Міністерство освіти і науки України
«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи магістра

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)

спеціальність *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*
(назва освітньої програми)

освітньо-наукова програма *кібербезпека*

на тему: «Метод захисту авторських прав в комп'ютерних іграх»

Виконавець: студент ІІ курсу, групи КБм-21

Євгеній СТЕЦЮК
(підпис) (прізвище ім'я по-батькові)

	Прізвище, ініціали	Оцінка	Підпис
Науковий керівник	Сергій БУЧИК		

Нормоконтроль	Олена БОГУСЛАВСЬКА		

Київ 2023

Міністерство освіти і науки України

«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри кібербезпеки
та захисту інформації
_____ Сергій ТОЛЮПА
«24» жовтня 2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *125 Кібербезпека*
(код і назва спеціальності)

студенту КБм-21 (група) Стецюку Євгенію Олеговичу (прізвище ім'я по-батькові)

Тема дипломної роботи Метод захисту авторських прав в комп'ютерних іграх

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Рішення засідання кафедри кібербезпеки та захисту інформації факультету інформаційних технологій протокол № 3 від 20.10.2022 р.

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень	процес захисту авторського права в комп'ютерних іграх
Предмет досліджень	механізми захисту авторського права в комп'ютерних іграх
Мета	розробити метод захисту авторських прав в комп'ютерних іграх
Вихідні дані для проведення роботи	комп'ютерна гра розроблена в середовищі ігрового рушія Unity, архітектура сервер-клієнт

3. ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна	удосконалення методу захисту авторського права в комп'ютерних іграх но основі інфраструктури сервер-клієнт

Практична цінність	Використання розробленого методу для захисту від нелегального копіювання комп’ютерних ігор та їх авторського права

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Робота виконана у повному обсязі відповідно до теми.

5. ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	Строки виконання робіт (початок-кінець)
Розробка плану для досягнення мети роботи	24.10.2022 – 23.01.2023
Аналіз існуючих методів захисту	24.01.2023 – 14.02.2023
Розробка методу захисту авторського права в комп’ютерних іграх	15.02.2023 – 24.04.2023
Оформлення і друк пояснювальної записки	25.04.2023 – 19.05.2023

6. РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект	Зниження збитків через викрадення даних нелегальним копіюванням змісту гри.
Соціальний ефект	Покращення захисту авторського права при розробці комп’ютерних ігор.

7. ДОДАТКОВІ ВИМОГИ

Завдання видав _____ Сергій БУЧИК
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ Євгеній СТЕЦЮК
(підпис) (прізвище, ініціали)

Дата видачі завдання: 24.10.2022 р.

Термін подання дипломної роботи до ЕК 19.05.2023 р.

УДК 004.056.53

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Метод захисту авторських прав в комп'ютерних іграх»: 64 сторінки, 31 рисунок, 12 літературних джерел та 1 додаток.

Мета роботи – розробка програмного забезпечення для захисту авторських прав в комп'ютерних іграх.

Об'єкт дослідження – процес захисту авторського права в комп'ютерних іграх.

Предмет дослідження – механізми захисту авторського права в комп'ютерних іграх.

Наукова новизна: удосконалено метод захисту авторського права в комп'ютерних іграх на основі клієнт-серверної архітектури з використанням шифрування.

У роботі проаналізовано роботу популярних ігрових рушіїв, які існують на ринку. Особливу увагу було надано роботі ігрового рушія «Unity», який є одним із найпопулярніших серед розробників комп'ютерних ігор. Оцінено існуючи засоби захисту авторських прав, такі як перевірка диска, CD-ключ, онлайн активація, захист на основі облікового запису, захист на основі постійного підключення до глобальної мережі.

Також було розроблено комп'ютерну гру де продемонстровано, декілька варіантів застосування створеного методу захисту на практиці. Проект був розроблений в середовищі ігрового рушія «Unity» за допомогою програмної мови C#, та серверної частини на веб-платформі «Django» за допомогою програмної мови Python.

Актуальність роботи: в сучасному світі компанії які розробляють ігри почали займати ключове місце у фінансовому світі. Бюджет ігор, які заплановані тільки на 2024-2025 роки вже перевищує 200 мільйонів доларів. Навіть ігри з невеликим бюджетом і з невеликою кількістю людей може продатися більше ніж 6 млн. копій.

Зазвичай невеликі компанії не можуть собі дозволити дорогі системи захисту чи постійну підтримку сервера.

Через недостатній захист авторських прав, такі проекти поширюються глобальною мережею, наносячи фінансові і репутаційні збитки. Тому при розробці представленого в цій роботі методу, було прийнято до уваги саме такі компанії. Розроблений метод вносить незначний вплив на роботи проекту і не коштує розробником великої кількості грошей на підтримку сервера, чи складності інтегрування в свій проект.

Ключові слова: клієнт-серверна архітектура, авторські права, комп’ютерні ігри, ігровий рушій, шифрування.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

DIS - Distributed Interactive Simulation

HLA - High-level Architecture

VE - Virtual Environment

IDE - Integrated development environment

DRM - Digital Right Management

ДДГ - Документ дизайну гри

ПЗ - програмне забезпечення

IDE - Integrated development environment

SDK - Software development kit

ІВ - інтелектуальна власність

BBS - Bulletin board system

ПК - персональний комп'ютер

УІП - Унікальний ідентифікатор пристрою

ВСТУП

В сучасних реаліях питанню захисту авторського права в комп’ютерних іграх приділяється багато уваги. Після появи таких площадок як «Steam» та відкритих безкоштовних ігрових рушії як «Unity» та «Unreal Engine», кількість ігор і компаній які їх видають значно підвищилась. Через зручність використання таких інструментів кожен бажаючий може створити свою гру і спробувати її продати.

Управління з питань конкуренції та ринків Великобританії виявило, що деякі великі видавці повідомляють, що їхні франшизи ААА можуть коштувати понад 1 мільярд доларів [1].

Ці кошти демонструють значне зростання порівняно з тим, коли більшість ігор класу ААА мали бюджет від 50 до 150 мільйонів доларів п’ять років тому.

Але з ростом кількості ігор зросла і кількість випадків нелегального копіювання і розповсюдження таких проектів в глобальній мережі. Тому в даній роботі представлено новий метод захисту таких проектів, який заснований на архітектурі клієнт-сервер, з головною ідеєю, зберігання частини гри на сервері, щоб клієнт не мав весь зміст проекту в себе на пристрої. При такому підході, вартість обслуговування серверів значно знижується, тому що сервер не проводить ніякі обчислення ігрових подій, як це представлено у вже існуючих варіантах схожих систем захисту.

Мета роботи – розробка програмного забезпечення для захисту авторських прав в комп’ютерних іграх.

Об’єкт дослідження – процес захисту авторського права в комп’ютерних іграх.

Предмет дослідження – механізми захисту авторського права в комп’ютерних іграх.

Наукова новизна: удосконалено метод захисту авторського права в комп’ютерних іграх на основі клієнт-серверної архітектури з використанням шифрування і маніпуляціями з файлами на пристрої користувача під час виконання проекту.

Також було розроблено комп’ютерну гру, яка демонструє випадки використання такого методу захисту в різних його варіаціях з різними типами файлу. При розробці було протестовано, що саме впливає на роботу такого захисту і надано рекомендації по його впровадженню.

Актуальність роботи: через недостатній захист авторських прав, комп’ютерні ігри без додаткового захисту авторського права поширюються глобальною мережею, наносячи фінансові і репутаційні збитки розробникам.

РОЗДІЛ 1
ОПИС ПОБУДОВИ КОМП'ЮТЕРНИХ ІГОР

1.1 Життєвий цикл і етапи розробки комп'ютерних ігор

Життєвий цикл розробки гри – це процес створення відеоігри від концепції до завершення. Подібно до життєвого циклу розробки проектів, життєвий цикл розробки ігор допомагає організовувати потік роботи так, щоб кожен знав, що йому потрібно виконати та коли.

Такий цикл також допомагає керувати графіком і бюджетом розробки гри, зменшуючи неефективність і проблемні місця.

Хоча життєві цикли різняться між проектами та студіями, процес досить схожий, незалежно від того, працюєте ви над ААА, інді-грою чи мобільною грою.

У індустрії відеоігор ААА – це неофіційна класифікація, яка використовується для класифікації відеоігор, створених і розповсюджених середнім або великим видавцем, який зазвичай має більший бюджети на розробку та маркетинг, ніж інші рівні ігор.

Інді-гра, скорочення від незалежної відеоігри, – це відеогра, яка зазвичай створюється окремими особами або невеликими командами розробників без фінансової та технічної підтримки великого видавця ігор.

Гра постійно розвивається, і те, що звучало чудово в теорії, може не працювати так добре в реальності. Тому життєвий цикл не обов'язково є лінійним процесом. Роботу потрібно надіслати на творче схвалення, і часто її можна повернути на доопрацювання. Життєві цикли повинні бути достатньо гнучкими, щоб враховувати перегляди та зміни курсу [2].

Розробка відеоігор зазвичай поділяється на 3 етапи: підготовка, виробництво та постпродакшн.

1. Підготовка до виробництва

З цього починається кожен проект. По суті, попереднє виробництво визначає, про що гра, навіщо її створювати та що потрібно для її створення.

Можливо, існує чудова ідея для типу гри, історії, яку є можливість втілити в життя, або є можливість створити таку, яка використовує певний тип технології (наприклад, VR, новий контролер або консоль).

При підготовці знаходяться відповіді на такі запитання [2]:

1) Про що гра?
2) Яка аудиторія?
3) Чи є для цього ринок? Яка конкуренція?
4) На якій платформі це буде опубліковано?
5) Як це буде монетизуватися? Чи буде гра продаватися на платформі чи це буде безкоштовно гра з покупками в середині?
6) Скільки часу знадобиться для розробки?
7) Який персонал і ресурси для цього будуть потрібні?
8) Який орієнтовний бюджет?

Цей етап може тривати від тижня до року, залежно від типу проекту, наявних ресурсів і фінансів, і зазвичай займає до 20% загального часу виробництва.

На даний момент команда досить маленька. Це може бути продюсер, програміст(и), концептуальний дизайнер.

Виробник відеоігор займається бізнес-аспектом проекту, зокрема фінансами. Вони керують бюджетом і розробляють маркетингові стратегії для продажу товару.

Для створення чіткого представлення про гру створюється документ дизайну гри. Документ дизайну гри (ДДГ) – це, по суті, північна зірка гри. Це живий документ, який допомагає кожному зрозуміти та прийняти ширше бачення проекту.

ДДГ включає такі речі, як [3]:

1. ідея або концепція,
2. жанр,
3. історія та персонажі,
4. основна ігрова механіка,

5. геймплей,
6. дизайн рівня та світу,
7. мистецтво та/або ескізи,
8. стратегія монетизації.

Будучи живим документом, ДДГ постійно оновлюється та вдосконалюється протягом усього виробництва. Причиною цього можуть бути технічні чи фінансові обмеження або просто усвідомлення того, що все виглядає не так добре, як ви спочатку сподівалися.

2. Виробництво

Більшість часу, зусиль і ресурсів витрачається на розробку відеоігор на стадії виробництва. Це також один із найскладніших етапів розробки відеоігор. Під час цього процесу [3]:

1. Моделі персонажів розробляються, відтворюються та повторюються, щоб виглядати саме так, як вони повинні виглядати в історії.
2. Аудіодизайн працює невпинно, щоб кожен раз, коли ваш персонаж ступає на пісок, гравій або цемент, він звучав автентично.
3. Дизайнери рівнів створюють середовища, які є динамічними, захоплюючими та підходять для багатьох типів стилів гри.
4. Актори озвучування читають великі стоси сценаріїв, дублюючи дубль за дублем, щоб отримати потрібну емоцію, час і тон.
5. Розробники пишуть тисячі рядків вихідного коду, щоб оживити кожну частину вмісту в грі.
6. Керівники проекту встановлюють етапи та графіки спринтів, забезпечуючи відповідальність кожного відділу та членів його команди. Це особливо важливо, якщо видавець регулярно перевіряє оновлення статусу.

Ці та багато інших подій можуть потребувати років ітерацій, щоб стати правильними, і це за умови, що на цьому шляху буде внесено лише кілька змін, що навряд чи є реальністю.

3. Постпродакшн

Після того, як виробництво завершено та гру завантажено, процес розробки гри продовжується, а деякі члени команди переходять до обслуговування (виправлення помилок, створення виправлень) або створення бонусного або завантажуваного вмісту. Інші можуть перейти до створення продовження або наступного проекту.

Може бути проведено підведення підсумків, щоб обговорити, що спрацювало/не спрацювало, і визначити, що можна було б зробити краще наступного разу. Усі проектні документи, активи та код завершуються, збираються та зберігаються на випадок, якщо вони знадобляться в майбутньому.

1.2 Сучасні методології розроблення комп'ютерних ігор

Що таке ігровий рушій? Наразі здається зрозумілим, що ігровий рушій – це частина програмного забезпечення, яке дозволяє створювати відеоігри. Проте ми досі не маємо правильного визначення того, чого можна очікувати в таких рамках. Основна мета ігрового рушія – абстрагувати загальні функції відеоігор, що дозволяє повторно використовувати код і ігрові ресурси в різних іграх. З цією метою ігровий рушій зазвичай містить такі функції [4]:

- механізм візуалізації для 2D або 3D-графіки;
- обробка вхідних даних (для клавіатури та миші, сенсорних пристроїв чи іншого обладнання);
- ігровий цикл (внутрішня процедура, яка перераховує ігрові події кожного кадру);
- фізичний механізм, із виявленням зіткнень і реагуванням;
- звук;
- граф сцени (який керує графічними елементами та їхніми взаємозв'язками на екрані);
- анімація (для 2D-спрайтів або 3D-моделей);
- керування пам'яттю;

- потоковість процесів (дозволяє використовувати декілька паралельних за раз).

Інші функції можуть включати:

- додаткові файли виконання;
- штучний інтелект;
- робота з мережею;
- потокове передавання даних;
- підтримка локалізації;
- публікація на багатьох платформах.

Найчастіше ці функції підтримуються агрегацією різноманітних сторонніх бібліотек (які в цьому контексті можна позначити як проміжне ПЗ). Таким чином, метою розробників ігрового рушія є їх об'єднання в одному рівні абстракції, в одному інтерфейсі редактора або інтегрованому редакторі розробки (IDE) із прозорою взаємодію та простотою використання. В даний час в ігрових рушіях можна спостерігати дві основні тенденції. По-перше, тепер ігрові рушії набагато частіше використовують мови високого рівня, такі як Java, C# або Python, що часто призводить до збільшення продуктивності для розробників. Враховуючи поточне апаратне забезпечення, накладні витрати на переклад для таких мов стали незначними, а продуктивність гри здебільшого пов'язана з потужністю доступної графічної карти.

Другою основною тенденцією для деяких ігрових движків є спільна мета уможливити кросплатформну публікацію, зберігаючи ту саму кодову базу. Таким чином, більшість ігрових движків наразі можуть публікувати не лише для середовищ персональних комп'ютерів, але й для основних операційних систем мобільних пристроїв і для різних систем ігрових консолей.

Модульна структура ігрових рушіїв.

У сучасних іграх з модульною структурою ігровий рушій відноситься до тієї колекції модулів симуляційного коду, які безпосередньо не визначають поведінку гри (логіку гри) або середовище гри (дані рівня). Механізм містить модулі, що

обробляють введення, вихід (3D-візуалізація, 2D-креслення, звук) і загальну фізику/динаміку для ігрових світів. На рисунку 1.1 показано цю загальну структуру [5].

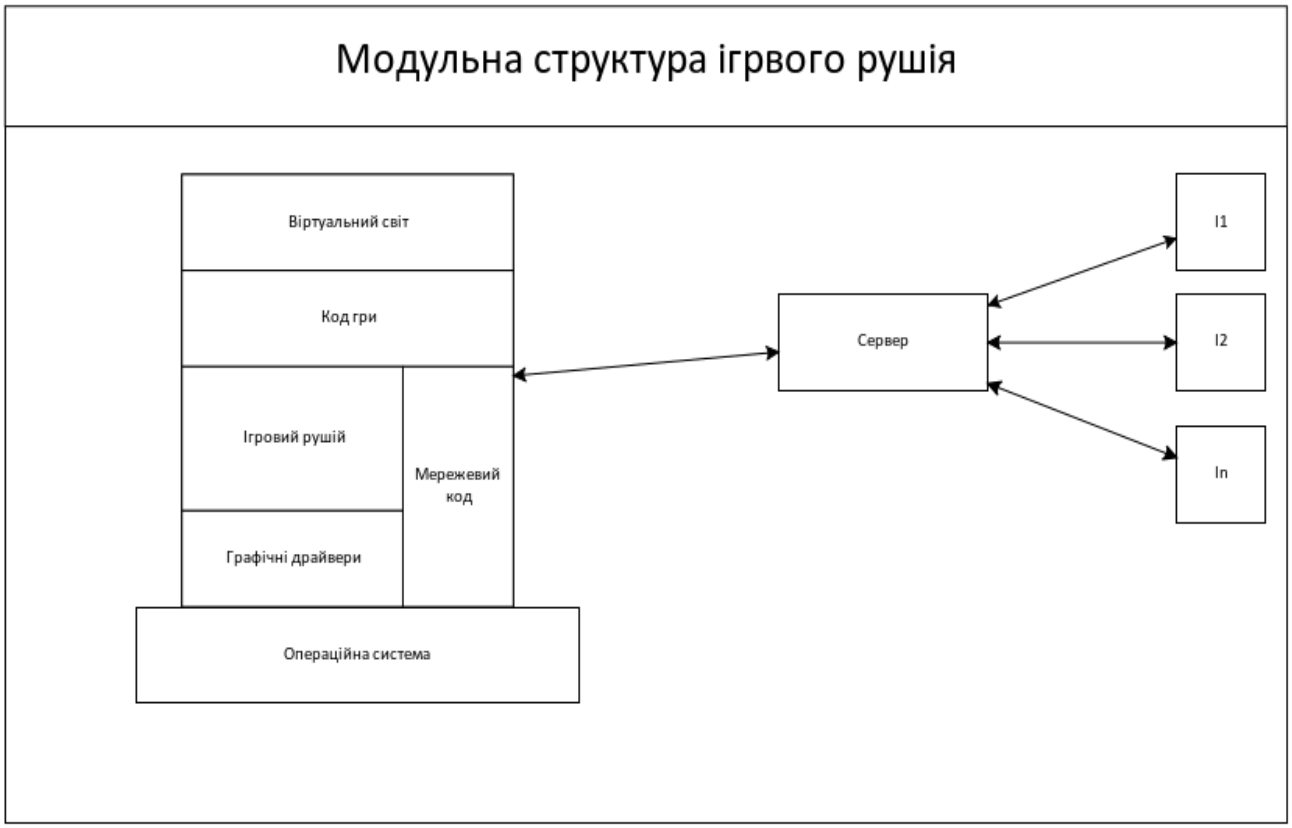

Рисунок 1.1 – Модульна структура ігрового рушія

На верхньому рівні знаходяться віртуальні світи або сценарії, з якими гравці взаємодіють. Вони мають ширший діапазон зовнішніх виглядів і правил взаємодії, навіть різні типи імітованої фізики. Багато з них створено фанатами гри, які переважно використовують розширене середовище розробки програм, яке безкоштовно постачається разом із самою грою. Часто вони включають компоненти існуючих світів або побудовані з використанням знань, якими автори вільно обмінюються через Інтернет.

Рівень нижче – це ігровий код, який обробляє більшість базових механізмів самої гри, як-от проста фізика, параметри відображення, мережа та дії базового або атомарного рівня для анімації, як автономні агенти чи поведінка власного аватару гравця. Зміна гри на цьому рівні потребує більш детального знання її внутрішніх функцій і виконується за допомогою мови сценаріїв, специфічної для гри.

Рушій візуалізації є перлиною гри. Він включає в себе весь складний код, необхідний для ефективного визначення та візуалізації погляду гравця зі складної 3D-моделі середовища. Механізм візуалізації є власною чорною скринькою і не відкривається для будь-яких змін користувачами.

Мережевий код підтримує надійний вбудований мережевий протокол, схожий на DIS/HLA, який дозволяє кільком користувачам у віддалених місцях досліджувати та взаємодіяти в одному віртуальному середовищі (VE). Один комп'ютер діє як сервер або хост для середовища, а інші підтримують окремих користувачів. Мережевий код настільки ефективний і добре розроблений в Unreal Tournament і Quake III, що гравці в локальній мережі з низькою затримкою можуть розраховувати на синхронізацію, яка перевищує частоту кадрів їхніх дисплеїв [5].

Графічні драйвери перетворюють загальні запити від механізму візуалізації в базову графічну бібліотеку, використовуючи такі API, як DirectX, OpenGL та інші. Оскільки драйвери є відкритими, їх можна легко модифікувати для адаптації до нових типів дисплеїв.

Нарешті, сервер - це окремий процес, зазвичай на іншій машині, який зберігає інформацію про віртуальний світ, який він підтримує в даний момент. Він спілкується з ігровими клієнтами, якими користуються гравці, щоб підтримувати глобальну інформацію про спільне середовище, взаємодію гравців (наприклад, пошкодження) та інформацію про синхронізацію.

Наприклад, кожен гравець має свій доступ в спільне віртуальне середовище. Сцени, представлені гравцям, мають бути послідовними, тому якщо один гравець змінює стан якогось об'єкта, то цей об'єкт має існувати для всіх інших гравців і перебувати в тому самому стані.

Ігровий рушій «Unity».

Ігровий рушій «Unity» був вперше оголошений на всесвітній конференції розробників Apple у 2005 році, і з того часу він спричинив значні зміни в індустрії відеоігор.

По-перше, вона відзначилася своїми зусиллями розширити діапазон цільових платформ. Цей ігровий рушій може експортувати ігри на 15 різних платформ (з

однаковою кодовою базою): iOS, Android, Windows & Windows Store, Windows Phone 8, BlackBerry 10, OSX, Linux, PlayStation 3, PlayStation 4, PlayStation Vita, PlayStation Mobile, Xbox One, Xbox 360, Wii U та всі основні веб-браузери через власний веб-плеєр. Крім того, у випадку Wii U це єдиний офіційний набір для розробки програмного забезпечення (SDK).

Однак слід пам’ятати, що публікація для ігрових консолей зазвичай потребує ліцензії розробника від власника цієї системи, що часто коштує дорого та потребує попереднього портфоліо ігор. Також варто відзначити ціну Unity, оскільки цей ігровий рушій пропонується безкоштовно для незалежних розробників або компаній з річним оборотом, що не перевищує 100 000 доларів США. Також доступна версія Pro, яка включає ряд функцій для розширеного використання, таких як доступ нижчого рівня. до графічних процедур.

Unity складається з візуального редактора та IDE, що дозволяє швидко створювати прототипи. Прості сцени (без повної ігрової механіки) можна навіть виконати без рядка коду. Його робочий процес заснований на компонентах, тобто він є модульним і достатньо гнучким, щоб складну механіку можна було створити з менших фрагментів логіки.

На практиці базова ігрова сутність називається «GameObject». Сцена – це контейнер для всіх «GameObjects». Просторові властивості «GameObjects», графічна поведінка, ігрова механіка визначаються в різних компонентах. На додаток до вбудованих компонентів (для візуалізації, анімації, звуку) програміст може визначити більше логіки, кодуючи нові компоненти сценарію (усі вони розширюють базовий клас «MonoBehaviour») [6].

Збудований прототип можна протестувати безпосередньо у візуальному редакторі (з часом компіляції, майже нульовим), а зміни в кодовій базі, як правило, можна побачити в реальному часі. Сценарії в «Unity» можна виконувати на «UnityScript» (це спеціальна мова з синтаксисом, схожим на «JavaScript») або «C#» – рекомендований варіант. Всередині «Unity» прозоро використовує різне проміжне програмне забезпечення для підтримки своїх функцій, а саме «Mono» для сценаріїв, «DirectX» і «OpenGL» для візуалізації, «beäst» і «Substance» для графіки, «fmod»

для звуку, «PhysX» як фізичний механізм, «RakNet» для мереж. Крім того, «Unity» продовжує інтегрувати функції, що полегшують інтеграцію соціальних і внутрішньо ігрових аспектів монетизації в його структуру. Іншим аспектом, який робить «Unity» чудовим орієнтиром для початку розробки ігор, є розмір спільноти та наявність ринку, де можна обмінюватися ігровими ресурсами та компонентами (щоб прискорити розробку або використовувати як навчальний матеріал). «Unity» використовується як для невеликих мобільних ігор, так і для галузевих виробництв AAA, що демонструє його універсальність. Кілька прикладів, розроблених за допомогою Unity, включають «Temple Run», «Bad Piggies», «Monument Valley», «HearthStone» або «Wasteland 2».

Висновки за розділом 1

В даному розділі було проаналізовано як працює і з яких компонентів складаються ігрові рушії різного типу. Аналіз складових основних складових, які використовуються при створенні ігрових рушій дає розуміння, як саме відбувається створення ігор та процес реалізації в реальному часі.

Особлива увага було надано ігровому рушію «Unity», так як цей ігровий рушій є одним із найпопулярніших для використання і саме на цьому ігровому рушії було створено демонстраційну версію розробленого методу захисту авторського права в комп’ютерних іграх.

РОЗДІЛ 2
ОСНОВНІ ЗАГРОЗИ АВТОРСЬКОГО ПРАВА

2.1 Загрози для комп'ютерних ігор

Захист інтелектуальної власності (ІВ) на відеоігри за допомогою авторського права, патентів і торгових марок має ті ж проблеми, що й захист авторського права на програмне забезпечення як відносно нова сфера законодавства про ІВ. Сама індустрія відеоігор побудована на природі повторного використання ігрових концепцій із попередніх ігор для створення нових стилів гри, але обмежена незаконним прямим клонуванням існуючих ігор, що ускладнило визначення захисту інтелектуальної власності, оскільки це не постійне середовище.

Проблеми, пов'язані із захистом інтелектуальної власності у відеоіграх.

1. Створення гри.

Існує кілька аспектів створення відеоігор, які призвели до того, що різні компоненти програмного забезпечення захищаються авторським правом окремо. Деякі взагалі не можуть бути захищені авторським правом через те, що вони знаходяться у публічному просторі.

2. Загальні ресурси.

Існує безліч веб-сайтів, які дозволяють творцям «позичати» ресурси для впровадження в гру. Модель надання доступу та дозволу на використання цих активів відрізняється на різних веб-сайтах і може варіюватися від авансового платежу до частини прибутку (якщо активи використовуються в комерційних цілях). Це не проблема для великих видавців відеоігор (таких як EA, Activision або Sony), але коли ці автономні компанії створюють великі та деталізовані світи, більшість створених ними ресурсів використовуються лише один раз. Це обмежує публічний пул ресурсів/активів.

Активи часто доведеться відтворювати, щоб створити нову гру (або продовження іншої компанії чи творця), що є загальною скаргою розробників ігор,

оскільки зазвичай створення комерційно життєздатних активів коштує великих грошей і ускладнює це для невеликих розробників для створення ігор.

3. Ігрові рушії.

Компанії створюють ігрові рушії, щоб дозволити розробникам (платно чи безкоштовно) створювати ігри. Однак через обмеження чи певні особливості рушія деякі конструкції або пасивні фонові завдання можуть відбуватися особливим чином для кожної гри, створеної з використанням цього рушія. Ця особливість (незалежно від того, наскільки вона є центральною для гри) по суті однакова в усіх іграх, створених за допомогою цього механізму, що виводить її за межі авторського права [7].

Це міркування також буде застосовано до ігор, створених за допомогою того самого рушія, які мають спільний вихідний код – це не вважатиметься копіюванням, оскільки подібність ігор притаманна інструментам. Великі будинки/видавці відеоігор можуть обійти це обмеження, розробивши власні ігрові рушії.

4. Відносини між видавництвами та розробниками.

Традиційно видавцям належало забезпечити (або надати) фінансування для гри, а також нести збитки та, у багатьох випадках, рекламувати гру. Ці витрати, як правило, були великими і могли доходити до десятків мільйонів доларів для назв ААА. Але з появою Інтернету та зростанням інді-культури з'явилася нова хвиля технологій фінансування та розповсюдження.

Такі сайти, як Kickstarter і Indiegogo, дозволяють зацікавленим споживачам безпосередньо робити внесок у розробку гри, купуючи гру заздалегідь. Цифрові розповсюджувачі, такі як Steam і GOG.com, усунули дорогу потребу у виробництві та розповсюдженні фізичних ігрових дисків. Ці нововведення дозволили галузі відійти від стандарту прав інтелектуальної власності, що належить видавцю, а не розробникам, оскільки «права інтелектуальної власності, які зазвичай надаються видавцям, тепер можуть бути спільно з видавцем або належати розробнику».

5. Геймплей.

Деякі елементи гри, захищені авторським правом, можуть бути створені через апаратні обмеження. Наприклад, класична гра Space Invaders, у яку спочатку грали

як аркаду, ставала швидшою, оскільки гравець вбивав більше прибульців на екрані, що звільняло системні ресурси.

6. Піратство відеоігор.

Піратство у відеоіграх – це несанкціоноване копіювання та розповсюдження програмного забезпечення для відеоігор, яке є формою порушення авторських прав. Це часто називають основною проблемою, з якою стикаються видавці відеоігор під час розповсюдження своїх продуктів, через легкість можливості розповсюджувати ігри безкоштовно, через торрент-завантаження або веб-сайти, що пропонують прямі посилання для завантаження.

Правовласники зазвичай намагаються протидіяти піратству своїх продуктів шляхом застосування Закону про захист авторських прав у цифрову епоху, хоча це ніколи не було повністю успішним. Цифрове розповсюдження піратських ігор історично відбувалося через системи дошок оголошень (BBS), а останнім часом через децентралізований одноранговий торрент. З точки зору фізичного розповсюдження, Тайвань, Китай і Малайзія відомі великими центрами виробництва та розповсюдження копій піратських ігор, тоді як Гонконг і Сінгапур є основними імпортерами [8].

2.2 Методи захисту комп’ютерних ігор

Антипіратські заходи.

Використання захисту від копіювання було звичним явищем протягом усієї історії відеоігор. Ранні заходи захисту від копіювання для відеоігор включали Lenslok, кодові колеса та спеціальні інструкції, які вимагали від гравця володіння посібником. Декілька ранніх заходів захисту від копіювання піддавалися критиці як за їх неефективність у запобіганні піратству, так і за незручність для гравця.

Одним із найбільш типових засобів захисту від копіювання є призначення серійного ключа кожній легітимній копії гри, щоб її можна було активувати, лише ввівши серійний номер. Однак це часто вдається обійти за допомогою злому програмного забезпечення або використання кейгена.

Генератор ключів (кейген) – це комп’ютерна програма, яка генерує ліцензійний ключ продукту, наприклад серійний номер, необхідний для активації для використання програмного забезпечення. Виробники програмного забезпечення можуть законно розповсюджувати кейгени для ліцензування програмного забезпечення в комерційному середовищі, де програмне забезпечення було ліцензовано масово для всього сайту чи підприємства, або вони можуть розповсюджуватися незаконно за обставин порушення авторських прав чи піратства програмного забезпечення [9].

Генератори нелегітимних ключів зазвичай розповсюджуються програмними зломщиками. Ці кейгени часто відтворюють «музику Keygen», яка може включати жанри дабстеп або чіптюн у фоновому режимі та мати художній інтерфейс користувача.

Останні спроби перешкодити піратству включали інструменти керування цифровими правами. Формою цього є продаж ігор на платформах цифрового розповсюдження, таких як Epic Games Store, Blizzard's Battle.net і Steam. Steam пропонує власні функції, такі як прискорене завантаження, збереження в хмарі, автоматичне встановлення виправлень і досягнення, яких немає в піратських копіях.

Мета цих функцій – зробити піратство менш привабливим і стимулювати законне придбання ігор. Гейб Ньюелл, творець Steam, заявив, що створення «цінності послуги» перешкоджає піратству більше, ніж додавання додаткового DRM. Деякі ігри, як-от Grand Theft Auto IV, використовують DRM, який негативно змінює процес гри, якщо виявляє, що гра є нелегітимною копією. У випадку з GTA IV він вимикає гальма на автомобілях і дає камері посилений ефект сп’яніння, що значно ускладнює ігровий процес, таким чином створюючи стимул для законної покупки гри.

Іноді ігри вимагають онлайн-автентифікації або постійно ввімкненого DRM. У 2021 році стався відомий інцидент із постійним DRM під час випуску Crash Bandicoot 4. Без постійного підключення до Інтернету, DRM гри взагалі забороняє будь-яку гру, навіть в одиночній грі, що, природно, викликало незадоволення

гравців. Однак група Warez зламала цю функцію DRM майже одразу. Зламана версія Crash Bandicoot 4, позбавлена цього DRM, почала циркулювати в мережі лише через день після офіційного релізу.

DRM означає керування цифровими правами. Коли ви купуєте будь-яку форму цифрового контенту, зазвичай ви насправді не купуєте вміст. Навпаки, ви купуєте ліцензію на використання цього вмісту на певній платформі. Наприклад, Steam є платформою DRM [10].

Щоразу, коли ви купуєте гру в Steam, ліцензія на завантаження та відтворення цієї гри додається до вашого облікового запису, і ви можете будь-коли скористатися цією ліцензією. Консолі нічим не відрізняються. PlayStation Store і Microsoft Store (раніше Xbox Marketplace) є платформами DRM.

Видавці використовують DRM для боротьби з піратством. Коли гравець запускає гру, платформа DRM перевіряє наявність ліцензії, пов’язаної з обліковим записом гравця, і якщо знаходить відповідність, запускає гру. Якщо ліцензія не знайдена, гра не запускається.

У крайніх випадках для DRM потрібне постійне активне підключення до Інтернету. На щастя, більшість видавців відмовилися від цієї тактики, особливо після катастрофічного запуску Diablo 3 і Sim City 2013 року.

Без платформи DRM користувач просто мали б доступ до будь-якої гри, незалежно від того, як був отриманий до неї доступ. Наприклад, якщо видавець випустив комп’ютерну гру без DRM, покупці можуть взяти виконуваний файл, який встановлює гру, і поділитися ним в Інтернеті.

Однак якщо видавець запускає гру з платформою DRM, покупці не зможуть поділитися грою в Інтернеті. Навіть якщо хтось вирішив завантажити всі файли гри, перевірка ліцензії не буде дійсною.

Незважаючи на те, що DRM легко відкинути як обмежувальний захід проти конфіденційності, є й плюси. Колекційні картки Steam, трофеї PlayStation і результати геймерів Xbox багато в чому завдячують DRM. Ці функції не пов’язані безпосередньо з керуванням вашими іграми чи ліцензіями, але вони надаються

шляхом обмеження способу розповсюдження ігор. За всю історію розробки ігор, було створено велику кількість методів захисту авторських прав типу DRM [10].

Перевірка диска.

Також відомий як перевірка CD/DVD, це старіша форма DRM, яка стає менш поширеною, оскільки комп'ютерні ігри переходять на цифрове розповсюдження. Такий тип DRM має наступні можливості:

- гра не запускатиметься, якщо відповідний CD/DVD не буде присутній у дисководі, і/або якщо буде виявлено певний файл, наявний лише на диску;
- ця система працюватиме незалежно від того, чи встановлено весь вміст гри на жорсткому диску.

Таку систему захисту можливо додатково поєднувати з іншими методами:

- фіктивні файли – вставлення фіктивних файлів, які вказують на сегменти інших файлів, що призводить до значного збільшення файлів під час їх копіювання;
- навмисні помилки диска – пошкодження диска під час виробництва, щоб помилка під час читання сектора підтвердила, що гра законна.

CD-ключ

Також відомий як серійний ключ або ключ продукту. Гра постачається з унікальним кодом (часто друкується в посібнику), який користувач повинен ввести, щоб завершити встановлення. Деякі ігри потребують ключа лише для багатокористувацького доступу, а не для всієї гри. Таким чином інсталятор може перевірити, чи була гра незаконно скопійована.

Онлайн активація

Потрібне підключення до Інтернету, щоб сповіщати власника прав про кожну установку гри. Таким чином, видавець може відстежувати, коли гру було встановлено вперше та скільки разів її встановлювали після цього.

На основі облікового запису

Коли гру придбано чи активовано, її копія прив’язується до певної адреси електронної пошти чи облікового запису, а отже, вимагає онлайн-активації. Ці служби часто дозволяють необмежену кількість активацій продуктів.

Обліковий запис можна використовувати лише на одному комп’ютері одночасно, тобто одна копія не може бути активною на кількох ПК одночасно. Ігри ніколи не можуть бути прив’язані до облікового запису користувача, тобто ними не можна торгувати чи продавати.

Постійне підключення до глобальної мережі

Деякі ігри потребують підключення до глобальної мережі лише під час кожного запуску. Наразі ці ігри не підходять до категорії «Постійне підключення». Щоб грати в гру, користувач повинен залишатися підключеним до Інтернету протягом усього сеансу. Будь-яка втрата з’єднання призведе до виходу гравця з гри після попередньо визначеної тривалості простою.

2.3 Проблематика захисту комп’ютерних ігор

Ризики DRM для ігор

Вилучені ігри – це архів усіх ігор, які протягом багатьох років були видалені з цифрових ринків. Станом на кінець 2020 року в каталозі наразі є 1097 таких випадків, починаючи від PlayStation 2 і закінчуючи Nintendo Switch. Коли гру вилучено з цифрового ринку, користувач більше не має можливості купити цю гру на відповідному ринку. Однак, якщо у користувача вже є ліцензія, він може продовжувати завантажувати цю гру та грати в неї (за винятком будь-яких мережевих обмежень).

Однак видалення зі списку може завдати шкоди, особливо для цифрових ігор і ігор з обмеженим фізичним випуском. Наприклад, Transformers Devastation - це назва з Platinum Games (Nier: Automata, Astral Chain), яка була виключена з біржі кілька років тому. Хоча є можливість знайти фізичні копії на Xbox One і PS4, більше немає можливості купити гру та грати в гру на персональному комп’ютері. Ще один

сумнозвісний приклад – «Назад у майбутнє» Telltale. Технічно доступні фізичні копії. Однак вони доступні лише в уживаному вигляді та продаються приблизно за 60 доларів навіть на таких консолях, як PlayStation 3 і Wii.

Основний страх платформ DRM полягає в тому, що вони просто закриються. Якби, наприклад, Steam припинив роботу або Microsoft вирішила залишити консольну гонку і усі цифрові ігри зникли б. Можна побачити, як це траплялось з іншими цифровими товарами раніше.

Технічні проблеми

З технічної точки зору, здається неможливим повністю заборонити користувачам робити копії носіїв, які вони купують, доки доступний «записувач», який може записувати на порожні носії. Для всіх типів носіїв інформації потрібен «програвач» – програвач компакт-дисків, DVD-програвач, відеоплеєр, комп'ютер або ігрова приставка, який повинен мати можливість читати медіафайли, щоб відобразити їх людині. Логічно можна створити програвач, який читає медіафайли, а потім записує точну копію прочитаного на той самий тип носія.

Як мінімум, цифровий захист від копіювання неінтерактивних творів підпадає під аналоговий отвір: незалежно від будь-яких цифрових обмежень, якщо музику чує людське вухо, її також можна записувати (принаймні, за допомогою мікрофона та магнітофон); якщо фільм можна переглянути людським оком, його також можна записати (принаймні, за допомогою відеокамери та диктофона).

На практиці майже ідеальні копії зазвичай можна зробити, торкнувшись аналогового виходу програвача (наприклад, виходу динаміків або роз'ємів для навушників), і після перецифрування в незахищену форму дублювати на невизначений термін. Копіювання текстового вмісту таким чином є більш нудним, але застосовується той самий принцип: якщо його можна роздрукувати або відобразити, його також можна відсканувати та розпізнати.

Маючи базове програмне забезпечення та трохи терпіння, ці методи можуть застосовуватися типовим комп'ютерно-грамотним користувачем.

Оскільки ці базові технічні факти існують, випливає, що цілеспрямована особа безперечно досягне успіху в копіюванні будь-якого носія, маючи достатньо часу та

ресурсів. Видавці це розуміють; Захист від копіювання призначений не для припинення професійних операцій, пов’язаних із несанкціонованим масовим тиражуванням медіафайлів, а для припинення уповільнення виходу таких копій у вільний доступ., а систему розшифровки можна зробити захищеною від втручання.

Висновки за розділом 2

В цьому розділі були розглянуті проблеми захисту авторського права в комп’ютерних іграх. Також було проаналізовано існуючі методи захисту, а саме системи DRM, їх можливості, типи і зміст їх роботи. DRM системи не тільки захищають ігри від нелегального копіювання і розповсюдження, а і додають нові функції які заохочують користувачів купляти ліцензії для використання продукту. На основі такого аналізу і було розроблено метод захисту авторського права.

РОЗДІЛ 3
РОЗРОБКА МЕТОДУ ЗАХИСТУ АВТОРСЬКИХ ПРАВ В КОМП'ЮТЕРНИХ ІГРАХ

3.1 Вибір програмного середовища

Так як даний метод захисту авторського права в комп'ютерних іграх складається з декількох частин, які обмінюються даними одна з одної, то і вибір програмного середовища складався з двох етапів:

1) вибір середовища для розробки комп'ютерної гри;
2) вибір середовища для розробки серверної частини захисту авторського права.

Для середовища розробки комп'ютерної гри було вибрано ігровий рушій «Unity». «Unity» – це крос платформний ігровий рушій, розроблений «Unity Technologies», вперше анонсований і випущений у червні 2005 року на Всесвітній конференції розробників Apple як ігровий рушій для Mac OS X. З тих пір цей рушій поступово розширювався для підтримки різноманітних настільних комп'ютерів, мобільних пристроїв, консолей і платформ віртуальної реальності.

Unity дає користувачам можливість створювати ігри та досвід як у 2D, так і в 3D, а механізм пропонує основний API сценаріїв у C# з використанням Mono як для редактора Unity у формі плагінів, так і для самих ігор, а також функціональність «drag and drop».

Він особливо популярний для розробки мобільних ігор для iOS та Android, вважається простим у використанні для розробників-початківців і популярний для розробки незалежних ігор.

По критеріям легкості розробки та популярності використання був обраний саме цей ігровий рушій. На ньому було розроблено базові ігрові функцію та створено основну частину системи захисту авторського права для комп'ютерних ігор.

Наступним етапом став вибір середовища для розробки саме серверної частини, так як основна ідея методу захисту складається в комунікації клієнта який був створений на «Unity» та сервером, який і буде постачати клієнт потрібними файлами. Серверна частина складається з операційної системи «Ubuntu» і встановленого на неї веб фреймворку «Django».

Так як основна частина розробленого методу складається з клієнтської складової, то для зручності розробки та легкості обміну файлів було вибрано веб-платформу «Django».

«Django» – це безкоштовна веб-платформа з відкритим вихідним кодом на основі Python, яка відповідає архітектурному шаблону модель–шаблон–подання. Головна мета «Django» – полегшити створення складних веб-сайтів, керованих базами даних. Фреймворк наголошує на багаторазовому використанні та «підключеності» компонентів, меншій кількості коду, низькому зв’язку, швидкому розвитку та принципі «не повторюйся».

3.2 Створення методів захисту та їх варіацій

Основний метод який створений для захисту авторського права в комп’ютерних іграх складається з двох частин та має декілька варіацій та доповнень при використанні його в різних середовищах та на різних платформах, так як не всі ігрові рушії можуть використовувати деякі функції. Наприклад для ігрових рушіїв які мають функцію додавання виконавчих файлів в режимі реального часу, не перебудовуючи основного змісту чи без перезавантаження можуть використовувати додаткові модулі основного методу.

Серверна частина методу складається з веб-платформи, на якій зберігаються файли (текстові файли, зображення, звуки і т.ін.) і можуть бути отриманні при запиті клієнтської частини.

Частина методу яка розташована на стороні клієнту має змогу завантажувати файли з серверу і додавати їх для використання безпосередньо в самому процесі виконання гри і після використання, повністю видаляти такі файли з пристрою.

На рис. 3.1 можна побачити схему такого методу захисту. Основна ідея такого захисту надає розробникам можливість не зберігати повністю всю гру на пристрої гравця, і тим самим захистити продукт від незаконного копіювання. При такому захисті також можна використовувати доповнення такі, як шифрування, використання унікального ідентифікатора пристрою, як ключ для шифрування, перевірка при запиті до файлу додаткові складові пристрою для зменшення вірогідності підробки ідентифікаторів.

Також якщо у користувача при деяких особливостях пристрою чи середовища в якому виконується гра з’явиться помилка при якій гра не зможе завантажити потрібні їй файлі, повинна виконуватися діагностика, яка покаже де саме трапляється помилка та її основна причина. Основними причинами в такому випадку можуть бути:

1) відсутність підключення до глобальної мережі;
2) відсутність підключення до серверу гри;
3) правила брандмауера на пристрої (міжмережевий екран);
4) конфлікт з іншими застосунками;
5) не вірний ідентифікатор (при наявності системи захисту на сервері якій потребує ідентифікатору кожен раз коли клієнт створює запит);
6) завантажений файл був пошкоджений при завантажені.

Якщо діагностика не допомогла виявити причину, то повинен створюватися звіт, який надсилається на сервер (при наявності підключення до глобальної мережі).

На цій схемі показано основну логіку такого захисту, але є варіанти коли її можна чи спростити для використання меншої кількості пам’яті фізичного диску, чи додати додаткові заходи для підвищення рівня захисту. На рисунках 3.2-3.3 показано такі варіації які можуть бути використані в одній грі але при різних умовах.

В деяких випадках файл який завантажується з серверу може не створюватися завдяки тому, що він використовується безпосередньо після завантаження і не потрібен більше в наступних функціях, чи файл який може використовуватися

декілька разів, але розмір такого файлу достатньо малий для того щоб можна було його завантажити швидко для наступного використання.

Рисунок 3.1 – Основна схема методу захисту авторського права в комп’ютерних іграх

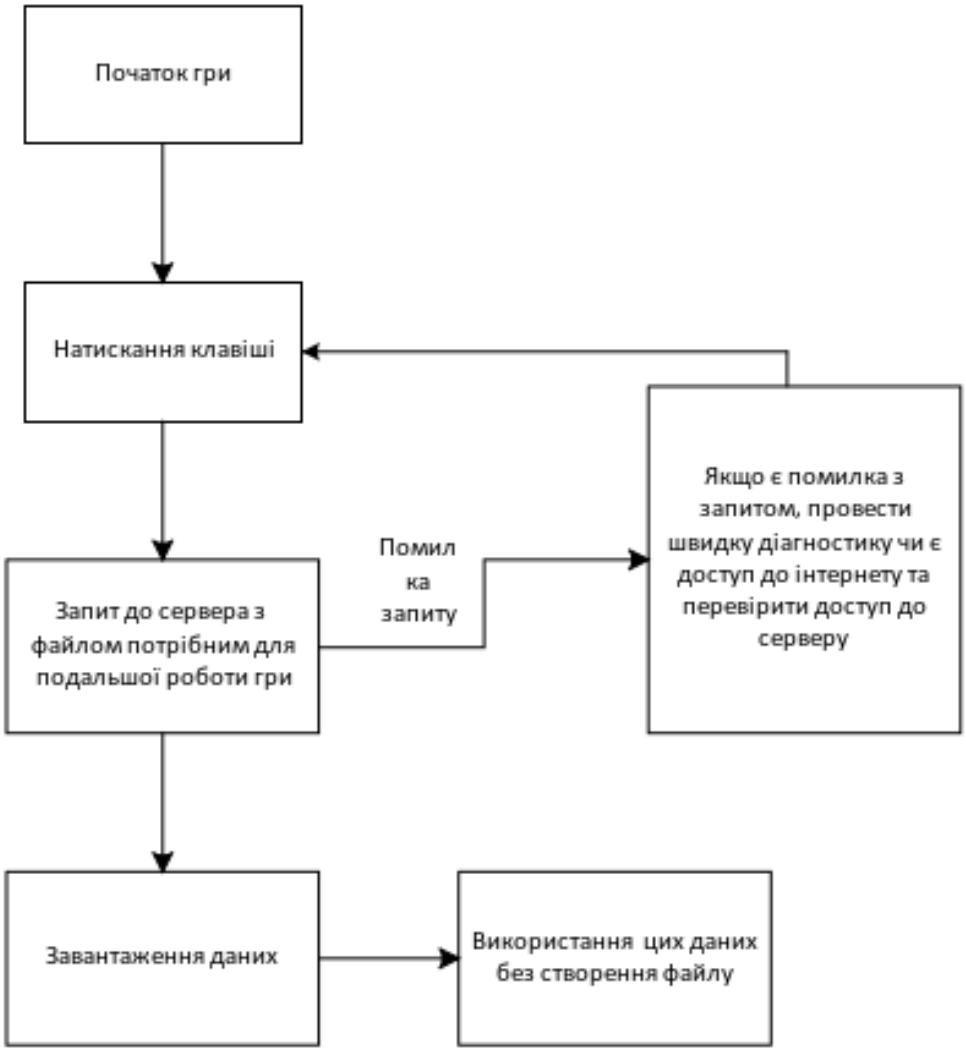

Рисунок 3.2 – Схема методу захисту без створення файлу у фізичній пам’яті пристроя

Такими файлами можуть бути конфігурації гравців, текстові файли з діалогами, якісь невеликі об’єкти чи функції. На рисунку 3.3 можна побачити, що до основної схеми додається ще й шифрування файлів на сервері.

При завантажені таких файлів, потрібно їх зберегти та розшифрувати у новий файл. Але якщо цей файл має невеликий розмір, або буде використаний лише один раз, то розшифровані данні можна не зберігати у новий для зменшення використання фізичної пам’яті пристрою.

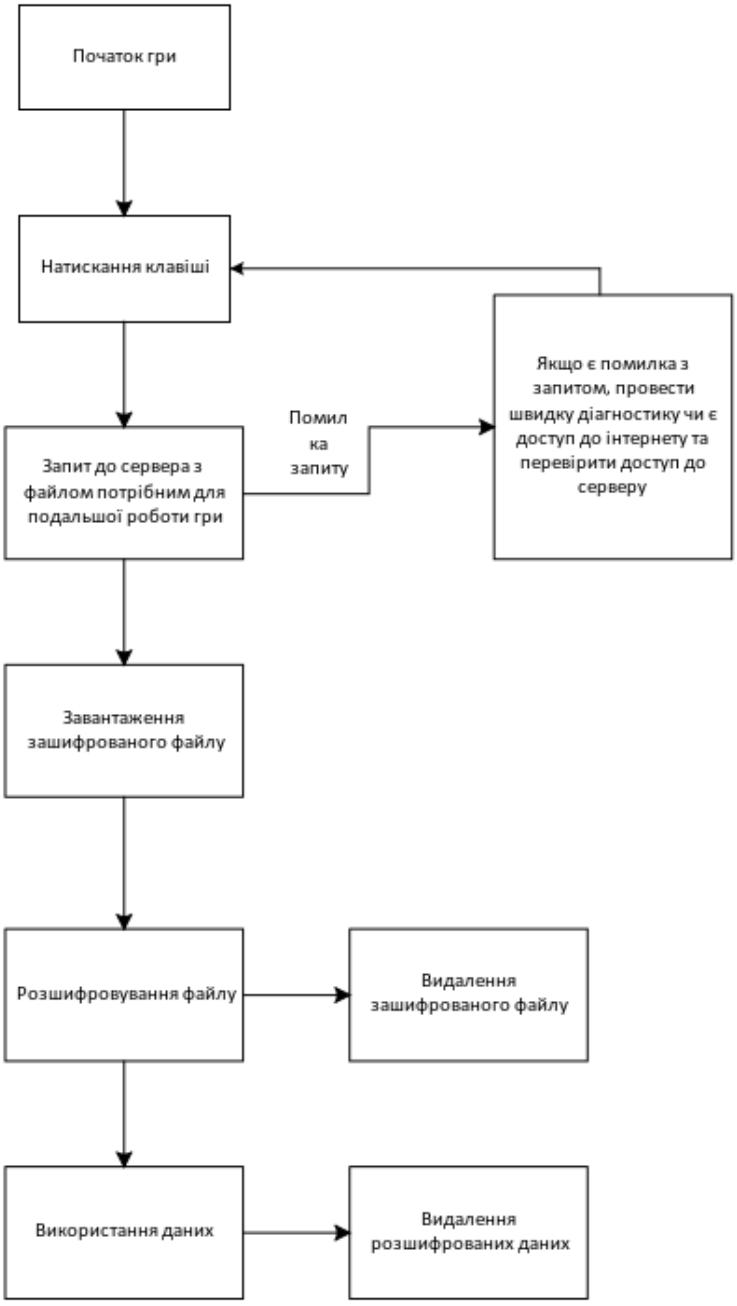

Рисунок 3.3 – Схема методу захисту з використання шифрування

Так, як досі не створили методу захисту який не можливо зламати, то сучасні методи захисту ставлять собі за ціль максимально подовжити час злому. При дослідженнях ігор та заробітку при їх продажах, більша кількість покупок припадає на перші два місяці продажу. Тому і розроблений метод в цій роботі не відійшов від цих цілей.

Через особливість такого захисту, щоб завантажити всі файли на комп’ютер хоча б раз, треба виконати велику кількість різних дій в самій грі для того, щоб файли які перебувають на сервері були потрібні для виконання певних умов у грі. В залежності від об’єму гри і об’єму файлів які знаходяться на сервері ефективність даного методу буде змінюватися.

3.3 Побудування інфраструктури програмного забезпечення

Для побудування клієнт серверної структури, було створено віртуальну машину за допомогою утиліти «VMware Workstation Pro». На віртуальну машину було встановлено операційну систему «Ubuntu 22.04.1 LTS». Для роботи веб-платформи «Django» був встановлений модуль програмної мови «Python3». Та для зручної розробки був встановлений сет команд для налаштування мережевої складової операційної системи «net-tools».

Після встановлення «Django» було створено проект «django_project», в якому було додано новий додаток в якому і зберігається вся логіка виконання обробки запитів на завантаження файлів.

На рис. 3.4 можна побачити функцію яка відповідає за завантаження файлів з серверу. Для зручності розробки та перевірки системи захисту, назва файлу і шлях до нього вказані у вигляді змінної. Функція зчитує файл з вказаного шляху і передає зміст як відповідь на «http» запит «GET» клієнтської частини.

Після встановлення серверної частини і завантаження необхідних файлів на неї, було створено безпосередньо тестову версію комп’ютерної гри, де і було додано клієнтську частину захисту авторського права.

При створені гри було обрано «2D» шаблон для швидкої роботи, так як такий варіант шаблону займає менше місця і використовує для роботи менше оперативної пам’яті та значно знижує вплив на центральний процесор та відеокарту пристрою.

Реалізований метод включає в себе декілька створених модулів які працюють разом при використанні захисту у дії.

Для роботи з файловою системою було створено модуль «FileReadWrite». Цей модуль включає в себе:

1) створення файлів,
2) видалення файлів,
3) створення тимчасових файлів,
4) автоматичне пошук та видалення тимчасових файлів,
5) перезапис файлів,
6) отримання змісту файлу,
7) створення файлу за допомогою масиву байтів (розроблено для роботи з шифрованими файлами для того, щоб стандартне кодування не змінило змісту такого файлу).

```
def download_file1(request):
    BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
    filename = 'rawtext.txt'
    filepath = BASE_DIR + '/filedownload/Files/' + filename
    path = open(filepath, 'r')
    mime_type, _ = mimetypes.guess_type(filepath)
    response = HttpResponse(path, content_type=mime_type)
    response['Content-Disposition'] = "attachment; filename=%s" % filename
    return response
```

Рисунок 3.4 – Функція завантаження файлів з серверу

Для роботи з криптографічної складовою методу захисту авторського права було створено модуль «EncryptionDiploma» [11]. Цей модуль включає в себе дві функції такі, як шифрування даних, та розшифрування. Так як шифрування даних в оперативній пам’яті може займати досить велике місце, ці функції було створені для використання саме шифрування і розшифрування файлів, які знаходяться на фізичному носії пристрою.

Для отримання даних з сервера, було створено модуль «WebRequestsDiploma», який включає в себе завантаження даних текстового формату, формату який при завантажені не змінює за допомогою кодування зміст і зберігає його в масиві байтів, та завантаження зображення з серверу та зміни його в формат «Texture2D» для використання безпосередньо в процесі гри.

Всі ці модулі можуть використовуватися в будь-якій частині гри, але треба зауважити, що функція завантаження файлів виконується асинхронно. В середовищі ігрового рушія «Unity» замість асинхронних функцій використовуються співпрограми, які мають схоже значення.

Співпрограми – це компоненти комп'ютерної програми, які дозволяють призупиняти та відновлювати виконання, узагальнюючи підпрограми для спільної багатозадачності. Співпрограми добре підходять для впровадження знайомих програмних компонентів, таких як кооперативні завдання, винятки, цикли подій, ітератори, нескінченні списки та канали. Тому дані які потрібно завантажити з серверу, можна використовувати лише після деякого часу. Цей час повністю залежить від деяких факторів, таких як об'єм файлу, швидкість інтернет з'єднання, шифрування файлу (так як на розшифровування файлу піде більше часу ніж при звичайному завантажені) і т.ін.

Для тестування такої систему захисту було створено чотири різних варіанти при яких використовуються різні варіації, але в одній системі під час процесу гри без перезавантажень. На рисунку 3.5 надано схему яка демонструє процес обробки інформації першим тестовим варіантом.

Після запуску гри, гравець виконує дію, це може будь-яка дія, натиск клавіші на клавіатурі, на миші чи на сенсорному екрані, в залежності від пристрою.

Texture2d – це клас, який обробляє і працює з текстурами в програмній мові C#. Цей клас використовується для створення нових текстур, або модифікації вже існуючих.

Після виконання певної дії починається завантажування файлу, яке непомітне для гравця і ніяк не впливає на загальну цілісність гри. Після завантаження інформація зберігається в створену зміну в оперативній пам'яті і використовується через деякий час. В даному випадку ця функція по закінченню завантаження змінює текст в користувацькому інтерфейсі на отриманий з серверу.

Після зміни тексту, отримані данні видаляється з оперативної пам'яті і гра продовжує свою роботу по створеному сценарію.

На рис. 3.6 подано схему коли з серверу завантажується зашифрований файл.

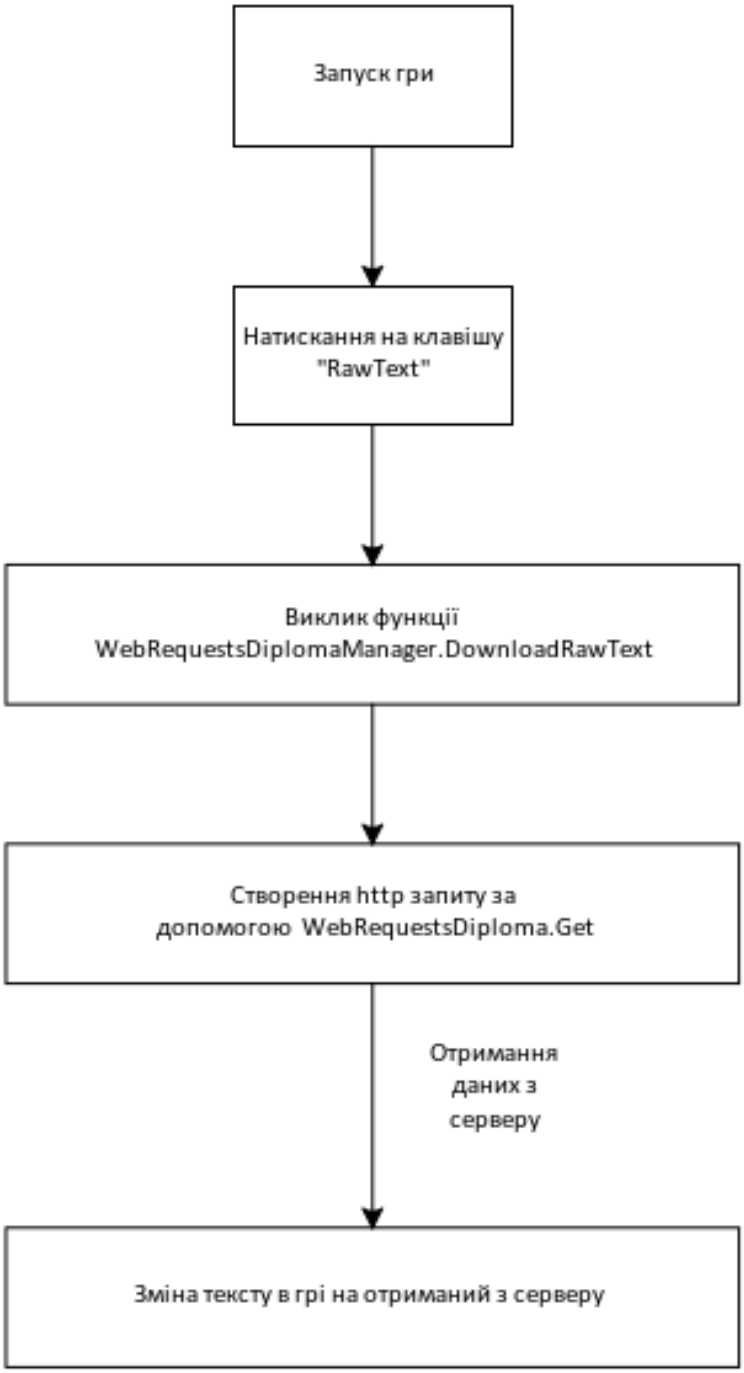

Рисунок 3.5 – Схема роботи завантаження звичайного тексту з сервера

Через те, що зашифрований файл не можна зчитати звичайним способом як тип даних «string», то вся передача і зчитування відбувається в форматі «byte».

При такій передачі даних, всі дані залишаться незмінними і при розшифруванні не відбудеться ніяких помилок.

Після завантаження, для розшифрування даних створюється файл. В цей файл записується масив байтів отриманий з серверу. Після обробки масиву байтів і запису в файл модуль шифрування розшифровує данні і перезаписує їх в новий файл, видаляючи старий зашифрований файл відразу після зчитування даних. Після запису в новий файл дані зберігаються до виконання функції яка і буде використовувати ці дані. Коли функція, яка використовує такі дані викликається, дані зчитуються з файлу, а сам файл видаляється з фізичного носія.

Для невеликих файлів є можливість не створювати окремі файли на фізичному носії і всі процеси проводити в оперативній пам’яті, але при таких навантаженнях є можливість перервати процес гри чи викликати занадто велике навантаження на систему. Тому створення файлу дозволяє не зберігати велику кількість даних в оперативній пам’яті, а використовувати замість цього фізичну пам’ять носія.

На рис. 3.7 демонструється завантаження більшого файлу по розміру а саме звичайного файлу з форматом «.png». Основні моменти схожі з завантаженням текстового файлу, але для того, щоб ігровий рушій «Unity» міг працювати і оброблювати файли формату «.png» потрібно змінити цей формат на формат «Texture2D».

Після перетворення файлу формату «.png» в «Texture2D» гра може використовувати файл і створювати з нього додаткові файли, змінювати вже існуючі об’єкти, замінювати одну текстуру на завантажену і т.ін.

Функція в цьому тестовому варіанті змінює зображення яке створене в користувацькому інтерфейсі на завантажене, після натискання клавіші. Так як це зображення займає не багато місця, то його можна не зберігати на фізичний носій, але це працює тільки за файлами які займають не велике місця, при використанні такого файлу не в одній функції, після якої цей файл буде видалено з оперативної пам’яті, а в багатьох, такий файл буде займати місце оперативної пам’яті достатньо довго.

Якщо в процесі гри треба буде завантажувати велику кількість таких файлів, то досить швидко можна використати всю оперативну пам’ять, що призведе до неприємних наслідків такі.

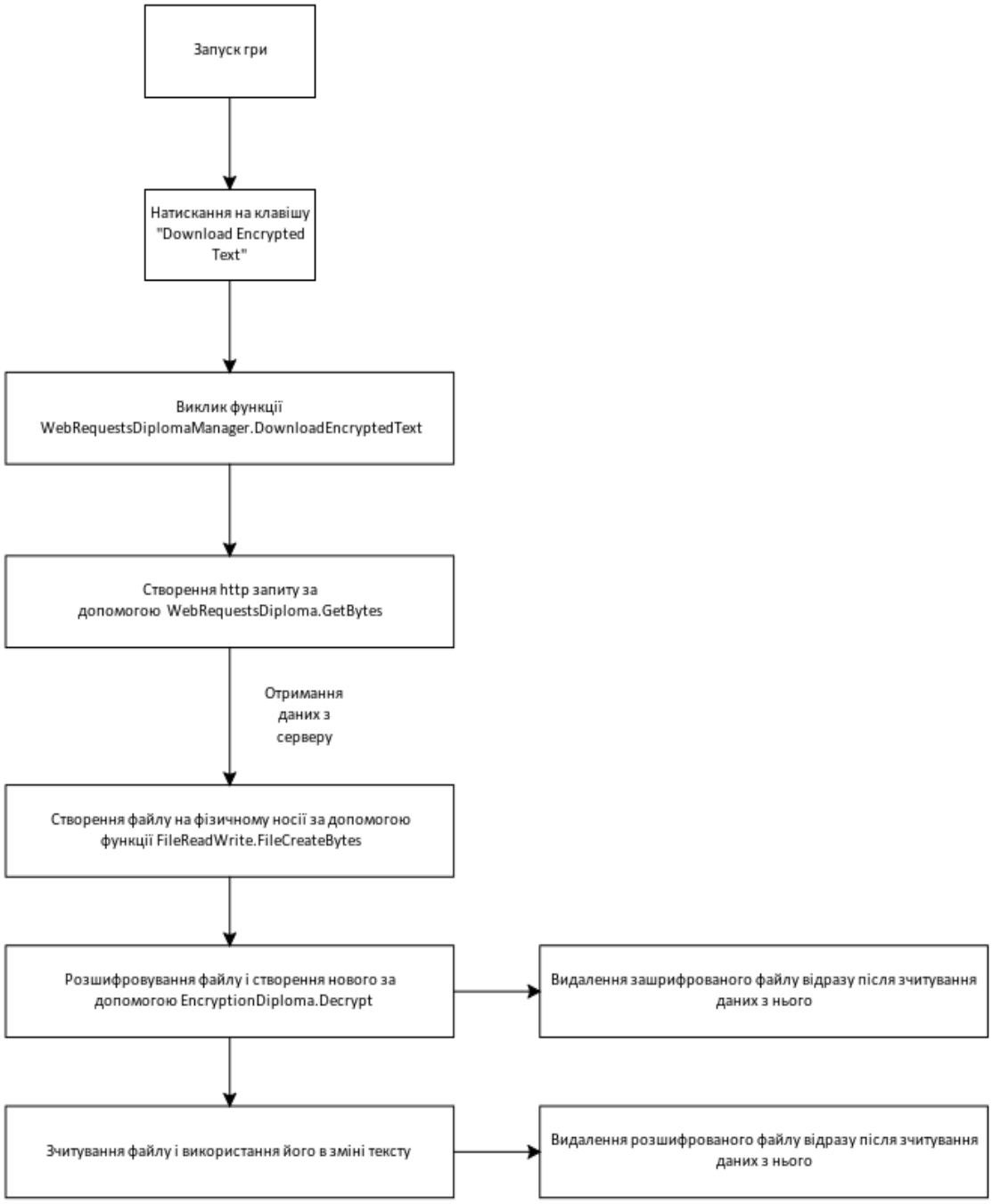

Рисунок 3.6 – Схема роботи завантаження зашифрованого тексту з сервера

Одним з наслідків є зупинка гри і вимкнення її, чи зниженню кількості кадрів до мінімального, тому що при недостатньому місці в оперативній пам’яті збільшується навантаження на процесор і т.ін. Тому такий варіант завантаження дуже зручний для невеликих файлів коли їх треба використати і видалити відразу після завантаження, і не рекомендований для використання при великому розмірі

файлів, або при багаторазовому використанні таких файлів під час довгого процесу гри.

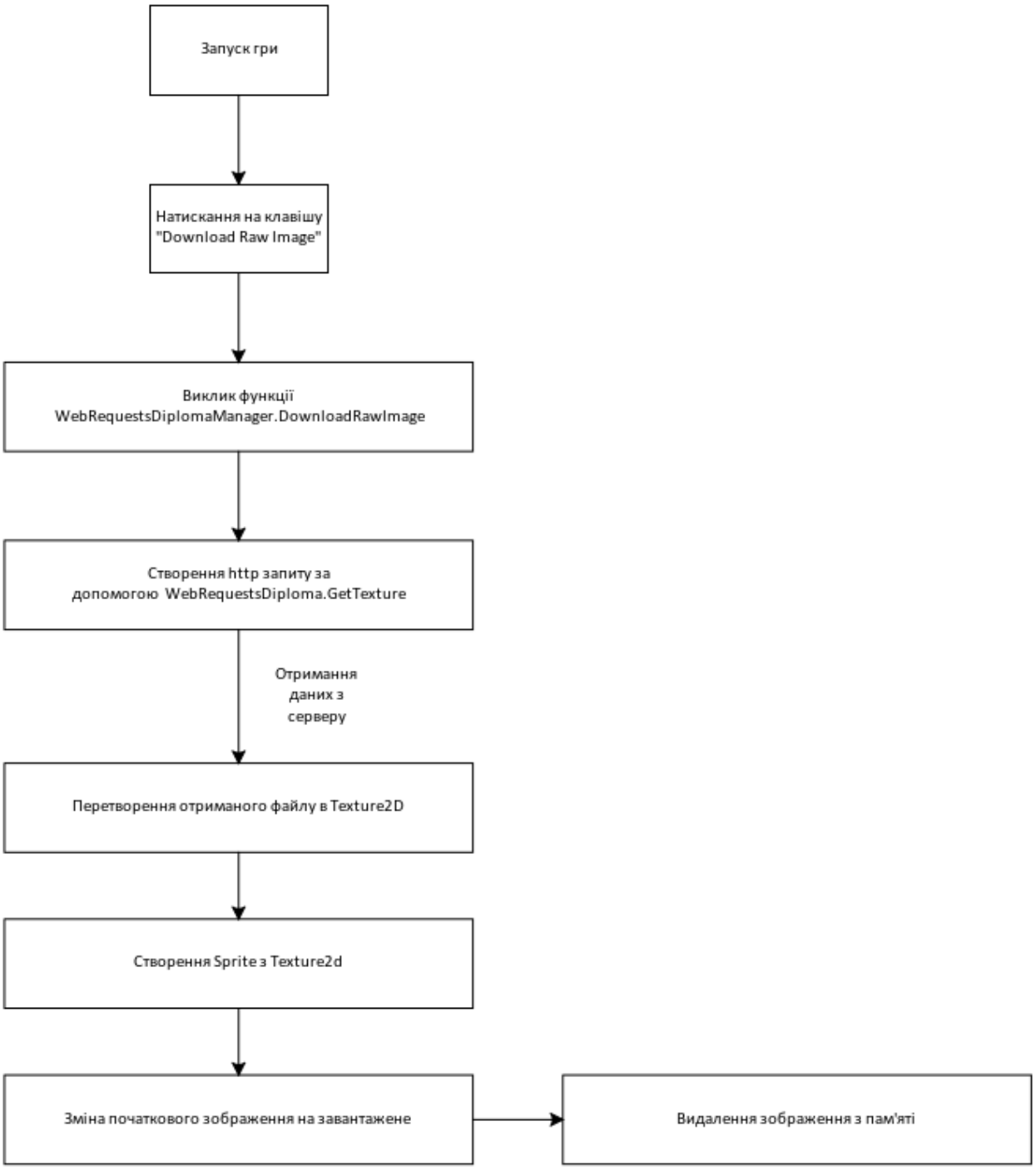

Рисунок 3.7 - Схема роботи завантаження звичайного файлу формату «.png» з сервера

У випадку цього тестового варіанту через особливість ігрового рушія «Unity» для заміни створеного зображення на завантажене треба створити окремий тип файлу «Sprite» (Спрайт).

Спрайти – це двовимірні графічні об’єкти, які використовуються для персонажів, реквізиту, снарядів та інших елементів двовимірного ігрового процесу. Графіка отримується з растрових зображень – «Texture2D». Клас «Sprite» насамперед визначає частину зображення, яку слід використовувати для конкретного спрайту. Потім ця інформація може бути використана компонентом «SpriteRenderer» на «GameObject» для фактичного відображення графіки.

При такому завантажені і використані зображення треба створити дві змінні які, будуть займати певне місце в оперативній пам’яті в залежності від розміру файлу, але після зміни зображення в користувацькому інтерфейсі ці змінні будуть видалені і місце яке вони займали в оперативній пам’яті буде очищено. Якщо таке зображення треба буде використовувати ще раз при наступних діях, то такий файл треба буде знову завантажувати.

Тому на рисунку 3.8 показано схему завантаження зашифрованого файлу з збереженням цих файлів на фізичний носій. Для завантаження зашифрованого файлу з серверу, як вже було згадано раніше, використовується завантаження масиву байтів для збереження цілісності файлу.

При роботі з зашифрованими файлами для збереження місця в оперативній пам’яті та для зручності використання, створюється файл в який записується масив байтів. За допомогою модуля «EncryptionDiploma.Decrypt» файл розшифровується і розшифровані данні зберігаються в новий файл, а файл з зашифрованими даними видаляється з фізичного носія.

Через особливість ігрового рушія «Unity», треба створити змінну типу «Texture2D» і вже з неї створювати файл типу «Sprite» для роботи з зображеннями.

Після перетворень даних з файлу в спрайт, розшифрований файл видаляється і в користувацькому інтерфейсі замінюється створене зображення у грі на завантажене. На рисунках 3.9-3.16 надано результати у вигляді скріншотів як, саме працюють ці тестові функції.

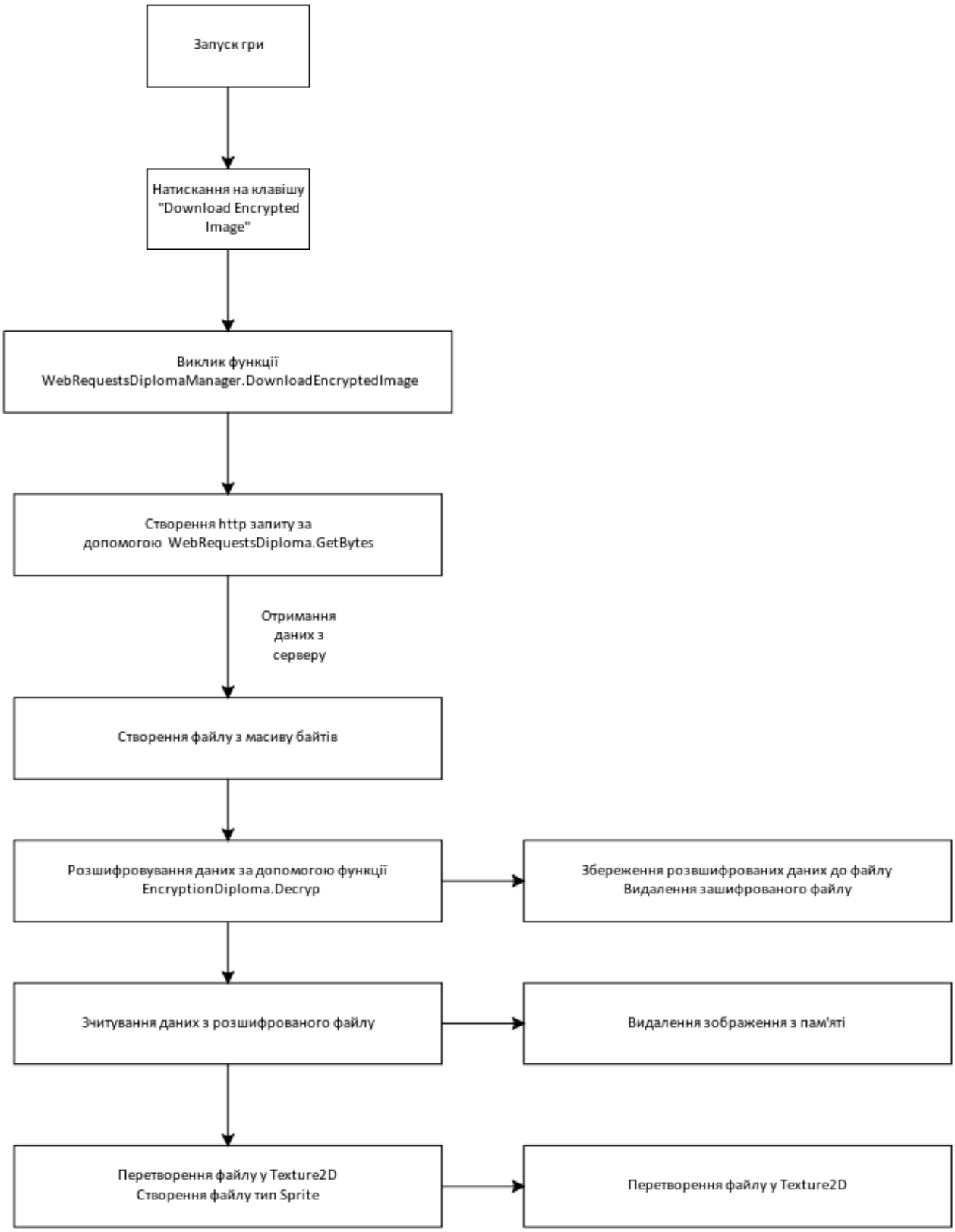

Рисунок 3.8 - Схема роботи завантаження зашифрованого файлу формату «.png» з сервера

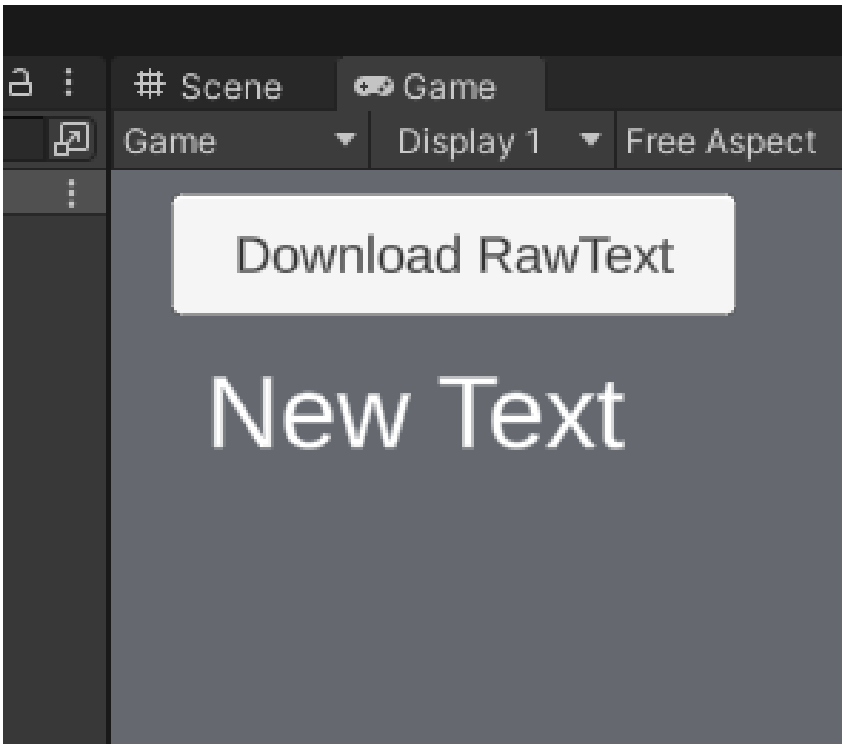

Рисунок 3.9 – Початковий вигляд елементів перед натисканням клавіші завантаження звичайного тексту

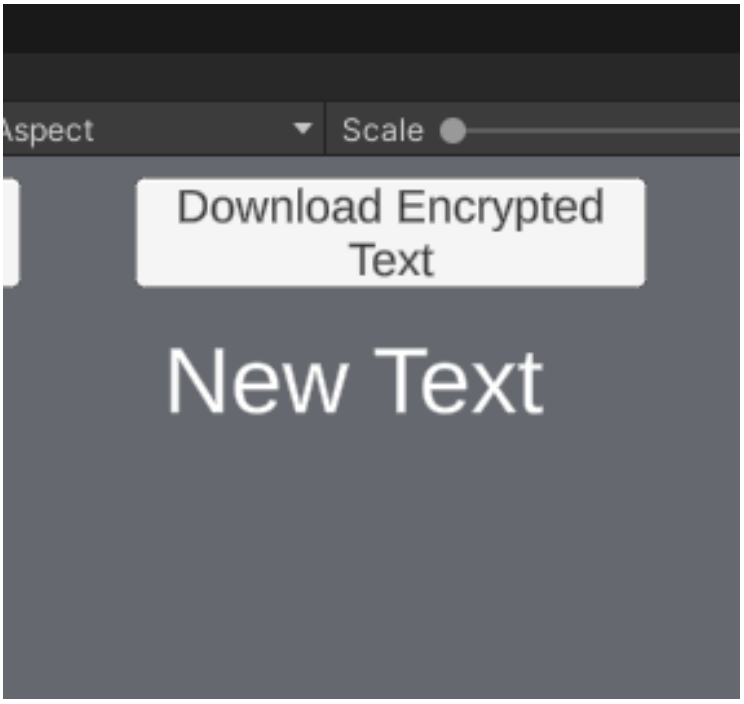

Рисунок 3.10 – Початковий вигляд елементів перед натисканням клавіші завантаження звичайного тексту

Рисунок 3.11 – Початковий вигляд елементів перед натисканням клавіші завантаження звичайного тексту

Рисунок 3.12 – Початковий вигляд елементів перед натисканням клавіші завантаження звичайного тексту

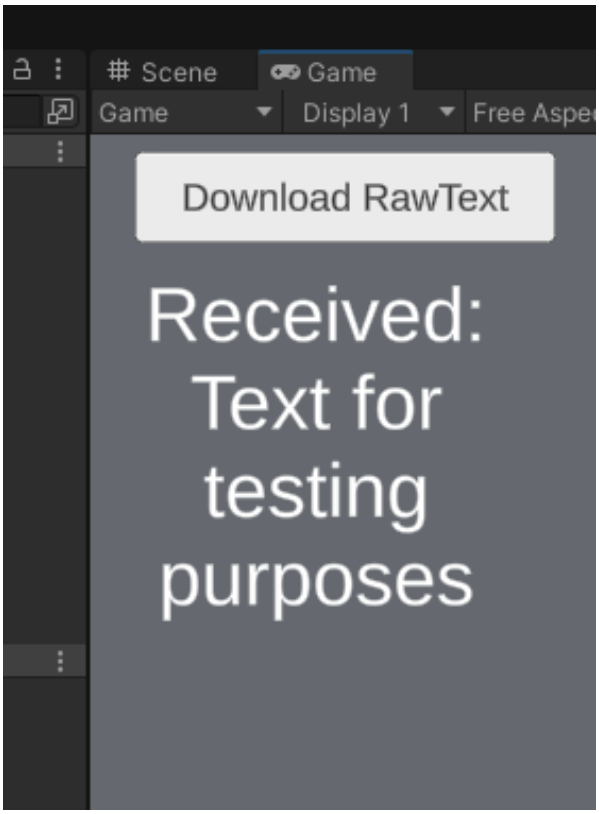

Рисунок 3.13 – Початковий вигляд елементів перед натисканням клавіші завантаження звичайного тексту

Рисунок 3.14 – Початковий вигляд елементів перед натисканням клавіші завантаження звичайного тексту

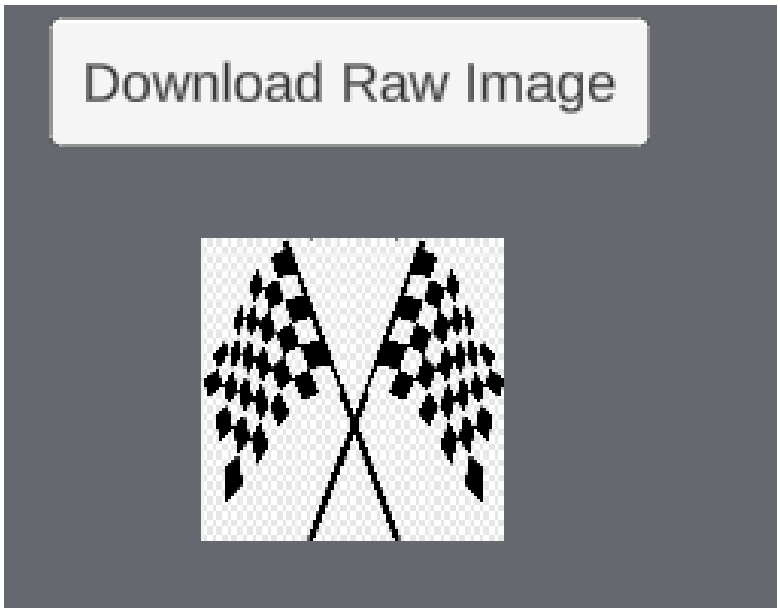

Рисунок 3.15 – Початковий вигляд елементів перед натисканням клавіші завантаження звичайного тексту

Рисунок 3.16 – Початковий вигляд елементів перед натисканням клавіші завантаження звичайного тексту

При завантажені можна спостерігати, що чіткість одного й того же зображення відрізняється в залежності від того, був чи зашифрований файл, чи ні. Це залежить від того, що при завантажені не зашифрованого файлу, створення текстури йде відразу після завантаження без створення файлу. При створені файлу ігровий рушій «Unity» створює стандартні налаштування для файлу типу «.png».

Щоб не втрачати чіткість зображення після розшифрування файлу треба змінити стандартні налаштування «Filter Mode» та «Compression». Самі ці показники і змінюють чіткість зображення при використанні в процесі гри.

Висновки за розділом 3

В даному розділі було розглянуто побудування схеми для метода захисту авторського права в комп'ютерних іграх. В цьому розділі було представлено чотири тестових варіанта захисту в створеній грі для демонстрації можливостей такого захисту.

Особливість такого захисту складається в тому, що вся гра ніколи не зберігається на стороні клієнта (гравця). Основна частина гри знаходиться на сервері. Під час процесу гри файли, які потрібні для проходження гри, надаються сервером тільки після запиту від клієнтської частини і не зберігаються після завантаження, а видаляються, що допомагає захистити проект від незаконного копіювання та модифікацій.

В розділі були розглянуті випадки коли клієнт завантажує звичайний текстовий файл, зашифрований текстовий файл, звичайний файл формату «.png», та зашифрований файл формату «.png».

РОЗДІЛ 4
РЕКОМЕНДАЦІЇ ЩОДО ЕКСПЛУАТАЦІЇ ТА ВПРОВАДЖЕННЯ МЕТОДУ ЗАХИСТУ АВТОРЬСКИХ ПРАВ В КОМП'ЮТЕРНИХ ІГРАХ

4.1 Впровадження методу захисту в середовищі ігрового рушія «Unity».

Для впровадження такого методу захисту в будь-якому проекті не тільки в середі розробки ігрового рушія «Unity», а і в інших, треба мати дві складові. Сервер на якому розташовані файлі і можливість їх отримання за допомогою будь-якої системи або веб-платформи яка була використана при створені демонстраційного варіанту.

Друга частина повинна розташовуватися на пристрої гравця при встановлені проекту. При створенні демонстраційного проекту було надано схему як саме працює такий захист, та було показано їх варіації. Створений проект має модулі які можна використовувати в будь-якому проекті який створюється за допомогою ігрового рушія «Unity», але вибір які саме файли потрібно зберігати на сервері вже робить розробник цього проекту.

Рекомендується зберігати на сервері конфігураційні файли, які легко і швидко можна завантажити з сервера без додаткового навантаження на систему. Такі файли в собі містять основні параметри ігрових об'єктів, які можна змінювати в процесі гри. Для прикладу, в конфігураційні файли можна прописувати колір об'єктів, розмір у вигляді «Vector3», чи у вигляді змінних типу «float», «int» і при завантаженні змінювати їх.

Дуже часто розробники при створенні ігор використовують спеціальні ігрові об'єкти «Prefab». Система «Prefab» від «Unity» дозволяє створювати, налаштовувати та зберігати ігрові об'єкти разом із усіма його компонентами, значеннями властивостей і дочірніми ігровими об'єктами як багаторазовий ресурс.

«Prefab Asset» діє як шаблон, за допомогою якого можливо створювати нові екземпляри «Prefab» у сцені.

Тому такі ігрові об’єкти також рекомендується зберігати на сервері, так як вони є ключовою складовою проекту. Такі об’єкти використовуються більше одного разу, тому для них буде створюватися окремий файл і робитися запит кожен раз, коли цей об’єкт буде потрібний в процесі гри. При останньому використанні за визначений проміжок часу такий файл повинен бути видалений. Також рекомендується очищати такі файли автоматично при завершені процесу гри користувачем.

4.2 Тестування швидкості завантаження різних тестових варіантів

При такому тестуванні до основної частини код, було додано модуль, який зберігає час початку виконання і виводить це на екран. Також цей модуль зберігає кінець і виводить час за який було завантажено і завершено виконання функції. В кінці завершення функції на екран виводиться 3 повідомлення. Перше повідомлення показує час який пройшов після запуску проекту при початку функції, друге повідомлення час який пройшов після запуску проекту після завершення виконання функції, третє показує сумарний час за який було виконано функцію.

На рисунку 4.1-4.2 надано два приклади завантаження звичайного тексту з сервера. При двох різних завантаженнях одного й того же тексту можна побачити різний час. Треба зауважити, що час який показаний в третьому повідомленні, включає в себе не тільки час завантаження, а і час виконання додаткових функцій. В цьому випадку додатковою функцією була зміна тексту на завантажений.

При такому рахуванні часу, функції які не витрачають приблизно один й той самий час на виконання рахуються константами і не беруться до уваги при рахуванні зміни часу. Тому функція по зміні тексту на завантажений не впливає на різницю в часі між першою і другою спробами. Але навіть така мала різниця в часі не дуже чітко показує на скільки це впливає на ігровий процес.

Для розуміння наскільки великим є вплив часу завантаження на весь ігровий процес, до першої функції по завантаженню тексту було додано ще один модуль, який рахує кількість кадрів за які було завантажено і змінено текст. Частота кадрів (виражена в кадрах за секунду) – це зазвичай частота (частота), з якою послідовні зображення (кадри) знімаються або відображаються.

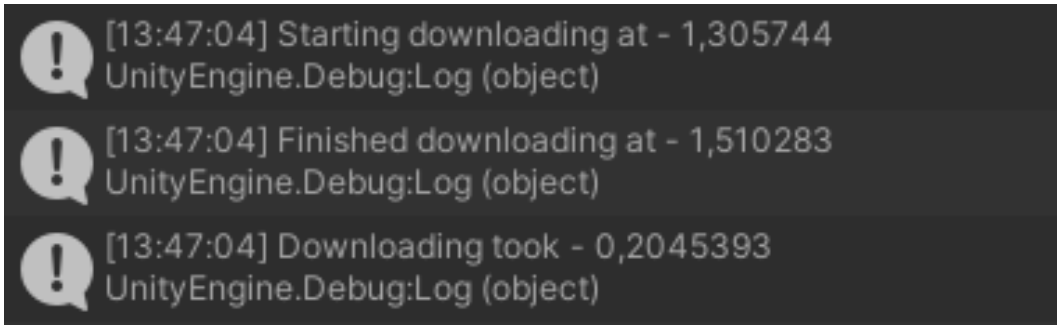

Рисунок 4.1 – Звичайний текст перша спроба

За кожний такий кадр в ігровому рушії «Unity» викликається функція «Update» на кожному об’єкті на сцені який реалізував її в середині коду. На рисунку 4.3 показано результат функції яка демонструє на екрані кількість кадрів які пройшли після початку функції завантаження тексту.

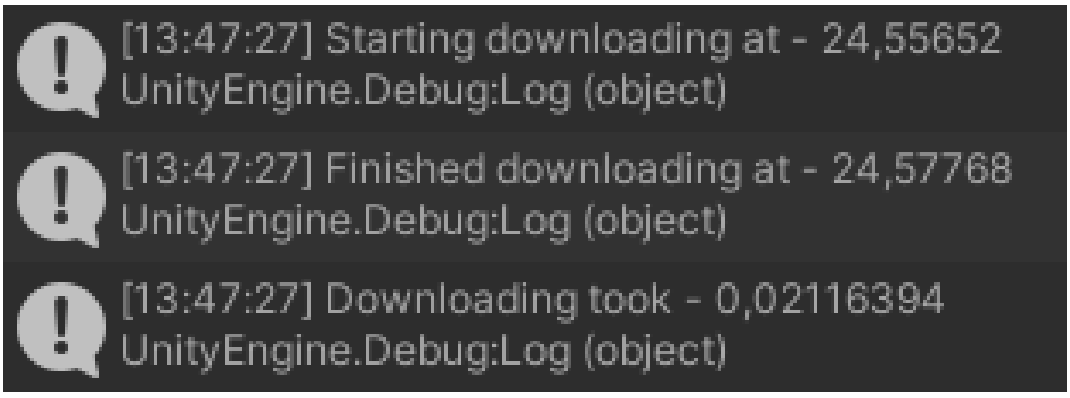

Рисунок 4.2 – Звичайний текст друга спроба

Для того щоб завантажити текст і замінити на нього зміст в створеному об’єкті знадобилося 0.058 секунді і тринадцять кадрів. За цей час функцію «Update» на всіх об’єктах на сцені було викликано 13 разів.

Тому при використанні такого підходу, треба розуміти, що об’єкти які будуть використовуватися в наступному кадрі завантажувати потрібно заздалегідь.

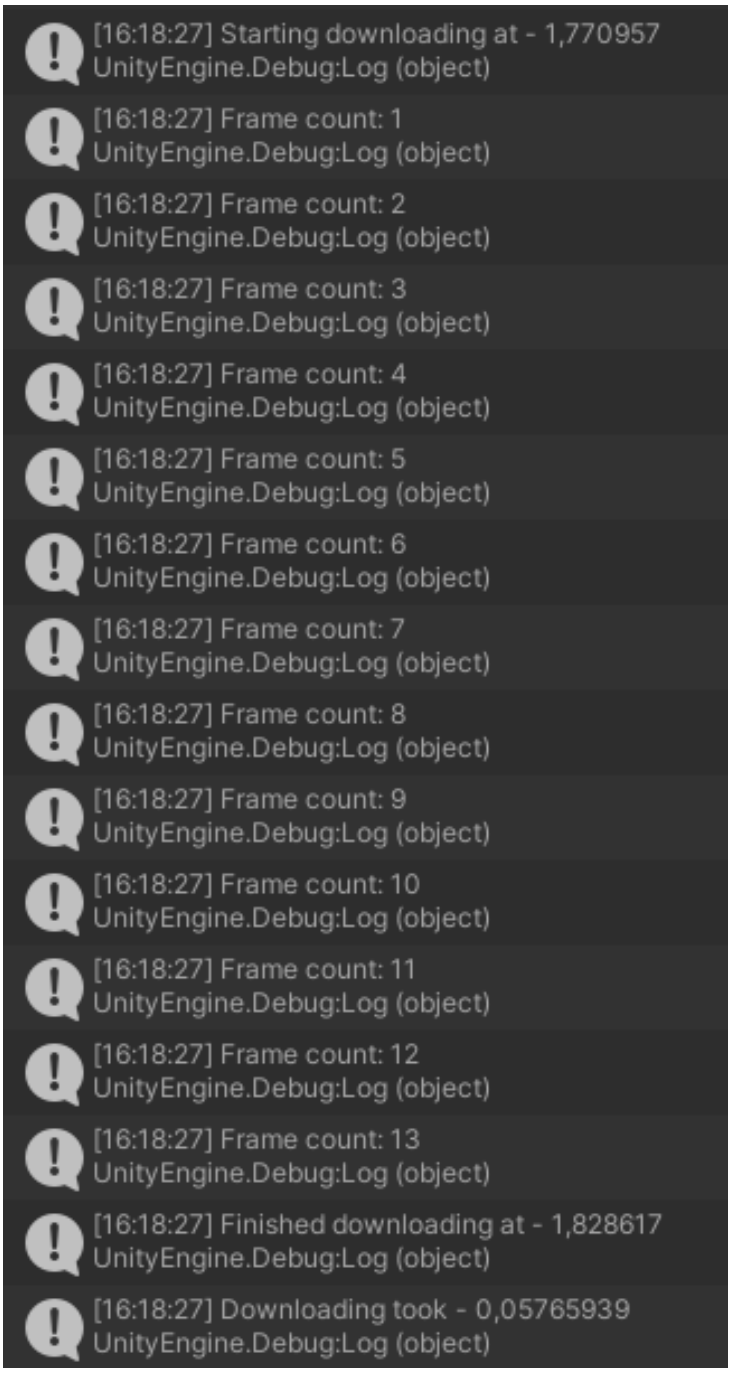

Рисунок 4.3 – Кількість кадрів яка пройшла за час завантаження звичайного тексту

Тому рекомендується файли, які будуть використовуватися більш ніж один раз і мають досить великий об’єм завантажувати при ввімкненні гри чи при переході з одної ігрової сцени на іншу.

Великі файли, як файли форматів «.wav» та схожого типу використовуються досить часто і різними ігровими об’єктами. Такі об’єкти не рекомендується завантажувати на сервер а краще зберігати на стороні клієнта для зменшення навантаження на пристрій. Зберігати на сервері рекомендується тільки якщо такі файли використовуються тільки в окремих випадках, чи невелику кількість разів в різних ділянках проекту.

Для проведення тестування довжини завантаження всіх типів які представленні в проекті, до кожної функції було додано модуль який рахує кількість часу яка потребується на завантаження. На рисунках 4.3-4.7 можна побачити час який знадобився для завантаження усіх типів файлу.

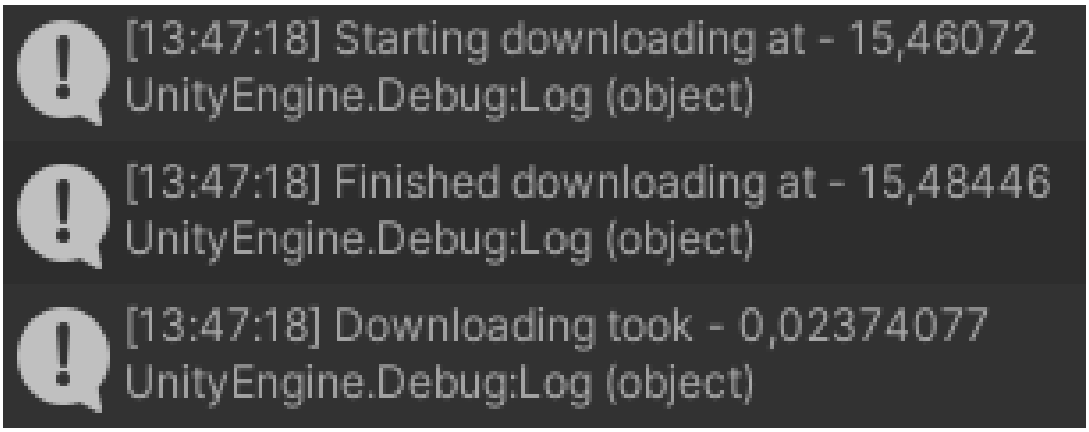

Рисунок 4.4 – Зашифрований текст

Окрема можна виділити довжину першого завантаження зашифрованого файлу зображення, яка становить цілих 15 секунд.

Так трапляється, коли антивірус чи брандмауер намагаються перевірити трафік який надходить на пристрій користувача.

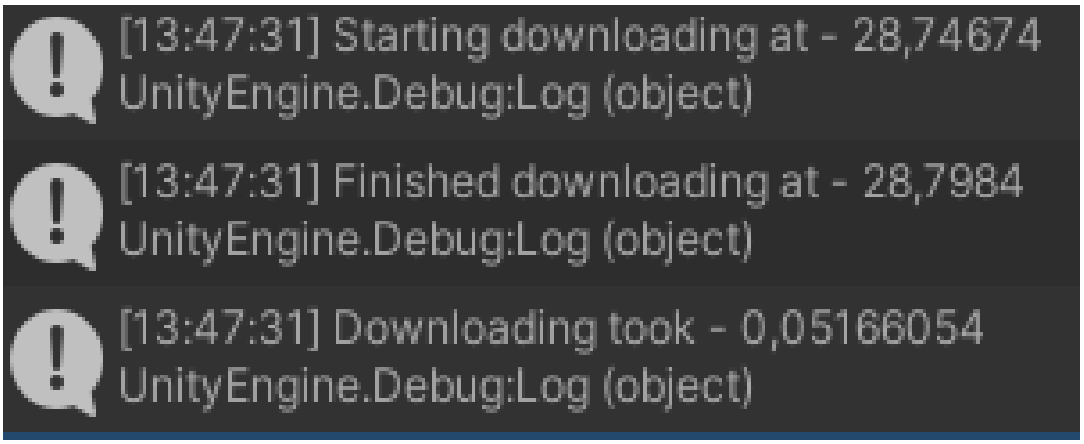

Рисунок 4.5 – Звичайне зображення

Після першої перевірки можна побачити, що швидкість повернулась до звичайних для такого об’єму файлу в 0.033 секунди.

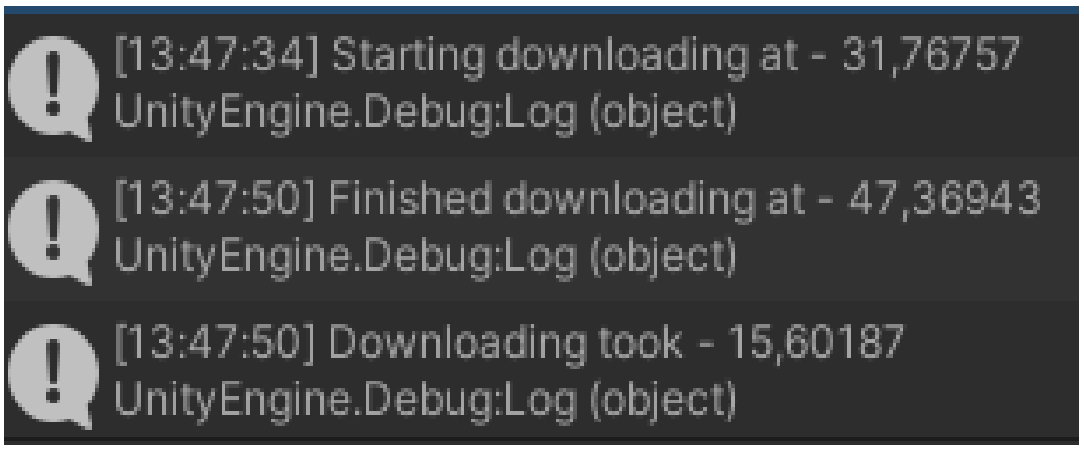

Рисунок 4.6 – Зашифроване зображення перша спроба

Це час за який було завантажено файл, записано на фізичний носій, розшифровано окремим модулем, який створив новий файл, а зашифрований видали і після цього це зображення було використано в користувацькому інтерфейсі а файл видалений.

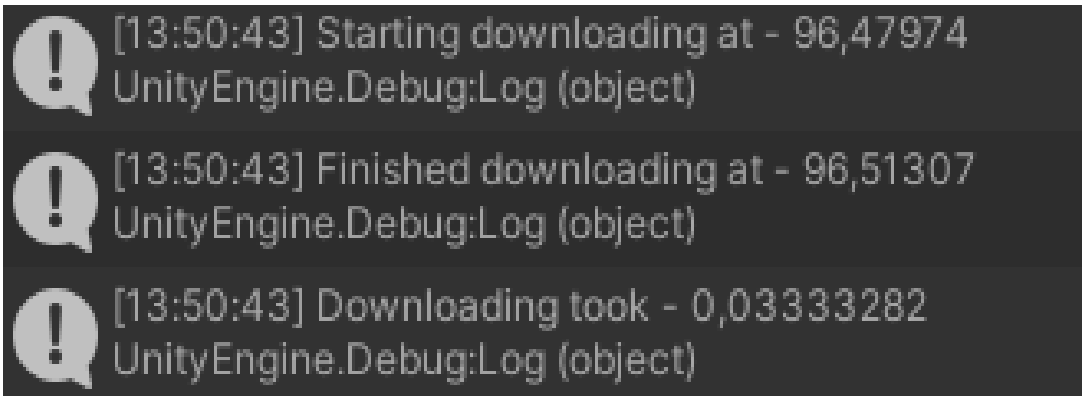

Рисунок 4.7 – Зашифроване зображення друга спроба

Як можна побачити всі ці дії з файлом не мають значного впливу на загальний час виконання функції як час завантаження. Але при порівнянні всіх запитів з різними типами та різним об’ємом файлів можна зробити висновок, що час завантаження файлу без додаткових функцій залежить від великої кількості чинників.

4.3 Рекомендації після тестування мережевої частини захисту

Через те, що цей захист потребує постійного обміну даними сервера з клієнтом є можливість, що зловмисники зможуть отримати всі файли і зберегти їх, якщо будуть сканувати мережу будь-яким аналізатором пакетів. Для цього було проведено низку тестів, щоб показати яку інформацію можна отримати таким шляхом. Для цього була встановлена додаткова утиліта «Wireshark».

«Wireshark» - це безкоштовний аналізатор пакетів із відкритим кодом. Він використовується для усунення несправностей мережі, аналізу, розробки програмного забезпечення та протоколу зв’язку, а також навчання.

За його допомогою було проаналізовано всі чотири запити на сервер, які продемонстровані в проекті, а саме: завантаження звичайного тексті, завантаження зашифрованого тексту, завантаження звичайного зображення і завантаження зашифрованого зображення.

На рисунку 4.8 можна побачити відфільтровані запити клієнта до сервера та його відповіді.

Time	Source	Destination	Protocol	Length	Info
18.386655	192.168.0.105	192.168.0.101	HTTP	262	GET /download1/ HTTP/1.1
18.399361	192.168.0.101	192.168.0.105	HTTP	344	HTTP/1.1 200 OK (text/plain)
19.006824	192.168.0.105	192.168.0.101	HTTP	262	GET /download2/ HTTP/1.1
19.019025	192.168.0.101	192.168.0.105	HTTP	357	HTTP/1.1 200 OK (text/plain)
19.610507	192.168.0.105	192.168.0.101	HTTP	262	GET /download3/ HTTP/1.1
19.626807	192.168.0.101	192.168.0.105	HTTP	886	HTTP/1.1 200 OK (PNG)
20.164842	192.168.0.105	192.168.0.101	HTTP	262	GET /download4/ HTTP/1.1
20.184137	192.168.0.101	192.168.0.105	HTTP	904	HTTP/1.1 200 OK (image/png)

Рисунок 4.8 – Всі запити

В даному випадку IP адреса клієнта – «192.168.0.105», а IP адреса сервера – «192.168.0.101». На рисунку 4.9 зображено результати запитів на самому сервері.

```
[12/May/2023 16:31:16] "GET /download2/ HTTP/1.1" 200 32
[12/May/2023 16:31:26] "GET /download1/ HTTP/1.1" 200 25
[12/May/2023 16:31:30] "GET /download3/ HTTP/1.1" 200 23924
[12/May/2023 16:31:48] "GET /download4/ HTTP/1.1" 200 23936
[12/May/2023 16:34:41] "GET /download4/ HTTP/1.1" 200 23936
[12/May/2023 16:47:16] "GET /download1/ HTTP/1.1" 200 25
[12/May/2023 16:47:16] "GET /download2/ HTTP/1.1" 200 32
[12/May/2023 16:47:17] "GET /download3/ HTTP/1.1" 200 23924
[12/May/2023 16:47:17] "GET /download4/ HTTP/1.1" 200 23936
[13/May/2023 11:14:21] "GET /download1/ HTTP/1.1" 200 25
[13/May/2023 11:14:22] "GET /download2/ HTTP/1.1" 200 32
[13/May/2023 13:18:27] "GET /download1/ HTTP/1.1" 200 25
```

Рисунок 4.9 – Всі запити

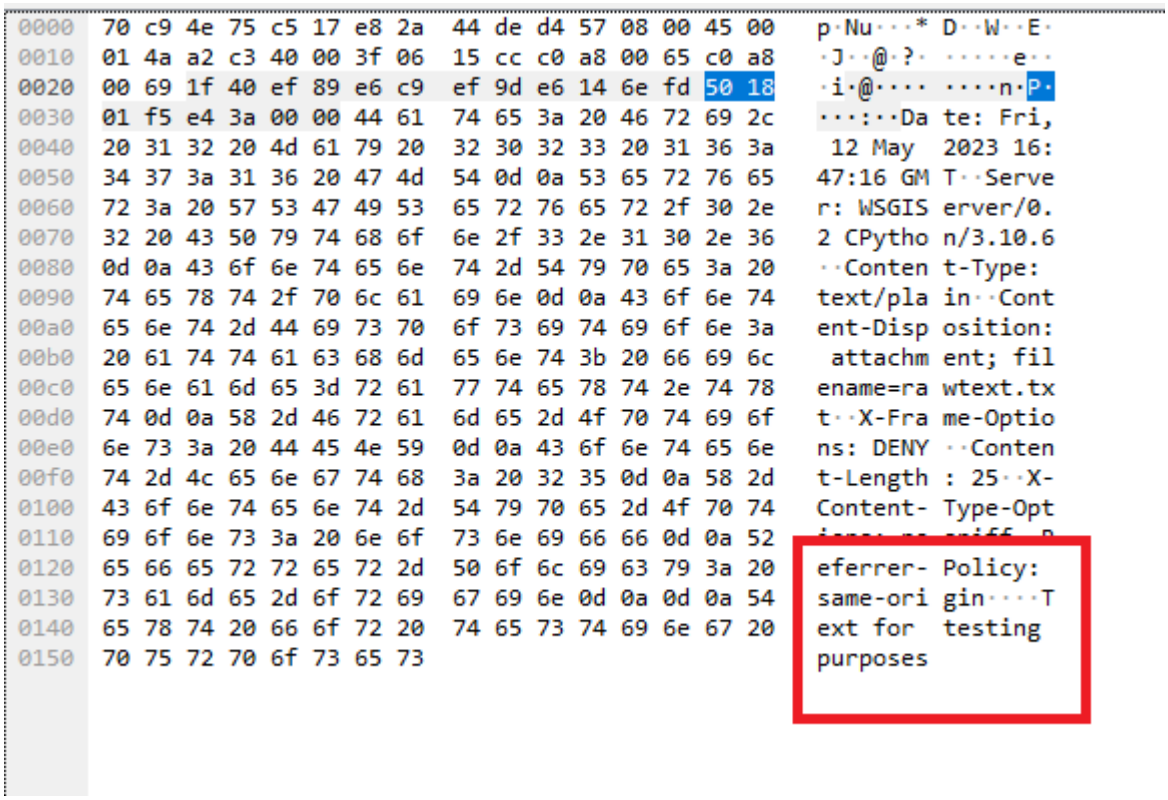

Рисунок 4.10 – Вигляд змісту відповіді на запит

Такі результати запитів складаються з:

1) «API» адреса запиту»
2) тип запиту;
3) статус відповіді (HTTP Code Response);
4) об’єм файлу який було надіслано на запит.

Зі списку з запитами можна побачити, що на кожен «API» було прив’язано окремий файл. В списку запитів «Wireshark» наведено, який тип файлу було отримано при запиті. Наприклад при запиті з адресою «/download1/» можна побачити, що у відповідь отримується файл типу «text/plain» і на рис. 4.10-4.11 можна побачити зміст такого запиту.

Так як текст не зашифрований, то можна побачити, що саме передається по цьому запиту. При шифруванні, як показано на рис. 4.11, такий текст неможливо прочитати, але клієнтська частина захисту все ще може з ним працювати. Щоб продемонструвати, що саме міститься в пакетах якими обмінюються клієнт з сервером були використані саме «HTTP» запити, а не «HTTPS».

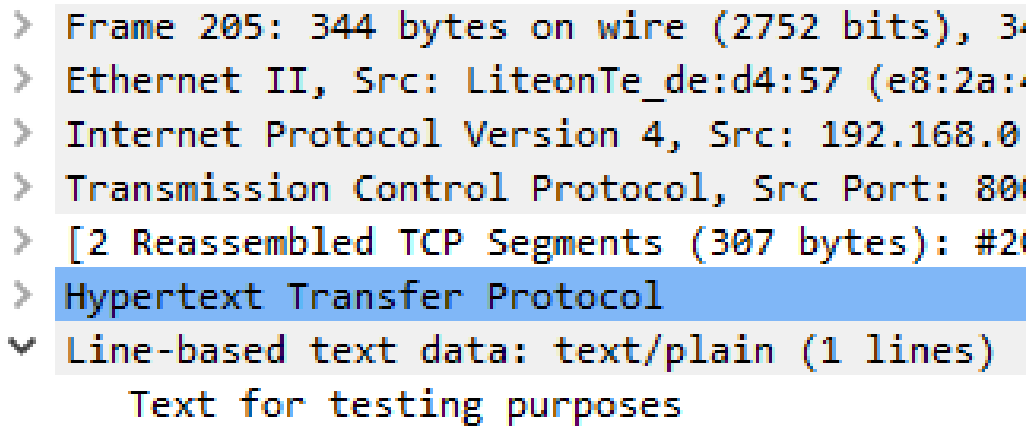

Рисунок 4.11 – Вигляд змісту в зручній формі

Так як при запиті «HTTPS» використовується шифрування для безпечного зв'язку в комп'ютерній мережі. При запитах «HTTPS» є автентифікація веб-сайту або сервера, до якого здійснюється доступ, а також захист конфіденційності та цілісності обмінюваних даних під час їх передачі.

Такий запит захищає від атак типу «людина посередині», а двонаправлене шифрування блочним шифром зв'язку між клієнтом і сервером захищає зв'язок від підслуховування та втручання.

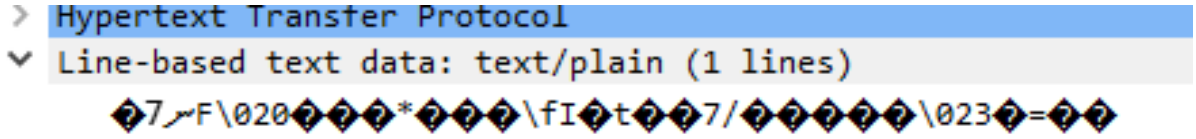

Рисунок 4.12 – Вигляд змісту відповіді на запит зашифрованого тексту

Тому звичайні аналізатори пакетів не зможуть показати зміст пакету. Якщо проаналізувати всі тестові варіанти (рис. 4.9-4.14), то при використанні http запитів, які не потребують додаткової аутентифікації, файли рекомендується зашифровувати.

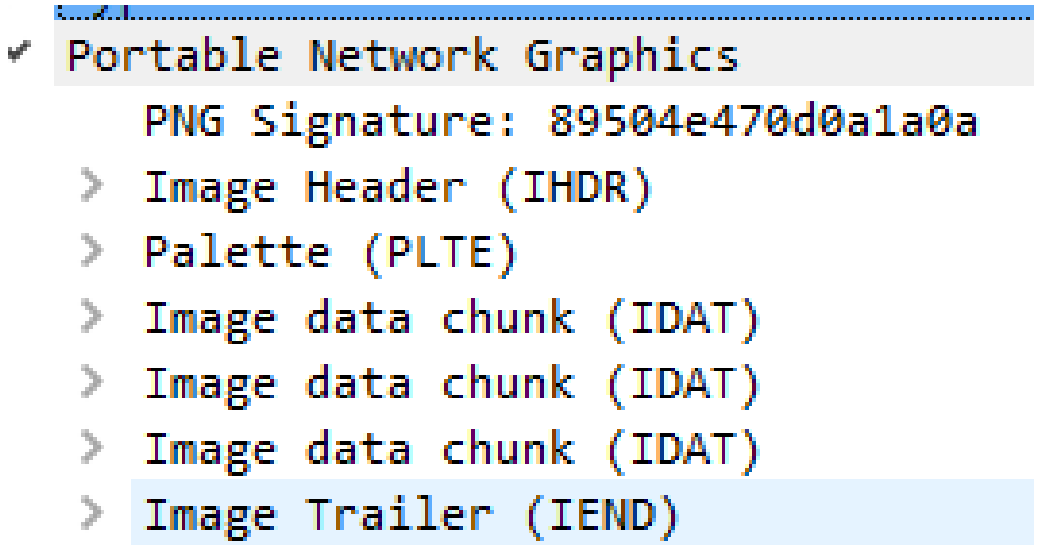

Рисунок 4.13 – Вигляд змісту відповіді на запит звичайного зображення

Але треба мати на увазі, що зашифрований файл, завжди трохи більший ніж розшифрований, що можна помітити на результатах які показує сервер.

«HTTP» запити використовуються тільки під час розробки проекту, для зручності, а при кінцевому результаті, такі ігри мають аутентифікацію з сервером, тому у фінальних версій проекту рекомендується змінити всі запити на формат «HTTPS».

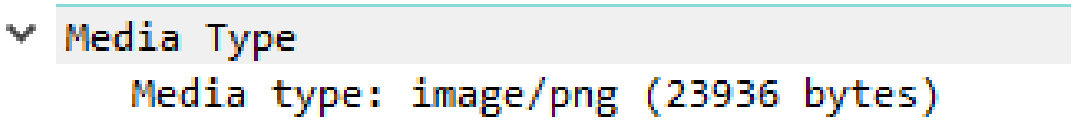

Рисунок 4.14 – Вигляд змісту відповіді на запит зашифрованого зображення

4.4 Рекомендації по створенню ключа для шифрування

При використанні модуля шифрування наведеного методу захисту треба створити особистий ключ для кожного користувача. Так як для «HTTPS» запитів потрібна окрема аутентифікація, для захисту від нелегального копіювання рекомендується використовувати унікальні ідентифікатори пристрою на якому була

проведена перша авторизація. Для створення такого ідентифікатору можна використати унікальний ідентифікатор пристрою.

Унікальний ідентифікатор пристрою (УІП) – це унікальний числовий або буквено-цифровий код, який зазвичай складається з наступного: ідентифікатор пристрою, обов'язкова фіксована частина УІП, яка ідентифікує видавця та конкретну версію або модель пристрою.

Такий ідентифікатор можна отримати за допомогою спеціального класу «SystemInfo» в ігровому рушії «Unity». Цей клас допомагає отримати доступ до інформації системи і інформації про обладнання пристрою. Цей клас використовується зазвичай для того, щоб перевірити який саме формат файлу підтримує процесор чи відеокарта. Також цей клас може надати досить великий обсяг інформації про систему, яку можна використовувати для створення унікального ідентифікатора користувача, а саме [12]:

1) модель пристрою;
2) ім'я пристрою;
3) тип пристрою;
4) унікальний ідентифікатор пристрою;
5) ідентифікатор графічної карти;
6) ім'я графічної карти;
7) виробник графічної карти;
8) операційну систему яка встановлена на пристрої;
9) кількість пам'яті системи.

Взагалі цей клас може надати більше інформації, але всю цю інформацію ніяк не можна використати при створенні унікального ідентифікатора користувача. Рекомендується використовувати більше ніж один варіант зі списку, так як існує можливість програмно змінити деякі дані в системі, щоб видати свій пристрій за пристрій іншого користувача.

Висновки за розділом 4

В даному розділі були протестовані частини захисту і за допомогою результатів тестування створено рекомендації по використанню такої системи захисту авторського права. Більшість прикладів були наведені для ігрового рушія «Unity», але цей метод захисту можна використовувати і на інших ігрових рушіях. Але треба пам’ятати, що кожен ігровий рушій має свої особливості при завантажені файлів і додавання їх у гру під час виконання.

Особливу увагу в цьому розділі було присвячено саме шифруванню даних і створенню унікального ідентифікатора користувача відштовхуючись від системних параметрів пристрою. Так як є можливість отримати файли перехопивши їх звичайним аналізатором пакетів, то такі дані потрібно шифрувати при використанні «http» і залишати без шифрування при використанні «https». Але навіть при використанні «https» рекомендується шифрувати файли, які будуть зберігатися досить довго на фізичному носія під час процесу гри, для підвищення захисту такого методу захисту.

ВИСНОВКИ

Захист авторського права в комп’ютерних іграх є ключовою складовою при розробці. Хоча ігрові рушії мають різну середу розробки, метод захисту розроблений в даній роботі не має чіткої прив’язки до якогось одного і його можна відтворювати і використовувати в різних середовищах як ігрових рушіїв так і серверних веб-платформ.

В першому розділі було розглянуто як працюють ігрові рушії з середини і проаналізовано складові життєвого циклу створення комп’ютерної гри як проекту. Особлива увага було приділена ігровому рушію «Unity» так як він був взятий за основу реалізації захисту розробленого в цій роботі.

В другому розділі були розглянуті проблеми з захистом авторського права. Було проаналізовані існуючи методи захисту, такі як DRM, різні його типи та варіанти роботи. Також були розглянуті проблеми вже з існуючими підходами і взято їх до уваги при розробці нового методу.

В третьому розділі було побудовано схему захисту авторського права. Також в цьому розділі було показано створену комп’ютерну гру з тестовими варіантами використання створеного методу в різних його варіаціях.

Особливість такого захисту складається в тому, що вся гра ніколи не зберігається на стороні клієнта (гравця). Основна частина гри знаходиться на сервері. Під час процесу гри файли, які потрібні для проходження гри, надаються сервером тільки після запиту від клієнтської частини і не зберігаються після завантаження, а видаляються, що допомагає захистити проект від незаконного копіювання та модифікацій.

В четвертому розділі було протестовано вже створену комп’ютерну гру та розроблено рекомендації на рахунок впровадження і використовування при розробці гри. Особливу увагу було приділено типу запитів до серверу, шифрування

та створення унікального ідентифікатора користувача за допомогою внутрішніх класів «Unity».

Поставлені завдання були виконані у повному обсязі. Розроблення метод у захисту авторських прав успішно завершено і додано у комп’ютерну гру. Також кожен аспект такого захисту був протестований для кращого процесу додавання такого методу в інші проекти.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Major Publishers Report AAA Franchises Can Cost Over a Billion to Make [Електронний ресурс] – Режим доступу до ресурсу: https://sea.ign.com/call-of-duty-infinity-ward-project/198451/news/major-publishers-report-aaa-games-can-cost-over-a-billion-to-make

2. THE SEVEN STAGES OF GAME DEVELOPMENT, Ross Branble, 04 January 2023 [Електронний ресурс] - Режим доступу до ресурсу: https://gamemaker.io/en/blog/stages-of-game-development

3. The 7 Stages of Game Development, Devin Pickell, October 15, 2019 [Електронний ресурс] - Режим доступу до ресурсу: https://www.g2.com/articles/stages-of-game-development

4. How video games are made: the game development process, Nadia Stefyn, 05/09/2022 [Електронний ресурс] - Режим доступу до ресурсу: https://www.cgspectrum.com/blog/game-development-process#:~:text=The%203%20stages%20of%20game,production%2C%20and%20post%2Dproduction.

5. 4 Phases of the Project Management Lifecycle Explained, Written by Coursera, Aug 10, 2022 [Електронний ресурс] - Режим доступу до ресурсу: https://www.coursera.org/articles/project-management-lifecycle

6. How Do Game Engines Work?, Nov 02, 2016 [Електронний ресурс] - Режим доступу до ресурсу: https://interestingengineering.com/innovation/how-game-engines-work

7. How does a Game Engine work? An Overview JANUARY 9, 2016 HAROLD SERRANO [Електронний ресурс] - Режим доступу до ресурсу: https://www.haroldserrano.com/blog/how-do-i-build-a-game-engine

8. Game engines: a survey, Virtual Campus Lda. Av. Fernão de Magalhães, 716, 1 PT 4350-151, Porto, Portugal [Електронний ресурс] - Режим доступу до ресурсу: https://www.researchgate.net/publication/283657797_Game_engines_a_survey

9. Game engines in scientific research, Jeffrey Jacobson, January 2002 [Електронний ресурс] - Режим доступу до ресурсу: https://www.researchgate.net/publication/278232144_Game_engines_in_scientific_research

10. Jin Zhang, Wei Lou “The digital rights management game in peer-to-peer streaming systems” May 2011 Conference: INFOCOM, DOI: https://www.researchgate.net/publication/224244000_The_digital_rights_management_game_in_peer-to-peer_streaming_systems

11. C# .Net documentation, System.Security.Cryptography Namespace [Електронний ресурс] - Режим доступу до ресурсу: https://learn.microsoft.com/en-us/dotnet/api/system.security.cryptography?view=net-7.0

12. Unity documentation SystemInfo [Електронний ресурс] - Режим доступу до ресурсу: https://docs.unity3d.com/ScriptReference/SystemInfo.html

ДОДАТОК А

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1. Бучик С.С. Захист авторського права та інтелектуальної власності в комп’ютерних іграх за допомогою технічних заходів / Бучик С.С., Стецюк Є.О. // Проблеми кібербезпеки інформаційно-телекомунікаційних систем : V Міжнародна науково-практична конференція, 27-28 жовтня 2022 року : тези доп. – К.: ВПЦ "Київський університет", 2022. – С. 46-47.
2. Бучик С.С. Керування цифровими правами в комп’ютерних іграх та вплив на систему / Бучик С.С., Стецюк Є.О. // Проблеми кібербезпеки інформаційно-телекомунікаційних систем : VI Міжнародна науково-практична конференція, 27 квітня 2023 року : тези доп. – К.: ВПЦ "Київський університет", 2023. – С. 9-10.