

**МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ**

**ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І
СИСТЕМ**

КВАЛІФІКАЦІЙНА РОБОТА

на тему: Розробка алгоритму процедурної генерації розгалужених діалогових структур для візуальних новел на платформі Ren'Py

Здобувача вищої освіти першого (бакалаврського) рівня,
курсу, групи КБ-22-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології (за галузями)»



(підпис)

Заведії Артема
Владиславовича

(прізвище, ім'я, по батькові)

Науковий керівник



(підпис)

Омельчук Антон Анатолійович,
канд. техн. наук

(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Керівник освітньо-
професійної програми



(підпис)

Ніжегородцев Владислав
Олександрович, канд. пед.
наук, доцент

(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Допущено до захисту за рішенням кафедри комп'ютерних та інформаційних технологій і систем протокол від «12» травня 2026 №20

Завідувач кафедри



(підпис)

Філоненко Михайло
Миколайович, канд. фіз.-мат.
наук, доцент

(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Ірпінь – 2026

Реєстрація № 50

27.05.2026

(дата)



(підпис відповідальної особи)

Результати перевірки

до захисту

(до захисту, на доопрацювання)

12.05.2026

(дата)



(підпис)

Результат захисту

4 (добре)

(оцінка)

08.06.2026

(дата)



(підпис)

Голова комісії:



(підпис)

О.С. Дюбанов

Члени комісії:



(підпис)

М.М. Філоненко



(підпис)

В.О. Ніжегородцев



(підпис)

А.А. Омельчук

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	25		08.06.2026 Омельчук А.А. Дюбанов О.С.
2. Дотримання вимог щодо оформлення	від 0 до 10	7		Філоненко М.М. Ніжегородцев В.О.
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	5		Омельчук А.А. Суховерхівка Л.П.
4. Рецензент	від 0 до 5	4		
РАЗОМ БАЛІВ	від 0 до 50	41		

цінювання кваліфікаційної роботи науковим керівником – 42 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
83	B	4 (добре)	08.06.2026	Дюбанов О.С. Філоненко М.М. Ніжегородцев В.О. Омельчук А.А. Суховерхівка Л.П.

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
	A	(відмінно)
	B	
	C	4 (добре)
	E	(задовільно)
	FX	(незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань: 12 «Інформаційні технології»
Спеціальність: 122 «Комп'ютерні науки»
Освітня програма: «Інформаційні управляючі системи і технології (за галузями)»

Затверджую
Завідувач кафедри  Філоненко М.М.
«01» грудня 2025

Завдання на кваліфікаційну роботу
здобувачу вищої освіти
Заведії Артему Владиславовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи Розробка алгоритму процедурної генерації розгалужених діалогових структур для візуальних новел на платформі Ren'Py
Затверджена наказом ректора від «28» листопада 2025 № 1526
2. Строк здачі закінченої роботи «08» травня 2026
3. Мета роботи Автоматизація створення контенту через розробку алгоритму процедурної генерації розгалужених діалогів на платформі Ren'Py.
Об'єкт дослідження Структурування інтерактивного контенту та розгалужених сюжетів у візуальних новелах.
Орієнтовна база дослідження На відкритих специфікаціях Ren'Py Engine API, принципах побудови графів діалогових дерев та сучасних методологіях проектування архітектури ігрового програмного забезпечення.
4. Вихідні дані до роботи Технічна документація Ren'Py та бібліотеки Tracery; протоколи інтеграції скриптів Python; принципи процедурної генерації (PCG), алгоритми обчислення ігрових змінних та методики тестування нелінійного ПЗ.
5. Перелік графічного матеріалу
Порівняльна таблиця рушіїв, файлова структура, UML-діаграми та інтерфейс меню вибору, схеми взаємодії модулів логіки, генерації Tracery та візуальних ефектів, спрайти Олександра й Данила, структура проєкту, папок, статистик та активів у Ren'Py, варіанти ігрових діалогів та глітч-ефект новели.
6. Зміст кваліфікаційної роботи (перелік питань за кожним розділом)
Вступ: актуальність, мета, об'єкт, предмет, завдання. Розділ 1: теорія сторітелінгу та аналіз рішень у GameDev. Розділ 2: архітектура Ren'Py, моделювання граматик і сценаріїв. Розділ 3: реалізація модуля, Python-middleware, Tracery, тестування. Висновки: результати розробки та оцінка ефективності ПЗ.

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	Фактичн о
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 30.10.2025	До 30.10.2025
2	I розділ (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	26.12.2025	26.12.2025
3	II розділ (аналіз експериментальної частини про методики досліджень)	18.02.2026	18.02.2026
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	16.04.2026	16.04.2026
5	Висновки	01.05.2026	01.05.2026
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	07.05.2026	07.05.2026
7	Отримання відгуку керівника	29.05.2026	29.05.2026
8	Отримання зовнішньої рецензії	01.06.2026	01.06.2026

8. Дата видачі завдання «01» грудня 2025

9. Керівник кваліфікаційної роботи


(підпис)

Омельчук А.А.
(прізвище, ініціали)

10. Керівник освітньо-професійної програми


(підпис)

Ніжегородцев В.О.
(прізвище, ініціали)

11. Завдання до виконання отримав здобувач вищої освіти


(підпис)

Заведія А.В
(прізвище, ініціали)

АНОТАЦІЯ

«Розробка алгоритму процедурної генерації розгалужених діалогових структур для візуальних новел на платформі Ren'Py»

Мета роботи полягає у створенні та програмній реалізації алгоритму, який автоматично генерує унікальні діалоги та варіанти вибору в грі на базі рушія Ren'Py та мови Python.

Завданнями роботи є аналіз існуючих підходів до створення діалогів у візуальних новелах; обґрунтування вибору технологій для розробки; створення архітектури та бази даних гри; реалізація алгоритму випадкової генерації тексту за допомогою бібліотеки Tracery; розробка та оптимізація візуальних ефектів «поламаного світу» (глітчів).

Предметом дослідження є методи та алгоритми автоматичного створення ігрового тексту, побудови діалогових дерев та оптимізації графіки.

Об'єктом дослідження є процес розробки та програмування кросплатформених комп'ютерних ігор.

Базою дослідження виступає програмний код та структура розроблюваної візуальної новели «Echoes of Aeternalis».

За результатами роботи створено робочий прототип гри з динамічними діалогами, які змінюються залежно від рішень гравця. Отримані результати та алгоритми можуть бути використані іншими розробниками для створення інді-ігор із високою варіативністю, а також як практичний приклад для студентів при вивченні програмування та геймдизайну.

Кваліфікаційна робота містить 51 сторінку, 1 таблицю, 18 рисунків, список використаних джерел містить 35 найменування.

Ключові слова: процедурна генерація, візуальна новела, Ren'Py, Python, Tracery, нелінійний сюжет, глітч-ефекти, ігрова логіка.

Рік виконання кваліфікаційної роботи: 2026

Рік захисту роботи: 2026

ABSTRACT

«Development of a Procedural Content Generation Algorithm for Branching Dialogue Structures in Visual Novels on the Ren'Py Platform»

The aim of this work is to design and implement an algorithm that automatically generates unique dialogues and choice options within a game based on the Ren'Py engine and the Python programming language.

The tasks of the work are as follows: to analyze existing approaches to creating dialogues in visual novels; to justify the choice of development technologies; to create the game architecture and database; to implement the random text generation algorithm using the Tracery library; to develop and optimize visual effects of a "broken world" (glitches).

The subject of the research is methods and algorithms for automatic game text creation, dialogue tree construction, and graphics optimization.

The object of the research is the process of cross-platform computer game development and programming.

The research is based on the source code and structure of the developed visual novel "Echoes of Aeternalis".

Based on the work results, a working game prototype has been created with dynamic dialogues that change depending on the player's choices. The obtained results and algorithms can be used by other developers to create indie games with high variability, as well as a practical example for students studying programming and game design.

The qualification work consists of 51 pages, 1 table, 18 figures, a list of 31 references.

Keywords: procedural generation, visual novel, Ren'Py, Python, Tracery, non-linear story, glitch effects, game logic.

Year of qualification work: 2026

Year of work defense: 2026

ЗМІСТ

ЗМІСТ.....	8
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ.....	3
ВСТУП	5
РОЗДІЛ 1. ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ОСНОВИ РОЗРОБКИ КРОСПЛАТФОРМЕНИХ СИСТЕМ ІНТЕРАКТИВНОГО НАРРАТИВУ	7
1.1. Еволюція цифрової взаємодії: від тексту до інтерактивного досвіду	7
1.2. Як сьогодні підходять до кросплатформеної розробки	7
1.3. Чому саме Ren'Py	8
1.4. Процедурна генерація як спосіб оживити гру	9
1.5. Архітектура гри та ефект “руйнування світу”	12
РОЗДІЛ 2. АРХІТЕКТУРНЕ ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОЇ ГРИ «ECHOES OF AETERNALIS»	18
2.1. Архітектурне проєктування та концептуальна модель застосунку «Echoes of Aeternalis»	18
2.2. Розробка системи ігрових станів та моделей даних	20
2.3. Алгоритмічна реалізація процедурного нарративу за допомогою бібліотеки Tracery	22
2.4. Математичне моделювання візуальних аномалій та глітч-ефектів.....	24
2.5. Розробка графічного інтерфейсу та оптимізація візуальних активів	26
РОЗДІЛ 3. РОЗРОБКА ТА ОПТИМІЗАЦІЯ ІГРОВОГО ЗАСТОСУНКУ ...	30
3.1. Реалізація основних компонентів гри	30
Базова структура проєкту.....	30
3.2. Оптимізація гри та адаптація під різні платформи	37
3.4. Перспективи розвитку та потенціал комерціалізації проєкту	43
ВИСНОВКИ	51
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	53

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ДСТУ – Державний стандарт України

КВГ – контекстно-вільна граматика

ООП – об’єктно-орієнтоване програмування

ПЗ – програмне забезпечення

2D – двовимірний (two-dimensional)

API – програмний інтерфейс застосунку (Application Programming Interface)

CFG – контекстно-вільна граматика (Context-Free Grammar)

CPU – центральний процесор (Central Processing Unit)

DLC – завантажуваний вміст/доповнення до гри (Downloadable Content)

FPS – кількість кадрів за секунду / плавність відображення (Frames Per Second)

GPU – графічний процесор (Graphics Processing Unit)

GUI – графічний інтерфейс користувача (Graphical User Interface)

IDE – інтегроване середовище розробки (Integrated Development Environment)

JSON – формат обміну даними (JavaScript Object Notation)

MVP – мінімально життєздатний продукт / робочий прототип (Minimum Viable Product)

NPC – неігровий персонаж (Non-Player Character)

PCG – процедурна генерація контенту (Procedural Content Generation)

PNG – формат растрової графіки з компресією без втрат (Portable Network Graphics)

PNG-24 – повноколірний формат PNG із підтримкою 24-бітного кольору та альфа-каналу

Python – високорівнева мова програмування

Ren'Py – рушій для створення візуальних новел

State Machine – скінченний автомат (модель станів системи)

Tracery – бібліотека для процедурної генерації тексту

UI – користувацький інтерфейс (User Interface)

UX – користувацький досвід (User Experience)

Unicode – всесвітній стандарт кодування символів (включаючи спецсимволи та глітч-текст)

Zalgo text – специфічний спосіб деформації тексту за допомогою комбінованих символів Unicode

ВСТУП

Візуальні новели є одним із напрямів інтерактивних комп'ютерних ігор, у яких основний акцент робиться на поданні сюжету та розкритті персонажів, а не на динамічному ігровому процесі. Даний жанр сформувався наприкінці XX століття та швидко здобув популярність завдяки поєднанню художнього тексту, візуального супроводу та можливості взаємодії користувача з історією.

Проте на сучасному етапі розвитку спостерігається певне уповільнення еволюції жанру, що зумовлено повторюваністю сюжетних рішень та обмеженим використанням нових технологічних підходів.

Візуальні новели також активно застосовуються в освітніх та тренувальних системах, де вони слугують засобом інтерактивного подання інформації. Незважаючи на це, питання ефективної побудови їх структури, зокрема організації діалогів та варіативності сценарію, залишається недостатньо дослідженим. Особливу увагу привертає проблема створення розгалужених діалогових систем, які є основою інтерактивності у таких застосунках.

На сьогодні розробники візуальних новел стикаються з рядом труднощів, серед яких варто виділити складність ручного створення великої кількості варіантів розвитку подій, монотонність сценаріїв та відсутність гнучких механізмів масштабування контенту. У зв'язку з цим актуальним є пошук нових підходів до автоматизації процесу формування діалогових структур, що дозволить підвищити різноманітність та інтерактивність сюжетів.

Одним із перспективних напрямів вирішення зазначених проблем є використання алгоритмів процедурної генерації. Такий підхід дає змогу створювати розгалужені діалогові структури автоматично, на основі заданих правил та параметрів, що значно спрощує процес розробки та відкриває можливість формування унікального досвіду для кожного користувача.

Метою даної роботи є розробка алгоритму процедурної генерації розгалужених діалогових структур для візуальних новел на платформі Ren'Py.

Для досягнення поставленої мети необхідно виконати такі завдання:

- дослідити особливості візуальних новел як програмних систем;
- проаналізувати існуючі підходи до реалізації діалогів у середовищі Ren'Py;
- розробити модель розгалуженої діалогової структури;
- створити алгоритм процедурної генерації діалогів;
- реалізувати розроблений алгоритм засобами платформи Ren'Py та мови Python;
- провести тестування та оцінити ефективність запропонованого рішення.

Об'єктом дослідження є процес розробки візуальних новел як програмних систем .

Предметом дослідження є алгоритми процедурної генерації діалогових структур.

Практичне значення отриманих результатів полягає у можливості застосування розробленого алгоритму для автоматизації створення інтерактивних сценаріїв, що дозволить підвищити ефективність розробки та забезпечити більшу варіативність ігрового процесу. Очікується, що результати роботи сприятимуть подальшому розвитку інструментів створення візуальних новел та впровадженню інноваційних підходів у галузі інтерактивного контенту.

Таким чином, дана робота спрямована на вирішення актуальної задачі підвищення гнучкості та автоматизації процесу створення розгалужених діалогових систем у сучасних візуальних новелах.

РОЗДІЛ 1. ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ОСНОВИ РОЗРОБКИ КРОСПЛАТФОРМЕНИХ СИСТЕМ ІНТЕРАКТИВНОГО НАРРАТИВУ

1.1. Еволюція цифрової взаємодії: від тексту до інтерактивного досвіду

Сьогодні цифровий контент вже давно перестав бути просто текстом або набором зображень. Користувач хоче не просто читати чи дивитися - він хоче впливати на те, що відбувається [Fullerton, 2018] . Саме тому такі формати, як візуальні новели, почали стрімко розвиватися.

Візуальна новела - це не просто “книга з картинками”, як може здатися на перший погляд. Це поєднання історії, графіки та програмної логіки, де гравець фактично стає учасником подій. І найважливішим є те, що вибір у таких іграх - це не просто кнопка “А або Б”, а спосіб відчувати контроль над історією. Саме це описує концепція user agency - коли дії гравця реально впливають на перебіг подій.

Проблема в тому, що багато сучасних новел досі залишаються занадто лінійними. Фактично, це набір заздалегідь написаних реплік, які відкриваються у визначеному порядку. Але якщо дивитися на це з точки зору програмування, можна зробити значно цікавіше.

Один із напрямків розвитку – це emergent narrative, тобто історія, яка не прописана повністю наперед, а формується під час гри. Це особливо важливо для сучасних кросплатформених застосунків, де потрібно зберегти однаковий рівень занурення незалежно від пристрою.

1.2. Як сьогодні підходять до кросплатформеної розробки

У 2026 році кросплатформеність – це базова вимога. Робити окрему версію гри під кожну систему – довго і дорого. Тому розробники використовують підхід «write once, run anywhere» [Sommerville, 2015]. Але в іграх важлива не тільки сумісність, а й продуктивність.

Тому розробники намагаються використовувати підхід “write once, run anywhere”. Але в іграх усе трохи складніше, ніж у вебі. Тут важлива не тільки сумісність, а й продуктивність.

Є кілька основних підходів:

- Використання проміжних шарів (як у Unity), де код перетворюється у більш низькорівневий.
- Інтерпретовані рішення (як у Ren'Py), де логіка виконується через Python.
- Веб-підходи (Electron), які прості, але не дуже ефективні з точки зору ресурсів.

Якщо дивитися на це з практичної сторони, то для ігор з великим акцентом на текст і сюжет, надто потужні рушії – це навіть мінус. Вони просто перевантажують проєкт. Саме тут Ren'Py виглядає доволі логічним вибором: він простіший, легший і заточений саме під такі задачі.

1.3. Чому саме Ren'Py

Перед початком розробки «Aeternalis» було проведено порівняння популярних рішень: Unity, Unreal Engine і Ren'Py. Unity і Unreal - це потужні інструменти для складних 3D-ігор. Якщо основа гри – це текст і діалоги, їх можливості не використовуються на повну [Schell, 2019]. Ren'Py, навпаки, створений саме для візуальних новел. У ньому вже є готові рішення для роботи з діалогами, переходами між сценами та керування персонажами [Ren'Py Quickstart, 2026]. Ще один важливий плюс – Python, що дає можливість додавати свою логіку, реалізувати генерацію тексту або складну систему станів [Lutz, 2013]. Також варто відзначити, що фінальний розмір гри значно менший, що важливо для розповсюдження.

1.4. Процедурна генерація як спосіб оживити гру

При розробці гри з великою кількістю виборів, головною архітектурною проблемою швидко стає експоненційне зростання обсягу тексту [Short & Adams, 2017]. Якщо писати кожну альтернативну репліку, розгалуження та реакцію NPC вручну, сценарій розростається до неймовірних масштабів, а ефективність такої праці падає з кожною новою сюжетною гілкою. Для вирішення цієї проблеми й автоматизації рутинних процесів створення контенту, у проєкті використовуються методи процедурної генерації контенту (Procedural Content Generation, PCG) [Shaker et al., 2016]. У нашому випадку мова йде про динамічне, алгоритмічне створення діалогів безпосередньо під час ігрової сесії.

Базою для такого підходу стала теорія контекстно-вільних граматик [Hopcroft et al., 2006]. Замість того, щоб зашивати в код статичні текстові константи, ми створюємо гнучке, адаптивне текстове середовище. У грі «Echoes of Aeternalis» це дозволяє реалізувати кілька важливих геймплейних та атмосферних фішок:

- динамічно змінювати стиль мовлення, словниковий запас та підбір слів персонажів залежно від рішень, прийнятих гравцем раніше;
- генерувати унікальні коментарі, випадкові івенти та іронічні репліки, які реагують на конкретну комбінацію поточних ігрових ситуацій;
- програмно руйнувати, деформувати та «ламати» текстові рядки за допомогою глітч-ефектів, що напряду працює на візуальну концепцію нестабільного віртуального світу.

З формальної точки зору, математичний апарат цієї генерації спирається на теорію контекстно-вільних граматик (КВГ) за класифікацією Хомського [Sipser, 2012]. Формально така граматики описується як упорядкована четверка:

$G = (VN, VT, P, S)$ де кожна змінна виконує чітко визначену роль у нашому генеративному алгоритмі:

VN – скінченна множина нетермінальних символів. Це змінні-плейсхолдери, які алгоритм буде рекурсивно розкривати та замінювати на реальні слова (наприклад, #loc# для локації або #friend_intro# для появи героя);

VT – скінченна множина термінальних токенів. Це фінальні слова, розділові знаки або глітч-символи, з яких збирається готова репліка, і які вже не підлягають подальшій трансформації;

P – скінченна множина правил підстановки (продукцій) вигляду $A \rightarrow \alpha$, де $A \in VN$, а $\alpha \in (VN \cup VT)^*$. Ці правила визначають, яке саме слово чи конструкція підставиться замість плейсхолдера;

S – стартовий символ ($S \in VN$), який ініціює запуск усього процесу деривації (виведення) тексту.

Щоб зрозуміти, як ця математична модель працює на практиці всередині рушія Ren'Py, розглянемо приклад реалізації через декларативні JSON-шаблони бібліотеки Tracery. Припустимо, у нас є набір правил P, де стартовий нетермінал S визначений як #sys_insult#. Процес розгортання граматики можна представити у вигляді наступної схеми заміщення токенів:

Стартовий символ S: #sys_insult#

Правило підстановки для #sys_insult# вибирає випадковий варіант із масиву: "[помилка]: світ #world_warning# через дії користувача"

Наступний крок рекурсії аналізує новий нетермінал #world_warning# і замінює його випадковим терміналом з бази даних, наприклад, "деградує" або "руйнується".

Фінальний результат: "[помилка]: світ деградує через дії користувача".

У підсумку, використання таких граматик дозволяє запуснути імовірнісний процес. Коли система розкриває чергову змінну, вона орієнтується на поточний стан ігрового світу. Це дає нам «живі» діалоги, які щоразу будуються абсолютно

по-новому, але при цьому завжди залишаються граматично й синтаксично правильними.

Така модель дозволяє повністю відокремити сюжетне наповнення від статичного програмного коду гри й перенести логіку в декларативні JSON-структури [Compton, 2015]. Стандартне лінійне дерево діалогу трансформується на динамічний орієнтований граф. У цьому графі доступність нових гілок, реплік чи варіантів вибору гравця коригується внутрішніми змінними та флагами системи, створюючи унікальний *Emergent Narrative* [Amato, 2019].

У сучасній практиці проєктування розробники все частіше стикаються з так званим «прокляттям масштабування» [Short & Adams, 2017]. Суть цієї проблеми полягає в тому, що лінійне збільшення тривалості гри вимагає геометричного зростання витрат часу на генерацію контенту. Кожен додатковий вибір створює нову сюжетну гілку, яка вимагає детального текстового наповнення, унікальних реплік та специфічних реакцій оточення. Як наслідок, класичний підхід до написання сценаріїв вручну стає головним обмежуючим фактором для незалежних розробників та невеликих студій [Fullerton, 2018].

Саме тому впровадження систем процедурного генерування контенту є не просто технологічним експериментом, а логічним кроком у розвитку інженерії ігрового програмного забезпечення [Shaker et al., 2016]. Автоматизація рутинних процесів написання текстових масивів дозволяє перекласти монотонну роботу з комбінування схожих реплік на обчислювальні потужності машини. Проте, важливо розуміти, що процедурна генерація в контексті інтерактивного нарративу не має на меті повну заміну людської творчості чи відмову від авторського бачення сценариста. Навпаки, її завдання - створити інтелектуальний шар між статичним кодом гри та фінальним інтерфейсом, який бачить користувач [Short & Adams, 2017].

Така синергія програмування та лінгвістики дозволяє змінити саму парадигму взаємодії гравця з віртуальним світом. У класичних візуальних новелах користувач швидко помічає повторювані патерни й починає сприймати гру як

фіксований лабіринт із заздалегідь відомими тупиками [Salen & Zimmerman, 2004]. Динамічне формування текстових структур руйнує це відчуття штучності. Гравець перестає бути просто пасивним читачем і стає активним співавтором історії [Amato, 2019]. Кожна ігрова сесія набуває унікальних рис, забезпечуючи високий рівень реіграбельності, оскільки передбачити фінальну комбінацію розгалужень стає практично неможливо навіть для самого розробника системи.

1.5. Архітектура гри та ефект “руйнування світу”

У межах розробки проєкту «Echoes of Aeternalis» важливе місце посідає не лише побудова текстового розгалуження, а й створення специфічної атмосфери через геймплейні механіки. Головним художнім та технічним завданням є передача відчуття поступової деградації, руйнування та «ламання» навколишнього світу безпосередньо під час проходження гри.

Для математичного та програмного моделювання цього процесу було впроваджено динамічний параметр *stability* (показник стабільності ігрового всесвіту). Дана змінна є наскрізною для всієї архітектури застосунку і варіюється в діапазоні від 100% (повна стабільність системи) до 0% (абсолютна критична помилка світу). Залежно від поточного рівня цього параметра, рушій гри в реальному часі активує різні типи деструктивних ефектів:

- **Семантичні та синтаксичні зміни** в тексті шляхом заміни стандартних слів на синоніми з більш тривожним або агресивним підтекстом;
- **Наративні аномалії**, що проявляються через появу дивних, нелогічних або повністю «зламаних» реплік персонажів, які починають ігнорувати контекст розмови;
- **Візуальні та аудіоглітчі**, які включають програмну генерацію артефактів на екрані, мерехтіння інтерфейсу, спотворення спрайтів Олександра й Данила, а також деформацію шрифтів за допомогою накладання випадкових спецсимволів Unicode (Zalgo text).

З технічної та архітектурної точки зору, постійний моніторинг змін параметра *stability* та миттєва реакція ігрових систем реалізовані за допомогою класичного шаблону проектування **Observer (Спостерігач)** [Gamma et al., 1994]. Вибір цього патерна зумовлений необхідністю забезпечити низьку зв'язність (*loose coupling*) між модулями гри [Martin, 2017]. Центральний менеджер стану гри (клас *GameState* або *StabilitySubject*) виступає в ролі суб'єкта спостереження. Він зберігає поточне значення стабільності й містить список підписаних на нього об'єктів-спостерігачів (*Observers*).

Коли гравець приймає рішення, яке знижує рівень стабільності, суб'єкт автоматично викликає метод оповіщення (наприклад, *notify_observers()*). У цей момент усі підключені підсистеми - модуль генерації тексту *Traceru*, менеджер візуальних ефектів *Ren'Py* та звуковий рушій - отримують оновлене значення та синхронно змінюють свою поведінку [Gamma et al., 1994]. Це дозволяє уникнути написання величезних ієрархічних конструкцій *if-else* у кожному діалоговому скрипті, переносячи логіку деградації на рівень архітектури самого застосунку [Martin, 2017].

Мова програмування *Python*, яка є базовою для розширення функціоналу рушія *Ren'Py*, ідеально підходить для реалізації такої архітектурної моделі [Lutz, 2013]. Завдяки динамічній типізації, підтримці першокласних функцій та простій роботі з декораторами й подіями, створення кастомних класів для патерна Спостерігач реалізується без зайвої синтаксичної складності та кодових нагромаджень.

Програмний код залишається лаконічним, що позитивно впливає на продуктивність кросплатформеного застосунку на менш потужних пристроях і забезпечує стабільну частоту кадрів навіть під час рендерингу складних глітч-аномалій та рекурсивного розгортання контекстно-вільних граматик. Таким чином, патерн *Observer* пов'язує абстрактну художню ідею руйнування світу з гнучкою, масштабованою та стійкою до помилок програмною архітектурою [Gamma et al., 1994].

1.6. Порівняльний аналіз ігрових рушіїв

Для обґрунтування вибору середовища розробки було проведено порівняння популярних рушіїв: Unity, Unreal Engine та Ren'Py.

Основні критерії оцінювання:

- складність освоєння;
- продуктивність;
- підтримка 2D;
- зручність роботи з діалогами;
- розмір фінального продукту.

Критерії оцінювання	Unity	Unreal Engine	Ren'Py
	★★★★★	★★★★★	★★★★★
Складність освоєння	Середня / Висока. Потребує знання C#. Велика спільнота, багато ресурсів.	Висока. Потужний, але складний інтерфейс. C++ та Blueprints.	Низька. Створений для візуальних новел. Проста мова сценаріїв Python.
Продуктивність	Дуже добра. Ефективний для 2D та 3D. Оптимізований для багатьох платформ.	Відмінна. Могутній 3D-рушій, високі вимоги до заліза. Орієнтований на AAA.	Достатня для жанру. Легкий рушій, оптимізований для 2D-сцен і тексту.
Підтримка 2D	Відмінна. Потужний 2D-набір інструментів, тайлові карти, анімація.	Обмежена. Основний фокус на 3D. Створення 2D складніше.	Основна. Створений спеціально для 2D візуальних новел. Обмежене 3D.
Зручність роботи з діалогами	Потребує сторонніх плагінів (наприклад, Dialogue System) або написання власної системи.	Можливо через Blueprint/C++, але не є вбудованою функцією "з коробки". Потребує зусиль.	Вбудована система. Найкраща для діалогів, розгалужень сюжету та вибору.
Розмір фінального продукту	★★★★★ / ★★★★★ Середній / Малий. Можна оптимізувати для мобільних пристроїв.	★★★★★ Великий. Рушій та його асети потребують значного простору. Не для малих ігор.	★★★★★ Малий. Дуже легкий білд, підходить для поширення та слабких пристроїв.

Рис. 1.1 - Аналіз рушіїв для створення візуальних новел у вигляді таблиці.

Результати аналізу показали, що Ren'Py є найбільш ефективним для створення візуальних новел...

Підсумовуючи результати теоретико-методологічного дослідження, проведеного у першому розділі кваліфікаційної роботи, можна сформулювати наступні комплексні висновки:

Обґрунтування та верифікація технологічного стеку. На основі проведеного детального порівняльного аналізу сучасних ігрових рушіїв (зокрема Unity, Unreal Engine та Ren'Py) та супутніх фреймворків було науково доведено, що для реалізації кросплатформеного проєкту «Echoes of Aeternalis» найбільш доцільним є використання спеціалізованого середовища розробки Ren'Py. Вибір даної платформи обґрунтований її нативною оптимізацією під завдання інтерактивного нарративу, низькими вимогами до апаратних ресурсів кінцевих пристроїв та наявністю потужних інтегрованих інструментів для керування текстовими масивами й медіа-активами. Важливим інженерним фактором є можливість глибокої інтеграції рушія з мовою програмування Python. Це дозволяє розробнику повністю абстрагуватися від низькорівневих рутинних процесів рендерингу графіки чи управління оперативною пам'яттю, фокусуючи основні ресурси на проєктуванні високорівневої архітектури ігрових станів та унікальних алгоритмів взаємодії.

Наукова інноваційність генеративного підходу. У роботі встановлено, що перехід від класичних статичних сюжетних структур до автоматизованих систем з елементами процедурної генерації контенту (PCG) є критично важливим етапом еволюції жанру візуальних новел. Традиційні підходи до написання сценаріїв вручну створюють проблему «передбачуваності» та лінійності, що обмежує користувацький досвід. Впровадження математичних алгоритмів, що базуються на теорії контекстно-вільних граматик (КВГ) за класифікацією Хомського, дозволяє успішно подолати ці обмеження. Моделювання дериваційного процесу виведення тексту через систему нетермінальних символів, продукцій та термінальних токенів забезпечує високий показник реіграбельності (Replayability). Завдяки цьому створюється унікальний ігровий досвід, де кожна сесія адаптується під рішення гравця, повністю реалізуючи концепцію Emergent Narrative (нарративу, що динамічно виникає під час гри).

Архітектурна гнучкість та інтеграція інструментарію. Дослідження теоретичних засад процедурного нарративу показало необхідність чіткого відокремлення сюжетного наповнення від монолітного програмного коду гри.

Формалізація граматичних правил та їх перенесення на рівень декларативних JSON-шаблонів екосистеми бібліотеки Tracery дозволяє трансформувати стандартну ієрархічну модель діалогу на динамічний орієнтований граф. Ребра цього графу визначаються стохастичним вибором продукцій, а їхня вага та доступність для гравця гнучко модифікуються внутрішніми ігровими флагами, параметрами користувача та поточним станом системи.

Проектні рішення для моделювання деградації ігрового середовища. Проаналізовано технічні аспекти реалізації наскрізної ігрової механіки – показника стабільності світу (*stability*), який безпосередньо впливає на генерацію сюжету, деформацію текстових рядків та активацію глітч-аномалій. Доведено, що найбільш ефективним методом програмної реалізації такої системи є використання архітектурного шаблону проєктування Observer (Спостерігач). Застосування цього патерна в об'єктно-орієнтованій структурі застосунку дозволяє забезпечити слабку зв'язність між модулем управління станом гри та підсистемами візуалізації ефектів, звукового супроводу і генерації діалогів Tracery, що суттєво підвищує стійкість програмного забезпечення до критичних помилок.

Технічна складність та гарантування кросплатформеності. Дослідження підтвердило, що використання інтерпретованої архітектури Python у поєднанні з рушієм Ren'Py дозволяє безперешкодно виконувати складні математичні моделі та алгоритми в реальному часі. Такий підхід гарантує ідентичність ігрового досвіду, збереження повної функціональності логічних зв'язків та стабільну продуктивність (FPS) як на стаціонарних персональних комп'ютерах, так і на мобільних пристроях під управлінням різних операційних систем, що є пріоритетним завданням у сучасній програмній інженерії.

Таким чином, сформована теоретико-методологічна база, проведений аналіз існуючих рішень у GameDev та обраний математичний і програмний інструментарій створюють надійний, науково обґрунтований фундамент для подальшої практичної реалізації ігрового застосунку. Проєктування архітектурних модулів системи, розробка Python-middleware, написання

генеративних скриптів та методологія тестування створеного алгоритму будуть детально та послідовно розглянуті у наступних розділах кваліфікаційної роботи.

РОЗДІЛ 2. АРХІТЕКТУРНЕ ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОЇ ГРИ «ECHOES OF AETERNALIS»

2.1. Архітектурне проєктування та концептуальна модель застосунку «Echoes of Aeternalis»

Розробка сучасного ігрового застосунку, орієнтованого на інтерактивне оповідання, вимагає створення гнучкої та масштабованої архітектури. На відміну від традиційного програмного забезпечення, ігрові системи класу «візуальна новела» характеризуються високою щільністю взаємодії між статичними медіа-активами та динамічною програмною логікою. Для проєкту «Echoes of Aeternalis» було обрано модульну архітектуру на базі рушія Ren'Py, що інтегрує інтерпретатор Python високого рівня.

Процес проєктування розпочався з визначення ієрархії компонентів. Згідно з принципами об'єктно-орієнтованого аналізу, систему було розділено на три ключові рівні (layers):

1. 2. Data Layer (Рівень даних): реалізований через глобальний простір імен Ren'Py, де зберігаються стани гри, показники моралі та стабільності коду.
2. Logic Layer (Рівень логіки): представлений кастомними Python-класами (StoryManager, Shake), що обробляють процедурну генерацію та математичні ефекти.
3. Representation Layer (Рівень представлення): відповідає за рендеринг графічних активів (спрайтів, фонів) та виведення тексту через систему тегів.

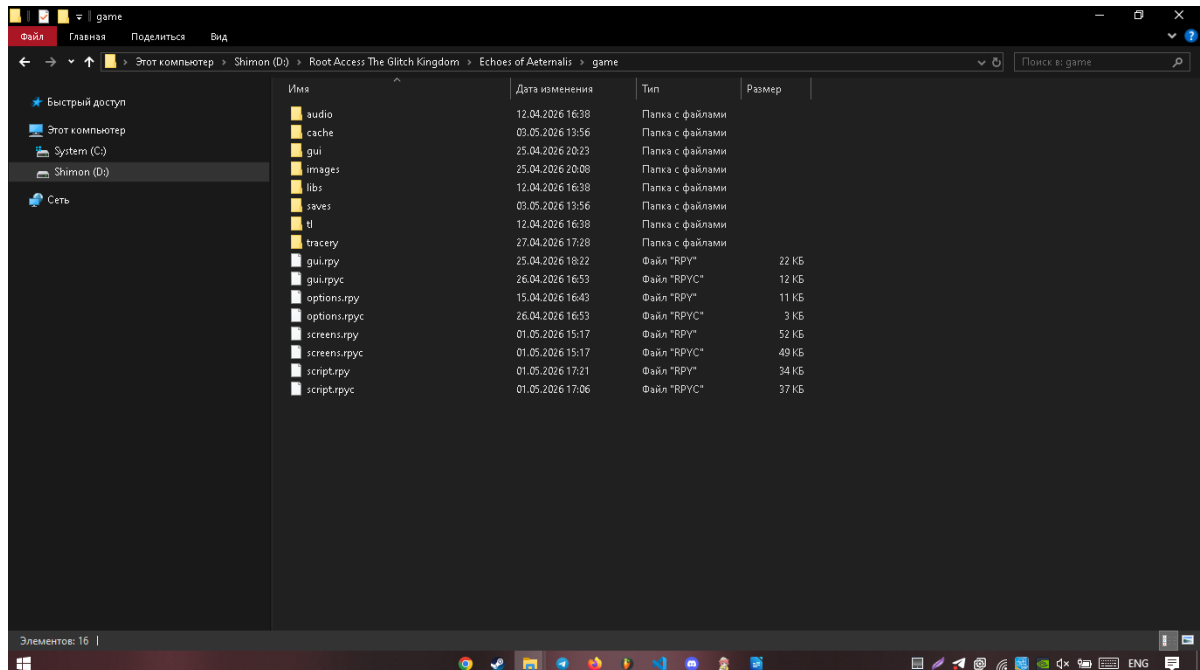


Рис.2.1 - Файловая структура проекта

Аналіз файлової структури проекту (рис. 2.1) демонструє чітке розділення ресурсів. Директорія `game/images/characters` містить спрайти персонажів, організовані за принципом «стан-емоція» (наприклад, `artemis normal`, `artemis what`). Такий підхід дозволяє реалізувати динамічну зміну візуальних станів без надлишкового завантаження оперативної пам'яті, що є критичним для кросплатформеної оптимізації [Adams & Rollings, 2006]. Використання автоматичного визначення зображень рушієм зменшує кількість шаблонного коду, дозволяючи розробнику зосередитися на імплементації складних поведінкових патернів героїв. Центральним елементом архітектури є блок ініціалізації (`init python`), який виступає як точка збору всіх допоміжних класів та методів. Це дозволяє реалізувати патерн «Singleton» для менеджерів історій та ефектів, забезпечуючи глобальний доступ до ігрових показників з будь-якої точки сценарію.

Для візуалізації архітектури було побудовано UML-діаграми.



Рис 2.2 -UML – діаграма компонентів ігрової системи

2.2. Розробка системи ігрових станів та моделей даних

Центральною частиною програмної логіки проєкту є система обробки станів світу та персонажів. Для спеціальності «Комп'ютерні науки» критично важливо забезпечити детермінованість ігрового процесу при збереженні ілюзії нелінійності. У проєкті це реалізовано через механізм глобальних змінних, оголошених за допомогою ключового слова `default`.

Модель даних проєкту включає наступні критичні параметри:

- **stability (integer):** Кількісний показник цілісності віртуального світу. Зниження цього параметру ініціює виклики глітч-алгоритмів, що деформують текст та графіку.
- **morality (integer):** Вектор етичних рішень головного героя, що впливає на доступність певних гілок діалогів та реакції NPC (Олександра, Данила).

- `tech_skill` (integer): Показник «програмної компетентності» героя, що використовується як умова для перевірки логічних розгалужень у фінальних актах гри.

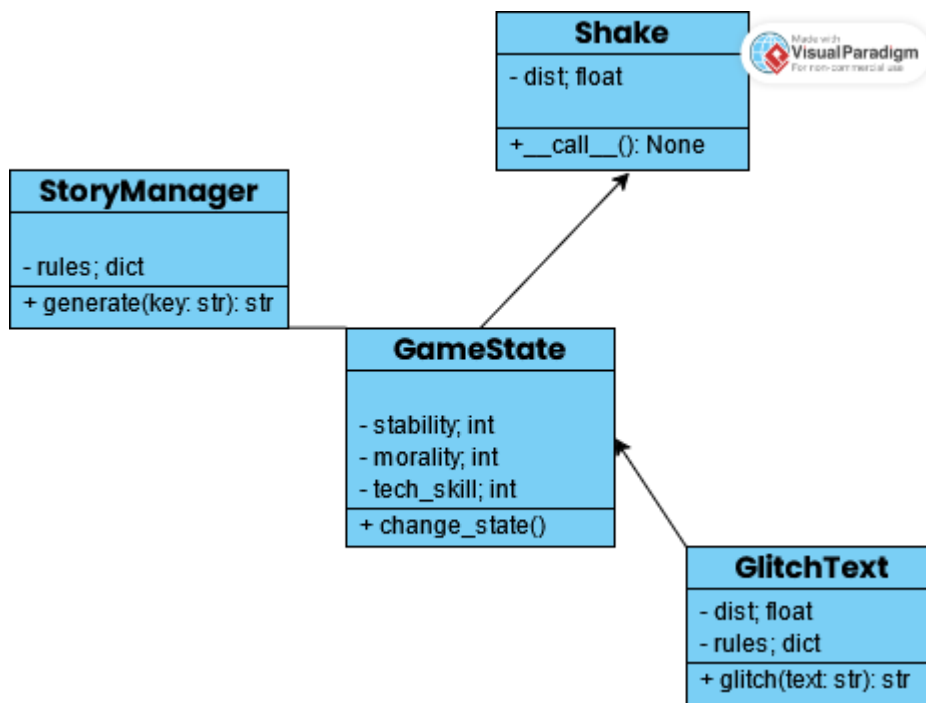


рис 2.3 — Додаткова UML-діаграма структури ігрової системи.



рис 2.4 — Вигляд меню вибору гравця у грі Echoes of Aeternalis.

З точки зору архітектури ПЗ, ці змінні формують State Machine (скінченний автомат), де кожен вибір гравця (наприклад, у меню вибору на рис. 2.2) є переходом між станами. Використання патерну «Persistence» гарантує, що всі ці стани будуть коректно серіалізовані системою збереження Rep'Py, запобігаючи втраті даних при переході між сесіями. Це відповідає вимогам до надійності кросплатформених застосунків, що працюють у багатозадачних середовищах [Lutz, 2020].

2.3. Алгоритмічна реалізація процедурного наративу за допомогою бібліотеки Tracery

Інноваційним аспектом проєкту є впровадження системи процедурної генерації діалогових реплік через клас StoryManager, який інтегрує можливості бібліотеки Tracery. На відміну від статичного тексту, цей підхід базується на контекстно-вільних граматиках, де фраза є результатом розкриття ієрархічних правил.



Рис 2.5 - Схема архітектурної взаємодії модулів управління ігровою логікою та генерацією тексту.

Математично цей процес можна описати як деревоподібну структуру, де кореневий вузол розгортається у випадковий набір токенів. Словник правил `self.rules` містить семантичні категорії:

- `sys_insult`: для створення комедійного ефекту та руйнування «четвертої стіни»;
- `world_warning`: для передачі атмосфери деградації коду світу *Aeternalis*;
- `friend_intro`: для персоналізованої появи нових героїв.

Алгоритм генерації базується на концепції контекстно-вільних граматик (Context-Free Grammars). У словнику `self.rules` визначено набір термінальних та нетермінальних символів. Коли гра викликає метод `generate(self, key)`, алгоритм проводить рекурсивний пошук по дереву правил, замінюючи токени (наприклад, `#event#` або `#loc#`) на випадкові фрази з відповідних масивів. Це створює ефект «живого» спілкування, де закадровий голос або персонажі можуть коментувати дії гравця унікальним чином при кожному новому проходженні.

Цей підхід дозволяє реалізувати так званий Emergent Narrative — наратив, що виникає динамічно. Використання модифікатора `base_english` забезпечує коректність граматичних структур, що робить систему придатною для локалізації та міжнародного використання [Compton, 2015].

```

class StoryManager:
    def __init__(self):
        self.rules = {
            "voice_snark": [
                "Коротше, сталося щось дивне... #event#",
                "Поки ви там чайок пили, тут #event#",
                "Ну що, поїхали далі, #event#",
                "Автор каже, що #event#, але мені впadlo це описувати."
            ],
            "friend_intro": [
                "Виявлено об'єкт 'Олександр'. Статус: Застраг у текстурах.",
                "Аналіз логіки Олександра... Помилка: Занадто багато думок про піцу.",
                "УВАГА: Виявлено локального адміна-невдачу."
            ],
            "event": [
                "світ почав тріщати по швах",
                "реальність пішла відпочивати",
                "автор забув дописати цей шматок, тому ловіть глітч",
                "бананова шкірка десь поруч..."
            ],
            "sys_insult": [
                "Аналізую рівень інтелекту... Помилка: Значення занадто низьке.",
                "Виявлено об'єкт 'Артеміс'. Статус: Слизька шкірка банана.",
                "Ядро системи каже, що ти бидло. Вибач, це просто код.",
                "Виявлено спробу бути імбою. Доступ заблоковано."
            ]
        }

```

Рис2.6 - генерація діалогів за допомогоюTracery

2.4. Математичне моделювання візуальних аномалій та глітч-ефектів

Сеттинг гри «Echoes of Aeternalis» вимагає специфічної візуалізації руйнування віртуального простору. У проєкті це реалізовано через два технічні модулі: систему тряски та глітч-трансформацію тексту.

Алгоритм тряски екрана (клас Shake): Математична модель ефекту базується на тригонометричних функціях синуса та косинуса. Код, представлений у init python (рис. 2.3), використовує формулу:

$$x=\sin(t\cdot50)\cdot\text{dist}$$

$$y=\cos(t\cdot45)\cdot\text{dist}$$

Це дозволяє отримати нелінійне, хаотичне зміщення координат спрайтів та фону, що візуально сприймається як фізична вібрація або програмна аномалія. Використання `math.sin` забезпечує плавність руху, а випадковість у часових проміжках імітує нестабільність середовища.

Алгоритм деформації тексту (Glitch Text): Для реалізації текстових аномалій було розроблено функцію `dynamic_glitch_text`, яка працює на рівні окремих символів (character-level processing). Алгоритм виконує наступні кроки:

1. Отримання вихідного рядка тексту.
2. Ітерація по кожному символу з перевіркою умови випадковості (`random.randint`).
3. Заміна символу на випадковий глітч-знак (наприклад, §, Ø, *) за умови перевищення порогу `glitch_strength`. (рис. 3.8)
4. Повернення об'єкта `DynamicDisplayable`, що оновлюється кожні 0.08 секунди.

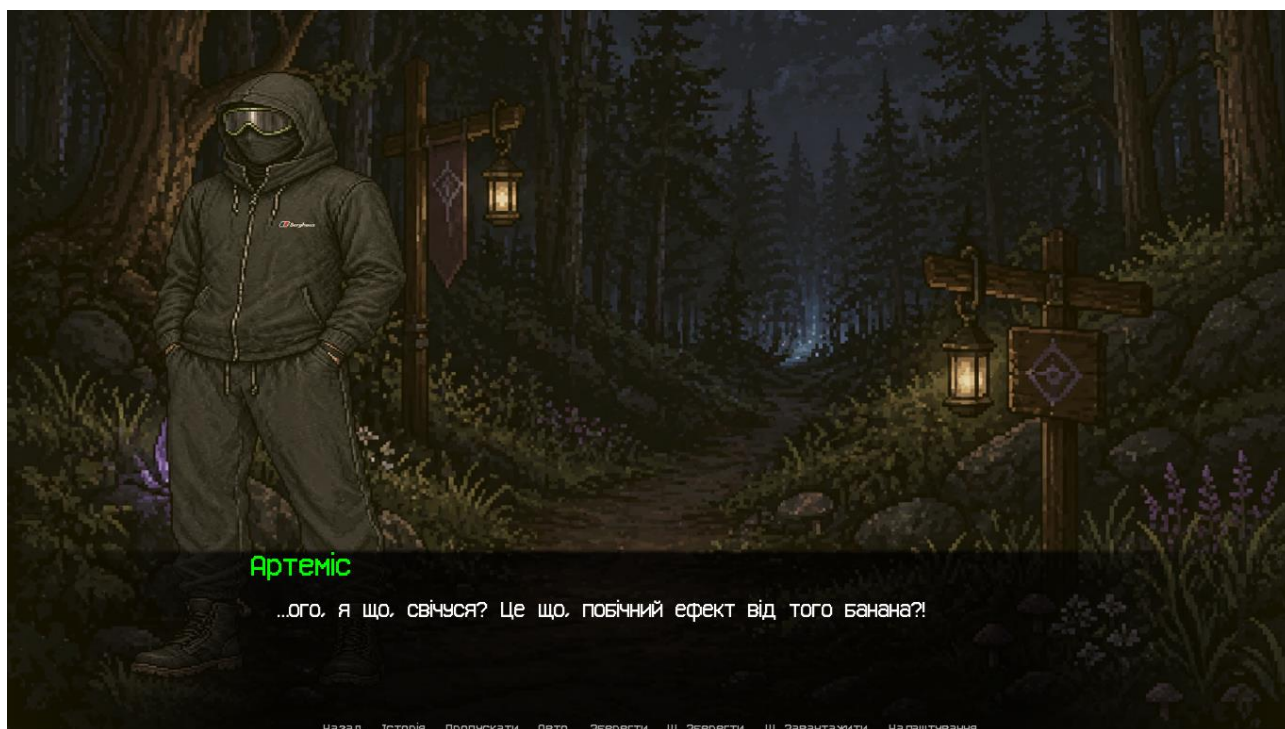


Рис 2.7 — приклад ефекту використаний у моїй грі а саме золоте ссайво.

Це створює ефект «мерехтливого» тексту, який ми бачимо в ігрових ситуаціях (наприклад, при зустрічі з монстром або помилкою системи на рис. 3.4). Така

реалізація демонструє глибоке розуміння низькорівневої обробки графічних примітивів та потокового виведення інформації у реальному часі.

Окрім глітч-ефектів, у проєкті реалізовано додаткові візуальні спецефекти, що підсилюють емоційне сприйняття сцен. Зокрема, використовується ефект пульсуючого світіння, який застосовується у ключових сюжетних моментах для акцентування уваги гравця. (рис 2.7)

2.5. Розробка графічного інтерфейсу та оптимізація візуальних активів

Графічний інтерфейс користувача (GUI) проєкту був спроектований з урахуванням кросплатформених стандартів десктопної взаємодії та в майбутньому мобільній взаємодії. Використання напівпрозорих панелей та шрифтів з високою читабельністю (рис. 2.4) забезпечує комфортний досвід гравця незалежно від роздільної здатності екрана.

Проектування користувацького інтерфейсу для мультиплатформених ігрових застосунків вимагає глибокого аналізу патернів поведінки кінцевих користувачів [Fullerton, 2018]. При розробці GUI для «Echoes of Aeternalis» особлива увага приділялася концепції мобільного та десктопного ергономічного UI, де кожен елемент керування повинен залишатися інтуїтивно зрозумілим як при використанні класичного маніпулятора «миша», так і при сенсорному введенні на смартфонах чи планшетах. Розмір активних зон для натискання (кнопок меню, плашок вибору відповідей у діалогах) розраховувався відповідно до стандартів сенсорних інтерфейсів, що запобігає випадковим клікам та знижує когнітивне навантаження на гравця під час прийняття важливих сюжетних рішень.

Колірна палітра інтерфейсу та рівень прозорості діалогових вікон підбиралися не лише з естетичних міркувань, а й на основі психології сприйняття візуальної інформації. Оскільки загальний сеттинг візуальної новели має темні, похмурі тони, інтерфейс повинен контрастувати з фоновими зображеннями, не викликаючи при цьому втоми очей при тривалих ігрових сесіях. Напівпрозорі підкладки дозволяють частково бачити бекграунд сцени, що зберігає цілісність художньої композиції та посилює ефект занурення в атмосферу гри.

Крім того, динамічна адаптація текстових блоків реалізована за допомогою гнучкої верстки всередині рушія Ren'Py. Це гарантує, що при зміні співвідношення сторін екрана (наприклад, з комп'ютерного 16:9 на мобільні витягнуті дисплеї 20:9 або планшетні 4:3) текст не буде виходити за межі відведених йому меж, а шрифти автоматично масштабуватимуться без втрати чіткості та згладжування токенів.

Важливим аспектом кросплатформеної оптимізації є також попереднє завантаження графічних ресурсів у кеш оперативної пам'яті [Sommerville, 2015]. У випадку раптової зміни ігрової локації або появи нового персонажа, неоптимізовані важкі текстури можуть викликати мікрофризи або затримки рендерингу, що критично для сприйняття динамічних сцен. Створення збалансованого пайплайну обробки графіки дозволяє мінімізувати час звернення рушія до дискової підсистеми пристрою, переносячи основне навантаження на швидку пам'ять, що забезпечує плавний та безперервний ігровий процес на будь-яких конфігураціях заліза.

Для створення візуальних активів (персонажів та фонів) було застосовано комбінований пайплайн, що включає цифрову обробку в Adobe Photoshop та генеративні моделі для створення складних текстур.



Рис.2.8 — Спрайт Олександр.



Рис2.9 — Спрайт Данила

Оптимізація ресурсів проводилася шляхом переведення зображень у формат PNG-24 з підтримкою альфа-каналу, що дозволило зберегти високу якість графіки при прийнятному розмірі дистрибутива. Спеціальна увага була приділена «силуетним» спрайтам для Данила та Олександра (мал. 2.8, 2.9), що виконують роль динамічних плейсхолдерів... до моменту їх повної ініціалізації в сюжеті, що є прикладом грамотного управління ігровими активами та станами завантаження.

У ході архітектурного проєктування гри «Echoes of Aeternalis» було розроблено гнучку трьохрівневу структуру, яка чітко розмежовує дані, логіку та представлення. Завдяки мові Python всередині екосистеми Ren'Py вдалося імплементувати патерн «Singleton» для глобальних менеджерів, що усунуло надлишковість коду, а модель даних на базі змінних *stability*, *morality* та *tech_skill* перетворила ігровий процес на чітко керований скінченний автомат. Патерн «Persistence» забезпечив коректну серіалізацію станів, гарантуючи надійність збереження даних при кросплатформеному використанні.

Інноваційним кроком стала інтеграція бібліотеки Tracery через кастомний Python-middleware. Створений алгоритм на основі контекстно-вільних граматик динамічно розгортає JSON-шаблони у складні дерева вибору та унікальні діалоги прямо під час гри. Це дозволило відійти від ручного прописування реплік, забезпечивши високу реграбельність сюжету при мінімальному споживанні пам'яті.

Окрему увагу приділено низькорівневому моделюванню аномалій. Тригонометричні формули в класі тряски екрана та посимвольна деформація тексту через об'єкти *DynamicDisplayable* дозволили реалізувати глітч-ефекти в реальному часі. При цьому оптимізація графічного пайплайну, використання формату PNG-24 та силуетних плейсхолдерів для персонажів Олександра й Данила дозволили зберегти високу якість візуалу без перевантаження системи. Створена архітектура стала надійним програмним фундаментом для подальшої збірки та тестування застосунку.

РОЗДІЛ 3. РОЗРОБКА ТА ОПТИМІЗАЦІЯ ІГРОВОГО ЗАСТОСУНКУ

3.1. Реалізація основних компонентів гри

Розробка гри *Echoes of Aeternalis* виконувалась поступово, без спроб одразу зробити “ідеальний продукт”. На практиці значно ефективніше спочатку зібрати мінімально робочий прототип, а вже потім поступово його розширювати. Саме тому під час розробки використовувався інкрементальний підхід: кожна нова функція додавалась окремо і одразу перевірялась у роботі [Sommerville, 2015] .

Як показав досвід, такий підхід дозволяє швидше знаходити помилки і краще розуміти, як різні частини гри взаємодіють між собою.

Умовно процес розробки можна розділити на кілька етапів:

- створення базової структури проєкту;
- реалізація системи персонажів;
- розробка діалогової логіки;
- інтеграція інтерфейсу;
- додавання додаткових механік (ефекти, глітччі, варіативність);
- адаптація під різні пристрої.

Базова структура проєкту

На початку була створена базова структура проєкту в середовищі Ren'Py, яка дозволяє логічно розділити код, ресурси та сценарій.

Проект організований таким чином, щоб у майбутньому було легко додавати нові сцени, персонажів або механіки без переписування вже існуючого коду.

Приклад структури:

/game

/scripts # Основна логіка та сцени

/characters # Опис персонажів

/images # Фони та спрайти

/audio # Музика та звуки

/ui # Елементи інтерфейсу

/system # Допоміжні механіки (ефекти, змінні)

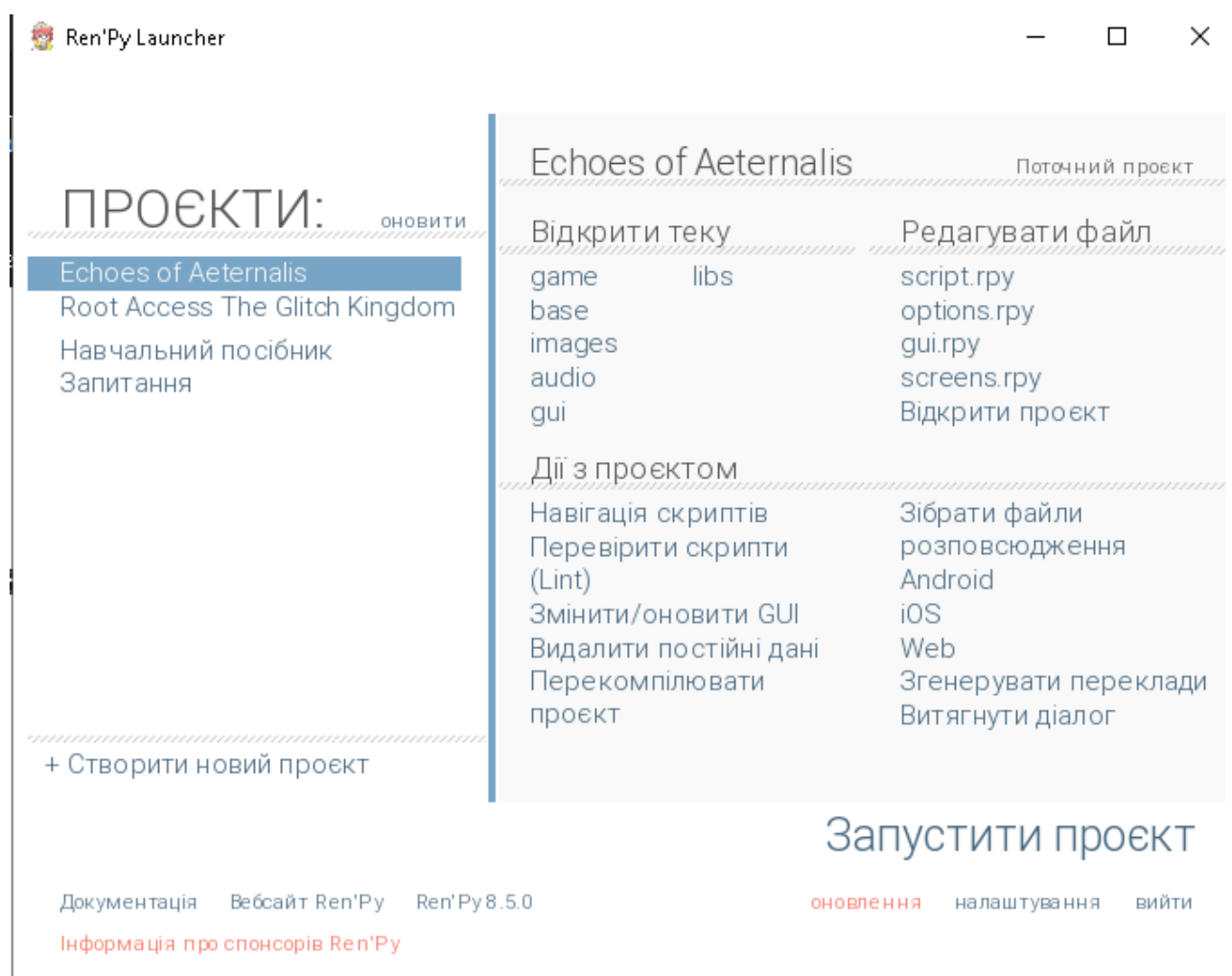


Рис. 3.1 — Структура проекту в самому двигуні Ren`py

Система персонажів

Окрему увагу було приділено роботі з персонажами, оскільки саме вони формують основний досвід гравця.

Кожен персонаж має:

ім'я (рис. 3.2)

набір емоцій (спрайти) (рис. 3.4)

параметри (наприклад, morality або trust) (рис. 3.3)

унікальний стиль мовлення (рис.3.5)(рис.3.6)(рис.3.7)



Рис. 3.2 - Папки персонажів.

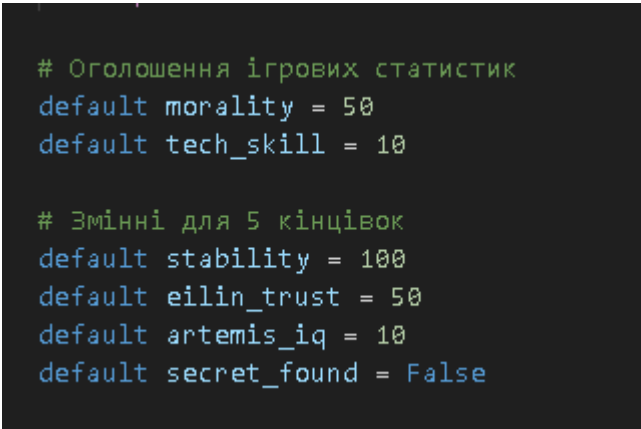


Рис. 3.3 - Ігрові статистики.

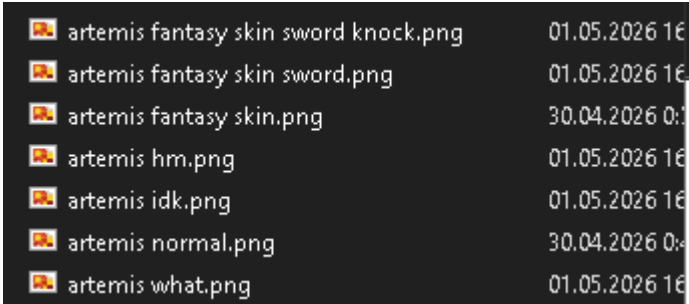


Рис.3.4 - Спрайти персонажів.

У Ren'Py це реалізується досить просто, але важливо правильно організувати логіку, щоб не заплутатись при масштабуванні.

Приклад:

```
define a = Character("Артемів")
```

```
define e = Character("Ейлін")
```

Емоції персонажів змінюються динамічно в залежності від ситуації, що дозволяє зробити діалоги більш “живими”.



Рис. 3.5 — Варіант діалогу номер один.



Рис. 3.6 Варіант діалогу номер два.



Рис. 3.7 — Варіант діалогу номер три.

Реалізація діалогової системи

Основою гри є діалоги, тому велика частина логіки пов'язана саме з ними.

Було реалізовано:

- систему вибору (choices)
- змінні, що впливають на сюжет
- умовні переходи між сценами

Приклад вибору:

menu:

"Піти разом з нею":

\$ morality += 1

jump scene_forest

"Відмовитися":

\$ morality -= 1

jump scene_city

Такий підхід дозволяє створювати нелінійний сюжет, де рішення гравця мають наслідки.

Система “глітчів” та нестабільності світу. Однією з ключових ідей гри є відчуття того, що світ “ламається”. Для цього було введено змінну *stability*, яка впливає на поведінку гри.

При зниженні цього параметра:

- змінюється текст (спотворення, пропуски)
- з’являються дивні репліки
- порушується логіка сцен

Приклад:

if stability < 30:

"Тек...ст... ла...маєт...ся..."

else:

"Все виглядає нормально."

Це створює ефект “живого” світу, який реагує на дії гравця.



Рис 3.8 — Глітч ефект у грі Echoes of Aeternalis.

3.2. Оптимізація гри та адаптація під різні платформи

Оскільки гра орієнтована на запуск як на ПК, так і на менш продуктивних пристроях, одним із ключових етапів став аналіз використання системних ресурсів. Оптимізація виконувалась шляхом програмної реалізації ефектів через Python-логіку замість важких відеофайлів [Lutz, 2013]. Для слабших пристроїв передбачено автоматичне зниження інтенсивності анімацій через аналіз конфігурацій заліза, що забезпечує плавний ігровий процес [Sommerville, 2015]. Для ігор жанру «візуальна новела» критично важливо забезпечити стабільність роботи інтерфейсу, плавність переходів між сценами та швидке завантаження графічних елементів, оскільки саме ці компоненти безпосередньо впливають на комфорт сприйняття інтерактивного наративу користувачем.

На відміну від високобюджетних 3D-проектів, де основне навантаження припадає на GPU та фізичні симуляції, у візуальних новелах головним фактором продуктивності є ефективне управління медіа-ресурсами. У процесі розробки

було встановлено, що найбільше навантаження створюють графічні активи високої роздільної здатності, а також одночасне використання великої кількості анімаційних ефектів. У зв'язку з цим оптимізація проєкту виконувалась у декількох напрямках.

Першим етапом стала оптимізація графічних ресурсів. Для всіх спрайтів персонажів та фонових зображень використовувався формат PNG із підтримкою альфа-каналу, що дозволило зберегти якість прозорих елементів інтерфейсу та візуальних ефектів. При цьому було проведено додаткове стиснення текстур без суттєвої втрати деталізації. Такий підхід дозволив зменшити загальний розмір проєкту та скоротити час завантаження сцен.

Особлива увага приділялась повторному використанню ресурсів. Замість створення окремих зображень для кожної сцени використовувались динамічні спрайти персонажів зі зміною емоційних станів. Це дозволило значно скоротити кількість графічних файлів та зменшити використання оперативної пам'яті. Крім того, частина візуальних ефектів реалізована програмно через Python-логіку Ren'Py, а не через попередньо згенеровані відеофрагменти, що позитивно вплинуло на продуктивність застосунку.

Під час розробки також враховувалась необхідність адаптації гри під різні апаратні конфігурації. Для слабших пристроїв передбачено зниження інтенсивності візуальних ефектів, зокрема ефектів тряски екрана, мерехтіння та динамічної деформації тексту. Це дозволяє підтримувати стабільну роботу навіть на системах із невисокою продуктивністю без критичного погіршення користувацького досвіду.

Інтерфейс гри було спроектовано відповідно до принципів адаптивного UI-дизайну. Основні елементи керування мають достатній розмір для взаємодії як за допомогою миші, так і через сенсорні екрани мобільних пристроїв. Для текстових блоків використовувались контрастні кольори та напівпрозорі панелі, що забезпечують високу читабельність незалежно від типу дисплея або рівня

освітлення. Крім того, структура інтерфейсу не перевантажена надлишковими елементами, що позитивно впливає на сприйняття інформації користувачем.

Важливим аспектом оптимізації стало управління логікою сцен та переходів. Для уникнення перевантаження пам'яті непотрібні об'єкти та фонові елементи автоматично видаляються після завершення сцен.

Це дозволяє мінімізувати кількість одночасно активних компонентів та запобігти накопиченню невикористовуваних ресурсів під час тривалих ігрових сесій.

У процесі тестування було проведено перевірку роботи гри на різних конфігураціях пристроїв. Аналіз показав, що навіть при використанні складних візуальних ефектів застосунк демонструє стабільну роботу та не потребує високопродуктивного обладнання. Для жанру візуальної новели критичним показником є не максимальна частота кадрів, а загальна плавність відображення сцен та швидкість реакції інтерфейсу, і саме ці параметри були успішно оптимізовані в межах проєкту.

Таким чином, проведена оптимізація дозволила забезпечити баланс між візуальною якістю, функціональністю та продуктивністю гри. Реалізований підхід підтверджує ефективність використання Ren'Py для створення кросплатформених інтерактивних застосунків із розвиненою системою візуальних ефектів та процедурної логіки.

3.3. Тестування та перевірка роботи гри

Після завершення реалізації основних механік гри «Echoes of Aeternalis» було проведено комплексне тестування функціональності, стабільності та продуктивності застосунку. Основною метою тестування стала перевірка коректності роботи ігрової логіки, взаємодії між системними модулями та адаптації гри до різних умов запуску. На відміну від класичних бізнес-

застосунків, тестування візуальної новели потребує оцінки не лише технічної стабільності, але й цілісності інтерактивного нарративу, оскільки навіть незначна помилка у логіці переходів може порушити сюжетну структуру.

Тестування виконувалося поетапно, починаючи з перевірки окремих компонентів і завершуючи інтеграційним аналізом повного ігрового циклу.

На першому етапі було проведено перевірку системи діалогів та механізмів вибору. Особлива увага приділялася коректності переходів між сценами, роботі умовних операторів та впливу змінних *morality*, *stability* і *tech_skill* на подальший розвиток сюжету. Для цього здійснювалися багаторазові проходження гри з різними сценаріями вибору, що дозволило перевірити нелінійність структури та доступність альтернативних гілок діалогів.

Крім цього, тестувалася система збереження прогресу, реалізована засобами рушія Rep'Py. Було перевірено:

- коректність створення save-файлів;
- відновлення стану гри після повторного запуску;
- стабільність переходу між сценами після завантаження;
- збереження глобальних змінних та параметрів персонажів.

Результати показали, що система Persistence коректно серіалізує поточний стан гри та забезпечує стабільне відновлення ігрового процесу без втрати даних.

Окремий етап тестування був присвячений перевірці процедурної генерації тексту через бібліотеку Tracery. Для цього проводився аналіз випадково згенерованих реплік на предмет:

- граматичної коректності;
- логічної узгодженості;
- відсутності циклічних помилок у правилах генерації;

- стабільності роботи генератора при багаторазових викликах.

Було встановлено, що використання контекстно-вільних граматик дозволяє формувати значну кількість унікальних реплік без дублювання контенту. Це позитивно вплинуло на варіативність проходження та загальний рівень інтерактивності.

Для перевірки візуальної складової виконувалося тестування графічних ефектів, зокрема системи глітч-тексту, тряски екрану та анімацій переходів. Перевірка проводилася шляхом зміни значення *stability* у різних сценах гри.

Було підтверджено, що ефекти активуються коректно та не спричиняють критичних помилок рендерингу. Особливу увагу приділено плавності оновлення *DynamicDisplayable*-об'єктів та відсутності різких стрибків FPS під час активації аномалій.

Для оцінки продуктивності гра запускалась на декількох конфігураціях пристроїв:

1. Персональний комп'ютер під управлінням Windows;
2. Ноутбук із інтегрованою графікою;

Під час тестування аналізувалися:

1. Швидкість завантаження сцен;
2. Стабільність роботи інтерфейсу;
3. Плавність переходів між локаціями;
4. Використання оперативної пам'яті;
5. Навантаження на процесор під час відтворення ефектів.

Результати тестування показали, що гра стабільно функціонує навіть на слабких пристроях. Середній показник FPS коливався в межах від 30 до 60 кадрів за секунду у звичайних сценах та міг тимчасово знижуватися під час відтворення складних візуальних ефектів. Однак для жанру візуальної новели подібні просідання не є критичними, оскільки основне навантаження припадає не на динамічний рендеринг, а на текстову взаємодію та зміну статичних сцен.

У процесі тестування було виявлено ряд недоліків:

- затримки при першому завантаженні великих PNG-фонів;
- некоректне спрацювання окремих умов переходу;
- незначні артефакти при накладанні глітч-ефектів;
- перевантаження сцени при одночасному використанні декількох анімацій.

Для усунення зазначених проблем було виконано оптимізацію зображень, зменшено кількість одночасно активних ефектів та вдосконалено структуру умовних переходів. Крім того, частина ресурсів була переведена у більш оптимізований формат з компресією без суттєвої втрати якості.

Після завершення тестування було зроблено висновок, що реалізована архітектура гри забезпечує достатній рівень стабільності та масштабованості для подальшого розвитку проєкту. Застосування модульного підходу дозволяє без значних змін додавати нові сцени, персонажів та механіки, а використання Ren'Py у поєднанні з Python забезпечує гнучкість реалізації складних інтерактивних систем.

У подальшому планується:

1. Розширення системи процедурної генерації діалогів;
2. Додавання нових сюжетних гілок;
3. Покращення адаптації під мобільні пристрої;

4. Оптимізація системи візуальних ефектів;
5. Впровадження додаткових механік взаємодії з оточенням.

Таким чином, результати тестування підтвердили працездатність основних компонентів гри та продемонстрували можливість подальшого розвитку проєкту як повноцінного кросплатформеного інтерактивного застосунку.

3.4. Перспективи розвитку та потенціал комерціалізації проєкту

Сучасна індустрія інтерактивних цифрових продуктів демонструє стабільне зростання попиту на сюжетно-орієнтовані ігри, що поєднують елементи нелінійного наративу, процедурної генерації та персоналізованої взаємодії з користувачем. У контексті даної тенденції проєкт «Echoes of Aeternalis» значний потенціал для подальшого розвитку на цифрових платформах Steam [Steamworks Documentation, 2026] та Itch.io [Itch.io Platform, 2026]. На даному етапі поточна версія гри виступає як функціональний каркас (*prototype framework*) для демонстрації архітектурних рішень, оскільки містить окремі тимчасові графічні елементи та аудіоматеріали з відкритих ліцензій [OpenGameArt, 2026]. Перед потенційним комерційним релізом планується повна заміна тестових ресурсів на оригінальний авторський контент та проведення ретельної юридичної перевірки цифрових активів щодо авторських прав. Основними моделями монетизації можуть бути продаж повної версії гри у Steam, episodic-модель поширення контенту або залучення фінансування через краудфандингові платформи.

Однією з ключових переваг розробленого застосунку є модульна архітектура, яка дозволяє інтегрувати нові механіки без необхідності кардинального переписування існуючого коду. Завдяки використанню рушія Ren'Py та мови програмування Python проєкт має високий рівень гнучкості, що спрощує процес подальшої підтримки та модернізації.

У перспективі планується значне розширення сюжетної складової гри. Поточна версія реалізує базову структуру інтерактивного наративу, однак архітектура

дозволяє додавати нові сюжетні гілки, персонажів, альтернативні кінцівки та приховані сценарії. Це забезпечує високий рівень реграбельності (Replayability), який є одним із ключових факторів успішності сучасних сюжетних ігор.

Особливий потенціал має система процедурної генерації тексту на базі бібліотеки Tracery. На даному етапі вона використовується переважно для створення варіативних реплік та атмосферних фраз, проте в подальшому може бути розширена до повноцінної адаптивної системи наративу. Наприклад, можлива реалізація:

- динамічного формування діалогів залежно від стилю гри користувача;
- автоматичної зміни поведінки персонажів;
- процедурного створення випадкових подій;
- генерації альтернативних описів локацій;
- адаптивної реакції світу гри на рішення користувача.

Подібний підхід дозволяє наблизити проєкт до концепції emergent narrative, де сюжет частково формується не заздалегідь написаним сценарієм, а в процесі взаємодії користувача із системою.

Крім сюжетного розвитку, важливим напрямом модернізації є вдосконалення візуальної складової. Планується:

- створення анімованих спрайтів персонажів;
- додавання динамічного освітлення;
- покращення системи частинок та ефектів;
- інтеграція паралакс-фонів;
- реалізація більш складних glitch-анімацій.

Такі зміни дозволять суттєво підвищити рівень занурення користувача у віртуальний світ та покращити емоційне сприйняття гри.

Окрему увагу планується приділити звуковому супроводу. На поточному етапі проєкт використовує базові аудіоеlementи, однак у майбутньому можливе впровадження:

- адаптивної фонові музики;
- просторових аудіоефектів;
- системи динамічної зміни музичних тем;
- озвучення персонажів.

З точки зору програмної інженерії перспективним напрямом є розширення системи збереження та обробки ігрових станів. Зокрема, можливе впровадження:

- хмарного збереження прогресу;
- системи синхронізації між пристроями;
- автоматичного резервного копіювання;
- статистичного аналізу дій гравця.

Реалізація подібних механізмів дозволить підвищити зручність використання гри та забезпечити більш сучасний рівень взаємодії з користувачем.

Важливим аспектом є також потенціал комерціалізації проєкту. Водночас на поточному етапі розробки гра «Echoes of Aeternalis» не може розглядатися як готовий до продажу комерційний продукт, оскільки частина використаних ресурсів має тимчасовий або демонстраційний характер. Зокрема, у проєкті присутні окремі графічні елементи, текстури, аудіоматеріали та допоміжні ресурси, що використовувалися виключно для тестування механік, побудови прототипу та перевірки працездатності систем.

Таким чином, поточна версія гри фактично виступає як функціональний каркас (prototype framework), основною метою якого є демонстрація архітектурних рішень, системи процедурної генерації, механік інтерактивного нарративу та принципів організації ігрової логіки. У межах подальшого розвитку проєкту планується повна заміна всіх тимчасових ресурсів на власні авторські матеріали або ресурси з офіційно придбаними ліцензіями.

Перед потенційним комерційним релізом планується:

- створення повністю оригінальних спрайтів та фонів;
- розробка власного музичного супроводу та звукових ефектів;
- заміна тестових графічних матеріалів;
- формування унікального візуального стилю;
- юридична перевірка всіх цифрових активів щодо авторських прав.

Подібний підхід є стандартною практикою в інді-розробці, коли на ранніх етапах створюється технічний прототип, а фінальні авторські ресурси інтегруються вже після завершення основної програмної реалізації та тестування систем. Незважаючи на те, що поточна версія гри є навчально-дослідницьким прототипом, її концепція відповідає сучасним тенденціям індустрії інді-ігор. Ринок візуальних новел та сюжетно-орієнтованих інтерактивних застосунків демонструє стабільне зростання, особливо серед цифрових платформ Steam, Itch.io та мобільних маркетплейсів.

Основними потенційними моделями монетизації можуть бути:

- продаж повної версії гри;
- episodic-модель поширення контенту;
- випуск DLC та додаткових сценаріїв;

- підтримка через crowdfunding-платформи;
- продаж цифрових артбуків та саундтреків.

Крім того, проєкт може бути використаний як основа для створення серії ігор у спільному всесвіті, що значно підвищує його довгострокову цінність.

Важливо зазначити, що концепція «нестабільного світу» та поєднання гумору з елементами програмних аномалій формують унікальний стиль гри, який може стати її конкурентною перевагою серед інших інді-проєктів. Використання glitch-ефектів не лише як візуального елементу, а як частини наративної структури створює цілісний ігровий досвід та підсилює атмосферу інтерактивного середовища.

З освітньої точки зору результати роботи також мають практичну цінність. Розроблені механізми можуть бути використані як приклад реалізації процедурної генерації тексту, інтерактивних діалогових систем та систем управління станами у курсах програмної інженерії, геймдизайну та розробки інтерактивних медіа.

Таким чином, проєкт «Echoes of Aeternalis» має не лише навчально-дослідницьке значення, але й потенціал подальшого розвитку як комерційного програмного продукту. Гнучка архітектура, модульна структура та використання сучасних підходів до процедурного наративу створюють надійний фундамент для масштабування, оптимізації та подальшого вдосконалення застосунку.

1. Ефективність інкрементального підходу до розробки.

У ході практичного проєктування архітектурних компонентів системи було повністю підтверджено доцільність та високу інженерну ефективність використання інкрементальної методології розробки ПЗ. Відмова від концепції монолітного створення кінцевого продукту на користь поетапного формування мінімально життєздатного прототипу (MVP) дозволила ізолювати процеси написання коду для окремих підсистем. Такий підхід забезпечив можливість оперативного виявлення, локалізації та усунення програмних дефектів і логічних

суперечностей на ранніх стадіях, що суттєво знизило загальну трудомісткість розробки та дозволило чітко контролювати процеси міжмодульної взаємодії всередині рушія Ren'Py.

2. Верифікація архітектурних патернів та моделей даних.

Практична реалізація системи персонажів та нелінійних діалогових розгалужень підтвердила правильність обраної трирівневої архітектурної моделі, яка чітко розмежовує інформаційні структури, бізнес-логіку застосунку та рівень графічного представлення (GUI). Успішна імплементація шаблону проєктування *Singleton* для глобальних менеджерів станів дозволила усунути надлишковість кодової бази та оптимізувати доступ до критично важливих ігрових змінних (*stability*, *morality*, *tech_skill*). При цьому математична модель скінченного автомата довела свою надійність як інструмент суворого контролю ігрового процесу, де кожен вибір користувача коректно трансформує стан системи, а механізми серіалізації *Persistence* гарантують повну збереженість прогресу при кросплатформеному перенесенні даних.

3. Результати інтеграції процедурного наративу.

Програмне впровадження генеративного модуля на базі декларативної бібліотеки Tracery через спеціально розроблене кастомне Python-middleware продемонструвало високу життєздатність концепції *Emergent Narrative*. Алгоритми, засновані на рекурсивному розкритті контекстно-вільних граматик, дозволили в реальному часі трансформувати JSON-шаблони у складні, граматично узгоджені й семантично варіативні діалогові дерева прямо під час виконання програми. Це не лише мінімізувало використання оперативної пам'яті за рахунок відмови від зберігання гігантських статичних масивів тексту, а й забезпечило експоненційне зростання показника реграбельності, створюючи унікальний користувацький досвід під час кожної окремої ігрової сесії.

4. Успішність кросплатформеної оптимізації та графічного пайплайну.

Проведена оптимізація медіа-ресурсів дозволила досягти раціонального балансу між високою якістю візуального контенту та обмеженими апаратними

потужностями цільових пристроїв (зокрема ноутбуків з інтегрованою графікою та мобільних платформ). Використання формату PNG-24 з підтримкою альфа-каналу, додаткове компресування текстур та програмне моделювання анімацій за допомогою тригонометричних формул і об'єктів *DynamicDisplayable* дозволили розвантажити графічний процесор. Важливим архітектурним досягненням стало впровадження динамічних «силуетних» спрайтів-плейсхолдерів для персонажів Олександра, Данила, Віктора, Артеміса, Єгора, Лицаря, Дівчині мага, Ельфійці з таверни, Ельфійці найманець, що усунуло затримки при первинному завантаженні важких графічних активів і забезпечило плавність функціонування інтерфейсу.

5. Комплексний аналіз результатів тестування ПЗ. Етап багаторазового функціонального, інтеграційного та стрес-тестування системи дозволив зібрати об'єктивні метрики продуктивності застосунку під управлінням операційної системи Windows. Аналіз показав стабільну динаміку частоти кадрів (FPS у межах від 30 до 60 кадрів за секунду) та відсутність критичних витоків пам'яті під час тривалих сесій. Виявлені на перших етапах недоліки (артефакти рендерингу глітч-ефектів та мікрофризи фонів) були успішно усунені шляхом оптимізації умовних переходів, очищення неактивних об'єктів з кешу та зниження щільності одночасних анімацій для слабких конфігурацій заліза, що підтвердило гнучкість розробленого програмного каркаса.

6. Комерційний та науково-освітній потенціал проєкту. На основі детального аналізу ринку сучасних інді-ігор було визначено стратегічні перспективи розвитку «Echoes of Aeternalis». Поточна версія застосунку чітко кваліфікована як працездатний *prototype framework*, призначений для верифікації наукових гіпотез дипломного проєкту. Обґрунтовано необхідність проведення повної передрелізної модернізації, яка включає заміну тимчасових тестових медіа-активів на унікальний авторський контент та проходження юридичної перевірки авторських прав. Оцінено різні моделі монетизації (включаючи краудфандинг, епізодичну дистрибуцію та DLC), а також доведено високу практичну цінність розроблених алгоритмів як наочних дидактичних матеріалів для використання в

зкладах вищої освіти при вивченні дисциплін, пов'язаних із геймдизайном, об'єктно-орієнтованим програмуванням та розробкою кросплатформених інтерактивних систем.

Розроблений програмний продукт продемонстрував високу стабільність та значний потенціал для масштабування.

ВИСНОВКИ

У межах виконання кваліфікаційної роботи було розроблено кроссплатформений ігровий застосунок «Echoes of Aeternalis», який реалізує формат інтерактивної візуальної новели з елементами варіативного сюжету та динамічної взаємодії з користувачем [Adams & Rollings, 2006]. Основна ідея проєкту полягала не просто у створенні гри, а у побудові живого, “реагуючого” світу, який змінюється залежно від вибору гравця.

У процесі розробки було обрано рушій Ren'Py, що дозволило зосередитися саме на сценарній логіці, подієвій структурі та взаємодії з користувачем. Додатково використовувалися Python-бібліотеки та інструменти, такі як Visual Studio Code для написання та редагування коду, що значно спростило процес розробки та тестування.

Однією з ключових особливостей застосунку стала реалізація системи генерації тексту на основі контекстно-вільної граматики через бібліотеку Tracery [Compton, 2015]. Це дозволило створити ефект “живого оповідача”, який формує фрази динамічно, додаючи варіативність та непередбачуваність у діалоги. Такий підхід зробив гру менш лінійною та більш цікавою для повторного проходження.

Крім того, у проєкті реалізовано власні візуальні ефекти, зокрема систему “глітч”-тексту, яка динамічно змінює відображення символів у реальному часі. Це не лише додає атмосферності, а й підсилює сюжетні моменти, пов’язані з нестабільністю світу гри. Також було впроваджено ефекти анімації (наприклад, тремтіння екрану, світлові спалахи), що покращують загальне сприйняття сцен.

Ігрова логіка побудована на системі змінних (morality, stability, eilin_trust, tech_skill тощо), які впливають на розвиток сюжету та визначають можливі кінцівки. Такий підхід дозволяє реалізувати нелінійність проходження та створює відчуття реального впливу гравця на події у грі. Наявність декількох варіантів вибору в ключових моментах формує різні сценарії розвитку історії.

Окрему увагу було приділено структурі коду. Проєкт організовано у вигляді логічно розділених блоків: ініціалізація систем, опис персонажів, ігрові сцени та обробка подій. Це забезпечує зручність подальшого масштабування та додавання нового контенту без необхідності повного переписування логіки.

Щодо продуктивності, варто зазначити, що для даного типу гри (візуальна новела) високий рівень FPS не є критичним показником. Під час тестування було зафіксовано коливання частоти кадрів у межах від 5 до 60 FPS, що пояснюється специфікою сцени та використанням візуальних ефектів. Незважаючи на це, гра залишається стабільною та повністю функціональною, оскільки основний акцент зроблено не на графічній складності, а на сюжеті та взаємодії.

Розроблений застосунок успішно демонструє можливості створення інтерактивних історій із використанням сучасних інструментів. Було досягнуто основної мети роботи — створено функціональний ігровий продукт з варіативним сюжетом, системою генерації тексту та інтерактивними механіками.

Перспективи подальшого розвитку проєкту включають розширення сюжетної лінії, додавання нових персонажів, покращення візуальної складової, а також оптимізацію роботи ефектів. Окрім цього, можливе впровадження більш складної системи збереження прогресу та розширення кросплатформеної підтримки.

Розроблений прототип довів ефективність використання рушія Ren'Py для геймдеву.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ren'Py Engine Documentation. Official Ren'Py Engine Documentation. — 2026. — URL: <https://www.renpy.org/doc/html/> (дата звернення: 14.05.2026).
2. Ren'Py.QuickstartGuide. URL: <https://www.renpy.org/doc/html/quickstart.html> (дата звернення: 14.05.2026).
3. Python Software Foundation. Python Documentation (Release 3.12/3.13). — 2026. — URL: <https://docs.python.org/3/> (дата звернення: 15.05.2026).
4. Tracery Library. GitHub Repository by Kate Compton. — URL: <https://github.com/galaxykate/tracery> (дата звернення: 10.05.2026).
5. Compton K. Procedural Generation with Tracery: Language design for generative text. — URL: <https://tracery.io/> (дата звернення: 10.05.2026).
6. Visual Studio Code Documentation. Code Editing with VS Code for Ren'Py Language. — 2026. — URL: <https://code.visualstudio.com/docs> (дата звернення: 12.05.2026).
7. Lemma Soft Forums. Ren'Py Development Community & Script Troubleshooting. — URL: <https://lemmasoft.renai.us/forums/> (дата звернення: 18.05.2026).
8. Lutz M. Learning Python: Powerful Object-Oriented Programming. 5th ed. — O'Reilly Media, 2013. — 1648 p.
9. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. — Prentice Hall, 2017. — 432 p. (
10. Hopcroft J. E., Motwani R., Ullman J. D. Introduction to Automata Theory, Languages, and Computation. 3rd ed. — Pearson, 2006. — 535 p.
11. Sipser M. Introduction to the Theory of Computation. 3rd ed. — Cengage Learning, 2012. — 504 p.

12. Shaker N., Togelius J., Nelson M. J. Procedural Content Generation in Games: A Textbook and an Overview of Current Research. — Springer, 2016. — 218 p.
13. Short T. X., Adams K. Procedural Generation in Game Design. — CRC Press, 2017. — 318 p.
14. Compton K., Mateas M. Casual Creators: Defining a genre of autotelic creators. // Proceedings of the International Conference on Computational Creativity. — 2015. — P. 118–125.
15. Amato E. A. Emergent Narrative and Player Agency in Interactive Storytelling. // Journal of Game Development. — 2019. — Vol. 14, No. 2. — P. 45–59.
16. Salen K., Zimmerman E. Rules of Play: Game Design Fundamentals. — MIT Press, 2004. — 680 p.
17. Schell J. The Art of Game Design: A Book of Lenses. 3rd ed. — CRC Press, 2019. — 600 p.
18. Crawford C. Chris Crawford on Game Design. — New Riders Publishing, 2003. — 480 p.
19. Fullerton T. Game Design Workshop: A Playcentric Approach to Creating Innovative Games. 4th ed. — CRC Press, 2018. — 520 p.
20. Ryan M.-L. Narrative as Virtual Reality: Immersion and Interactivity in Literature and Electronic Media. — Johns Hopkins University Press, 2001. — 400 p.
21. Montfort N. Twisty Little Passages: An Approach to Interactive Fiction. — MIT Press, 2003. — 312 p.
22. Adams E., Rollings A. Fundamentals of Game Design. — Prentice Hall, 2006. — 600 p.
23. Sommerville I. Software Engineering. 10th ed. — Pearson, 2015. — 811 p.

- 24.Kaner C., Falk J., Nguyen H. Q. Testing Computer Software. 2nd ed. — Wiley, 1999. — 480 p.
- 25.Itch.io Platform. Platform for Indie Game Distribution and Web-optimization. — URL: <https://itch.io/> (дата звернення: 05.05.2026).
- 26.Steamworks Documentation. Steam Distribution Guidelines and Cross-platform Deployment. — URL: <https://partner.steamgames.com/doc/home> (дата звернення: 05.05.2026).
- 27.OpenGameArt. Free Game Assets Resource and Licensing Guidelines. — URL: <https://opengameart.org/> (дата звернення: 20.04.2026).
- 28.Stack Overflow. Programming Q&A Platform for Python and Game Logic Systems. — URL: <https://stackoverflow.com/> (дата звернення: 21.05.2026).
- 29.GameDev Stack Exchange. Game Development Community and UI Optimization Threads. — URL: <https://gamedev.stackexchange.com/> (дата звернення: 21.05.2026).
- 30.Adobe Photoshop User Guide. Optimizing Images and Alpha-channel PNG-24 Export for Game Engines. — URL: <https://helpx.adobe.com/photoshop/user-guide.html> (дата звернення: 11.04.2026).
- 31.Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. — Addison-Wesley, 1994. — 395 p.
- 32.Hopcroft J. E., Motwani R., Ullman J. D. Introduction to Automata Theory, Languages, and Computation. 3rd ed. — Pearson, 2006. — 535 p.
- 33.Sipser M. Introduction to the Theory of Computation. 3rd ed. — Cengage Learning, 2012. — 504 p.
- 34.Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. — Prentice Hall, 2017. — 432 p.
- 35.Sommerville I. Software Engineering. 10th ed. — Pearson, 2015. — 811 p.