МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»
Факультет філології, психології та педагогіки
Кафедра германської філології та перекладу

Рекомендовано до захисту
Протокол засіданні кафедри № 7
від «22» грудня 2025р.
В.о. завідувача кафедри Палій К.В
(прізвище та ініціали)

(підпис)

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня «Магістр»

ПЕРЕКЛАД АНГЛІЙСЬКОМОВНОЇ OPEN-SOURCE ДОКУМЕНТАЦІЇ: ТЕРМІНОЛОГІЧНА КОНСИСТЕНТНІСТЬ У ПЕРЕКЛАДІ (НА ПРИКЛАДІ ДОКУМЕНТАЦІЇ GODOT ENGINE)

Виконавець:
Студент 6 курсу, групи 601-ФФ
Пінчук Євген Вікторович
(прізвище, ім'я, по батькові)

Керівник роботи:
Павельєва А.К., кандидат філологічних наук, доцент, доцент кафедри германської філології та перекладу
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада)

Рецензент:
Северіна Т.М., кандидат педагогічних наук, доцент кафедри іноземних мов Хмельницької гуманітарно-педагогічної академії
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада)

Підсумкова оцінка:
за національною шкалою: _____
кількість балів: _____
Підпис керівника _____

Полтава – 2026

ЗМІСТ

РЕФЕРАТ

МР: 78 с., 1 дод., 51 джерело.

Об’єктом дослідження є англійськомовна технічна документація Godot Engine.

Предметом дослідження стають особливості відтворення термінологічної системи рушія українською мовою та методи забезпечення її консистентності.

Мета дослідження полягає у здійсненні системного аналізу проблеми узгодженості термінології в українській локалізації Godot Engine та розробці комплексної методики її стандартизації.

У роботі використано методи порівняльного та контекстуального аналізу, метод суцільної вибірки, а також елементи емпіричного тестування (верифікація термінів у середовищі рушія).

Теоретичний розділ присвячено аналізу специфіки об'єкта перекладу. Встановлено, що open-source документація є динамічним жанром, інтегрованим в інфраструктуру розробки («Docs as Code»). Визначено, що модель краудсорсингу, попри перевагу залучення носіїв технічних знань, несе ризики «термінологічної ентропії» через відсутність фахової лінгвістичної підготовки у більшості волонтерів (до 68%). Розглянуто основні підходи до перекладу: від повного еквівалента до транслітерації та калькування. У практичній частині дослідження виявлено, що ключовими проблемами локалізації є лексична варіативність, міжмовна інтерференція та розсинхронізація термінології з інтерфейсом, спричинені специфікою краудсорсингу та якістю вихідного тексту. Для вирішення цих викликів розроблено методику контекстуальної верифікації, обґрунтовано необхідність залучення перекладачів-«техно-лінгвістів» та запропоновано інструменти уніфікації: обов’язковий глосарій, стилістичний довідник і систему пре-модерації.

ТЕХНІЧНИЙ ПЕРЕКЛАД, ЛОКАЛІЗАЦІЯ, GODOT ENGINE, ТЕРМІНОЛОГІЧНА КОНСИСТЕНТНІСТЬ, OPEN-SOURCE

ВСТУП

Стрімкий розвиток індустрії розробки програмного забезпечення та інклюзивність цифрового середовища зумовили трансформацію підходів до створення та розповсюдження контенту. Особливе місце у цій екосистемі посідають проєкти з відкритим вихідним кодом (open-source), які базуються на принципах колаборації та вільного доступу до знань, що підкреслює соціально-комунікативну значущість сучасного програмного продукту [2, с. 45]. Глобалізація ІТ-сектору вимагає якісної локалізації продуктів, де технічна документація виступає ключовим інструментом взаємодії між розробниками та кінцевими користувачами, стаючи об’єктом прискіпливої уваги перекладознавців у контексті функціонального підходу до перекладу [37, с. 12].

Актуальність теми дослідження зумовлена необхідністю наукового осмислення феномену краудсорсингу (колективного перекладу), який стає домінуючою моделлю локалізації у сфері вільного ПЗ, та пов'язаних із ним ризиків термінологічної розупорядкованості [5, с. 12]. Відсутність централізованого контролю, різний рівень кваліфікації учасників та специфічна якість вихідних текстів, написаних розробниками, створюють складні виклики для збереження термінологічної консистентності. Це безпосередньо впливає на функціональність та зрозумілість кінцевого продукту для українського користувача, оскільки уніфікація термінології у вихідному тексті та перекладі є головною запорукою якості технічної комунікації [4, с. 113].

Мета дослідження полягає у здійсненні системного аналізу проблеми узгодженості термінології в українській локалізації Godot Engine та розробці комплексної методики її стандартизації з урахуванням технічних та людських факторів, що ґрунтується на застосуванні сучасних методологічних парадигм перекладознавства [36, с. 34].

Для досягнення мети поставлено такі завдання:

- визначити теоретичні засади перекладу open-source документації та дослідити вплив краудсорсингу на термінологічну єдність у світлі концепції перекладу як соціальної активності в цифрову епоху;
- проаналізувати структурно-семантичні особливості документації Godot Engine та виявити вплив якості оригіналу на процес перекладу;
- дослідити стан української локалізації рушія та класифікувати типові термінологічні помилки;
- розробити практичні рекомендації щодо уніфікації термінології та сформулювати компетентнісний профіль перекладача технічної документації в ігровій індустрії.

Об’єктом дослідження є англійськомовна технічна документація Godot Engine.

Предметом дослідження стають особливості відтворення термінологічної системи рушія українською мовою та методи забезпечення її консистентності.

Матеріалом дослідження слугували офіційні ресурси проєкту Godot (Class Reference, Online Manual) [30] та корпус перекладів відкритої платформи Weblate [49]. У роботі використано комплекс методів, що відповідають дескриптивній парадигмі сучасного перекладознавства: методи порівняльного та контекстуального аналізу для виявлення термінологічних девіацій, метод суцільної вибірки для формування корпусу дослідження, а також елементи емпіричного тестування (звіряння термінів у середовищі рушія) для верифікації технічної точності перекладу [46, с. 24].

Наукова новизна одержаних результатів полягає у тому, що в роботі вперше комплексно проаналізовано українську локалізацію Godot Engine крізь призму концепції «Docs as Code», що розглядає документацію як невід'ємну частину програмного коду та вимагає відповідних інженерних підходів до її перекладу. Визначено роль «Developer English» як

специфічного дестабілізуючого фактора, що зумовлює термінологічну неоднорідність оригіналу та ускладнює процес локалізації в ігровій індустрії. Науково обґрунтовано концепцію «техно-лінгвіста» (гібридного спеціаліста) як критично необхідного типу фахівця для роботи в умовах безперервної локалізації та краудсорсингових моделей.

Практичне значення дослідження полягає у розробці конкретних інструментів уніфікації (рекомендацій щодо глосарія та стайлгайду), що базуються на міжнародних стандартах технічного письма та практичному досвіді українських локалізаційних спільнот [35; 43].

Апробація результатів дослідження. Наукове дослідження було апробоване під час VI Міжнародної науково-практичної конференції «АКТУАЛЬНІ ПРОБЛЕМИ ІНОЗЕМНОЇ ФІЛОЛОГІЇ ТА ПЕРЕКЛАДОЗНАВСТВА» (11-12 листопада 2025); 77-ї наукової конференції професорів, викладачів, наукових працівників, аспірантів та студентів університету (Полтава, 16 трав. – 22 трав. 2025 р.). За матеріалами доповіді було опубліковано тези: Пінчук Є.В. Переклад англомовної open-source документації: термінологічна консистентність у перекладі на прикладі документації Godot Engine / Є.В. Пінчук, Т.В. Кушнірова // Тези 77-ї наукової конференції професорів, викладачів, наукових працівників, аспірантів та студентів університету (Полтава, 16 трав. – 22 трав. 2025 р.). – Полтава : Нац. ун-т ім. Юрія Кондратюка, 2025. – Т. 1. – С. 295–296. https://reposit.nupp.edu.ua/handle/PoltNTU/19034

Обсяг і структура роботи. Магістерська робота складається зі вступу, трьох розділів, висновків до розділів, загальних висновків, резюме англійською мовою (Summary), списку використаних джерел та додатків. Загальний обсяг роботи становить 78 сторінок.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ПЕРЕКЛАДУ OPEN-SOURCE ДОКУМЕНТАЦІЇ

1.1. Особливості open-source документації як об'єкта перекладу

У сучасному перекладознавстві технічна документація розглядається як окремий жанр, що характеризується прагматичною спрямованістю та специфічною мовною організацією, проте документація до проєктів із відкритим вихідним кодом (open-source) становить унікальну категорію текстів [9, с. 15]. Вона виконує роль комунікаційного моста між розробниками та кінцевими користувачами, і її якість часто стає вирішальним фактором у прийнятті рішення про використання конкретного програмного продукту, безпосередньо впливаючи на успішність локалізації [28, с. 8]. На відміну від комерційного програмного забезпечення, де документація створюється штатом професійних технічних письменників за чітко затвердженими стандартами, open-source документація формується спільнотою (community-driven). Цей фактор накладає фундаментальні особливості на її структуру, зміст та процес перекладу, перетворюючи її на динамічний об'єкт цифрового середовища [38, с. 12].

Насамперед, варто визначити природу open-source документації. На відміну від комерційних продуктів, де контент створюється професійними технічними письменниками за чіткими регламентами, документація відкритих проєктів (таких як Godot Engine [30], Linux, Blender) часто формується за принципом краудсорсингу (crowdsourcing). Це означає, що авторами вихідного тексту виступають сотні різних людей - від розробників ядра до звичайних користувачів, що перетворює процес написання на децентралізовану соціальну активність [38, с. 14]. Така гетерогенність авторського колективу породжує ключову особливість об'єкта перекладу - стилістичну та термінологічну неоднорідність оригіналу, де часто спостерігається поєднання високого рівня технічного жаргону з розмовними конструкціями [9, с. 142; 23, с. 54]. Для перекладача це створює складний

виклик: необхідно не просто забезпечити еквівалентність, а й здійснити стилістичну уніфікацію та збереження семантичної цілісності тексту, щоб для кінцевого реципієнта документація виглядала як логічно завершений твір [21, с. 32; 34, с. 18].

Другою важливою рисою є концепція «Docs as Code» (документація як код), яка передбачає застосування ідентичних інструментів та робочих процесів як для написання програмного коду, так і для створення супровідних текстів [28, с. 14]. У проєктах рівня Godot Engine документація не є статичним масивом тексту; вона глибоко інтегрована в інфраструктуру розробки та зберігається у репозиторіях систем контролю версій (наприклад, Git). Тексти створюються у полегшених форматах розмітки (Markdown, reStructuredText), що вимагає від перекладача не лише філологічної підготовки, а й володіння навичками технічного редагування та базового розуміння синтаксису програмування [24, с. 92]. Помилки у збереженні службових тегів чи посиланнях можуть призвести до порушення цілісності автоматизованих систем збірки документації, що робить перекладацьку працю частиною загального інженерного процесу, де лінгвіст фактично виступає в ролі технічного спеціаліста [32, с. 116; 35].

Специфіка підходу «Docs as Code» накладає на перекладача низку технічних вимог, які виходять за межі традиційної філологічної компетенції. Насамперед, це вимагає прецизійної роботи з синтаксисом: фахівець повинен вміти чітко диференціювати текст, що підлягає перекладу, від елементів програмного коду, посилань та змінних, що є обов’язковою умовою використання сучасних інструментів автоматизованого перекладу [24, с. 94]. Будь-яка помилка у збереженні синтаксичної структури службових тегів може призвести до критичного збою, за якого сторінка документації просто не пройде етап компіляції (build failed), що робить технічну грамотність лінгвіста запорукою цілісності всього продукту [28, с. 18]. Додатковим ускладнюючим фактором є контекстуальна розривність: у спеціалізованих системах перекладу, інтегрованих із репозиторіями (як-от Weblate [49]),

вихідний матеріал подається сегментовано - окремими рядками або реченнями (strings). Така атомізація тексту призводить до того, що перекладач часто не бачить повної візуальної картини сторінки, що суттєво підвищує ризик помилкової інтерпретації термінів та стилістичного розбалансування контенту [34, с. 29; 12, с. 115].

Третьою важливою особливістю об’єкта перекладу є динамічність та безперервність оновлень, що в індустрії розробки програмного забезпечення відомо як модель «Rolling Release». Open-source проєкти розвиваються надзвичайно стрімко, тому документація не сприймається як статичний масивом даних - вона змінюється практично щодня. Це явище в сучасному перекладознавстві класифікують як роботу з «живим текстом», що вимагає переходу до парадигми безперервної локалізації [26, с. 42; 49]. Перекладач змушений не лише відстежувати появу нових сегментів, а й оперативно оновлювати раніше перекладені частини, щоб уникнути дезінформації користувача через термінологічні розбіжності між версіями продукту [28, с. 148]. Яскравим прикладом такої динаміки є перехід від версії рушія Godot 3 до Godot 4 [30], який супроводжувався радикальними змінами в архітектурі та, як наслідок, необхідністю тотальної ревізії відповідних розділів документації. За таких умов підтримка термінологічної консистентності трансформується із разового завдання у безперервний ітеративний процес, що потребує постійного термінологічного моніторингу [45, с. 56].

Окремо слід виділити фактор цільової аудиторії, яка в ігровій індустрії відрізняється високим ступенем гетерогенності. Читачами документації open-source проєктів є як початківці, для яких важлива доступність і простота пояснень, так і досвідчені програмісти, які вимагають абсолютної термінологічної точності, що змушує перекладача орієнтуватися на багатошарову прагматику тексту [39, с. 142]. Переклад має балансувати між цими двома полюсами, уникаючи як надмірного спрощення, так і незрозумілої калькованої термінології, що відповідає нормам культури фахового мовлення [16, с. 54]. Також значним пластом проблем виступає

локалізація фрагментів програмного коду та супровідних коментарів. Документація містить численні приклади, де назви змінних та літерали потребують адаптації; при цьому вибір стратегії між калькуванням та транслітерацією назв функцій є нетривіальним завданням, оскільки переклад має сприяти розумінню алгоритму, не порушуючи логіку його реалізації в самому рушії [23, с. 118; 25, с. 210]. Крім того, перекладач повинен враховувати культурні особливості аудиторії, адаптуючи технічні метафори та аналогії, що використовуються для пояснення абстрактних концепцій програмування [19, с. 145].

Слід виділити економіко-мотиваційний аспект, оскільки переклад open-source проєктів зазвичай здійснюється на волонтерських засадах за моделлю краудсорсингу, що відображає сучасні трансформації мовної ситуації в глобальному цифровому просторі [1, с. 42]. Це створює унікальне середовище, де перекладачі керуються внутрішнім інтересом та прагненням до розвитку спільноти, а не фінансовою винагородою, перетворюючи локалізацію на форму спільного виробництва знань [34, с. 18]. Така модель дозволяє залучати фахівців із глибокою технічною експертизою, що часто забезпечує вищу точність передачі концептуального змісту, ніж при залученні зовнішніх агенцій [8, с. 112]. Водночас волонтерський характер праці та відсутність централізованого контролю, поєднані з плинністю кадрів, часто призводять до появи термінологічних дублетів та руйнування системності фахової мови [10, с. 206]. Наприклад, термін *viewport* може перекладатися як «в’юпорт», «видовий екран» або «область перегляду». Відтак, ризики фрагментації термінології потребують впровадження жорстких механізмів координації: спільних глосаріїв, стайлгайдів та систем перехресного рецензування, що є критично важливим для менеджменту термінології у великих проєктах [50, с. 214; 43]. Отже, open-source документація як об'єкт перекладу - це складний, динамічний текст, інтегрований у технічну інфраструктуру, що вимагає від лінгвіста поєднання перекладацьких навичок із розумінням інженерних принципів розробки ПЗ.

Отже, open-source документація як об'єкт перекладу - це складний, динамічний, мультиавторський текст, інтегрований у технічну інфраструктуру проєкту, що вимагає від перекладача поєднання лінгвістичних навичок із розумінням принципів розробки програмного забезпечення та роботи систем контролю версій.

1.2. Феномен краудсорсингу в перекладі: переваги та ризики для термінологічної єдності

У сучасному перекладознавстві та практиці локалізації програмного забезпечення відбулася фундаментальна зміна парадигм: перехід від традиційної лінійної моделі TEP (*Translation, Editing, Proofreading*), де задіяні професійні лінгвісти, до моделі колаборативного перекладу, відомої як краудсорсинг (*crowdsourcing*) [5, с. 8]. Цей термін описує практику передачі певних виробничих функцій невизначеному колу осіб через відкриту оферту. У контексті таких open-source проєктів, як Godot Engine, краудсорсинг стає безальтернативним методом локалізації через величезні обсяги текстового матеріалу та відсутність бюджету на оплату праці професійних агенцій [34, с. 22]. Як зазначає М. О'Хаган, переклад у цифрову епоху трансформується із суто професійної послуги на соціальну активність спільноти, де ключову роль відіграє колективна взаємодія навколо спільного продукту [38, с. 11].

Специфікою перекладацького краудсорсингу є радикальна зміна статусу виконавця. У цій моделі перекладачем виступає не дипломований фахівець, а так званий «прозюмер» (*prosumer* - від *producer* + *consumer*) - активний користувач продукту, який одночасно споживає його і бере участь у його створенні [34, с. 45; 38, с. 14]. Це має безумовну перевагу: такий перекладач володіє імпліцитним знанням предметної області, що є критично важливим для адекватної інтерпретації технічних функцій [23, с. 12]. Користувач Godot, який щодня працює з рушієм, розуміє механіку роботи вузла *RayCast* або принципи рендерингу краще за філолога, який ніколи не

запускав середовище розробки, що забезпечує високу термінологічну точність на рівні розуміння продукту [39, с. 156]. Проте відсутність фахової лінгвістичної підготовки у більшості волонтерів створює системний ризик зниження мовної якості тексту: ігнорування граматичних норм, зловживання інтерференційними кальками та загальну стилістичну еклектику, що потребує додаткових зусиль для редагування [5, с. 32; 9, с. 158].

Технологічно процес краудсорсингового перекладу базується на деконструкції тексту. На спеціалізованих вебплатформах, таких як Weblate [49] або Crowdin, вихідний матеріал розбивається на мікрозавдання - окремі речення або синтагми («стрічки»), що є характерним для сучасних систем автоматизованого перекладу [24, с. 102]. Це дозволяє залучити до роботи сотні учасників, кожен з яких може виконати невеликий обсяг перекладу у вільний час [34, с. 24]. Однак така атомізація тексту часто руйнує його смислову цілісність: перекладач працює з ізольованим сегментом, не бачачи контексту сусідніх речень, що призводить до явища «контекстуальної сліпоти» та унеможливлює відстеження логічної зв’язності [34, с. 29]. Як наслідок, виникає ефект «клаптикової ковдри», коли один розділ документації містить терміни, перекладені в різних стилістичних регістрах та з різним ступенем адаптації, що суттєво знижує когнітивну якість фінального тексту [38, с. 15; 45, с. 112].

З погляду термінологічної консистентності, головним викликом краудсорсингу є явище, яке можна визначити як «термінологічна ентропія». У відкритій системі без жорсткої централізованої регуляції кількість варіантів перекладу одного терміна прагне до нескінченності, пропорційно до кількості учасників процесу, що зумовлює термінологічну розупорядкованість [34, с. 32]. Кожен волонтер приносить у проєкт власний ідіолект та бачення професійного жаргону, що за відсутності примусових механізмів уніфікації (інтегрованих глосаріїв) неминуче призводить до термінологічного хаосу [50, с. 142; 4, с. 113]. Наприклад, термін *rendering* може одночасно фігурувати як «рендерінг», «візуалізація» або

«відмальовування». Така варіативність, допустима в художніх текстах, є критичною вадою для технічної документації, де однозначність (моносемія) терміна в межах однієї системи є запорукою точного сприйняття інформації та коректного виконання інструкцій користувачем [10, с. 206; 6, с. 54].

Однак краудсорсинг має і внутрішні механізми стабілізації якості, відомі як «мудрість натовпу» (*wisdom of the crowd*), де колективний інтелект спільноти дозволяє мінімізувати індивідуальні помилки через масовий перехресний контроль [34, с. 56]. У великих спільнотах помилкові варіанти перекладу часто виправляються іншими учасниками через систему голосування або пропозицій, що перетворює процес на форму безперервного редагування [38, с. 18]. На платформі Weblate [49] це реалізовано через функцію коментарів та перевірок, де будь-який користувач може позначити переклад як сумнівний. Проте для документації рівня Godot Engine саморегуляції спільноти недостатньо. Необхідним стає впровадження ієрархічної структури з модераторами (*language leads*), які мають фінальне слово у виборі терміна та забезпечують суворе дотримання глосарія і системності фахового викладу [5, с. 41; 50, с. 230].

Окремим аспектом, що визначає специфіку краудсорсингового перекладу, є психологічна мотивація учасників. На відміну від професійного перекладача, обмеженого контрактними зобов'язаннями та термінами (*deadlines*), волонтер керується насамперед внутрішнім інтересом та ідеєю «спільного блага», що перетворює процес локалізації на акт альтруїстичної технічної комунікації [38, с. 12]. Проте відсутність фінансової та юридичної відповідальності часто призводить до зниження порогу самокритики. У прагненні якнайшвидше побачити результат своєї праці в готовому продукті, волонтери можуть нехтувати етапом глибокого термінологічного пошуку, обираючи найбільш очевидні, але не завжди коректні лексичні відповідники [34, с. 68]. У цьому контексті виникає проблема «адаптивного перекладу», де волонтер підсвідомо спрощує технічні поняття, орієнтуючись на власний обмежений досвід. У масштабах усього проєкту це призводить до

поступового розмивання наукового стилю документації, втрати прецизійності та порушення норм фахової мови, які є критично важливими для науково-технічного дискурсу [9, с. 156; 45, с. 84; 5, с. 34].

Важливим дестабілізуючим фактором є проблема комунікаційного розриву між розробниками та лінгвістами. У краудсорсингових проєктах, таких як Godot Engine [30], процес перекладу часто відбувається у відриві від процесу розробки концепції продукту. Комунікація між авторами вихідного тексту (програмістами) та перекладачами (волонтерами) зазвичай опосередкована платформою локалізації Weblate [49], яка не завжди дозволяє вести розгорнуті дискусії щодо значення специфічних термінів, що поглиблює ризики неправильної інтерпретації контенту [34, с. 42]. Це створює ситуацію, за якої перекладач змушений самостійно трактувати авторські неологізми. Як зазначає Б. Есселінк, без наявності оперативного каналу зв'язку з розробниками для уточнення семантичних нюансів, спільнота волонтерів починає виробляти власну «народну етимологію» термінів, яка може суттєво відрізнятися від первинного задуму авторів рушія [28, с. 116; 23, с. 144].

Нерівномірність внеску учасників також суттєво впливає на термінологічну стабільність, оскільки будь-яка комунікативна система потребує високого ступеня впорядкованості для ефективного передавання фахової інформації [2, с. 88]. Дослідження відкритих спільнот підтверджують дію «правила 1%» (або закону Парето), згідно з яким переважну більшість контенту створює критично мала частка учасників, тоді як решта вносить лише епізодичні правки, що є характерною рисою сучасних перекладацьких екосистем у цифровому середовищі [34, с. 72]. У локалізації Godot Engine це проявляється в тому, що основний масив термінологічних рішень приймається вузькою групою найбільш активних контриб'юторів. Це створює ризик «термінологічного авторитаризму», коли суб'єктивне рішення однієї впливової особи стає нормою для всього проєкту, що може призводити до відхилень від загальнолінгвістичних стандартів на користь

вузькоспеціалізованого жаргону [38, с. 19; 5, с. 44]. Водночас хаотичні правки від випадкових учасників, які перекладають лише кілька рядків, вносять у текст елементи випадковості, що ще більше розхитує систему та вимагає постійного втручання модераторів для збереження консистентності [34, с. 115].

Таким чином, краудсорсинг у перекладі постає як багаторівневе явище, де лінгвістичні виклики тісно переплетені з психологічними та організаційними чинниками. Забезпечення консистентності в таких умовах потребує не лише лінгвістичного інструментарію, а й розробки складних механізмів соціальної модерації та технічного контролю, які б дозволили гармонізувати колективні зусилля навколо єдиного термінологічного стандарту.

1.3. Специфіка перекладу технічної документації

Переклад технічної документації є окремим видом перекладацької діяльності, що вирізняється суворою регламентованістю, прагматичною спрямованістю та специфічними вимогами до точності передачі інформації [44, с. 15]. На відміну від художнього перекладу, де домінує естетична функція, основною метою технічного перекладу є забезпечення точної, повної та однозначної комунікації між автором (розробником) і користувачем продукту [28, с. 22]. Ключовою рисою перекладу технічної документації є домінування денотативного значення над конотативним. Його головною прагматичною метою є точна передача конкретної інформації, необхідної фахівцю для виконання певних дій або розуміння роботи системи [45, с. 38]. Як зазначає В. Карабан, технічний текст вимагає від перекладача повної відмови від емоційно-експресивних засобів на користь логічності, об’єктивності та, насамперед, однозначності викладу [9, с. 210]. Перекладач повинен суворо уникати синонімії там, де вона може призвести до змістової плутанини: один термін в оригіналі має відповідати одному стабільному відповіднику в перекладі. Це правило є фундаментальним для збереження

термінологічної консистентності, особливо в контексті складних програмних продуктів, таких як ігрові рушії, де термінологічна варіативність прямо корелює з ризиками експлуатаційних помилок [4, с. 113; 50, с. 15].

Специфіка перекладу технічної документації виявляється на кількох мовних рівнях:

1. Лексичний рівень: Основою технічного тексту є терміни, які, за визначенням, мають бути однозначними, точними та системно пов'язаними [10, с. 206]. При перекладі документації лінгвіст стикається з проблемою вибору між транслітерацією (наприклад, «шейдер» від *shader*), калькуванням («затінювач») або описовим перекладом, що вимагає глибокого аналізу внутрішньої форми терміна [6, с. 54]. Особливу складність становлять неологізми, які ще не зафіксовані у словниках, а також багатозначні слова загальновживаної лексики, що набувають специфічного вузькогалузевого значення (наприклад, *parent, child* у контексті ієрархії вузлів сцени Godot). Це вимагає від перекладача не просто мовної інтуїції, а глибокого когнітивного розуміння технічного контексту та архітектури продукту [28, с. 154; 24, с. 110].
2. Граматичний та синтаксичний рівень: Технічна документація англійською мовою тяжіє до використання пасивного стану, безособових форм дієслова та складних іменникових груп (наприклад, *rigid body physics integration*). Українська мова, навпаки, частіше використовує активні конструкції або безособові речення на *-но, -то*, що потребує від перекладача застосування відповідних граматичних трансформацій для досягнення адекватності [12, с. 142; 48, с. 84]. Важливим є дотримання імперативного стилю в інструкціях (наприклад, «Натисніть», «Виберіть»), що забезпечує чіткість вказівок та відповідає галузевим стандартам локалізації ІТ-продуктів [35].
3. Стилістичний рівень: Стиль технічної документації характеризується лаконічністю, логічністю та максимальною об’єктивністю викладу.

Перекладач має дотримуватися принципів «контрольованої мови» (*controlled language*) - обмеженого набору лексичних та граматичних засобів, що спрощує сприйняття тексту та мінімізує ризики неоднозначного тлумачення [44, с. 156; 9, с. 240]. Це особливо важливо для *open-source* документації, яку часто читають користувачі з різним рівнем мовної підготовки або ті, для кого мова перекладу не є рідною.

Неправильний переклад ключового терміна «node» як «елемент» або «точка» може спричинити критичну плутанину в розумінні ієрархічної структури сцени, що є засадничим механізмом роботи з рушієм [30]. Це підтверджує тезу Т. Кияка про те, що ідеальний термін повинен бути моносемічним у межах своєї терміносистеми, тобто мати лише одне фіксоване значення, що виключає двозначність при читанні технічних інструкцій [10, с. 207].

Іншою важливою особливістю є контекстуальна залежність термінів. Багато лексичних одиниць є багатозначними, і вибір відповідника безпосередньо залежить від конкретної технічної ситуації [13, с. 45]. Яскравим прикладом є термін *script*, який у загальному сенсі може перекладатися як «сценарій» (план дій), але в контексті Godot Engine він означає виключно «скрипт» - файл із програмним кодом [18, с. 112]. Перекладач повинен не лише володіти лінгвістичними відповідниками, а й розуміти, як ці об’єкти функціонують у програмному середовищі, щоб уникнути помилкових конотацій та забезпечити функціональну адекватність перекладу [39, с. 162]. До цього ж відноситься й проблема перекладу абревіатур та скорочень (API, GUI), де помилкова розшифровка або некоректна адаптація може призвести до повного викривлення змісту інструкції, тому в цьому аспекті критично важливою є опора на вузькоспеціалізований контекст та галузеві термінологічні ресурси [50, с. 341].

Окремий виклик становить інтеграція природної мови з елементами коду. У технічній документації часто зустрічаються приклади, що містять

ключові слова (func, var, signal), назви змінних та літерали. Ці елементи зазвичай залишаються незмінними, оскільки є частиною синтаксису мов програмування (GDScript, C#), проте завдання перекладача полягає в тому, щоб супровідний текст коректно пояснював логіку роботи алгоритму, не порушуючи технічної достовірності прикладів [30; 23, с. 148]. Це вимагає від лінгвіста базових навичок читання коду, що стає обов'язковою компетенцією в сучасній IT-локалізації [24, с. 115].

Зрештою, значною проблемою залишається відсутність стандартизованої української термінології для новітніх концепцій, що змушує перекладачів самостійно шукати відповідники, створюючи ризик термінологічних розбіжностей [3, с. 84]. Наприклад, *«scene tree»* може перекладатися як «дерево сцени» або «ієрархія сцени», як було згадано вище у пункті про проблеми на лексичному рівні, що без належної координації руйнує системність викладу. У таких випадках критично важливим стає узгодження термінології зі спільнотою та створення глосаріїв для забезпечення єдності перекладу в усьому масиві документації [15, с. 76; 50, с. 214].

Отже, специфіка перекладу технічної документації полягає у поєднанні жорстких вимог до термінологічної точності з необхідністю адаптації тексту під норми цільової мови, і таким чином, переклад технічної документації open-source проєктів є комплексним процесом, що вимагає від фахівця поєднання глибокої філологічної підготовки з технічною компетентністю.

1.4. Основні підходи до перекладу open-source проектів

Переклад проектів із відкритим вихідним кодом (open-source), зокрема їхньої документації, базується на специфічних підходах, що зумовлені колаборативною природою розробки вільного програмного забезпечення, та вимагає застосування специфічних стратегій, що відрізняються від традиційних методів перекладу пропрієтарного програмного забезпечення. Якщо у закритих проектах переклад зазвичай виконується невеликою групою

професіоналів, то в open-source спільнотах домінує підхід краудсорсингу (crowdsourcing) або спільного перекладу. Цей підхід визначає вибір лексичних трансформацій та методів забезпечення якості. Іншим із провідних підходів до перекладу open-source документації є використання колаборативних платформ, які функціонально поділяються на системи контролю версій та спеціалізовані інструменти локалізації [32, с. 116; 34, с. 24]. Документація Godot Engine інтегрована в інфраструктуру розробки на платформі GitHub, що дозволяє будь-якому учаснику спільноти пропонувати зміни через механізм *Pull Request*, забезпечуючи повну прозорість процесу [30]. Проте для безпосереднього перекладу текстових сегментів найчастіше застосовуються спеціалізовані хмарні платформи на кшталт Weblate, на якій і відбувається переклад документації Godot Engine [49]. Ці інструменти дозволяють волонтерам працювати паралельно, обговорювати термінологію в режимі реального часу та миттєво бачити прогрес проєкту. Як зазначає М. Хіменес-Креспо, такі платформи фактично демократизують процес перекладу, перетворюючи його на динамічну соціальну взаємодію [34, с. 102].

Основним викликом при локалізації таких масштабних проєктів, як Godot Engine, є вибір стратегії передачі термінології, що вимагає від перекладача глибокого аналізу семантичної структури терміна та його функціонального навантаження [28, с. 22]. Аналіз науково-технічної літератури дозволяє виділити кілька ключових способів відтворення термінів, які застосовуються в open-source документації [9, с. 210; 12, с. 142]:

– Еквівалентний переклад. Це найбільш бажаний метод, при якому англійському терміну відповідає усталений український відповідник, зафіксований у словниках (наприклад, *library* - *бібліотека*, *value* - *значення*) [9, с. 210]. Цей підхід забезпечує точність і зрозумілість тексту для широкого загалу, проте в ІТ-сфері, що стрімко розвивається, словникових відповідників часто бракує [18, с. 5].

– Транскодування (транслітерація та транскрипція). У випадках, коли еквівалент відсутній або термін є новим, перекладачі вдаються до передачі графічної або звукової форми терміна (*shader - шейдер*, *cache - кеш*) [12, с. 142]. Цей метод дозволяє зберегти впізнаваність терміна для фахівців, які звикли до англомовного інтерфейсу, але може бути незрозумілим для новачків без належного пояснення [23, с. 54].
– Калькування. Передбачає переклад складових частин слова або словосполучення відповідними елементами мови перекладу (наприклад, *rigid body - жорстке тіло*) [9, с. 576]. Калькування дозволяє інтегрувати іншомовне поняття в граматичну систему української мови, зберігаючи його внутрішню форму [6, с. 54].
– Експлікація (описовий переклад). Застосовується, якщо термін не має відповідника і його транслітерація не дає чіткого уявлення про зміст [14, с. 158]. Хоча цей метод забезпечує прозорість змісту, його недоліком є громіздкість, що суперечить принципу лаконічності технічної документації, тому в перекладі інтерфейсу (UI) його намагаються уникати [28, с. 154].

Важливим аспектом при виборі методу перекладу є аналіз контексту, оскільки більшість технічних термінів є полісемантичними (багатозначними), і лише мікроконтекст дозволяє визначити їхнє точне денотативне значення [13, с. 45; 21, с. 110]. Наприклад, у документації *Godot Engine* лексема *node* може позначати як «вузол» (базовий елемент ієрархії сцени), так і «комп’ютер» у контексті мережевих налаштувань [30]. Помилка в аналізі контексту призводить до викривлення змісту інструкції та зниження прагматичної цінності тексту.

Як згадується раніше, не менш важливою частиною перекладу open-source документації є активна участь спільноти волонтерів (краудсорсинг). Цей підхід дозволяє залучати до процесу людей із різним технічним досвідом та знаннями, що сприяє багатогранному покращенню якості перекладу [5, с.

34; 38, с. 14]. У випадку Godot Engine перекладацька спільнота оперативно реагує на оновлення оригінальної документації, що особливо важливо при значних змінах продукту. Проте волонтерська модель вимагає розробки чітких стандартів локалізації та термінологічної політики. Оскільки технічні тексти Godot містять велику кількість понять, що не мають усталених українських еквівалентів (наприклад, *«node», «scripting API»*), перекладачі змушені колективно узгоджувати відповідники, щоб уникнути термінологічного розбігу [4, с. 113; 50, с. 214]. У цьому контексті використання спільних глосаріїв стає запобіжником проти фрагментації терміносистеми [43; 50, с. 341]. Для забезпечення єдності термінології в умовах, коли над проєктом працюють десятки волонтерів, критично важливим є використання глосаріїв та баз пам'яті перекладів (*Translation Memory*) [24, с. 102; 50, с. 214]. Це дозволяє уникнути ситуації, коли один і той самий термін (наприклад, *viewport*) різні учасники перекладають по-різному, оскільки синонімія термінів є небажаною у технічному перекладі та заважає адекватному сприйняттю інструкцій [10, с. 206]. Застосування автоматизованих інструментів (*CAT-інструментів*), завдяки яким і відбувається використання пам’яті перекладів, значно прискорює процес локалізації, дозволяючи повторно використовувати вже перекладені фрагменти у нових версіях документації [24, с. 185]. Це забезпечує не лише значну економію часу, а й сталість стилю та термінології в межах усього масиву текстів. Як підкреслює Л. Боукер, автоматизація в технічному перекладі є критичною для підтримання консистентності, особливо у великих проєктах, де загальний обсяг тексту перевищує фізичні можливості однієї людини [24, с. 116].

Окрему увагу в методології *open-source* перекладу приділяють синхронізації та безперервній локалізації (*Continuous Localization*) [34, с. 29]. Оскільки проєкт *Godot* постійно розвивається, його документація оновлюється майже щодня, що вимагає від перекладачів не лише відтворювати новий текст, а й оперативно адаптувати існуючі приклади коду,

коментарі та культурні реалії до норм цільової мови [30]. Це вимагає від лінгвіста навичок роботи безпосередньо в екосистемі розробки, де переклад розглядається не як фінальна точка, а як постійний ітеративний процес, глибоко інтегрований у життєвий цикл продукту [28, с. 14].

Отже, сучасний підхід до перекладу open-source проектів базується на поєднанні автоматизованих засобів контролю якості (CAT-tools) із гнучким використанням різних перекладацьких трансформацій - від повного еквівалента до транслітерації - залежно від типу документації (інтерфейс, довідник API чи навчальний туторіал) та рівня підготовки цільової аудиторії, таким чином, успішний переклад open-source проєкту є результатом синергії технологічних інструментів та колективного інтелекту спільноти. Поєднання колаборативних платформ, автоматизації та суворих стандартів термінології дозволяє забезпечити високу якість локалізації, яка відповідає технічним вимогам продукту та потребам україномовних розробників.

ВИСНОВКИ ДО РОЗДІЛУ 1

У результаті теоретичного дослідження, проведеного у першому розділі, було проаналізовано ключові аспекти перекладу технічної документації та специфіку функціонування термінології в середовищі open-source.

З’ясовано, що термін як центральна одиниця науково-технічного тексту є складним об’єктом лінгвістичного аналізу, що вимагає однозначності, точності та системності. У контексті ІТ-сфери термін не лише номінує поняття, а й забезпечує ефективну комунікацію між розробниками та користувачами програмного продукту. Встановлено, що для документації ігрових рушіїв, зокрема Godot Engine, характерна висока концентрація вузькоспеціалізованої лексики, яка поєднує в собі елементи програмування, комп'ютерної графіки та математики.

Особливу увагу приділено поняттю термінологічної консистентності, яка визначена як обов'язкова вимога до якісного перекладу. Консистентність забезпечує єдність сприйняття тексту та запобігає дезорієнтації користувача. Доведено, що в проєктах із відкритим вихідним кодом досягнення такої єдності ускладнюється через специфіку краудсорсингу. Хоча залучення великої кількості волонтерів пришвидшує процес локалізації, воно водночас створює ризики стилістичного розриву та термінологічної розрізненості.

Висвітлено роль допоміжних інструментів, таких як глосарії та стайл-гайди, що є критично важливими для регулювання перекладацького процесу в спільнотах. Таким чином, теоретичний фундамент дослідження підтверджує, що переклад open-source документації потребує не лише лінгвістичної компетенції, а й суворого дотримання термінологічних стандартів в межах динамічної та багаторівневої системи професійної комунікації. Це створює підґрунтя для подальшого практичного аналізу стратегій перекладу на прикладі документації Godot Engine.

РОЗДІЛ 2. АНАЛІЗ ПЕРЕКЛАДУ ДОКУМЕНТАЦІЇ GODOT ENGINE

2.1. Огляд Godot Engine та його документації

Godot Engine є сучасним кросплатформним ігровим рушієм із відкритим вихідним кодом, що станом на сьогодні є одним із лідерів у сегменті вільних технологій для розробки 2D та 3D ігор. Його архітектура, заснована на принципах модульності та відкритості, сприяє широкому використанню серед незалежних (*інді*) розробників, професійних студій та в освітніх цілях. Важливим аспектом, що визначає унікальність екосистеми *Godot*, є наявність розгалуженої комплексної документації, яка не лише виконує довідкову функцію, а й слугує основним інструментом онбордингу нових користувачів [30]. Як зазначає Б. Есселінк, у сучасному програмному забезпеченні документація перестає бути лише додатковим описом, а стає невід'ємною частиною продукту, що безпосередньо впливає на його функціональну придатність та успішність [28, с. 8]. Історичний розвиток рушія розпочався у 2007 році, коли аргентинські програмісти Хуан Лінієтскі (*Juan Linietsky*) та Арієль Манзур (*Ariel Manzur*) створили його як внутрішній інструмент для власних проєктів. Поворотним моментом в еволюції продукту став 2014 рік, коли рушій було випущено під ліцензією *MIT*, що забезпечило його повну відкритість та відсутність будь-яких ліцензійних відрахувань (*royalties*) для комерційних розробок [30].

Філософія *Godot* базується на унікальній сцена-орієнтованій архітектурі, де кожен ігровий об'єкт представлений у вигляді вузла (*Node*), що комбінується у складні ієрархічні структури - дерева сцен (*SceneTree*). Ця об'єктно-орієнтована структура породжує специфічний лексичний пласт, де загальновживані слова набувають нових, суто технічних значень [19, с. 144]. Наприклад, поняття *parent* (батьківський вузол) та *child* (дочірній вузол) у документації *Godot* описують не родинні зв'язки, а ієрархічні відносини підпорядкування в дереві сцени, тоді як термін *scene* виходить далеко за межі традиційного театрального чи кінематографічного значення, стаючи

універсальною одиницею організації даних [30]. Розуміння цієї термінологічної опозиції є критичним для перекладача, оскільки ці назви є не просто лексемами, а фундаментальними концептами логіки рушія [2, с. 54; 19, с. 145]. Тексти, що описують функціонал рушія, часто пишуться безпосередньо розробниками ядра або досвідченими контриб'юторами (учасниками спільноти), для яких пріоритетом є технічна точність, а не лінгвістична довершеність. Це створює для перекладача додаткові виклики у вигляді професійного жаргону та специфічного синтаксису, що потребує не лише мовної інтуїції, а й глибокого когнітивного розуміння архітектури продукту [9, с. 15; 38, с. 12].

Крім того, важливою рисою *Godot Engine* є його власна скриптова мова *GDScript*, синтаксис якої нагадує *Python* [30]. Документація рясніє прикладами коду, назвами вбудованих функцій, сигналів та констант, які є невід'ємною частиною тексту [30]. Перекладач постійно постає перед дилемою розмежування мовного простору: де закінчується описовий текст, який необхідно перекладати українською, і де починається зарезервована лексика мови програмування, яку слід залишати англійською [28, с. 156; 34, с. 102]. Будь-яка помилка в цьому розмежуванні, наприклад, випадковий переклад назви методу *«get_node()»* як *«отримати_вузол()»*, робить інструкцію технічно невірною та непридатною для використання, що підкреслює критичну важливість розуміння перекладачем не лише мови, а й самої логіки роботи рушія [9, с. 15; 38, с. 12].

Документація *Godot Engine* має чітко детерміновану структуру, що реалізована за принципом *«Docs as Code»* із використанням системи генерації текстів *Sphinx* та формату *reStructuredText* [30; 32, с. 118]. Вона включає чотири основні блоки:

1. «Початок роботи» (*Getting Started*): інструкції зі встановлення та базовий скриптинг.
2. «Посібник» (*Manual*): глибокий аналіз інструментарію (фізика, анімація, світло).

3. «Довідник класів» (*Class Reference*): технічні описи *API*, методів та сигналів.
4. «Спільнота та ресурси»: посилання на зовнішні платформи та форуми.

Така структура вимагає від перекладача диференційованого підходу до стилістики: від навчально-педагогічного в посібниках до суворо технічного в довіднику класів [9, с. 570; 45, с. 88]. Специфікою локалізації *Godot* є необхідність збереження оригінальних назв класів (наприклад, *RigidBody2D* або *Area2D*) для забезпечення коректної роботи пошуку та відповідності синтаксису коду, тоді як описові частини потребують адаптації та пояснення [28, с. 164; 35]. Наприклад, поняття *Tween* часто транслітерується як «твін», проте вимагає коментаря щодо його призначення для інтерполяції властивостей об’єктів [19, с. 146; 30].

Кожен блок вартий окремої уваги:

Розділ «Початок роботи» (*Getting Started*) становить фундамент документації та виконує роль вхідного шлюзу для нових користувачів, спрямовуючи їх через процес первинної адаптації до середовища розробки [30]. Його структура вибудована за методикою поступового занурення (*onboarding*), що включає не лише технічні інструкції зі встановлення *Godot* на різні операційні системи, а й серію покрокових інтерактивних проєктів, таких як *«Ваша перша 2D-гра»* та *«Ваша перша 3D-гра»*. З лінгвістичного погляду цей блок характеризується відносно помірною термінологічною щільністю та використанням педагогічних стратегій пояснення ключових концептів, зокрема архітектури *«вузол-сцена»* та основ мови *GDScript* [9, с. 544]. Для перекладача цей розділ є стратегічно важливим, оскільки саме тут формується базовий термінологічний апарат користувача. Окрім лінгвістичної точності, цей блок потребує ретельної локалізації візуальних елементів (скриншотів та схем), щоб забезпечити відповідність між текстом посібника та інтерфейсом редактора, що вимагає особливої уваги до прагматики перекладу [28, с. 345].

Розділ «Посібник» (*Manual*), на відміну від вступних матеріалів, являє собою розгалужений масив спеціалізованих знань, що детально описує всі функціональні підсистеми рушія в їхній взаємодії [30]. Ця частина документації охоплює критично широкий спектр технічних доменів: від фізики твердих тіл і скелетної анімації до систем обробки звуку, мережевої архітектури та шейдерного програмування. Кожна стаття *«Посібника»* фокусується на глибокому аналізі конкретного функціоналу, поєднуючи теоретичний опис механізмів рушія з практичними рекомендаціями щодо оптимізації та використання «кращих практик» (*best practices*) розробки. Для перекладача цей блок становить значну складність з погляду термінологічної консистентності, адже він вимагає високої інтердисциплінарної компетентності: фахівець має вільно оперувати поняттями комп'ютерної графіки, лінійної алгебри та інженерії програмного забезпечення одночасно [45, с. 112]. Величезний обсяг *«Посібника»* в умовах краудсорсингового перекладу часто зумовлює стилістичну неоднорідність та виникнення термінологічних розбіжностей між розділами [34, с. 88]. Це робить даний масив текстів пріоритетним об'єктом для системного редагування та уніфікації з метою забезпечення цілісності всієї терміносистеми проєкту [4, с. 112].

Розділ «Довідник класів» (*Class Reference*) є найбільш формалізованим, технічно насиченим та об'ємним сегментом документації, оскільки він являє собою вичерпний опис прикладного програмного інтерфейсу (*API*) рушія [30]. Особливістю «Довідника класів» є те, що він генерується безпосередньо з коментарів у вихідному коді *Godot* та *XML*-файлів репозиторію, що підкреслює його нерозривний зв'язок з архітектурою продукту в межах парадигми «Docs as Code» [32, с. 118]. Блок містить суворо структуровану інформацію про ієрархію успадкування, методи, властивості, сигнали та константи кожного класу - від базового *Object* до вузькоспеціалізованих систем рендерингу чи мережевої взаємодії.

Головний перекладацький виклик тут полягає у необхідності дотримання граничної лаконічності та термінологічної «стерильності» [9, с. 570]. Оскільки ідентифікатори класів, методів та властивостей не підлягають перекладу для збереження відповідності синтаксису мов програмування, перекладач має зосередитися на дескриптивній частині [28, с. 164]. Тут критично важливо забезпечити повну семантичну відповідність між англомовним описом функції та українським відповідником, щоб уникнути неоднозначності при використанні *API*. Велика кількість повторюваних структурних елементів та високий ступінь уніфікації описів роблять цей блок ідеальним для застосування *CAT*-інструментів та систем пам'яті перекладів (*Translation Memory*), проте це водночас вимагає від перекладача безкомпромісного дотримання затвердженого глосарія [24, с. 102; 50, с. 341].

Розділ «Спільнота та ресурси» (*Community and Resources*) виконує роль інтеграційного хаба, що поєднує статичну документацію з динамічним середовищем соціальної взаємодії розробників [30]. Цей блок містить посилання на офіційні форуми, сервери *Discord*, платформи запитань і відповідей (*Q&A*) та системи відстеження помилок, формуючи цілісну екосистему підтримки користувачів. З погляду локалізації цей розділ вимагає не лише перекладу навігаційних елементів, а й глибокого розуміння соціокультурного та правового контексту *open-source* спільноти [38, с. 14]. Важливою частиною блоку є матеріали, присвячені принципам внеску в проєкт (*Contributing*), де описуються процеси звітування про помилки (*issue reporting*) та подання пропозицій щодо вдосконалення рушія. Оскільки саме через ці ресурси координується робота волонтерів, зокрема і перекладацька діяльність на платформі *Weblate*, цей розділ документації є живим свідченням колективного характеру розробки *Godot* [49]. Локалізація цього блоку має на меті не просто передачу технічної інформації, а й зниження мовного бар’єра для залучення нових українських контриб’юторів до глобального процесу вдосконалення продукту, що підкреслює стратегічну роль перекладу в розбудові вітчизняного *GameDev*-середовища [34, с. 56; 43].

Загалом, процес локалізації документації *Godot Engine* організовано на засадах відкритості та взаємодопомоги, що є типовим для великих *open-source* проєктів [38, с. 11]. Головним середовищем для координації зусиль спільноти виступає спеціалізована веб-платформа *Weblate*, яка синхронізується з репозиторіями *GitHub* і дозволяє сотням волонтерів працювати над перекладом одночасно [34, с. 24; 49]. Саме з цієї платформи було вивантажено матеріал для практичного дослідження - корпус текстів у форматі *CSV* (файл *godot-engine-godot-docs-uk.csv*), що містить масив пар «англомовний оригінал - український переклад». Цей набір даних є типовим зрізом поточного стану української локалізації, фіксуючи як затверджені, стабільні переклади, так і спірні або незавершені варіанти, що перебувають у процесі активного обговорення спільнотою [5, с. 34; 34, с. 102].

Аналіз даного масиву даних унаочнює одну з ключових проблем перекладу технічної документації сучасних ІТ-проєктів - дефрагментацію або уривчастість контексту [34, с. 88]. У середовищі *Weblate* та похідних експортованих файлах текст представлений у формі ізольованих сегментів (окремих речень, заголовків або абзаців), що вилучені із загальної логічної та візуальної структури сторінки [49]. Це явище зумовлює виникнення ефекту так званої «контекстуальної сліпоти», коли лінгвістичне рішення щодо вибору терміна приймається без опори на візуальний ряд чи архітектуру розділу, що суттєво підвищує ризик термінологічної неузгодженості [38, с. 14]. Додатковим ускладнюючим фактором, що чітко простежується в досліджуваному корпусі перекладу, є висока насиченість тексту елементами розмітки *reStructuredText* [32, с. 118]. Перекладач змушений оперувати не лише семантичними одиницями, а й службовими символами: подвійними зворотними апострофами для виділення коду, специфічними конструкціями для перехресних посилань (наприклад, *:ref:*) та директивами форматування. Будь-яка некоректна модифікація або випадкове видалення цих індикаторів у процесі перекладу призводить до критичних помилок при подальшій

компіляції документації, що робить переклад технічно непридатним для інтеграції в систему [24, с. 116].

Однією з головних проблем локалізації *open-source* документації залишається висока динамічність оновлень продукту [28, с. 14; 34, с. 29]. Вихід глобальної версії *Godot 4.0* спричинив масштабні зміни в архітектурі *API*, що безпосередньо вплинуло на термінологічну систему всього масиву текстів [30]. Зокрема, перейменування базового класу *Spatial* у *Node3D* вимагало тотальної ревізії всіх існуючих перекладів та оновлення баз пам'яті перекладів для збереження консистентності викладу [4, с. 113; 30]. Окрім суто технічних модифікацій, важливою складовою процесу є лінгвокультурна адаптація: приклади, що містять англомовні ідіоми або специфічні культурні реалії, мають замінюватися на нейтральні українські аналоги без втрати технічної точності та функціональної релевантності для розробника [12, с. 142; 39, с. 160].

Таким чином, Godot Engine являє собою унікальну екосистему з чітко структурованою документацією, яка охоплює всі рівні взаємодії користувача з рушієм - від перших кроків до глибокого технічного аналізу API. Розуміння архітектури цих інформаційних блоків та їхнього функціонального призначення є необхідною умовою для вибору адекватних перекладацьких стратегій та забезпечення високої якості локалізації продукту.

2.2. Структурно-семантичні особливості документації Godot Engine

Аналіз оригінальної документації *Godot Engine* виявляє низку специфічних структурно-семантичних характеристик, які зумовлені її технічною природою та відкритим способом створення [30]. Структурно текст документації є гібридним утворенням, де лінійна оповідна частина (навчальні посібники) тісно переплітається з нелінійними фрагментами коду та автоматично згенерованими описами *API* [30; 32, с. 116]. Як зазначає Б. Есселінк, структура технічного тексту в ПЗ детермінована його

функціональністю: кожен сегмент має бути максимально автономним, щоб користувач міг отримати вичерпну інформацію без необхідності читання всього масиву тексту [28, с. 22]. Це призводить до високої рекурсивності та повторюваності термінологічних конструкцій, що є значним викликом для збереження стилістичної гнучкості при перекладі.

Текст, що надходить до перекладача, деконструйовано на ізольовані сегменти - «стрічки» (*strings*), які можуть варіюватися від розгорнутих абзаців теоретичного характеру до лаконічних, еліптичних конструкцій інтерфейсу, як-от назви властивостей, кнопок чи пунктів меню [34, с. 29; 49]. Семантична специфіка оригіналу характеризується надзвичайно високою термінологічною щільністю: у текстах *Godot Docs* терміни становлять до 40% лексичного складу речення, що вимагає від перекладача особливої уваги до відтворення синтаксичних зв'язків та збереження логічної цілісності повідомлення [9, с. 210]. Така атомізація тексту в поєднанні з його технічною насиченістю змушує лінгвіста постійно вдаватися до стратегії термінологічного пошуку та верифікації значень через зовнішні ресурси [28, с. 154]. Високий показник термінологічної щільності у документації *Godot Engine* не є випадковим, а зумовлений жанровою приналежністю тексту до науково-технічного дискурсу, де термін виступає головним носієм пресупозитивної інформації [9, с. 15]. Застосування методу суцільної вибірки до розділу *Class Reference* підтверджує, що більшість синтагм побудовані навколо термінологічних ядер, де на кожні 10 лексем припадає в середньому 4 спеціалізовані одиниці [30]. Як підкреслює Р. Штольце, така насиченість фаховою лексикою створює «термінологічне поле», яке вимагає від перекладача не просто лінгвістичної, а предметної компетенції для ідентифікації меж терміна в багатокомпонентних конструкціях [45, с. 84].

Проте найбільш проблемним аспектом є феномен, який у перекладознавстві отримав назву «Developer English» (англійська мова розробників) [23, с. 54]. Оскільки документацію *open-source* проєктів пишуть переважно програмісти, а не професійні технічні письменники, вихідний

текст часто демонструє відхилення від літературної норми: надмірну стисліть, специфічний сленг та порушення логічних зв'язків [9, с. 142]. За визначенням М. Берналь-Меріно, такі тексти часто страждають на «контекстуальну сліпоту», коли автор пропускає важливі пояснення, вважаючи їх самоочевидними для фахівця [23, с. 45]. Наприклад, англійське дієслово *Run* в ізольованому вигляді може інтерпретуватися як команда запуску ігрової сцени (*«Запустити»*), як іменник, що позначає процес виконання (*«Виконання»*), або навіть як частина ідіоматичного виразу [30]. Ця структурна особливість змушує перекладача постійно звертатися до екстралінгвістичних джерел інформації - вихідного коду рушія або запущеного редактора - для верифікації обраного значення, перетворюючи процес перекладу на складну дослідницьку діяльність [28, с. 154].

Однією з головних семантичних вад «Developer English» у документації *Godot Engine* є синтаксична компресія - нанизування іменників у ролі означень (наприклад, *«Viewport container input event handling»*) [9, с. 145; 30]. Така структура породжує амфіболію (двозначність), де перекладач без занурення в код не може визначити, до якого саме слова відноситься конкретне означення [23, с. 45]. Це підтверджує тезу В. Карабана про те, що багатозначність у технічному тексті є прямим наслідком порушення логіки побудови вихідного повідомлення або надмірного стиснення його структури [9, с. 210]. У випадку *Godot Engine* це часто виявляється в описах методів, де назва методу (дієслово) вживається як іменник без відповідних артиклів чи маркерів, що дезорієнтує лінгвіста та вимагає додаткової верифікації через програмне середовище [28, с. 154; 30].

Важливим структурним чинником, що суттєво ускладнює процес локалізації, є глибока інтеграція синтаксису мови розмітки *reStructuredText* (*reST*) безпосередньо у семантичну структуру речення. Вихідні сегменти насичені специфічними службовими символами, які виконують функції форматування, типізації або навігації: подвійні зворотні апострофи для маркування літералів коду, зірочки для емфази, а також складні директиви

для перехресних посилань типу *:ref:* або *:doc:*. У цьому аспекті роль перекладача виходить за межі суто лінгвістичної трансформації та набуває ознак технічного редагування, оскільки він зобов’язаний безпомилково відтворювати синтаксичну архітектуру оригіналу для забезпечення коректної збірки проєкту. Особливу небезпеку становлять конструкції з прихованими гіперпосиланнями, де дескриптор посилання відокремлено від цільового *URI* кутовими дужками. Така насиченість нетекстовими компонентами значно підвищує когнітивне навантаження на фахівця, змушуючи його постійно перемикати увагу між змістом повідомлення та його технічною оболонкою. Як зазначає Л. Боукер, робота з тегами та службовою розміткою в системах автоматизованого перекладу вимагає від лінгвіста особливої «технологічної пильності», оскільки будь-яка синтаксична деформація робить перекладений матеріал функціонально непридатним [24, с. 116].

На семантичному рівні документація Godot Engine репрезентує унікальну онтологію віртуального простору, яка базується на специфічній метафоричній моделі. Центральним елементом цієї системи є концепт «Дерева сцени» (Scene Tree), який екстраполює біологічні та генеалогічні поняття на ієрархічну архітектуру програмних об'єктів. Терміни *parent* (батьківський вузол) та *child* (дочірній вузол) у цьому контексті зазнають повної десемантизації, втрачаючи первинне антропоморфне значення спорідненості та набуваючи суто топологічного змісту. Це вимагає від перекладача свідомої відмови від побутових асоціацій на користь нормативної технічної термінології: наприклад, лексема *inheritance* має позначати не «спадщину», а технічний процес «успадкування» властивостей класу. Семантичне поле документації також охоплює вузькоспеціалізовані концепти розробки ігор, такі як *instancing* (створення екземпляра сцени за шаблоном), *baking* (попередній розрахунок даних освітлення чи навігації, що традиційно фіксується як «запікання») та *viewport* (область перегляду). Оскільки така лексика часто не має прямих еквівалентів у загальнотехнічних

словниках, перекладач змушений вдаватися до калькування або термінологічної деривації для створення адекватних неологізмів[19, с. 145].

Суттєвим викликом є синтаксична організація англійського оригіналу, що тяжіє до граничної компресії інформації. Для технічної мови *Godot* характерне домінування багатокомпонентних атрибутивних ланцюжків (noun clusters), де низка іменників у препозиції виконує функцію розгорнутого означення (наприклад: *Physics Body State Check*) [9, с. 145]. Дослівне відтворення таких конструкцій в українській мові є неможливим через типологічні відмінності граматичного ладу, що потребує застосування глибинних синтаксичних трансформацій - розгортання стислих англійських фраз у логічно послідовні українські конструкції з використанням родового відмінка або прийменникових зв'язків (наприклад, «Перевірка стану фізичного тіла») [9, с. 210; 30]. Додаткову складність становить репрезентація пасивного стану, притаманного англійській технічній прозі. Задля збереження динаміки та природності звучання українського перекладу фахівець має замінювати пасивні структури на безособові форми на -но/-то або активні конструкції, що відповідає функціональним нормам українського наукового стилю [13, с. 112]. Це дозволяє уникнути штучної «важковаговості» тексту та зробити його більш зрозумілим для кінцевого користувача - розробника ігор.

Отже, структурно-семантичний аналіз оригіналу Godot Docs свідчить, що перекладач працює з дефектним у лінгвістичному плані текстом, який потребує глибокої передперекладацької підготовки. Проблема «Developer English» робить неможливим використання стратегії дослівного перекладу, натомість висуваючи на перший план потребу в термінологічній верифікації та логічній адаптації тексту для кінцевого споживача.

2.3. Вплив якості вихідного тексту на процес перекладу

Якість локалізованого продукту в технічному дискурсі визначена не лише рівнем професіоналізму перекладача, а й лінгвістичною валідністю

вихідного тексту. Встановлено, що у контексті документації *Godot Engine*, як типового *open-source* проєкту, актуалізується специфічна проблема, яку в сучасному перекладознавстві кваліфікують як феномен «Developer English» (англійська мова розробників), що вже згадувався раніше в роботі у контексті «контекстуальної сліпоти» [23, с. 45]. Згідно з Б. Есселінком, технічна документація програмного забезпечення часто створюється безпосередньо інженерами-розробниками, які пріоритезують технічну фактологію, нехтуючи стилістичними нормами та правилами ясності викладу [28, с. 14]. Це породжує специфічний стиль викладу, де лаконічність межує з недостатністю контексту, що створює додаткові бар'єри при декодуванні інформації перекладачем.

Оскільки ядро проєкту та його супровідні матеріали формуються силами децентралізованої міжнародної спільноти, значна частина оригінальних текстів генерується контриб’юторами, для яких англійська мова виконує функцію лінгва франка (*English as a Lingua Franca* - ELF). Як зазначає Мінако О’Хаган, у середовищі масового відкритого перекладу це призводить до появи текстів, написаних фахівцями з глибокою технічною експертизою, проте позбавленими навичок професійного технічного письма (*technical writing*) [38, с. 14]. Така специфіка породжує синтаксичну та лексичну інтерференцію в оригіналі, що створює додаткові когнітивні бар’єри для адекватної інтерпретації змісту перекладачем та вимагає від нього не лише мовної, а й глибокої технічної дедукції [34, с. 102; 38, с. 17].

Розвиваючи тезу про феномен «Developer English», варто детальніше проаналізувати його синтаксичний вимір, зокрема проблему надмірної компресії та еліпсису. Встановлено, що розробники часто прагнуть лаконічності, притаманної коду, що призводить до опущення важливих логічних ланок. Як зазначає В. І. Карабан, така «телеграфна» стилістика часто позбавляє речення підметів або вказівок на об’єкт дії [9, с. 210]. Показовим прикладом є фраза *«Returns true if valid»* замість повної конструкції *«The function returns a boolean 'true' value if the input data is valid»*.

Для перекладача така редукція створює ситуацію семантичної невизначеності: фахівець змушений самостійно реконструювати логіку висловлювання. Згідно з М. Хіменесом-Креспо, в умовах відсутності візуального контексту на платформах типу *Weblate*, така необхідність когнітивного «добудовування» змісту суттєво підвищує ризик помилки, оскільки перекладач може невірно визначити суб'єкт дії [34, с. 102].

Наступним дестабілізуючим фактором визначено внутрішню термінологічну нестабільність безпосередньо англомовного оригіналу. Аналізом вихідних текстів *Godot Engine* було встановлено, що автори-розробники схильні до використання синонімічних рядів для позначення однієї і тієї ж сутності навіть у межах одного структурного розділу [30]. Зокрема, зафіксовано варіативне іменування об’єкта як *node, object, entity* або *element* залежно від авторства конкретного фрагмента тексту. Така варіативність порушує фундаментальний принцип термінознавства - принцип однозначності (моносемії), згідно з яким одному поняттю має відповідати лише один затверджений термін [4, с. 112]. Як зазначають Сью Еллен Райт та Герхард Будін, відсутність стандартизації на етапі створення технічного контенту неминуче призводить до розмивання понятійного апарату [50, с. 341]. Ця номенклатурна невизначеність автоматично екстраполюється на процес перекладу: перекладач-волонтер, намагаючись дотриматися принципу вірності оригіналу, механічно відтворює цю синонімію засобами української мови, що призводить до поглиблення «термінологічного хаосу». У цьому контексті, як стверджує Тетяна Горностай, проблема інконсистентності перекладу кваліфікується як пряма похідна від відсутності уніфікації у вихідному тексті [4, с. 114].

Окремим аспектом дослідження визначено проблему міжмовної лінгвістичної інтерференції, іманентно присутньої у вихідному тексті. Оскільки демографічний склад спільноти контриб’юторів *Godot* охоплює носіїв різноманітних мовних груп (романської, германської, слов’янської), продукований ними англомовний контент часто містить інтерференційні

нашарування та кальковані синтаксичні структури з рідних мов авторів. Це явище проявляється у порушенні фіксованого порядку слів, помилковому прийменниковому керуванні та використанні лексем, що стають «хибними друзями перекладача» вже на рівні англійського узусу [12, с. 145]. Для українського перекладача це створює подвійний когнітивний бар'єр. Як зазначає Ентоні Пім, у роботі з інтернаціоналізованим контентом фахівець змушений спершу здійснювати «декодування» або реконструкцію первинної авторської інтенції, продираючись крізь ненормативну граматику *Lingua Franca*, і лише після етапу змістової верифікації переходити до пошуку адекватного українського відповідника [40, с. 92]. Такий двоступеневий процес значно підвищує часовитрати та ризик змістового викривлення порівняно з перекладом нормативного тексту носія мови [38, с. 17].

Додаткового аналізу потребує вживання вузькоспеціалізованого професійного жаргону та метафоричних конструкцій, семантика яких є прозорою виключно для вузького кола розробників ядра рушія. Встановлено, що документація *Godot Engine* насичена термінами на кшталт *stub*, *boilerplate*, *vanilla* або *sugar* (у значенні «синтаксичний цукор»), що ставить перекладача перед проблемою безеквівалентності [30].

Згідно з Мігелем Берналь-Меріно, подібні одиниці функціонують як «когнітивні метафори», розуміння яких вимагає глибокого занурення в контекст програмування, а не лише володіння мовою [23, с. 48]. Ситуація ускладнюється низькою якістю лексикографічного опрацювання оригіналу: якщо автор тексту не надає дефініції метафори або контекстуального уточнення, перекладач опиняється в стані «інформаційної ентропії». Як наслідок, фахівець часто вдається до стратегії буквального перекладу (калькування). Однак, як застерігає В. І. Карабан, буквалізм при відтворенні сленгу є деструктивним, оскільки він переносить зовнішню оболонку слова, але повністю втрачає його змістове навантаження, роблячи український текст абсолютно незрозумілим для кінцевого користувача [9, с. 235].

Підсумовуючи, можна констатувати, що якість вихідного тексту в проєкті *Godot Engine* виступає первинним детермінантом термінологічної інконсистентності перекладу. Специфічні вади оригіналу - від синтаксичної редукції та інтерференції до термінологічної амбівалентності - створюють надмірне когнітивне навантаження на перекладача, змушуючи його виконувати роль технічного дедуктора. Таким чином, подолання негативного впливу «Developer English» потребує від фахівця не лише лінгвістичної адаптації, а й розробки особливих стратегій верифікації та уніфікації контенту, що й буде розглянуто у наступному розділі дослідження.

2.4. Проблеми перекладу документації Godot Engine

Практичним аналізом корпусу перекладів документації *Godot Engine* було виявлено низку системних проблем, які суттєво впливають на якість локалізації та її прагматичне сприйняття кінцевим користувачем. Найбільш виразною девіацією, що простежується на всьому масиві досліджуваного матеріалу, визначено порушення термінологічної консистентності, зумовлене відсутністю централізованого нормативного глосарія та постійною ротацією волонтерського складу в проєкті [34, с. 102; 38, с. 14]. Як наслідок, один і той самий англомовний термін у різних розділах документації (а іноді й у межах однієї сторінки) отримує варіативні українські відповідники, що дезорієнтує розробника та руйнує цілісність технічного дискурсу [4, с. 113].

Одним із найбільш показових прикладів такої неузгодженості є відтворення поняття *asset*, яке є фундаментальним для сфери розробки ігрових продуктів. Встановлено, що в різних розділах документації, а подекуди й у межах одного текстового сегмента, цей термін фіксується як «асет», «ресурс», «актив» або «надбання». Варіант «асет» репрезентує пряму транслітерацію професійного жаргонізму, зрозумілу досвідченим розробникам, тоді як «ресурс» є спробою адаптації, що, однак, створює критичну термінологічну колізію з іншим базовим технічним об’єктом *Godot* - класом *Resource* [30]. Варіант «актив» тяжіє до фінансово-економічної

лексики, що виглядає контекстуально недоречно в межах опису файлової системи, а «надбання» кваліфікується як архаїзм, що не відповідає функціональному стилю сучасної IT-документації.

Така лексична варіативність дезорієнтує реципієнта, оскільки унеможливлює ідентифікацію об’єкта як єдиної сутності, що прямо суперечить принципу однозначності технічного терміна, обґрунтованому в працях Т. Р. Кияка [10, с. 24]. Як зазначає В. І. Карабан, подібна синонімія в технічних текстах є неприпустимою, оскільки вона руйнує логічну цілісність інструкції та перешкоджає коректному виконанню технічних завдань розробником [9, с. 248].

Аналогічну ситуацію було зафіксовано при аналізі дієслівної лексики, що описує специфічні процеси функціонування рушія. Показовим прикладом вважається термін baking (процес попереднього розрахунку даних освітлення або навігаційних сіток). Встановлено, що в українській локалізації конкурентними є три основні варіанти: «запікання» (пряма калька, поширена в індустрії), «випікання» та дескриптивний переклад «попередній розрахунок». Попри те, що для пересічного носія мови лексема «запікання» може сприйматися як стилістично знижена або суто кулінарна, у межах професійного дискурсу комп'ютерної графіки вона набула статусу сталого терміна [23, с. 48]. Як зазначає Мігель Берналь-Меріно, ігрова локалізація часто оперує специфічними метафорами, які, попри свою позірну неформальність, є ключовими для когнітивної ідентифікації функцій досвідченими користувачами [23, с. 52]. Намагання окремих волонтерів-перекладачів уникнути цього професіоналізму шляхом використання розлогих описових конструкцій призводить до втрати лаконічності, що є критичним для технічного стилю. Крім того, це суттєво ускладнює семантичне співставлення тексту документації з англомовним інтерфейсом редактора, де відповідна функція позначена ідентифікатором Bake [30]. У цьому контексті важливо зважити на принцип, сформульований Бертом Есселінком, згідно з яким термінологічна єдність між графічним інтерфейсом

користувача (GUI) та супровідною документацією є першочерговою вимогою якості локалізації [28, с. 67]. Будь-яка розбіжність між назвою кнопки в програмі та її описом у посібнику кваліфікується як груба помилка, що перешкоджає ефективному використанню програмного продукту.

Окрему групу проблем було виокремлено у межах явища міжмовної омонімії та полісемії, що за умов дефіциту контексту призводить до прийняття помилкових перекладацьких рішень. Встановлено, що термін *key* у документації *Godot* може репрезентувати кілька денотатів: «клавіша» (елемент пристрою введення), «ключ» (ідентифікатор у структурі даних) або «ключ/ключовий кадр» (параметр у системах анімації) [30]. Аналізом файлів локалізації підтверджено випадки, коли словосполучення *animation key* було помилково відтворено як «клавіша анімації» замість нормативного «ключ анімації», що повністю деформує технічний зміст інструкції.

Як наголошує Мігель Хіменес-Креспо, стратегічною вразливістю хмарних платформ локалізації (таких як *Weblate*) є «атомізація» тексту, коли перекладач бачить лише ізольований сегмент без зв’язку з попередніми та наступними реченнями [34, с. 102]. Це фактично позбавляє фахівця необхідного семантичного оточення, провокуючи вибір помилкових значень для багатозначних слів. Аналогічну складність було зафіксовано при відтворенні терміна *remote*, який у контексті налагодження гри (*Remote Debugger*) коректно перекладається як «віддалений», проте в описі вузла *RemoteTransform* позначає об’єкт, що здійснює дистанційне керування трансформаціями іншого вузла [30]. У зв’язку з цим Лінн Боукер зауважує, що за відсутності прямого доступу до візуального контексту перекладачі схильні механічно обирати найбільш частотний словниковий відповідник [24, с. 110]. У технічних текстах такий підхід неминуче призводить до смислових розбіжностей, оскільки суть об’єкта часто вимагає вибору вузькоспеціалізованого значення. Таким чином, механічна трансляція без аналізу функціональної природи вузла визначена як одна з найбільш типових помилок у досліджуваному матеріалі.

Суттєвим викликом для фахівця з локалізації визначено забезпечення граматичної узгодженості текстів, що містять змінні підстановки (*placeholders*). У повідомленнях на кшталт *«%s was imported successfully»* змінна *%s* може бути замінена назвою файлу (чоловічий рід), сцени (жіночий рід) або зображення (середній рід). Оскільки в англійській мові категорія граматичного роду для неживих назв відсутня, оригінальна конструкція є універсальною. Проте в українській мові переклад *«%s був імпортований успішно»* кваліфікується як граматично некоректний у випадку підстановки іменника жіночого чи середнього роду.

Як зауважує Мона Бейкер, граматичні категорії (зокрема рід та число) є одними з найбільш складних для відтворення при перекладі між мовами з різним ступенем морфологічної складності [21, с. 95]. У таких випадках перекладач змушений вдаватися до стратегії «граматичної адаптації», що передбачає пошук таких структур у цільовій мові, які б дозволили нівелювати розбіжності в категоріях роду. Це зумовлює необхідність використання безособових конструкцій (наприклад, «%s успішно імпортовано») або пасивних форм, що, однак, не завжди дозволяє зберегти природність звучання тексту. Згідно з позицією Крістіани Норд, функціональна придатність перекладу залежить від його відповідності нормам цільової мови [37, с. 72]. Нехтування проблемою узгодження в технічних повідомленнях призводить до появи системних мовних помилок, що суттєво знижує рівень довіри реципієнта до якості програмного продукту. Таким чином, розробка універсальних синтаксичних шаблонів визначена як необхідний етап підготовки глосарія для забезпечення стабільності проєкту.

Окремої уваги заслуговує проблема надмірних пуристичних тенденцій, які в процесі локалізації періодично вступають у конфлікт із усталеним фаховим узусом. Намагання замінити загальноприйняті терміни, такі як *render* («рендер»), *shader* («шейдер») або *instance* («інстанс»/«екземпляр»), питомими українськими відповідниками («відмальовування», «затінювач», «взірець») часто наштовхується на опір професійної спільноти. Встановлено,

що лексема «затінювач», попри свою етимологічну прозорість, не охоплює всього семантичного спектра сучасного поняття *shader*, яке в архітектурі *Godot* відповідає не лише за розрахунок тіней, а й за будь-які маніпуляції з геометрією та кольором пікселів [30].

Як зазначає Ентоні Пім, у процесі локалізації цифрових продуктів часто виникає напруження між лінгвістичною нормою та комунікативною ефективністю. Надмірна адаптація термінології може призвести до «відчуження» користувача, який звик до глобальної англомовної номенклатури, що фактично нівелює прагматичну мету перекладу [40, с. 112]. У межах ігрової індустрії ця дилема постає особливо гостро. Мігель Берналь-Меріно підкреслює, що технічні терміни в *GameDev* функціонують як «когнітивні ікони»; їхня радикальна заміна на національні еквіваленти може розірвати асоціативний зв'язок із функціоналом рушія, особливо якщо ці еквіваленти мають застаріле або занадто вузьке значення [23, с. 50]. Таким чином, встановлено, що пошук балансу між мовною чистотою та професійною ефективністю залишається одним із найбільш дискусійних аспектів локалізації документації *Godot Engine*.

Вагомим фактором, що ускладнює процес адаптації, визначено проблему синтаксичної асиметрії між англійською та українською мовами. Це особливо гостро проявляється при відтворенні багатокомпонентних термінологічних словосполучень. Встановлено, що англійська технічна мова тяжіє до активного використання атрибутивних груп (noun clusters), де іменники нанизуються один на одний без зміни морфологічної форми, створюючи складні семантичні єдності (наприклад: *Audio Stream Player 3D settings*).

Для перекладача це становить нетривіальну задачу, оскільки, як зазначає В. І. Карабан, пряме калькування таких структур в українській мові є неможливим через відмінності в граматичному ладі. Розгортання подібних конструкцій вимагає використання довгих ланцюжків іменників у родовому відмінку (наприклад, «налаштування просторового програвача аудіопотоку»)

[9, с. 145]. Такі синтаксичні побудови часто стають громіздкими, що змушує фахівця шукати компроміс між термінологічною точністю та милозвучністю (евфонією). Водночас Петер Шмітт застерігає, що надмірна перестановка компонентів задля покращення стилю може призвести до зміщення смислових акцентів [46, с. 88]. У контексті локалізації це створює критичний ризик, оскільки ускладнює пошук відповідного пункту в ієрархічному меню, де користувач очікує прямої відповідності між документацією та назвою елемента керування.

Вагоме місце у переліку перекладацьких проблем посідає колізія між номенклатурою класів рушія та загальновживаною лексикою. Встановлено, що *Godot Engine* використовує прості англійські лексеми для номінації своїх базових сутностей: *Path, Curve, Animation, Camera*. У цьому контексті графічна репрезентація (капіталізація) виступає єдиним семантичним диференціатором: лексема з великої літери позначає конкретний програмний об’єкт (клас), тоді як з малої - загальне поняття. Як зазначає Берт Есселінк, у технічній документації до програмного забезпечення регістр літер виконує не граматичну, а морфологічно-розрізнювальну функцію, фактично перетворюючи слово на частину коду [28, с. 156]. Проблема актуалізується у випадках, коли перекладач ігнорує цю дистинкцію або коли в українській мові складно дібрати відповідник, який би функціонував органічно і як назва класу, і як загальновживане слово.

Яскравим прикладом такої асиметрії є базовий клас *Control*, від якого успадковуються всі елементи інтерфейсу (*GUI*). Його прямий переклад як «Контроль» кваліфікується В. І. Карабаном як типова помилка «хибних друзів перекладача», оскільки в українській мові це слово означає нагляд або перевірку (*monitoring/inspection*), а не керування [9, с. 342]. Водночас варіант «Керування» є занадто абстрактним для позначення візуального об’єкта, а термін «Елемент керування» є надмірно громіздким для використання як назви типу даних. У результаті аналізу текстів було зафіксовано системну термінологічну флуктуацію: в одних сегментах *Control* залишають

латиницею, в інших транслітерують як «контрол», а подекуди замінюють описовими конструкціями. Це руйнує цілісність сприйняття архітектури рушія та порушує принцип уніфікації, на якому наполягають експерти *SBT Localization Team* [48, с. 12].

Суттєвою перешкодою для сприйняття цілісності матеріалу виступає стилістична еклектика в інструктивних текстах, спровокована відмінністю підходів волонтерів до відтворення категорії модальності. В оригінальних англомовних навчальних посібниках (*tutorials*) домінує пряме звернення до користувача через імператив (*Select the node, Click the button*) або використання займенника *You* [30]. Натомість в українських перекладах аналізом було зафіксовано хаотичне співіснування трьох конкурентних стратегій: безособової («Слід вибрати вузол», «Вузол вибирається»), інфінітивної («Вибрати вузол») та особистої («Виберіть вузол»).

Згідно з дослідженнями Петера Шмітта, така варіативність, коли стратегії змішуються в межах однієї статті чи навіть абзацу, призводить до втрати ритмічної та стилістичної когерентності тексту, перетворюючи його на так звану «клаптикову ковдру» [46, с. 142]. Хоча це безпосередньо не впливає на технічну валідність виконання алгоритму дій, подібна різнорідність, на думку Крістіани Норд, суттєво знижує комунікативну якість матеріалу [37, с. 84]. У реципієнта формується враження професійної недбалості та відсутності єдиної редакційної політики, що є критичним фактором для іміджу *open-source* проєкту.

Значні труднощі також супроводжують процес відтворення вузькоспеціалізованої математичної та фізичної термінології, яка в документації *Godot Engine* вирізняється високим ступенем специфічності. Встановлено, що термінологічні одиниці, такі як *quaternion* (кватерніон), *gimbal lock* (складання рамок), *dot product* (скалярний добуток) або *look-at vector*, вимагають від перекладача не лише філологічної компетенції, а й фахових знань у галузі лінійної алгебри та обчислювальної механіки [30].

Типовою девіацією, виявленою в ході дослідження, визначено спробу перекладу цих понять за допомогою загальновживаної лексики (наприклад, відтворення терміна *Basis* як «Основа» замість нормативного математичного «Базис»). Згідно з В. І. Карабаном, така підміна понять робить науково-технічний текст функціонально непридатним для фахівця, який очікує побачити усталену номенклатуру, а не побутові синоніми [9, с. 238]. Коренем цієї проблеми є специфіка кадрового забезпечення краудсорсингових проєктів. Як зазначає А. С. Дебела, статистичний аналіз волонтерських спільнот свідчить, що до 68% учасників не мають профільної перекладацької або технічної освіти, керуючись здебільшого альтруїстичними мотивами, а не професійними навичками [15, с. 74]. Саме відсутність академічного бекграунду призводить до появи у документації «псевдотермінів», які не корелюють ні з українською науковою традицією, ні з професійним жаргоном розробників ігор.

Додатковим дестабілізуючим чинником, виявленим у ході аналізу, стала проблема десинхронізації термінології між корпусом документації та графічним інтерфейсом користувача (GUI) редактора *Godot*. За своєю функціональною природою документація виступає пояснювальним супроводом інтерфейсу, виконуючи директивну функцію: вона інструктує користувача щодо навігації та налаштування параметрів. Однак встановлено, що локалізація самого рушія та документації до нього реалізується як два паралельні, автономні процеси на платформі *Weblate*, які часто виконуються різними групами волонтерів за відсутності належної крос-координації [34, с. 102].

Це зумовлює виникнення критичних розбіжностей, коли номенклатура елементів керування в інструкції не збігається з фактичними назвами на екрані. Наприклад, було зафіксовано випадки, коли параметр *Output* в інтерфейсі редактора локалізовано як «Вивід», тоді як у тексті посібника фігурують варіанти «Вихідні дані» або «Результат». Така термінологічна асиметрія провокує у реципієнта когнітивний дисонанс і суттєво гальмує

процес навчання. Згідно з Бертом Есселінком, подібна неузгодженість кваліфікується як груба помилка технічного тексту, оскільки вона порушує головний принцип юзабіліті - передбачуваність системи, руйнуючи асоціативний зв'язок між інструментом та інструкцією до нього [28, с. 68].

Лексичний пласт досліджуваного матеріалу також демонструє суттєві ризики, пов'язані з явищем міжмовної паронімії, відомим у перекладознавстві як «хибні друзі перекладача». Враховуючи домінування англійської мови в ІТ-сфері, значна частина термінологічного апарату характеризується графічною та фонетичною подібністю до українських лексем, проте має відмінну семантичну структуру. Згідно з В. І. Карабаном, ігнорування цієї розбіжності призводить до інтерференції, коли зовнішня форма слова підміняє його внутрішній зміст [9, с. 352].

Аналізом перекладів *Godot* було виявлено непоодинокі випадки механічного перенесення таких одиниць без урахування контексту. Типовим прикладом є прикметник *original*, який у словосполученні *original node* часто помилково відтворюється як «оригінальний вузол» (що в українській мові має конотацію незвичності чи унікальності), тоді як нормативним технічним відповідником у цьому контексті виступають варіанти «початковий», «вихідний» або «первинний вузол». Схожа ситуація простежується з лексемою *regular*, яка в описі геометричних фігур (*regular polygon*) означає «правильний» (рівносторонній), а не «регулярний», або словом *invalid*, яке в контексті перевірки даних означає «недійсний» чи «некоректний», а не «інвалід». Наявність подібних помилок, на думку І. В. Корунця, є маркером недостатньої фахової кваліфікації перекладача та його надмірної довіри до інтернаціональної форми слова [12, с. 148]. Це ще раз підтверджує раніше наведену статистику щодо освітнього рівня волонтерів у краудсорсингових проєктах.

Наостанок, значний масив проблем було ідентифіковано у площині відтворення абревіатур та акронімів, якими насичена технічна документація досліджуваного об'єкта. Встановлено, що термінологічна система *Godot*

Engine базується на використанні як загальнотехнічних скорочень (API, GUI, HTTP), так і вузькоспеціалізованих одиниць комп'ютерної графіки та архітектури рушія (AABB - *Axis-Aligned Bounding Box*, RID - *Resource ID*, GDScript).

У проаналізованих перекладах зафіксовано відсутність уніфікованої стратегії щодо опрацювання цих одиниць. Як зазначає В. І. Карабан, переклад скорочень є однією з найскладніших проблем науково-технічного стилю, що вимагає чіткого вибору між введенням еквівалента, транслітерацією або описовим розшифруванням [9, с. 182]. Натомість у текстах *Godot* спостерігається ситуативна варіативність: в одному розділі скорочення залишаються в оригінальній латинській графіці, в іншому - транслітеруються кирилицею (наприклад, фіксація «ГУІ» замість GUI), а в третьому - підлягають повному перекладу. Окрему методологічну складність становлять абревіатури, що набули статусу ідентифікаторів класів або типів даних (наприклад, *Transform2D*). Спроби їхньої адаптації або транслітерації кваліфікуються як критичні помилки, оскільки, згідно з Бертом Есселінком, це призводить до розриву функціонального зв'язку з програмним кодом, де використовується виключно англійська номенклатура [28, с. 164]. Водночас залишення всіх абревіатур без супровідних пояснень робить текст «герметичним» для початківців. Ця дихотомія між прагненням до комунікативної доступності та технічної прецизійності визначена як один із ключових невирішених викликів у поточній локалізації проєкту.

Таким чином, системна інконсистентність українського перекладу *Godot Engine* зумовлена специфікою волонтерської праці та відсутністю нормативного глосарія. Виявлені лінгвістичні проблеми підкреслюють критичну необхідність розробки єдиної стратегії уніфікації для забезпечення технічної точності та комунікативної якості проєкту.

ВИСНОВКИ ДО РОЗДІЛУ 2

У другому розділі було проведено детальний аналіз практичних аспектів локалізації документації ігрового рушія *Godot Engine*, що дозволило визначити ключові чинники, які впливають на формування термінологічного корпусу. Встановлено, що досліджувана документація має розгалужену структуру та охоплює декілька суміжних галузей: програмування, комп'ютерну графіку, фізику та UI-дизайн. Це зумовлює високу складність перекладу, оскільки вимагає від волонтерів не лише знання мови, а й розуміння технічної архітектури рушія.

Дослідження процесу організації перекладу показало, що використання платформи *Weblate* як основного інструменту краудсорсингової локалізації значно пришвидшує роботу, проте створює певні виклики для консистентності. Було з’ясовано, що основними механізмами контролю якості в межах спільноти Godot є автоматичні перевірки (checks), система коментування та глосарії. Однак, незважаючи на наявність цих інструментів, динамічне оновлення документації та постійний приплив нових перекладачів призводять до виникнення варіантності в передачі ключових термінів.

Аналіз вибірки текстових сегментів дозволив класифікувати основні труднощі, з якими стикаються перекладачі: відсутність сталих українських еквівалентів для неологізмів геймдеву та необхідність адаптації синтаксичних конструкцій під вимоги технічного стилю. Вивчення ролі спільноти (community-driven approach) підтвердило, що термінологічна консистентність у таких проєктах тримається на консенсусі активних учасників, що робить процес локалізації гнучким, але вразливим до суб'єктивних рішень. Отримані результати практичного аналізу стають базою для подальшої розробки рекомендацій щодо уніфікації термінології в межах open-source проєктів.

РОЗДІЛ 3. ПРАКТИЧНІ АСПЕКТИ ПЕРЕКЛАДУ OPEN-SOURCE ДОКУМЕНТАЦІЇ

3.1. Методика перекладу технічної документації

Розробку ефективної методики перекладу технічної документації для проєктів із відкритим вихідним кодом (на прикладі Godot Engine) визначено як завдання, що вимагає комплексного підходу, який виходить за межі суто лінгвістичних трансформацій. З огляду на виявлені у попередніх розділах девіації - зокрема фрагментарність тексту та термінологічну інконсистентність - методику запропоновано базувати на принципах системності, контекстуальної верифікації та стандартизації. Такий підхід відповідає сучасним вимогам інженерії локалізації, де переклад розглядається не як механічна заміна слів, а як цілеспрямована діяльність із забезпечення функціональної еквівалентності продукту [37, с. 14; 28, с. 18].

Процес перекладу в даному контексті не може розглядатися як лінійне заміщення лексичних одиниць. Він трансформується у багатоетапну процедуру, де власне текстовим операціям передує етап глибокого аналізу предметної області та визначення прагматичної мети перекладу [28, с. 22]. Ключовою вимогою стає розуміння екстралінгвістичної реальності: фахівець повинен чітко ідентифікувати, який саме програмний об'єкт або процес стоїть за конкретним терміном. Це означає, що опрацювання тексту має відбуватися паралельно з роботою у середовищі самого рушія, що дозволяє емпіричним шляхом перевірити функціональне призначення описуваних елементів. Саме такий підхід забезпечує доступність інформації та її технічну точність, що є пріоритетними критеріями якості в умовах громадського перекладу (*community translation*) [38, с. 14].

Центральним елементом запропонованої методики визначено алгоритм вибору перекладацької відповідності, що базується на чіткій ієрархії джерел [50, с. 14]. На первинному рівні здійснюється пошук усталених термінів у авторитетних технічних словниках та національних стандартах, що дозволяє

інтегрувати документацію Godot у загальноукраїнський науково-технічний дискурс [3, с. 84]. У ситуаціях, коли нормативний словниковий відповідник відсутній (явище термінологічної лакунарності) або є застарілим, методика передбачає звернення до другого рівня - професійного узусу, тобто живої мови спілкування українських розробників на профільних платформах [44, с. 112]. Однак, враховуючи насиченість цього середовища неадаптованими англіцизмами, критично важливим етапом стає селективна фільтрація варіантів. На цьому етапі фахівець зобов'язаний оцінити доцільність використання транслітерації (наприклад, «асет», «бейкінг») у зіставленні з пошуком питомого відповідника («ресурс», «запікання»). Рішення приймається на основі критеріїв прозорості змісту для початківців та технічної точності для експертів, причому пріоритет надається варіантам, що забезпечують моносемію (однозначність) інтерпретації та унеможливлюють виникнення хибних асоціацій із побутовою лексикою [10, с. 206].

Невід'ємною складовою запропонованої методики визначено забезпечення структурної валідності перекладеного тексту. Враховуючи, що документація Godot Engine генерується з використанням легковагової мови розмітки reStructuredText (reST), стратегія локалізації мусить включати обов'язковий етап технічної верифікації коду [32, с. 118]. Цей процес передбачає суворе збереження всіх службових символів, коректне відтворення синтаксису внутрішніх гіперпосилань та змінних підстановки. Методика постулює необхідність ставлення до тексту як до фрагмента програмного коду: будь-яке втручання в синтаксичну структуру речення повинно узгоджуватися з вимогами парсера документації (Sphinx). На практичному рівні це імплементується через заборону на переклад вмісту зарезервованих тегів та обмеження на зміну порядку слів у сегментах, що містять змінні, у випадках, коли це може порушити логіку динамічної підстановки даних. Такий підхід, який Берт Есселінк класифікує як «інженерію локалізації», дозволяє уникнути критичних помилок при компіляції фінальної версії документації [28, с. 14].

Інтегральним компонентом розробленої методики визначено уніфікацію термінологічного апарату та його повну синхронізацію з графічним інтерфейсом користувача (GUI). З метою нівелювання явища синонімії імплементується принцип «один термін - один відповідник» (one concept - one term), який у теорії термінознавства, за визначенням Т. Р. Кияка, є базовою умовою функціонування фахової мови [10, с. 206]. Практична реалізація цього принципу забезпечується шляхом укладання та імперативного дотримання нормативного глосарія проєкту, що фіксує єдиний валідний варіант перекладу ключових понять. Окрім того, методика постулює необхідність обов'язкової крос-верифікації номенклатури елементів керування (кнопок, меню, параметрів) у тексті документації з їхніми фактичними назвами в локалізованому інтерфейсі редактора Godot. Ця вимога корелює з принципами Берта Есселінка, який наголошує, що будь-яка розбіжність між інструкцією та програмним продуктом є критичною помилкою юзабіліті, яка дезорієнтує користувача [28, с. 154]. Встановлено, що лише такий системний підхід, який синтезує лінгвістичний аналіз, технічну верифікацію та термінологічну дисципліну, здатен гарантувати створення якісного продукту, придатного для використання як навчальний та довідковий матеріал.

Окремим структурним елементом запропонованої методики визначено прагматичну адаптацію тексту, що передбачає варіативність перекладацьких стратегій залежно від жанрової специфіки конкретного сегмента документації. Встановлено, що переклад розділу «Довідник класів» (API Reference) мусить підпорядковуватися принципам суворої формалізації та термінологічної моносемії [10, с. 207]. Використання емотивно забарвленої лексики або вільного порядку слів у цьому контексті класифікується як неприпустима девіація, здатна викривити технічний зміст специфікації. На противагу цьому, опрацювання «Онлайн-посібника» (навчальних туторіалів) вимагає зміни функціональної домінанти. Згідно з теорією Скопосу Крістіани Норд, яка наголошує на пріоритеті мети комунікації, методика у цьому

випадку рекомендує застосування стратегії доместикації - наближення тексту до мовних реалій читача [37, с. 29]. Це актуалізує використання природних синтаксичних конструкцій, прямих звертань та розгорнутих пояснень, що суттєво знижує когнітивне навантаження на користувача-початківця [39, с. 144]. Такий диференційований підхід дозволяє досягти необхідного балансу між технічною прецизійністю довідкового матеріалу та педагогічною ефективністю навчального контенту.

Завершальним етапом методики визначено процедуру забезпечення якості (*Quality Assurance*), яка в умовах відкритого проєкту базується на системі колаборативної ревізії. Враховуючи специфіку краудсорсингу, методика пропонує імплементацію двоступеневої перевірки. На першому ступені здійснюється автоматизований контроль формальних параметрів тексту за допомогою вбудованих інструментів платформи Weblate [49] (перевірка цілісності тегів розмітки reST, відповідності глосарію та відсутності порожніх сегментів). На другому ступені запроваджується механізм перехресного рецензування (*Peer Review*) досвідченими учасниками спільноти. Така модель реалізує концепцію «мудрості натовпу» (*wisdom of the crowd*), де колективна взаємодія дозволяє нівелювати суб’єктивні помилки та досягти стилістичної однорідності [34, с. 56]. Лише після проходження обох стадій верифікації переклад маркується як стабільний і підлягає інтеграції у основну гілку репозиторію. Встановлено, що такий ітеративний цикл забезпечує високу якість локалізації та її життєздатність в умовах безперервного оновлення продукту [38, с. 18].

Отже, процес перекладу кваліфікується не як одноразовий акт, а як ітеративна процедура, в межах якої текст підлягає перманентному вдосконаленню через механізми зворотного зв'язку. Такий підхід, який Мігель Хіменес-Креспо визначає як «колаборативну ревізію» (*collaborative revision*), дозволяє нівелювати ризики, пов'язані з людським фактором, та компенсувати брак професійної підготовки окремих волонтерів колективним інтелектом спільноти.

3.2. Приклади перекладу документації Godot Engine

З метою емпіричної верифікації теоретичних положень та систематизації виявлених девіацій, у цьому підрозділі проведено детальний аналіз конкретних перекладацьких рішень, ексцерпованих з корпусу документації Godot Engine. Репрезентативна вибірка зразків ілюструє найбільш типові випадки термінологічної варіативності, морфологічних неузгодженостей та когнітивних труднощів, зумовлених архітектурною специфікою ігрового рушія. Компаративний аналіз відібраних одиниць дозволяє експлікувати логіку селекції відповідників та наочно продемонструвати детермінуючий вплив технічного контексту на якість локалізації.

Перший і найбільш показовий кейс стосується відтворення базового поняття розробки ігор - терміна asset. У ході аналізу файлів локалізації (*.po / *.csv) було зафіксовано критичну розбіжність у стратегіях перекладу цього поняття, що призводить до явища «термінологічної колізії».

Розглянемо два приклади з суміжних розділів документації:

1. У сегменті, що описує файлову структуру (*"Assets are files stored in your project folder"*), перекладач обрав стратегію транслітерації: *«Асети - це файли, що зберігаються у теці вашого проєкту»*.
2. Натомість у розділі про імпорт даних (*"Importing 3D assets"*), зустрічається варіант із використанням загальновживаного відповідника: *«Імпорт 3D-ресурсів»*.

На перший погляд, синонімічна заміна «асет» на «ресурс» видається виправданою з точки зору літературної норми української мови, яка уникає надмірних запозичень. Однак у контексті архітектури Godot Engine таке рішення є хибним. Справа в тому, що в екосистемі цього рушія існує окремий фундаментальний клас Resource (скрипти, шейдери, текстури), який має чітко визначені властивості та поведінку в коді.

Коли користувач-початківець бачить у документації слово «ресурс» у значенні *asset* (будь-який файл), а згодом стикається з терміном «Ресурс» у значенні *клас Resource*, виникає когнітивний дисонанс. Він може помилково припустити, що всі файли проєкту автоматично наслідують властивості класу Resource, що не відповідає дійсності. Цей приклад наочно демонструє, як намагання уникнути англіцизму («асет») через використання побутового слова призводить до розмивання технічної точності. Це порушує базовий термінознавчий принцип однозначності відповідності між поняттям та терміном, на якому наголошує Т. Р. Кияк, зазначаючи, що в межах однієї терміносистеми синонімія є деструктивним явищем [10, с. 206].

Окремий масив перекладацьких викликів становить блок фізичної симуляції (Physics Engine), що оперує термінами класичної механіки. Аналіз показав, що саме тут спостерігається найбільша кількість випадків хибного вибору відповідника через нерозуміння фізичної сутності описуваних процесів. Особливої уваги заслуговують термінологічні пари, пов'язані з динамікою твердих тіл (*RigidBody*).

Показовим прикладом є переклад параметра damping, який відповідає за втрату енергії рухомим тілом. У сегменті *"Linear damping affects the body's movement"* («Лінійне демпфування впливає на рух тіла») зафіксовано вагання перекладачів між варіантами «загасання» та «демпфування».

Варіант «загасання» є більш зрозумілим для загального кола користувачів і часто вживається в акустиці або оптиці. Проте з точки зору теоретичної механіки, яку імплементує рушій Godot, цей термін є менш точним, оскільки *damping* у даному контексті означає штучну силу в'язкого опору середовища, що сповільнює об'єкт. Використання побутового слова «загасання» розмиває цю фізичну дефініцію, перетворюючи точний параметр на абстрактне поняття. Варіант «демпфування», хоч і є запозиченням, чітко корелює з професійною інженерною термінологією, на яку очікує розробник ігор при налаштуванні фізики.

Ще більш вираженою є проблема невиправданих запозичень, виявлена при аналізі перекладу властивості friction. У реченнях описового характеру, як-от *"Friction affects how objects slide"*, перекладачі здебільшого обирали нормативний український відповідник: *«Тертя впливає на ковзання об'єктів»*. Це рішення є абсолютно коректним і зрозумілим.

Однак у контексті інтерфейсних налаштувань фізичних матеріалів (*PhysicsMaterial*) було виявлено випадки вживання транслітерованого варіанта «Фрикція». Такий вибір є яскравим прикладом «хибної точності»: перекладач, ймовірно, вважав, що науковий термін має звучати складніше, ніж загальновживане слово. Насправді ж, у вітчизняній науковій традиції термін «фрикція» має вузьку сферу вживання (переважно у техніці та медицині) і не є повним синонімом фізичного явища тертя. Поява таких псевдотермінів у документації створює ефект неприродності мови та свідчить про відсутність фахового редагування. Згідно з В. І. Карабаном, вживання транслітерованого терміна за наявності повноцінного національного відповідника кваліфікується як стилістична помилка, що засмічує науково-технічний стиль [9, с. 235].

Цей кейс підкреслює важливу закономірність: якщо у випадку з *Asset* (див. вище) транслітерація була вимушеним кроком для уникнення колізій, то у випадку з *Friction* вона стає стилістичною вадою, оскільки українська мова має повноцінний, семантично точний відповідник.

Окрім лексичних розбіжностей, значний пласт проблем локалізації лежить у площині словотвору. Специфіка технічної англійської мови (зокрема «Developer English») характеризується високою продуктивністю конверсії - переходу слів з однієї частини мови в іншу без зміни форми. Яскравим прикладом є дієслівне вживання іменників, що становить неабиякий виклик для перекладу флективною українською мовою. Ключовим кейсом у цій категорії визначено переклад процесу instancing - унікальної архітектурної механіки Godot, що дозволяє вкладати одну сцену всередину іншої як шаблон. В оригіналі це поняття часто вербалізується

дієсловом, як у фразі: *«You can instance a scene from a file»*. Аналіз перекладів виявив гостру конкуренцію двох діаметрально протилежних стратегій відтворення цього процесу.

Стратегія 1: Описовий переклад (Експлікація) Перший підхід полягає у розгортанні значення терміна через нормативну словосполуку: *«Ви можете створити екземпляр сцени з файлу»*. Цей варіант є бездоганним з точки зору літературної норми та академічного стилю. Він чітко передає суть процесу без порушення лексичних правил. Однак у контексті технічної інструкції така конструкція виявляється надмірно громіздкою (три слова замість одного). Це суперечить принципу «мовної економії», який є визначальним для професійної комунікації розробників. Крім того, багаторазове повторення довгих фраз у межах одного абзацу («створити екземпляр», «видалення екземпляра», «посилання на екземпляр») перевантажує текст і сповільнює читання.

Стратегія 2: Калькування (Неологізація) Другий підхід базується на використанні транслітерованого дієслова-неологізму: *«Ви можете інстанціювати сцену...»*. Хоча лексема «інстанціювати» відсутня у загальних нормативних словниках української мови, вона міцно закріпилася у професійному жаргоні ІТ-спільноти. Її перевага полягає у прямій відповідності програмному коду (методу .instance()) та лаконічності, що дозволяє зберегти динаміку англійського оригіналу.

Головним недоліком проаналізованих текстів є не сам вибір тієї чи іншої стратегії, а їх хаотичне чергування. Встановлено, що в межах одного розділу документації (а іноді навіть одного абзацу) сцену можуть спочатку «інстанціювати», а через речення - «створювати її екземпляр». Така стилістична еклектика руйнує цілісність сприйняття матеріалу. Читач змушений витрачати додатковий когнітивний ресурс на зіставлення різних назв одного й того самого процесу. Цей приклад наочно доводить, що у виборі між «академічною правильністю» та «професійним сленгом»

найважливішим критерієм має залишатися консистентність обраного варіанта.

Насамкінець, окремим викликом, що межує з програмуванням, є забезпечення граматичної коректності у реченнях зі змінними підстановки (placeholder variables). Технічна документація Godot рясніє конструкціями типу %s, [param name], або {0}, замість яких рушій автоматично підставляє назви вузлів, класів або помилок у реальному часі. Оскільки англійська мова має аналітичний склад і позбавлена категорії відмінка для іменників, пряма підстановка змінної в речення зазвичай не викликає проблем. Натомість для флективної української мови це створює ситуацію невизначеності.

Розглянемо типовий приклад рядка: *«%s properties»*. У процесі локалізації перекладач бачить лише цей код, не знаючи, яке саме слово буде підставлено замість %s. Якщо він обирає прямий переклад *«Властивості %s»*, результат залежатиме від граматичного роду підставленого слова, який неможливо передбачити:

- Якщо %s = *Body* (тіло, сер. рід), фраза *«Властивості Тіла»* звучить коректно.
- Якщо %s = *Camera* (камера, жін. рід), фраза *«Властивості Камера»* є граматичною помилкою (потрібен родовий відмінок: *«Камери»*).
- Якщо змінна залишається англійською (*«Властивості KinematicBody2D»*), конструкція є прийнятною лише умовно.

Ця проблема є наслідком явища об’єднання рядків, яке Берт Есселінк визначає як одну з головних технічних перешкод локалізації, рекомендуючи уникати складання речень з окремих слів-змінних [28, с. 18].

Аналіз успішних кейсів демонструє, що єдиною ефективною стратегією у таких випадках є синтаксична перебудова речення з використанням «захисних» іменників. Розглянемо інший приклад: *«Select the %s node»*. Замість дослівного відтворення, перекладачі використали конструкцію з родовим словом: *«Виберіть вузол %s»*. У цьому варіанті граматичне навантаження (знахідний відмінок від дієслова «виберіть»)

приймає на себе українське слово «вузол», тоді як змінна %s виступає у ролі незмінюваної прикладки в називному відмінку. Це дозволяє коректно підставити будь-яку назву (*«Виберіть вузол Камера», «Виберіть вузол Тіло»*) без порушення граматичних норм.

Для глибшого розуміння специфіки перекладу технічної документації Godot Engine та пошуку оптимальних стратегій для української мови доцільно провести зіставний аналіз із німецькою локалізацією. Німецька мова демонструє альтернативні підходи до вирішення термінологічних проблем, що дозволяє виявити певні універсальні тенденції. Аналіз корпусу німецької документації Godot виявив домінування стратегії прямого запозичення, що корелює з концепцією «міжнародної технічної мови», описаною Петером Шміттом [44, с. 88]. Зокрема, терміни, які в українському перекладі викликають дискусії щодо доцільності транслітерації (наприклад, *Asset*), у німецькій версії здебільшого залишаються без перекладу, адаптуючись лише графічно через капіталізацію іменників (англ. *Asset* - нім. *das Asset*). Такий підхід дозволяє уникнути небажаної синонімії та зберегти однозначний зв'язок із програмним кодом, що підтверджує тезу про пріоритет технічної точності над мовним пуризмом у професійних спільнотах розробників [40, с. 112].

Окрім запозичень, німецька локалізація ефективно використовує стратегію калькування шляхом утворення складних іменників (композитів), що є характерною рисою цієї мови. Наприклад, англійський термін *Rigid Body* відтворюється як *Starrkörper*, що є повним семантичним еквівалентом, який зберігає компактність єдиного поняття, на відміну від українського двослівного словосполучення «тверде тіло». Особливої уваги заслуговує морфологічна адаптація дієслів, що підтверджує валідність словотвірних процесів, зафіксованих і в українському перекладі. Так, англійське дієслово *to instance* у німецькій документації трансформувалося в неологізм *instanziieren*. Цей факт слугує вагомим аргументом на користь легітимізації українського терміна «інстанціювати», оскільки демонструє, що навіть у

сталих європейських мовах фахівці надають перевагу створенню точного однослівного дієслова замість використання громіздких описових конструкцій («створювати екземпляр»). Згідно з функціоналістським підходом Крістіани Норд, така стратегія є виправданою, оскільки вона забезпечує економію мовних засобів та динаміку тексту, необхідну для інструктивного жанру [37, с. 29]. Таким чином, німецький досвід засвідчує, що відхід від літературної норми на користь професійного узусу є глобальною тенденцією в локалізації IT-продуктів, яку варто враховувати при формуванні української термінологічної політики.

Проведений аналіз прикладів засвідчує, що переклад документації Godot Engine вимагає від фахівця постійного балансування між трьома факторами: технічною точністю (уникнення колізій *Asset/Resource*), стилістичною доцільністю (вибір між *«інстанціювати»* та *«створити екземпляр»*) та граматичною безпекою (робота зі змінними %s). Встановлено, що найбільш якісними є ті перекладацькі рішення, де лінгвістична форма підпорядковується архітектурній логіці програмного забезпечення, навіть якщо це вимагає певних поступок у літературному стилі.

3.3. Компетентнісний профіль перекладача технічної документації в ігровій індустрії

Узагальнюючи результати аналізу специфіки open-source документації та якості вихідного тексту, стає очевидним, що для забезпечення належного рівня локалізації таких проєктів, як Godot Engine, традиційна філологічна підготовка виявляється недостатньою. Складність предметної області та глибока технічна інтеграція текстів у програмне середовище диктують необхідність формування фахівця нового типу, чий компетентнісний профіль має виразно міждисциплінарний характер. На відміну від класичного технічного перекладача, який працює з лінійним текстом, локалізатор ігрового рушія діє на перетині лінгвістики, програмування та геймдизайну. Згідно з Мінако О'Хаган, ця діяльність вимагає «технологічного повороту» у

свідомості фахівця, де лінгвістичні навички стають лише одним із компонентів загальної компетентності [39, с. 15]. Тому базовою вимогою стає не лише досконале володіння мовними парами, а й глибока предметна компетенція у сфері GameDev. Перекладач зобов'язаний розуміти архітектуру рушія, принципи об’єктно-орієнтованого програмування (ООП) та фізику віртуальних середовищ. Емпіричний аналіз підтверджує: без розуміння алгоритму роботи *RayCast* (променя) або *Viewport* (видового екрана) неможливо підібрати адекватний термінологічний відповідник, який би забезпечував функціональну точність інструкції та не вводив користувача в оману.

Окрім суто лінгвістичних та предметних знань, критично важливою складовою профілю визначено інструментально-технологічну компетентність, адаптовану до сучасної парадигми «Docs as Code» (документація як код). Сучасний перекладач open-source проєктів зобов'язаний володіти навичками роботи з системами контролю версій (Git), розуміти синтаксис мов розмітки (reStructuredText, Markdown) та ефективно взаємодіяти зі спеціалізованими хмарними платформами, такими як Weblate чи Crowdin. Ця вимога, яку Берт Есселінк класифікує як елемент «інженерії локалізації», продиктована необхідністю не просто трансформувати текст, а й гарантувати його структурну валідність: коректно обробляти змінні підстановки, внутрішні гіперпосилання та фрагменти коду [28, с. 14]. Більше того, здатність читати та аналізувати вихідний код (наприклад, файли C++ або GDScript) трансформується з опціональної переваги в обов’язкову професійну навичку, оскільки в умовах «контекстуальної сліпоти» саме код часто залишається єдиним верифікаційним джерелом для з’ясування граматичних категорій чи логіки роботи алгоритму.

Ще одним внутрішнім компонентом професійного профілю визначено розвинені дослідницькі та аналітичні навички (*information mining competence*), необхідні для нейтралізації лінгвістичних та логічних вад, властивих *«Developer English»*. Фахівець мусить діяти у ролі дослідника-

верифікатора, здатного критично оцінювати якість вхідного матеріалу, ідентифікувати когнітивні розриви в тексті розробників та реконструювати інтенцію автора через емпіричну перевірку [45, с. 112]. На практичному рівні це реалізується шляхом моделювання описаних ситуацій: відтворення функції або налаштування параметра безпосередньо в середовищі Godot для з'ясування їхньої реальної поведінки. Такий підхід вимагає високого рівня когнітивної гнучкості та готовності до перманентного самонавчання, оскільки функціонал рушія оновлюється в режимі *rolling release* (щоденно). У цьому контексті перекладацька діяльність трансформується з репродуктивної (лінійне відтворення тексту) у евристичну (пошук, висування гіпотези та верифікація знань), що за своєю суттю наближає процес локалізації до науково-дослідної роботи.

Окремим вектором професійного розвитку є формування соціокультурної компетенції, що передбачає глибоку імерсію (занурення) фахівця у специфічний дискурс розробників відеоігор. Спільнота Godot Engine функціонує не як закрита корпоративна структура, а як демократична екосистема, що диктує особливий тонус комунікації - менш формальний, ніж у традиційному IT-секторі, але технічно суворий [38, с. 12]. Тому перекладач повинен володіти професійним чуттям, що дозволяє інтуїтивно розрізняти доречний фаховий узус, який робить текст ергономічним, від неприпустимого сленгу чи надмірної академічної сухості. Це вимагає від фахівця бути активним суб'єктом індустрії: моніторити обговорення на платформах Reddit чи Discord та розуміти специфічний контекст ігрових подій (*Game Jams*) [23, с. 48]. Лише за умови такого культурного бекграунду перекладений текст сприйматиметься цільовою аудиторією як автентичний продукт, створений за принципом «від розробників для розробників», а не як відчужений текст, механічно трансформований стороннім лінгвістом.

Завершальним структурним елементом профілю визначено блок соціально-комунікативних навичок (*soft skills*), актуалізація яких зумовлена специфікою горизонтальної взаємодії у волонтерських *open-source*

спільнотах. В умовах відсутності директивного ієрархічного управління, пріоритетного значення набуває комунікативна компетентність та здатність до конструктивної колаборації. Перекладач зобов'язаний володіти культурою фахової аргументації для обстоювання своїх рішень у дискусіях, а також психологічною стійкістю для коректного сприйняття критики під час процедур перехресного рецензування (*peer review*) [34, с. 56]. Окремим етичним імперативом є дисциплінноване дотримання колективних конвенцій (затверджених глосаріїв, стайлгайдів), навіть якщо вони суперечать індивідуальним стилістичним уподобанням виконавця.

Таким чином, компетентний перекладач документації Godot Engine кваліфікується як «гібридний» спеціаліст, який гармонійно синтезує філологічну ерудицію, технічне мислення інженера та етику учасника відкритої спільноти. Саме такий синергетичний набір компетенцій дозволяє йому ефективно мінімізувати інформаційну ентропію та забезпечувати високу якість україномовного технічного контенту.

3.4. Рекомендації щодо покращення перекладу open-source документації

Узагальнення результатів проведеного лінгвістичного та структурного аналізу дає підстави для формулювання комплексу практичних рекомендацій, спрямованих на системну оптимізацію якості української локалізації Godot Engine. Оскільки етіологія більшості виявлених девіацій лежить у площині організації процесу та координації гетерогенної групи волонтерів, першочерговим заходом визначено розробку та імплементацію єдиного нормативного глосарія проєкту. Згідно з принципами стандартизації, описаними Сью-Еллен Райт, цей документ мусить мати не рекомендаційний, а прескриптивний (обов'язковий) статус для всіх учасників перекладацької спільноти [50, с. 197]. Глосарій має фіксувати уніфіковані відповідники для критичних термінологічних вузлів (зокрема *asset, instance, bake, viewport*) та регламентувати правила транслітерації нових понять. Критично важливою

умовою ефективності є технічна інтеграція цього ресурсу безпосередньо у платформу Weblate. Як зазначає Л. Боукер, функція автоматичного розпізнавання термінології (*terminology recognition*) дозволяє системі підсвічувати терміни в оригіналі та пропонувати затверджений варіант перекладу [24, с. 84]. Це дозволяє мінімізувати вплив людського фактору та забезпечити наскрізну консистентність тексту навіть за умови різного рівня кваліфікації волонтерів.

Паралельно з лексичною уніфікацією, наступним стратегічним кроком визначено розробку та впровадження Стилістичного довідника (Style Guide), адаптованого до структурних особливостей української мови. Цей нормативний документ покликаний формалізувати комунікативну стратегію звернення до користувача, нівелюючи синтаксичну асиметрію між англійською та українською мовами. Згідно з галузевими стандартами (наприклад, Microsoft Manual of Style), довідник має закріпити пріоритет використання активних конструкцій та наказового способу в інструкціях (імператив), а також чітко регламентувати типографічні норми оформлення елементів інтерфейсу (використання лапок, жирного шрифту, курсиву).

Окремий структурний блок довідника необхідно присвятити профілактиці технічних помилок при роботі з кодом розмітки reStructuredText. Враховуючи високий поріг входження для новачків, інструкція повинна детально пояснювати функціональне значення службових символів та критичні наслідки їх випадкового видалення або модифікації [32, с. 119]. Наявність такого «бортового журналу» дозволить оптимізувати криву навчання (*learning curve*) для нових контриб'юторів, пришвидшити їхню інтеграцію в робочий процес та суттєво зменшити обсяг стилістичних виправлень на етапі модерації. З метою елімінації явища термінологічної розсинхронізації між текстовою документацією та програмним інтерфейсом (GUI), рекомендується інституалізувати процедуру обов'язкової контекстуальної крос-верифікації. Методика постулює відмову від ізольованого опрацювання тексту; еталонним сценарієм затверджується

паралельна робота перекладача з активною сесією редактора Godot, перемкнутим на українську локаль.

Такий підхід дозволяє емпірично верифікувати номенклатуру елементів керування (кнопок, меню, діалогових вікон), гарантуючи наскрізну відповідність термінології та високий рівень юзабіліті для кінцевого користувача. У складних випадках, коли локалізація інтерфейсу ще відсутня або містить помилки, пріоритет тимчасово надається документації. Однак, згідно з принципами Agile-розробки, про такі колізії необхідно оперативно повідомляти координаторів локалізації самого рушія через систему баг-трекінгу (GitHub Issues), ініціюючи виправлення першоджерела помилки та гармонізацію всієї екосистеми продукту. Встановлено, що саме така активна взаємодія перекладача з екосистемою розробки дозволяє перетворити локалізацію з вторинного процесу на повноцінний інструмент вдосконалення програмного продукту [26, с. 92].

Насамкінець, стратегічним організаційним заходом визначено інституціоналізацію дворівневої системи модерації контенту. Зважаючи на відкриту архітектуру *open-source* спільноти, яка допускає до процесу учасників без підтвердженої кваліфікації, пряма публікація перекладів (*commit*) від нових контриб'юторів класифікується як критичний ризик для цілісності документації. Рекомендована методика передбачає налаштування робочого процесу (*workflow*) за моделлю премодерації, де пропозиції неавторизованих учасників проходять обов'язковий етап рецензування (*peer review*) досвідченими редакторами або мовними лідерами (*maintainers*) проєкту. Такий фільтр, який М. Хіменес-Креспо визначає як необхідну умову професіоналізації краудсорсингу, дозволяє відсіювати типові девіації - зокрема «хибних друзів перекладача» та порушення граматичної узгодженості змінних - ще на етапі *Pull Request*, до моменту оновлення офіційної документації [34, с. 112]. Системна імплементація запропонованих рекомендацій дозволить трансформувати стихійний потік волонтерських

зусиль у керований, стандартизований процес створення високоякісного технічного контенту.

У третьому розділі розроблено та обґрунтовано комплексну методику перекладу технічної документації open-source проєктів (на прикладі Godot Engine), яка базується на поєднанні лінгвістичного аналізу з інженерними принципами локалізації програмного забезпечення.

Встановлено, що специфіка предметної області вимагає відмови від лінійного відтворення тексту на користь функціоналістського підходу. Емпіричний аналіз перекладацьких рішень (зокрема кейсів *Asset, Instancing, Friction*) засвідчив, що найбільші виклики для локалізації становлять термінологічні колізії з назвами класів, конкуренція між нормативною лексикою та професійним жаргоном, а також технічна проблема граматичної узгодженості змінних підстановки. Доведено, що успішне подолання цих труднощів можливе лише за умови переходу від інтуїтивного перекладу до стратегії, що включає етапи контекстуальної верифікації в середовищі рушія та перевірки структурної цілісності коду розмітки.

Сформовано компетентнісний профіль сучасного локалізатора ігрового ПЗ як «гібридного» фахівця. Визначено, що окрім філологічної підготовки, критично необхідними є предметні знання у сфері GameDev, інструментальна грамотність (робота з Git, Weblate, синтаксисом reStructuredText) та соціокультурна компетентність, що дозволяє відтворювати автентичний дискурс спільноти розробників.

За результатами дослідження запропоновано пакет практичних рекомендацій, спрямованих на систематизацію волонтерської роботи. Ключовими елементами визначено впровадження прескриптивного глосарія з інтеграцією у CAT-інструменти, розробку адаптованого Style Guide та інституалізацію дворівневої системи модерації (пре-модерації). Реалізація цих заходів дозволяє мінімізувати вплив людського фактору (низької кваліфікації волонтерів) та забезпечити створення стандартизованого, технічно точного та стилістично цілісного україномовного контенту.

ВИСНОВКИ ДО РОЗДІЛУ 3

У третьому розділі було проведено практичний аналіз термінологічної консистентності перекладу документації *Godot Engine*, що дозволило виявити реальний стан уніфікації фахової лексики в українському сегменті проєкту. Шляхом зіставлення англомовних оригіналів та їхніх українських відповідників було встановлено, що попри зусилля спільноти, у текстах зберігається певний рівень термінологічної варіантності.

Дослідження продемонструвало, що найбільші труднощі виникають при перекладі базових концептів ігрового рушія (наприклад, термінів *node, scene, signal, viewport*), для яких у волонтерському перекладі часто використовуються різні стратегії: від повної транслітерації до пошуку описових еквівалентів. Аналіз виявив випадки «контекстуальної розрізненості», коли один і той самий термін у різних розділах документації перекладається синонімічними парами, що може ускладнювати сприйняття технічних інструкцій кінцевим користувачем.

На основі отриманих даних було розроблено низку практичних рекомендацій для покращення консистентності. Доведено, що ключовим фактором стабілізації терміносистеми є активне розширення інтегрованого глосарія в системі *Weblate* та впровадження обов’язкового етапу редактури (review) досвідченими учасниками проєкту. Такий підхід дозволить мінімізувати вплив суб'єктивного чинника, притаманного краудсорсингу.

Підсумовуючи результати аналізу, можна стверджувати, що досягнення високого рівня термінологічної єдності в open-source проєктах є ітеративним процесом, який потребує балансу між гнучкістю живої мови та суворою регламентацією технічного стилю. Результати проведеного аналізу не лише підтверджують гіпотезу дослідження, а й мають практичну цінність для подальшої локалізації документації Godot Engine та інших програмних продуктів з відкритим кодом.

ЗАГАЛЬНІ ВИСНОВКИ

У результаті проведеного комплексного дослідження було детально опрацьовано теоретичну базу та практичні аспекти перекладу англомовної документації проєктів із відкритим сирцевим кодом, що дозволило повністю реалізувати мету та виконати всі завдання, поставлені на початку роботи. На основі аналізу наукової літератури було з’ясовано, що термін як фундаментальна одиниця професійної комунікації залишається одним із найскладніших об’єктів для дефінування в сучасній лінгвістиці через свою багатогранність та залежність від галузевого контексту. У роботі було доведено, що термінологічна система в галузі інформаційних технологій характеризується надзвичайною динамічністю, де переклад виступає не просто інструментом передачі інформації, а потужним джерелом формування національного фахового лексикону. Було встановлено, що ключові ознаки терміна, такі як системність, однозначність та відсутність експресивного забарвлення, набувають особливого значення саме в технічній документації, оскільки від точності їх перекладу залежить коректність експлуатації складного програмного забезпечення.

Окрему увагу в роботі було приділено вивченню специфіки open-source середовища, яке диктує особливі умови для перекладацької діяльності. Було виявлено, що на відміну від комерційної локалізації, де процеси суворо регламентовані корпоративними стандартами, переклад документації проєктів з відкритим кодом, зокрема Godot Engine, базується на принципах краудсорсингу та добровільної участі спільноти. Це створює специфічний децентралізований робочий процес, де велика кількість виконавців з різним рівнем лінгвістичної та технічної підготовки працює над одним масивом тексту. Такий підхід зумовлює основну проблему дослідження - ризик втрати термінологічної консистентності, що виникає через відсутність прямого комунікативного зв'язку між усіма учасниками перекладу та суб'єктивне сприйняття фахової лексики кожним волонтером окремо. При аналізі

інструментарію локалізації, зокрема платформи Weblate, було зроблено висновок, що технічні засоби лише частково вирішують проблему уніфікації, надаючи можливості для створення глосаріїв, проте вирішальним фактором залишається людський чинник та інтелектуальна робота над вибором еквівалентів.

Значущим дестабілізуючим фактором визначено специфіку мови оригіналу, яка у проєкті Godot Engine часто функціонує як «англійська як лінгва франка» (ELF). Оскільки вихідні тексти створюються розробниками з усього світу, вони часто демонструють ознаки «Developer English»: синтаксичну компресію, еліпсис та термінологічну нестабільність навіть у межах одного розділу. Така лінгвістична неоднорідність першоджерела стає первинним чинником, що провокує термінологічну амбівалентність та змушує перекладача виконувати роль «технічного дедуктора». Відтак, якість української локалізації прямо залежить від здатності фахівця нейтралізувати вади вихідного тексту через глибоку змістову верифікацію. Це підкреслює, що робота з open-source документацією вимагає не просто лінгвістичної адаптації, а й деконструкції авторської інтенції, прихованої за ненормативною граматикою оригіналу.

Практичний аналіз фактичного матеріалу, а саме українського сегмента документації ігрового рушія Godot Engine, підтвердив гіпотезу про те, що термінологічна консистентність у великих open-source проєктах перебуває під постійним впливом децентралізованого середовища розробки. Дослідивши структуру та змістове наповнення документації, ми встановили, що вона є складною міждисциплінарною системою, де класичні терміни програмування тісно переплітаються з поняттями комп’ютерної графіки, фізичного моделювання та проєктування інтерфейсів. Така багатогранність об’єкта дослідження висуває надзвичайно високі вимоги до перекладача, який має не лише володіти мовною нормою, а й розуміти технічну логіку функціонування ігрового рушія. У ході розвідки було виявлено, що основною причиною порушення термінологічної єдності є відсутність усталених

українських еквівалентів для новітніх концептів геймдеву, що змушує волонтерів самостійно обирати між стратегіями прямого запозичення, калькування або пошуку описових відповідників.

Також, для глибшого розуміння специфіки перекладу технічної документації Godot Engine та пошуку оптимальних стратегій для української мови було проведено зіставний аналіз із німецькою локалізацією. Німецька мова демонструє альтернативні підходи до вирішення термінологічних проблем, що дозволяє виявити певні універсальні тенденції. Аналіз корпусу німецької документації виявив домінування стратегії прямого запозичення, що корелює з концепцією міжнародної технічної мови. Зокрема, терміни, які в українському перекладі викликають дискусії щодо доцільності транслітерації (наприклад, Asset), у німецькій версії здебільшого залишаються без перекладу, адаптуючись лише графічно через капіталізацію іменників. Такий підхід дозволяє уникнути небажаної синонімії та зберегти однозначний зв'язок із програмним кодом, що підтверджує тезу про пріоритет технічної точності над мовним пуризмом у професійних спільнотах розробників. Окрім запозичень, німецька локалізація ефективно використовує стратегію калькування шляхом утворення складних іменників, що дозволяє зберігати компактність єдиного поняття (наприклад, Starrkörper для Rigid Body).Це демонструє, що навіть у сталих європейських мовах фахівці надають перевагу створенню точного однослівного дієслова замість використання громіздких описових конструкцій. Німецький досвід засвідчує, що відхід від літературної норми на користь професійного узусу є глобальною тенденцією в локалізації IT-продуктів, яку варто враховувати при формуванні української термінологічної політики.

Під час детального вивчення перекладів ключових системних одиниць, таких як вузли, сцени та сигнали, було зафіксовано випадки контекстуальної варіативності, коли один і той самий англомовний термін отримує різні українські інтерпретації в межах одного розділу документації. Це безпосередньо впливає на когнітивне сприйняття тексту користувачем,

оскільки термінологічна синонімія в технічному дискурсі сприймається як помилка або як позначення різних об’єктів, що може призвести до хибних дій розробника під час роботи з програмою. Аналіз роботи спільноти на платформі Weblate показав, що попри наявність інструментів автоматичного контролю та вбудованих словників, рівень консистентності значною мірою залежить від активності модераторів та частоти оновлення глосаріїв. Ми дійшли висновку, що в проєктах із відкритим кодом термінологічна єдність є не статичним результатом, а динамічним процесом постійного узгодження, де кожне нове оновлення вихідного коду англійською мовою створює черговий виклик для української локалізації. Було доведено, що найбільша розбіжність спостерігається в нових модулях документації, які ще не пройшли етап колективного обговорення та редактури досвідченими учасниками проєкту.

На основі виявлених під час дослідження проблем було сформульовано та обґрунтовано низку практичних рекомендацій, спрямованих на підвищення рівня термінологічної консистентності в українському сегменті документації Godot Engine. Ми встановили, що найефективнішим методом подолання термінологічної розрізненості є впровадження жорсткого механізму синхронізації глосаріїв між різними гілками перекладу та розробка розширеного стайл-гайду, який би регламентував не лише переклад окремих слів, а й специфічні синтаксичні конструкції, притаманні технічному стилю. Особлива роль у цьому процесі належить інституту рецензування, де досвідчені учасники спільноти виконують функцію мовних та технічних редакторів, забезпечуючи перевірку нових внесків на відповідність уже прийнятим стандартам. У роботі доведено, що лише поєднання автоматизованих засобів перевірки платформи Weblate із регулярною інтелектуальною модерацією дозволяє створити цілісний і зрозумілий продукт, який буде відповідати очікуванням професійної спільноти розробників. Крім того, було зазначено, що активна взаємодія перекладачів із розробниками самого рушія сприяє кращому розумінню контексту

використання термінів, що суттєво мінімізує помилки на етапі вибору еквівалентів.

Загалом, проведене дослідження дозволяє стверджувати, що поставлена мета була повністю досягнута, а всі завдання - виконані в повному обсязі. Ми не лише проаналізували специфіку перекладу open-source документації та виявили чинники, що впливають на її якість, а й надали практичний інструментарій для покращення процесів локалізації. Наукова новизна та практична значущість роботи полягають у системному підході до вивчення термінологічної консистентності в умовах волонтерського перекладу, що раніше не отримувало достатнього висвітлення в українській лінгвістиці стосовно ігрових рушіїв. Результати цієї роботи можуть бути використані як теоретичне підґрунтя для подальших досліджень у галузі спеціального перекладу та технічної комунікації, а також як практичний посібник для команд локалізації, що працюють над іншими проєктами з відкритим кодом. Таким чином, дипломна робота представляє собою завершене дослідження, яке підтверджує важливість дотримання суворих термінологічних стандартів для розвитку сучасної української IT-термінології та її успішної інтеграції у світовий науково-технічний простір.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ажнюк Б. М. Мовна політика і мовна ситуація в Україні : монографія. К. : Вид. дім «Дмитра Бураго», 2015. 168 с.
2. Бацевич Ф. С. Основи комунікативної лінгвістики : підручник. К. : Академія, 2004. 344 с.
3. Вакуленко М. О. Українська термінологія: питання стану та перспективи розвитку. К. : Наук.-техн. центр при Президії НАН України, 2011. 224 с.
4. Горностай Т. О. Уніфікація термінології у вихідному тексті як запорука якості перекладу : *Проблеми української термінології : зб. наук. пр.* : Нац. ун-т «Львівська політехніка». Львів, 2010. С. 112–115.
5. Дебела. А. С. Особливості краудсорсингу в перекладі. Київ, 2021. 76 с.
6. Д'яков А. С., Кияк Т. Р., Куделько З. Б. Основи термінотворення: семантичні та соціолінгвістичні аспекти. К. : КМ Academia, 2000. 218 с.
7. Зубков М. Г. Сучасний український діловий стиль. Х. : Торсінг, 2005. 448 с.
8. Іващенко В. Л. Концептуальна репрезентація фрагментів знання в терміносистемах. К. : Вид. Дім Дмитра Бураго, 2006. 328 с.
9. Карабан В. І. Переклад англійської наукової і технічної літератури. Граматичні труднощі, лексичні, термінологічні та жанрово-стилістичні проблеми. Вінниця : Нова книга, 2004. 576 с.
10. Кияк Т. Р. Фахові мови та проблеми термінознавства : Нова філологія. 2007. № 27. С. 205–208.
11. Комова М. В. Документна лінгвістика. Львів : Тріада плюс, 2009. 216 с.
12. Корунець І. В. Теорія і практика перекладу (аспектний переклад). Вінниця : Нова книга, 2003. 448 с.
13. Литвин І. М. Перекладознавство. Черкаси : Брама-Україна, 2013. 272 с.
14. Максімов С. Є. Практичний курс перекладу (англійська та українська мови) : теорія та практика перекладу. К. : Ленвіт, 2006. 158 с.
15. Панько Т. І., Кочан І. М., Мацюк Г. П. Українське термінознавство. Львів : Світ, 1994. 216 с.

16. Пономарів О. Д. Культура слова: Мовностилістичні поради. К. : Либідь, 2001. 240 с.
17. Селіванова О. О. Сучасна лінгвістика: термінологічна енциклопедія. Полтава : Довкілля-К, 2006. 716 с.
18. Цимбал Н. О. Англо-український словник термінів комп'ютерних технологій. К. : Книжкове вид-во НАУ, 2007. 320 с.
19. Шамілі О. В. Лексико-семантичні особливості англомовної термінології відеоігор : *Науковий вісник Міжнародного гуманітарного університету. Серія : Філологія.* 2018. № 34. С. 144–147.
20. Arntz R., Picht H., Mayer F. Einführung in die Terminologiearbeit. Hildesheim : Georg Olms Verlag, 2014. 332 s.
21. Baker M. In Other Words: A Coursebook on Translation. 2nd ed. London ; New York : Routledge, 2011. 332 p.
22. Bell R. T. Translation and Translating: Theory and Practice. Longman, 1991. 298 p.
23. Bernal-Merino M. Á. Translation and Localisation in Video Games: Making Entertainment Global. London ; New York : Routledge, 2014. 282 p.
24. Bowker L. Computer-Aided Translation Technology: A Practical Introduction. Ottawa : University of Ottawa Press, 2002. 185 p.
25. Chandler H. M., Deming S. O. The Game Localization Handbook. 2nd ed. Hingham : Charles River Media, 2011. 370 p.
26. Cronin M. Translation in the Digital Age. London ; New York : Routledge, 2013. 168 p.
27. Diaz Cintas J. New Trends in Audiovisual Translation. Bristol ; Buffalo ; Toronto : Multilingual Matters, 2009. 272 p.
28. Esselink B. A Practical Guide to Localization. Amsterdam ; Philadelphia : John Benjamins Publishing Company, 2000. 488 p.
29. Gambier Y., Gottlieb H. (Multi)Media Translation: Concepts, Control and Cognition. Amsterdam ; Philadelphia : John Benjamins Publishing Company, 2001. 273 p.

30. Godot Engine documentation : електрон. версія. URL : https://docs.godotengine.org/en/stable/ (дата звернення: 24.12.2025).
31. Göpferich S. Interkultureller Transfer von Software-Lokalisierung : Textlogistik. Software-Lokalisierung und technische Dokumentation. Tübingen : Gunter Narr Verlag, 1998. S. 169–202.
32. Hartley T. Technology and Translation : The Oxford Handbook of Translation Studies : ed. by Kirsten Malmkjær, Kevin Windle. Oxford : Oxford University Press, 2011. P. 116–127.
33. House J. Translation Quality Assessment: Past and Present. London ; New York : Routledge, 2015. 160 p.
34. Jiménez-Crespo M. A. Crowdsourcing and Online Collaborative Translations: Expanding the Limits of Translation Studies. Amsterdam : John Benjamins Publishing Company, 2017. 209 p.
35. Microsoft Manual of Style : електрон. версія. 4th ed. Microsoft Press, 2012. 448 p. URL : https://learn.microsoft.com/en-us/style-guide/welcome/ (дата звернення: 28.12.2025).
36. Munday J. Introducing Translation Studies: Theories and Applications. 4th ed. London ; New York : Routledge, 2016. 394 p.
37. Nord C. Translating as a Purposeful Activity: Functionalist Approaches Explained. Manchester : St. Jerome Publishing, 1997. 154 p.
38. O'Hagan M. Community Translation: Translation as a Social Activity and Its Possible Consequences in the Digital Age : Digling. 2011. Vol. 2. P. 11–23.
39. O'Hagan M., Mangiron C. Game Localization: Translating for the Global Digital Entertainment Industry. Amsterdam ; Philadelphia : John Benjamins Publishing Company, 2013. 374 p.
40. Pym A. Exploring Translation Theories. London ; New York : Routledge, 2014. 178 p.
41. Reiß K., Vermeer H. J. Grundlegung einer allgemeinen Translationstheorie. Tübingen : Niemeyer, 1984. 245 s.

42. Routledge Encyclopedia of Translation Studies : ed. by Mona Baker, Gabriela Saldanha. Routledge, 2009. 674 p.
43. SBT Localization : офіц. сайт. URL : https://sbt.com.ua/ (дата звернення: 20.12.2025).
44. Schmitt P. A. Handbuch Technisches Übersetzen. Berlin : BDÜ Fachverlag, 2019. 420 s.
45. Stolze R. Fachübersetzen – Ein Lehrbuch für Theorie und Praxis. Berlin : Frank & Timme, 2009. 312 s.
46. Toury G. Descriptive Translation Studies and Beyond. Amsterdam ; Philadelphia : John Benjamins Publishing Company, 1995. 311 p.
47. Venuti L. The Translator's Invisibility: A History of Translation. 2nd ed. London ; New York : Routledge, 2008. 353 p.
48. Vinay J.-P., Darbelnet J. Comparative Stylistics of French and English: A Methodology for Translation. Amsterdam ; Philadelphia : John Benjamins Publishing Company, 1995. 358 p.
49. Weblate : Open source continuous localization : офіц. сайт. URL : https://weblate.org/ (дата звернення: 15.12.2025).
50. Wright S. E., Budin G. Handbook of Terminology Management. Amsterdam ; Philadelphia : John Benjamins Publishing Company, 1997. Vol. 1. 370 p.
51. Zimmer R. Äquivalenz und Adäquatheit. Tübingen : Niemeyer, 2006. 182 s.

ДОДАТКИ

Додаток А

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

AABB – паралелепіпед, вирівняний по осях (англ. «Axis-Aligned Bounding Box»)

API – прикладний програмний інтерфейс (англ. «Application Programming Interface»)

CAT – автоматизований переклад (англ. «Computer-Aided Translation» або «Computer-Assisted Translation»)

CSV – значення, розділені комами (англ. «Comma-Separated Values»)

ELF – англійська як лінгва франка (англ. «English as a Lingua Franca»)

GUI – графічний інтерфейс користувача (англ. «Graphical User Interface»)

IT – інформаційні технології

MIT – ліцензія Массачусетського технологічного інституту (англ. «Massachusetts Institute of Technology»)

ПЗ – програмне забезпечення

QA – забезпечення якості (англ. «Quality Assurance»)

reST / reStructuredText - мова розмітки, що використовується для створення документації

RID – ідентифікатор ресурсу (англ. «Resource ID»)

TEP – переклад, редагування, коректура (англ. «Translation, Editing, Proofreading»)

TM – пам’ять перекладів (англ. «Translation Memory»)

UI – інтерфейс користувача (англ. «User Interface»)

URI – уніфікований ідентифікатор ресурсу (англ. «Uniform Resource Identifier»)

XML – розширювана мова розмітки (англ. «eXtensible Markup Language»)

ООП – об’єктно-орієнтоване програмування 2D / 3D – двовимірний / тривимірний

SUMMARY

The work investigated the structural, semantic, and technical features of English-language open-source documentation translation, specifically focusing on ensuring terminological consistency within the Godot Engine project. It identified systemic problems arising during the collaborative translation process (crowdsourcing) and proposed a comprehensive methodology for its optimization. Open-source documentation is a rapidly developing field where technical content is created and localized by global communities, necessitating new academic approaches to maintain linguistic quality and functional accuracy.

The generalization of the results of the scientific search gives grounds for making the following conclusions: the definition of the concept of "open-source documentation" has been carried out as a dynamic, community-driven technical text integrated into the software development infrastructure under the "Docs as Code" paradigm. Crowdsourcing is defined as a collaborative translation model where functions are performed by an undefined group of volunteers, often referred to as "prosumers" who both consume and produce the content.

The study highlighted that the main challenge in such projects is "terminological entropy" - a state of chaotic accumulation of synonyms caused by the lack of centralized coordination and varying levels of volunteer qualifications. It has been investigated that the linguistic quality of the source text, often termed "Developer English," significantly complicates translation due to syntactic compression, ellipsis, and technical jargon. The study classified the main types of terminological errors in the Ukrainian localization of Godot Engine: unmotivated synonymy (e.g., *Asset* as *aset*, *resurs*, or *aktyv*), lexical interference ("false friends" like *Kontrol* instead of *Keruvannia* for the *Control* node), and grammatical desynchronization of placeholders like *%s*.

The work also took into account the main ways of translating technical terms: equivalent translation, transliteration, calquing, and explication. A comparative analysis with German localization (e.g., the use of the verb *instanziieren*) confirmed the validity of morphological adaptation strategies such as

instantsiiuvaty for the term *instancing*. Furthermore, the study found that the desynchronization between the documentation and the graphical user interface (GUI) poses a significant difficulty for the user, requiring a mandatory cross-verification process.

Thus, the study found that a competent translator in the gaming industry must be a "hybrid" specialist (techno-linguist) who possesses not only philological expertise but also subject-matter knowledge in GameDev, instrumental literacy (Git, Weblate), and socio-cultural competence. It can be concluded that ensuring terminological consistency is a complex process involving the implementation of prescriptive glossaries, style guides, and a two-level moderation system (peer review). These measures allow for transforming chaotic volunteer efforts into a managed process of creating professional technical content that meets both linguistic standards and the functional requirements of the software product.