МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет водного господарства та природокористування
Навчально-науковий інститут кібернетики, інформаційних технологій
та інженерії
Кафедра комп'ютерних технологій та економічної кібернетики

Допущено до захисту:
Завідувач кафедри
_____ д. е. н., професор П. М. Грицюк
«_____» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня «бакалавр»
за освітньо-професійною програмою «Інформаційні системи та технології»
спеціальності 126 «Інформаційні системи та технології»
на тему «Проектування та розробка сюжетного доповнення до гри S.T.A.L.K.E.R. Call of Pripyat»

Виконав:
Здобувач вищої освіти 4 курсу, групи ІСТ-41
Власов Владислав Андрійович

Керівник:
к. т. н, доцент Джоші О.І.

Рецензент:
к. т. н, доцент Гладка О.М.

Рівне – 2025

Анотація

Власов В. А. Проектування та розробка сюжетного доповнення до гри S.T.A.L.K.E.R. Call of Pripyat. Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр»: 66 ст., 11 рис., 9 табл., 2 додатки на 4 ст., 13 літературних джерел.

Об’єкт дослідження –-процес проектування та розробки комп’ютерних ігор, зокрема модифікацій до ігор жанру FPS на прикладі рушія X-Ray Engine.

Предмет досліджень – компоненти сюжетного доповнення до гри S.T.A.L.K.E.R.: Call of Pripyat — включно з програмною реалізацією, дизайном локацій, ігровими механіками, сценарною побудовою та користувацьким інтерфейсом (сюжетна лінія, ігрова логіка, механіки, інтерфейс, рівень та оптимізація).

Методи дослідження – аналіз літературних джерел, порівняльний аналіз, технічний аналіз, концептуальне моделювання, проєктування, програмування, експериментальний метод, метод тестування, кількісний аналіз, оптимізаційний метод

Кваліфікаційна робота присвячена проєктуванню та реалізації сюжетного доповнення до комп’ютерної гри жанру шутер від першої особи (*First-Person Shooter*), з фокусом на дослідження психологічного наративу, інтерфейсної мінімалістики та внутрішньоігрової логіки. У рамках роботи було створено модифікацію *Catalyst: Complementation* для гри *S.T.A.L.K.E.R.: Call of Pripyat*, побудовану на базі модифікованого рушія OpenXRay. Особливістю доповнення є відмова від традиційних бойових механік на користь діалогової взаємодії, сценарної циклічності, квестової структури та локаційного зонування. Проєкт охоплює всі етапи створення ігрового доповнення — від концептуального моделювання та дизайну рівнів до програмної реалізації логіки, інтеграції інтерфейсу, тестування та оцінювання якості реалізованого ігрового досвіду.

Ключові слова: рушій OpenXRay, сюжетне доповнення, S.T.A.L.K.E.R., ігрова логіка, UX-дизайн, FPS.

Зміст

ВСТУП

У сучасному цифровому світі індустрія комп’ютерних ігор перетворилася з нішевого захоплення на повноцінну культурну, економічну та технологічну сферу. Відеоігри сьогодні є не лише засобом розваги, а й формою мистецтва, наративного експерименту, соціального висловлювання та технічного виклику. Їхній вплив відчутний у кінематографі, літературі, освіті, рекламі, а сама ігрова індустрія обганяє за прибутками кіно та музичні ринки разом узяті. Особливе місце у цій екосистемі займають ігри з відкритим світом, глибоким сюжетом та можливістю модифікації [1]. Серед них — серія S.T.A.L.K.E.R., розроблена українською студією GSC Game World [2].

Модифікації до гри S.T.A.L.K.E.R. стали окремим феноменом, який об’єднав тисячі ентузіастів, програмістів, дизайнерів і гравців по всьому світу. Саме у цьому середовищі виникає потреба не лише створювати ігрові модифікації, а й систематизувати підходи до їхнього проєктування, розробки, тестування та аналізу. Кваліфікаційна робота присвячена створенню сюжетного доповнення до гри S.T.A.L.K.E.R.: Call of Pripyat — модифікації **Catalyst: Complementation**, яка є прикладом інтеграції технічної роботи, геймдизайну, сценарної логіки та креативного мислення.

Актуальність дослідження обумовлена зростаючим інтересом до ігрової розробки як складової цифрової економіки та культури. Успішне створення сюжетних доповнень вимагає знань у галузі програмування, дизайну, наративу, тестування та роботи з рушієм. В умовах обмежених ресурсів модифікації виступають як полігони для експериментів, де можливо апробувати нові ідеї, протестувати ігрові механіки та створити повноцінний геймплей. Водночас більшість доповнень страждають на слабку оптимізацію, логічні помилки, надмірну складність або відсутність цілісної ігрової концепції. Робота над **Catalyst: Complementation** покликана вирішити ці проблеми через системний підхід до проєктування.

Метою дослідження є повноцінне проєктування та реалізація сюжетного доповнення до гри S.T.A.L.K.E.R.: Call of Pripyat із використанням

модифікованого рушія OpenXRay, що включає у себе нову сюжетну лінію, ігрову логіку, механіки, інтерфейс і оптимізацію.

Для досягнення поставленої мети були визначені наступні **завдання**:

1) Провести аналіз жанрових, технічних та наративних особливостей ігор типу FPS включаючи механіки бою, структуру рівнів, способи інтеграції сюжету та взаємодію з гравцем;
2) Визначити ключові технічні компоненти рушія X-Ray Engine та можливості його модифікації;
3) Розробити нову концепцію сюжетного доповнення включаючи тематику, основних персонажів, сюжетні арки та зв'язок із оригінальною грою;
4) Реалізувати внутрішньоігрову логіку, діалоги та квестову систему для підтримки сюжетного доповнення;
5) Розробити або вдосконалити інтерфейс гри, адаптувавши його до нової сюжетної лінії та механік
6) Здійснити редизайн рівня на основі локації Кордон (build 1842);
7) Провести тестування, налагодження, балансування геймплею та оптимізацію продуктивності доповнення;
8) Оцінити якість реалізованої модифікації за основними метриками, такими як стабільність, залученість гравців, відповідність сюжету та зручність інтерфейсу.

Об'єктом дослідження є процес проектування та розробки комп'ютерних ігор, зокрема модифікацій до ігор жанру FPS на прикладі рушія X-Ray Engine.

Предметом дослідження є компоненти сюжетного доповнення до гри S.T.A.L.K.E.R.: Call of Pripyat — включно з програмною реалізацією, дизайном локацій, ігровими механіками, сценарною побудовою та користувацьким інтерфейсом (сюжетна лінія, ігрова логіка, механіки, інтерфейс, рівень та оптимізація).

Методи дослідження, що були використані в роботі, включають: аналіз літературних джерел, порівняльний аналіз, технічний аналіз,

концептуальне моделювання, проєктування, програмування, експериментальний метод, метод тестування, кількісний аналіз, оптимізаційний метод.

Наукова новизна дослідження:

1) Розроблено нову концепцію сюжетного доповнення для FPS-ігор, яка інтегрує унікальну сюжетну лінію, механіки та дизайн рівнів на базі OpenXRay.

2) Визначено та систематизовано можливості й обмеження модифікованого рушія OpenXRay для створення сюжетних доповнень, що раніше не досліджувалися в такому обсязі.

3) Запропоновано нові підходи до інтеграції наративних елементів (діалогів, квестів) і механік у FPS-ігри, адаптовані до технічних особливостей OpenXRay.

4) Розроблено й протестовано унікальний набір ігрових механік, які розширюють геймплей і взаємодію гравця з новою сюжетною лінією.

5) Сформовано методику оцінки якості сюжетних доповнень, що включає метрики стабільності, залученості та відповідності наративу.

Практичне значення дослідження:

1) Створене сюжетне доповнення може бути інтегровано в існуючу комп’ютерну гру, розширюючи її контент і підвищуючи інтерес гравців.

2) Результати аналізу OpenXRay надають розробникам практичний посібник із модифікації рушія для створення подібних доповнень.

3) Розроблені механіки, квестова система та інтерфейс можуть бути адаптовані для інших проєктів на базі OpenXRay або подібних рушіїв.

4) Оптимізовані методи тестування та балансування геймплею дозволяють підвищити якість і стабільність майбутніх модифікацій.

5) Результати дослідження можуть слугувати основою для навчання геймдизайнерів і розробників, які працюють із FPS-іграми та модифікованими рушіями.

6) Запропонована методика оцінки якості доповнень може бути застосована для аналізу та вдосконалення інших ігрових проєктів.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ ПРОЕКТУВАННЯ ТА РОЗРОБКИ КОМП'ЮТЕРНИХ ІГОР

1.1. Комп'ютерні ігри: концептуалізація поняття

Комп'ютерні ігри як феномен сучасної цифрової культури займають особливе місце серед інших форм візуального, аудіального та інтерактивного мистецтва. Стрімкий розвиток даної сфери цифрових розваг упродовж останніх десятиліть зумовив необхідність формулювання чіткого наукового визначення та розуміння цього поняття в контексті міждисциплінарних досліджень [3].

Комп'ютерна гра — це електронна гра, в ігровому процесі якої гравець використовує інтерфейс користувача, щоб отримати зворотну інформацію з відеопристрою. Електронні пристрої, які використовують для того, щоб грати, називаються ігровими платформами. Наприклад, до таких платформ належать персональний комп'ютер та гральна консоль. Пристрій введення, який використовують для керування грою, називається ігровим контролером. Це може бути, наприклад, джойстик, клавіатура та мишка, геймпад або сенсорний екран [1, 5]. Тобто це інтерактивна програмна система, створена з метою розваги, навчання, симуляції або мистецького вираження, яка передбачає взаємодію користувача (гравця) із віртуальним середовищем за допомогою інтерфейсу. Основною відмінною рисою гри є наявність чітко визначених правил, цілей та зворотного зв'язку, що стимулює залучення та участь.

Згідно з визначенням Джеспера Джулла, одного з провідних теоретиків у сфері ігрових досліджень, гра є системою, в якій гравці беруть участь у штучному конфлікті, що має встановлені правила та призводить до вимірного результату [6]. У контексті комп'ютерних ігор це визначення доповнюється цифровою природою середовища, візуальною подачею, можливістю модифікації та багатокористувацькою взаємодією.

До ключових ознак комп'ютерних ігор належать [7]:

- ***інтерактивність*** — можливість безпосереднього впливу гравця на події в ігровому середовищі;

- ***цілеспрямованість*** — наявність конкретної мети, якої має досягти гравець;
- ***правила*** — набір умов, які визначають можливі дії гравця;
- ***віртуальність*** — існування ігрового світу у цифровій формі;
- ***зворотний зв'язок*** — система винагород або покарань, що реагує на дії гравця;
- ***наративність (у більшості ігор)*** — наявність історії, яка розгортається в процесі гри.

Сучасні дослідники все частіше розглядають комп'ютерні ігри як форму мистецтва та культурного висловлення. Ігри здатні передавати складні наративи, викликати емоції, досліджувати філософські теми та створювати унікальні світи. Вони перетинають межі між кінематографом, літературою, музикою і театром, утворюючи нову форму — інтерактивне мистецтво [3].

Таким чином, комп'ютерна гра — це не просто інструмент розваги, а складна багаторівнева структура, яка поєднує програмування, дизайн, наратив і гейміфіковану взаємодію. Її аналіз вимагає системного підходу та врахування культурного, соціального й технологічного контекстів. Концептуалізація поняття «комп'ютерна гра» дозволяє глибше дослідити її значення у сучасному цифровому світі.

1.2. Класифікація комп'ютерних ігор

Класифікація комп'ютерних ігор є важливим інструментом для аналізу, проєктування та розуміння ігрових продуктів. Ігри різняться між собою за жанром, геймплейними механіками, стилем подачі, формами взаємодії, а також за метою створення [2, 9]. Така різноманітність зумовила появу кількох підходів до класифікації, серед яких найпоширенішими є жанрова, за типом взаємодії та функціональна.

1) Жанрова класифікація. Жанр гри визначає її основні геймплейні елементи, тип викликів для гравця та стиль подачі.

Найбільш популярними жанрами ігор є наступні:

Стратегії. Часто сюжетні ігри, що вимагають від гравця постійно аналізувати ситуацію для того, щоб приймати найкращі рішення. Самі по собі ігри часто щось більше, ніж просто логічні ігри, як ті ж шашки або шахи;

Аркади. Ігри, що, вимагають від геймера спритність та уважність;

Симулятори. Ігри, що є віртуальними прообразами реальних об’єктів та процесів, що пов’язані з отриманням віртуального, проте, часто, релевантного досвіду з певним транспортом, пристроєм і т.п.;

Шутери. У цих іграх, основа геймплею полягає у стрільбі зі зброї по цілям, що можуть бути як об’єктами, так і віртуальними прототипами людей. У таких іграх часто головною задачею є знищення усіх цілей на рівні або отримання найбільшої кількості очок).

Більшість сучасних ігор поєднують елементи кількох жанрів, утворюючи гібридні форми (наприклад, *Elden Ring* — це action-RPG з відкритим світом і елементами пригодницької гри [10]).

2) Класифікація за типом взаємодії:

Одиночні **(Single-player)** — розраховані на одного гравця, зосереджені на індивідуальному досвіді.

Мережеві **(Multiplayer)** — передбачають участь кількох гравців, які можуть співпрацювати або змагатися.

Кооперативні **(Co-op)** — варіант мультиплеєру, де гравці об'єднуються для спільного проходження.

MMO **(Massively Multiplayer Online)** — ігри з великою кількістю учасників у загальному онлайн-середовищі (*World of Warcraft*, *Final Fantasy XIV*).

Freeplay / Sandbox — відкриті ігрові світи без жорсткої структури сюжету, де гравець має велику свободу дій (*Minecraft*, *Garry’s Mod*).

3) Функціональна класифікація:

Розважальні — більшість комерційних ігор, створені для задоволення гравця.

Навчальні **(edutainment)** — поєднують розвагу з навчальними цілями (наприклад, *Duolingo*, *Kerbal Space Program*).

Соціальні — спрямовані на комунікацію між гравцями (наприклад, *Second Life*, *Among Us*).

Серйозні ігри **(serious games)** — створені для моделювання ситуацій, навчання, тренувань, досліджень тощо.

4) Інші критерії класифікації комп'ютерних ігор за:

Платформою (ПК, консолі, мобільні пристрої, VR/AR).

Стилістикою графіки (реалізм, піксель-арт, мультяшна графіка).

Кампанією чи сесією (лінійна сюжетна гра чи короткі матчі).

Поглядом камери (від першої особи, третьої особи, ізометричний вигляд тощо).

Отже, класифікація комп'ютерних ігор дозволяє не лише систематизувати величезну кількість ігрових продуктів, а й глибше зрозуміти їхній зміст, механіки та цільову аудиторію. Вона є необхідною умовою для подальшого аналізу, розробки та ефективного використання ігор у різних галузях — від індустрії розваг до освіти та науки.

1.3. Життєвий цикл розробки комп'ютерної гри для жанру «Шутер від першої особи»

Шутер від першої особи (від англ. First-person shooter), або FPS — жанр відеоігор (підвид шутерів), де основна частина ігрового процесу це знищення ворогів із різноманітної вогнепальної зброї. Характерною особливістю жанру є вид «з очей» головного героя [11].

Розробка комп'ютерної гри, зокрема жанру "шутер від першої особи" (англ. *First-Person Shooter*, FPS), — це складний багатостадійний процес, який поєднує в собі творчі, технічні та управлінські аспекти. Хоча життєвий цикл розробки може незначно варіюватися залежно від специфіки проєкту чи студії, зазвичай він складається з наступних основних етапів: пре-продакшн, продакшн, пост-продакшн та підтримка після релізу.

1. Пре-продакшн (початкова стадія). На цьому етапі формується загальна концепція гри, визначаються її цілі, жанрова специфіка, технічні можливості та ринкова стратегія.

Для FPS-ігор це включає:

- ***Розробка концепту***: формулювання основної ідеї, сеттингу (наприклад, сучасна війна, наукова фантастика, постапокаліпсис), визначення унікальних особливостей гри.
- ***Створення геймдизайн-документа*** **(GDD)**: детальний документ, що описує механіки стрільби, інтерфейс, поведінку ворогів, типи зброї, рівні складності, мультиплеєр (за наявності) тощо.
- ***Прототипування***: створення базового ігрового прототипу, який дозволяє оцінити ідеї в дії (наприклад, базова стрільба, рух персонажа від першої особи).
- ***Підбір технологій***: вибір рушія (найчастіше використовуються Unreal Engine або Unity), інструментів для моделювання, анімації, звуку тощо.
- ***Формування команди***: геймдизайнери, програмісти, 3D-художники, аніматори, звукорежисери, тестувальники.

2. Продакшн (етап основної розробки). Цей етап є найдовшим та найінтенсивнішим. Усі складові гри поступово втілюються у функціональний продукт:

- ***Програмування ігрових механік***: створення логіки стрільби, перезарядки, прицілювання, ворожого ІІ, системи укриттів, пошкоджень, переміщення гравця тощо.
- ***Дизайн рівнів*** **(левел-дизайн)**: розробка карт, розташування об'єктів, точок появи ворогів, укриттів, зони бойових дій. У FPS-іграх рівень повинен стимулювати рух і бойові дії.
- ***Графіка та анімація***: моделювання зброї, рук персонажа, ворогів, оточення, а також анімація стрільби, ходьби, перезарядки, вибухів тощо.

- ***Звуковий супровід***: запис звуків пострілів, кроків, голосів ворогів, музики, атмосферних ефектів. У шутері звук часто є критично важливим для орієнтації гравця.
- ***Тестування***: регулярне виявлення помилок (багів), перевірка балансу гри, стабільності, рівня складності. У FPS-жанрі особливу увагу приділяють точності механіки стрільби та фізиці.

3. Пост-продакшн (завершення розробки). Після завершення основної роботи починається підготовка до релізу:

- ***Оптимізація***: покращення продуктивності, зменшення часу завантаження, оптимізація ресурсів.
- ***Полірування***: усунення дрібних помилок, покращення анімацій, графіки, UI/UX.
- ***Маркетинг та реліз***: створення трейлерів, демо-версій, сторінки гри в онлайн-магазинах (Steam, Epic Games Store тощо), організація рекламної кампанії.

4. Післярелізна підтримка. Після випуску гра не перестає розвиватися. Залежно від моделі розповсюдження та реакції спільноти:

- ***Патчі*** **та оновлення**: виправлення помилок, оновлення II, нові режими гри.
- **DLC** ***та новий контент***: додаткові місії, нова зброя, карти, скіни.
- ***Онлайн-підтримка*** **(якщо є мультиплеєр)**: балансування зброї, запобігання чітам, підтримка серверів.
- ***Збір зворотного зв'язку***: взаємодія з гравцями через форуми, соціальні мережі, оновлення згідно з відгуками.

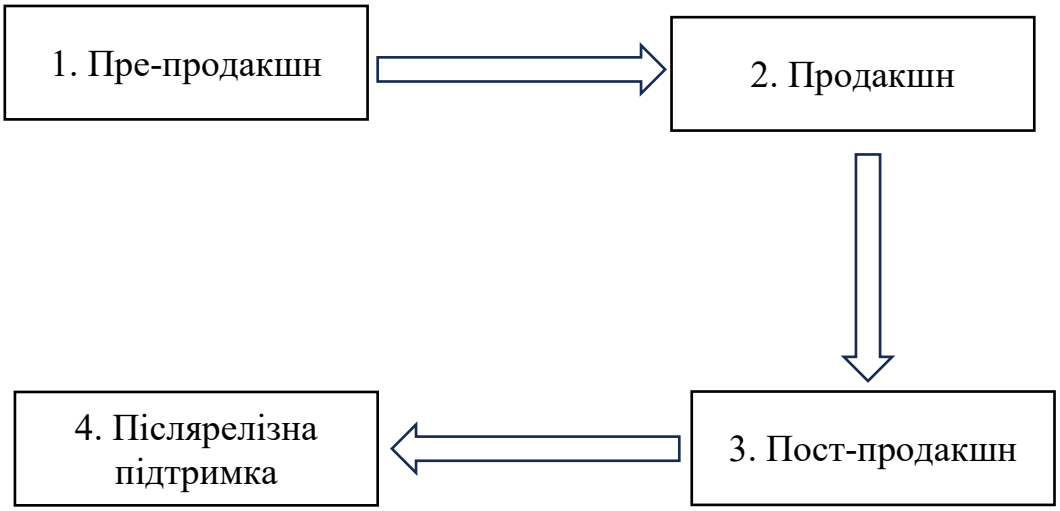


Рис. 1.1. Життєвий цикл розробки комп'ютерної гри для жанру «Шутер від першої особи»

Таким чином життєвий цикл розробки гри у жанрі "шутер від першої особи" — це складна, поетапна робота, що вимагає синергії технічних знань, творчого підходу та глибокого розуміння очікувань гравця. Успішний FPS-проєкт поєднує в собі якісний геймплей, збалансовану стрільбу, привабливий візуальний стиль і звукову атмосферу, що занурює гравця у світ гри.

1.4. Аналіз технологічного стеку

Технологічний стек комп'ютерної гри — це сукупність програмних і апаратних технологій, інструментів, бібліотек, фреймворків і систем, які використовуються для розробки, реалізації та запуску гри. Для жанру «Шутер від першої особи» (FPS, First-Person Shooter) технологічний стек має специфічні компоненти, оскільки цей жанр вимагає швидкого рендерингу, точної фізики, інтерактивності в реальному часі, сильного штучного інтелекту для ворогів і, часто, підтримки мультиплеєра. Перелік ключових елементів технологічного стеку для FPS-ігор наведено в табл. 1.1.

Таблиця 1.1

Перелік ключових елементів технологічного стеку для FPS-ігор

№ з/п	Компонент	Опис	Критерії оцінки
1	Ігровий рушій	Платформа для рендерингу, фізики, анімації, логіки та управління грою	Продуктивність, підтримка FPS-механік, документація, масштабованість, ціна
2	Фізична система	Моделює рух, колізії, балістику, руйнування об’єктів	Реалізм, швидкість обчислень, інтеграція з рушієм, підтримка руйнувань
3	Мережевий код	Забезпечує мультиплеєр, синхронізацію, онлайн-матчі	Затримка (лаг), стабільність, безпека, масштабованість серверів
4	Аудіо система	Обробка звуку, 3D-аудіо, ефекти, музика для занурення	Якість звуку, 3D-позиціонування, інтеграція, гнучкість
5	Графічна система	Забезпечує рендеринг 3D-моделей, текстур, освітлення, ефектів	Якість графіки, оптимізація, сумісність, підтримка реального часу
6	Штучний інтелект	Керує поведінкою ворогів, NPC, адаптацією до дій гравця	Розумність ворогів, гнучкість, швидкість реакції, масштабованість
7	Інтерфейс користувача	Елементи HUD (приціл, здоров’я, боєприпаси), меню, взаємодія з гравцем	Зручність, адаптивність, швидкість відгуку, дизайн
8	Інструменти моделювання	Створення 3D-моделей зброї, персонажів, локацій	Зручність, якість експорту, сумісність із рушієм, деталізація
9	Система оптимізації	Забезпечує продуктивність на різних платформах, зменшує навантаження	FPS, сумісність із низькими конфігураціями, стабільність, розмір гри
10	Тестування та дебаг	Виявлення помилок, балансування геймплею, перевірка стабільності	Точність виявлення багів, автоматизація, покриття тестами, зручність

1. Ігровий рушій (англ. *game engine*) — це програмна платформа, яка забезпечує базову функціональність для створення комп’ютерних ігор. Він об'єднує низку систем і модулів, таких як графічний рендеринг, фізичний рушій, анімація, аудіо, штучний інтелект, система скриптів, керування ресурсами та інтерфейс користувача. Вибір рушія має вирішальне значення для технічної реалізації гри, її продуктивності, гнучкості та майбутньої масштабованості.

Основні функції ігрового рушія:

- ***Візуалізація*** **(рендеринг)** — генерація 2D- або 3D-зображень у реальному часі.
- ***Фізика*** — обчислення взаємодії між об'єктами, гравітації, зіткнень, руху.
- ***Система анімації*** — відтворення рухів персонажів, об'єктів, інтерфейсу.
- ***Штучний інтелект*** — логіка поведінки NPC (неігрових персонажів).
- ***Аудіо*** — відтворення звукових ефектів, музики, голосу.
- ***Скриптинг*** — програмування логіки гри за допомогою мов сценаріїв (наприклад, Blueprint або Lua).
- ***Мережева підтримка*** — реалізація мультиплеєрних можливостей (за потреби).
- ***Інструменти розробки*** — редактори рівнів, UI-будівники, профайлери тощо.

Для жанру “шутер від першої особи” ігровий рушій має відповідати певним технічним вимогам: висока продуктивність при рендерингу 3D-графіки; підтримка реалістичної фізики та баллистики; гнучка система створення анімацій і переходів між ними; можливість створення складного ІІ; інструменти швидкої розробки прототипів та тестування.

Найпопулярнішими рушіями для FPS-ігор є:

Unreal Engine — один із найпотужніших рушіїв із підтримкою високоякісної графіки, системи Blueprint (візуального скриптингу), просунутого фізичного моделювання та готовими шаблонами для FPS.

Unity — гнучкий рушій із широкими можливостями кросплатформної розробки. Має підтримку C#, великий маркет ассетів, але дещо складніший у налаштуванні реалістичної стрільби.

CryEngine — спеціалізований на фотореалістичній графіці, історично асоціюється з FPS (наприклад, Crysis), але вимагає високих технічних навичок.

Source Engine **(**від **Valve)** — класичний рушій для FPS, використаний у Half-Life 2, CS:GO та інших. Потужний, проте застарілий для сучасних вимог без модифікацій.

У межах кваліфікаційної роботи як базовий рушій для реалізації FPS-ігрового проєкту обрано **X-Ray Engine** — рушій, розроблений українською студією GSC Game World для серії ігор *S.T.A.L.K.E.R.*

Переваги X-Ray Engine наведено в табл. 1.2.

Таблиця 1.2

Переваги рушія X-Ray Engine

Компонент	**Опис**
Відкрита архітектура	рушій має відкритий код (через форки типу OpenXRay), що дозволяє глибоку модифікацію та налаштування
Вбудована підтримка FPS-механік	початково створювався саме для ігор від першої особи з елементами симуляції
Розвинена система ШІ	включає поведінкові дерева, планування маршрутів, реакції на звук та зір
Збалансована графіка	хороша оптимізація для відкритих просторів, динамічного освітлення та погодних ефектів
Модульність	підтримує кастомні скрипти, нові локації, квести, інтерфейси
Спільнота	існує активна база користувачів і модифікаторів, що дозволяє швидше вирішувати технічні проблеми

Використання **X-Ray Engine**, зокрема його модернізованої версії **OpenXRay**, дає змогу не лише зберегти атмосферу класичних FPS-проєктів, а й

вбудувати нову ігрову логіку, систему діалогів, анімації та інші елементи, що відповідають сучасним потребам жанру.

2. Фізика та анімація — це ключові елементи, що забезпечують реалістичність та динамічність взаємодії гравця з ігровим середовищем. У жанрі "шутер від першої особи" (FPS) ці компоненти відіграють особливо важливу роль, оскільки безпосередньо впливають на бойову систему, рух персонажа, ефекти зіткнень, поведінку об’єктів та загальне сприйняття ігрового процесу.

Фізичний модуль у грі відповідає за: *обробку зіткнень* (щоб кулі, предмети або персонажі взаємодіяли з оточенням правдоподібно); *гравітацію та рух* (моделювання падінь, стрибків, інерції при пересуванні); *балістику* (розрахунок траєкторій куль, кутів відскоку, пробивної здатності); *розрухованість об'єктів* (наприклад, щоб коробки чи меблі реагували на вибухи); *фізику тіл NPC* (щоб вороги реалістично падали після пострілу). Більшість сучасних рушіїв використовують вбудовані або зовнішні фізичні бібліотеки, такі як Havok, Bullet чи PhysX. У старіших або спеціалізованих рушіях, як-от X-Ray Engine, використовується власна фізична система, адаптована під конкретні потреби гри.

Анімація в іграх забезпечує візуальне представлення дій персонажів, об’єктів і навколишнього середовища. Основні компоненти системи анімації: *скелетна анімація* (використовується для персонажів і монстрів, де рухи задаються через кісткову структуру); *морфінг/блендинг* (плавні переходи між анімаціями. Наприклад, ходьба → біг → стрибок); *інверсна кінематика (IK)* (для більш реалістичної взаємодії кінцівок із середовищем. Наприклад, коли персонаж ставить ногу на сходинку); *анімації об'єктів* (двері, ящики, зброя та інші предмети можуть мати власні цикли руху); *контекстні анімації* (дії, що змінюються в залежності від ситуації (перезарядка, взаємодія з предметами, реакція на поранення)).

Сучасні рушії підтримують як ручне, так і процедурне створення анімацій, а також використання motion capture для захоплення реального руху.

У FPS-іграх це особливо важливо для реалістичного відображення стрільби, переміщення рук гравця, або смерті ворогів.

Для реалізації фізики та анімації у кваліфікаційній роботі застосовується інструментарій, наданий рушієм **X-Ray Engine**, який включає:

- базову симуляцію фізики з підтримкою колізій, гравітації та реакцій об’єктів;
- систему скелетної анімації з можливістю комбінування кількох рухів;
- скриптову інтеграцію анімаційних подій через Lua.

Це дозволяє ефективно реалізовувати ігрову логіку, не вдаючись до складних зовнішніх бібліотек.

3. Мережевий код. Мережева підсистема є основою для багатокористувацького режиму комп’ютерної гри. Вона відповідає за обмін даними між клієнтами та сервером, синхронізацію ігрових об'єктів, обробку дій гравців у реальному часі та захист від мережевих загроз. Для шутера від першої особи (FPS) критично важливими є стабільна передача даних з мінімальними затримками та коректна обробка одночасних дій кількох користувачів.

Основні компоненти мережевого коду наведено в табл. 1.3.

Таблиця 1.3

Компоненти мережевого коду

Компонент	**Опис**
Архітектура «клієнт-сервер»	Сервер зберігає основну логіку гри, авторитетний стан об'єктів і здійснює перевірку дій гравців. Клієнти надсилають команди (рух, стрільба тощо) та отримують оновлення стану гри
Синхронізація	Забезпечує однаковий вигляд гри для всіх гравців: положення, анімації, події. Застосовується інтерполяція та предикція, щоб згладити затримки (lag compensation)
Стиснення та оптимізація трафіку	Щоб уникнути перевантаження мережі, дані передаються у стислому вигляді, використовуючи дельта-кодування (передача тільки змін)
Обробка відставання та втрачених пакетів	Використовуються алгоритми відновлення послідовності, повторної передачі або компенсації втрат

Безпека	Впроваджується перевірка достовірності клієнтських даних (anti-cheat логіка), шифрування та контроль доступу

У межах розробки гри на базі **X-Ray Engine**, мережевий код є частиною вбудованої інфраструктури рушія. Рушій підтримує базову клієнт-серверну модель, яка застосовувалася в багатокористувацькому режимі гри *S.T.A.L.K.E.R.: Shadow of Chernobyl* та її наступниках.

Можливості X-Ray Engine у сфері мережевої гри:

- Синхронізація руху, анімацій, подій та стану об'єктів між клієнтами.
- Робота з UDP-пакетами для забезпечення швидкої реакції.
- Підтримка LAN- та інтернет-гри.
- Простий у впровадженні механізм розгортання локального сервера.

Проте, варто зазначити, що X-Ray Engine має обмежену підтримку масштабованих онлайн-режимів. У разі потреби розширення функціональності можлива інтеграція сторонніх рішень або перепис мережевого модуля із сучаснішими підходами.

4. Аудіо. Аудіосистема є одним із ключових елементів ігрового досвіду, особливо в шутерах від першої особи. Звуковий супровід формує атмосферу, допомагає орієнтуватися у просторі, сигналізує про небезпеку або події навколо, а також підсилює емоційний вплив на гравця.

Основні компоненти аудіосистеми наведено в табл. 1.4.

Таблиця 1.4

Компоненти аудіосистеми

Компонент	**Опис**
Фонове озвучення (амбінт)	Звуки навколишнього середовища: вітер, дощ, тріскотіння проводів, ехо в тунелях тощо. Створює атмосферу та занурення у світ гри
Звуки дій гравця	Кроки, стрибки, перезарядка зброї, удари. Важливі для зворотного зв’язку та тактичної орієнтації
Озвучення противників та NPC	Репліки, крики, звуки наближення чи пересування. Дає змогу гравцю оцінити ситуацію без візуального контакту
Звуки зброї та вибухів	Мають бути чіткими, реалістичними, з урахуванням різних типів зброї. Важливі для створення відчуття

	сили та впливу
Динамічне аудіо	Зміна гучності, панорамування, ефекти затухання залежно від відстані до джерела. Створюється ілюзія простору та напряму звуку (3D-звук)
Музичний супровід	Саундтрек, що супроводжує ігровий процес, змінюється залежно від дій: бойові моменти, розслаблені ділянки, сюжетні сцени. Допомагає керувати емоційною динамікою гри

У межах дослідження, що розробляється на основі **X-Ray Engine**, реалізація звукової системи базується на вбудованому модулі, який забезпечує:

- *Підтримку багатоканального звуку* (включаючи 3D-позиціювання).
- *Динамічну зміну гучності* та напрямку звуку залежно від розташування гравця.
- *Підтримку просторового загасання* (об'єкти, що віддаляються, звучать тихіше та менш чітко).
- *Можливість прив'язки звуку до об'єкта* (наприклад, шум двигуна техніки, постріли зі зброї, голоси NPC).

Звукові файли у форматах, сумісних із рушієм (наприклад, .ogg), легко імпортуються в проєкт та можуть керуватися як через конфігураційні файли, так і скриптами (Lua).

Таким чином, аудіосистема в FPS-проєкті виконує не лише роль атмосферного супроводу, а й стає інструментом геймплейної взаємодії, що покращує занурення гравця у віртуальне середовище.

1.5. Базові метрики ігор

Аналіз метрик — важлива складова процесу розробки комп'ютерних ігор. Вони дозволяють розробникам оцінити, наскільки ефективно реалізовано ігрові механіки, наскільки задоволені гравці, а також виявити «вузькі місця» в дизайні або балансі. Метрики дають змогу обґрунтовано приймати рішення щодо подальшої розробки, оптимізації, оновлень та монетизації гри.

Метрики дають змогу обґрунтовано приймати рішення щодо подальшої розробки, оптимізації, оновлень та монетизації гри, забезпечуючи комплексний аналіз як технічних аспектів, так і користувацького досвіду. Вони дозволяють ідентифікувати проблемні зони, відстежувати ефективність нововведень і прогнозувати поведінку гравців, що сприяє підвищенню якості продукту та задоволеності аудиторії.

Метрики в контексті ігрової індустрії — це кількісні показники, які фіксують різні аспекти взаємодії гравця з грою: технічні параметри, ігрову поведінку, прогрес, задоволеність, економіку тощо. Ці дані можуть збиратися автоматично або вручну (через тести й опитування), і використовуються як у внутрішньому продакшн-циклі, так і після релізу продукту.

Метрики умовно поділяють на ігрові, поведінкові, бізнес-метрики та технічні (табл. 1.5).

Таблиця 1.5

Метрики відеоігор

Категорія метрик	**Метрика**	**Опис**
Ігрові метрики	Час сесії	скільки часу гравець проводить за одну гру
	Частота смертей/поразок	показує складність та баланс гри
	Прохідність рівнів	яка частка гравців завершує певні рівні
	Використання зброї/умінь	які інструменти гравець використовує найчастіше
	Кількість повторних проходжень	свідчить про реіграбельність гри
	Середня тривалість бою/сутички	важливо для FPS-жанру
Поведінкові метрики	Теплові карти (heatmaps)	візуалізація найчастіше відвідуваних зон карти
	Маршрути руху	які шляхи обирають гравці у відкритому рівні
	Частота використання інтерфейсу (UI)	які кнопки натискають, які ігнорують

	Повторне повернення до гри (retention)	відсоток гравців, які повертаються на 1-й, 3-й, 7-й день
	Рівень агресії/пасивності гравця	чи гравець активно шукає бої чи уникає їх
Бізнес-метрики	DAU (Daily Active Users)	кількість активних користувачів на день
	MAU (Monthly Active Users)	активні користувачі протягом місяця
	ARPU (Average Revenue Per User)	середній прибуток з одного користувача
	LTV (Lifetime Value)	загальний дохід, який приносить один користувач за весь період гри
	Conversion rate	відсоток гравців, які здійснюють покупки (у разі наявності внутрішніх транзакцій)
	Churn rate	відтік користувачів
Технічні метрики	FPS (Frames Per Second)	частота кадрів, показує плавність гри
	Ping / Latency	затримка з'єднання в мережевих іграх
	Час завантаження рівнів	швидкість роботи гри
	Кількість крашів / багів	стабільність програмного коду
	Використання пам'яті та ресурсів	оптимальність роботи з RAM, CPU, GPU

Ігрові метрики. Ці показники стосуються безпосередньо геймплею, механік, балансу та прогресії. Їхнє значення — зрозуміти, як гравці проходять гру, де вони застрягають, що їх приваблює або відштовхує. Ці метрики допомагають віднайти дисбаланс, "мертві зони" рівнів або механіки, які ігноруються.

Поведінкові метрики. Ці дані відображають, *як саме гравці взаємодіють з грою*, які маршрути обирають, як ухвалюють рішення та як реагують на зміни в геймдизайні.

Бізнес-метрики. Ці показники особливо важливі для комерційних проєктів, адже дозволяють оцінити успіх гри як продукту. Навіть у

некомерційних проєктах частина бізнес-метрик може використовуватися для оцінки популярності, впізнаваності, зацікавленості.

Технічні метрики. Ці дані важливі для *оптимізації продуктивності*, підтримки стабільності гри та задоволення гравця з технічного боку. У FPS-іграх критично важливо забезпечити високу частоту кадрів і низький ping, бо ці фактори напряму впливають на якість ігрового процесу.

Інструменти збору метрик. Для збору метрик використовуються як *вбудовані інструменти рушія* (наприклад, у X-Ray — лог-системи, статистика гравців), так і *сторонні сервіси*:

- Google Analytics for Games
- Unity Analytics / Unreal Insights (аналогічні рішення)
- PlayFab, GameAnalytics, Amplitude
- Heatmap API та телеметрія
- Лог-файли та дебаг-інтерфейси

У шутері від першої особи метрики можуть бути використані наступним чином:

- якщо аналітика показує, що більшість гравців залишають гру на 3-му рівні — це сигнал про надмірну складність або поганий дизайн карти.
- часте використання лише 1-2 видів зброї може свідчити про дисбаланс арсеналу.
- низький LTV і високий churn rate — показники того, що гравці не залишаються в грі надовго, і варто працювати над залученням.

Таким чином, базові метрики є не лише засобом вимірювання ефективності гри, але й *потужним інструментом для постійного поліпшення*. Їх аналіз дозволяє глибше зрозуміти потреби гравців, виявити сильні та слабкі сторони продукту, покращити баланс, зручність, технічну стабільність та прибутковість гри. Для сучасної гейм-індустрії метрики є невіддільною частиною життєвого циклу гри — від прототипу до релізу та подальшої підтримки.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СЮЖЕТНОГО ДОПОВНЕННЯ ДО ГРИ S.T.A.L.K.E.R. Call of Pripyat

2.1. Жанрові, технічні та наративні особливості ігор типу «Шутер від першої особи»

Шутер від першої особи (FPS, від англ. First-Person Shooter) — один з найвідоміших і найпопулярніших жанрів комп'ютерних ігор, який передбачає, що гравець спостерігає і взаємодіє з ігровим світом від імені персонажа, тобто з точки зору його очей. Це створює ефект глибокого занурення (англ. immersion), що робить жанр особливо придатним для створення напружених, драматичних або динамічних ігрових сценаріїв. За даними аналітичних звітів, жанр FPS займає одне з провідних місць за кількістю релізів та обсягом продажів [1, 2].

Класичною основою FPS-ігор є бойовий геймплей, орієнтований на точність, швидкість реакції, управління ресурсами (боєприпаси, здоров'я, укриття). Проте починаючи з другої половини 2000-х років відбулося активне жанрове розширення FPS, що включає в себе елементи survival-horror, RPG, stealth, а також наративно-орієнтовані проєкти. Прикладом такого зсуву є серії Half-Life, Metro, S.T.A.L.K.E.R., де бойові механіки часто поєднуються з розповіддю, дослідженням, психологічними елементами та впливом середовища на гравця.

Ключові риси FPS:

- занурення гравця через камеру від першої особи;
- домінуюча бойова механіка (стрільба, укриття, ближній бій);
- чітко структуровані рівні або відкритий світ;
- швидкий темп ігрових подій або поєднання з повільними наративними вставками;
- інтерфейс, що фіксує увагу на центральній частині екрана;
- використання ресурсів: боєприпаси, медичні засоби, енергія тощо;
- гнучка система збереження прогресу (автосейви, чекпойнти).

У сучасній геймдев-теорії FPS класифікується як один з базових жанрів, який найкраще реалізує механіки швидкого зворотного зв’язку, психологічного напруження та інтенсивної взаємодії гравця зі світом.

Темп геймплею у FPS може варіюватися від ультрашвидкого (DOOM, Quake, Apex Legends) до повільного і побудованого на нарративі (Amnesia, Metro, Firewatch). Темп задається через кількість подій, частоту зіткнень, складність проходження, наявність діалогів, потребу в дослідженні тощо. У класичному S.T.A.L.K.E.R. темп змінюється динамічно — дослідження світу чергується з інтенсивними перестрілками або сюжетними катсценами.

У модифікації *Catalyst: Complementation* темп свідомо знижено. Відсутність стрілецької зброї, ворогів, мутантів, вибухів чи хаотичних сцен створює умови для психологічного трилера. Гравець має повільно, крок за кроком занурюватися у ситуацію, вивчати місце, слухати персонажів, обдумувати репліки. Такий темп дозволяє створити глибше емоційне залучення, нагадуючи ігри-інтерактивні драми на кшталт The Vanishing of Ethan Carter, What Remains of Edith Finch або SOMA.

Типовий FPS-рівень має такі елементи: стартову зону, вузли конфлікту, місця відпочинку/перезарядки, точку збереження, кінцеву мету. У сюжетних FPS ці вузли інтегруються зі сценарієм. Рівні можуть бути тунельними (лінійними), відкритими (sandbox), або умовно відкритими, як у S.T.A.L.K.E.R чи Far Cry 2.

У *Catalyst: Complementation* застосовано одну локацію — «Кордон» із білду 1842, переосмислену як простір дослідження і діалогу. Локація поділена на зони тригерів, кожна з яких активує нову сюжетну подію або розмову з персонажем. Навігація не керується маркерами — гравець отримує інформацію через ситуації, діалоги, просторові підказки. Це дозволяє уникати ефекту «точки на мапі», і замість цього стимулює осмислене пересування.

Умовна структура наведена в табл 2.1.

Таблиця 2.1

Зона	Дія	Умова переходу
Початкова база	Знайомство з NPC	Після першої катсцени
Стара дорога	Внутрішній монолог	Перебування в зоні більше 10 сек
Точка біля мосту	Діалог	Активна інфопорція + поблизу гравець

FPS-ігри використовують різні методи подачі наративу:

- текстові документи (Resident Evil);
- внутрішньоігрові катсцени (Half-Life 2);
- діалоги з NPC (Metro, Fallout);
- флешбеки (SOMA);
- дослідження середовища (The Last of Us);
- поведінка персонажа (розмови вголос, внутрішній голос).

У *Catalyst: Complementation* реалізовано наративний підхід через:

- діалоги з NPC, які активуються за логічними умовами (meet, info);
- скриптові переходи між подіями (on_info, condlist);
- символізм: відсутність музики, приглушені кольори, фрагментарна інформація;
- структурування ліній, в яких NPC поступово втрачають глузд.

Це наближує мод до «інтерактивного кіно» (interactive fiction), де вибір гравця не завжди веде до конкретного результату, але формує досвід. Важливу роль відіграє також атмосфера, створена за допомогою погодного редактора, світлових акцентів, і звукової порожнечі. Ігрова механіка FPS зазвичай орієнтується на комбатну частину, але у випадку з нашою модифікацією бойова частина взагалі відсутня. Вся взаємодія будується навколо логіки скриптових подій, інформаційних прапорців, переміщення по зоні. Навіть ніж — єдина «зброя» — виконує декоративну функцію, символізуючи беззахисність або втрату сили.

Типи взаємодії:

- активація діалогів — при підході до NPC з певною інфопорцією;
- запуск подій — після виконання умов (наприклад, +dialog_end → condlist_0);
- сценарна логіка NPC — їхня поведінка змінюється залежно від лінії гравця;
- ілюзія вибору — кілька варіантів у діалозі, які ведуть до психологічно різних наслідків, але не змінюють сюжет прямо.

Цей підхід можна співвіднести з іграми Telltale (наприклад, The Walking Dead), де діалоги визначають темп гри і глибину персонажів, а не конкретні геймплейні результати.

Таким чином, жанр FPS може адаптуватися під інші форми ігрового досвіду, відходячи від стрілецької механіки на користь занурення, драми, діалогу і атмосфери. Модифікація *Catalyst: Complementation* — приклад саме такого підходу, що базується на рушії FPS-гри, але трансформує його у психодраму з інтерактивними вузлами.

2.2. Огляд ігрового рушія X-Ray Engine та його модифікованої версії OpenXRay

Ігровий рушій X-Ray Engine є фундаментом серії ігор S.T.A.L.K.E.R., розроблених українською студією GSC Game World. Він поєднує графічний рендерер, фізичний движок, логіку ІІ, систему діалогів, скриптову інфраструктуру та підтримку відкритого світу в єдину цілісну технологічну екосистему. З 2007 року рушій зазнав багатьох змін і став базою для сотень модифікацій. У рамках проєкту *Catalyst: Complementation* було використано готову збірку модифікованого рушія **OpenXRay**, що забезпечує сучасні технічні можливості для стабільної розробки.

X-Ray — ігровий рушій, створений українськими розробниками відеоігор GSC Game World для ігор S.T.A.L.K.E.R.: Тінь Чорнобиля, S.T.A.L.K.E.R.: Чисте небо та S.T.A.L.K.E.R.: Поклик Прип'яті.

Над графічною частиною рушія в основному працювали програмісти Олесь Шишковцов і Олександр Максимчук (вони ж працювали у 4A Games над проектом Metro 2033 по однойменній книзі Дмитра Глуховського)[12].

Базовий каталізатор роботи рушія — це два ключові блоки: папка **`bin`** (двійкові файли рушія, бібліотеки, виконувані модулі) та **`gamedata`** (ігрові конфігурації, сценарії, ресурси).

Структура папки bin

Папка `bin` містить компіляції основних компонентів рушія:

- `xrEngine.exe` — головний виконавчий файл рушія, який ініціює завантаження рівня та виконує основний цикл.
- `xrGame.dll` — DLL-бібліотека, що містить усю логіку геймплею, включаючи управління актором, обробку діалогів, динаміку івентів, інфопорції.
- `xrRender_R1.dll`, `xrRender_R2.dll`, `xrRender_R3.dll` — модулі для рендерингу відповідно до DirectX 8, 9, 10/11.
- `xrPhysics.dll` — обробка фізики через Open Dynamics Engine (ODE).
- `xrSound.dll` — система обробки звуку.
- `xrInput.dll`, `xrParticles.dll` — допоміжні модулі для ефектів та контролю.

Модифікація *Catalyst: Complementation* не вимагає власної компіляції рушія. Для запуску було використано готову збірку OpenXRay — одна з переваг рушія, який доступний як precompiled binary.

Структура gamedata

Усі дані гри розташовані в папці `gamedata`, яка включає:

- `config\` — `.ltx` файли: конфігурації ігрових об'єктів, NPC, логіки, діалогів, погоди, системи лута, рестрикторів тощо.
- `scripts\` — `.script` файли, написані на Lua, які керують подіями, тригерами, геймплейними змінами.
- `dialogs\` — конфігурації розмов між персонажами.
- `spawns\` — стартові позиції NPC та акторів.
- `textures\` та `meshes\` — візуальні ресурси.
- `levels\` — збережені геометрії локацій.

В рамках модифікації були самостійно створювані нові `.ltx` та `.xml` файли для визначення поведінки персонажів, завдань, логіки діалогів, інфопорцій, а також структури smart-террейнів. Всі ці файли базувались на оригінальних шаблонах серії S.T.A.L.K.E.R., але були змінені або доповнені відповідно до нового сюжету.

Під час запуску `xrEngine.exe` здійснює наступне:

1. Ініціалізація драйверів рендеру (через вибір DirectX);
2. Завантаження `xrGame.dll` — обробка геймплейної логіки;
3. Завантаження картки рівня (із `levels`), обробка `spawn`
4. Прочитання конфігурацій (`gamedata\config`);
5. Підключення скриптів Lua (`gamedata\scripts`);
6. Активація діалогових та тригерних систем;
7. Основний цикл: update → render → event → control.

Особливість рушія — розподіл відповідальності між конфігураційними файлами та логікою в `xrGame.dll`. З одного боку, рушій надзвичайно гнучкий — майже все можна змінити без перекомпіляції. З іншого — рушій має спадщину з 2000-х, деякі частини застарілі.

Оригінальний рушій має низку недоліків:

- Прив’язка до першого ядра ЦП (Windows XP scheduler);
- 32-бітна архітектура (обмеження у використанні оперативної пам’яті до ~2 ГБ);

- Відсутність багатопоточності для фізики та логіки;
- Застарілий інтерфейс OpenAL для звуку;
- Неможливість використання сучасних форматів ресурсів без переробки.

У контексті *Catalyst: Complementation* це означає: ризик вильотів при складних скриптах, необхідність оптимізації smart-террейнів, обмеження на кількість NPC на мапі, спрощення діалогів.

OpenXRay вирішує частину проблем X-Ray 1.6:

- **64-бітна архітектура:** дозволяє рушію працювати без обмеження у 2 ГБ пам'яті.
- **LuaJIT:** використовується як заміна стандартного Lua, працює швидше, стабільніше.
- **xrWeatherEditor:** дозволяє редагувати погодні цикли (використовується ключ -weather).
- **Кросплатформеність:** рушій можна зібрати для Windows, Linux, macOS.
- **Оновлена багатоядерність:** логіка більше не залежить від першого ядра[13].

У *Catalyst: Complementation* рушій працює на 64-бітній OpenXRay-збірці з підтримкою DX9/10. Однак геймплейна реалізація не потребувала поглибленого редагування рушійного коду — весь сценарій реалізований через `gamedata`.

Модифікація використовує OpenXRay як платформу, але вся робота ведеться в межах `gamedata`. Автор:

- створює конфігурації `.ltx` (смарттеррейни, рестриктори, секції логіки);
- пише нові `.xml` (опис діалогів, опис завдань);
- використовує стандартну систему `meet`, `walker`, `on_info`, `condlist`;

- не втручається в компіляцію `xrGame.dll`, не використовує C++/Visual Studio;
- не редагує `engine.cfg` або `user.ltx` (лише мінімально для тестів).

Таким чином, рушій дозволяє реалізувати повноцінну сюжетну історію без жодного перекомпільованого коду — все через скрипти і текстові файли. Це підтверджує придатність OpenXRay для наративних модів, де пріоритетом є не стрілянина, а атмосфера, діалоги, логіка подій.

X-Ray Engine, попри свій вік, є надзвичайно гнучкою та надійною базою для реалізації сюжетно-орієнтованих доповнень. Завдяки OpenXRay, автор модифікації *Catalyst: Complementation* мав змогу реалізувати повноцінну ігрову логіку, інтерфейс, діалоги та рівневу структуру без поглиблення в рушійний код. Умовна відсутність зброї, NPC-битв та відкритий дизайн — навпаки, зіграли на користь рушію: він показав себе ефективним для повільного, глибоко сценарного, атмосферного ігрового досвіду.

2.3. Розробка сюжетної концепції доповнення

Сюжет є однією з ключових складових модифікації *Catalyst: Complementation*. На відміну від оригінального S.T.A.L.K.E.R.: Call of Pripyat, що робив ставку на дослідження світу, аномалії, боротьбу з мутантами та стрілецький геймплей, ця модифікація навмисно відмовляється від зброї, бойових дій та надприродних істот. У центрі — внутрішній психологічний конфлікт персонажів, абсурдність ізоляції, повторення та втрати контролю над власним розумом. Автор модифікації поставив собі за мету створити досвід, у якому сюжет буде не лише фоном, а самим сенсом гри.

Тематика доповнення — **замкнене існування, втрачена реальність, психічна деградація через повторення**. Ігровий простір — це локація Кордон, стилізована як «петля», в яку потрапляють персонажі, і яка не має виходу.

Гравець поступово дізнається, що головні герої вже не вперше переживають одні й ті самі події, але не здатні повністю цього усвідомити.

Важливо підкреслити, що тут відсутній традиційний конфлікт «герой vs антагоніст». Весь конфлікт — внутрішній, філософський, подекуди екзистенційний. Простір гри — це не світ, у якому щось відбувається, а ментальна площина, де герої намагаються осмислити, хто вони є і чи існує вихід.

Мета такого підходу — викликати у гравця відчуття тривожного повторення, тиску, приреченості. Це досягається не через зовнішні ефекти, а через поступове загострення нелогічності світу, неузгодженість поведінки персонажів, повторення одних і тих самих діалогів з варіативними інтонаціями або формулюваннями.

Гравець виступає у ролі дослідника не Зони, а **ментального простору**, що імітує Зону. Таким чином модифікація пропонує осмислення сталкерської міфології як внутрішньої проекції людської свідомості. Петля на Кордоні — не лише ігрова, а й метафізична.

Гравець керує **Дятлом** (справжнє ім’я — Влад), молодим чоловіком із важким минулим, родом із Рівного. Його біографія не подається прямо — гравець дізнається її з діалогів інших персонажів, уривків, натяків. Він умовно «німий», але ближче до фіналу петлі починає промовляти окремі репліки. Це створює ілюзію «пробудження» свідомості, яка спостерігала, але не брала участі.

Ім’я «Дятел» має багатозначний символізм. З одного боку — це глузливе прізвисько, яке натякає на неприйняття. З іншого — образ істоти, яка постійно стукає по дереву в намаганні дістатися глибше. Такий метафоричний образ ідеально відповідає головному герою, який повторно проживає однакові події у спробі дістатися істини.

Його інструмент — ніж — це не зброя. Це уламок минулого, символ намагання впливати, але вже неактуальний. Гравець не може ним завдати

шкоди, лише носити. Це візуальний маркер — він постійно перед очима, нагадує про намагання щось змінити, хоча всі зміни вже сталися до гравця.

У фінальних діалогах Дятел починає згадувати власне минуле. Чи дійсне воно — залишається відкритим. В одному з монологів Родрігеса є фраза: "Ти — той, хто збудував усе це. Ти — той, хто боїться прокинутись". Це натяк на те, що Дятел — не просто гравець, а свідомість, що розпалася, і намагається зібрати себе.

Ключовий елемент структури доповнення — п’ятеро NPC, кожен із яких репрезентує різні аспекти свідомості. Вони функціонують не лише як персонажі, а як архетипи. Окрім цього, кожен має власну біографію, яка формує його особисту драму. Перелік персонажів та їх характеристика наведена в табл 2.2.

Таблиця 2.2.

Ім’я	**Походження**	**Архетип**	**Характеристика**	**Арка персонажа**
Шахтар	Донецька область	Пам’ять	Мовчазний, обережний, пригнічений	Поступове згасання пам’яті та відмова від розмов
Родріге с	Ірпінь	Розум/Логік а	Вишуканий, істеричний, одержимий книгою	Втрата зв’язку з реальністю через невдачу в письмі
Хвіст	—	Воля/Контр оль	Авторитарний, холодний, прагне порядку	Руйнування авторитету та внутрішня розгубленість
Уріч	Нетішин	Параноя/Інт уїція	Підозрілий, гнівливий, недовірливий	Самоізоляція та ментальне саморуйнування
Дятел	Рівне	Свідомість/ Я	Мовчазний, уважний, не реагує відкрито	Пробудження та спроба вийти з петлі

Історії персонажів

Дятел — це сам гравець, але також — внутрішній центр історії. Через нього інші герої бачать себе. Він є тим, хто з’єднує їх і водночас є чужим. Візуал персонажа зображено на рис 2.1.

Рис 2.1. Головний герой доповнення – Дятел

Шахтар — людина з важким минулим, яка все життя пропрацювала на шахті. Він потрапив у Зону — ніби шукаючи відплату або прощення. Його промовисте мовчання — це наслідок внутрішньої травми. На рис 2.2. також зображено візуал персонажу.

Рис 2.1. Персонаж доповнення – Шахтар

Родрігес — колишній письменник, що намагався написати книгу про Зону. Його рукопис став нав'язливою ідеєю, і він втрапив у Кордон у пошуках "справжньої історії". Його фрази часто алогічні, він говорить про речі, які трапилися, але не могли трапитися. Візуал персонажа зображено на рис 2.3.

Рис 2.3. Персонаж доповнення – Родрігез

Хвіст — самопроголошений лідер групи. Він прагне впорядкувати світ навколо себе, створити "графік", роздати завдання. Але з кожним циклом його структура руйнується, і він все більше кричить, замість пояснювати. На рис 2.4. зображено візуал персонажа.

Рис 2.4. Персонаж доповнення – Хвіст

Уріч — одинак, який ізолювався від інших. Він переконаний, що решта — змовники. Часто говорить про “черв'яка”. Його монологи все частіше — набір випадкових слів. Візуал персонажа зображено на рис 2.5.

Рис 2.5. Персонаж доповнення – Уріч

Кожен NPC має унікальні діалоги, що змінюються в залежності від прапорів (`info`), прогресу в сюжеті, часу на локації. У першому циклі вони видають стандартні фрази, у другому — починають сумніватися, у третьому — бояться, у четвертому — впадають у безсилля.

Це створює ефект «живого світу», який змінюється без явних катастроф. Персонажі не кричать, не вимагають, не атакують — вони **занурюють**, як божевільні друзі в театрі абсурду.

Сюжет не має чіткої кульмінації або розв’язки. Він побудований за принципом **повторення з варіаціями**. Це дозволяє гравцеві самостійно визначати, коли «щось пішло не так». На першому колі — стандартна сталкерська історія. На другому — непомітні зміни. На третьому — гравець починає сумніватися. На четвертому — настає прийняття.

Ключові зміни між циклами:

- фрази NPC стають агресивнішими або безглуздішими;
- зникають деякі предмети ландшафту;

- інтерфейс (hud) поступово затемнюється;
- кольорова гама змінюється на сірішу;
- звук починає дезорієнтувати: луна, шепіт, шум, що наростає.

Кожен цикл має своє "завдання", хоч і не формалізоване. Наприклад:

1. Поговорити з усіма NPC.
2. Почекати на конкретній точці більше 30 секунд.
3. Спровокувати діалог, який був уже почутий.
4. Отримати інфопорцію, яку вже мав раніше.
5. Знайти "вихід" і виявити, що його немає.

Сюжет не веде до розв’язки, а до **розкладання**. Гравець не вирішує ситуацію, він проживає її до точки, де вона втрачає сенс. І лише тоді з’являється нова — порожня локація, яка символізує звільнення або смерть.

Атмосфера формується не лише через світло і звук, а через **повторення та мовчання**. Частина діалогів — це «мовчазні сцени», коли персонаж просто дивиться. Акт мовчання в рушії реалізовано як meet-сцена без meet_dialog, що триває визначений час.

Наратив не має лінії — він **фрактальний**. Гравець може почати з будь-якого NPC, і кожен шлях створює власний контекст. Немає правильного шляху. Усі вони ведуть у середину. Внаслідок цього формується нераціональна, але глибоко емоційна структура сприйняття.

Особлива увага приділяється деталям: положенню предметів, реакції персонажа на гравця, кольору неба. У циклі №3 небо стає темно-синім, а звук — дещо нижчий. Ці дрібниці грають на підсвідомості, створюючи ефект нав’язливості.

Сюжетна концепція *Catalyst: Complementation* — це експеримент на межі геймплею, драматургії та інтерактивної літератури. Вона спирається на можливості рушія X-Ray та його скриптову архітектуру, але виходить за межі звичного формату FPS-гри. Відсутність ворогів, стрілянини та цілей не є

браком — це осмислена позиція автора, яка дозволяє перенести акцент з виживання у зовнішньому світі на **виживання у внутрішньому просторі**.

Гравець не лише взаємодіє з грою — він піддається впливу. І саме це, у поєднанні з інтегрованим сценографічним підходом, робить *Catalyst: Complementation* унікальним наративним досвідом, що функціонує у межах, але й поза межами рушія S.T.A.L.K.E.R.

2.4. Реалізація внутрішньоігрової логіки, діалогів та квестової системи

Модифікація *Catalyst: Complementation* значною мірою спирається на внутрішньоігрову логіку, що керується файлами конфігурації, скриптами та структурою діалогів. Попри відсутність стрілецького елементу та бойових механік, гра містить складну систему взаємозв'язків між подіями, NPC та гравцем. У цьому розділі розглядаються особливості реалізації цієї логіки, включаючи діалогову систему, квестову архітектуру та загальну структуру сценарної реалізації.

Сценарна логіка в рушії X-Ray реалізується через такі компоненти:

- **Інфопорції (`info`)** — текстові ідентифікатори станів, які використовуються для активації діалогів, подій, зміни поведінки NPC;
- **Секції `logic@`** — основний блок для опису поведінки NPC, що вказує шляхи пересування, анімації, реакції;
- **Умовні блоки `condlist`** — логічні вирази у конфігураційних файлах, що перевіряють наявність певних інфопорцій;
- **Скрипти на Lua** — керують загальною логікою, але в рамках цієї модифікації використовувались обмежено.

Завдяки цим системам гравець, наближаючись до NPC або виконуючи певні умови (наприклад, пройшовши до певної точки), запускає діалоги, змінює поведінку персонажів або активує нові завдання.

Діалоги є головним засобом подачі сюжету у грі. Вони структуровані у `.xml` файлах, розміщених у директорії `gamedata\dialogs`. Діалоги реалізуються через секції `meet` та `meet@`, які ініціюються у логічних секціях NPC або через прямий контакт із гравцем.

Приклад схеми діалогів на прикладі одного діалогу:

```
<dialog id="hvost_dialog">
   <has_info>hvost_new_dialog</has_info>
   <dont_has_info>hvost_end</dont_has_info>
   <phrase_list>
      <phrase id="1">
         <text>hvost_dialog_1</text>
         <next>2</next>
      </phrase>
      <phrase id="2">
         <text>hvost_dialog_2</text>
         <next>3</next>
      </phrase>
      <phrase id="3">
         <text>hvost_dialog_3</text>
         <next>4</next>
      </phrase>
      <phrase id="5">
         <text>hvost_dialog_5</text>
         <next>6</next>
      </phrase>
      <phrase id="6">
         <text>hvost_dialog_6</text>
         <next>7</next>
```

```
        </phrase>
        <phrase id="7">
            <text>hvost_dialog_7</text>
            <next>8</next>
        </phrase>
        <phrase id="8">
            <text>hvost_dialog_8</text>
            <next>9</next>
        </phrase>
        <phrase id="9">
            <text>hvost_dialog_9</text>
            <next>10</next>
        </phrase>
        <phrase id="10">
            <text>hvost_dialog_10</text>
            <action>dialogs_escape_old.quest_skelet</action>
            <give_info>hvost_end</give_info>
        </phrase>
        <phrase id="4">
            <text>hvost_dialog_4</text>
            <next>5</next>
        </phrase>
        <phrase id="0">
            <text>hvost_dialog_0</text>
            <next>1</next>
        </phrase>
    </phrase_list>
</dialog>
```

При цьому `hvost_dialog` містить весь набір реплік, їхні відповіді, посилання на інфопорції (`give_info`, `has_info`, `dont_has_info`), і може змінювати логіку залежно від стану гри.

Інфопорції слугують як "флаги". Наприклад:

condlist_0 = {+hvost_end} complete

означає, що при наявності інфопорції `hvost_end` діалог завершується, а гравець отримує нову задачу або відкривається наступний діалоговий вузол.

У модифікації реалізовані кілька унікальних діалогових гілок, де один і той самий персонаж із кожною новою зустріччю змінює свої репліки. Це реалізовано через `condlist` на основі різних інфопорцій. Такий підхід дозволяє створити ілюзію глибини і багатогранності персонажа без скриптової складності.

Хоч у грі немає класичних квестів із позначенням на карті, система завдань все ж присутня — у вигляді інформативної структури, що базується на:

- **`task_manager.ltx`** — конфігураційний файл, де описуються основні параметри завдань: назви, опис, пріоритет, умови завершення;
- **Інфопорції** — що виступають умовами запуску і завершення квестів;
- **Діалогова активація** — більшість квестів видається через розмову.

Приклад квесту:

[quest_hvost_razgovor]
icon = ui_inGame2_Put_v_pripyat
prior = 112
title = talked_hvost_name
descr = talked_hvost_text
on_init = %+hvost_new_dialog%
condlist_0 = {+hvost_end} complete

Тут `on_init` запускає інфопорцію `hvost_new_dialog`, а `+hvost_end` визначає момент завершення завдання. Таку структуру використовує більшість внутрішніх сюжетних блоків. Також приклади структур квестів наведені в Додатку Б.

Квести в модифікації часто мають вигляд не чіткої задачі, а процесу дослідження — наприклад, пройти через серію діалогів, відвідати зону, де активується монолог, або спостерігати за NPC у визначеній точці. Цей підхід краще відповідає загальній ідеї гри.

Тут `on_init` запускає інфопорцію `hvost_new_dialog`, а `+hvost_end` визначає момент завершення завдання. Таку структуру використовує більшість внутрішніх сюжетних блоків.

Квести в модифікації часто мають вигляд не чіткої задачі, а процесу дослідження — наприклад, пройти через серію діалогів, відвідати зону, де активується монолог, або спостерігати за NPC у визначеній точці. Цей підхід краще відповідає загальній ідеї гри.

NPC мають індивідуальні секції логіки. Наприклад:

```
[logic@stalker_2_hvost]
suitable = {=check_npc_name(stalker_hvost)}
active = walker@6
prior = 200
```

У цій секції описано, що NPC активує `walker@6` при відповідності імені. Далі NPC рухається певним маршрутом, а при отриманні нової інфопорції — перемикається на іншу секцію.

```
[walker@6]
path_walk = hvost_1_walk
```

path_look = hvost_1_look
def_state_moving1 = run
on_info = {+hvost_nineth_dialog_end} walker@7

Такі секції дозволяють NPC не лише стояти і видавати репліки, а бути частиною сцени. Вони ходять, дивляться, зупиняються, реагують на дії гравця.

Ігрові події контролюються переважно через інфопорції та секції `on_info`, `on_signal`, а також обмежені скрипти на Lua (у `xr_effects.script`, `bind_stalker.script`).

Типовий приклад:

on_signal = dlg_3 | %+hvost_prishel_2%

Це означає, що при сигналі `dlg_3` активується інфопорція `hvost_prishel_2`, яка може впливати на завдання, діалоги, логіку пересування NPC або видимість об’єктів на мапі.

Тригерна система, хоч і проста, дозволяє гнучко налаштовувати сценарні сцени. Особливо це важливо в умовах модифікації, де сцени не мають бойового розвитку, і потрібно покладатися на інтелектуальну інтеракцію з гравцем.

Логіка, діалоги та квестова система модифікації *Catalyst: Complementation* реалізовані виключно засобами, передбаченими рушієм X-Ray Engine. Без потреби в розширенні функціоналу через модифікацію рушія, автор зміг створити глибоку й багатоетапну структуру подій. Всі механіки — від простих info до логіки meet — використані як інструменти створення напруженого психологічного сюжету, що поступово відкривається гравцеві. Реалізація логіки наведена в схемі на рис 2.6.

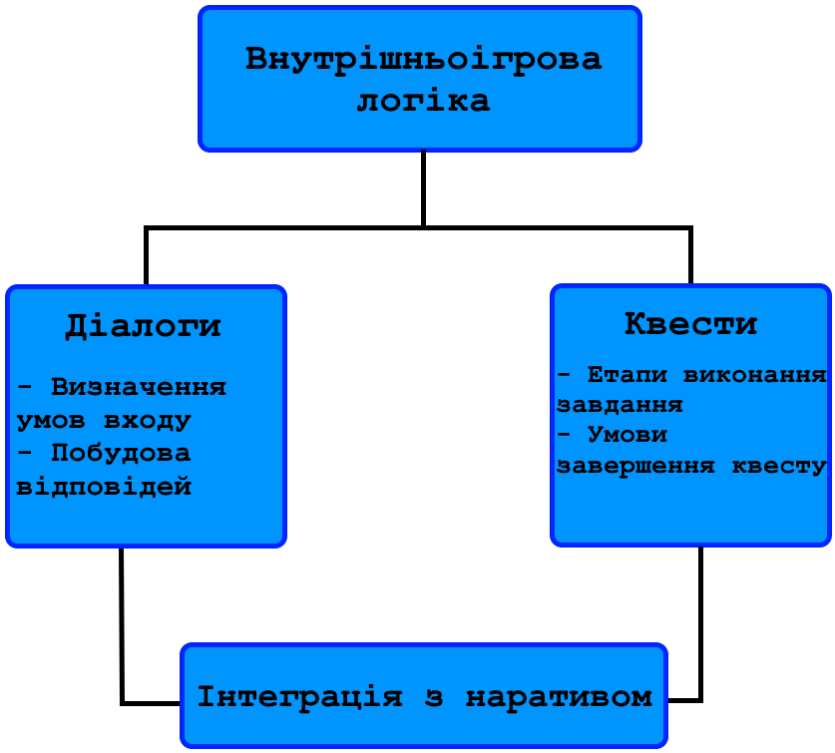


Рис 2.6. Реалізація внутрішньоігрової логіки, діалогів та квестової системи

Така реалізація підтверджує, що навіть базовий інструментарій рушія дозволяє досягти високої наративної складності, за умови ретельного сценарного проєктування та послідовної логіки в побудові світу.

РОЗДІЛ 3. РОЗРОБКА СЮЖЕТНОГО ДОПОВНЕННЯ ДО ГРИ S.T.A.L.K.E.R. Call of Pripyat

3.1. Розробка та адаптація інтерфейсу користувача

Інтерфейс користувача (UI) та загальна структура взаємодії гравця з грою (UX) є важливими складовими будь-якого ігрового досвіду. У модифікації *Catalyst: Complementation* інтерфейс виконує не лише утилітарну функцію, а й служить наративним засобом. Він спрямований на посилення відчуття занурення, підтримку атмосфери ізоляції та психологічного напруження, а також упорядкування діалогів і логіки завдань.

Розробка інтерфейсу у модифікації *Catalyst: Complementation* базувалась на трьох ключових принципах:

1. **Мінімалізм і лаконічність** — відмова від зайвих індикаторів, декоративних елементів, кольорової гами. Це дозволяє гравцеві повністю зосередитись на внутрішньому конфлікті гри та атмосфері, не відволікаючись на стандартні ігрові UI-компоненти.
2. **Інтеграція з наративом** — інтерфейс відображає внутрішній стан персонажа. Наприклад, відсутність карти чи вказівників є не лише механічним обмеженням, а і частиною світоглядної структури персонажа, який не знає, де знаходиться, і не може орієнтуватися.
3. **Поступове розкриття** — на початку гри більшість інтерфейсних елементів або приховані, або не активні. У міру просування гравець відкриває нові елементи лише тоді, коли це потрібно. Таким чином, відбувається побудова довіри між грою та гравцем.

HUD (Head-Up Display)

HUD є центральною точкою взаємодії гравця з ігровим середовищем. У *Catalyst: Complementation* HUD складається лише з мінімального набору:

- індикатор здоров’я;
- іконка активної інфопорції (позначка, що вказує на значущу подію);

- слот ножа (один-єдиний доступний предмет);
- індикатор втоми (у затемненому вигляді).

Всі інші елементи (зброя, броня, артефакти, лічильник патронів) були видалені або замінені умовними декоративними текстурами. Таким чином створюється відчуття того, що гравець — не боєць, а в’язень простору.

Інтерфейс HUD демонструється на рис 3.1.

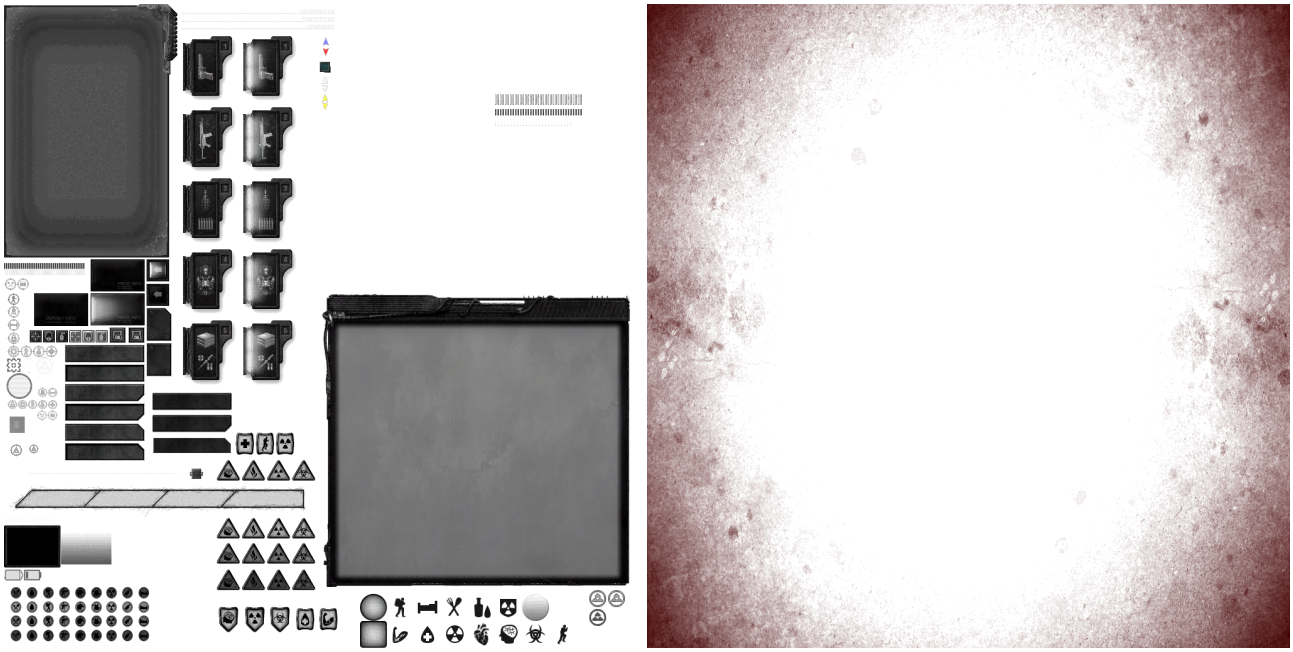


Рис 3.1. Текстура HUD-дісплею

Інвентар

Інтерфейс інвентарю побудований на базі **`ui_actor_menu.dds`**, але модифікований таким чином, що більшість функцій або відсутні, або заблоковані. Гравець має доступ до:

- слоту ножа;
- однієї кишені (умовно — для записок);
- перегляду завдань (через кнопку «J», однак без маркерів).

Інвентар відображається як щось застаріле, залишене з минулого — він тут не для управління, а для створення ілюзії контролю, якого насправді немає. Візуал інвентарю наведений на рис 3.2.

Рис 3.2. Текстура інвентарю

Діалоги

Система діалогів зазнала найбільших змін. Вона базується на файлі **`ui_actor_dialog_screen.dds`**. Оновлення включали:

- зміну шрифтів на менш агресивні;
- затемнення фону вікна діалогу;
- використання символів замість кольору для позначення варіантів відповіді;
- можливість деактивації курсору для максимального занурення.

Діалоги мають кілька рівнів глибини. Залежно від інфопорцій, що зібрав гравець, NPC можуть або реагувати емоційно, або уникати відповідей. Це створює емоційний зв’язок із інтерфейсом. Текстура інтерфейсу діалогу демонструється на рис 3.3.

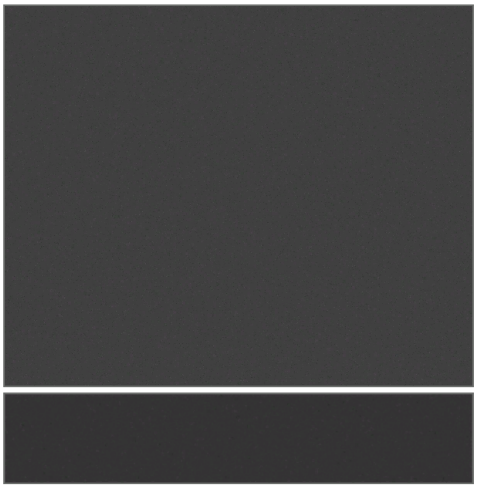


Рис 3.3. Текстура інтерфейсу діалогу

Інтерфейс не відображає жодної стандартної підказки, окрім тих, що потрібні для базової взаємодії. Наприклад, «Натисніть F для розмови» з’являється лише на першому NPC, у подальшому вона не дублюється. Таке рішення дає змогу посилити ефект ізоляції: гравець має запам’ятати правила гри та орієнтуватися інтуїтивно.

Курсор замінено на темно-сірий трикутник, що майже не помітний у динаміці. Це дозволяє сприймати екран як частину простору, а не як інтерфейс. Будь-яка візуальна допомога — лише тимчасова і функціональна.

UX (User Experience) був розроблений із фокусом на психологічне навантаження:

- **Контроль через втрату контролю** — чим більше гравець намагається керувати ситуацією, тим менше у нього виходить. Інтерфейс не відповідає на спроби взаємодії (натискання клавіш, прокрутка, відсутність реакції).
- **Напруження через тишу** — інтерфейс не сповіщає про зміни звуками, лише — візуально. Це створює ефект тиші, яка «тисне».
- **Інтерфейс як дзеркало** — у деяких сценах (особливо в циклі повторень), інтерфейс повністю зникає. Це підсилює ідею, що Дятел втрачає зв’язок із реальністю.

Розробка інтерфейсу для модифікації *Catalyst: Complementation* здійснювалася з акцентом на підтримку психологічного наративу та збереження цілісного візуального стилю. Модифіковані DDS-файли інтерфейсу створюють атмосферу порожнечі, приглушеності й пригніченості, яка ідеально відповідає загальному настрою гри. Завдяки мінімалістичному UX та лаконічному HUD гравець зосереджується на внутрішньому конфлікті героїв і нарративному наповненні, а не на механіці виживання. Інтерфейс стає не просто оболонкою гри — а її метафоричним відображенням.

3.2. Розширення ігрової карти та рівнів

У модифікації *Catalyst: Complementation* було реалізовано концепцію ізольованої локації, що функціонує як замкнена петля. Основою для неї стала одна-єдина локація — **Кордон** із збірки build 1842. Незважаючи на обмежений простір, дизайнерське рішення було скероване на максимальне використання цієї території через зміну сценарної логіки, інфопорцій, тригерів та послідовного редагування зон доступу.

Особливості використаної локації

Build 1842 — це одна з ранніх версій гри S.T.A.L.K.E.R., що містила альтернативні версії знайомих локацій. Кордон з цього білду відрізняється від фінального релізу:

- відсутністю великої кількості декору;
- мінімальною заселеністю;
- спрощеною структурою геометрії;
- технічною стабільністю для модифікацій без надмірного навантаження.

Це зробило її ідеальним варіантом для створення замкненого світу, де важливі не фізичні межі, а логічні й наративні.

Оскільки сюжет модифікації обертається навколо петлі, сама структура карти також піддається трансформації — не через фізичну зміну геометрії, а через **інформаційну маніпуляцію**.

- Деякі шляхи штучно заблоковані (рестриктори).
- Переходи між зонами контролюються скриптами, що імітують зміну часу доби або внутрішні стани героя.
- Повернення до початкової точки часто відбувається з неочікуваним зміненим станом об’єктів.

Таким чином карта виглядає цілісною, але щоразу трохи зміненою. Гравець фізично переміщується тими самими стежками, однак отримує щоразу інший емоційний досвід.

Оскільки сюжет модифікації обертається навколо петлі, сама структура карти також піддається трансформації — не через фізичну зміну геометрії, а через **інформаційну маніпуляцію**.

- Деякі шляхи штучно заблоковані (рестриктори).
- Переходи між зонами контролюються скриптами, що імітують зміну часу доби або внутрішні стани героя.
- Повернення до початкової точки часто відбувається з неочікуваним зміненим станом об’єктів.

Таким чином карта виглядає цілісною, але щоразу трохи зміненою. Гравець фізично переміщується тими самими стежками, однак отримує щоразу інший емоційний досвід. Ігровий вид локації Кордон зображений на рис 3.4.

Рис 3.4. Локація Кордон (build 1842)

Простір Кордону поділений на **психологічні зони впливу**. Це не геймплейні арени, а сегменти, в яких змінюється:

- поведінка NPC;
- колірна палітра;
- доступність діалогів;
- активність тригерів.

Приклад зонування:

- **Сектор спокою** — NPC пасивні, звуки ледь чутні.
- **Сектор тривоги** — підвищений фоновий шум, з’являються повторні репліки.
- **Сектор конфлікту** — зміна логіки поведінки персонажів, реакції NPC більш агресивні.

Це досягається через smart-террейни, інфопорції та `restrictor`-зони. Підхід дозволяє досягнути ефекту поступового "стискання" простору — гравець відчуває, що він не може залишити локацію не тому, що її фізично не існує, а тому, що весь світ працює за новими, нелінійними законами. На рис 3.5.

зображено локацію Кордон в редакторі розробника. Також сам інтерфейс редактора та вид на локацію Кордон зображено в Додатку А.

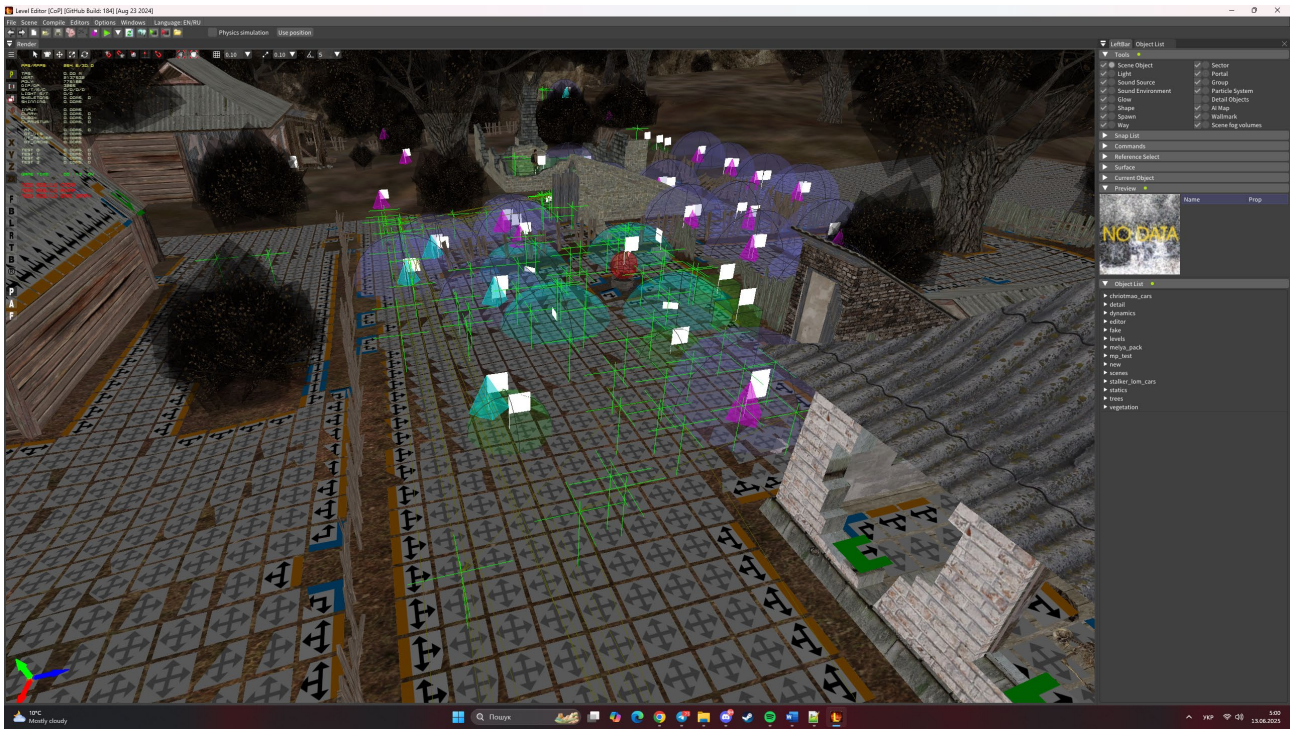


Рис 3.5. Локація Кордон в редакторі X-Ray SDK.

Було проведено редагування AI-сітки (`game.graph`) через базові інструменти `xrAI`, що дозволило:

- обмежити маршрути NPC;
- виключити недоступні ділянки;
- уникнути небажаних маршрутів під час переміщення по скриптах.

AI-сітка була адаптована до нового сценарного ритму. NPC переміщуються не в реальному режимі патрулювання, а за скриптованими секціями walker, активними лише в потрібний момент.

Гравець починає гру в зоні, стилізованій під умовну "точку пробудження". Цей простір оформлено як нейтральна ділянка без звуків, NPC, з напівтемним освітленням. Це дозволяє:

- зменшити сенсорне навантаження;
- поступово включити гравця в атмосферу;

- розпочати діалог без нав’язливих елементів геймплею.

Після цього відкривається доступ до основної зони, яка циклічно змінюється в залежності від дій гравця. Локація працює як симулятор занурення — щораз більш глибокий шар психологічного простору.

Розширення ігрової карти у модифікації *Catalyst: Complementation* не передбачало фізичного зростання кількості рівнів чи площ. Навпаки — фокус зміщено на **якісну трансформацію обмеженого простору**. Локація Кордон з build 1842 стала основою для концептуально нового підходу до ігрового дизайну: карта — не арена, а лабіринт свідомості, де події повторюються, а простір — стискається. Це дозволяє зробити з обмеженого середовища справжній механізм психологічного впливу на гравця без втручання в рушійні обмеження.

3.3. Тестування, налагодження та оптимізація доповнення

Після завершення основного етапу розробки модифікації *Catalyst: Complementation* виникла необхідність у проведенні багаторівневого тестування, діагностики помилок, їх усунення, а також оптимізації логіки, продуктивності та загального досвіду гри. Оскільки проєкт реалізовано на рушії OpenXRay із великою кількістю нових конфігурацій, діалогів, умов і сценаріїв — тестування стало критично важливим етапом.

Модифікація тестувалася вручну, у кілька послідовних фаз. Основний підхід — **пунктуація**: створення документа з переліком проблем, які потрібно виправити або додати. Кожен пункт отримував номер, короткий опис проблеми та статус виконання. Приклад пунктуації зображений в табл 3.1.

Таблиця 3.1.

№	Тип	Опис	Статус
1	Помилка	Персонаж не з’являється в точці hvost_1	Виправлено
2	Оптимізація	Затримка при переході з walker@6 на walker@7	Виконано
3	Атмосфера	Додати приглушений звук у	Додано

		секторі "тривога"	
4	Баланс	Занадто швидкий рух NPC Родрігеса	В процесі

Такий підхід дозволив ефективно координувати завдання, уникнути повторів і забезпечити поступовий прогрес у стабілізації модифікації.

Під час тестування виявлялися такі типи помилок:

- **Логічні** — неправильна робота `condlist`, `info`, `on_signal`;
- **Сценарні** — повторення діалогу, NPC не реагує на інфопорцію;
- **Візуальні** — відсутність текстури або її некоректне відображення;
- **Маршрутні** — NPC не може перейти до наступної точки (невірна AI-сітка);
- **Звукові** — аудіотрек не активується або програється надто голосно.

Багато з помилок були локалізовані завдяки системі логів OpenXRay, а також через ручне спостереження за поведінкою персонажів у різних сценаріях.

Через складну систему інфопорцій і повторюваних сценаріїв виник ризик навантаження на пам’ять та просідання FPS. Для оптимізації були виконані наступні дії:

- **Очищення зайвих info-флагів** — видалення інфопорцій, що не використовуються;
- **Зменшення кількості walker-секцій** — об’єднання дій у менше число маршрутів;
- **Оптимізація скриптів meet@** — зменшення кількості перевірок `dist_to_actor`, `see_actor`, `!actor_enemy`;
- **Видалення зайвих елементів карти** — прибрано декоративні об’єкти, що не впливали на сюжет, але навантажували рушій.

У результаті середній FPS на тестовій машині (Intel Core i5, GTX 1050Ti, 8GB RAM) залишався в межах 45–60 навіть у складних зонах.

Окремо проводилось тестування на базі **поведінкових сценаріїв новачка** — тобто симуляції гравця, який нічого не знає про мод:

Чи інтуїтивно зрозуміло, з ким говорити першочергово?

Чи логічно працює порядок відкриття діалогів?

Чи відчувається вплив дій на світ?

Виявлено, що перші 5–7 хвилин гри можуть здаватися гравцеві «порожніми». У зв’язку з цим було додано кілька **невидимих тригерів**, які автоматично активують meet-поведінку NPC, якщо гравець довго нічого не робить. Це дозволяє уникнути ситуації «застою».

Після кожного великого блоку змін проводилось **регресійне тестування** — перевірка вже виправлених помилок. Часто після правки логіки одного NPC могла порушитися поведінка іншого. Було створено таблицю залежностей info-прапорів між NPC, яка дозволила контролювати зв’язки та уникнути конфліктів.

Тестування та оптимізація модифікації *Catalyst: Complementation* стали критичним етапом, без якого неможливо було б досягти цілісного ігрового досвіду. Завдяки системі пунктуації, ручному баг-фіксу, оптимізації логіки та регресійному контролю — вдалося створити стабільну версію гри, яка зберігає атмосферу, але не перевантажує рушій. Це підготувало мод до фінального етапу — **оцінки геймплейного балансу та якості реалізації**, який розглядається у наступному розділі.

3.4. Оцінка якості реалізованої модифікації та її геймплейного балансу

Завершальним етапом розробки модифікації *Catalyst: Complementation* стала систематична оцінка її якості, досягнення цілей проєкту та відповідність очікуванням цільової аудиторії. Через специфіку гри — відсутність бойової механіки, зброї та класичних ігрових викликів — процес оцінювання мав спиратись не лише на технічні метрики, а й на психологічні, наративні та суб’єктивно переживані характеристики.

Було використано декілька підходів:

- **Суб’єктивна оцінка гравця** — наскільки гра «втягує», викликає інтерес, відчуття тривоги, занурення;
- **Формалізована пунктуація** — як метод внутрішньої оцінки виконаних завдань і ступеня доведення ідей до логічного завершення;
- **Контроль за регресією** — повторна перевірка після баг-фіксів для оцінки стабільності;
- **Наративний тест** — перевірка того, чи гравець розуміє, що відбувається, і як інтерпретує історію без зовнішніх підказок.

Основні метрики якості наведені в табл 3.2.

Таблиця 3.2.

Метрика	**Критерій оцінки**	**Результат**
Стабільність	Відсутність крашів, лагів, критичних помилок	Стабільна
Залученість	Час, який гравець проводить у моді без перерв	40–60 хв
Емоційний вплив	Відгуки про атмосферу, тривогу, напругу	Високий
Довершеність циклу	Чи сприймається «петля» як завершена структурно	Так
Відповідність інтерфейсу стилю	Мінімалізм, відсутність зайвого	Так
Відсутність бойових помилок	Враховуючи відсутність зброї	Повна
Орієнтація в просторі	Гравець розуміє, куди йти, навіть без карти	Частково
Наративна зрозумілість	Чи гравець усвідомлює роль кожного NPC	Так

Гра тестувалась серед вузького кола ентузіастів S.T.A.L.K.E.R.-моддингу. Більшість відзначили:

- атмосферу психологічної напруги;
- незвичну реалізацію класичного сталкерського простору;
- сильне наративне ядро;

- новий досвід без потреби в стрілянині або бойовій взаємодії.

Критика стосувалась лише деяких аспектів:

- часткова дезорієнтація на початку (через відсутність підказок);
- потреба у візуальних натяках на сценарний прогрес (додано в пізнішій збірці).

Оскільки в грі відсутні бойові механіки, балансування стосувалось:

- Темпу діалогів — щоб гравець не отримував надто багато інформації одночасно;
- Розташування NPC — уникнення ситуацій, коли NPC стоять надто близько або в «мертвих» зонах;
- Тривалості сцен — перевага циклам по 5–10 хвилин, що дозволяють гравцеві зануритись, але не втомитись;
- Мінімум підказок, максимум інтуїції — контроль за тим, щоб гра була зрозумілою, але не прямолінійною.

Було визначено, що **всі NPC взаємодіють з гравцем у межах логіки без потреби в зміні рушія**, а основний ігровий досвід формується через багатозначні інфопорції, змінні діалоги та атмосферу циклічності.

Оцінка якості модифікації *Catalyst: Complementation* показала, що навіть в умовах обмежених технічних змін, на основі одного рівня, без зброї, ворогів та бойових механік, можна створити наративний досвід, що впливає на гравця емоційно та когнітивно. Залучення, стабільність, відповідність візуального стилю, підтримка атмосфери та логіка взаємодії з NPC — усі ці параметри були досягнуті без компромісу якості. Таким чином, гра є завершеним експериментальним твором у жанрі психологічної драми в межах світу S.T.A.L.K.E.R.

ВИСНОВОК

У ході виконання кваліфікаційної роботи було повністю реалізовано поставлені завдання щодо створення сюжетного доповнення до гри *S.T.A.L.K.E.R.: Call of Pripyat* під назвою *Catalyst: Complementation*. Робота охопила всі ключові етапи: від аналітичного дослідження жанру до практичної реалізації, тестування та оцінки якості. Мета проєкту — створити наративно орієнтоване доповнення в межах постапокаліптичного FPS-світу — була досягнута повною мірою.

Було проведено **аналіз жанрових, технічних та наративних особливостей FPS-ігор**, зокрема таких аспектів, як бойова система, структура рівнів, типологія ворогів, способи подачі сюжету (через діалоги, щоденники, катсцени), що дозволило визначити наративні акценти для власного доповнення.

В рамках дослідження **проаналізовано рушій X-Ray Engine та його модифіковану відкриту версію OpenXRay**, визначено технічні обмеження і можливості для модифікації без зміни ядра. Це включало підтримку скриптів, діалогової системи, анімації, AI-сітки, редакторів рівнів та системи інфопорцій.

Було **розроблено концепцію сюжетного доповнення**, що досліджує теми особистості, ізоляції, повторення та психічної деградації. У межах цього проєкту було створено п’ять ключових персонажів із прописаними біографіями, архетипами, психологічною глибиною та внутрішніми арками. Сюжет побудовано за фрактальною структурою з багаторівневими діалогами.

Реалізовано **повноцінну ігрову логіку, сценарну послідовність, діалоги та квестову систему**, використовуючи механізми infoposition, condlist, meet@, logic@ та walker. Всі внутрішньоігрові події налаштовано без втручання в рушійний код, з використанням лише конфігураційних та скриптових засобів.

Було **адаптовано інтерфейс гри**, включаючи HUD, інвентар, екран діалогів, підказки та UX-моделі. Оформлення підпорядковане наративному стилю: мінімалізм, відсутність бойових індикаторів, атмосферна приглушеність. Зроблено кастомізацію DDS-файлів та розмітки XML-діалогів.

Локація Кордон з білду 1842 зазнала редизайну: збережено її геометрію, однак повністю змінено функціональну структуру через рестриктори, AI-сітку, змінну логіку та психологічне зонування. Простір використано як драматичний механізм, а не як бойову арену.

Здійснено **повноцінне тестування та оптимізацію**: реалізовано систему пунктуації для фіксації помилок, логічних збоїв і покращень; проведено багатоступеневе регресійне тестування. Оптимізовано інфопорції, маршрути, скрипти, що дозволило уникнути надмірного навантаження на рушій та підвищити стабільність гри.

Оцінка якості реалізованої модифікації здійснювалася за ключовими метриками: стабільність, час проходження, емоційна залученість, відповідність інтерфейсу наративній концепції. Усі показники підтвердили досягнення очікуваного рівня якості проєкту.

Таким чином, кваліфікаційна робота довела можливість реалізації повноцінного сюжетного доповнення до гри *S.T.A.L.K.E.R.* у межах обмежених технічних ресурсів, без додавання нового рушійного функціоналу. Результат став демонстрацією навичок у галузі геймдизайну, програмування, сценарного моделювання, інтерфейсної розробки та тестування. *Catalyst: Complementation* є завершеним прикладом інтеграції ігрової логіки, наративу та психологічного впливу на базі X-Ray Engine.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Безчотнікова А. Комп'ютерні та відеоігри як соціально-комунікаційний феномен. Діалог: Медіа-студії, 2015. Вип. 20. С. 246-255. URL: http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/dialog_2015_20_26.pdf (Дата звернення: 03.06.2025)

2. S.T.A.L.K.E.R. (серія відеоігор). Офіційний сайт https://www.stalker-game.com/uk

3. Лугова Т.А., Блажко О.А. Проектування комп’ютерних ігор для навчання : навчальний підручник. Одеса : ФОП «Побута». 2018. 212 с.

4. Відеогра – Вікіпедія. URL: https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%B4%D0%B5%D0%BE%D0%B3%D1%80%D0%B0

5. Computer Games [Електронний ресурс] // Nanyang Technological University – Режим доступу до ресурсу: https://www3.ntu.edu.sg/home/asschui/Computer%20Games.PDF

6. Jesper Juul. Half-Real: Video Games between Real Rules and Fictional Worlds. 2005. Pp. 36. https://www.scribd.com/document/248076157/Juul-Jesper-Half-Real

7. Katie Salen, Eric Zimmerman. Rules of Play: Game Design Fundamentals. 2003. Pp. 694. https://gamifique.wordpress.com/wp-content/uploads/2011/11/1-rules-of-play-game-design-fundamentals.pdf

8. Методи реалізації мережевого шутеру від першої особи на основі_вбудованих можливостей ігрового двигуна Unreal Engine 4.27. URL: https://ela.kpi.ua/server/api/core/bitstreams/c1fb5e86-ce62-4e1c-bbfc-db92d025f647/content

9. Нікітін С. О., Нікітіна Л. О. Основи комп’ютерних ігор та ігрових програм : довідник модуля. / . Х. : «Друкарня Мадрид», 2018. 138 с.

https://ec.europa.eu/programmes/erasmus-plus/project-result-content/28f2f10e-27c9-4223-af95-19a4445a41a1/prohramy_2018_Osnovy_kompiuternykh_ihor.pdf

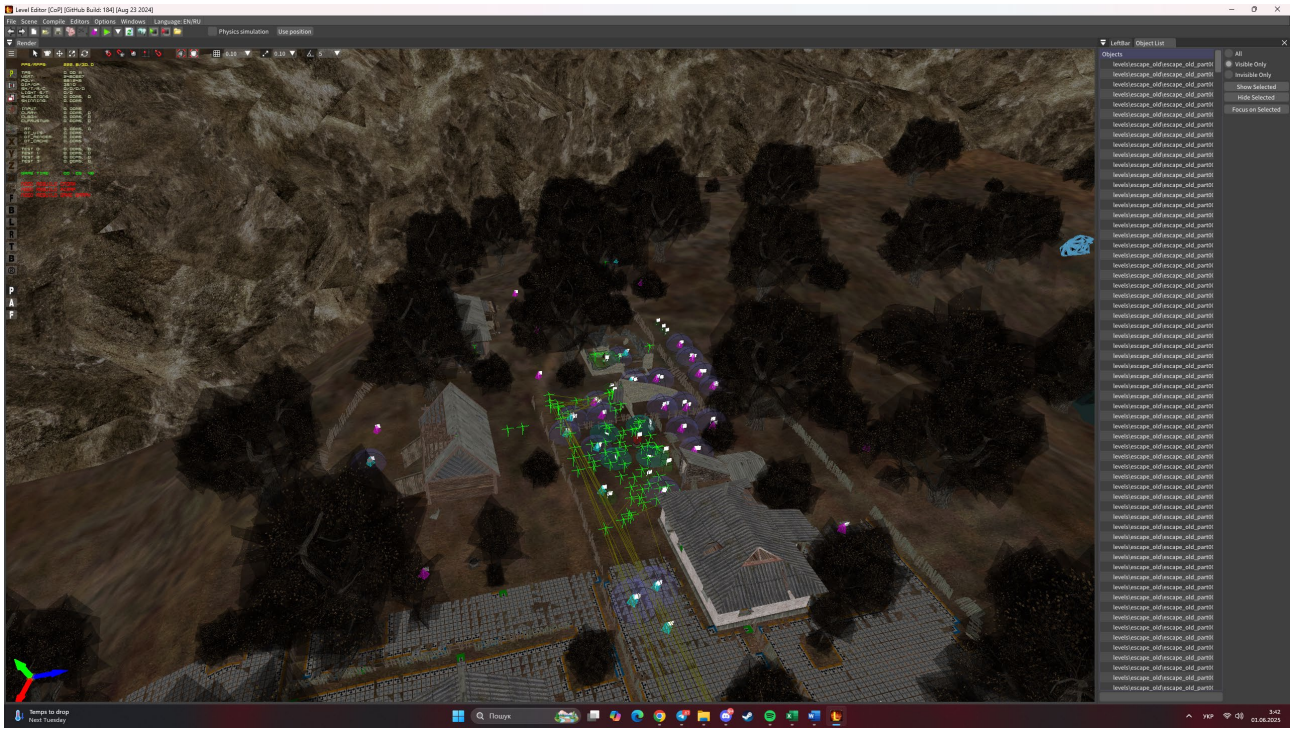
10. *Elden Ring. Офіційний сайт* https://en.bandainamcoent.eu/elden-ring/elden-ring-nightreign

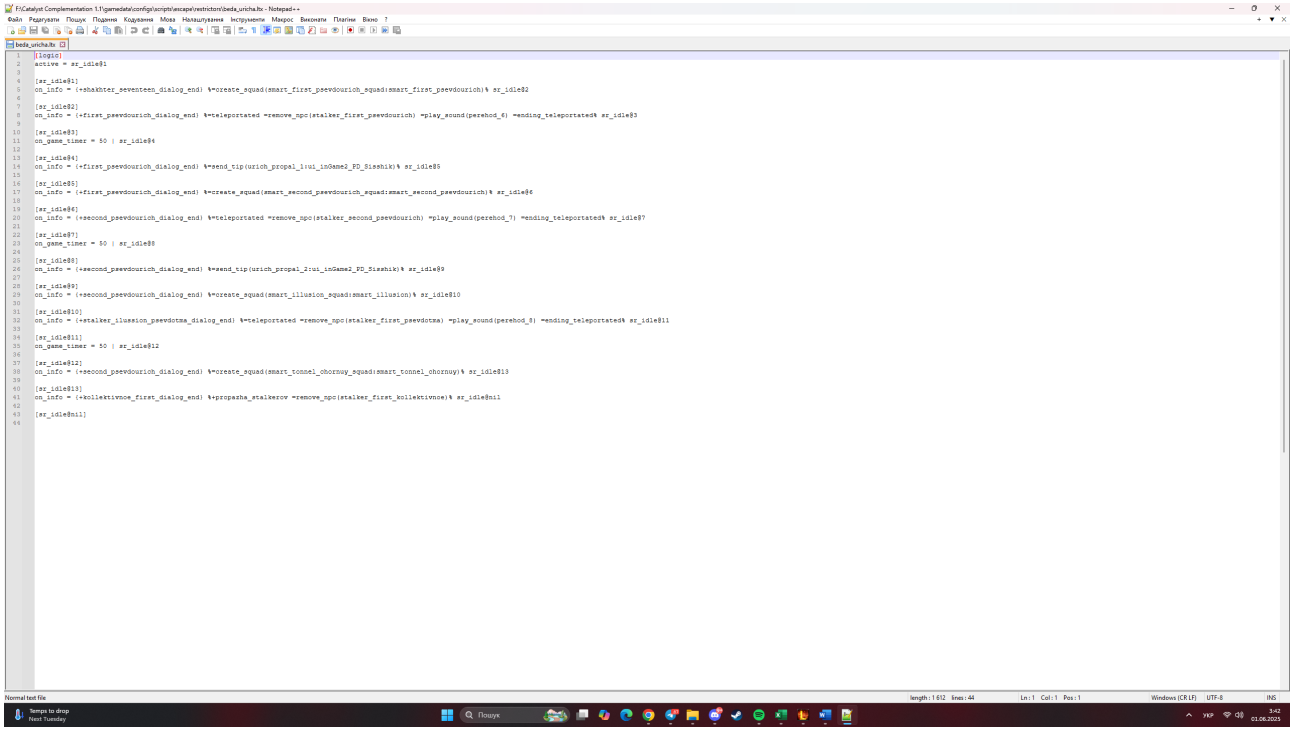
11. Шутер від першої особи – Вікіпедія. URL: https://shorturl.at/ssfTA

12. X-Ray – Вікіпедія. URL: https://uk.wikipedia.org/wiki/X-Ray_(%D1%80%D1%83%D1%88%D1%96%D0%B9_%D0%B3%D1%80%D0%B8)

13. – OpenXRay. URL: https://stalker-news.info/forum/rushij-x-ray-engine/openxray/

ДОДАТОК А





ДОДАТОК Б

```
[quest_hvost_seventh_razgovor]
icon = ui_inGame2_Put_v_pripyat
prior = 112
storyline = true
title = seventh_talked_hvost_name
descr = seventh_talked_hvost_text
on_init = %+hvost_seventh_dialog_nachalo%
condlist_0 = {+hvost_seventh_dialog_end} complete %=give_task(rodrigez_palit)%

[rodrigez_palit]
icon = ui_inGame2_Put_v_pripyat
prior = 112
storyline = true
title = rodrigez_palit_name
descr = rodrigez_palit_text
condlist_0 = {+hvost_nineth_dialog_end} complete

[quest_z_shakhterom]
icon = ui_inGame2_Put_v_pripyat
prior = 112
storyline = true
title = quest_z_shakhterom_name
descr = quest_z_shakhterom_text
condlist_0 = {+shakhter_fifth_dialog_end} complete %=give_task(quest_pohod_z_shakhterom)%

[quest_pohod_z_shakhterom]
icon = ui_inGame2_Put_v_pripyat
```

prior = 112

storyline = true

title = quest_pohod_z_shakhterom_name

descr = quest_pohod_z_shakhterom_text

condlist_0 = {=actor_has_item_count(medkit:3)} complete %=give_task(quest_pohod_z_shakhterom_2)%

on_complete = %+medkit_used%

[quest_pohod_z_shakhterom_2]

icon = ui_inGame2_Put_v_pripyat

prior = 112

storyline = true

title = quest_pohod_z_shakhterom_2_name

descr = quest_pohod_z_shakhterom_2_text

condlist_0 = {+shakhter_sixth_dialog_end} complete

[quest_pohod_z_shakhterom_3]

icon = ui_inGame2_Put_v_pripyat

prior = 112

storyline = true

title = quest_pohod_z_shakhterom_3_name

descr = quest_pohod_z_shakhterom_3_text

condlist_0 = {+shakhter_seventh_dialog_end} complete %=give_task(quest_scream)%