label,    property.objectReferenceValue,
typeof(GameObject), true);

                        if (property.objectReferenceValue != null)
                        {
                            Component   go   =   property.objectReferenceValue   as
Component;
                            if          (go          !=          null          &&
go.GetComponent(constraint.InterfaceType) == null)
                            {
                                property.objectReferenceValue = null;
                                Debug.LogError(string.Format(
                                    "Object  assigned  to  '{0}'  must  implement
interface '{1}'",
                                    property.name,
                                    constraint.InterfaceType));
                            }

                        }
                    }
                }
            }
            #endif
```

ДОДАТОК Е

Балансування характеристик гравця та ворогів

Рівень	Прибавка здоров'я	Мінімальний модифікатор здоров'я	Максимальний модифікатор здоров'я	Мінімальне здоров'я на поточному рівні	Максимально здоров'я на поточному рівні	Базовий показник здоров'я
1	0	1	1,25	100,00	125,00	100
2	0	1,25	1,5	125,00	150,00	
3	0	1,5	1,75	150,00	175,00	
4	0	1,75	2	175,00	200,00	
5	50	2	2,25	300,00	337,50	
6	0	2,25	2,5	337,50	375,00	
7	0	2,5	2,75	375,00	412,50	
8	0	2,75	3	412,50	450,00	
9	0	3	3,25	450,00	487,50	
10	75	3,25	3,5	731,25	787,50	
11	0	3,5	3,75	787,50	843,75	
12	0	3,75	4	843,75	900,00	
13	0	4	4,25	900,00	956,25	
14	0	4,25	4,5	956,25	1012,50	
15	100	4,5	4,75	1462,50	1543,75	
16	0	4,75	5	1543,75	1625,00	
17	0	5	5,25	1625,00	1706,25	
18	0	5,25	5,5	1706,25	1787,50	
19	0	5,5	5,75	1787,50	1868,75	
20	125	5,75	6	2587,50	2700,00	
21	0	6	6,25	2700,00	2812,50	
22	0	6,25	6,5	2812,50	2925,00	
23	0	6,5	6,75	2925,00	3037,50	
24	0	6,75	7	3037,50	3150,00	
25	150	7	7,25	4200,00	4350,00	

Рисунок Е.1 – Балансування здоров’я гравця

Рівень	Мінімальна шкода на поточному рівні	Максимально шкода на поточному рівні	Середній розмір шкоди гравця	Огр	Скілет	Маленький демон	Розбійник	Бос огр	Огр	Скілет	Маленький демон	Розбійник	Бос огр
1	10	11	10,50	52,50	31,50	10,50	42,00	105,00	5	3	1	4	10
2	11	12	11,50	57,50	34,50	11,50	46,00	115,00	5	3	1	4	10
3	12	13	12,50	62,50	37,50	12,50	50,00	125,00	5	3	1	4	10
4	13	14	13,50	67,50	40,50	13,50	54,00	135,00	5	3	1	4	10
5	21	22,5	21,75	104,40	60,90	19,58	82,65	216,42	4,8	2,8	0,9	3,8	9,95
6	22,5	24	23,25	111,60	65,10	20,93	88,35	231,34	4,8	2,8	0,9	3,8	9,95
7	24	25,5	24,75	118,80	69,30	22,28	94,05	246,27	4,8	2,8	0,9	3,8	9,95
8	25,5	27	26,25	126,00	73,50	23,63	99,75	261,19	4,8	2,8	0,9	3,8	9,95
9	27	28,5	27,75	127,65	72,15	22,20	99,90	274,73	4,6	2,6	0,8	3,6	9,9
10	38	40	39,00	179,40	101,40	31,20	140,40	386,10	4,6	2,6	0,8	3,6	9,9
11	40	42	41,00	188,60	106,60	32,80	147,60	405,90	4,6	2,6	0,8	3,6	9,9
12	42	44	43,00	197,80	111,80	34,40	154,80	425,70	4,6	2,6	0,8	3,6	9,9
13	44	46	45,00	198,00	108,00	31,50	153,00	443,25	4,4	2,4	0,7	3,4	9,85
14	46	48	47,00	206,80	112,80	32,90	159,80	462,95	4,4	2,4	0,7	3,4	9,85
15	60	62,5	61,25	269,50	147,00	42,88	208,25	603,32	4,4	2,4	0,7	3,4	9,85
16	62,5	65	63,75	280,50	153,00	44,63	216,75	627,94	4,4	2,4	0,7	3,4	9,85
17	65	67,5	66,25	278,25	145,75	39,75	212,00	649,25	4,2	2,2	0,6	3,2	9,8
18	67,5	70	68,75	288,75	151,25	41,25	220,00	673,75	4,2	2,2	0,6	3,2	9,8
19	70	72,5	71,25	299,25	156,75	42,75	228,00	698,25	4,2	2,2	0,6	3,2	9,8
20	87	90	88,50	371,70	194,70	53,10	283,20	867,30	4,2	2,2	0,6	3,2	9,8
21	90	93	91,50	366,00	183,00	45,75	274,50	892,13	4	2	0,5	3	9,75
22	93	96	94,50	378,00	189,00	47,25	283,50	921,38	4	2	0,5	3	9,75
23	96	99	97,50	390,00	195,00	48,75	292,50	950,63	4	2	0,5	3	9,75
24	99	102	100,50	402,00	201,00	50,25	301,50	979,88	4	2	0,5	3	9,75
25	119	122,5	120,75	458,85	217,35	48,30	338,10	1171,28	3,8	1,8	0,4	2,8	9,7

Рисунок Е.2 – Балансування здоров’я ворогів

Рівень	Прибавка шкоди	Мінімальний модифікатор шкоди	Максимальний модифікатор шкоди	Мінімальна шкода на поточному рівні	Максимальна шкода на поточному рівні	Базовий показник шкоди
1	0	1	1,1	10,00	11,00	10
2	0	1,1	1,2	11,00	12,00	
3	0	1,2	1,3	12,00	13,00	
4	0	1,3	1,4	13,00	14,00	
5	5	1,4	1,5	21,00	22,50	
6	0	1,5	1,6	22,50	24,00	
7	0	1,6	1,7	24,00	25,50	
8	0	1,7	1,8	25,50	27,00	
9	0	1,8	1,9	27,00	28,50	
10	5	1,9	2	38,00	40,00	
11	0	2	2,1	40,00	42,00	
12	0	2,1	2,2	42,00	44,00	
13	0	2,2	2,3	44,00	46,00	
14	0	2,3	2,4	46,00	48,00	
15	5	2,4	2,5	60,00	62,50	
16	0	2,5	2,6	62,50	65,00	
17	0	2,6	2,7	65,00	67,50	
18	0	2,7	2,8	67,50	70,00	
19	0	2,8	2,9	70,00	72,50	
20	5	2,9	3	87,00	90,00	
21	0	3	3,1	90,00	93,00	
22	0	3,1	3,2	93,00	96,00	
23	0	3,2	3,3	96,00	99,00	
24	0	3,3	3,4	99,00	102,00	
25	5	3,4	3,5	119,00	122,50	

Рисунок Е.3 – Балансування шкоди гравця

Рівень	Мінімальне здоров'я на поточному рівні	Максимально здоров'я на поточному рівні	Середній розмір здоров'я гравця	Огр	Скілет	Маленький демон	Розбійник	Бос огр	Огр	Скілет	Маленький демон	Розбійник	Бос огр
1	100,00	125,00	112,50	8,04	5,93	6,25	7,04	9,38	14	19	18	16	12
2	125,00	150,00	137,50	9,83	7,24	7,64	8,60	11,46	14	19	18	16	12
3	150,00	175,00	162,50	11,61	8,56	9,03	10,16	13,55	14	19	18	16	12
4	175,00	200,00	187,50	13,40	9,87	10,42	11,72	15,63	14	19	18	16	12
5	300,00	337,50	318,75	21,99	15,94	17,71	19,93	26,57	14,5	20	19	16,7	12,3
6	337,50	375,00	356,25	24,57	17,82	19,80	22,27	29,69	14,5	20	19	16,7	12,3
7	375,00	412,50	393,75	27,16	19,69	21,88	24,61	32,82	14,5	20	19	16,7	12,3
8	412,50	450,00	431,25	29,75	21,57	23,96	26,96	35,94	14,5	20	19	16,7	12,3
9	450,00	487,50	468,75	31,25	22,33	26,05	29,30	39,07	15	21	20	17,4	12,6
10	731,25	787,50	759,38	50,63	36,17	42,19	47,47	63,29	15	21	20	17,4	12,6
11	787,50	843,75	815,63	54,38	38,84	45,32	50,98	67,97	15	21	20	17,4	12,6
12	843,75	900,00	871,88	58,13	41,52	48,44	54,50	72,66	15	21	20	17,4	12,6
13	900,00	956,25	928,13	59,88	42,19	51,57	58,01	77,35	15,5	22	21	18,1	12,9
14	956,25	1012,50	984,38	63,51	44,75	54,69	61,53	82,04	15,5	22	21	18,1	12,9
15	1462,50	1543,75	1503,13	96,98	68,33	83,51	93,95	125,27	15,5	22	21	18,1	12,9
16	1543,75	1625,00	1584,38	102,22	72,02	88,03	99,03	132,04	15,5	22	21	18,1	12,9
17	1625,00	1706,25	1665,63	104,11	72,42	92,54	104,11	138,81	16	23	22	18,9	13,2
18	1706,25	1787,50	1746,88	109,18	75,96	97,05	109,18	145,58	16	23	22	18,9	13,2
19	1787,50	1868,75	1828,13	114,26	79,49	101,57	114,26	152,35	16	23	22	18,9	13,2
20	2587,50	2700,00	2643,75	165,24	114,95	146,88	165,24	220,32	16	23	22	18,9	13,2
21	2700,00	2812,50	2756,25	167,05	114,85	153,13	172,27	229,69	16,5	24	23	19,6	13,5
22	2812,50	2925,00	2868,75	173,87	119,54	159,38	179,30	239,07	16,5	24	23	19,6	13,5
23	2925,00	3037,50	2981,25	180,69	124,22	165,63	186,33	248,44	16,5	24	23	19,6	13,5
24	3037,50	3150,00	3093,75	187,50	128,91	171,88	193,36	257,82	16,5	24	23	19,6	13,5
25	4200,00	4350,00	4275,00	251,48	171,00	237,50	267,19	356,25	17	25	24	20,3	13,8

Рисунок Е.4 – Балансування шкоди ворогів

ДОДАТОК Ж

Стаття для Міжнародної науково-практичної конференції

Рисунок Ж.1 - Обкладинка збірки зі статтею

Рисунок Ж.2 – Перша сторінка збірки

UDC 082:001
D 29

Président du comité d'organisation: Goldenblat M.[1]
Vice-président du comité d'organisation: Blandin G.[2]

L'organisation au nom de laquelle le livre est publié:
[1] *PO Plateforme scientifique européenne, Ukraine*
[2] *SCI SORBONNE, République française*

Responsable de la mise en page: Bilous T.
Responsable de la conception: Bondarenko I.

D 29 **Débats scientifiques et orientations prospectives du développement scientifique**: c avec des matériaux de la VI conférence scientifique et pratique internationale, Paris, 1er Mars 2024. Paris-Vinnytsia: La Fedeltà & UKRLOGOS Group LLC, 2024.

ISBN 978-617-8312-01-5
ISBN 978-2-37467-148-2 (PDF)
DOI 10.36074/logos-01.03.2024

UKRLOGOS Group LLC, Ukraine
«La Fedeltà», République française

Les résumés et articles des participants à la VI conférence multidisciplinaire scientifique et pratique internationale «Débats scientifiques et orientations prospectives du développement scientifique», qui s'est tenue à Paris le 1er Mars 2024, sont présentés.

The conference is certified by Euro Science Certification Group (**Certificate № 22512 dated January 7, 2024**):

The conference is also included in the catalog of International Scientific Conferences by ResearchBib; and registered by State Scientific Institution «Ukrainian institute of scientific and technical expertise and information» in the database «Scientific and technical events of Ukraine» (**Certificate № 66 dated 5 January 2024**).

Bibliographic descriptions of the conference proceedings are indexed by Google Scholar, CrossRef, OpenAIRE, OUCI, Scilit, Semantic Scholar, Mendeley, WarldCat and ORCID.

UDC 082:001

ISBN 978-617-8312-01-5
ISBN 978-2-37467-148-2 (PDF)

Рисунок Ж.3 – Друга сторінка збірки

1er Mars, 2024;
Paris, République française

CONTENU

SECTION XVII.
ÉNERGIE ET INGÉNIERIE ÉLECTRIQUE

ARTICLES

ВИЗНАЧЕННЯ МЕЖ ДИНАМІЧНОЇ СТІЙКОСТІ ГЕНЕРАТОРІВ ЕЛЕКТРОСТАНЦІЙ З ОБЛІКОМ НАВАНТАЖЕННЯ ДВИГУНІВ
Пантєлєєва І.В., Блудов І.В. **194**

ПЕРСПЕКТИВИ ЗАСТОСУВАННЯ МОДУЛЬНИХ ТВЕРДОПАЛИВНИХ КОТЕЛЬНИХ ПРИ ДИВЕРСИФІКАЦІЇ ДЖЕРЕЛ ТЕПЛОВОЇ ЕНЕРГІЇ АДМІНІСТРАТИВНИХ БУДІВЕЛЬ
Козячина Б.І., Смілян М.Ю. **200**

РОЗВИТОК НЕТРАДИЦІЙНИХ ТА АЛЬТЕРНАТИВНИХ ДЖЕРЕЛ ЕНЕРГІЇ В ЯКОСТІ СВІТОВОГО ТРЕНДУ
Бойко Н.В. **207**

ABSTRACTS

УЗГОДЖЕННЯ РОБОТИ ДЖЕРЕЛ РЕЗЕРВНОГО ЖИВЛЕННЯ ПІД ЧАС АВАРІЙНИХ ВІДКЛЮЧЕНЬ
Соколовський О.Ф., Качан Д.В. **212**

SECTION XVIII.
ECOLOGIE ET TECHNOLOGIES DE PROTECTION DE L'ENVIRONNEMENT

ABSTRACTS

ВПЛИВ ВІДХОДІВ СОНЯЧНИХ ПАНЕЛЕЙ НА ДОВКІЛЛЯ
Катенін В.Д., Самойленко Н.М. **215**

ПЕРСПЕКТИВИ ТА ПРОБЛЕМИ ВИКОРИСТАННЯ БІОПЛАСТИКУ
Гадаєва Ю.С. **217**

SECTION XIX.
GÉNIE INFORMATIQUE ET LOGICIEL

ARTICLES

ОПТИМІЗАЦІЯ ВИПАДКОВОЇ ГЕНЕРАЦІЇ ROGUELIKE РІВНІВ
Долгий А.І. **220**

Рисунок Ж.4 – Сторінка змісту із посиланням на розділ із текстом статті

ΛΌΓΟΣ

Débats scientifiques et orientations prospectives
du développement scientifique

SECTION 19.
GÉNIE INFORMATIQUE ET LOGICIEL

DOI 10.36074/logos-01.03.2024.050

ОПТИМІЗАЦІЯ ВИПАДКОВОЇ ГЕНЕРАЦІЇ ROGUELIKE РІВНІВ

Долгий Андрій Іванович[1]
Науковий керівник: Новіков Юрій Сергійович[2]

1. здобувач вищої освіти факультету комп'ютерних наук
Харківський національний університет радіоелектроніки, УКРАЇНА
ORCID ID: 0009-0005-5412-891X

2. старший викладач кафедри програмної інженерії
Харківський національний університет радіоелектроніки, УКРАЇНА
ORCID ID: 0000-0003-1910-3256

Анотація. *У роботі розглянуто питання вибору підходів до випадкової генерації рівнів у іграх Roguelike жанру, що складаються з кімнат. Оприявлено сильні та слабкі сторони кожного підходу.*
Автор узагальнює та розділяє за суттю методи генерації та оптимізації відтворення мапи та комірок відповідно. Обґрунтовано умови поєднання підходів за різними вимогами та можливостями. Автором розглянуто оптимізаційні можливості кожного з методу генерації.
Визначено оптимальний підхід для випадкової генерації рівнів гри. Проаналізовано метод створення мапи у інді-проєкті «The fate of the liar».

У сучасній ігровій індустрії існує багато продуктів із випадковою генерацією рівнів, що складаються з кімнат, та вони продовжують анонсуватись та виходити на різноманітних платформах та сервісах. Такі проєкти мають спільний жанр – Roguelike [1]. Використання випадкової генерації рівнів є дуже корисним підходом при створенні ігор, оскільки він вирішує одразу декілька проблем. По-перше, при використанні автоматизованого створення локацій, зникає потреба пророблювати великі мапи власноруч, та навпаки може збільшити різноманіття рівнів та їх наповнення, за рахунок випадковості їх виникнення. Цей аспект дуже допомагає інді-розробникам, в яких не вистачає грошей, часу та людей на інші рішення. Разом з тим, більші команди, можуть витратити ресурси, які було вивільнено, на поглиблення інших механік чи

Рисунок Ж.5 – Перша сторінка статті

створення нових [2]. При цьому вони нічого не втрачають, отримуючи натомість такий самий, робочий, ігровий світ. По-друге, підхід із випадковою генерацією рівнів, може бути додатково обумовлений особливостями жанру, стилістики чи наративу проєкту, що розроблюється. Однак за спрощенням та пришвидшенням процесу розробки, слідує питання оптимізації генерації рівнів та подальшого їх використання у грі [3].

Слід зазначити, що існують генератори мап, що працюють за межами кімнат, та створюють світ більш цілісним та відкритим тощо. Однак тематика роботи спирається саме на ігри з рівнями, які складаються з кімнат.

У таких проєктах постає декілька питань. Перше – як генерувати послідовність приміщень, зі збереженням цілісності мапи, їх наповнення згідно певним умовам рівня (тобто розмір, наявність певних кімнат, їх наповнення та тому подібне) тощо. Друге – як зберігати мапу у пам'яті додатку та у ігровому просторі, та як демонструвати їх гравцю. Третє – питання оптимізації зазначених вище моментів та визначення кращих підходів для вирішення поставлених задач.

Загально, підходи до генерації рівнів, які складаються з кімнат, можна розділити на два напрямки: спочатку згенерувати кімнати, а лише потім будувати шляхи тощо; та навпаки, почати зі структури мапи. При першому підході спочатку випадково створюються комірки у просторі, не перетинаючи одна одну, після чого, алгоритмічно створюються шляхи між ними так, щоб мапа могла бути пройденою гравцем та не мала відокремлених кімнат тощо. На відміну від першого, другий напрямок загалом спирається на швидку, абстрактну побудову структури рівня, після чого заповнює комірки кімнатами, згідно правилам генерації.

З огляду на описані напрямку, можна дійти до висновків, що перший варіант, може вийти більш унікальним з огляду структури. Однак він має низку недоліків, а саме більший час генерації, через побудову маршрутів, проблему з наративної сторони ігрового світу та занурення у нього (як приклад, випадковість розташування кімнат, може призвести до певного дисонансу, а саме випадки, коли комірки, що асоціюються з низом знаходяться в горі та навпаки тощо), накладні витрати на створення додаткових переходів складної форми тощо. У свою чергу підхід зі створенням мапи заздалегідь у спрощеному форматі, є хоч і більш звичайним за виглядом, однак витрачає менше часу на генерацію, не потребує додаткових переходів між кімнатами, та уникає наративного дисонансу, через первинність створення за низкою правил.

Стосовно питання цілісності рівня, наявності необхідних кімнат та відповідності внутрішнього вмісту останніх, то при кожному підході все залежить від правильно заданих правил до заповнення комірок мапи.

Рисунок Ж.6 – Друга сторінка статті

Згідно другому загальному питання, що стосується зберіганню мапи та комірок у пам'яті та як відтворювати у ігровому оточені, також слід розглянути загальні напрямки. Перший – всі кімнати створюються на моменті генерації рівня та залишаються у оточуючому просторі до виходу з мапи, іншими словами, всі комірки існують одночасно та гравець може побачити одну з іншої. Другий підхід ґрунтується на тому, що заздалегідь, у пам'яті знаходиться лише загальний план кімнат та переходів, поточна комірка, яка є єдиною у ігровому просторі та інформація про стан вже відвіданих кімнат. Згідно цьому підходу, кожна наступна кімната з'являється перед гравцем лише у момент свого візиту, а всі інші зникають, повністю вивільняючи ресурси, що були на них витрачені. Третій варіант збереження та відтворення комірок, містить змішаний характер, а саме одночасно зберігає у пам'яті всі комірки, що були створенні під час генерації мапи рівня, однак у кожен момент часу, оброблюється лише одна – поточна кімната.

Розглядаючи описані вище підходи, слід визначити їх позитивні та негативні сторони з боку оптимізації та нюансів використання. Перший варіант, є найбільш затратним, як з боку пам'яті, так і зі сторони навантаження на графічний компонент, через необхідність відображення значно більшої кількості предметів, так само як і їх обробки центральним процесором. Однак за певних умов та при необхідній оптимізації, такий підхід може показувати себе дуже гарно з наративної та візуально сторони проєкту, проте все ще вимагаючи більших можливостей від ігрової платформи. Другий підхід, є найбільш невибагливим до можливостей засобу, на якому запущена гра, оскільки майже нічого не зберігає у пам'яті (за виключенням вказаних вище деталей) та не оброблює додаткових предметів у решті кімнат. Однак такий варіант має свої недоліки, а саме навантаження та додатковий час для створення наступної кімнати, при переході з попередньої, що викликає певний наративний дисонанс через перерву у ігровому процесі (особливо гостро відчувається при швидкому та агресивному дизайні останнього). Стосовно третьої варіації – вона є балансом між першими двома, оскільки, хоча все ще витрачає пам'ять на всі кімнати та об'єкти в них, такий підхід не витрачає ані ресурсів на обробку додаткових об'єктів, ані часу гравця, на очікування іншої кімнати. Однак такий підхід має свої підводні камені з точки зору контролю станів анімацій, особливих подій у кімнатах тощо.

Підходячи до останнього – третього питання: оптимізації та пошуку оптимального рішення до поставлених задач, слід відзначити, що для кожного окремого випадку, що виходить з унікального контексту наративної сторони гри та її цільової аудиторії, є власне рішення. Так для ігор, що створюються із сучасною графікою, ефектами, можливостями тощо, що обмежує цільову

Рисунок Ж.7 – Третя сторінка статті

аудиторію до гравців із достатньо потужними платформами, підійдуть будь-які із зазначених вище підходів до генерації та відтворення рівнів. Однак найбільш відповідним буде застосувати більш яскраву генерацію – перший тип та першу варіацію відтворення комірок, оскільки мається на увазі, що потужностей буде вдосталь. При розгляданні проєктів для менш потужних або застарілих систем, може бути доречно застосувати більш простий підхід до генерації – другий тип та другий варіант відтворення: з економією пам'яті та ресурсів на обробку. Така економія обумовлена більш сильною обмеженістю обчислювальної спроможності ігрових платформ цільової аудиторії. Водночас, для більш масового прошарку гравців, що мають більш середні системи, буде оптимально застосувати поєднання другого типу генерації та третього підходу до відтворення рівнів. Таке поєднання забезпечить баланс між вимогами до обчислювальної потужності, складністю реалізації та наративного сприйняття.

Слід зазначити, що для певних ситуацій, буде правильніше використовувати конкретні підходи, але це стосується особливих вимог з наративного боку, чи сторони дизайну.

У проєкті «The fate of the liar», що розроблюється у форматі інді-розробки, було застосовано саме оптимальний варіант генерації рівнів. Згідно першому типу генерації мапи, спочатку створюється загальна структура комірок та переходів. Після цього рівень завантажується у пам'ять, однак оброблюється лише поточне приміщення, згідно третьому підходу до відтворення мапи.

При генерації використовується алгоритм, що рекурсивно створює дерево з кімнат, певної довжини, починаючи зі стартової комірки (рис. 1). За таким підходом, середня часова складність становить O(n), тобто лінійна, через те, що на кожну кімнату витрачається єдина ітерація у середньому. Складність за пам'яттю також складає лише O(n), через те, що алгоритм зберігає лише відомість про дані комірок. Після створення комірок та маршрутів, алгоритм зафарбовує вільні місця кімнатами, за відповідними правилами (додає комірки босів, скарбів, ресурсів, тощо). Складність другої частини алгоритму вже являє собою O(n*m), де n – загальна кількість кімнат, а m – число видів комірок, якими наповнюється мапа. Слід відзначити, що m є невеликим числом, тому O(n*m) є ближчою до лінійної складності ніж до квадратичної. Також у другій частині алгоритму додатково пам'ять не виділяється, оскільки використовується вже задані та записані раніше значення тощо. Водночас, побудова мапи із використанням дерева є доцільним підходом через зручну модель збереження кімнат та переходів між ними.

У свою чергу відтворення рівня, з лише однією активною коміркою, значно виграє за ресурсами що виділяються при відтворенні всіх кімнат, тим паче обробки об'єктів в них (табл. 1). У таблиці відображені данні стосовно

Рисунок Ж.8 – Четверта сторінка статті

поточного показника кадрової частоти (FPS), кількості трикутників у кадрі (Tris), обсягу вершин (Verts), час прорахунку кадру центральним процесором та тривалість побудови зображення (затримка CPU та GPU відповідно), навантаження саме додатку на комп'ютер (навантаження CPU та GPU) та кількість зайнятої оперативної пам'яті (RAM). Усі тести проводились на єдиній тестовій сцені із використанням згенерованого рівня на двадцять дев'ять кімнат. Реалізовано проєкт було на рушії Unity3D. Було проведено чотири спостереження. Перший тест: з відображенням усіх комірок. Другий тест: з відображенням усіх кімнат, проте оптимізувавши неактивні кімнати, вимкнувши більшість процесів у них. Третій тест: із використанням зазначеного підходу, коли активна лише поточна кімната, а всі інші вимкнуті та не оброблюються взагалі. Четвертий тест: із використанням підходу з завантаженням поточної кімнати у пам'ять у реальному часі, разом з видаленням даних про решту неактивних комірок. Слід також відзначити, що варіант відображення лише поточної кімнати, без збереження у пам'яті інших, не буде відрізнятись у тестах за більшістю вказаних показників, лише за виділеною пам'яттю, натомість створюючи додаткові підвантаження в момент переходу між комірками.

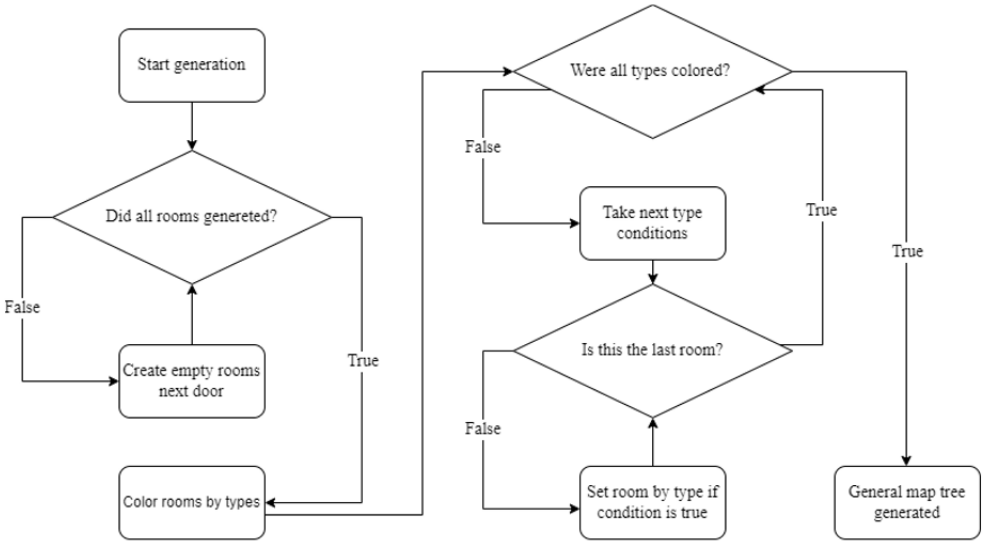


Рис. 1. **Блок-схема алгоритму генерації мапи**

За результатами вимірювання, можна чітко побачити, домінантність з продуктивності останніх підходів, а саме оптимізації через відключення усіх

Рисунок Ж.9 – П'ята сторінка статті

кімнат окрім поточної та тримання у пам'яті лише поточної. Однак хоча потреби четвертого тесту у оперативній пам'яті менші, він надає додаткові ускладнення як з точки зору реалізації більш складного механізму підвантажень кімнат, так і гравець отримає певні екрани завантаження між кімнатами через це.

Таблиця 1

Порівняння показників ігрового процесу

Тест	FPS	Tris	Verts	Затр. CPU	Навант. CPU	Затр. GPU	Навант. GPU	RAM
1	≈ 10	1.8m	1.3m	≈ 100 мс	≈ 50%	≈ 8 мс	≈ 40%	44 МБ
2	≈ 44	1.8m	1.3m	≈ 23 мс	≈ 40%	≈ 8 мс	≈ 40%	44 МБ
3	≈ 76	210k	176.3k	≈ 13 мс	≈ 33%	≈ 4 мс	≈ 18%	44 МБ
4	≈ 76	210k	176.3k	≈ 13 мс	≈ 33%	≈ 4 мс	≈ 18%	22 МБ

[авторська розробка]

Висновки. Таким чином, найбільш оптимізованим підходом до реалізації випадкової генерації рівнів, що складаються з кімнат є застосування первинної побудови мапи та відтворення лише поточної комірки з подальшим завантаженням інших за потреби. Однак більш привабливим, проте витратним, буде більш складна первинна генерація кімнат з відображенням усіх комірок одночасно. Разом з тим найбільш оптимальним варіантом слугуватиме первинна генерація мапи та підхід із прихованим збереженням комірок, без підвантажень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

[1] Cerny, V., & Dechterenko, F. (2015). Rogue-like games as a playground for artificial intelligence–evolutionary approach. In *Entertainment Computing-ICEC 2015: 14th International Conference, ICEC 2015, Trondheim, Norway, September 29-Ocotober 2, 2015, Proceedings 14* (pp. 261-271). Springer International Publishing.

[2] Bacher, T. (2018). Procedural Level Generation Algorithms.

[3] Smith, A. J., & Bryson, J. J. (2014). A logical approach to building dungeons: Answer set programming for hierarchical procedural content generation in roguelike games. In *Proceedings of the 50th Anniversary Convention of the AISB*.

Рисунок Ж.10 – Шоста сторінка статті

ДОДАТОК И

Презентація проєкту на виставці Міжнародного молодіжного форуму

Рисунок И.1 – Перший слайд презентації

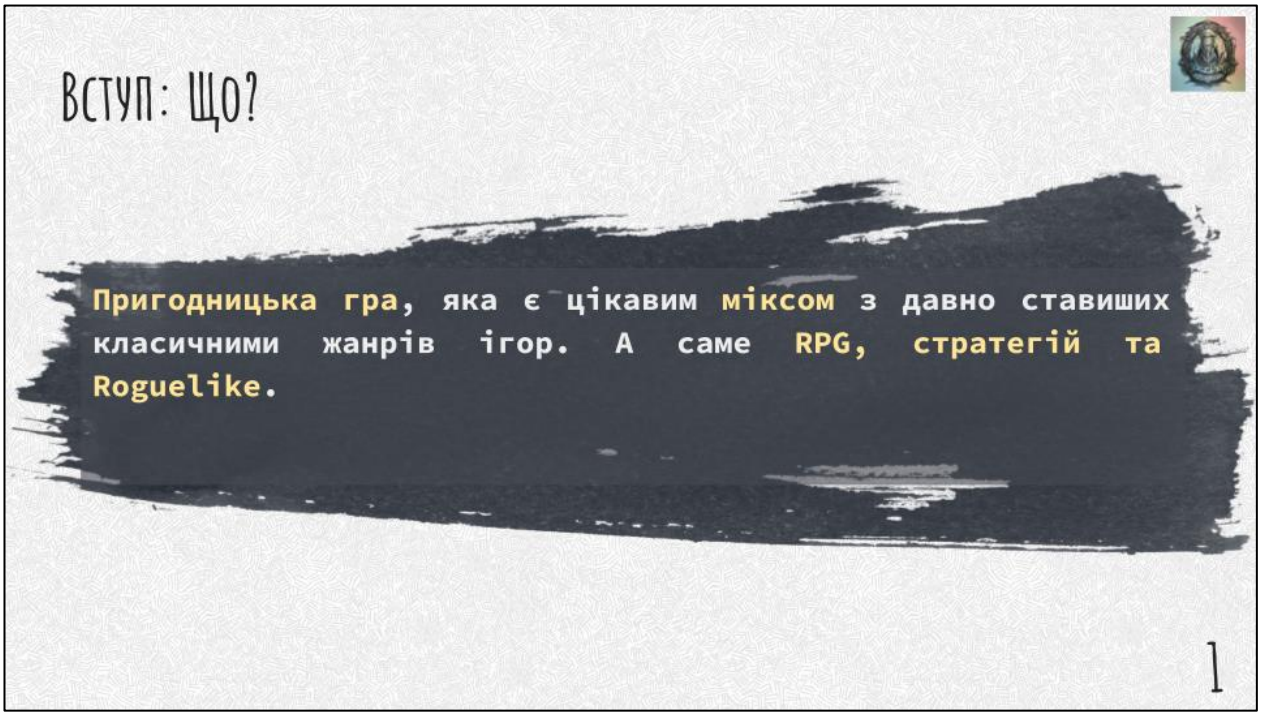


Рисунок И.2 – Другий слайд презентації

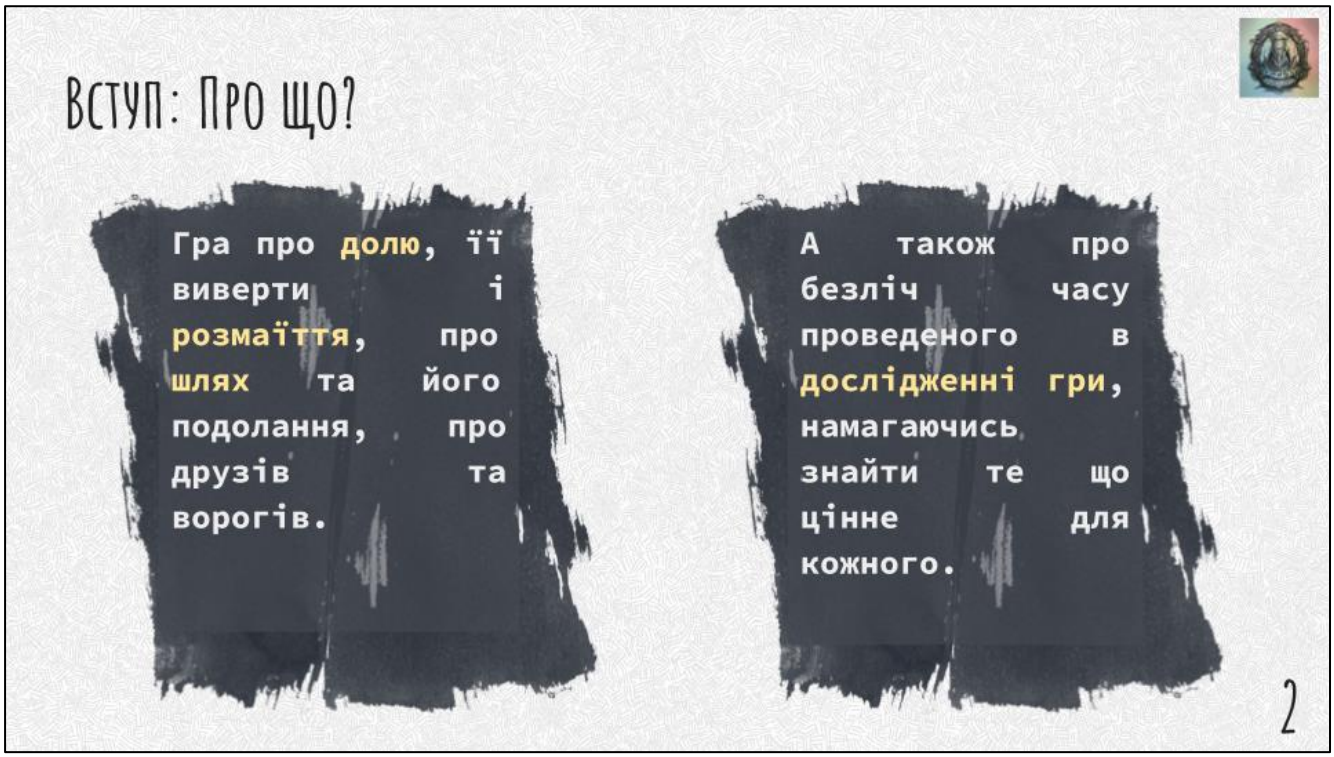


Рисунок И.3 – Третій слайд презентації

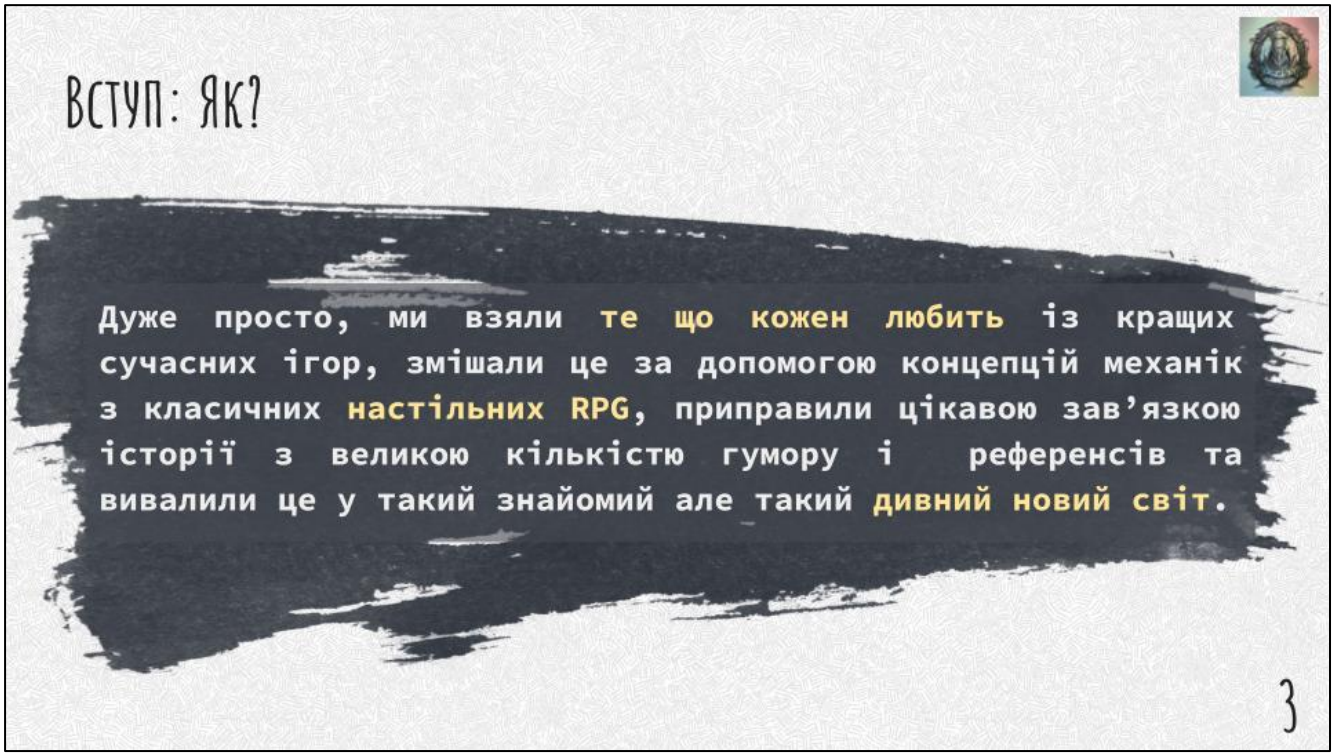


Рисунок И.4 – Четвертий слайд презентації

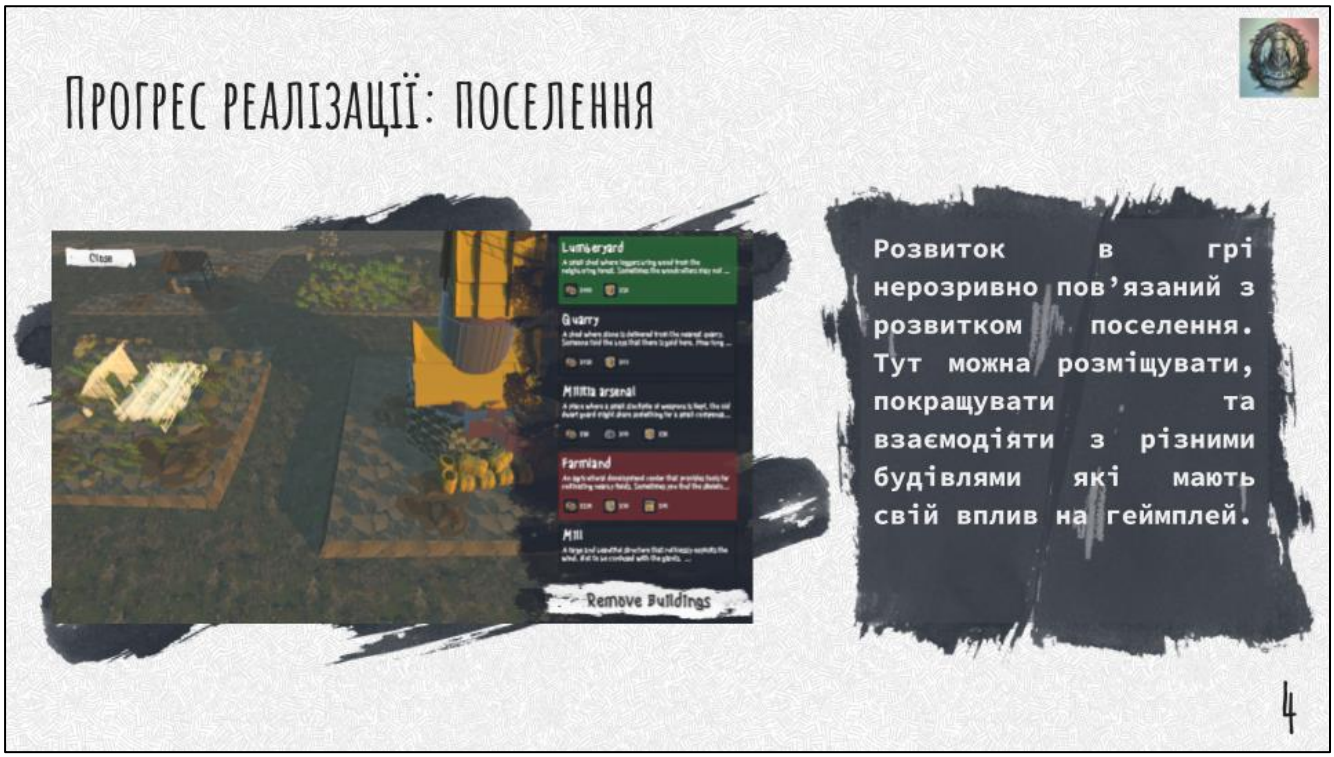


Рисунок И.5 – П'ятий слайд презентації

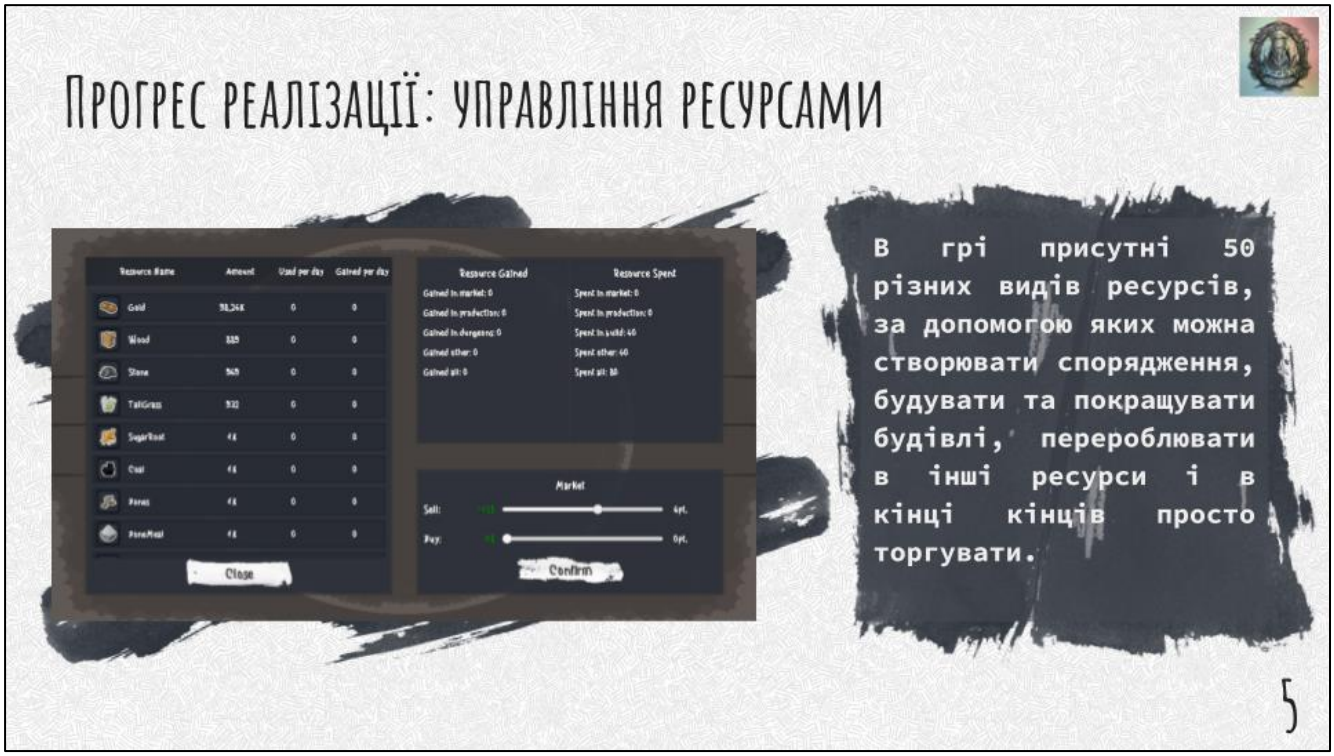


Рисунок И.6 – Шостий слайд презентації

Рисунок И.7 – Сьомий слайд презентації

Рисунок И.8 – Восьмий слайд презентації

Рисунок И.9 – Дев’ятий слайд презентації

Рисунок И.10 – Десятий слайд презентації

ДОДАТОК К

Каталог проєктів виставки Міжнародного молодіжного форуму

Рисунок К.1 – Обкладинка каталогу

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.

КАТАЛОГ ВИСТАВКИ
ТЕХНІЧНОЇ ТВОРЧОСТІ МОЛОДІ

Харків 2024

Рисунок К.2 – Перша сторінка каталогу

XXVIII Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». Каталог виставки технічної творчості молоді. – Харків: ХНУРЕ. 2024. – 27 с.

61166 Україна, Харків, просп. Науки, 14
тел./факс: (057) 7021397

E-mail: mref21@nure.ua

Рисунок К.3 – Друга сторінка каталогу

Ігрові технології

1. Ігровий програмний застосунок в жанрі RPG з елементами Roguelike та економічної стратегії

Автори: ***Прудіус Владислав Юрійович, Долгий Андрій Іванович,*** ст. гр. ПЗПІ-20-4, ХНУРЕ.

Науковий керівник: Новіков Юрій Сергійович, старший викладач. каф. ПІ, ХНУРЕ.

Розроблено ігровий програмний застосунок що комбінує в собі елементи популярних жанрів. В грі присутні елементи розвитку поселення, управління ресурсами, розвитку персонажа, участі у економічних та соціальних процесах ігрового світу, а також можливості дослідження процедурно генерованих підземель та динамічна бойова система.

Основною ціллю розробки є заохочення гравця до повторного проходження з мінімізацією повторного отримання ідентичного ігрового досвіду.

Для цього впроваджено низку ігрових механік та підходів які допомагають змінювати деякі елементи ігрового процесу за бажанням гравця або автоматично при повторному проходженні. Таким чином гравцеві для того щоб повноцінно пройти гру, випробувавши увесь арсенал можливих функцій, не вистачатиме одного проходження, а повторні проходження стимулюватимуть зміну ігрового стилю.

2. Ігровий додаток «Шахи»

Автор: ***Бєлєвцова Олена Сергіївна***, ст. гр. КІУКІ-22-8, ХНУРЕ.

Науковий керівник: Малькова Ірина Анатоліївна, ас. каф. ІУС, ХНУРЕ.

Інтерактивна гра "Шахи" надає гравцям можливість взаємодіяти з шаховою дошкою, здійснювати ходи та виконувати різноманітні дії, такі як рокіровка, взяття на проходженні, атаки та захисти фігур.

Основною функціональністю гри є перевірка правильності здійснених гравцем ходів та визначення матових ситуацій. Гравці можуть піддаватися мату або завдавати його опоненту, використовуючи різні шахові стратегії та тактики.

Гра розроблена з використанням мови програмування C++ для написання логіки гри та графічної бібліотеки для створення візуального інтерфейсу.

3. Ігровий додаток «ColorMix»

Автори: ***Двінов Ілля Юрійович, Кіров Микита Русланович***, ст. гр. КНТ-21-3, ХНУРЕ.

Науковий керівник: Малькова Ірина Анатоліївна, ас. каф. ІУС, ХНУРЕ.

Ігровий додаток "ColorMix" призначений для розваги та цікавого проведення вільного часу. Гру виконано у жанрі аркада-головоломка, що характеризується коротким часом, але інтенсивним ігровим процесом. Ігровий додаток допоможе гравцеві зняти стрес, а також розвинути свої інтелектуальні здібності та увагу.

Гру розроблено з використанням ігрового рушія Unity, для написання скриптів використано мову програмування C#. Ігрову графіку створено за допомогою програми Blender.

Рисунок К.4 – Сторінка із описом зазначеного проєкту

ДОДАТОК Л

Диплом за призове місце на виставці Міжнародного молодіжного форуму

Рисунок Л.1 – Диплом за призове місце на виставці