КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет автоматизації інформаційних технологій

Кафедра інформаційні технології

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

на тему:

Розробка ігрового додатку для Minecraft: Witcher Mod

Бєлов Дмитро Борисович

Київ 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизацій і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТ
д.т.н., професор Гончаренко Т.А.
_____
“___” _____2025 року

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

на тему: «Розробка ігрового додатку для Minecraft: Witcher Mod»

Я як здобувач вищої освіти КНУБА розумію і підтримую політику закладу з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Здобувач Бєлов Дмитро Борисович

122 «Комп’ютерні науки»

Інформаційні управляючі системи і технології

Групи КН-21-2_____

Керівники Гончаренко Т.А., д.т.н., професор,

Мацієвський О. О., асистент

Рецензент к.т.н., доц. Доля О.В.

Ідентичність підтверджую

Київ 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет:	Автоматизації інформаційних технологій
Випускова кафедра:	Інформаційних технологій
Освітній ступінь:	Бакалавр
Спеціальність:	Комп'ютерні науки
Освітня програма:	Інформаційні управляючі системи та технології

ЗАТВЕРДЖУЮ
Завідувачка кафедри ІТ
д.т.н., професор Гончаренко Т.А.
_____
„___" _____2025 року

З А В Д А Н Н Я
ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА
ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

Бєлов Дмитро Борисович
1. Тема роботи - Розробка ігрового додатку для Minecraft: Witcher Mod

затверджена наказом ректора КНУБА № 235/23/25 від "14" лютого 2025 року

2.Керівники роботи	Гончаренко Тетяна Андріївна, д.т.н. професор Мацієвський Олексій Олегович, асистент

3. Строк подання студентом роботи до захисту: травень 2025 р.

4. Зміст пояснювальної записки за розділами:

Р.1	ТЕМАТИКА ТА ІГРОВИЙ СВІТ "ВІДЬМАКА" У КОНТЕКСТІ MINECRAFT
Р.2	ТЕХНОЛОГІЇ ТА ОСОБЛИВОСТІ РОЗРОБКИ МОДІВ ДЛЯ MINECRAFT
Р.3	РЕАЛІЗАЦІЯ ФУНКЦІОНАЛУ WITCHED MOD: РОЗРОБКА МОДЕЛЕЙ, ТЕКСТУР І МЕХАНІК
Р.4	ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ МОДИФІКАЦІЇ

5. Графічний матеріал за розділами:

Робота викладена 65 аркушах, містить 7 таблиць, 20 рисунки, список використаної літератури із 16 найменувань.

6. Календарний план виконання роботи:

Види робіт та їх зміст	Дата виконання
Розділ 1	17.03.2025
Розділ 2	20.04.2025
Розділ 3	15.05.2025
Розділ 4	02.06.2025
Остаточне оформлення роботи	08.06.2025
Направлення роботи для перевірки на плагіат	
Попередній захист роботи на випусковій кафедрі	
Направлення роботи на рецензування	

7. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта	Перевірив	
		Дата	Підпис
Розділ 1.	Мацієвський О. О., асистент кафедри ІТ	17.03.2025	
Розділ 2.	Рябчун Ю.В., доц.каф.ІТ	20.04.2025	
Розділ 3.	Мацієвський О.О., асистент кафедри ІТ	15.05.2025	
Розділ 4.	Т Рябчун Ю.В., доц.каф.ІТ	02.06.2025	

8. Дата видачі завдання листопад 2025 р.

Зав. кафедри			Гончаренко Т.А.
	(підпис)		(прізвище та ініціали)
Керівники			Гончаренко Т.А.
			Мацієвський О.О.
	(підпис)		(прізвище та ініціали)
Здобувач			Бєлов Д.Б.
	(підпис)		(прізвище та ініціали)

АНОТАЦІЯ

Бєлов Д.Б. Розробробка ігрового додатку Witcher Mod для Minecraft. Атестаційна випускна робота бакалавра за спеціальністю: 122 «Комп’ютерні науки». - Київський національний університет будівництва та архітектури. - Київ, 2025.

Обсяг роботи: 65 сторінок, 7 таблиць, 20 рисунків, 16 джерел.

Шифр та назва напряму підготовки: 122 Комп’ютерні науки.

Мета та основний результат роботи: Метою даної роботи є розробка тематичної модифікації *Witcher Mod* для гри Minecraft, яка відтворює ключові елементи всесвіту «Відьмака», поєднуючи геймплейні механіки серії з можливостями гри-пісочниці. У межах проєкту реалізовано додавання зброї, броні, монстрів, нових блоків, торгової системи, генерації ресурсів, кастомних ефектів та ігрового прогресу, з урахуванням канонічного лору та ігрового балансу.

Основний результат роботи полягає у створенні повноцінної модифікації для Minecraft (версія 1.19.2), що підтримує власну архітектуру предметів, мобів, блоків та механік, а також у забезпеченні її стабільного функціонування, сумісності з рушієм гри та потенціалу для подальшого розвитку. Також продемонстровано можливості застосування сучасних інструментів розробки, таких як Forge, GeckoLib, Blockbench та Data Generators для ефективної реалізації модифікації.

Ключові слова: Minecraft, модифікація, Forge, Witcher Mod, геймдизайн, моделювання, Java, Blockbench, GeckoLib, Data Generator, ігрові механіки, броня, зброя, моби, лут, RPG, ігровий процес.

ABSTRACT

Belov D.B. *Development of the Witcher Mod Game Add-on for Minecraft.* Bachelor's qualification thesis in the specialty 122 "Computer Science". – Kyiv National University of Construction and Architecture. – Kyiv, 2025.

Scope of the work: 65 pages, 7 tables, 20 figures, 16 references.

Specialty code and name: 122 Computer Science.

Objective and main result of the work: The aim of this work is to develop a thematic modification — *Witcher Mod* — for the game *Minecraft*, which recreates key elements of the "Witcher" universe, combining the gameplay mechanics of the series with the sandbox nature of Minecraft. The project includes the implementation of custom weapons, armor, monsters, new blocks, a trading system, resource generation, custom effects, and gameplay progression while adhering to canonical lore and game balance.

The main result of the work is the creation of a complete modification for *Minecraft* (version 1.19.2) featuring a custom architecture for items, mobs, blocks, and mechanics. The mod demonstrates stable functionality, compatibility with the game engine, and potential for further development. The project also showcases the use of modern development tools such as *Forge*, *GeckoLib*, *Blockbench*, and *Data Generators* for efficient mod implementation.

Keywords: Minecraft, modification, Forge, Witcher Mod, game design, modeling, Java, Blockbench, GeckoLib, Data Generator, game mechanics, armor, weapons, mobs, loot, RPG, gameplay.

Зміст

Вступ 9

Структура роботи: 11

1. Тематика та ігровий світ "Відьмака" у контексті Minecraft 12

1.1 Ігровий всесвіт «Відьмака» 12

1.2 Популярність франшизи «Відьмак» 14

1.3 Minecraft як середовище для адаптації 15

2. Технології та особливості розробки модів для Minecraft 18

2.1. Програмні засоби та середовище розробки 18

2.2 Вибір між Forge та NeoForged для створення модифікації 23

2.3 Інструменти для розробки модифікацій Minecraft 29

3. Реалізація функціоналу Witcher Mod: розробка моделей, текстур і механік 40

3.1 Загальний підхід до створення предмета в моді Minecraft (на прикладі Forge 1.19.2) 40

3.2 Особливості текстурування броні в Minecraft модах 43

3.3 Створення блоків у Minecraft моді 44

3.4 Додавання ефектів на броню в Minecraft моді 45

3.5 Реалізація власних Creative Tabs у Minecraft моді 47

3.6 Data Generation у Minecraft модах 49

3.7 Генерацію глобальних модифікаторів луту ModGlobalLootModifiersProvider 52

3.8 Вампіри в Minecraft моді з GeckoLib 53

3.9 Кастомна рослина 58

Розділ 4. Тестування та оцінка ефективності модифікації 60

4.1 Методика тестування 60

4.2 Виявлені помилки та способи усунення 60

4.3 Потенціал подальшого розвитку Witcher Mod 62

Висновок 64

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 66

Вступ

Minecraft є однією з найпопулярніших відеоігор у світі, маючи понад 300 мільйонів проданих копій та мільйони активних гравців щомісяця. Її популярність пояснюється не лише відкритим світом, творчою свободою та нескінченними можливостями дослідження, а й потужною системою модифікації гри, яка дозволяє користувачам створювати власний контент. Ця особливість відкриває двері для реалізації нових ідей, механік, персонажів та сюжетів, що значно розширюють базовий ігровий досвід.

Спільнота моддерів Minecraft активно розвивається і щоденно випускає тисячі нових модифікацій, які змінюють або доповнюють гру: додають нові блоки, предмети, біоми, мобів, інтерфейси, а іноді — навіть повністю нові виміри та правила гри. Модифікації не лише урізноманітнюють геймплей, а й стають інструментом для навчання програмуванню, 3D-моделюванню та ігровому дизайну.

Серед безлічі існуючих модів особливе місце посідають тематичні модифікації, що базуються на відомих франшизах. Такі моди дають змогу фанатам перенести улюблені світи у пісочницю Minecraft, зберігаючи їхню атмосферу та ключові ігрові елементи. Одним із таких проєктів є Witcher Mod — модифікація, що інтегрує у Minecraft контент із всесвіту "Відьмака" — серії книг Анджея Сапковського, відомої також завдяки популярним відеоіграм і серіалам.

Мета роботи: створення модифікації (мода) для гри Minecraft, яка додає до гри елементи світу "Відьмака", зокрема нову зброю, броню, класифікацію відьмацьких шкіл, вампірів, приготування їжі, рецепти, торгову систему та генерацію ресурсів у світі.

Об'єкт дослідження: процес модифікації гри Minecraft та її архітектура, яка дозволяє реалізовувати нові ігрові механіки через Forge API для версій 1.19.2 та 1.19.4.

Предмет дослідження: процеси розробки нових елементів у грі Minecraft, зокрема:

- додавання предметів та сутностей (мечі, броня, вампіри),
- реалізація приготування їжі й створення унікальних рецептів,
- створення системи торгівлі з NPC,
- генерація нових ресурсів у світі відповідно до лору "Відьмака".

Методи дослідження

У ході роботи було використано такі методи:

- Аналіз вихідного коду Minecraft і документації Forge API;
- Об'єктно-орієнтоване програмування для модульної побудови структури мода;
- Моделювання 3D-об’єктів і мобів у редакторі Blockbench;
- Проєктування крафт-рецептів і системи приготування їжі за допомогою JSON-конфігурацій;
- Інтеграційне тестування функціональності мода у тестовому середовищі Minecraft;
- Дослідницький підхід до побудови економіки торгівлі та розміщення ресурсів у світі гри.

Структура роботи:

Розділ 1. Тематика та ігровий світ "Відьмака" у контексті Minecraft
У цьому розділі проводиться аналіз літературного та ігрового всесвіту "Відьмака", визначаються ключові елементи, які можуть бути адаптовані до Minecraft. Також розглядається роль тематичних модів у геймінговій культурі та приклади схожих проєктів.

Розділ 2. Технології та особливості розробки модів для Minecraft
Розглядаються інструменти, мови програмування та архітектура модів. Деталізується процес налаштування середовища розробки, структура Forge-мода, підключення ресурсів.

Розділ 3. Реалізація функціоналу Witcher Mod: контент, механіки, інтеграція
У розділі описано процес реалізації основного функціоналу мода: додавання зброї, броні, кулінарії, торговельної системи, генерації ресурсів. Також пояснюється, як забезпечується відповідність контенту лору "Відьмака".

Розділ 4. Тестування та оцінка ефективності модифікації
Описується методика тестування мода в умовах гри, виявлені помилки та способи їх усунення. Проводиться аналіз балансу ігрових механік, зручності користування та потенціалу подальшого розвитку.

1. Тематика та ігровий світ "Відьмака" у контексті Minecraft

1.1 Ігровий всесвіт «Відьмака»

Всесвіт «Відьмака» бере свій початок з книжкової серії польського письменника Анджея Сапковського, яка охоплює романи та оповідання у жанрі темного фентезі. У центрі подій — Геральт із Рівії, мутований мисливець на чудовиськ, який користується магією, алхімією та бойовими навичками для знищення монстрів.

Головні риси світу «Відьмака» — це моральна неоднозначність, політичні інтриги, расові конфлікти та багатогранні персонажі. У відеоігровій адаптації, особливо в The Witcher 3: Wild Hunt, гравцям надається відкритий світ із розгалуженим сюжетом, сотнями квестів, глибоким бестіарієм та можливістю впливати на розвиток подій через вибір.

Ігрова механіка включає два типи мечів (сталевий для людей, срібний — для монстрів), магічні знаки (Ігні, Квен, Аард тощо), збір алхімічних інгредієнтів, прокачку персонажа за стилем RPG та складну бойову систему з ухиленнями, комбінаціями та тактичними паузами.

Крафт у всесвіті «Відьмака» — мистецтво виживання та підготовки

У грі «Відьмак» крафт є невід’ємною складовою ігрової механіки, що дає гравцям змогу глибше зануритись у світ і підготуватися до численних небезпек, що чатують на Ґеральта. Від створення зброї й обладунків до варіння зіль та виготовлення бомб — всі ці елементи формують тактичний арсенал мисливця на чудовиськ.

Основні аспекти крафту:

1) **Виготовлення зброї та обладунків**

Ґеральт може замовляти у ковальських майстрів виготовлення нових мечів і броні за схемами — кресленнями, знайденими або купленими у подорожах. Кожен предмет має рівень якості, а також можливість модифікації через вставлення руни або покращення

майстром. Крафтоване спорядження часто має унікальні властивості, що допомагають адаптуватись до конкретних ворогів чи стилю бою.

2) **Алхімічний крафт — приготування зіль, масел, відварів і бомб**

Одна з ключових особливостей «Відьмака» — алхімія, що дозволяє створювати потужні зілля для підсилення здібностей Ґеральта, а також спеціальні масла, які наносяться на мечі для додаткової шкоди певним типам монстрів. Бомби — вибухові або димові — використовуються для контролю бою чи завдання шкоди. Всі ці рецепти ґрунтуються на збиранні рідкісних інгредієнтів, які гравець може знайти у світі або добути з трупів монстрів.

3) **Збирання ресурсів і інгредієнтів**

Ефективний крафт неможливий без пошуку матеріалів — трав, рідкісних мінералів, тваринних шкур, та інших ресурсів. Ґеральт регулярно досліджує локації, збирає рослини, розбирає обладунки ворогів і добуває компоненти для виготовлення необхідних предметів. Це робить відкритий світ ще більш насиченим і інтерактивним.

4) **Модифікації та покращення**

Після виготовлення або придбання зброї та обладунків їх можна покращувати, застосовуючи майстерність ковальства. Це дозволяє підвищувати характеристики предметів, збільшувати їхню міцність і бойову ефективність, що є життєво необхідним для виживання у більш складних битвах.

5) **Стратегічне значення крафту**

Крафт у «Відьмаку» — це не просто механічна дія, а частина стратегії. Гравець повинен враховувати слабкості ворогів і заздалегідь готуватися до сутичок: правильно підібрані масла і зіль можуть змінити перебіг бою, а добротна броня і меч — врятувати життя. Такий підхід до крафту додає глибини і тактичної варіативності.

1.2 Популярність франшизи «Відьмак»

Франшиза «Відьмак» має багатомільйонну фан-базу по всьому світу. Основні статистичні дані свідчать про її надзвичайний успіх:

- Продажі The Witcher 3: Wild Hunt перевищили 50 мільйонів копій станом на травень 2023 року. Це дозволило грі увійти до топ-10 найбільш продаваних ігор в історії.
- Сумарні продажі всіх ігор франшизи «Відьмак» перевищили 75 мільйонів копій.
- Серіал "Відьмак" від Netflix, прем’єра якого відбулася у грудні 2019 року, зібрав **76** мільйонів переглядів за перший місяць, що зробило його одним із найуспішніших фентезі-проєктів платформи на той момент.
- **Книжки Анджея Сапковського** були перекладені понад на 30 мов, а сумарні продажі перевищили 15 мільйонів копій.

Ці дані вказують на стабільний та високий рівень популярності франшизи серед фанатів фентезі, геймерів і широкої масової аудиторії. Відповідно, створення модифікації за мотивами «Відьмака» має високий потенціал для популярності, оскільки поєднує впізнаваний всесвіт із гнучкою платформою Minecraft.

Рисунок 1.1 Офіційні дані. Продано 50 млн. копій

1.3 Minecraft як середовище для адаптації

Minecraft є найбільш продаваною грою в історії, з понад 300 мільйонами проданих копій (станом на жовтень 2023 року) та понад 200 мільйонами активних гравців щомісяця. Гра побудована за принципом пісочниці та має розгалужену систему модифікацій, яку підтримують платформи Forge, Fabric, Datapack тощо.

На сайтах на кшталт CurseForge та Modrinth щодня публікуються тисячі модів, які додають:

- нові механіки,
- зброю й броню,
- мобів і босів,
- інтерфейси, класи та здібності.

Серед найуспішніших тематичних модів можна відзначити:

- The Lord of the Rings Mod (50 тис. завантажень щомісяця),
- Star Wars Mod,
- Demon Slayer Mod,
- Harry Potter Mod.

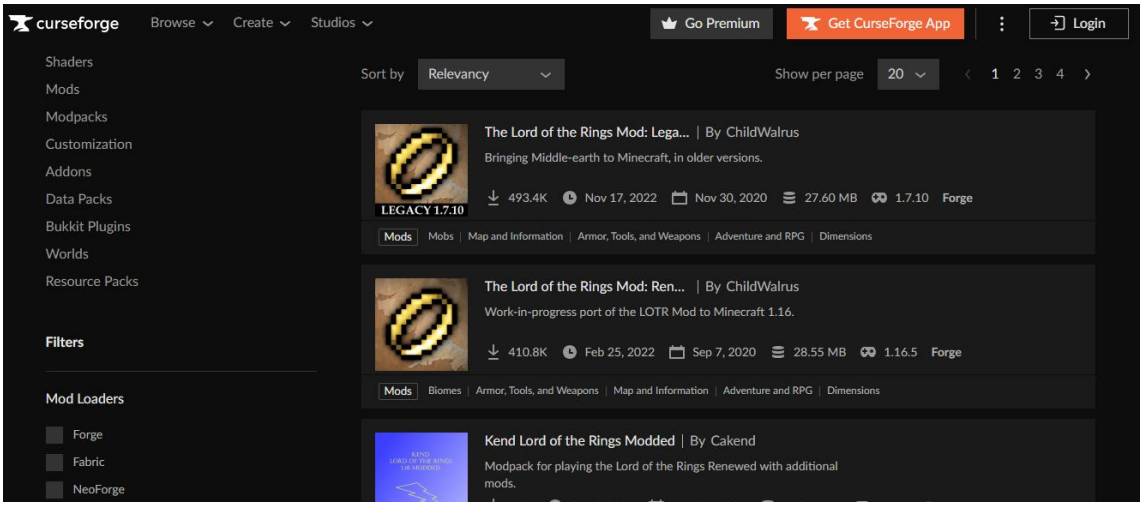

Рисунок 1.2 CurseForge. The Lord of the Rings Mod

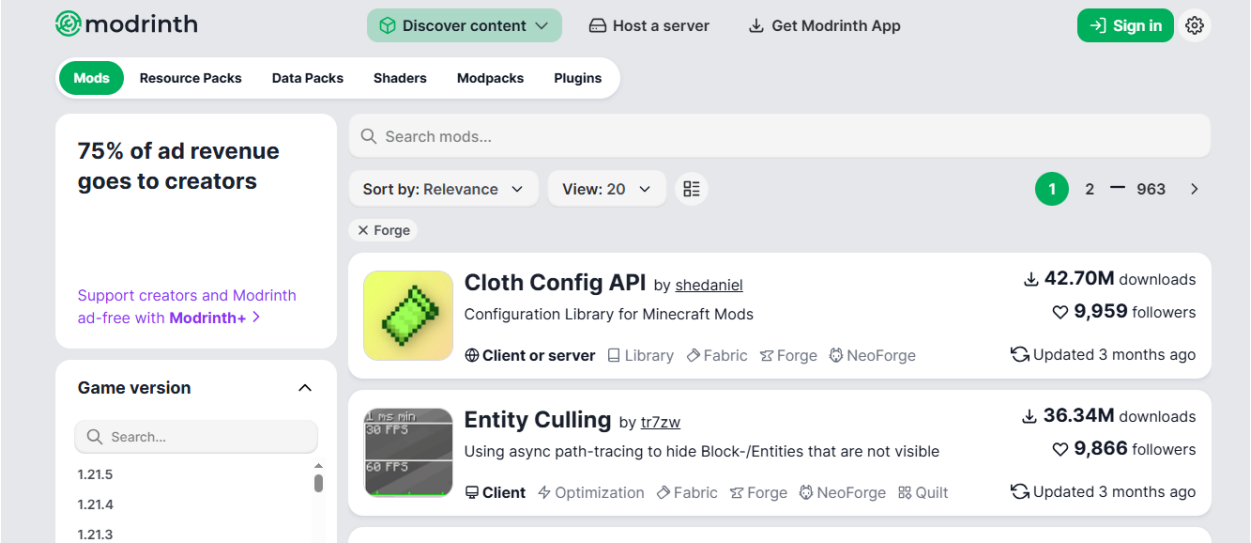

Рисунок 1.3 Modrinth

Приклад тематичного моду: The Lord of the Rings Mod: Renewed

У рамках дослідження адаптації всесвітів у Minecraft доцільно розглянути один із найуспішніших прикладів — **The Lord of the Rings Mod: Renewed** (скорочено — LOTR Mod: Renewed). Цей мод продемонстрував високий рівень інтеграції тематичного контенту у світ Minecraft та став еталоном для подібних проєктів.

Загальна характеристика

LOTR Mod: Renewed — це глобальна модифікація Minecraft, що переносить гравця у Середзем’я — вигаданий світ, створений Дж. Р. Р. Толкіном. Мод включає детально пророблену карту, фракції, раси, предмети, квести та ігрові механіки, які повністю змінюють геймплей Minecraft. Розробники ставили собі за мету не просто "перенести контент", а максимально автентично передати дух і логіку всесвіту "Володаря Перснів".

Ключові функції моду:

- Нове вимірювання – Middle-Earth (Середзем’я)

Мод додає окремий вимір із картою, що географічно відтворює регіони Середзем’я, включаючи Шир, Мінас-Тіріт, Мордор, Лотлоріен та інші.

-

- Фракції та репутація

Більше 30 фракцій (ельфи, гноми, орки, урук-хаї, люди різних земель). Гравець може приєднуватися до них, виконуючи завдання та отримуючи унікальні винагороди.

- Унікальні моби та раси

Ворожі та союзні істоти, такі як на́згули, енти, уруки, гобліни, хоббіти тощо. Кожна раса має свою поведінку, спорядження та поселення.

- Квестова система

Завдання NPC, гілки розвитку персонажа, вибір фракційної належності, динамічна репутація.

- Розширене фермерство та торгівл

Нові типи їжі, ремесел і механіка торгівлі з NPC, що відображає економіку кожного регіону.

- Предмети, броня, архітектура

Тематичні блоки (гондорський мармур, мордорський базальт тощо), спеціальна зброя (мечі ельфів, кільця, артефакти), оригінальна броня та щити.

Технічна реалізація

- Версії Minecraft: від 1.7.10 (класичний LOTR Mod) до 1.16.5 (Renewed).
- Розроблений на Forge, з підтримкою зовнішніх бібліотек (наприклад, GeckoLib).
- Має окремий лаунчер-контролер налаштувань.
- Власна система UI: мінікарта, журнал квестів, дипломатія з фракціями.

Статистика популярності

- Понад 2,5 млн завантажень на CurseForge
- Оцінка користувачів: 4.8 з 5.
- Активна спільнота Discord — понад 10 000 учасників.
- Сотні відео на YouTube з мільйонами переглядів (огляди, геймплей, гіди).

2. Технології та особливості розробки модів для Minecraft

2.1. Програмні засоби та середовище розробки

У розробці модів для Minecraft найважливішим вибором є вибір моддінг-платформи, яка визначає доступні інструменти, структуру проєкту, швидкість розробки, сумісність із іншими модами та продуктивність. Найпоширенішими платформами є **Forge**, **Fabric** та **Quilt**.

Таблиця 2.1

Порівняльна таблиця платформ для модифікацій Minecraft

Характеристика	Forge	NeoForged	Fabric	Quilt
Рік запуску	2011	2023 (форк Forge)	2018	2021
Мова розробки	Java	Java	Java	Java
Архітектура	Монолітна	Модульна, сучасні практики	Легка, модульна	Модульна (покращена версія Fabric)
Продуктивність	Середня	Вища за Forge	Висока	Висока
Підтримка версій Minecraft	1.7.10 – 1.20+	1.20+ (починаючи з 1.20.1)	Дуже швидке оновлення	Дуже швидке, навіть раніше за Fabric

Сумісність модів	Величезна екосистема	Зростає, частково несумісна з Forge	Обмежена, багато модів несумісні	Сумісний з більшістю модів Fabric
Гнучкість API	Важче розширювати	Гнучкий, чистіший API	Простий, але часто обмежений	Дуже модульний, підтримує нові API
Інструменти для розробки	ForgeGradle, GeckoLib	NeoGradle, GeckoLib	Loom, Architectury	Loom, Quilt Standard Libraries (QSL)
Ліцензування	Менш ліберальне	Відкрите, активне спільнотне	Дуже відкрите	Повністю відкритий проєкт
Кількість модів (оцінка)	~80 тис.	~5 тис. (і зростає)	~35 тис.	~5 тис.
Використання серед розробників	~60%	~10–15% (росте)	~25–30%	~3–5%
Основні проєкти	JEI, Mekanism, Tinkers' Construct	Modern Industrialization, Create Neo	Sodium, Lithium, Iris	Quilted Fabric API, QSL

Підтримка GeckoLib (моделі)	Так	Так	Так (через Fabric port)	Так
Простота для початківців	Важкий поріг входу	Краща документація, зручніше	Простий старт	Як у Fabric

Коротке пояснення:

Forge

- Найстаріша та найпопулярніша платформа.
- Складна архітектура, велика кількість готових API, але застарілі підходи.
- Добре підходить для великих, старих модів з підтримкою багатьох інструментів.

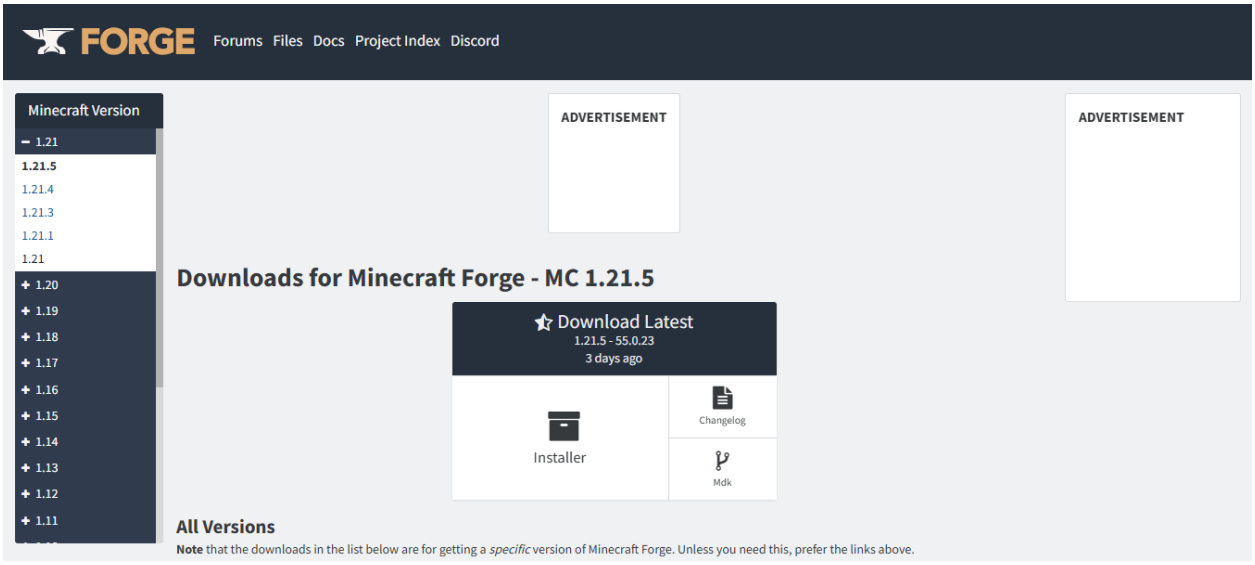

Рисунок 2.1 Головна сторінка Forge

NeoForged

- Новий проєкт, що виник у 2023 році як відповідь на стагнацію Forge.

- Краще структурування коду, швидше оновлення під нові версії Minecraft.
- Мета — зберегти сумісність з Forge-модами, але очистити кодову базу.

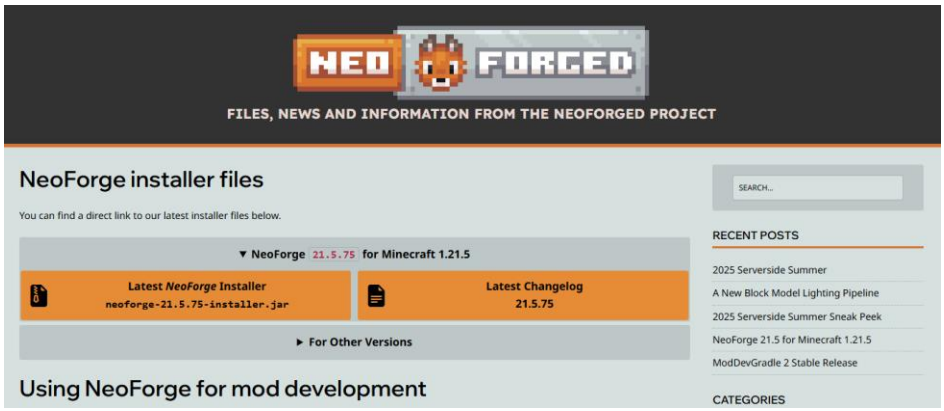

Рисунок 2.2 Головна сторінка NeoForged

Fabric

- Легка та швидка платформа, орієнтована на швидкий перехід до нових версій.
- Прекрасний вибір для клієнтських модів, оптимізацій та невеликих модифікацій.
- Часто не підходить для складних модів (через обмеженість API).

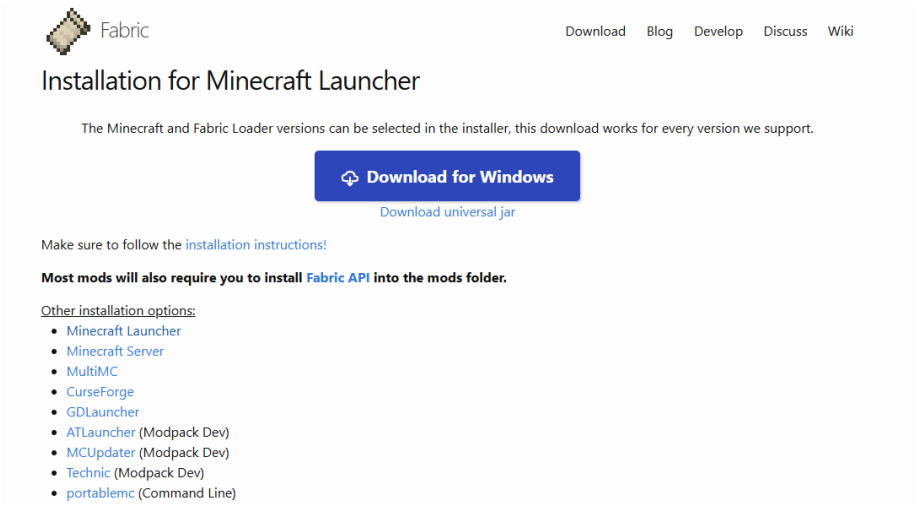

Рисунок 2.3 Головна сторінка Fabric

Quilt

- Розширення Fabric, що додає ще більше модульності та відкритості.
- Сумісний із більшістю Fabric-модів, має власний Quilt Standard Library.
- Поки що має невелику базу користувачів, але швидко зростає.

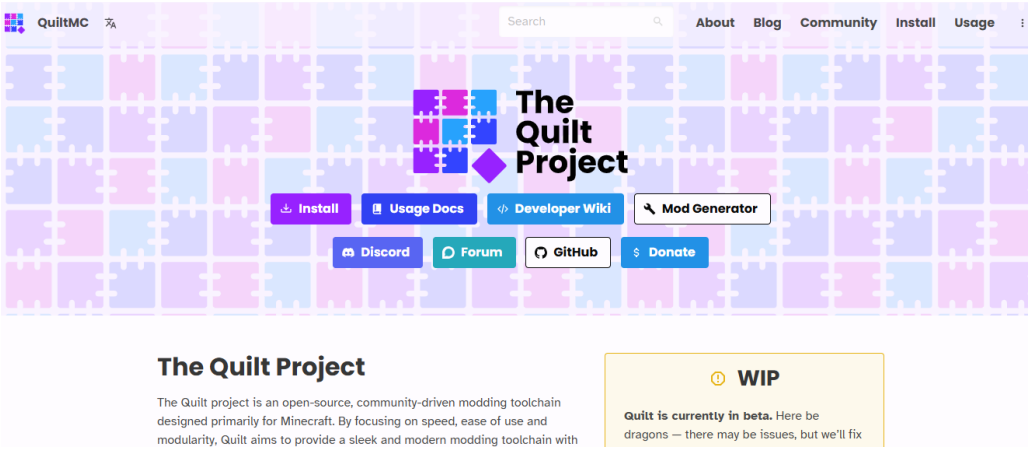

Рисунок 2.4 Головна сторінка quilt

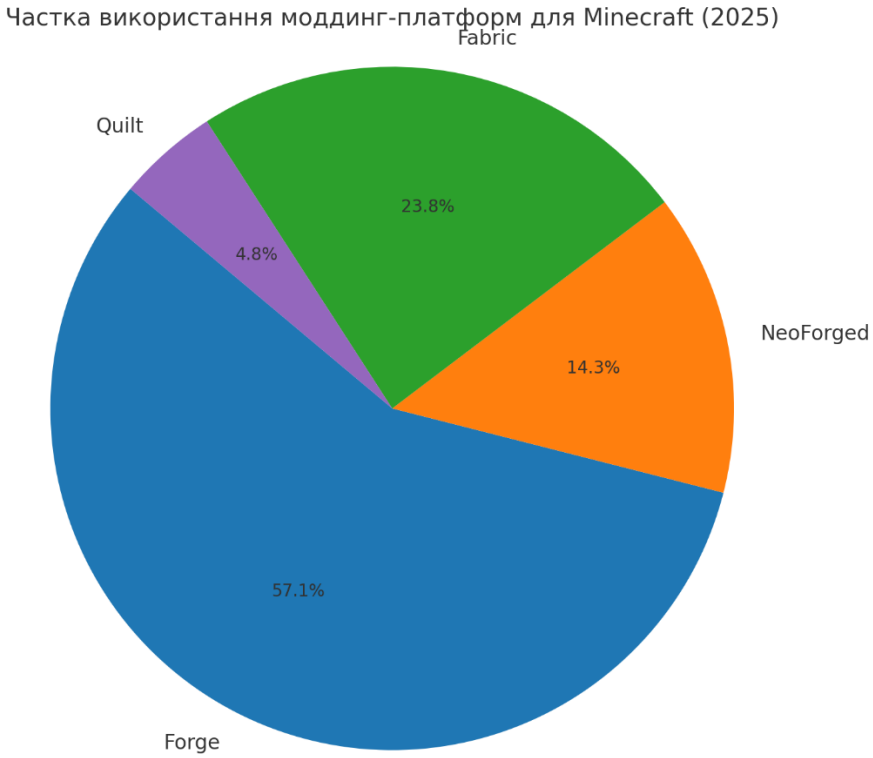

Рисунок 2.5 Графік, що ілюструє частку використання моддинг-платформ для Minecraft у 2025 році.

2.2 Вибір між Forge та NeoForged для створення модифікації

При створенні модифікації для Minecraft одним із перших рішень, яке маємо прийняти — це вибір між двома популярними платформами: Forge та NeoForged.

Forge

Forge — це найстаріша та найпоширеніша моддинг-платформа для Minecraft. Вона з’явилася ще в 2011 році та є стандартом де-факто для багатьох великих модів.

Переваги:

- Широка підтримка модів — Tinkers’ Construct, Ice and Fire, Biomes O' Plenty та сотні інших працюють на Forge.
- Розвинена екосистема — документація, форуми, GitHub, тисячі туторіалів.
- Стабільність — зріла та перевірена структура.
- Недоліки:
- Повільне оновлення до нових версій Minecraft.
- Важча структура API — більше шаблонного коду та класів.

NeoForged

NeoForged — нова моддинг-платформа, яка з’явилася в 2023 році як форк Forge. Вона зосереджена на спрощенні API, швидких оновленнях і сучасній структурі.

Переваги:

- Швидка адаптація до нових версій Minecraft (1.20+).
- Менше шаблонного коду — зручніша архітектура.
- Активний розвиток — відкритий GitHub, активна спільнота.
- Недоліки:
- Мала сумісність з наявними модами.
- Менше документації, хоча вона поступово розширюється.

Для більше точного порівняння зробемо **таблицю з технічним порівнянням Forge і NeoForged** з точки зору API, структури модів, сумісності та зручності розробки:

Таблиця 2.2

Критерій	Forge	NeoForged
Версії Minecraft	Підтримує більшість версій (особливо 1.12.2 – 1.20.1)	Переважно нові версії (з 1.19.4 і вище)
Структура модів	Більш класична, дещо застаріла	Сучасна, зменшена кількість boilerplate-коду
Ініціалізація (Mod Lifecycle)	Використовує @Mod, FMLCommonSetupEvent, ClientSetupEvent тощо	Схожа система, але з очищеною архітектурою
Реєстрація об'єктів	Через DeferredRegister + RegistryObject	Теж DeferredRegister, але краще API для розширень
Сумісність з іншими модами	Висока – підтримується багатьма модами	Часткова – нові моди поступово переходять
Документація та туторіали	Багато уроків, відео, прикладів	Менше, але документація активно поповнюється

Інтеграція з Gradle	Потребує ручного налаштування build.gradle	Краще інтегрована система залежностей (непотрібно правити руками JSON)
Підтримка Fabric-like API	Відсутня	Частково реалізовано через модульну структуру
Обробка подій (Events)	ForgeBus, ModBus	Подібна система, але з більшим контролем
Мережева синхронізація (Packets)	Через SimpleChannel з кодом вручну	Простішим способом із шаблонами
Створення кастомної логіки	Потребує більше boilerplate	Зменшено кількість необхідного коду
Гнучкість API	Менш модульний	Краще структурований модульний API
Час оновлення під нові версії	Повільніше	Значно швидше

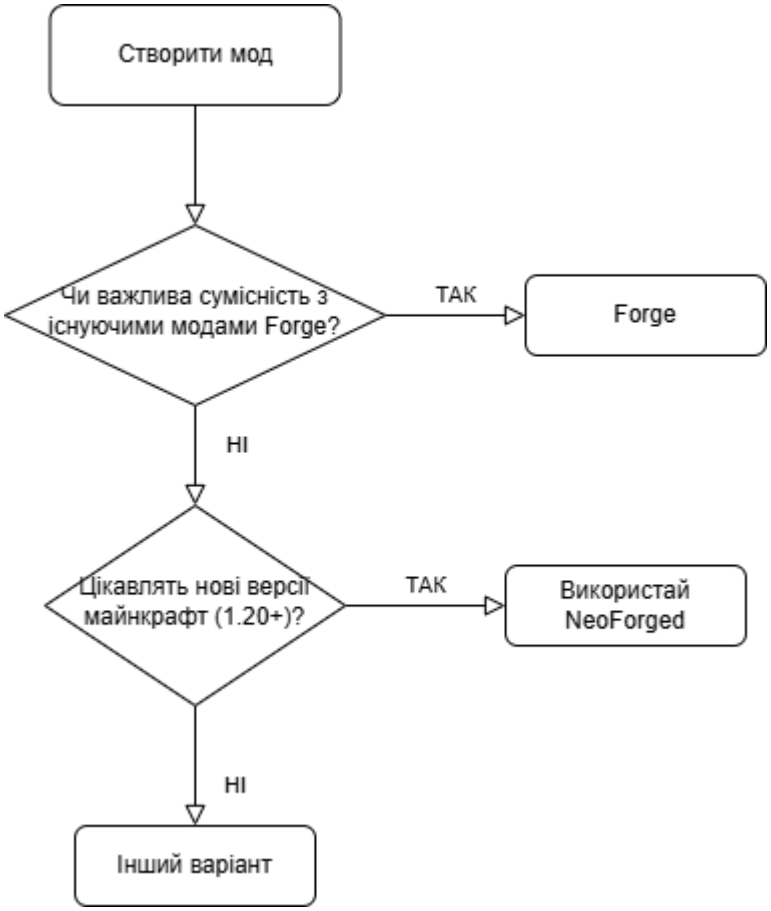

Рисунок 2.6 Діаграма вибору між Forge і NeoForged

Практичне порівняння Forge і NeoForged: приклад реєстрації предмета

Щоб оцінити різницю між Forge та NeoForged з погляду розробника, розглянемо найпростіший приклад: **реєстрацію предмета (Item)**. Цей процес показує, наскільки багато коду потрібно написати, як організована структура, і як виглядає API.

Реєстрація предмета у Forge (Minecraft 1.19.2)

```
public class ModItems {
    public static final DeferredRegister<Item> ITEMS =
        DeferredRegister.create(ForgeRegistries.ITEMS, "examplemod");
    public static final RegistryObject<Item> EXAMPLE_ITEM =
        ITEMS.register("example_item", () -> new Item(new Item.Properties()));
```

```
    public static void register(IEventBus eventBus) {
        ITEMS.register(eventBus);
    }
}
```

✓ Знайома структура, стабільність.

✗ Багато шаблонного коду

Реєстрація предмета у NeoForged (Minecraft 1.20.4+)

```
@Mod("examplemod")
public class ExampleMod {
    public static final RegistrySupplier<Item> EXAMPLE_ITEM =
        Registries.ITEMS.register("example_item", () ->
            new Item(new Item.Properties()));
    public ExampleMod(IEventBus eventBus) {
      Registries.ITEMS.register(eventBus);
    }}
```

✓ **Переваги**: менше коду, немає необхідності створювати окремий клас лише для предметів.

✗ **Недоліки**: потрібно звикнути до оновленої системи RegistrySupplier і змін у назвах класів/структурі.

Висновок

Вибір між **Forge** і **NeoForged** залежить насамперед від пріоритетів розробника та цілей самого проєкту.

Forge — це стабільна, перевірена роками платформа, яка має широку підтримку в моддерській спільноті. Якщо основна мета — створення мода, який буде сумісним із великою кількістю вже існуючих модифікацій, особливо з тими, які не оновлювалися під нові версії Minecraft, то саме Forge є оптимальним вибором. Завдяки великій кількості документації, прикладів і гайдів, Forge також може бути корисним для новачків, які вперше знайомляться з моддингом, хоча початковий поріг входу через складну структуру API може здатися вищим.

У той же час, **NeoForged** є сучаснішою та активнішою альтернативою, яка була створена як відповідь на обмеження та повільні темпи розвитку Forge. NeoForged пропонує більш чисту архітектуру, менше шаблонного (boilerplate) коду та зручніший API, що робить його привабливим для досвідчених розробників, які хочуть будувати масштабовані та гнучкі модифікації. Завдяки швидшим оновленням під нові версії Minecraft, NeoForged дозволяє розробникам оперативно адаптуватися до останніх змін у грі та використовувати найновіші можливості.

Однак слід враховувати, що станом на 2025 рік **NeoForged ще не досяг такого рівня сумісності**, як Forge. Багато популярних модів ще не були перенесені на цю платформу, а отже, створення модів під NeoForged може обмежити гравців у використанні мода спільно з іншими.

Отже, вибір між Forge та NeoForged зводиться до компромісу між стабільністю та інноваційністю:

- Якщо ви створюєте мод для широкої аудиторії, плануєте інтеграцію з існуючими великими модами або вам важлива стабільність — обирайте **Forge**.
- Якщо ж ви хочете будувати сучасний, легкий у підтримці проєкт з новими підходами до архітектури, і готові миритися з меншою сумісністю — варто звернути увагу на **NeoForged**.

В ідеалі, доцільно також враховувати довгострокові плани: **для нових проєктів, орієнтованих на майбутнє та розвиток разом з Minecraft**, NeoForged є перспективнішою платформою, яка продовжує активно зростати й оновлюватися.

2.3 Інструменти для розробки модифікацій Minecraft

У процесі створення модифікацій Minecraft важливу роль відіграють не лише вибір платформи (Forge або NeoForged), а й набір інструментів, що використовуються для розробки, побудови, тестування та візуального дизайну контенту. Нижче описано основні з них, які становлять стандартний стек для моддерів:

IntelliJ IDEA / Eclipse

Це дві основні середовища розробки (IDE), які підтримують мову програмування Java і є повністю сумісними з проєктами Forge та NeoForged.

- **IntelliJ IDEA** вирізняється сучасним інтерфейсом, розширеними можливостями автодоповнення, рефакторингу, а також підтримкою Git, Gradle і багатьох плагінів.
- **Eclipse** більш легка й проста у використанні, що може бути перевагою для новачків. Водночас вона менш гнучка у роботі з сучасними фреймворками, і з нею частіше виникають конфлікти при роботі з Gradle.

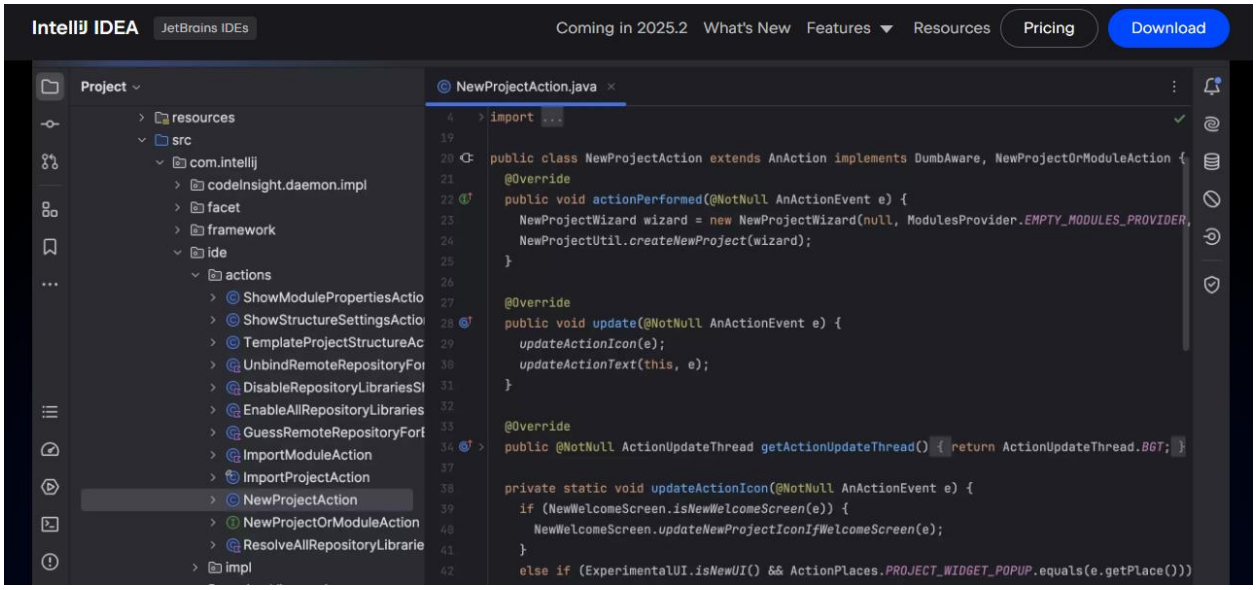

Рисунок 2.7 Головна сторінка IntelliJ IDEA

Вибір між цими IDE зазвичай залежить від особистих уподобань розробника, однак IntelliJ IDEA рекомендується для більш продуктивної та сучасної розробки, особливо у великих проєктах.

Gradle

Gradle — це система автоматизації складання проєкту, яка є невід'ємною частиною моддингу Minecraft. Вона використовується для:

управління залежностями (наприклад, GeckoLib),

компіляції коду,запуску тестів, автоматизованого запуску гри з підключеним модом.

Gradle дозволяє уникати ручного налаштування структури проєкту та дає змогу працювати за принципом «один раз налаштував — запускай будь-де». Усі офіційні шаблони для Forge та NeoForged використовують Gradle як основу.

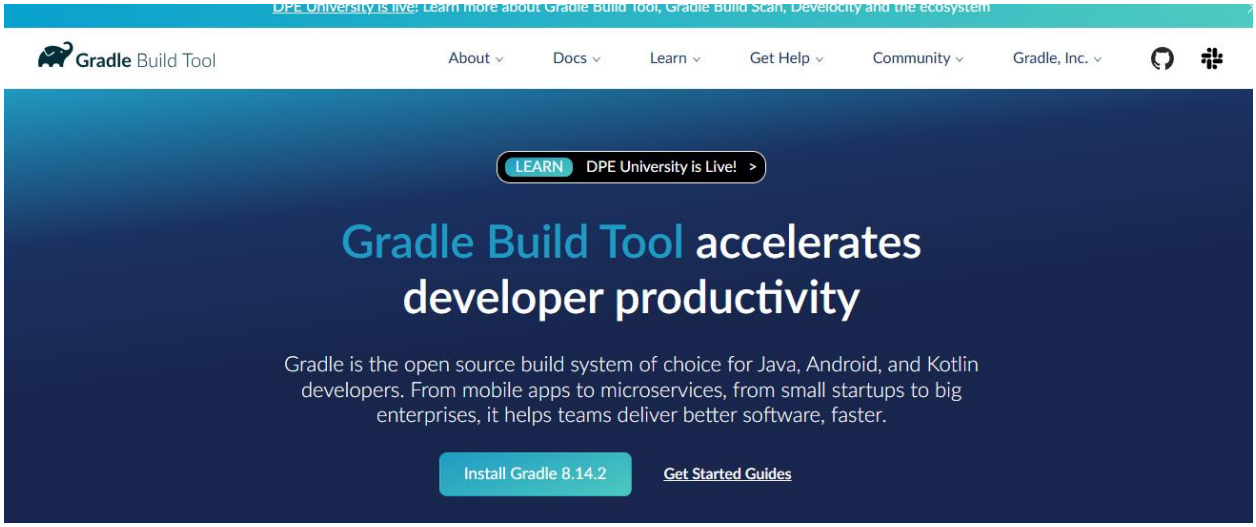

Рисунок 2.8 Gradle

Blockbench – візуальний 3D-редактор моделей та анімацій

Blockbench — це спеціалізований 3D-редактор, створений для роботи з кубічними (воксельними) моделями у стилі Minecraft. Його основне призначення — розробка тривимірних моделей мобів, предметів, блоків, а також створення кісткових анімацій (тобто анімацій, що ґрунтуються на скелеті моделі).

Основні функції:

1. Моделювання:
    a. Створення моделей мобів, тварин, монстрів, зброї, броні, меблів тощо.
    b. Усі моделі створюються з простих прямокутних блоків, подібно до стилю Minecraft.
    c. Кожен елемент можна розміщувати, масштабувати, обертати, фарбувати або прив'язувати до "кістки".
2. Створення кісткової структури (bones):
    a. Дозволяє розділити модель на частини (наприклад: голова, тіло, руки, ноги).
    b. Кожна частина пов’язується з "кісткою", яку потім можна анімувати окремо.
    c. Це важливо для мобів з рухомими частинами, як-от монстри, істоти, дракони, птахи.

3. Анімації:
    a. Blockbench має таймлайн-аніматор — як у професійних 3D редакторах.
    b. Можна створити анімацію ходьби, атаки, смерті, руху крил, хвоста тощо.
    c. Анімація зберігається у форматі .animation.json, який потім використовується в GeckoLib.
4. Експорт до Minecraft (Java Edition):
    a. Можна експортувати модель у формат .java або .json, сумісний з Minecraft.
    b. При створенні мобів обирається шаблон GeckoLib Model (Java Entity).
    c. Після експорту модель можна імпортувати до коду модифікації та відтворювати через GeckoLib.

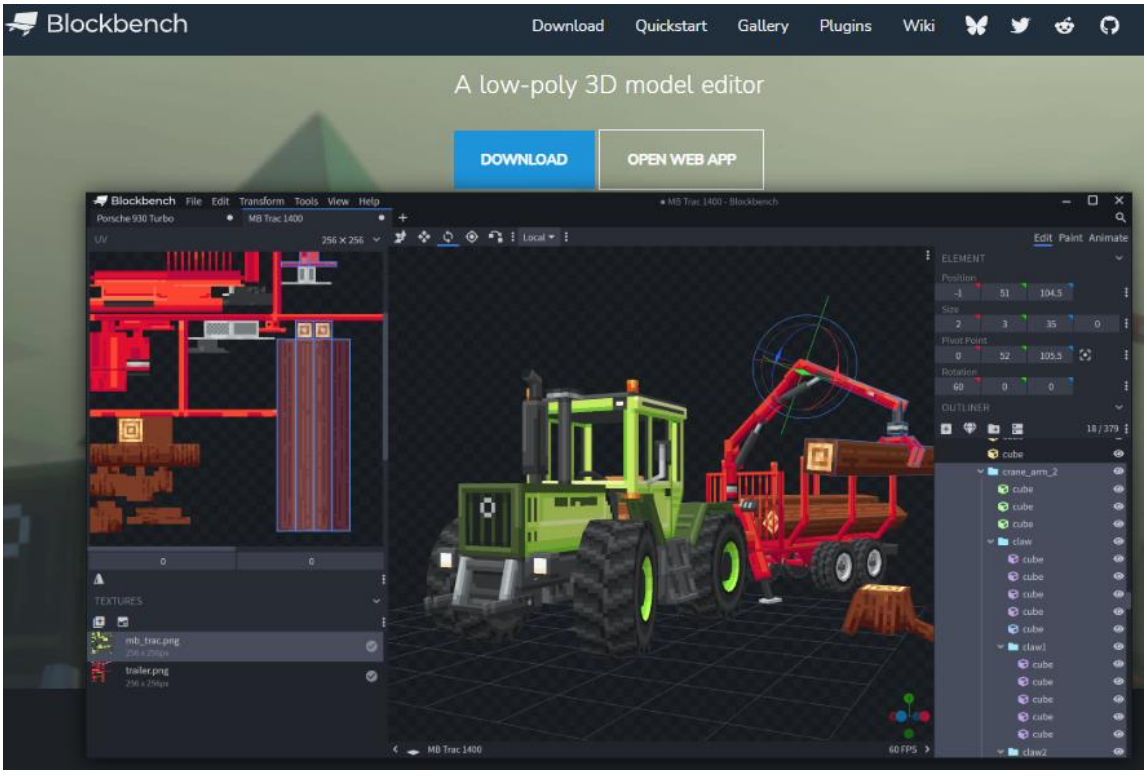

Рисунок 2.9 Blockbench

GeckoLib – бібліотека анімацій для Minecraft

GeckoLib — це потужна бібліотека, яка дозволяє використовувати анімації з Blockbench без написання складного коду вручну. Вона підтримує складну логіку анімацій для мобів, предметів та блоків у модах Minecraft.

Що робить GeckoLib:

1. Зчитує .json файли анімацій з Blockbench:
    a. Можна створити анімацію візуально, а потім просто імпортувати до гри.
    b. Не потрібно вручну задавати кути повороту, позицію чи час — усе робиться графічно.

2. Працює з "кістками":
    a. Кожна частина моделі (наприклад, рука або крила) керується окремою кісткою.
    b. GeckoLib дозволяє задавати логіку — коли і яка кістка має рухатися.

3. Динамічні анімації в залежності від стану моба:
    a. Наприклад:
        i. Якщо моб іде — включається анімація ходьби.
        ii. Якщо моб атакує — запускається анімація удару.
        iii. Якщо моб стоїть — працює idle animation (бездіяльність).
    b. Усе це налаштовується через **AnimationController.**

4. Можливість запуску анімацій зі сценарієм (іф-умовами):
    a. Ви можете запускати анімацію тільки при певних умовах, наприклад:
        i. Якщо гравець близько — моб ричить.
        ii. Якщо отримано урон — моб здригається або завдає контратаку.

Для чого потрібен GeckoLib?

GeckoLib потрібен у розробці модів Minecraft, коли виникає потреба зробити моделі мобів, предметів чи блоків більш живими та реалістичними. У стандартній системі Minecraft анімації дуже обмежені, і створення складної поведінки мобів або предметів вимагає написання великої кількості низькорівневого коду. GeckoLib вирішує цю проблему, дозволяючи використовувати зручний графічний редактор Blockbench для створення як моделей, так і повноцінних кісткових анімацій. Потім ці анімації легко імпортуються в гру, де вони можуть реагувати на дії гравця, зміну станів моба, бойові ситуації чи інші події.

Завдяки GeckoLib моб може, наприклад, дихати, агресивно витягати пазурі перед атакою або розкривати крила під час стрибка, а меч може світитися, пульсувати чи змінювати форму залежно від ситуації. GeckoLib не лише полегшує реалізацію такої поведінки, а й значно підвищує загальний рівень якості моду. Він дає змогу створювати анімації, що виглядають плавно, природно і не поступаються тим, які використовуються у професійних іграх. Таким чином, GeckoLib стає невід'ємним інструментом для тих, хто прагне зробити свій мод візуально привабливим, сучасним і технічно просунутим.

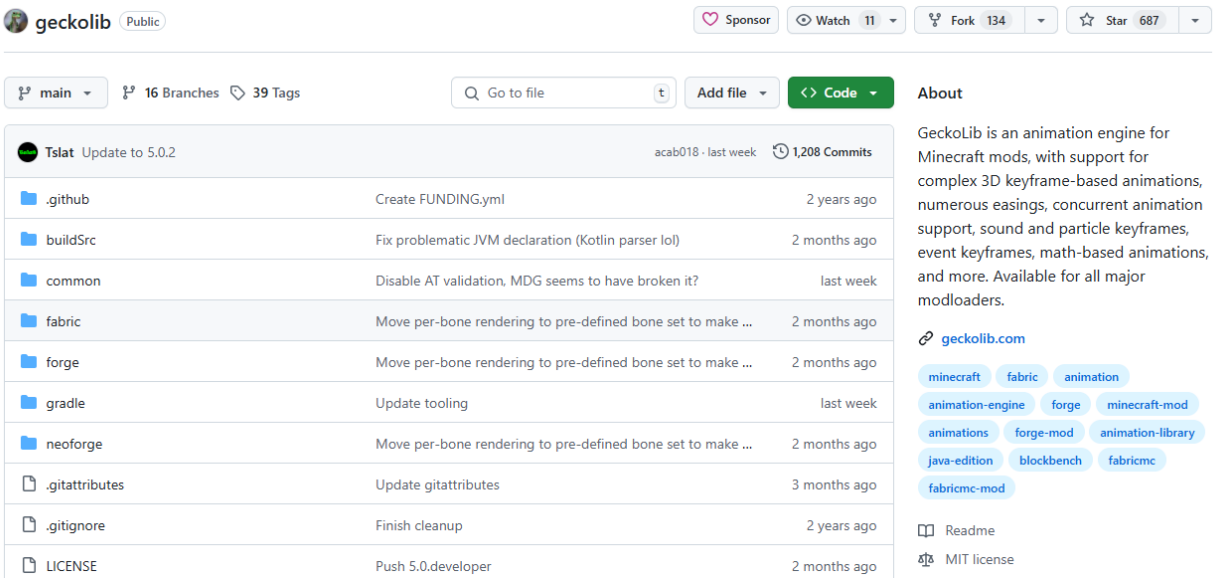

Рисунок 2.10 geckolib

MCreator – графічний конструктор модів

MCreator — це інструмент для створення модів Minecraft без необхідності програмувати вручну. Його основна мета — надати можливість створювати моди навіть тим користувачам, які не мають досвіду в програмуванні. Інтерфейс MCreator побудований таким чином, що більшість дій виконується через графічні меню, блоки умов, вибір параметрів і логіку на основі подій. Це дозволяє створювати мобів, блоки, предмети, біоми, структури, рецепти, досягнення й навіть просту поведінку або логіку — наприклад, що моб атакує тільки вночі, або що певний предмет телепортує гравця.

Важливо розуміти, що хоч MCreator і дає змогу створити власний мод з нуля, його функціональність дещо обмежена в порівнянні з тим, що можна зробити вручну за допомогою Forge або NeoForged. Проте для багатьох новачків або для тих, хто хоче швидко протестувати ідею, не заглиблюючись у код, це дуже зручне рішення. Також він може бути корисним для створення простих прототипів або для навчання: зрозуміти, як працюють події, змінні, моделі та інші компоненти Minecraft.

У більшості випадків MCreator використовують як стартовий майданчик для знайомства з моддингом. Досвідчені розробники рідше працюють з ним, оскільки їм потрібна повна свобода дій, яку можна отримати лише у «ручній» розробці через IDE (наприклад, IntelliJ IDEA або Eclipse) і написання Java-коду безпосередньо. Проте в рамках навчання, шкільних проєктів або хобі — MCreator є надзвичайно зручним і доступним інструментом, який відкриває світ модифікацій Minecraft для ширшої аудиторії.

Основні функції:

1. Створення предметів, блоків, мобів, біомів, світів:
    a. Усе створюється через "блоки логіки" або форми.
    b. Наприклад, можна створити новий меч, обрати його урон, текстуру, ефекти.
    c. Для мобів задається розмір, агресивність, модель, звуки, дроп тощо.

2. Візуальне програмування:
    a. Логіка подій реалізується блоками (подібно до Scratch).
    b. Наприклад, "при смерті моба → викликати вибух", "при ударі мечем → накласти ефект".
3. Підтримка GeckoLib:
    a. Плагіни дозволяють імпортувати моделі з Blockbench та вмикати анімації.
4. Експорт у готовий .jar файл:
    a. Один клік — і у вас вже мод, який можна встановити в Minecraft.

Переваги:

- Не потребує знання Java.
- Підходить для школярів, початківців, вчителів інформатики.
- Дуже швидкий прототипинг (створити новий меч — 3 хвилини).
- Обмеження:
- Складні системи (бойові механіки, власні інтерфейси, унікальна логіка) реалізувати важко або неможливо.
- Код, який генерується, часто не оптимізований.
- Для серйозного проєкту доведеться перейти на ручну розробку.

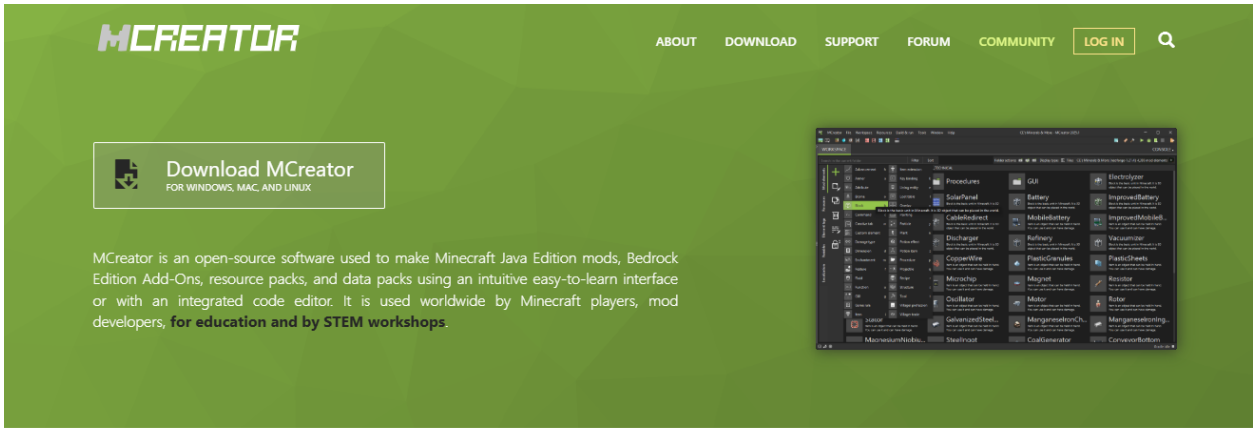

Рисунок 2.11 Головна сторінка MCreator

Покрокове створення Forge-мода на прикладі "WitcherMod".

Налаштування середовища розробки

Першим кроком у створенні Minecraft-мода на основі Forge є підготовка робочого середовища. Для цього необхідно встановити інструменти, які дозволяють писати, збирати та тестувати код мода.

1) Передусім потрібно обрати зручне середовище розробки. Найпопулярнішими є **IntelliJ IDEA** і **Eclipse** — це інтегровані середовища розробки (IDE), які підтримують мову Java, забезпечують зручну навігацію по проєкту, підсвічування синтаксису, автоматичне завершення коду, а також мають вбудовані інструменти для роботи з Gradle.

Gradle — це система збирання, яка автоматизує ключові процеси: компіляцію коду, підключення бібліотек, запуск Minecraft з модом та створення готових збірок. Вона дозволяє легко підключати додаткові залежності, зокрема **Forge** або **GeckoLib**, що особливо важливо для створення складних модів з анімаціями.

Після встановлення IDE та Gradle потрібно створити базову структуру проєкту Forge-мода, вказати назву мода у settings.gradle і прописати необхідні залежності у build.gradle. Для генерації конфігурацій запуску Minecraft з модом слід виконати відповідні Gradle-команди (genIntellijRuns для IntelliJ або genEclipseRuns для Eclipse).

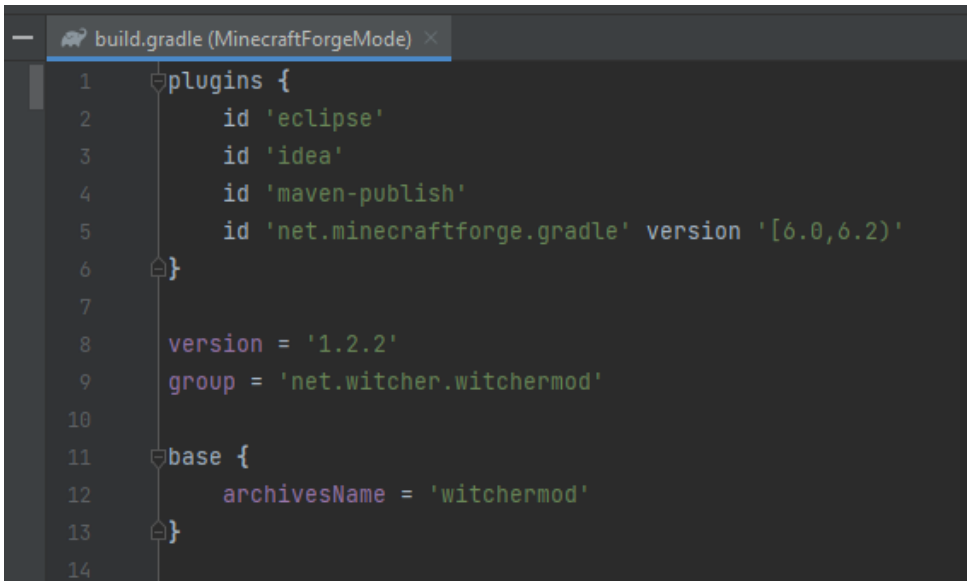

Рисунок 2.12 Build.gradle

```
# The Minecraft version must agree with the Forge ver
minecraft_version=1.19.4
# The Minecraft version range can use any release ver
# Snapshots, pre-releases, and release candidates are
# as they do not follow standard versioning conventio
minecraft_version_range=[1.19.4,1.20)
# The Forge version must agree with the Minecraft ver
forge_version=45.4.0
# The Forge version range can use any version of Forg
forge_version_range=[45,)
# The loader version range can only use the major ver
loader_version_range=[45,)
# The mapping channel to use for mappings.
```

Рисунок 2.13 Gradle.properties

2) Створення структури мода

```
@Mod(WitcherMod.MODID)
public class WitcherMod {
    public static final String MODID = "witchermod";
    public WitcherMod() {
        IEventBus bus = FMLJavaModLoadingContext.get().getModEventBus();
         // Регістрація предметів, блоків, мобів, вкладок
        ModItems.register(modEventBus);//предмети
        ModBlocks.register(modEventBus);// блоки
        ModEntities.ENTITY_TYPES.register(modEventBus);
        ModTabs.register(modEventBus); //таби - вкладки
         bus.addListener(this::setup);
        bus.addListener(this::doClientStuff);
    }
```

```
private void setup(final FMLCommonSetupEvent event) {
    // Код для налаштувань серверної частини
}
 private void doClientStuff(final FMLClientSetupEvent event) {
    // Код для клієнтської частини (рендери, GUI) // для GUI і рендер мобів
}}
}}
```

3. Реалізація функціоналу Witcher Mod: розробка моделей, текстур і механік

У цьому розділі описано процес реалізації ключових елементів мода **Witcher Mod**, зокрема створення унікальних предметів, броні, мобів, генерації руди, торгівлі з жителями, а також автоматичної генерації даних. Особливу увагу приділено відповідності контенту ігровому лору "Відьмака".

3.1 Загальний підхід до створення предмета в моді Minecraft (на прикладі Forge 1.19.2)

Створення нового предмета у моді для Minecraft складається з кількох ключових етапів, які охоплюють як **візуальну частину** (текстури, моделі), так і **програмну реалізацію** (реєстрація, властивості, локалізація). Нижче наведено покрокову схему та перелік необхідних компонентів.

Що потрібно для створення предмета:

Таблиця 3.1

№	Що створюється	Формат / Тип	Опис
1	Текстура	.png	Зображення предмета, зазвичай 16x16 або 32x32 пікселів
2	JSON-модель	.json	Вказує, як виглядає предмет

			у грі (наприклад, у руці)
3	Реєстрація предмета	Java-код	Додається до реєстраційного класу (ModItems)
4	Локалізація назви	lang/en_us.json	Додається переклад назви предмета для відображення в інтерфейсі
5	Додавання до вкладки	Java-код	Додається до креативної вкладки для зручності в грі

Основні кроки створення предмета

1. Створити текстуру предмета
    a. Намалювати або згенерувати PNG-файл
    b. Помістити його у папку: resources/assets/назва_мода/textures/item/назва_предмета.png

2. Створити модель (JSON)
    a. Файл назва_предмета.json з описом структури предмета (ручний чи інгредієнт)
    b. Зберігається в: resources/assets/назва_мода/models/item/

3. Зареєструвати предмет у коді
    a. Використовується клас типу ModItems, де реєструється об’єкт предмета через Forge API

4. Додати назву до локалізаційного файлу
    a. Наприклад, "item.назва_мода.назва_предмета": "Назва предмета"
    b. У файлі en_us.json (англійська) або uk_ua.json (українська)

5. (Опціонально) Додати предмет до креативної вкладки
    a. Щоб гравець міг знайти його в креативному режимі

Приклад структури папок:

```
src/
└── main/
    ├── java/
    │   └── com.назва_мода/
    │       └── item/
    │           └── ModItems.java
    └── resources/
        └── assets/
            └── назва_мода/
                ├── lang/
                │   └── en_us.json
                ├── textures/
                │   └── item/
                │       └── назва_предмета.png
                └── models/
```

└── item/

└── назва_предмета.json

Це — універсальний шаблон. Його можна використовувати для:

- Мечів, сокир, інструментів
- Інгредієнтів, алхімії, їжі
- Артефактів або магічних предметів

3.2 Особливості текстурування броні в Minecraft модах

Шари текстур броні: _layer_1.png та _layer_2.png

- Minecraft використовує двошарову систему текстур для броні:
  - **Layer 1 (_layer_1.png)** — текстура для голови, грудей і черевиків (шолом, нагрудник, черевики).
  - **Layer 2 (_layer_2.png)** — текстура для понож (штанів).
- Тобто, для кожного виду броні потрібно створити дві текстури:
  - назва_матеріалу_layer_1.png
  - назва_матеріалу_layer_2.png
- Вони розміщуються у: resources/assets/назва_мода/textures/models/armor/

Метод getArmorTexture

- У класі броні, що наслідує ArmorItem, потрібно перевизначити метод:

@Override public String getArmorTexture(ItemStack stack, Entity entity, EquipmentSlot slot, String type)

{ return "назва_мода:textures/models/armor/назва_матеріалу_layer_" + (slot == EquipmentSlot.LEGS ? "2" : "1") + ".png"; }

- Цей метод повертає шлях до потрібної текстури залежно від того, яка частина броні надягнута.

3.3 Створення блоків у Minecraft моді

Що потрібно для створення блока:

Таблиця 3.2

№	Що створюється	Формат / Тип	Опис
1	Текстура	.png	Зображення для зовнішнього вигляду блока
2	JSON-модель	.json	Опис форми та текстури блока (зазвичай куб)
3	Клас блока	Java-код	Визначає поведінку блока (наприклад, руйнівність, світіння)
4	Реєстрація блока	Java-код	Додавання блока до реєстру у класі ModBlocks
5	Локалізація	lang/en_us.json	Назва блока

Основні кроки:

1. Створення текстури блока
    a. Зазвичай 16x16 пікселів

b. Розміщується у: resources/assets/назва_мода/textures/block/

2. Створення JSON-моделі блока
   a. Вказує тип блока (куб, плита, колода тощо)
   b. Зберігається в: resources/assets/назва_мода/models/block/
3. Реалізація класу блока
   a. Наслідує Block або розширює стандартний функціонал
4. Реєстрація блока
   a. Реєструється у класі ModBlocks
5. Додавання локалізованої назви
   a. У мовних файлах

3.4 Додавання ефектів на броню в Minecraft моді

1) Ідея

Ефекти на броню — це спеціальні властивості, які активуються, коли гравець одягає певний комплект броні або окремий її елемент. Наприклад:

- Підвищена швидкість
- Захист від вогню
- Регенерація здоров’я
- Нічне бачення
- Збільшення стійкості до пошкоджень

2) Загальна логіка

Щоб додати ефекти, потрібно у класі броні:

- Перевизначити метод, який викликається при оновленні інвентарю гравця,
- Перевірити, чи гравець одягнув потрібний елемент або весь комплект броні,

- Якщо так — застосувати потрібний ефект через **PotionEffect** (Minecraft називає їх StatusEffect).

3) Приклад реалізації

Крок 1. Створення класу броні

```
public class CustomArmorItem extends ArmorItem {
    public CustomArmorItem(ArmorMaterial material, EquipmentSlot slot, Item.Properties properties) {
        super(material, slot, properties);
    }
    @Override
    public void onArmorTick(ItemStack stack, Level world, Player player) {
        if (!world.isClientSide()) {
            // Перевірка, що гравець одягнув цей елемент броні
            if (player.getInventory().armor.get(slot.getIndex()).getItem() == this) {
                // Додаємо ефект, наприклад, Регенерація (ID 10) на 210 тік (10.5 секунд)
                player.addEffect(new MobEffectInstance(MobEffects.REGENERATION, 210, 0, false, false, true));
            }
        } }}
```

Крок 2. Перевірка комплекта броні (опціонально)

Якщо хочете давати бонус лише при носінні повного комплекту, треба перевіряти всі слоти броні:

```
public static boolean isFullSet(Player player, Item helmet, Item chestplate, Item leggings, Item boots) {
    return player.getInventory().getArmor(3).getItem() == helmet &&
        player.getInventory().getArmor(2).getItem() == chestplate &&
        player.getInventory().getArmor(1).getItem() == leggings &&
```

player.getInventory().getArmor(0).getItem() == boots;

}

І в методі onArmorTick:

if (isFullSet(player, ModItems.CUSTOM_HELMET.get(), ModItems.CUSTOM_CHESTPLATE.get(), ModItems.CUSTOM_LEGGINGS.get(), ModItems.CUSTOM_BOOTS.get())) { player.addEffect(new MobEffectInstance(MobEffects.DAMAGE_RESISTANCE, 210, 1, false, false, true)); }

3.5 Реалізація власних Creative Tabs у Minecraft моді

1. Що таке Creative Tabs?

Creative Tabs — це вкладки у креативному інвентарі Minecraft, які допомагають групувати предмети для зручного доступу.
У модах часто створюють власні вкладки для свого контенту (зброї, броні, ресурсів тощо).

2. Приклад створення простої вкладки

public static final CreativeModeTab WITCHER_TAB = new CreativeModeTab("witcher_tab") { @Override public ItemStack makeIcon() {

// Іконка вкладки — предмет, що буде відображатися return new ItemStack(ModItems.IRON_SWORD.get()); // заміни на свій предмет } };

3. Як додати предмет у вкладку

При реєстрації предмета вказуєш вкладку, в яку його додати:

public static final RegistryObject IRON_SWORD = ITEMS.register("iron_sword", () -> new SwordItem(Tiers.IRON, 3, -2.4f, new Item.Properties().tab(WITCHER_TAB)));

//На версії 1.19.4 - таби більше не працюють, потрібно создавати ModCreativeTabContents і вручну зберігати предмети)

4. Більше вкладок — окремий клас

Зручно зробити клас для креативних вкладок:

public class ModCreativeTabs { public static final CreativeModeTab WITCHER_TAB = new CreativeModeTab("witcher_tab") { @Override public ItemStack makeIcon() { return new ItemStack(ModItems.IRON_SWORD.get()); } };

public static final CreativeModeTab WOLF_TAB = new CreativeModeTab("wolf_tab") {
  @Override
  public ItemStack makeIcon() {
    return new ItemStack(ModItems.WOLF_MEDAL.get());
  }
};

// можно додати інші

}

5. Розміщення файлів ресурсів

Назва вкладки, яку ти вказуєш у конструкторі ("witcher_tab") буде використана для локалізації.

Для неї потрібно додати переклад у файл локалізації:

resources/assets/witchermod/lang/en_us.json

{
  "itemGroup.witcher_tab": "Witcher Mod",
  "itemGroup.wolf_tab": "School of the Wolf"
}

3.6 Data Generation у Minecraft модах

1. Що таке Data Generation (Datagen)?

Data Generation — це процес автоматичного створення необхідних **JSON файлів ресурсів** для мода замість їх ручного написання.

Ці файли містять:

- Рецепти крафту
- Тексти предметів (локалізація)
- Моделі предметів і блоків
- Текстури
- Лути
- Теги (tags)
- Рендер-інформацію
- І багато іншого

2. Чому це важливо?

- JSON-файли важко і довго писати вручну, особливо якщо предметів і блоків багато.
- DataGen дозволяє пишеш код — і він генерує ці файли автоматично.
- Допомагає уникнути помилок у синтаксисі JSON.
- Спрощує підтримку й масштабування мода.

3) Як працює Data Generation у Forge?

1. Створюються класи-генератори:

- Наприклад, клас для генерації рецептів (RecipeProvider),
- Клас для генерації локалізації (LanguageProvider),
- Клас для генерації моделей (ItemModelProvider),

- Клас для генерації тегів (TagProvider),
- Клас для генерації лута в сундуках (LootTableProvider).

2. Як це запускається?

- В моді створюють спеціальний клас, який наслідує від DataGenerator (або просто реєструють провайдери генерації).
- Під час запуску спеціальної команди (gradlew genData або через IDE) **генератор створює файли у папці generated/resources**.
- Потім ці файли копіюються у відповідну папку ресурсу, щоб використовуватися в грі.

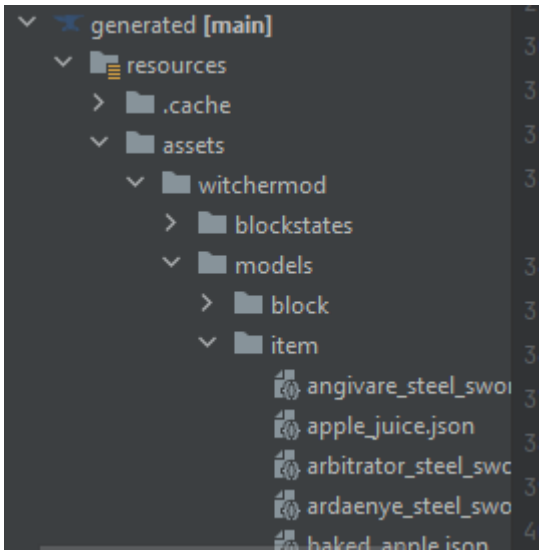

Рисунок 3.1 Generated/resources\

3. Приклад коду генерації рецептів (спрощено)

public class ModRecipeProvider extends RecipeProvider { public ModRecipeProvider(PackOutput output) { super(output); }

@Override

protected void buildRecipes(Consumer<FinishedRecipe> consumer) { ShapedRecipeBuilder.shaped(ModItems.WITCHER_SWORD.get())

```
        .pattern("  I  ")
        .pattern("  I  ")
        .pattern("  S  ")
        .define('I', Items.IRON_INGOT)
        .define('S', Items.STICK)
        .unlockedBy("has_iron", has(Items.IRON_INGOT))
        .save(consumer);
}
```

}

Які основні види DataGen можна реалізувати?

Таблиця 3.3

DataGen тип	Що генерує
RecipeProvider	Рецепти крафту (json)
ItemModelProvider	Моделі предметів (json)
BlockStateProvider	Моделі і стани блоків (json)
LanguageProvider	Файли локалізації (json)
LootTableProvider	Лути мобів, сундуків, блоків (json)
TagProvider	Теги (json) — групування предметів і блоків
AdvancementProvider	Досягнення (json)

Переваги DataGen

- Швидко змінювати і додавати рецепти та інші ресурси в коді.
- Не потрібно вручну робити сотні JSON-файлів.
- Автоматична перевірка структури і валідності.
- Легко інтегрувати з системою збірки Gradle.

3.7 Генерацію глобальних модифікаторів луту ModGlobalLootModifiersProvider

- Це клас, який відповідає за **генерацію глобальних модифікаторів луту** (loot modifiers).
- Глобальні модифікатори луту дозволяють змінювати або додавати предмети у всі таблиці луту, що відповідають певним умовам.
- Наприклад, додати додаткові предмети до будь-якого моба, сундука або блоку без зміни кожної окремої loot table.

Як це працює?

1. Клас наслідує GlobalLootModifierProvider (у Forge).
2. У методі start() ти реєструєш різні модифікатори луту (global loot modifiers).
3. Для кожного модифікатора ти визначаєш логіку, як саме змінювати лут (через спеціальні JSON-файли).
4. Ці JSON-файли потім автоматично генеруються під час Data Generation.

Для чого потрібні глобальні модифікатори?

- Додати предмети до луту всіх (або вибраних) мобів, наприклад, щоб у всіх зомбі шанс випадіння була броня з Witcher Mod.
- Додати новий лут у сундуки різних структур.
- Змінити лут у певних умовах (наприклад, при використанні чарів).

Приклад реалізації ModGlobalLootModifiersProvider

```
public class ModGlobalLootModifiersProvider extends GlobalLootModifierProvider {
public ModGlobalLootModifiersProvider(PackOutput output) {
  super(output, WitcherMod.MODID);
}
```

```
@Override
protected void start() {
  // Реєстрація глобального модифікатора луту
  add("add_witcher_armor_to_zombies",
    ModLootModifiers.ADD_WITCHER_ARMOR.get(), // посилання на модифікатор
    new AddItemModifier(new LootItemCondition[] {
      // тут можна вказати умови, наприклад, лут тільки у зомбі
    }, new ItemStack(ModItems.WITCHER_HELMET.get()))
  );
}
```

}

Як це інтегрується з DataGen?

- При запуску генерації даних (gradlew genData) цей провайдер створить JSON файл у папці data/witchermod/loot_modifiers.
- JSON описує, які предмети і за яких умов додаються або змінюються у луті.
- При запуску гри цей модифікатор застосовується автоматично до всіх відповідних лут-таблиць.

3.8 Вампіри в Minecraft моді з GeckoLib

Загальний метод створення вампіра в Minecraft-моді з GeckoLib

1. Створити модель, текстуру й анімацію в Blockbench

- Відкрити **Blockbench**, вибрати шаблон GeckoLib (GeckoLib Animated Java Block/Entity).
- Створити 3D-модель вампіра.
- Додати базові анімації:

- idle
- walk
- attack (опційно)

- Експортувати у GeckoLib-формат:
  - .geo.json — геометрія
  - .animation.json — анімації
  - .png — текстура

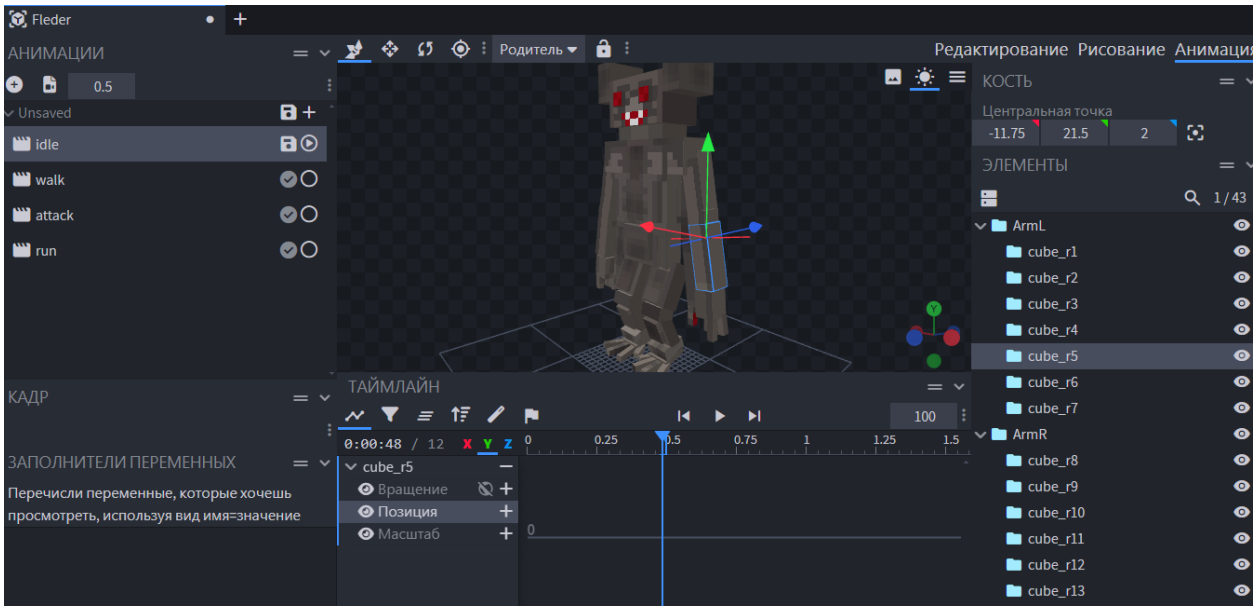

Рисунок 3.2 Анімації вампіра

2. Створити клас сутності (Entity)

- Клас повинен наслідувати Monster (чи інший Mob клас).
- Імплементує IAnimatable (GeckoLib).
- Додається AnimationFactory, а також методи registerControllers() і getFactory().

3. Налаштувати штучний інтелект (AI)

- У методі registerGoals() прописати цілі (Goals):

- атака ближнього бою
- пересування
- цілі для атак (гравець, інші моби)

4. Створити модель-клас (GeoModel)

- Наслідувати AnimatedGeoModel<VampireEntity>.
- Указати шляхи до .geo.json, .animation.json, .png.

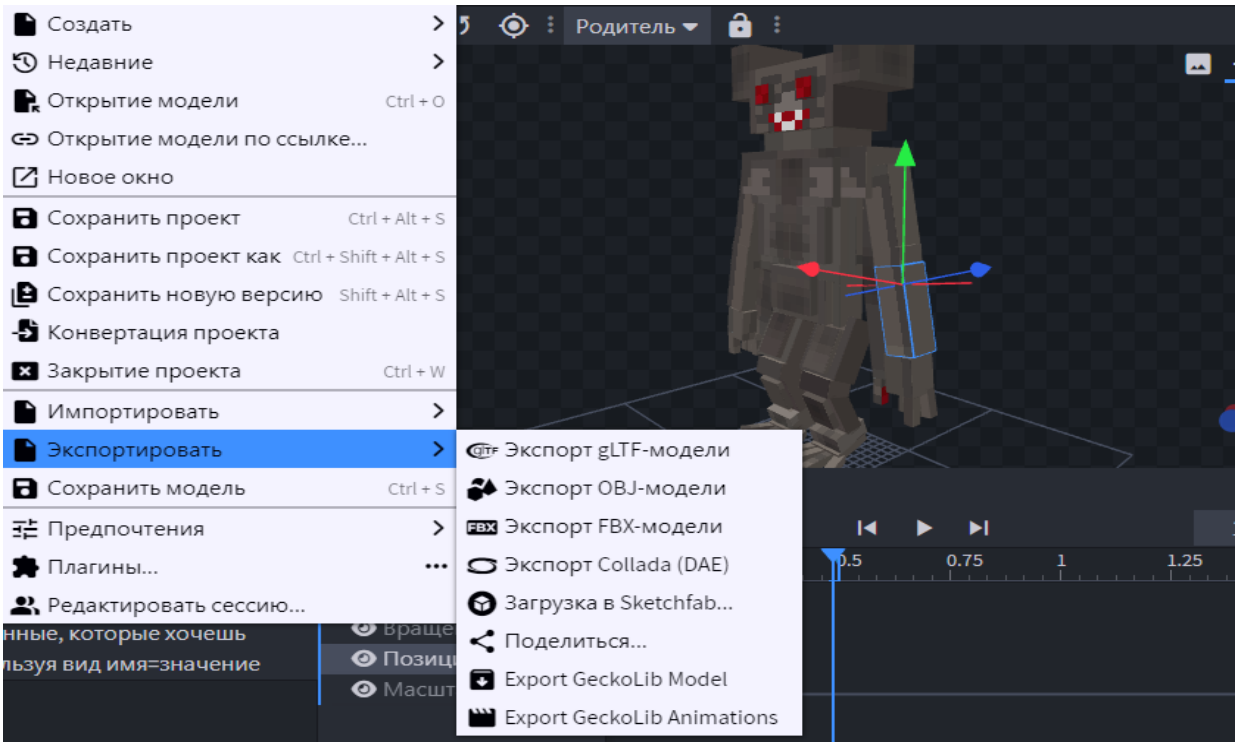

Рисунок 3.3 Експорт Моделі і Анімації

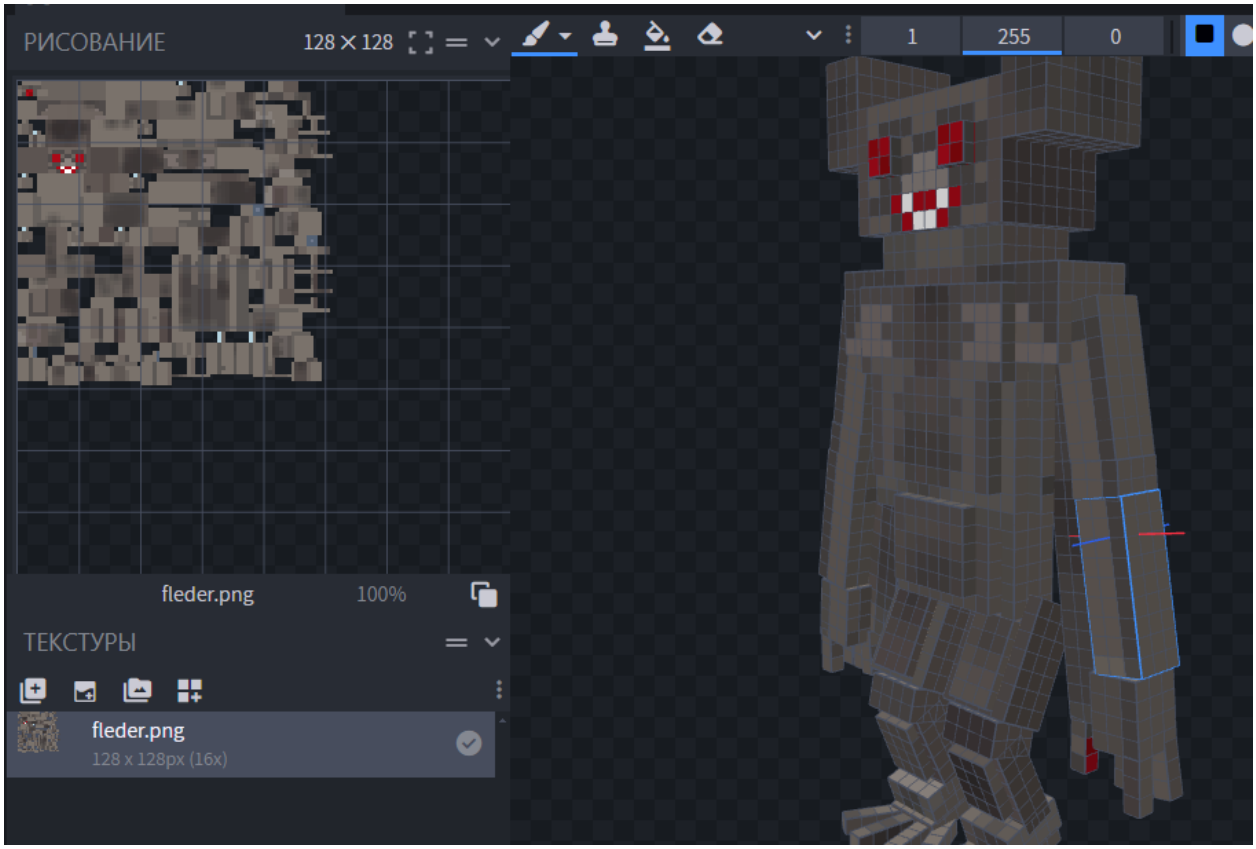

Рисунок 3.4 Fleder.png

5. Створити рендерер

- Наслідувати GeoEntityRenderer<VampireEntity>.
- Указати модель і текстуру.
- Налаштувати розмір тіні, освітлення тощо.

6. Зареєструвати сутність

- Використати DeferredRegister<EntityType<?>>.
- Прописати тип, розміри, категорію (MobCategory.MONSTER).
- У ClientModEvent зареєструвати рендерер через event.registerEntityRenderer().

7. Додати локалізацію

- У en_us.json

8. (Опціонально) Додати спавнер або природну генерацію

- Створити яйце призову (SpawnEggItem) або структури спавну.
- Додати до біомів чи Dungeon loot через BiomeModifications чи LootModifier.

```
resources/
└── assets/witchermod/
    ├── models/entity/vampire.geo.json
    ├── animations/vampire.animation.json
    ├── textures/entity/vampire.png
    └── lang/en_us.json
```

Що потрібно для створення вампіра:

Таблиця 3.4

Компонент	Що робить
Blockbench модель	Візуальна 3D-модель
.geo.json	Геометрія (модель)моба
.animation.json	Анімації поведінки
.png	Текстура моба
VampireEntity	Клас моба
VampireModel	Модель з GeckoLib
VampireRenderer	Рендерер моба
EntityType<VampireEntity>	Реєстрація моба
registerGoals()	Штучний інтелект
registerRenderers()	Відображення в грі

3.9 Кастомна рослина

Як працює?

Наслідування від CropBlock: Створюється новий клас, який розширює CropBlock — базовий клас для культур у Minecraft (як пшениця).

Прив’язка насіння: У методі getBaseSeedId() вказується предмет, який буде саджати цю культуру (наприклад, pepper_seeds).

Реєстрація:

- Блок (pepper_crop) і предмет насіння (pepper_seeds) реєструються в системі реєстрації модифікацій.
- Насіння — це предмет типу ItemNameBlockItem, який при використанні саджає відповідний блок.

Етапи росту:

- Рослина проходить кілька стадій росту (зазвичай від 0 до 7).
- Для кожного етапу створюється окрема модель (pepper_stage0 ... pepper_stage7) і текстура.
- У blockstates створюється JSON, який прив’язує стадії росту до моделей.
- Ріст і добрива:
- Ріст відбувається природньо або за допомогою кісткового борошна.
- Поведінка автоматично успадковується від CropBlock.
- Збір врожаю:
- Коли рослина досягає максимальної стадії (наприклад, age=7), її можна зібрати.
- Після збору вона може дати плід (перець) і нове насіння.
- Необхідні компоненти для реалізації культури:

Таблиця 3.5

Компонент	Приклад
Клас блоку	PepperCropBlock.java
Насіння	pepper_seeds (ItemNameBlockItem)
Текстури	pepper_stage0.png ... pepper_stage7.png
Моделі	block/pepper_stage0.json ... block/pepper_stage7.json
Blockstates	blockstates/pepper_crop.json
Loot table	Видача насіння/плодів при зборі

Розділ 4. Тестування та оцінка ефективності модифікації

4.1 Методика тестування

Для перевірки працездатності Witcher Mod використовувалося тестування у внутрішньоігрових умовах Minecraft версії 1.19.4 з установленим Forge. Основні етапи тестування:

- **Функціональне тестування**: перевірка базової роботи предметів, блоків, мобів, броні, ефектів.
- **Тестування генерації**: перевірка появи руди, лута в сундуках, торгівлі з жителями.
- **Тестування взаємодій**: бойові механіки, ефекти від броні, взаємодія з іншими модами (за потреби).
- **Візуальне тестування**: перевірка коректного відображення текстур, моделей (особливо створених у BlockBench або з Geckolib).
- **Консольне тестування**: виявлення винятків або помилок через лог-файли (latest.log і debug.log).

4.2 Виявлені помилки та способи усунення

Під час тестування було виявлено кілька типових проблем:

Тип помилки	Причина	Спосіб вирішення
Невидима текстура	Відсутній або неправильно вказаний шлях до texture_location	Перевірка ресурсного шляху в model.json

Броня не відображається	Не створено layer_1.png / layer_2.png або getArmorTexture повертає некоректний шлях	Додавання текстур та правильний override методу
Моб не анімується	Неправильна прив'язка анімацій у Geckolib-моделі	Перевірка JSON-файлів та класів анімацій
Лут не генерується	Неправильна конфігурація GlobalLootModifier або LootTable	Виправлення JSON-файлів або провайдерів у DataGen
Зброя не наносить ефекти	Не реалізовано hurtEnemy() або не перевірено LivingHurtEvent	Додана подія або метод атаки в Item-класі

Баланс

- Зброя модифікації була протестована на різних типах мобів (включно з ванільними та кастомними).
- Броня зі спецефектами не дає перевагу, яка повністю порушує баланс — ефекти обмежені або працюють лише при виконанні певних умов.
- Дроп і рідкість ресурсів були налаштовані згідно з тематикою "Відьмака" (наприклад, вампірські матеріали рідкісні й падають лише з певних мобів).
- Зручність
- Всі предмети згруповані у вкладки Creative Mode (Witcher, Wolf, Cat, тощо).
- Назви предметів, текстури та логіка відповідають лору оригінального всесвіту.
- Модифікація підтримує плавну інтеграцію з іншими модами, не порушуючи їх механіки.

4.3 Потенціал подальшого розвитку Witcher Mod

Розроблений мод на даному етапі містить основні елементи світу «Відьмака», зокрема зброю, броню, мобів, руди, торгівлю та лут. Проте потенціал модифікації дозволяє розширювати її значно глибше як у контентному, так і в механічному аспектах. Нижче наведено можливі напрями розвитку:

1. Система алхімії та еліксирів

У всесвіті «Відьмака» алхімія відіграє ключову роль. Її впровадження в мод додасть глибину та стратегію в бою:

- **Крафт еліксирів** зі спеціальними інгредієнтами, які випадають з унікальних істот.
- **Систему токсичності**, яка обмежує одночасне використання кількох еліксирів.
- **Ефекти еліксирів**: нічне бачення, швидкість, сила, регенерація, збільшений урон по монстрам, тощо.

2. Реалізація знаків (магічних умінь)

У грі «Відьмак» гравець використовує п’ять базових магічних знаків:

- **Аард** – відкидає ворогів.
- **Ігні** – підпалює.
- **Квен** – створює захисний щит.
- **Аксій** – контролює розум ворога.
- **Ірден** – створює пастку, що уповільнює ворогів.

Їх реалізація можлива через систему активованих здібностей або за допомогою спеціального UI.

3. Повноцінна система квестів

- Впровадження **NPC-квестодавців** з діалоговими вікнами.
- Можливість обирати **етичні варіанти** відповіді (добрий/нейтральний/жорстокий вибір).

- Завдання на полювання за монстрами, збирання інгредієнтів, дослідження територій.

4. Дерево вмінь (Skill Tree)

- Впровадження **прокачки персонажа** за аналогією з RPG:
  - шкала досвіду окремо від основного XP;
  - відкриття нових ефектів, покращення знаків, зменшення токсичності.
- Створення **UI-інтерфейсу** для прокачки.

5. Розширення категорій монстрів

- Впровадження **нових видів вампірів**, чудовиськ, привидів, духів лісу тощо.
- Створення **поведінкових шаблонів** AI через Geckolib або Custom Goals.
- Додавання **механіки бою з босами**: унікальні арени, поведінка, фази.

6. Удосконалення економіки та торгівлі

- Додавання торговців-алхіміків, чорних ковальських цехів, мисливців за головами.
- Впровадження **валюти** (орени, флорини, крони).
- Можливість продажу трофеїв, купівлі рецептів, замовлень на мобів.

7. Додаткові структури та локації

- Генерація руїн, таборів відьмаків, підземель, лісових хатин, тощо.
- Додавання данжів із пастками, босами й лутом.
- Інтеграція із зовнішніми біомами через BiomeModifiers.

8. Технічні покращення

- Повна підтримка **датагенерації** всього контенту.
- Перехід на архітектуру модів для Fabric або NeoForge (опціонально).
- **Міжнародна локалізація** (англ., укр., пол., рос. тощо).
- Підтримка **модифікацій на сервері** (оптимізація через Dist.CLIENT/SERVER)

Висновок

Висновок до Розділу 1. Тематика та ігровий світ "Відьмака" у контексті Minecraft

У цьому розділі було проаналізовано літературну та ігрову спадщину серії "Відьмак", що дозволило виокремити ключові елементи для адаптації у форматі Minecraft. Було визначено, що світ "Відьмака" має багатий лор, впізнавану атмосферу, унікальну систему монстрів, зброї, алхімії та знаків, що ідеально підходить для реалізації в модифікації. Розгляд схожих проєктів дав змогу оцінити сучасний стан тематичних модів у геймінговій культурі та виявити актуальні підходи до адаптації відомих всесвітів. В результаті, було сформульовано основні концепти, які стали базисом для реалізації власного моду.

Висновок до Розділу 2. Технології та особливості розробки модів для Minecraft

У другому розділі було розглянуто програмно-технологічну складову розробки модифікацій для Minecraft. Здійснено налаштування середовища розробки з використанням Forge для версії 1.19.4, описано структуру проєкту, основні пакети, принципи реєстрації елементів гри. Було охоплено інструменти, що застосовуються в процесі розробки — серед яких Java, JSON, Data Generation, Blockbench, Geckolib. Це дало змогу забезпечити технічну основу для створення функціонального, масштабованого та добре організованого моду.

Висновок до Розділу 3. Реалізація функціоналу Witcher Mod: контент, механіки, інтеграція

У цьому розділі було реалізовано ключовий ігровий функціонал моду Witcher Mod. Створено нові предмети, зброю, броню з унікальними ефектами, блоки, мобів-вампірів із анімаціями, нові кулінарні рецепти, механіку торгівлі, генерацію ресурсів у світі, а також інтегровано систему лута. Значна увага приділена адаптації контенту відповідно до лору "Відьмака". Завдяки використанню сучасних технологій, таких як Geckolib і DataGenerators, розробка пройшла ефективно й структуровано. Також реалізовано

категоризацію контенту за відьмацькими школами з використанням вкладок (Creative Tabs), що підвищило зручність використання.

Висновок до Розділу 4. Тестування та оцінка ефективності модифікації

Завершальний розділ було присвячено тестуванню моду у реальних умовах гри. Проведено перевірку коректності генерації структур, функціонування механік, боїв з мобами, ефектів предметів. Було виявлено та виправлено ряд помилок, покращено продуктивність і логіку взаємодії. Оцінено баланс між ігровими елементами, а також зручність користування модом. Встановлено, що мод може стати основою для розширення: у майбутньому можлива реалізація алхімії, знаків, квестової системи, додаткових шкіл та відкритого світу з унікальними структурами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сапковський А. Останнє бажання / Анджей Сапковський ; пер. з пол. С. Легеза. — Харків : Клуб Сімейного Дозвілля, 2016. — 320 с.
2. CD Projekt RED. The Witcher Official Website [Електронний ресурс]. – Режим доступу: https://thewitcher.com/ – Назва з екрана.
3. Minecraft Wiki. Офіційна вікі гри Minecraft [Електронний ресурс]. – Режим доступу: https://minecraft.fandom.com/ – Назва з екрана.
4. Forge Documentation. Minecraft Forge Docs for 1.19.x [Електронний ресурс]. – Режим доступу: https://docs.minecraftforge.net/ – Назва з екрана.
5. Geckolib Library. Офіційна документація GeckoLib [Електронний ресурс]. – Режим доступу: https://geckolib.com/ – Назва з екрана.
6. Blockbench. Онлайн-редактор моделей для Minecraft [Електронний ресурс]. – Режим доступу: https://blockbench.net/ – Назва з екрана.
7. Kaupenjoe. Minecraft Modding Tutorials (YouTube-канал) [Електронний ресурс]. – Режим доступу: https://www.youtube.com/c/Kaupenjoe – Назва з екрана.
8. TheGreyGhost. Minecraft Modding Tutorials Blog [Електронний ресурс]. – Режим доступу: https://greyminecraftcoder.blogspot.com/ – Назва з екрана.
9. Mojang Studios. Minecraft Java Edition. Офіційна версія гри [Електронний ресурс]. – Режим доступу: https://minecraft.net/ – Назва з екрана.
10. GitHub. Приклади Forge-модів для версії 1.19.x [Електронний ресурс]. – Режим доступу: https://github.com/ – Назва з екрана.
11. Minecraft Forge Docs. Data Generation у Forge [Електронний ресурс]. – Режим доступу: https://mcforge.readthedocs.io/en/1.19.x/datagen/ – Назва з екрана.
12. JetBrains. IntelliJ IDEA – середовище розробки [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/idea/ – Назва з екрана.
13. Бойко М. С. Метод генерації ігрового світу та структур для гри в жанрі RPG : кваліфікаційна робота магістра / М. С. Бойко ;

14. Nováková, P. *The Witcher: A Transmedia Saga* : Master's thesis / Pavla Nováková ; Charles University, Faculty of Arts. – Prague, 2022. – 93 p. – Available at: https://dspace.cuni.cz/bitstream/handle/20.500.11956/171653/120411674.pdf?isAllowed=y&sequence=1.
15. Ставцев Д. І. Гра у жанрі платформер : дипломний проєкт на здобуття ступеня бакалавра з напряму підготовки 123 «Комп'ютерна інженерія» / Д. І. Ставцев ; НТУУ «КПІ ім. Ігоря Сікорського», Факультет інформатики та обчислювальної техніки, Кафедра обчислювальної техніки. – Київ, 2020.
16. Ковалик А. С. Мобільний додаток створення та ведення довідника методичних матеріалів навчальних курсів кафедри : дипломна робота на здобуття ступеня бакалавра за освітньо-професійною програмою «Програмне забезпечення розподілених систем» / А. С. Ковалик ; НТУУ «КПІ ім. Ігоря Сікорського», Теплоенергетичний факультет, Кафедра автоматизації проектування енергетичних процесів і систем. – Київ, 2021.