Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
(повна назва)

Факультет Інфокомунікацій

Кафедра Інформаційно-мережної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Проектування та розробка освітньої платформи із використанням методів game-based learning
(тема)

Виконав: студент 2 курсу, групи ІМІм-21-2
Кравченко В.В
(прізвище, ініціали)

Спеціальність: 172 Телекомунікації та радіотехніка
(код і повна назва спеціальності)

Тип програми: освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма: Інформаційно-мережна інженерія
(повна назва освітньої програми)

Керівник: проф. Тихонов В.А.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Безрук В. М.
(підпис) (прізвище, ініціали)

Харків 2023

Не містить відомостей, заборонених
до відкритого публікування

Студент _____ _____ / В.В Кравченко /

Керівник__ _____/ В.А Тихонов /

Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій
(повна назва)
Кафедра Інформаційно-мережної інженерії
(повна назва)
Рівень вищої освіти другий (магістерський)

Спеціальність 172 Телекомунікації та радіотехніка
(код і повна назва)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Інформаційно-мерережна інженерія
(повна назва)

ЗАТВЕРДЖУЮ
Зав. кафедри «ІМІ»
_____проф. Безрук В. М.
(підпис)
«____»_____2023р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Кравченко Володимир Володимирович
(прізвище, ім'я, по батькові)

1. Тема роботи: Проектування та розробка освітньої платформи із використанням методів game-based learning
затверджена наказом по університету від «27» квітня 2023р. №398 Ст.
2. Термін подання студентом роботи до екзаменаційної комісії 18.05.2023р.
3. Вихідні дані до роботи: спроектувати та розробити освітньою платформу з використанням методів game-based learning. Описати подальші кроки розвитку.
4. Перелік питань, що потрібно опрацювати в роботі:
    4.1 Аналіз існуючих підходів, методів застосування game-based.
    4.2 Обґрунтування і вибір платформи, інструментів для розробки.

4.3 Розробка клієнтської та серверної частини платформи.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Ознайомлення і уточнення завдання	17.03-20.03.2023	Виконано
2	Підбір і вивчення літератури	17.03-27.03.2023	Виконано
3	Розробка розділу 1	27.03-16.04.2023	Виконано
4	Розробка розділу 2	27.03-16.04.2023	Виконано
5	Розробка розділу 3	10.03-20.04.2023	Виконано
6	Розробка розділу 4	10.03-20.04.2023	Виконано
7	Розробка розділу 5	10.03-20.04.2023	Виконано
8	Розробка розділу 6	10.04-30.04.2023	Виконано
10	Оформлення кваліфікаційної роботи	01.05-08.05.2023	Виконано

Дата видачі завдання 17.03.2023 року

Студент _____ Кравченко В.В.
(підпис) (прізвище, ініціали)

Керівник роботи _____ проф. Тихонов В.А
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи: 32 с., 19 рис., 8 джерел, 27 слайдів.

Ключові слова: ГЕЙМІФІКАЦІЯ, ВЕБ БРАУЗЕР, ОСВІТНЯ ПЛАТФОРМА, MINECRAFT.

Мета дослідження - дослідити можливості застосування гейміфікації в освітній сфері.

Метод дослідження. У роботі описані методи дослідження з використанням метрик - статистичних даних, зібраних від користувачів платформи.

Основним результатом роботи є готова освітня платформа з елементами гри, що використовує методологію game-based learning. Основною метою цієї платформи є підвищення ефективності навчання учнів.

Використання метрик дозволяє оцінити ефективність застосованого підходу до освіти та його окремих компонентів. Процес дослідження проводиться на окремих групах учнів.

ABSTRACT

Explanatory Note for the Diploma Thesis: 32 pages, 19 figures, 8 sources, 27 slides.

Keywords: Gamification, Web Browser, Educational Platform, Minecraft.

The aim of this study is to explore the possibilities of applying gamification in the educational field.

Research Methodology: The study describes the research methods used, which involve the use of metrics - statistical data collected from platform users.

The main outcome of this work is an educational platform with gamified elements, employing the methodology of game-based learning. Its primary objective is to enhance learning effectiveness among students.

The use of metrics allows for the evaluation of the effectiveness of the applied educational approach and its individual components. The research process is conducted with separate groups of students.

ЗМІСТ

ВСТУП

В сучасному світі розвиток інформаційних технологій відкриває безмежні можливості для покращення якості освіти та навчання. Одним із найбільш перспективних напрямків є застосування методів game-based learning, що дає можливість поєднувати процес навчання з елементами гри. В рамках даної дослідницької роботи розглядається проектування та розробка освітньої платформи з використанням методів game-based learning, що дозволить створити цікавий та ефективний інструмент для навчання.

Основними цілями кваліфікаційної роботи є дослідження потенціалу застосування методів game-based learning у навчальному процесі, створення концепції освітньої платформи та її розробка з використанням новітніх технологій.

Також буде спроектований опис подальшого розвитку платформи як бренду.

В результаті дослідження буде розроблений та створений готовий продукт для поліпшення навчального процесу серед учнів з використанням методів game-based learning.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Поняття гейміфікації

Гейміфікація - це процес застосування ігрових механік в незвичних ситуаціях для забезпечення більш ефективного використання, взаємодії та залучення до дії. В основі гейміфікації лежить складний інструментарій дизайну ігор, що дозволяє перетворити будь-яку діяльність на цікаву гру.

Мета гейміфікації полягає в тому, щоб за допомогою елементів гри стимулювати учасників до досягнення певних цілей або завдань. Вона дозволяє залучати користувачів, спонукаючи їх до переходу на новий рівень у розвитку, виконуючи певні завдання або отримуючи певні навички.

Кінцева мета гейміфікації - забезпечити активну мотивацію учасників, щоб розв’язати певні проблеми, виконати завдання або досягнути певного рівня знань і навичок. Гейміфікація може застосовуватися в різних сферах, включаючи освіту, бізнес, соціальні мережі та спорт.

1.2. Поняття game-based learning

Game-based learning (навчання з використанням ігор) - це конкретний тип гейміфікації, який використовує ігри для полегшення процесу навчання. Через гру, учні можуть набувати нові навички та розвивати критичне мислення в легкій та інтерактивній формі навчання. На відміну від традиційних методів викладання, навчання з використанням ігор приводить до залучення користувачів до оточення, подібного до гри, і тому більш ймовірно підвищує їхню мотивацію та стимулює активну участь у процесі навчання. Game-based learning стає все популярнішим останніми роками через свою ефективність у покращенні результативності навчання. Однак, дуже важливо також впевнитись, що використання ігор не стимулює негативну поведінку чи надмірну залежність. Шляхом використання ігрових

елементів для доповнення традиційних методів викладання, ми можемо створити високоефективний інструмент навчання та розвитку, придатний для учнів будь-якого віку.

1.3 Особливості методології game-based learning

Основні особливості методології:

1. Інтерактивність - гравець активно залучається до процесу навчання через взаємодію з ігровим середовищем та персонажами.
2. Мотивація - гра дозволяє створити мотивацію для навчання через захоплюючий сюжет та можливість отримати нагороди за досягнення цілей.
3. Контекст - ігри можуть бути створені для навчання різних тематик, що дозволяє створити контекст для засвоєння знань та навичок.
4. Тренування навичок - гра може допомогти у тренуванні певних навичок, наприклад, логічного мислення, спостережливості та координації рухів.
5. Співпраця та конкуренція - гра може сприяти розвитку співпраці та конкуренції між гравцями, що позитивно впливає на процес навчання.
6. Підсумкова оцінка - гра може надати можливість оцінити рівень засвоєння знань та навичок у вигляді підсумкової статистики.
7. Взаємодія з новими технологіями - використання ігор для навчання дозволяє взаємодіяти з новітніми технологіями та розвивати цифрову грамотність.
8. Автоматизація – використання автоматизації в навчальному процесі в грі дозволить замістити роль вчителя, контролювати і оцінювати успішність гравців, тим самим даючи змогу їм самостійно вчитися без необхідності постійного нагляду вчителя або з його мінімальною присутністю.

В цілому, game-based learning є ефективним методом навчання, який може забезпечити захоплюючий та ефективний процес засвоєння знань та навичок.

1.4 Приклади існуючих рішень

Game-based learning є популярним методом навчання, який активно використовується у різних галузях, від освіти до бізнесу. Нижче наведений огляд деяких існуючих рішень game-based learning:

1. Minecraft: Education Edition - освітня версія Minecraft, яка дозволяє використовувати ігрове середовище для навчання математики, науки, історії та інших предметів[1].

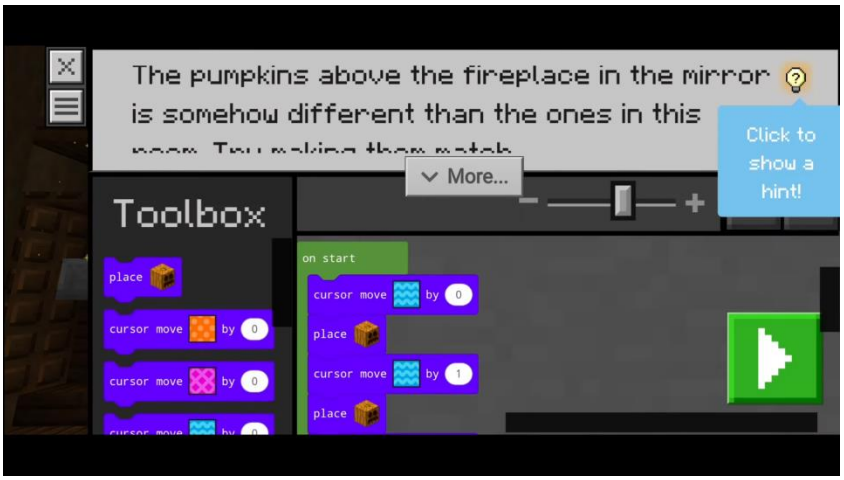


Рисунок 1.1 – Навчання програмуванню

2. Duolingo - це безкоштовний мобільний додаток для вивчення іноземних мов, який використовує ігрову механіку для навчання слів, граматики та інших аспектів мови[2].

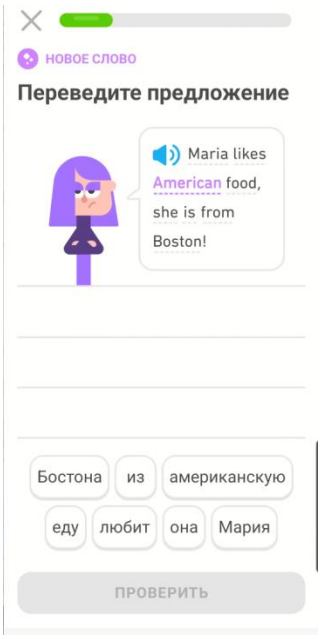


Рисунок 1.2 – Інтерфейс Duolingo з вивченням англійських слів

3. CodeCombat - це ігрове середовище для навчання програмуванню, де гравці виконують завдання та розв'язують головоломки, використовуючи код[3].

Рисунок 1.3 – Вибір мови програмування

2. ВИБІР ПЛАТФОРМИ

2.1 Аналіз потреб проекту

Моя мета була створити не гру для вивчення якогось конкретного предмета, а цілу платформу з інструментами, яка дозволить переносити будь-який матеріал, будь то математика, програмування або історія. Для досягнення цієї мети я вирішив інтегрувати практики в існуючу гру, замість того, щоб створювати її з нуля. Це особливо корисно з урахуванням обмежених ресурсів часу та фінансів, тому що створення гри з нуля вимагає набагато більше ресурсів.

2.2 Визначення вимог до платформи

З урахуванням написаного вище, були сформовані такі вимоги:

1. Платформа повинна бути гнучкою і надавати можливість створювати контент по будь якій дисципліни, а не прив'язуватися до якоїсь однієї як це роблять інші платформи.
2. Можливість надавати матеріал який може в себе включати не тільки текст, а й зображення, презентації та відеофайли.
3. Цікава ігрова складова, яка зможе утримувати увагу гравця та буде мотивувати грати та розвиватися далі.
4. Бути лояльною до модифікації клієнтської та серверної частини з урахуванням ліцензій

2.3 Обране рішення

Ігри для навчання можуть бути створені на різних платформах, таких як Minecraft, Roblox, Unity і багато інших. Провівши дослідження можливостей кожної з них, я зробив вибір на користь використання гри Minecraft як основи платформи з гейміфікацією.

Minecraft - це відеогра, в якій гравці можуть будувати світи, досліджувати, виживати та взаємодіяти з іншими гравцями[4]. Завдяки своїй відкритій структурі та безмежному творчому потенціалу, Minecraft є однією з найпопулярніших платформ для навчання.

Ось деякі переваги використання Minecraft для навчання:

5. Відкрите середовище: Minecraft має безмежний віртуальний світ, який дає можливість гравцям досліджувати та взаємодіяти з ним у будь-який спосіб. Це дозволяє використовувати Minecraft для навчання різних предметів, включаючи історію, точні науки та географію.
6. Підтримка від спільноти: Minecraft має велику та активну спільноту, яка створює різні матеріали та інструменти для навчання.
7. Впізнаваність серед аудиторії. Minecraft є однією з найпопулярніших відеоігор серед дітей та молоді. Тому використання цієї платформи для навчання може бути більш привабливим для учнів та студентів. Учні вже знайомі з Minecraft та які знають, як з ним працювати, можуть бути більш зацікавлені у навчальному процесі, якщо він базується на їхній улюбленій відеогрі. Крім того, використання популярних платформ, таких як Minecraft, може привернути більше уваги до навчання та підвищити мотивацію учнів для навчання.
8. Можливість модифікації гри. Minecraft має одну із самих розвинутих спільнот розробників, які створюють модифікації гри, вони дозволяють налаштувати різні аспекти гри, додавати нові функції та взаємодіяти з іншими програмами. Це дозволяє створювати унікальний та персоналізований навчальний досвід для учнів.

Вибір Minecraft як платформи для розробки ігор з використанням методів game-based learning є обґрунтованим через наявність великої кількості робочих прикладів на основі цієї платформи, зокрема, більшість з

них спрямовані на навчання програмуванню. Але ніщо не заважає використовувати Minecraft як інструмент для вивчення інших дисциплін.

3. ПРОЕКТУВАННЯ ПЛАТФОРМИ

Сама гра включає клієнтську та серверну частини. Клієнтська частина складається з двох частин: лаунчеру – програми, яку користувач використовує для входу в свій обліковий запис та завантаження файлів гри, та ігрового клієнта – самої гри, якою гравець підключається до ігрового сервера.

Ігровий сервер обробляє ігрову логіку та повертає інформацію про те, що відбувається у грі гравцеві.

Перш ніж розпочати розробку, треба було визначити, які саме компоненти ми будемо модифікувати. Гра дозволяє додавати власний контент, модифікуючи лише серверну частину, але це у свою чергу накладає обмеження. Сама гра не має вбудованого голосового та відеочату, а також можливості відтворення мультимедійного контенту, такого як відео, фотографії, презентації. Тому було прийнято рішення модифікувати не лише серверну, але й клієнтську частину.

3.1 Компоненти клієнтської частини

Клієнтська частина включає в себе ігровий лаунчер, оригінальні файли гри, спеціальні інструменти для модифікації цієї гри та веб-браузер, який ми вбудуємо в цю гру. Рішення про вбудування веб-браузера в уже існуючу гру було прийнято через його можливість реалізації наших вимог, таких як голосовий та відеочат і можливість відображення мультимедійного контенту. Крім того, це дає можливість використовувати його для створення складного та гнучкого інтерфейсу користувача, використовуючи вже існуючий великий набір інструментів для веб-розробки.

3.2 Компоненти серверної частини

Серверна частина гри зазвичай складається з одного серверного додатку, який обробляє підключення гравців. У нашому випадку, через необхідність створення власного контенту, було обрано кастомізований сервер, який розвивається спільнотою гри. Він дає можливість писати власні розширення до серверу, змінюючи оригінальну поведінку і додавати власні механіки до існуючої гри.
Зважаючи на наші задачі, серверна частина не обмежується лише цим. Крім цього, необхідно втілити сервер індивідуальних завдань, який буде підбирати індивідуальні завдання для кожного учня та створювати персоналізовану навчальну програму. Також потрібна серверна реалізація нашого голосового чату, для якого ми вбудували браузер в клієнтську частину гри.

Необхідно також враховувати, що в процесі реалізації поставлених задач може з'явитися необхідність у базі даних для зберігання прогресу учня та додаткових компонентів.

4. РОЗРОБКА КЛІЄНТСЬКОЇ ЧАСТИНИ

4.1 Вибір інструментів для модифікації гри

Модифікація гри здійснюється шляхом розробки модів. Вони дозволяють змінювати різні аспекти гри, починаючи від зовнішнього вигляду персонажів до створення нових світів та локацій. Для того, щоб розробляти моди, потрібні спеціальні інструменти. Завдяки великій спільноті гравців гри, їх не потрібно створювати самостійно, а можна створити на основі інших.

На сьогоднішній день існує два основних інструментів для створення власних модів під Minecraft: Forge та Fabric[5].

Основні відмінності між ними:

1. Вихідний код: Fabric має відкритий вихідний код, Forge - закритий.
2. Підтримка версій: Fabric оновлюється швидше, що робить його зручнішим для тих, хто хоче грати на останній версії Minecraft. Forge оновлюється повільніше і зазвичай затримується на попередніх версіях.
3. Розмір гри: Fabric створює менший розмір модифікованого клієнта, ніж Forge.
4. Сумісність: деякі моди можуть бути сумісні тільки з Forge, а деякі тільки з Fabric.
5. Офіційна підтримка: Forge має офіційну підтримку Mojang, тоді як Fabric підтримує CurseForge.
6. Наявність готових рішень: Forge з'явився набагато раніше, ніж Fabric, завдяки чому має набагато більше готових додатків, але Fabric розроблений якісніше за рахунок того, що враховує недоліки Forge.

Я в свою чергу обрав Fabric, оскільки цей інструмент набагато зручніше для розробки з використанням новітніх технологій. Хоч і існує недолік відсутності великої кількості готових рішень, це компенсується тим, що і під Forge не знайшлося готової реалізації нашої задачі.

4.2 Інтеграція веб браузера у гру

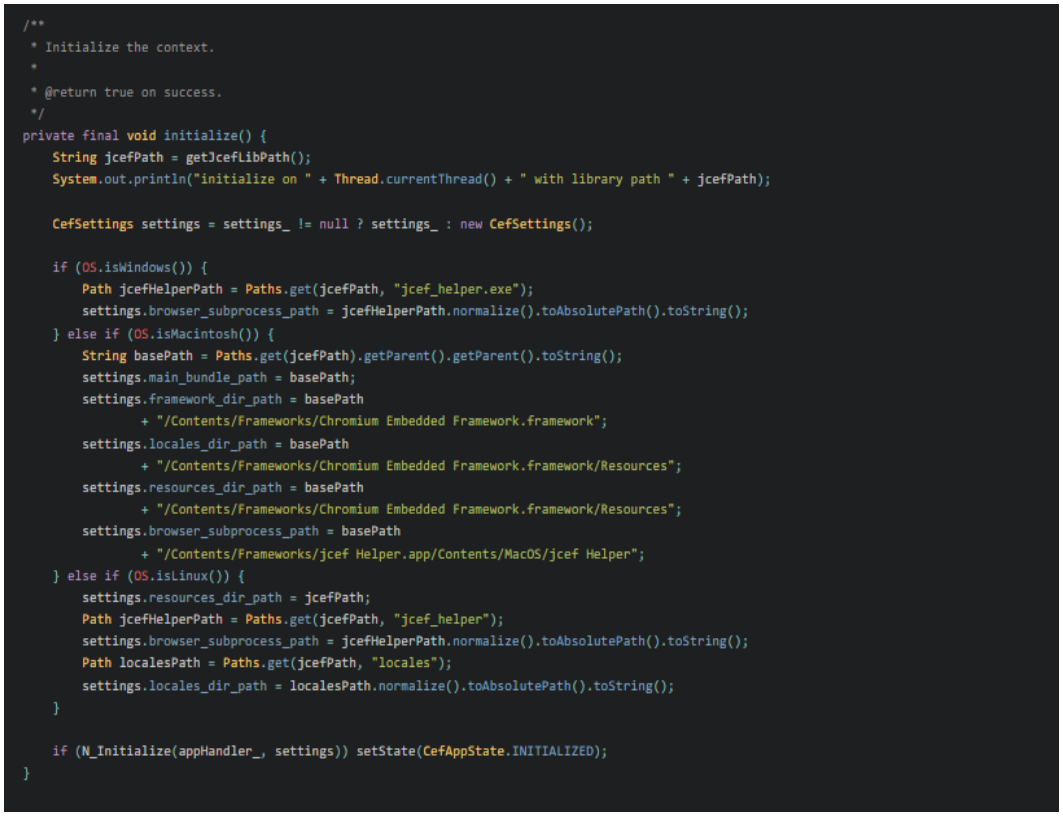
Для реалізації всіх необхідних механік було прийнято рішення спробувати інтегрувати веб-браузер безпосередньо в гру. Це дозволить користувачам зручно отримувати освітній матеріал, проводити відповідні завдання та перевіряти свої знання прямо зі свого ігрового середовища.

Для цієї задачі існує Chromium Embedded Framework (CEF) - фреймворк, який дозволяє вбудовувати браузерну технологію в додатки[6]. За допомогою нього, ми можемо безпосередньо інтегрувати веб-браузер у нашу гру, що дозволило нам забезпечити зручний доступ користувачів до навчального матеріалу прямо з гри.

Цей фреймворк дозволяє відображати вкладки браузера як елементи ігрового інтерфейсу, що надає нам більшу гнучкість та можливості для інтерактивного навчання в межах гри. Крім того, використання веб-браузера надає нам доступ до безлічі існуючих рішень та бібліотек, що використовуються в розробці веб-додатків. Це дозволяє нам ефективно використовувати ці інструменти для розширення можливостей нашого освітнього продукту та створення нових унікальних інтерактивних елементів, які допомагають користувачам засвоювати матеріал більш ефективно та захопливо.

Сама гра та серверна частина розроблені на мові програмування Java, у той час коли Chromium Embedded Framework використовує мову C++. Тому для можливості інтеграції в клієнтську частину гри нам потрібна мовна прослойка для Java. Однак, нам не довелося її розробляти, оскільки існує готове рішення від спільноти під назвою Java Chromium Embedded Framework.

Для створення нашого проекту ми використали систему автоматичної збірки Gradle для мови програмування Java. Підключено залежності, такі як JCEF (Java Chromium Embedded Framework) та Fabric. За допомогою бібліотеки JCEF ми зуміли запустити двигун браузера при запуску гри.



```
/**
 * Initialize the context.
 *
 * @return true on success.
 */
private final void initialize() {
    String jcefPath = getJcefLibPath();
    System.out.println("initialize on " + Thread.currentThread() + " with library path " + jcefPath);

    CefSettings settings = settings_ != null ? settings_ : new CefSettings();

    if (OS.isWindows()) {
        Path jcefHelperPath = Paths.get(jcefPath, "jcef_helper.exe");
        settings.browser_subprocess_path = jcefHelperPath.normalize().toAbsolutePath().toString();
    } else if (OS.isMacintosh()) {
        String basePath = Paths.get(jcefPath).getParent().getParent().toString();
        settings.main_bundle_path = basePath;
        settings.framework_dir_path = basePath
                + "/Contents/Frameworks/Chromium Embedded Framework.framework";
        settings.locales_dir_path = basePath
                + "/Contents/Frameworks/Chromium Embedded Framework.framework/Resources";
        settings.resources_dir_path = basePath
                + "/Contents/Frameworks/Chromium Embedded Framework.framework/Resources";
        settings.browser_subprocess_path = basePath
                + "/Contents/Frameworks/jcef Helper.app/Contents/MacOS/jcef Helper";
    } else if (OS.isLinux()) {
        settings.resources_dir_path = jcefPath;
        Path jcefHelperPath = Paths.get(jcefPath, "jcef_helper");
        settings.browser_subprocess_path = jcefHelperPath.normalize().toAbsolutePath().toString();
        Path localesPath = Paths.get(jcefPath, "locales");
        settings.locales_dir_path = localesPath.normalize().toAbsolutePath().toString();
    }

    if (N_Initialize(appHandler_, settings)) setState(CefAppState.INITIALIZED);
}
```

Рисунок 4.1 – Код ініціалізації веб браузера у нашому моді

Після запуску браузера, ми повинні відобразити його всередині елементів самої гри. Для цього було створено два варіанти відображення:

1. Оверлей - відображення над стандартним інтерфейсом з можливістю перехоплення управління. Використовується для відображення вікон відеозв'язку, діалогів та інших інтерактивних елементів.
2. Монітор - відображення всередині самої гри. Це може бути будь-який елемент. У нашому випадку, дошка або монітор.

Для відображення інтерфейсу ми використовуємо графічну бібліотеку для роботи з тривимірною графікою OpenGL. Для роботи з нею нам також необхідна мовна прослойка під назвою LWJGL.

```java
protected void onPaint(boolean popup, CefRectangle[] dirtyRects, ByteBuffer buffer, int width, int height, boolean completeReRender) {
    if (transparent_) // Enable alpha blending.
        GlStateManager.enableBlend();

    final int size = (width * height) << 2;
    if (size > buffer.limit()) {
        System.out.println("Bad data passed to CefRenderer.onPaint() triggered safe guards... (1)");
        return;
    }

    // Enable 2D textures.
    GlStateManager.enableTexture();
    GlStateManager.bindTexture(texture_id_[0]);

    int oldAlignement = glGetInteger(GL_UNPACK_ALIGNMENT);
    glPixelStorei(GL_UNPACK_ALIGNMENT, 1);

    if (completeReRender || width != view_width_ || height != view_height_) {
        // Update/resize the whole texture.
        view_width_ = width;
        view_height_ = height;
        glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, view_width_, view_height_, 0, EXTBGRA.GL_BGRA_EXT, GL_UNSIGNED_BYTE, buffer);
    } else {
        glPixelStorei(GL_UNPACK_ROW_LENGTH, view_width_);

        // Update just the dirty CefRectangles.
        for (CefRectangle rect : dirtyRects) {
            if (rect.x < 0 || rect.y < 0 || rect.x + rect.width > view_width_ || rect.y + rect.height > view_height_)
                System.out.println("Bad data passed to CefRenderer.onPaint() triggered safe guards... (2)");
            else {
                glPixelStorei(GL_UNPACK_SKIP_PIXELS, rect.x);
                glPixelStorei(GL_UNPACK_SKIP_ROWS, rect.y);
                glTexSubImage2D(GL_TEXTURE_2D, 0, rect.x, rect.y, rect.width, rect.height, EXTBGRA.GL_BGRA_EXT, GL_UNSIGNED_BYTE, buffer);
            }
        }

        glPixelStorei(GL_UNPACK_SKIP_PIXELS, 0);
        glPixelStorei(GL_UNPACK_SKIP_ROWS, 0);
        glPixelStorei(GL_UNPACK_ROW_LENGTH, 0);
    }

    glPixelStorei(GL_UNPACK_ALIGNMENT, oldAlignement);
    GlStateManager.bindTexture(0);
}
```

Рисунок 4.2 – Фунція рендеренгу зображення веб браузеру всередені гри

Ось приклад того, як це виглядає у грі з використанням оверлею (зверху справа) та відображення трансляції екрана робочого стола у середині ігрового світу

Рисунок 4.3 – Відображення вкладок браузера у самій грі

5. РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ

5.1 Вибір реалізації для ігрового сервера

За звичайних умов, розробники гри не надають механізмів для створення власних модифікацій. Але за допомогою копій оригінального сервера, розроблених великою спільнотою гри, було створено безліч різних версій ядра сервера з можливістю додавати сторонні модифікації та інструменти для їх створення.

Ось список деяких з них:

1. Bukkit - є одним з перших та дуже популярним ядром для Minecraft, який дозволяє додавати плагіни для сервера, проте воно є застарілим та його більше не оновлюють.
2. Sponge - це ядро з відкритим вихідним кодом, яке дозволяє створювати власні плагіни та модифікації для сервера Minecraft, створене для Forge.
3. Fabric - має аналогічну назву, як і інструменти для модифікації клієнтів. Воно також дозволяє модифікувати серверну частину гри.
4. Spigot - є найбільш популярним ядром для сервера Minecraft, яке дозволяє додавати плагіни та модифікації для покращення продуктивності та функціональності сервера. Це спадкоємець Bukkit.
5. Paper - є форком Spigot, який оптимізує продуктивність та покращує стабільність сервера Minecraft.

Мною був зроблений вибір у сторону Paper, тому що він поєднує в собі високу продуктивність та підтримку великої кількості вже існуючих плагінів. Це досягається завдяки тому, що Paper є нащадком Spigot, який, у свою чергу, є наслідником Bukkit.

Процес запуску сервера виглядає таким чином:

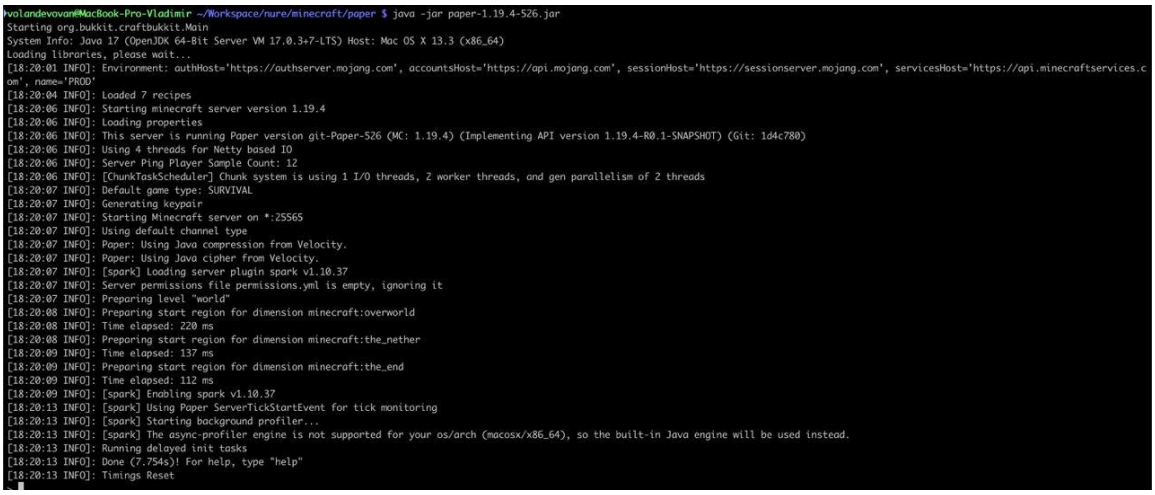


Рисунок 5.1 – Процес запуску сервера у терміналі

5.2 Розробка інструментів для створення контенту

Після запуску серверу була зроблена ставка на створення інструментів для подальшого прискорення процесу побудови контенту замість того, щоб створювати освітні світи самостійно. Це дозволить від самого початку максимально підготувати процес виробництва контенту та створить основу на якій все буде базуватись.

Для досягнення цієї мети було прийняте рішення додати підтримку JavaScript, щоб спростити взаємодію з API та прискорити процес подальшої розробки контенту.

Все це було зроблено за допомогою технології Polyglot в GraalVM, яка дозволяє запускати код на різних мовах програмування на одній віртуальній машині (JVM), що дозволяє отримати переваги від різних мов та бібліотек. У нашому проекті ми використовували цю технологію для написання JavaScript коду, який виконується на віртуальній машині Java (JVM).

Для спрощення роботи з Java API нашого продукту для розробників, які не мають досвіду роботи з Java, ми розробили спеціальні обгортки для API нашого продукту на Java. Ці обгортки генерують класи на льоту за допомогою бібліотек Bytebuddy та ASM, що дозволяє уникнути необхідності вручну писати код для роботи з API.

Інтеграція зовнішнього інтерфейсу для простих скриптів на JS прискорює розробку та робить її більш доступною для широкого кола розробників. Цей інтерфейс дозволяє розробникам писати прості скрипти на JavaScript для роботи з функціональністю нашої платформи, не затрагуючи складні аспекти роботи з Java API.

```
let fillWord = factory.fillWordsProcess(player.ref);
fillWord.newWord("[0]КН[0]", "1213,42,411", BlockFace.west, BlockFace.north);
fillWord.newWord("Ф[0]НАРЬ", "1208,42,411", BlockFace.west, BlockFace.north);
fillWord.newWord("[Я]БЛ[0]КО", "1211,41,411", BlockFace.west, BlockFace.north);
fillWord.start();
```

Рисунок 5.2 – Приклад скрипта механіки написаного на Javascript

5.3 Розробка сервісу підбору індивідуальних завдань

Сервіс підбору індивідуальних завдань включає програмне забезпечення, яке автоматично генерує та підбирає індивідуальні завдання для кожного користувача на основі їхніх попередніх досягнень та потреб. Ця система забезпечує максимальну ефективність навчання, оскільки кожен користувач отримує завдання, що відповідають його рівню знань та потребам. При цьому, система також забезпечує достатню складність завдань, щоб викликати зацікавленість та стимулювати до подальшого розвитку навичок.

Система підбору індивідуальних завдань може бути побудована на основі різноманітних алгоритмів та методів машинного навчання, що дозволяє забезпечувати максимальну точність та ефективність в підборі завдань для кожного користувача.

5.3.1 Вимоги до сервісу

Система підбору індивідуальних завдань повинна відповідати певним вимогам, щоб забезпечити максимальну ефективність та комфорт для користувачів. Основні вимоги до сервісу підбору індивідуальних завдань можуть включати такі:

1. Функціональність: Сервіс повинен бути здатен підбирати індивідуальні завдання для користувачів на основі їхніх попередніх знань та вмінь. Система повинна забезпечувати адаптивність завдань до особистих потреб користувача та рівня їхньої підготовки.
2. Масштабованість: Система повинна бути здатна масштабуватися під зростаючі потреби користувачів та обробляти великий обсяг даних.
3. Оцінювання користувача: Система повинна мати можливість оцінювати рівень знань та вмінь користувача та враховувати цю інформацію при підборі індивідуальних завдань. Таким чином, система може забезпечити оптимальний підбір завдань, що відповідають рівню знань та вмінь користувача та збільшити ефективність навчання.
4. Наявність статистики: Платформа повинна збирати та зберігати статистику про процес підбору завдань для кожного користувача. Це може допомогти користувачам відстежувати свій прогрес та оцінювати ефективність системи. Крім того, статистика може допомогти розробникам системи у вдосконаленні алгоритмів підбору завдань та оптимізації функціоналу системи.
5. Наявність рекомендацій: необхідно забезпечити можливість відображення користувачам рекомендацій щодо подальших завдань та занять на основі їхніх попередніх результатів. Це може допомогти користувачам збільшити ефективність навчання та досягнути кращих результатів.

Загалом, система підбору індивідуальних завдань повинна забезпечувати максимально ефективне та комфортне навчання для кожного користувача, забезпечуючи підбір індивідуальних завдань на основі рівня знань та вмінь користувача та класифікування завдань на різні категорії та

рівні складності. Для цього необхідно мати належну функціональність, ефективність, безпеку та можливість налаштування параметрів підбору індивідуальних завдань.

5.3.2 Алгоритм підбору завдань

Ігрок перебуває у віртуальному світі, створеному на базі платформи Minecraft. Це може бути будь-яка локація, побудована спеціально для вивчення якоїсь певної дисципліни. У цій локації існують різноманітні способи отримання завдання, наприклад, через взаємодію з ігровим персонажем (NPC). Під час взаємодії з NPC відправляється HTTP-запит до нашого сервісу підбору індивідуальних завдань, який включає унікальний ідентифікатор учня та іншу допоміжну інформацію.

Сервіс підбору індивідуальних завдань повинен обробити цей запит, враховуючи всі можливі параметри:

1. Унікальний ідентифікатор учня, який буде використано для отримання всієї інформації, що відноситься до учня, такої як вік, поточний рівень знань по кожній окремій дисципліні, матеріал, який вивчається, особисті побажання та інше.
2. Інформація про контекст, який може залежати від локації, де знаходиться гравець, або етапу виконання завдання, якщо це якась цепочка завдань.
3. Формат завдання. Сервіс повинен мати змогу обробляти не тільки завдання у вигляді тестів, а й інші формати завдань, наприклад, завдання, що потребують програмування або створення проектів.

Після отримання запиту сервіс повинен використовувати алгоритми машинного навчання та інші методи аналізу даних для відбору найбільш підходящих завдань для учня. Система може використовувати статистичні дані для порівняння успішності попередніх учнів, які виконували схожі

завдання, а також враховувати особисті побажання учня, щоб вибрати найбільш зацікавлюючі та корисні завдання для нього.

Після того, як сервіс повернув список завдань, учень може почати виконувати їх у грі. Під час виконання завдань, учень може давати відповіді на поставлені запитання або виконувати завдання на практиці. Сервіс повинен обробляти відповіді учня, щоб враховувати їх у подальшому підборі та оцінці рівня знань.

У залежності від формату завдання, сервіс може автоматично оцінювати відповіді учня або використовувати методи оцінки з використанням алгоритмів машинного навчання. Наприклад, у випадку завдань на програмування, сервіс може автоматично перевіряти код учня і надавати йому зворотний зв'язок про те, як можна покращити його роботу. У випадку завдань на розв'язання математичних задач, сервіс може автоматично перевіряти відповіді та оцінювати рівень знань учня з даної теми.

Отже, сервіс повинен не тільки підбирати найкращі завдання для учня, але й збирати інформацію про відповіді учня на ці завдання, щоб покращувати його подальший підбір та оцінку рівня знань. Це допоможе підвищити ефективність навчання та забезпечити найбільш оптимальний навчальний досвід для кожного учня.

5.3.3 Реалізація сервісу

Для вирішення поставленої задачі я обрав мову програмування Node.js. На початковому етапі розроблений мінімальний REST API сервіс, який обробляє запити на видачу індивідуальних завдань, які будуть зберігатися в базі даних MongoDB.

На початкових етапах ми не будемо використовувати механізми машинного навчання, але будемо повертати вже згенеровані завдання та обробляти відповіді на них. Це повинно робитися індивідуально для кожного

учня з використанням таких параметрів, як його рівень знань, дисципліна, яка вивчається, а також попередні результати та відповіді.

Для реалізації необхідного функціоналу, ми створюємо три моделі документів для зберігання даних в базі даних MongoDB: "Student", "Task" та "Result".

Модель "Student" містить поля для зберігання даних про учня, такі як його ім'я, вік, рівень знань та прогрес з вивчення кожної окремої дисципліни. Для збереження цих даних використовується тип Map, що дозволяє зберігати значення для кожного ключа у форматі ключ - значення.

Модель "Task" містить поля для зберігання даних про завдання, такі як заголовок, опис, дисципліна, формат, складність та саме завдання. Для кожного завдання зберігається інформація про його характеристики, які використовуються для подальшого підбору індивідуальних завдань для учнів.

Модель "Result" містить поля для зберігання даних про результати виконання завдань учнями. Для кожного результату зберігається посилання на моделі "Student" та "Task", дату виконання та отриманий бал. Ця модель дозволяє зберігати та аналізувати результати виконання завдань для кожного учня окремо, а також зв'язувати результати з відповідними учнями та завданнями.

За допомогою цих моделей ми можемо створювати документи для зберігання даних в базі даних MongoDB та здійснювати запити до бази даних для отримання даних про учнів, завдання та їх результати.

Використовуючи фреймворк Fastify у нашому сервісі підбору індивідуальних завдань, ми можемо обробляти запити на отримання завдань та на відповідь від учня за допомогою різних HTTP методів.

Ось приклад запиту на отримання завдання, який ми можемо обробити:

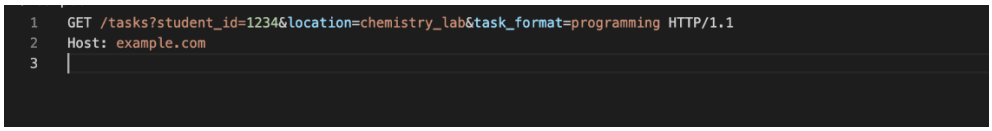


Рисунок 5.3 – Приклад запита на отримання завдання

У цьому запиті ми використовуємо метод GET та передаємо параметри запиту у URL-адресі. Ключовим параметром є student_id, який ідентифікує учня, для якого необхідно підібрати завдання. Також передаються параметри location та task_format, які дозволяють врахувати контекст вивчення та формат завдання.

При обробці запиту на отримання завдання, ми можемо виконати наступні кроки:

1. Перевірити параметри запиту та знайти відповідного учня за його ідентифікатором student_id.
2. Отримати контекст вивчення учня за допомогою параметра location.
3. Отримати формат завдання за допомогою параметра task_format.
4. Виконати запит до бази даних MongoDB та знайти всі завдання, що відповідають параметрам запиту.
5. Підібрати найкраще завдання за допомогою алгоритму підбору, який враховує рівень знань та прогрес вивчення дисципліни учнем, формат завдання та складність завдання.
6. Повернути результат у форматі JSON зі списком відповідних завдань.

Ось приклад запиту на запис відповіді від учня:

```
POST /results HTTP/1.1
Host: example.com
Content-Type: application/json

{
  "student_id": "1234",
  "task_id": "5678",
  "date_completed": "2022-04-01",
  "score": 80
}

```

Рисунок 5.4 – Вигляд запита на передачу результату завдання

У цьому запиті ми використовуємо метод POST та передаємо дані в форматі JSON у тілі запиту. Об'єкт JSON містить наступні поля:

1. student_id - ідентифікатор учня, для якого записується результат виконання завдання.
2. task_id - ідентифікатор завдання, для якого записується результат виконання.
3. date_completed - дата виконання завдання.
4. score - бал, отриманий учнем за виконання завдання.

При обробці запиту на запис відповіді в базу даних MongoDB, ми маємо виконати наступні кроки:

5. Отримати дані з тіла запиту та перевірити їх коректність.
6. Знайти відповідного учня та завдання за їх ідентифікаторами student_id та task_id.
7. Зберегти результат виконання завдання в базі даних MongoDB за допомогою моделі "Result".
8. Повернути відповідь у форматі JSON зі статусом успішності записування.

5.3.4 Інтерфейс користувача

У нашому випадку, клієнтський інтерфейс реалізований як ігровий інтерфейс. Він буде використовуватися для відображення завдань та інформації про них у вигляді ігрового елементу. Гравець зможе взаємодіяти з ним, виконувати завдання та навчатися новому матеріалу.

Після отримання завдань від сервісу підбору індивідуальних завдань, ми можемо відобразити їх у грі для учня. Це може бути зроблено за допомогою різноманітних методів, залежно від особливостей самої гри та її інтерфейсу.

Рисунок 5.5 – Приклад відображення питання гравцю

Таким чином ми підбираємо індивідуальні завдання для кожного учня з використанням наших алгоритмів та бази даних, щоб кожен гравець міг розвиватися відповідно до своїх потреб та здібностей. Ці завдання потім відображаються в грі за допомогою методів гейміфікації, що допомагають зробити навчальний процес більш цікавим та захоплюючим для учнів.

5.4 Розробка відеочату

У зв'язку з потребою у дистанційному навчанні, розробка відеочату для платформ стає все більш актуальною темою, оскільки вона дозволяє забезпечити максимальний рівень спілкування та інтерактивності між користувачами. Це може бути особливо корисно для групових проектів та інтерактивних завдань, де співпраця та комунікація є ключовими елементами успішного навчання.

У розробці системи аудіо та відеозв'язку для нашого освітнього продукту ми будемо використовували технологію WebRTC (Web Real-Time Communication), яка дозволяє проводити голосовий та відеозв'язок в режимі реального часу. Цей протокол вже вбудований в наш браузерний фреймворк CEF[7].

Для забезпечення зручного та ефективного процесу навчання ми використовували відкрите програмне забезпечення OpenVidu, яке забезпечує ефективну та стабільну роботу веб-конференцій з використанням технології WebRTC.

OpenVidu є готовим рішенням для відеоконференцій з використанням технології WebRTC, що дозволяє значно спростити розробку системи аудіо та відеозв'язку. Використання OpenVidu дозволяє забезпечити відеозв'язок у режимі реального часу між викладачем та студентами для ефективної реалізації онлайн-навчання.

Архітектура OpenVidu включає в себе готовий медіа сервер, серверну частину нашого додатку та клієнтську бібліотеку для взаємодії з медіа сервером. Завдяки використанню OpenVidu ми позбавляємося необхідності реалізувати власний медіа сервер, що відповідає за обробку аудіо та відео потоків. Відповідно, ми зосереджуємося на розробці веб-інтерфейсу та серверної частини додатку.

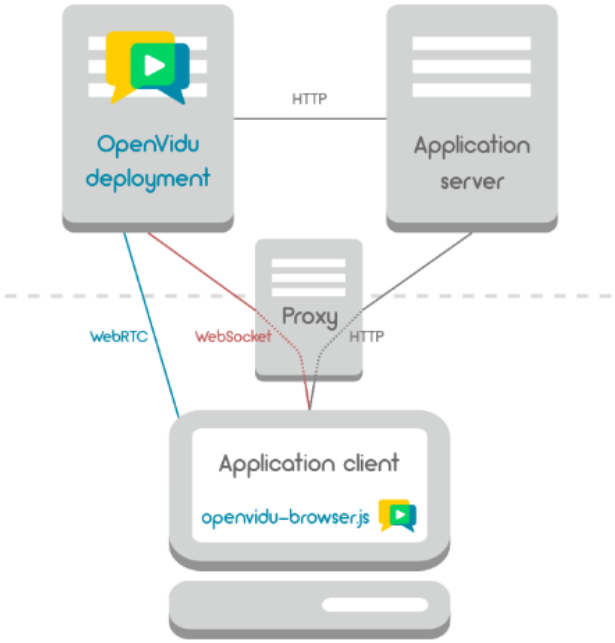


Рисунок 5.6 – Архитектура OpenVidu

Розробка системи складалася з декількох етапів:

1. Встановлення MediaServer Openvidu

Для MediaServer’а нам потрібен сервер с дистрибутивом linux і з підтримкою docker контейнеру.

Процес установки автоматизований і для нього існує готовий скрипт

```
root@c26c17d8220f:~# cd /opt/
root@c26c17d8220f:/opt# curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/install_openvidu_latest.sh |
 bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 15619  100 15619    0     0  65193      0 --:--:-- --:--:-- --:--:-- 65351
```

Рисунок 5.7 – Автоматична установка OpenVidu Media Server

Після цього ми налаштовуємо сервер в файлі .env та запускаємо його командою ./openvidu start

2. Розробка серверної частини нашого додатку

Серверна частина розроблена на Node.js з використанням бібліотеки для створення веб-додатків. Наш сервіс має обробляти HTTP REST API запит, створюючи токен та сесію у media server’і для користувачів.

```
app.post('/token', async (req, res) => {
    const { session: sessionName, role } = req.body

    let oRole

    if (role === 'publisher') {
        oRole = PUBLISHER
    } else if (role === 'subscriber') {
        oRole = SUBSCRIBER
    } else if (role === 'moderator') {
        oRole = MODERATOR
    } else {
        res.send({ error: 'You must specify role in request' })
        return
    }

    // Check auth and set username
    if (oRole === MODERATOR) {
        return
    }

    // Update active sessions list
    await OV.fetch()

    let session = OV.activeSessions.find(x => x.sessionId === sessionName)

    if (!session) {
        console.log(`New session: ${sessionName}`)
        session = await OV.createSession({ customSessionId: sessionName })
    }

    const connection = await session.createConnection({
        role: oRole
    })

    res.send({
        token: connection.token
    })
})
```

Рисунок 5.8 – Обробка запита на підключення або створення сесії

3. Розробка клієнтської частини нашого додатку

Інтерфейс повинен надавати можливість створювати або підключатися до готової сесії та відображати відео співрозмовників.

Для розробки клієнтської частини був обраний фреймворк React. Він отримує доступ до мікрофона та камери, та підключається до нашого серверного додатку для авторизації та отримання токену. З отриманим токеном він підключається до медіа-сервера та починає транслювати свій потік а також отримувати та відтворювати потік співрозмовників.

```
const connect = async () => {
    const token = await getToken(sessionName, role)

    window.onbeforeunload = () => {
        session.disconnect()
    }

    session.on('streamCreated', (event: any) => {
        const subscriber = session.subscribe(event.stream, '')
        setSubscribers(oldState => [...oldState, subscriber])
    })

    session.on('streamDestroyed', (event: any) => {
        setSubscribers(oldState => oldState.filter(item => item !== event.stream.streamManager))
    })

    await session.connect(token, JSON.stringify({ username: mcef.username }))
}

useDidMount(async () => {
    OV = new OpenVidu()
    session = OV.initSession()

    await mcef.init()

    await connect()
    await devices.init(OV)

    const publisher = OV.initPublisher('', {
        audioSource: devices.hasMicrophone ? devices.microphone!.deviceId : false,
        videoSource: devices.hasCamera ? devices.camera!.deviceId : false
    })

    session.on('publisherStartSpeaking', (event: any) => {
        console.log(`Publisher ${event.connection.connectionId} start speaking`)

        if (event.connection.connectionId &&
            event.connection.connectionId !== publisher?.stream.connection.connectionId) {
                setNowSpeaking(event)
            }
    })

    setPublisher(publisher)
    session.publish(publisher)

    setLoaded(true)
})
```

Рисунок 5.9 – Код ініціалізації сторінки відеочату

6. ПОДАЛЬШИЙ РОЗВИТОК

На даний момент наш продукт перебуває у стадії мінімально життєздатного продукту (MVP). Для створення повноцінної робочої платформи потрібно сконцентруватися на розвитку продукту, включаючи не лише аспекти технічної реалізації, а й на розвитку освітньої складової, способів поширення та підвищення впізнаваності продукту за допомогою грамотного використання можливостей, які надає сама платформа.

Головним пріоритетом, окрім лояльності користувачів до платформи, є підтримка LTV (життєвої цінності клієнта), що є показником прибутку, отриманого компанією за весь час взаємодії з конкретним покупцем[8]. Для досягнення цієї мети продукту потрібно багато контенту (віртуальних світів та освітнього матеріалу) для тривалішого утримання користувача. Щоб забезпечити платформу такою великою кількістю контенту, потрібно перейти до конвеєрного процесу розробки контенту. Був створений план конвеєрного підходу до розробки, який складається з 5 основних процесів.

Рисунок 6.1 – План конвейерного підходу до розробки

6.1 Сценарій

Весь процес навчання відбувається в окремих віртуальних світах, які створюються для окремих дисциплін. Для виконання кейсу розробляється сценарій світу та історії в ньому.

Сюжет може бути створений за мотивами:

1. Історичних подій,
2. Літературних творів,
3. Сюжетів фільмів або мультфільмів,
4. Будь яких інших фантастичних творів.

Це також не виключає розробку власних унікальних сюжетів, які не базуються на існуючих творах і, таким чином, підлягають створенню та розвитку свого власного всесвіту всередині гри зі своїми персонажами.

6.2 Дизайн персонажів

Розробка віртуального світу повинна також включати дизайн зовнішнього вигляду діючих персонажів. Це можуть бути як існуючі історичні персонажі, які постають в межах створеної легенди віртуального світу, так і персонажі, які ми самі створюємо. Дизайн персонажів адаптований до стилістики самої гри.

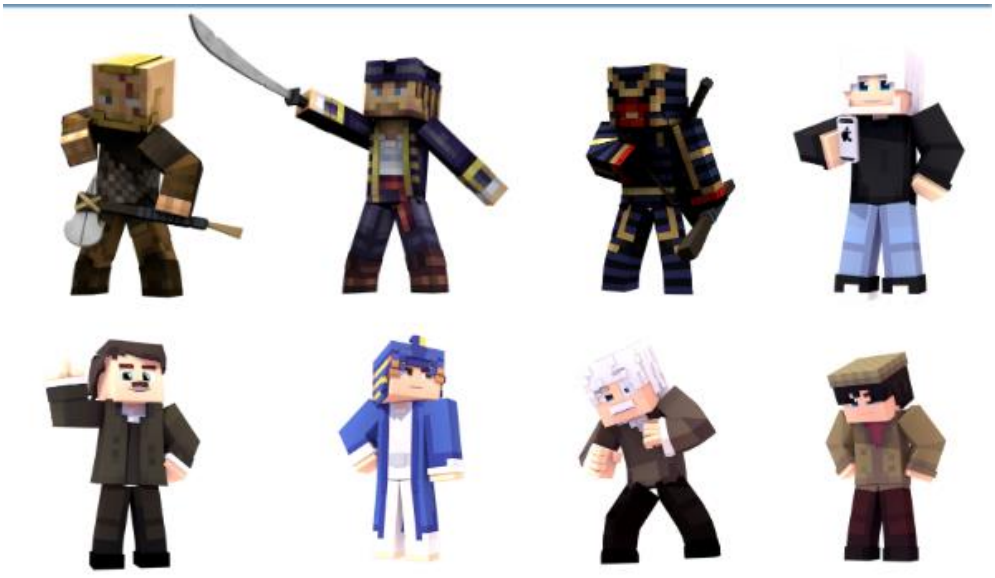


Рисунок 6.2 – Внутрішньоігрові персонажі

6.3 Підготовка локацій

Сама гра надає величезні можливості щодо побудови власних локацій. На сьогоднішній день існує велика кількість готових локацій та інструментів для спрощення процесу їх побудови. Це дає нам можливість побудувати як фантастичні чи історичні локації, так і перекласти існуючі локації з нашого світу в онлайн.

Рисунок 6.3 – Приклад існуючої локації побудованої в грі

Сама гра має велику популярність, що дозволяє привертати до побудови локацій гравців зі спільноти. Як приклад, можна навести уже існуючі великі світи, створені спільнотою за мотивами всесвітів, таких як "Гра Престолів" або "Гаррі Поттер".

Рисунок 6.4 – Мапа Вестероса із Гри Престолів перенесена повністю у грі

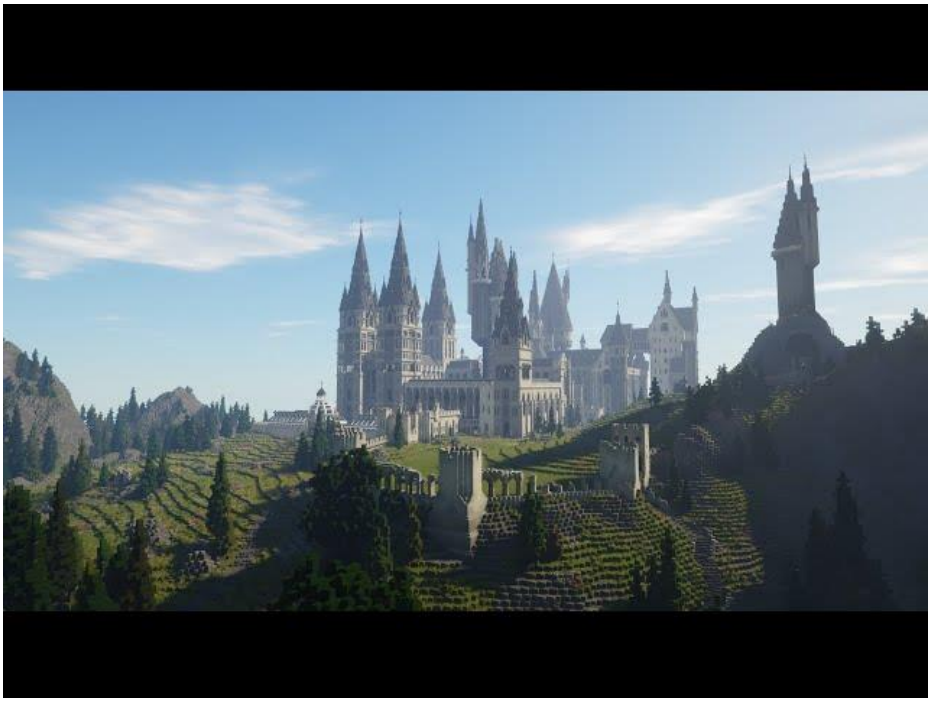


Рисунок 6.5 – Мапа Хогварсту із вселеної Гарри Поттер яка перенесена в гру

6.4 Програмування

Процес програмування включає розробку скрипта квестів, логіки грального механіки та освітньої частини.

Маючи на поточний момент робочий інструмент для написання сценаріїв грального світів з використанням JavaScript та робочу платформу,

ми можемо залучати освітніх методистів для інтеграції методік навчання шляхом створення грального механік всередині свого продукту. До процесу реалізації у вигляді програмного коду можна залучати як студентів університетів, так як на перший час був зроблений акцент на створення дуже простих програмних інтерфейсів, так і учнів, які в подальшому можуть навчатися розробці на JavaScript, зокрема використовуючи нашу власну платформу.

6.5 Тестування

Для будь-якого проекту на цьому етапі дуже важливо визначитись з ефективністю самого продукту та окремих його складових частин для формулювання чіткого плану його розвитку. Для цього необхідно збирати невеликі групи людей та проводити ігрові тести. Ігрові тести повинні супроводжуватися зворотним зв'язком від самих гравців та збором різноманітної інформації. Для цього дуже важливо інтегрувати інструменти, які збиратимуть всілякі метрики на основі яких в подальшому буде проводитись вимірювання навчального чи будь-якого іншого ефекту.

Приклад метрик:

1. Натискання на кнопки або внутригрові елементи (важіль, плити, інші механізми).
2. Пересування користувачів по віртуальних локаціях. (Збір координат та побудова теплових карт).
3. Факт виконання завдання або механіки.
4. Час виконання завдання.
5. Помилки під час виконання завдання.

ВИСНОВКИ

У рамках кваліфікаційної роботи було досліджено потенціал застосування методів gama-based у навчальному процесі. Наші дослідження показали, що використання такого методу може покращити якість освіти. Розроблено нову освітню платформу, використовуючи передові технології. Наша розробка базується на існуючій грі, що дозволило знизити вартість розробки і поєднати механізми навчання з елементами гри, зробивши процес більш ефективним та цікавим для студентів. Платформа розроблена з урахуванням індивідуальних потреб користувачів і з можливістю розширення на будь-які дисципліни. Також надано опис подальшого розвитку платформи, створивши план конвейєрного підходу до розробки, включаючи всі етапи розробки та дозволяючи масштабувати наш проект. Ми впевнені, що наш продукт не тільки задовольнить потреби користувачів, але й допоможе підвищити ефективність процесу навчання та освоєння нових знань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Minecraft Education Edition [Електронний ресурс] // Minecraft. – 2023. - Режим доступу до ресурсу: https://education.minecraft.net/
2. About Duolingo [Електронний ресурс] // duolingo. - Режим доступу до ресурсу: https://research.duolingo.com/
3. About CodeCombat [Електронний ресурс] // codecombat. – Режим доступу до ресурсу: https://codecombat.com/about
4. Minecraft [Електронний ресурс] // Minecraft. – 2023. - Режим доступу до ресурсу: https://minecraft.net/
5. Fabric Wiki [Електронний ресурс] // fabricmc. – Режим доступу до ресурсу: https://fabricmc.net/wiki/start.
6. Chromium Embedded Framework [Електронний ресурс] // Вікіпедія. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Chromium_Embedded_Framework.
7. Johnston, A.B. WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web. Boston, MA: Pearson Education, 2012.
8. Narayandas, D. Customer Centricity: Focus on the Right Customers for Strategic Advantage. Boston, MA: Harvard Business Review Press, 2017.