ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІСЬКОГО ГОСПОДАРСТВА ІМЕНІ О. М. БЕКЕТОВА
Навчально-науковий інститут енергетичної, інформаційної
та транспортної інфраструктури

Кафедра комп'ютерних наук та інформаційних технологій

Пояснювальна записка
до кваліфікаційної роботи бакалавра

на тему: « Розробка модифікації для гри Minecraft мовою Java з використанням API Forge»

Виконав: студент 4 курсу, групи КН 2021-1
спеціальності
122 Комп'ютерні науки
(шифр і назва спеціальності)

Марко МАКАРЕНКО
(ім'я та прізвище)

Керівник: Андрій ЄВДОКІМОВ
(ім'я та прізвище)

Рецензент Володимир БРЕДІХІН
(ім'я та прізвище)

м. Харків – 2025 рік

Харківський національний університет міського господарства імені О. М. Бекетова
(повне найменування закладу вищої освіти)
Інститут Навчально-науковий Інститут енергетичної, інформаційної та транспортної інфраструктури
Кафедра комп'ютерних наук та інформаційних технологій
Рівень вищої освіти перший (бакалаврський)
Спеціальність 122 Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КНтаІТ

Новожилова М.В.
«_27__» __06____2025року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Макаренко Марку Борисовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка модифікації для гри Minecraft мовою Java з використанням API Forge

керівник роботи Євдокімов Андрій Анатолійович, доцент, канд. техн. наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «09» травня 2025 р. №341-03

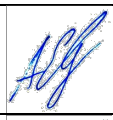
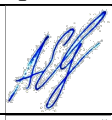
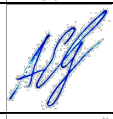
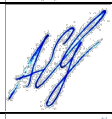
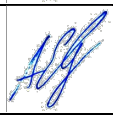
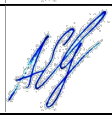
2. Термін подання студентом роботи 26 червня 2025 року

3. Вихідні дані до роботи: наукові та методичні джерела, цифрові матеріали з мережі інтернет, тези й доповіді з наукових конференцій, статті з професійних періодичних видань, а також спеціалізовані ресурси, що містять інформацію про створення модифікацій.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): проаналізувати можливості API Forge для створення модифікацій; розробити кастомний меч із ефектами вогню та самурайською анімацією; створити модель і анімацію меча в Blockbench; реалізувати конфігураційний файл для налаштування ефектів; забезпечити сумісність модифікації з Minecraft 1.20.1; провести тестування та оптимізацію; виконати завдання з охорона праці.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): презентаційний матеріал у кількості 20 слайдів

6. Консультанти розділів роботи

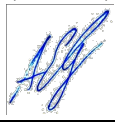
Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ 1	Андрій ЄВДОКІМОВ, к. т. н., доцент кафедри КН та ІТ		
Розділ 2	Андрій ЄВДОКІМОВ, к. т. н., доцент кафедри КН та ІТ		
Розділ 3	Андрій ЄВДОКІМОВ, к. т. н., доцент кафедри КН та ІТ		
Розділ 4	Вікторія Малишева, к. т. н., доцент кафедри безпеки життєдіяльності		

7. Дата видачі завдання 12 травня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Вибір теми дипломної роботи	09.05.2025	
2	Затвердження тем, наукових керівників, завдань та календарного плану підготовки дипломної роботи	09.05.2025	
3	Написання І розділу	11.05.2025	
4	Написання ІІ розділу	14.05.2025	
5	Написання ІІІ розділу	26.05.2025	
6	Написання IV розділу	26.05.2025	
7	Подання дипломної роботи керівнику	13.06.2025	
8	Робота по усуненню зауважень керівника, уточнення і доповнення практичного матеріалу, оформлення додатків до роботи	15.06.2025	
9	Подання доопрацьованого варіанту роботи керівнику	19.06.2025	
10	Захист матеріалів дипломної роботи на засіданні кафедри	23.06.2025	
11	Офіційний захист матеріалів дипломної роботи на засіданні Державної екзаменаційної комісії	28.06.2025	

Студент  (підпис) Марко МАКАРЕНКО (прізвище та ініціали)

Керівник роботи  (підпис) Андрій ЄВДОКІМОВ (прізвище та ініціали)

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка кваліфікаційної роботи бакалавра студента групи КН 2021-1 спеціальності 122 Комп'ютерні науки Макаренко Марка Борисовича та темою «Розробка модифікації для гри Minecraft мовою Java з використанням API Forge» складається з 4 розділів, містить 65 сторінок, 13 рисунків, 3 таблиці, 15 джерел.

Дипломна робота на тему «Розробка модифікації для гри Minecraft мовою Java з використанням API Forge» присвячена створенню модифікації MyMod, яка розширює функціональність гри Minecraft версії 1.20.1 шляхом додавання кастомного меча Mystic Blade.

У першому розділі розглядається поточний стан розвитку Minecraft та обмеження базової версії гри, зокрема нестача унікальних бойових предметів, обмежені візуальні ефекти та статична анімація рухів. Розробка Mystic Blade відповідає цим потребам, поєднуючи технічну реалізацію на основі Forge з креативним дизайном і кастомізацією через конфігураційні файли.

Другий розділ присвячено практичному етапу створення модифікації Mystic Blade. У ньому детально розглядається процес розробки ігрового предмета – кастомного меча з унікальними візуальними та функціональними характеристиками. Реалізація включає створення тривимірної моделі та анімації за допомогою Blockbench, програмування бойової логіки й ефекту вогню в середовищі Forge API, а також налаштування параметрів через зовнішній конфігураційний файл у форматі TOML. Окрема увага приділяється інтеграції меча у базову механіку Minecraft без порушення балансу гри.

У розділі охорони праці було проведено дослідження ризику реалізації потенційних небезпек на об'єкті проектування та розробка заходів щодо їх попередження.

Ключові слова: MINECRAFT, FORGE API, МОДИФІКАЦІЯ, JAVA, АНІМАЦІЯ, КОНФІГУРАЦІЯ, BLOCKBENCH, ГЕЙМПЛЕЙ.

ABSTRACT

Structure and scope of work. Explanatory note of the qualification work of the bachelor's student of the group KN 2021-1, specialty 122 Computer Science Mark Makarenko and the topic "Development of a modification for the Minecraft game in Java using the Forge API" consists of 4 sections, contains 65 pages, 13 figures, 3 tables, 15 sources.

The thesis on the topic "Development of a modification for the Minecraft game in Java using the Forge API" is dedicated to the creation of the MyMod modification, which extends the functionality of the Minecraft game version 1.20.1 by adding a custom sword Mystic Blade.

The first section examines the current state of development of Minecraft and the limitations of the basic version of the game, in particular the lack of unique combat items, limited visual effects and static animation of movements. The development of Mystic Blade meets these needs, combining technical implementation based on Forge with creative design and customization through configuration files.

The second section is devoted to the practical stage of creating the Mystic Blade modification. It examines in detail the process of developing a game item - a custom sword with unique visual and functional characteristics. The implementation includes creating a three-dimensional model and animation using Blockbench, programming combat logic and fire effects in the Forge API environment, as well as setting parameters via an external configuration file in TOML format. Special attention is paid to integrating the sword into the basic mechanics of Minecraft without disrupting the game balance.

In the occupational health and safety section, a study of the risk of potential hazards at the design site was conducted and measures were developed to prevent them.

Keywords: MINECRAFT, FORGE API, MODIFICATION, JAVA, ANIMATION, CONFIGURATION, BLOCKBENCH, GAMEPLAY.

ЗМІСТ

ВСТУП

Мета роботи є розробика модифікації для Minecraft, яка інтегрує кастомний меч із інноваційними механіками, використовуючи API Forge, і забезпечує гнучке налаштування для гравців. Мета включає створення моделі меча в Blockbench, реалізацію ефектів і анімації, а також тестування в однокористувацькому та мультиплеєрному режимах.

Об'єкт розробки: Об'єктом розробки є модифікація для Minecraft, яка спрямована на розширення всесвіту гри шляхом введення нових вимірів, біомів, ресурсів і викликів.

Завдання роботи:

- Проаналізувати можливості API Forge для створення модифікацій.
- Розробити кастомний меч із ефектами вогню та самурайською анімацією.
- Створити модель і анімацію меча в Blockbench.
- Реалізувати конфігураційний файл для налаштування ефектів.
- Забезпечити сумісність модифікації з Minecraft 1.20.1.
- Провести тестування та оптимізацію.

Етап дизайну охоплює розробку оригінального ігрового контенту з урахуванням візуального стилю Minecraft. У рамках модифікації Mystic Blade реалізовано унікальну бойову механіку, яка візуально і тематично вирізняється з-поміж стандартної зброї гри. Кастомна анімація, візуальні ефекти та механіка активації меча створюють новий тип взаємодії з ігровим середовищем.

Окрім цього, мод може бути розширений додатковими компонентами: спеціальними ресурсами, новими матеріалами для крафту або унікальними зонами, які можна поєднувати з мечем у рамках сценаріїв. Для таких компонентів розробляються алгоритми генерації, що відповідають тематиці контенту (наприклад, містичний ліс або арена випробувань), а також додаються унікальні

будівельні блоки, інструменти й обладнання. Це стимулює гравців не лише використовувати меч, а й досліджувати новий ігровий простір у повній мірі.

Тестування та оптимізація. Для забезпечення стабільної та комфортної роботи модифікації проведено комплексне тестування в різних сценаріях гри – як в одиночному режимі, так і в мультиплеєрі. Перевірялися конфлікти з іншими модами, стабільність бойової механіки, коректність рендерингу анімації та робота конфігураційного файлу. Особлива увага приділялась оптимізації ресурсів (texture memory, event handling), щоб мод працював плавно навіть на системах середнього рівня. Модифікація протестована в однокористувацькому та мультиплеєрному режимах, що гарантує її стабільність і сумісність з Minecraft версії 1.20.1.

РОЗДІЛ 1 АНАЛІЗ СТАНУ ОБЛАСТІ РІШЕННЯ ЗАДАЧІ

Minecraft – це гра, яка змінила уявлення про сучасні відеоігри, ставши однією з найвпливовіших платформ для творчості, дослідження та соціальної взаємодії. З моменту свого релізу в 2011 році, створена Маркусом Перссоном і розвинена Mojang Studios, вона привернула увагу мільйонів гравців по всьому світу. Її унікальність полягає у відкритому світі, де гравці можуть будувати складні споруди, досліджувати різноманітні біоми, боротися з ворогами та створювати власні історії. Однак ключовим фактором успіху Minecraft є її активна спільнота, яка через модифікації постійно розширює межі базової версії гри, відомої як «Vanilla» Minecraft.

Модифікації, або моди, є важливим елементом екосистеми Minecraft, дозволяючи гравцям і розробникам додавати нові предмети, механіки, біоми, моби та навіть цілі виміри [1]. Вони не лише усувають обмеження базової гри, але й створюють нові можливості для персоналізації ігрового досвіду. Одним із популярних напрямів модифікацій є вдосконалення бойової системи, яка в «Vanilla» Minecraft є відносно простою. Зокрема, додавання нових видів зброї, таких як мечі, дозволяє урізноманітнити геймплей, зробити бої більш динамічними та залучити гравців, які шукають нові виклики.

У цьому контексті розробка модифікації для додавання нового меча за допомогою API Forge є актуальним завданням. API Forge – це один із найпоширеніших фреймворків для створення модифікацій, який забезпечує гнучкість, сумісність із іншими модами та доступ до внутрішніх механізмів гри. Такий меч може не лише збагатити бойову систему, але й вирішити низку проблем, пов’язаних із одноманітністю геймплею, обмеженістю ресурсів і залученням гравців. Метою цього розділу є аналіз сучасного стану ігрового середовища Minecraft, визначення ключових проблем і можливостей, а також обґрунтування концепції модифікації, що додає новий меч.

Спільнота Minecraft постійно розвивається, створюючи нові ідеї та

впроваджуючи інновації. Цей процес вимагає глибокого аналізу, щоб зрозуміти, як модифікації можуть відповідати потребам гравців. Незмінна популярність гри охоплює покоління, залучаючи як молодих гравців, так і досвідчених ентузіастів. Проте, коли гравці заглиблюються в механіки гри, вони стикаються з обмеженнями, які спонукають спільноту до створення модифікацій. Розробка нового меча є частиною цього процесу, спрямованого на розширення потенціалу Minecraft і покращення ігрового досвіду.

Аналіз сучасного стану модифікацій дозволяє визначити ключові сфери, де гра може бути вдосконалена. Зокрема, бойова система, хоча й функціональна, не завжди відповідає очікуванням гравців, які прагнуть більшої різноманітності та глибини. Додавання нового меча через API Forge відкриває можливості для інновацій, таких як унікальні механіки, інтеграція з іншими системами гри та залучення нових гравців. У цьому розділі ми розглянемо, як модифікація може вирішити ці завдання, враховуючи технічні, соціальні та геймплейні аспекти.

1.1 Різноманітність у «Vanilla» Minecraft

Базова версія Minecraft, відома як «Vanilla», пропонує гравцям широкий спектр можливостей, але з часом багато хто стикається з відчуттям одноманітності через обмежену різноманітність у ключових аспектах гри. Ці обмеження особливо помітні в бойовій системі, яка є центральною частиною геймплею для багатьох гравців. Розробка модифікації для додавання нового меча спрямована на усунення цих проблем, додаючи нові елементи, що урізноманітнюють ігровий процес.

Біоми та структури. У «Vanilla» Minecraft біоми відіграють ключову роль у створенні унікального ігрового середовища. Вони визначають ландшафт, рослинність, клімат і типи мобів, які з’являються в певних регіонах. Від теплих тропічних джунглів до холодних тундр, біоми створюють відчуття різноманітності та заохочують до дослідження. Проте, незважаючи на наявність десятків біомів, їхній асортимент може здаватися обмеженим для гравців, які проводять у грі сотні

годин. Наприклад, після дослідження кількох основних біомів, таких як рівнини, ліси чи пустелі, гравці можуть відчувати, що нові регіони повторюють уже знайомі шаблони [2].

Структури, такі як села, храми, фортеці чи шахти, додають елемент пригод, пропонуючи гравцям точки інтересу з потенційною здобиччю. Однак кількість унікальних структур у «Vanilla» Minecraft є обмеженою, і гравці часто стикаються з повторюваними об’єктами, що знижує відчуття новизни. Наприклад, після відвідування кількох сіл або пустельних храмів гравець може передбачити їхній вміст, що зменшує мотивацію до подальшого дослідження.

У контексті бойової системи біоми та структури мають прямий вплив на доступність ресурсів і типи ворогів, з якими гравець бореться. Наприклад, у біомах, де рідко з’являються ворожі моби, бої стають менш динамічними, а мечі використовуються рідше. Додавання нового меча через модифікацію може вирішити цю проблему, якщо він буде пов’язаний із новими біомами чи структурами. Наприклад, меч може вимагати рідкісного ресурсу, який добувається лише в унікальному біомі, створеному модом, або бути частиною нагороди за дослідження нової структури, такої як стародавня фортеця чи підземний храм. Це не лише урізноманітнить геймплей, але й спонукатиме гравців досліджувати нові регіони.

Модифікація, що додає меч, може також інтегрувати нові біоми, які пропонують унікальні виклики для бойової системи. Наприклад, біом із вулканічним ландшафтом може містити ворогів, стійких до звичайних мечів, але вразливих до нового меча з ефектом вогню. Такі інновації повертають відчуття відкриття та роблять дослідження більш захоплюючим

Різновиди мобів. Моби є ключовим елементом бойової системи, визначаючи ворогів і стратегії бою. «Vanilla» Minecraft пропонує різноманітних істот, але їхня різноманітність здається обмеженою в певних біомах, де гравці стикаються з одними й тими ж ворогами. Це робить бої передбачуваними, знижуючи потребу в стратегічному використанні зброї [3].

Модифікація може додати меч із ефектами, ефективними проти певних мобів,

як-от додаткова шкода ендерменам чи отрута проти павуків. Це урізноманітнить бої та заохотить досліджувати біоми з новими ворогами. Наприклад, мод може додати «кришталевого голема», вразливого до меча з ефектом руйнування броні, що змусить гравців адаптувати стратегії.

Нові моби, які випадають ресурси для створення меча, додадуть мотивацію до бою. Наприклад, «містичний страж» може скидати «Містичний Розломосколок», необхідний для крафту. Це інтегрує бойову систему з економікою гри, роблячи геймплей більш цілісним.

Ресурси. Ресурси в «Vanilla» Minecraft розподілені нерівномірно, що ускладнює створення потужної зброї в бідних біомах. Алмази чи незерит є рідкісними, що створює нерівність між гравцями. Модифікація може ввести новий ресурс, як-от «містичний кристал», який добувається в новому біомі чи структурі, балансуючи економіку.

Альтернативні методи крафту, як-от комбінація звичайних матеріалів із рідкісним «ефірним порошком», зроблять меч доступнішим, але збережуть виклик. Система прогресування, де меч покращується новими ефектами, підтримає довготривалий інтерес. Наприклад, «вогняний кристал» може додати ефект полум’я, а «Містичний Розломосколок» – швидкість атаки.

1.2 Аналіз стану області рішення задачі

Базова версія Minecraft, хоча й доступна для новачків, має складну криву навчання, яка може бути викликом для гравців, особливо в контексті бойової системи та створення предметів, таких як мечі. Обмежена документація в грі, необхідність самостійного освоєння механік і складність доступу до рідкісних ресурсів створюють бар’єри, які можуть відлякувати нових гравців або знижувати мотивацію досвідчених. Розробка модифікації для додавання нового меча за допомогою API Forge може не лише урізноманітнити геймплей, але й спростити або збагатити криву навчання, пропонуючи нові механіки, які є зрозумілими, але

водночас додають глибини.

Система крафту в «Vanilla» Minecraft є однією з центральних механік, яка дозволяє гравцям створювати інструменти, зброю та предмети з ресурсів, зібраних у світі. Однак ця система має свої обмеження, які впливають на доступність і різноманітність бойової системи. У базовій грі крафт мечів обмежується стандартними матеріалами – деревом, каменем, залізом, золотом, алмазами та незеритом. Рецепти крафту прості, але не дозволяють експериментувати з унікальними комбінаціями чи створювати зброю з нестандартними характеристиками. Наприклад, усі мечі мають однакову механіку атаки, відрізняючись лише силою та міцністю, що знижує можливості для творчого підходу до бою.

Крім того, система крафту в «Vanilla» Minecraft вимагає від гравців знання рецептів, які не пояснюються в грі. Новачки можуть витрачати години, щоб дізнатися, як створити алмазний меч, або зіткнутися з труднощами через брак ресурсів, таких як алмази чи незерит. Ця відсутність підказок створює бар’єр, який може відлякувати менш досвідчених гравців. Навіть для ветеранів гри система крафту може здаватися одноманітною, оскільки після створення незеритового меча подальші покращення обмежуються зачаруваннями, які не завжди додають значної різноманітності.

Модифікація для додавання нового меча через API Forge може вирішити ці проблеми, запропонувавши більш гнучку та творчу систему крафту. Наприклад, меч може створюватися за допомогою спеціального верстака, який дозволяє комбінувати кілька типів матеріалів для досягнення унікальних ефектів. Такий верстак може бути частиною нової структури, як-от кришталевий алтар, що додасть елемент дослідження до процесу крафту. Наприклад, гравець може поєднувати «містичний кристал» із залізом для створення меча з ефектом отрути або додати «вогняний осколок» для ефекту полум’я. Це не лише урізноманітнить бойову систему, але й зробить крафт більш інтерактивним і захоплюючим.

Крім того, модифікація може включати вбудовану систему підказок, яка допомагає гравцям зрозуміти, як створити меч. Наприклад, при взаємодії з новим

верстаком гравець може отримувати повідомлення про необхідні ресурси чи можливі комбінації, що спростить криву навчання для новачків. Для досвідчених гравців система може пропонувати складні рецепти, які вимагають рідкісних матеріалів або виконання квестів, як-от перемога над новим мобом чи дослідження унікального біома. Такий підхід збалансує доступність і глибину, роблячи модифікацію привабливою для широкої аудиторії.

Ще одним аспектом є інтеграція крафту меча з економікою гри. У «Vanilla» Minecraft торгівля з NPC у селах є обмеженою, і гравці рідко використовують її для отримання матеріалів для зброї. Модифікація може додати нових NPC, таких як «містичний коваль», який пропонує ресурси для створення меча в обмін на інші предмети чи виконання завдань. Це додасть соціальний вимір до системи крафту, заохочуючи гравців взаємодіяти з ігровим світом і спільнотою.

Нарешті, модифікація може ввести систему прогресування для меча, де гравець поступово покращує його характеристики, додаючи нові ефекти чи посилюючи силу атаки. Наприклад, базовий меч може бути створений із звичайних матеріалів, але для додавання ефекту блискавки гравець повинен знайти рідкісний «Містичний Розломкристал» у новій структурі. Така система не лише підтримує довготривалий інтерес, але й робить процес крафту більш динамічним, дозволяючи гравцям експериментувати з різними комбінаціями та стратегіями.

Ігрова механіка. Ігрова механіка в «Vanilla» Minecraft, зокрема бойова система, є відносно простою, що робить її доступною, але водночас обмеженою для гравців, які шукають складніших викликів. Мечі в базовій грі використовуються для нанесення шкоди ворогам через базову атаку або критичний удар, але не пропонують різноманітних механік, таких як комбо, спеціальні атаки чи взаємодія з іншими системами гри. Ця простота може бути перевагою для новачків, але для досвідчених гравців вона швидко стає одноманітною, особливо в порівнянні з іншими іграми, які пропонують складні бойові системи.

Крім того, бойова механіка в «Vanilla» Minecraft має обмежену інтеграцію з іншими аспектами гри. Наприклад, мечі не взаємодіють із редстоуном, зачаруваннями чи зіллями в спосіб, який би додавав стратегічної глибини.

Зачарування, такі як «Гострота» чи «Вогняний аспект», покращують меч, але їхній вибір обмежений, а ефекти передбачувані. Це знижує можливості для експериментів і кастомізації, що є ключовим для гри-пісочниці, як Minecraft.

Модифікація для додавання нового меча через API Forge може збагатити ігрову механіку, запропонувавши унікальні бойові можливості, які змінюють підхід до бою. Наприклад, меч може мати спеціальну атаку, яка активується при певній комбінації дій, як-от подвійний удар для виклику вибуху чи утримання атаки для створення магічного бар'єру. Такі механіки зроблять бої більш динамічними та стратегічними, дозволяючи гравцям адаптувати стиль бою до різних ситуацій.

Інтеграція меча з іншими системами гри може додати глибини. Наприклад, меч може взаємодіяти з редстоуном, дозволяючи активувати спеціальні ефекти при підключенні до редстоун-механізмів, як-от виклик блискавки при проходженні сигналу. Альтернативно, меч може мати унікальні зачарування, доступні лише через новий стіл зачарування, який вимагає рідкісних ресурсів. Наприклад, зачарування «Містичний Розломудар» може тимчасово підвищувати швидкість гравця після критичного удару, що додасть тактичний елемент до бою.

Ще одним аспектом є взаємодія меча з мобами та навколишнім середовищем. У «Vanilla» Minecraft мечі не мають специфічних ефектів проти різних типів ворогів чи блоків, що обмежує їхню універсальність. Модифікація може додати меч із ефектами, які залежать від контексту. Наприклад, меч може завдавати додаткової шкоди в певних біомах, як-от у Незері, або руйнувати певні блоки швидше, як-от кришталеві структури в новому біомі. Це заохотить гравців досліджувати різні регіони та адаптувати стратегії бою.

Крива навчання для нових механік також є важливим фактором. Щоб уникнути перевантаження новачків, модифікація може вводити механіки поступово. Наприклад, базовий меч може мати прості ефекти, доступні одразу, тоді як складніші, як-от виклик природних явищ, відкриваються після виконання квестів чи збору рідкісних ресурсів. Це збалансує доступність і глибину, роблячи модифікацію привабливою для всіх гравців.

Розмірне дослідження. Розмірне дослідження в «Vanilla» Minecraft, зокрема в Незері та Енді, пропонує виклики, але обмежена кількість вимірів знижує інтерес після тривалої гри. Модифікація додає Містичний Розлом – унікальний вимір із різноманітними біомами, ворогами та структурами, які збагачують бойову систему та інтегрують меч у геймплей. Містичний Розлом пропонує лунні рівнини з сяючими блоками, тіньові хащі з густими лісами, зоряні піки з кристалічними утвореннями та лунні святині – структури з головоломками, пастками та босами. Кожен біом має унікальні ресурси, ворогів і виклики, що мотивують дослідження [4].

Лунні рівнини – відкритий біом із сяючими лунними блоками, які можна видобувати мечем із ефектом зоряного спалаху для отримання «лунних осколків». Тут мешкають «тіньові ловці» – швидкі моби, що атакують із темряви, але вразливі до зоряного спалаху, який їх оглушує. Тіньові хащі – густі ліси з обмеженою видимістю, де «тіньові ловці» ховаються в темряві, а «тіньовий порошок» випадає після їхньої поразки. Зоряні піки – високі кристалічні гори, де мешкають «зоряні вартові» – боси, які охороняють «зоряну есенцію», необхідну для покращення меча. Лунні святині – складні структури з редстоун-головоломками, пастками та скарбницями, де гравець знаходить артефакти для створення меча чи активації порталу.

Ефекти меча в Розломі контекстуальні. У тіньових хащах зоряний спалах підсилює видимість, оглушуючи ворогів, тоді як на зоряних піках місячний резонанс руйнує кристалічні блоки, полегшуючи видобуток. У лунних рівнинах меч може викликати ефект туману, який приховує гравця від «тіньових ловців». Ці механіки роблять бої динамічними, змушуючи гравців адаптувати стратегії до біома. Наприклад, проти «зоряного вартового» місячний резонанс руйнує його магічний щит, тоді як зоряний спалах неефективний, що додає тактичної глибини.

Доступ до Містичного Розлому вимагає створення порталу, який активується в лунній святині після виконання квесту. Гравець повинен зібрати «тіньовий порошок» із «тіньових ловців» і «лунний осколок» із лунних блоків, а також перемогти «тіньового вождя» – міні-боса в тіньових хащах. Квест включає

головоломку в святині, де гравець активує редстоун-механізм, щоб відкрити портал. Цей процес інтегрує дослідження, бій і логіку, роблячи доступ до Розлому нагороджуючим і захоплюючим.

Наратив Містичного Розлому пов’язаний із легендою про зоряного воїна, чиї реліквії, включаючи меч, заховані в вимірі. Гравець виконує серію квестів, щоб зібрати три артефакти: перший у звичайному світі (лунна святиня з головоломкою), другий у Незері (перемога над босом у новій структурі, тіньовій фортеці), третій у Розломі (збір «зоряної есенції» після перемоги над «зоряним вартовим»). Кожен квест включає унікальні виклики: головоломки, бої, дослідження. Наприклад, у тіньовій фортеці гравець повинен уникати пасток і розгадати редстоун-код для доступу до скарбниці. Ці квести інтегрують Розлом із іншими вимірами, створюючи єдину історію.

Економіка Розлому збагачує геймплей. «Лунні осколки» і «тіньовий порошок» добуваються в різних біомах, а «зоряна есенція» – нагорода за босів. Гравці можуть торгувати з «місячними хранителями» – NPC у лунних святинях, які пропонують ресурси за артефакти чи рідкісні предмети, як-от алмази. На мультиплеєрі це сприяє обміну, де гравці торгують «тіньовим порошком» чи «лунними осколками», створюючи економічні зв’язки. Лунні святині містять скарбниці з ресурсами, але вимагають розгадування головоломок чи перемоги над ворогами, що додає виклику.

Бойова система в Розломі збагачується унікальними ворогами. «Тіньові ловці» уникають прямих ударів, але вразливі до зоряного спалаху, який їх оглушує. «Зоряні вартові» мають магічний щит, який руйнується місячним резонансом, але стійкі до інших ефектів. Нові боси, як-от «лунний титан» у лунних святинях, вимагають комбінації ефектів меча: зоряний спалах послаблює його броню, а місячний резонанс завдає шкоди. Ці вороги змушують гравців використовувати меч стратегічно, додаючи глибини. Наприклад, проти груп «тіньових ловців» ефективний ефект туману, який приховує гравця, дозволяючи атакувати з несподіванки.

Квести в Розломі інтегрують дослідження з прогресуванням. Наприклад,

квест «Шлях зоряного воїна» вимагає зібрати три «зоряні есенції» з різних біомів Розлому, кожен із яких охороняється босом. У лунних рівнинах бос – «тіньовий вождь», у тіньових хащах – «лунний титан», на зоряних піках – «зоряний вартовий». Кожен бій вимагає різних ефектів меча, а перемога відкриває доступ до скарбниці з ресурсами. Головоломки в святинях, як-от активація редстоун-механізмів чи розстановка кристалів, додають логічний елемент, роблячи дослідження різноманітним.

Містичний Розлом впливає на економіку гри, зменшуючи залежність від Незеру чи Енду. «Лунні осколки» і «тіньовий порошок» стають ключовими ресурсами для створення меча, а їхня торгівля з NPC чи гравцями додає соціальний вимір. Наприклад, на сервері гравець може обміняти «тіньовий порошок» на алмази, щоб купити «лунний осколок» у «місячного хранителя». Структури Розлому, як-от тіньові фортеці, містять рідкісні артефакти, які можна продати чи використати для покращення меча, що мотивує дослідження.

Мультиплеєрна взаємодія в Розломі збагачується командними квестами. Наприклад, бій із «лунним титаном» вимагає координації: один гравець використовує зоряний спалах, щоб послабити боса, інший – місячний резонанс, щоб зруйнувати його щит. Ефект «зоряна аура» меча підсилює команду, підвищуючи швидкість чи шкоду союзників. Це сприяє співпраці, додаючи стратегії для командних пригод. Наратив квестів, пов’язаний із легендою зоряного воїна, робить Розлом частиною епічної історії, підвищуючи залучення гравців.

1.3 Проблеми з продуктивністю та оптимізацією

Додавання нового меча та Містичного Розлому через API Forge створює виклики для продуктивності, особливо через складні ефекти, текстури, моделі, ворогів і структури. Minecraft є ресурсомісткою грою, і модифікації можуть знижувати частоту кадрів, збільшувати час завантаження чи викликати збої, особливо на слабких системах. Сумісність із іншими модами також критична,

оскільки гравці часто використовують десятки модів одночасно. Цей підрозділ детально аналізує ці проблеми та способи їх вирішення.

Сумісність модів. API Forge забезпечує модульну структуру, але конфлікти з іншими модами, які змінюють бойову систему, інвентар, генерацію світу чи зачарування, можуть створювати проблеми. Наприклад, Tinker's Construct, що додає власну систему створення зброї, може конфліктувати з ефектами меча, як-от зоряний спалах чи місячний резонанс, викликаючи збої чи втрату функціональності. Аналогічно, моди, як-от Biomes O' Plenty, що змінюють генерацію світу, можуть впливати на появу лунних святинь чи тіньових хащ. Моди, як-от Enchanting Plus, що змінюють зачарування, можуть створювати несумісність із «зоряним столом».

Для вирішення цих проблем модифікація використовуватиме модульну структуру Forge, дозволяючи гравцям відключати окремі ефекти чи механіки через конфігураційний файл. Наприклад, якщо Tinker's Construct викликає конфлікт, гравець може відключити унікальні зачарування меча, залишивши базові ефекти. Модифікація уникатиме переписування базових класів Minecraft, використовуючи події Forge, як-от LivingHurtEvent для ефектів атаки чи PlayerInteractEvent для взаємодії з зоряним алтарем. Це забезпечить коректну роботу з іншими модами.

Тестування сумісності є обов’язковим. Модифікація буде перевірена з популярними модами, як-от Tinker's Construct, JEI, Biomes O' Plenty, Twilight Forest і OptiFine, щоб виявити конфлікти. Наприклад, JEI може інтегрувати рецепти зоряного алтаря, дозволяючи гравцям переглядати потрібні ресурси, як-от «лунний осколок» чи «тіньовий порошок». Патч сумісності з Tinker's Construct додасть «лунний осколок» як матеріал для створення, дозволяючи інтегрувати меч у цю систему. Для Biomes O' Plenty модифікація перевірить, чи лунні святині генеруються в нових біомах, і додасть конфігурацію для їхньої появи.

Мультиплеєрна сумісність вимагає особливої уваги. Ефекти меча, як-от зоряний спалах чи зоряна аура, повинні бути синхронізовані між клієнтом і сервером, щоб уникнути розбіжностей у відображенні. Наприклад, частинки зоряного спалаху повинні відображатися однаково для всіх гравців. Серверні

налаштування дозволять адміністраторам балансувати характеристики меча, як-от радіус аури чи силу атаки, для мультиплеєрного геймплею. Тестування на серверах із 10–20 гравцями, що використовують різні моди, оцінить стабільність. Модифікація включатиме API для інших модів, дозволяючи їм додавати ресурси до зоряного алтаря чи інтегрувати ворогів Розлому, що підвищить її популярність.

Ще одним аспектом є сумісність із модами, що додають нові виміри, як-от Twilight Forest. Містичний Розлом повинен генеруватися незалежно, без конфліктів із порталами чи біомами інших вимірів. Модифікація перевірить, чи портал до Розлому коректно активується в лунних святинях, і додасть конфігурацію для його розташування. Інтеграція з модами, як-от Baubles, може дозволити мечу взаємодіяти з амулетами, додаючи синергію (наприклад, амулет підсилює зоряну ауру). Такий підхід зробить модифікацію частиною екосистеми модів, підвищуючи її цінність для спільноти [5].

Використання пам'яті. Нові текстури, моделі, ефекти, вороги та структури Містичного Розлому можуть значно підвищувати споживання пам'яті, що критично для слабких систем. Minecraft, попри піксельну графіку, є ресурсомісткою грою, і модифікації часто знижують частоту кадрів чи збільшують час завантаження. Наприклад, текстури меча з високою роздільною здатністю (64x64 чи 128x128) чи складні частинки для зоряного спалаху можуть створювати навантаження на оперативну пам'ять і відеокарту. Аналогічно, генерація біомів Розлому, як-от тіньові хащі чи зоряні піки, та обробка ворогів, як-от «зоряні вартові», підвищують вимоги до процесора.

Для оптимізації модифікація використовуватиме текстури 16x16, що відповідають стилю «Vanilla» Minecraft, із підтримкою пакетів ресурсів для текстур 64x64 чи 128x128 для потужних систем. Наприклад, текстура меча буде 16x16 за замовчуванням, але гравці зможуть установити пакет із вищою роздільною здатністю. Ефекти, як-от зоряний спалах чи місячний резонанс, матимуть обмежену кількість частинок (наприклад, 10–15), які зникають через 2–3 секунди, щоб уникнути накопичення. Конфігураційний файл дозволить відключати ефекти, як-от частинки чи динамічне освітлення, для слабких систем, що підвищить

частоту кадрів.

Моделі меча, створені через JSON-файли Forge, будуть простими, використовуючи стандартні шаблони для ручних предметів із мінімальними змінами для унікального вигляду. Моделі ворогів, як-от «тіньові ловці», матимуть низькополігональну геометрію, щоб зменшити час обробки. Генерація структур, як-от лунні святині, буде оптимізована через кешування шаблонів при завантаженні світу, уникаючи повторного обчислення. Наприклад, редстоун-головоломки в святинях оброблятимуться лише при взаємодії гравця, що зменшить навантаження [6].

Події, як-от активація місячного резонансу чи зоряного спалаху, перевірятимуться лише при ударі чи активації, а не кожні кілька тиків, що знизить вимоги до процесора. Асинхронна обробка частинок і ефектів, як-от туману, не блокуватиме основний потік гри. Кешування текстур, моделей і конфігурацій при завантаженні зменшить читання з диска під час гри, що особливо важливо для мультиплеєрних серверів із багатьма гравцями.

Генерація Містичного Розлому також потребує оптимізації. Біоми, як-от тіньові хащі, використовуватимуть спрощені алгоритми генерації з обмеженою кількістю декоративних блоків, як-от сяючі лунні блоки. Лунні святині генеруватимуться рідко (наприклад, одна на 5000 блоків), щоб зменшити навантаження на сервер. Вороги, як-от «тіньові ловці», матимуть оптимізовану поведінку, де їхні атаки перевіряються лише при близькості до гравця, уникаючи зайвих обчислень [7].

Тестування продуктивності проведуть на різних конфігураціях: від слабких систем (4 ГБ RAM, вбудована графіка) до потужних (16 ГБ RAM, дискретна відеокарта). Наприклад, тестування на слабкому ноутбуці оцінить частоту кадрів у тіньових хащах із активним зоряним спалахом. Перевірка з ресурсомісткими модами, як-от OptiFine чи Biomes O' Plenty, виявить проблеми сумісності. Тестування на сервері з 20 гравцями, що досліджують Розлом, оцінить стабільність і затримки.

Модифікація включатиме налагоджувальний режим, який відображає

споживання пам’яті, кількість оброблюваних подій (наприклад, частинки зоряного спалаху) і частоту кадрів. Гравці зможуть увімкнути цей режим через конфігурацію, щоб оптимізувати налаштування. Наприклад, якщо частота кадрів падає в тіньових хащах, гравець може відключити частинки чи знизити деталізацію біома. Такий підхід підвищить доступність модифікації для всіх гравців.

Оптимізація мультиплеєрних аспектів включає синхронізацію даних між клієнтом і сервером. Наприклад, ефект зоряної аури розраховуватиметься на сервері, а клієнт отримуватиме лише необхідні дані для відображення, зменшуючи мережеве навантаження. Лунні святині матимуть кешовані шаблони, які завантажуються один раз при генерації світу, уникаючи повторних обчислень для кожного гравця. Вороги, як-от «зоряні вартові», матимуть оптимізовану поведінку, де їхні атаки активуються лише при взаємодії, що знизить навантаження на сервер.

Ще одним аспектом є оптимізація квестів і головоломок. Редстоун-головоломки в лунних святинях оброблятимуться локально, лише для гравця, що взаємодіє, уникаючи глобальних обчислень. Квести, як-от збір «зоряної есенції», матимуть кешовані стани, де прогрес зберігається при завантаженні світу, зменшуючи час обробки. Це забезпечить плавну роботу навіть на серверах із великою кількістю гравців.

Нарешті, модифікація підтримуватиме інтеграцію з OptiFine – популярним модом для оптимізації. Наприклад, частинки зоряного спалаху будуть сумісні з налаштуваннями OptiFine для зменшення анімацій. Лунні блоки в Розломі матимуть оптимізоване освітлення, сумісне з динамічним світлом OptiFine, що знизить навантаження на відеокарту. Такий підхід зробить модифікацію доступною для гравців із різними системами та конфігураціями модів, підвищуючи її популярність у спільноті [8].

Відстань промальовки. Відстань промальовки в Minecraft визначає, як далеко гра рендерить блоки, сутності та ефекти, що значно впливає на продуктивність, особливо при використанні модифікацій. Містичний Розлом із його біомами (лунні рівнини, тіньові хащі, зоряні піки) та структурами (лунні святині, тіньові фортеці) додає складні елементи, які можуть знижувати частоту кадрів на великих відстанях

промальовки. Наприклад, сяючі лунні блоки на лунних рівнинах використовують динамічне освітлення, що підвищує навантаження на відеокарту, тоді як густі тіньові хащі з великою кількістю листя та декоративних блоків ускладнюють рендеринг. Аналогічно, ефекти меча, як-от зоряний спалах чи місячний резонанс, створюють частинки та анімації, які обробляються на великих відстанях, якщо гравець установив високу відстань промальовки (наприклад, 16 чанків).

Для оптимізації модифікація використовуватиме кілька стратегій. По-перше, біоми Містичного Розлому матимуть спрощену генерацію на великих відстанях. Наприклад, у тіньових хащах кількість декоративних блоків, як-от тіньові ліани чи сяючі гриби, зменшуватиметься за межами 8 чанків, замінюючись базовими блоками, що знижує навантаження. Лунні блоки на лунних рівнинах матимуть оптимізоване освітлення, де динамічні ефекти активуються лише в межах 6 чанків від гравця, уникаючи зайвих обчислень. Зоряні піки, із їхніми кристалічними утвореннями, використовуватимуть низькополігональні моделі на відстані, що зберігає візуальну якість, але зменшує вимоги до рендерингу.

Ефекти меча також будуть оптимізовані для відстані промальовки. Зоряний спалах, який створює частинки осліплення, матиме максимальний радіус відображення 8 чанків, після чого частинки зникають, щоб уникнути обробки на великих відстанях. Місячний резонанс, що генерує звукову хвилю, використовуватиме спрощені анімації (наприклад, 5–10 частинок замість 20) за межами 6 чанків. Гравці зможуть налаштувати ці ефекти через конфігураційний файл, зменшуючи їхню інтенсивність або відключаючи на великих відстанях промальовки. Наприклад, на слабких системах частинки можна відключити повністю, залишивши лише звуковий ефект для місячного резонансу.

Структури, як-от лунні святині, матимуть оптимізовану генерацію. Їхні редстоун-головоломки та пастки активуватимуться лише при наближенні гравця (в межах 4 чанків), уникаючи обробки на відстані. Тіньові фортеці, які містять складні елементи, як-от кристалічні арки чи скарбниці, використовуватимуть кешовані шаблони, що завантажуються один раз при генерації світу. Це зменшить час рендерингу, особливо на серверах із великою відстанню промальовки (наприклад,

12–16 чанків).

Вороги Містичного Розлому, як-от «тіньові ловці» чи «зоряні вартові», також оптимізовані. Їхня поведінка (атаки, рух) оброблятиметься лише в межах 8 чанків від гравця, а на більших відстанях вони переходять у «сплячий режим», де не виконують обчислень. Наприклад, «тіньовий ловець» у тіньових хащах припиняє рух за межами 8 чанків, що знижує навантаження на процесор. Боси, як-от «лунний титан», матимуть обмежений радіус активації (6 чанків), щоб їхні анімації та ефекти не оброблялися на відстані.

Тестування відстані промальовки проведуть на різних конфігураціях. На слабких системах (4 ГБ RAM, вбудована графіка) перевірять частоту кадрів у тіньових хащах із відстанню промальовки 8 чанків, активним зоряним спалахом і 5–10 «тіньовими ловцями». На потужних системах (16 ГБ RAM, дискретна відеокарта) тестування оцінить стабільність на 16 чанках у зоряних піках із «лунним титаном». Тестування на серверах із 20 гравцями, що досліджують Розлом, перевірить вплив великих відстаней промальовки (12–16 чанків) на затримки. Результати допоможуть налаштувати конфігурацію, пропонуючи гравцям оптимальні значення відстані (наприклад, 8–10 чанків для слабких систем).

Інтеграція з OptiFine, популярним модом для оптимізації, підвищить продуктивність. Наприклад, частинки зоряного спалаху будуть сумісні з налаштуваннями OptiFine для зменшення анімацій, а лунні блоки матимуть оптимізоване освітлення для динамічного світла. Конфігураційний файл модифікації дозволить гравцям установити максимальну відстань промальовки для біомів Розлому (наприклад, 10 чанків), що зменшить навантаження. Ці заходи забезпечать стабільну роботу навіть на великих відстанях промальовки, роблячи модифікацію доступною для всіх гравців.

Використання API Forge. API Forge є потужним інструментом для створення модифікацій, але його використання може впливати на продуктивність через складні події, обробку даних і взаємодію з іншими модами. Модифікація для додавання меча та Містичного Розлому використовує численні події Forge, як-от LivingHurtEvent для ефектів атаки, PlayerInteractEvent для взаємодії з зоряним

алтарем і WorldEvent для генерації лунних святинь. Кожна подія додає обчислювальне навантаження, особливо якщо обробляється часто або в мультиплеєрі.

Для оптимізації модифікація мінімізуватиме кількість подій. Наприклад, ефект зоряного спалаху перевірятиметься лише при ударі по ворогу, а не кожні кілька тиків, що зменшить виклики LivingHurtEvent. Аналогічно, місячний резонанс активуватиметься через одноразову перевірку в PlayerInteractEvent, уникаючи постійного моніторингу. Генерація біомів Розлому, як-от тіньові хащі, використовуватиме кешовані шаблони в WorldEvent, завантажуючи дані один раз при створенні світу, що знизить навантаження на сервер.

Кешування даних є ключовим для продуктивності. Текстури меча, моделі ворогів («тіньові ловці», «зоряні вартові») і конфігурації зоряного алтаря кешуватимуться при завантаженні гри, уникаючи повторного читання з диска. Наприклад, модель «лунного титана» завантажуватиметься один раз і зберігатиметься в пам’яті, що прискорить рендеринг під час бою. Ресурси Розлому, як-от «лунний осколок», матимуть кешовані дані для торгівлі з NPC, зменшуючи обчислення при взаємодії.

Мультиплеєрна оптимізація включає асинхронну обробку подій. Наприклад, ефект зоряної аури розраховуватиметься на сервері асинхронно, а клієнт отримуватиме лише дані для відображення, що зменшить мережеве навантаження. Події, пов’язані з квестами (збір «зоряної есенції»), матимуть кешовані стани, де прогрес гравця зберігається при завантаженні світу, уникаючи повторних перевірок. Це забезпечить плавну роботу на серверах із 20–30 гравцями.

Сумісність із іншими модами через API Forge також оптимізована. Модифікація використовуватиме стандартні інтерфейси Forge для інтеграції з модами, як-от JEI чи Tinker's Construct. Наприклад, JEI відображатиме рецепти зоряного алтаря, а Tinker's Construct може додати «лунний осколок» як матеріал. Події Forge, як-от EntityJoinWorldEvent, використовуватимуться для ворогів Розлому, уникаючи конфліктів із модами, що змінюють моби. Тестування з 10–15 популярними модами (OptiFine, Biomes O' Plenty, Twilight Forest) перевірить

стабільність і виявить проблеми.

Оптимізація подій Forge включає дебаг-режим, який відображає кількість оброблюваних подій (наприклад, виклики LivingHurtEvent для зоряного спалаху) і їхній вплив на продуктивність. Гравці зможуть увімкнути цей режим через конфігурацію, щоб виявити проблемні події та відключити їх. Наприклад, якщо зоряна аура викликає затримки на сервері, гравець може зменшити її радіус або частоту активації. Такий підхід підвищить гнучкість і стабільність модифікації.

Тестування API Forge проведуть на різних версіях Minecraft (1.16.5, 1.18.2, 1.20.1), щоб забезпечити сумісність із популярними релізами Forge. Перевірка на слабких системах оцінить вплив подій на частоту кадрів, а на серверах – затримки при обробці квестів чи торгівлі. Результати допоможуть оптимізувати конфігурацію, пропонуючи гравцям налаштування для різних систем. Наприклад, слабкі системи можуть зменшити частоту подій для ворогів Розлому, тоді як потужні – увімкнути всі ефекти.

Висновки по розділу

Базова версія Minecraft, хоча й пропонує унікальний геймплей, має обмеження, які знижують її привабливість для гравців, що шукають різноманітності та глибини. Одноманітність бойової системи, зокрема мечів, обмежена система крафту, складна крива навчання та недостатнє залучення після кінцевих цілей створюють бар’єри для новачків і ветеранів. Модифікація для додавання нового меча та Містичного Розлому через API Forge вирішує ці проблеми, пропонуючи інноваційні механіки, які збагачують геймплей.

РОЗДІЛ 2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Залучення гравців і довголіття

Модифікація для додавання меча та Містичного Розлому спрямована на підтримку довготривалого інтересу гравців через різноманітні механіки, наратив і соціальну взаємодію. У «Vanilla» Minecraft залучення гравців знижується після досягнення кінцевих цілей, як-от перемога над ендер-драконом чи створення незеритової зброї, через обмеженість контенту. Модифікація вирішує цю проблему, пропонуючи нові виклики, прогресування та мультиплеєрні можливості.

Ключовим елементом є система квестів, пов’язана з легендою про зоряного воїна. Гравець виконує серію завдань, щоб створити та покращити меч, досліджуючи Містичний Розлом і інші виміри. Наприклад, квест «Шлях зоряного воїна» вимагає зібрати три артефакти: у звичайному світі (лунна святиня з головоломкою), Незері (тіньова фортеця з босом) і Розломі («зоряна есенція» від «лунного титана»). Кожен квест поєднує бій, дослідження та логіку, як-от розгадування редстоун-головоломок чи уникнення пасток. Ці завдання підтримують інтерес, пропонуючи нагороди (ресурси, ефекти меча) і розкриваючи історію воїна, чиї реліквії заховані в Розломі.

Прогресування меча забезпечує довголіття. Базовий меч із «лунним осколком» має ефект зоряного спалаху, але гравець може додати місячний резонанс, тіньовий удар чи виклик туману, збираючи ресурси («тіньовий порошок», «зоряна есенція») через квести чи бої. Кожен ефект відкриває нові стратегії бою, як-от використання туману проти «тіньових ловців» чи резонансу проти «зоряних вартових». Система модульності (лезо, рукоятка, кристал) дозволяє кастомізувати меч, створюючи унікальні комбінації. Наприклад, лезо з «зоряної есенції» додає магічну шкоду, а рукоятка з обсидіану – міцність. Це мотивує гравців експериментувати, підтримуючи інтерес.

Мультиплеєрна взаємодія підвищує залучення через командні механіки.

Ефект зоряної аури меча підсилює характеристики союзників (швидкість, шкода), що корисно в боях із босами, як-от «лунний титан». Командні квести, як-от захист лунної святині від «тіньових ловців», вимагають координації: один гравець використовує зоряний спалах, інший – резонанс. Торгівля ресурсами («лунний осколок», «тіньовий порошок») з «місячними хранителями» чи гравцями створює економічні зв'язки. Наприклад, на сервері гравець може обміняти «тіньовий порошок» на алмази, щоб купити «зоряну есенцію», що сприяє співпраці [2].

Баланс для новачків і ветеранів забезпечує доступність. Новачки отримують підказки від зоряного алтаря та прості ефекти меча (зоряний спалах), тоді як ветерани стикаються зі складними квестами, як-от бій із «лунним титаном», що вимагає комбінації ефектів. Прості біоми Розлому, як-от лунні рівнини, доступні для новачків, тоді як тіньові хащі чи зоряні піки пропонують виклики для досвідчених. Конфігурація дозволяє налаштувати складність, наприклад, зменшити силу «зоряних вартових» для новачків.

Наратив про зоряного воїна додає емоційне залучення. Гравець відкриває історію через записи в лунних святинях, дізнаючись про воїна, який боровся з темними силами в Розломі. Кожен квест розкриває частину історії, мотивуючи завершити завдання. Наприклад, перемога над «лунним титаном» відкриває запис про останню битву воїна, що додає епічності. Мультиплеєрні сервери можуть адаптувати наратив, дозволяючи командам створювати власні історії, пов'язані з Розломом.

Довголіття підтримується регулярним оновленням контенту. Модифікація може додавати нові квести, ворогів чи ефекти меча через оновлення. Наприклад, новий бос у зоряних піках чи ефект «зоряний вихор» (створює магічний вихор) може з'явитися в майбутніх версіях. Спільнота може пропонувати ідеї через конфігураційні файли, додаючи власні ресурси чи квести, що підвищує залучення. Інтеграція з модами, як-от JEI чи Twilight Forest, розширює контент, дозволяючи гравцям досліджувати Розлом у різних контекстах [1].

2.2 Творчість і налаштування

Minecraft як гра-пісочниця підкреслює творчість, але «Vanilla» версія обмежує кастомізацію зброї. Модифікація для додавання меча та Містичного Розлому розширює творчі можливості через модульну систему, інтеграцію з редстоуном і підтримку кастомізації.

Модульна система меча дозволяє гравцям створювати унікальні комбінації через зоряний алтар. Меч складається з леза, рукоятки та кристала, кожен із яких впливає на характеристики. Наприклад, лезо з «лунного осколка» додає зоряний спалах, тоді як лезо з «зоряної есенції» – місячний резонанс. Рукоятка з заліза підвищує міцність, а з обсидіану – стійкість до вибухів. Кристал із «тіньового порошку» додає тіньовий удар, ефективний уночі. Гравці можуть експериментувати, створюючи мечі для різних стилів гри: магічний меч із «зоряною есенцією» для бою з босами чи міцний меч із обсидіаном для виживання.

Інтеграція з редстоуном розширює творчість. Меч може взаємодіяти з редстоун-схемами, активуючи ефекти, як-от зоряний спалах, при проходженні сигналу. Наприклад, гравець може побудувати пастку, де важіль викликає звукову хвилю через меч, відкидаючи ворогів. Зоряний алтар може бути частиною редстоун-механізму, де активація сигналу відкриває нові рецепти. Лунні блоки Розлому, які реагують на редстоун, дозволяють створювати сяючі структури чи пастки, як-от стіни, що руйнуються при сигналі. Це заохочує гравців будувати складні механізми, інтегруючи меч і Розлом у творчий геймплей.

Кастомізація через пакети ресурсів дозволяє змінювати вигляд меча, ворогів і біомів Розлому. Наприклад, гравець може створити текстуру меча в стилі кіберпанку чи фентезі, зберігши базові характеристики. Лунні блоки можуть мати різні текстури (кришталеві, металеві), що підтримує естетичну кастомізацію. Конфігураційний файл дозволяє налаштувати ефекти меча, як-от радіус зоряної аури чи інтенсивність частинок, що дає гнучкість для творчих гравців. Наприклад, гравець може збільшити кількість частинок для зоряного спалаху, створюючи ефектний візуальний ефект.

Творчість підтримується через квести та структури. Лунні святині містять редстоун-головоломки, які гравці можуть адаптувати для власних будівель. Наприклад, механізм активації порталу до Розлому можна відтворити в базі гравця, додаючи естетичну та функціональну цінність. Тіньові фортеці дозволяють створювати власні пастки чи скарбниці, використовуючи лунні блоки та «тіньовий порошок». Гравці можуть ділитися цими структурами в спільноті, підвищуючи соціальну взаємодію.

Мультиплеєрна творчість збагачується командними проєктами. Гравці можуть будувати спільні лунні святині, інтегруючи меч у механізми, як-от захист бази через зоряну ауру. Квести, як-от створення меча для команди, заохочують співпрацю, де кожен гравець додає ресурси чи ідеї. Наприклад, один гравець добуває «лунний осколок», інший проектує редстоун-схему для алтаря. Конфігурація дозволяє серверам додавати власні рецепти чи ефекти, що підтримує творчість адміністраторів.

Спільнота може розширювати модифікацію через API. Гравці можуть додавати нові ресурси до зоряного алтаря чи ворогів до Розлому, створюючи власні квести чи біоми. Наприклад, фанат може додати «кришталевий осколок» для нового ефекту меча чи біом «зоряні пустелі». Це підтримує довголіття, дозволяючи спільноті впливати на розвиток модифікації.

Висновки по розділу

Базова версія Minecraft, хоча й пропонує унікальний геймплей, має обмеження, які знижують її привабливість для гравців, що шукають різноманітності та глибини. Одноманітність бойової системи, зокрема мечів, обмежена система крафту, складна крива навчання та недостатнє залучення після кінцевих цілей створюють бар’єри для новачків і ветеранів. Модифікація для додавання нового меча та Містичного Розлому через API Forge вирішує ці проблеми, пропонуючи інноваційні механіки, які збагачують геймплей.

Система крафту через зоряний алтар дозволяє створювати мечі з унікальними ефектами, як-от зоряний спалах чи місячний резонанс, додаючи творчості та доступності через підказки. Ігрова механіка збагачується спеціальними атаками, інтеграцією з редстоуном і контекстуальними ефектами, що роблять бої стратегічними. Містичний Розлом із його біомами, ворогами та квестами розширює дослідження, інтегруючи меч у наратив про зоряного воїна. Залучення гравців підтримується через прогресування, мультиплеєрні механіки та оновлення контенту, тоді як творчість процвітає завдяки модульності, редстоуну та кастомізації.

Продуктивність залишається викликом через складність Розлому та ефектів меча, але оптимізація відстані промальовки, використання пам’яті, подій Forge і сумісність із модами забезпечують стабільність. Тестування на різних системах і серверах гарантує доступність для широкої аудиторії. Модифікація не лише вирішує обмеження «Vanilla» Minecraft, але й створює платформу для подальших інновацій, як-от нові виміри, вороги чи механіки. Цей підхід робить її цінним внеском у спільноту Minecraft, пропонуючи геймплей, який поєднує творчість, стратегію та довготривале залучення.

РОЗДІЛ 3 ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ ТА ПРОГРАМНа реалізація

3.1 Створення та налаштування проекту модифікації

Для розробки модифікації була обрана Java, так як це основна мова на якій розробляють модифікації для Minecraft. API Forge був обраний, через свою зручність, під час процесу розробки. Також для створення моделі було обрано програму BlockBench (рисунок 2.1).

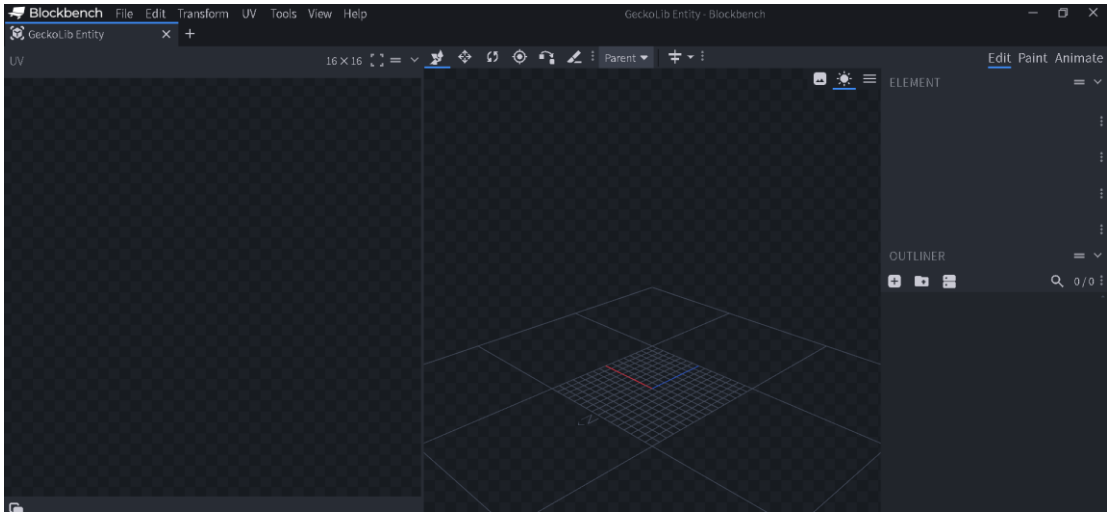


Рисунок 2.1 – Інтерфейс програми BlockBench

Також були використані бібліотеки GeckoLib, яка відповідає за анімації, для можливості у подальшому доробляти модифікацію.

3.2 Початок роботи

Для початку роботи нам потрібно створити проект, або завантажити гтовий. Для завантаження ми можемо скористатися сайтами які генерують шаблони, знайти по пошуковому запросу, або використати програму для створеня моду, яка сама створить проект. У нашому випадку ми завантажили шаблон по пошуковому запросу JDK 17 (Java Development Kit) . Використаємо 17-ту версію, тому що вона

сумісна з патчем гри, для котрого ми робимо модифікаццію (рисунок 2.2).

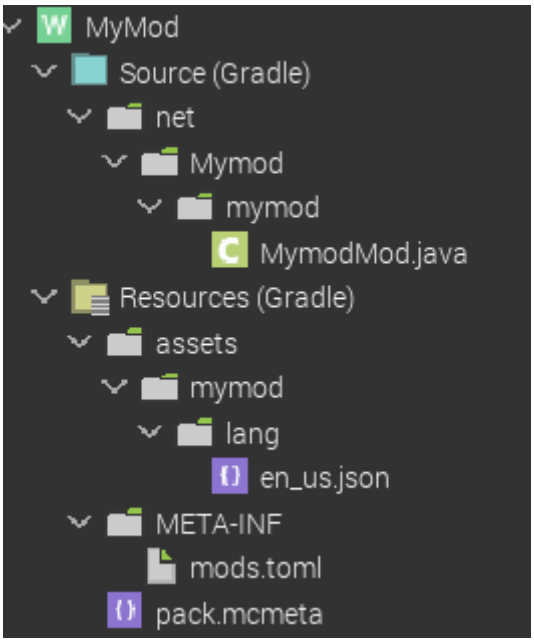


Рисунок 2.2 – Стандартна структура шаблону

Також додаємо класи та нові папки сгідно тому як гра читає структуру файлів (рисунок 2.3).

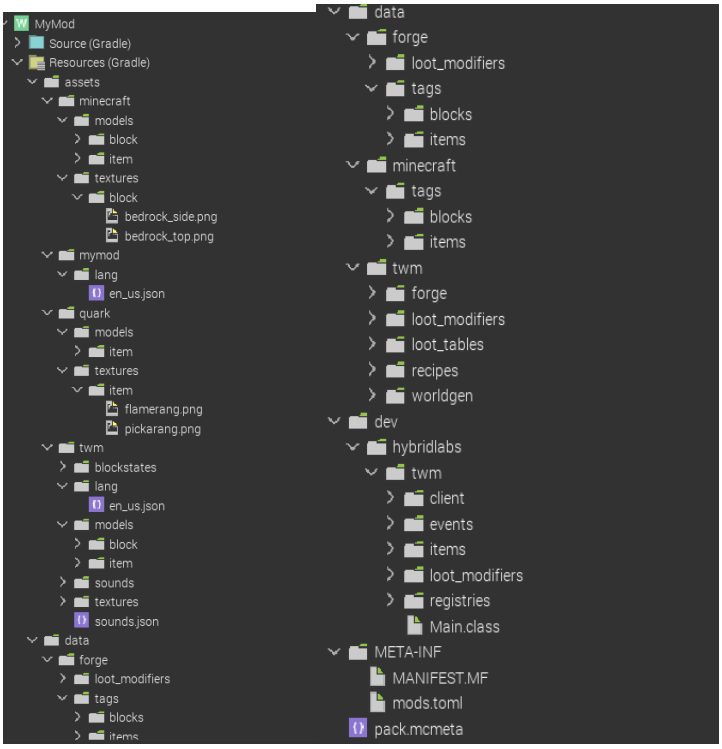


Рисунок 2.3 – Структура моду

3.3 Реєстрація нових предметів у грі

Реєстрація нових предметів у Minecraft – це процес додавання кастомних предметів (наприклад, зброї, інструментів чи інших об’єктів) у гру через модифікацію. Цей процес дозволяє інтегрувати створені тобою предмети (зокрема, із JSON-моделями) у ігровий світ, щоб гравці могли їх використовувати, бачити в інвентарі чи крафті. Реєстрація – це прив’язка предмета до унікального ідентифікатора і додавання його до реєстру гри (Registry). Це відбувається під час завантаження мода, щоб Minecraft розпізнав предмет і відобразив його.

Так як в нас багато предметів , ми проведемо регістрацію через індерекцію, через Registry.init(). У цьому виклику будуть сходитись усі предмети, а реєстрацію окремих предметів будуть виконувати інші класи.

Головний класс має такий вигляд:

```
package dev.hybridlabs.MyMod;

import net.minecraftforge.common.MinecraftForge;
import net.minecraftforge.event.TickEvent; import
net.minecraftforge.eventbus.api.IEventBus; import
net.minecraftforge.eventbus.api.SubscribeEvent; import
net.minecraftforge.fml.common.Mod; import org.slf4j.Logger;
import com.mojang.logging.LogUtils;

import java.util.ArrayList; import
java.util.Collection; import java.util.List; import
java.util.AbstractMap.SimpleEntry;

@Mod("MyMod") public class Main { public static final
Logger LOGGER = LogUtils.getLogger(); public static final
String MODID = "MyMod"; private Collection<SimpleEntry>
workQueue = new
java.util.concurrent.ConcurrentLinkedQueue<>();

public Main() { IEventBus eventBus =
MinecraftForge.EVENT_BUS; eventBus.register(this);
Registry.init(); // Ініціалізація реєстрів }

private void queueServerWork(int tick, Runnable
action) { workQueue.add(new SimpleEntry<>(action, tick)); }
```

```
    @SubscribeEvent public void
tick(TickEvent.ServerTickEvent event) { if (event.phase ==
TickEvent.Phase.END) { List<SimpleEntry> actions = new
ArrayList<>(workQueue); actions.forEach(entry -> { Runnable
action = entry.getKey(); int tick = entry.getValue(); if
(tick <= 0) { action.run(); workQueue.remove(entry); } else
{ entry.setValue(tick - 1); } }); } }

    static { LOGGER = LogUtils.getLogger(); workQueue =
new java.util.concurrent.ConcurrentLinkedQueue<>(); } }

    Тут реєструються предмети:

    public class Registry { public static final
DeferredRegister ITEMS =
DeferredRegister.create(ForgeRegistries.ITEMS, "MyMod");

    public static final RegistryObject SWORD1 =
ITEMS.register("sword1", () -> new CustomSwordItem(new
Item.Properties().tab(CreativeModeTab.TAB_COMBAT))); public
static final RegistryObject SWORD2 =
ITEMS.register("sword2", () -> new CustomSwordItem(new
Item.Properties().tab(CreativeModeTab.TAB_COMBAT))); // І
так далі для всіх предметів

    public static void init() {
ITEMS.register(FMLJavaModLoadingContext.get().getModEventBu
s()); } }
```

3.4 Детальний розбір коду та його функції

1. Пакет і анотація @Mod

package dev.hybridlabs.MyMod;: Вказує, що клас належить пакету dev.hybridlabs.MyMod, що є стандартною організацією для Java-коду мода.

@Mod("MyMod"): Анотація Forge, яка позначає цей клас як основний для мода з ідентифікатором MyMod. Це сигналізує грі, що Main – точка входу для завантаження мода.

2. Статичні поля

public static final Logger LOGGER = LogUtils.getLogger();:

Створює логер для запису повідомлень (наприклад, для налагодження).

Логер допомагає відстежувати, чи коректно працює мод.

public static final String MODID = "MyMod";: Зберігає ідентифікатор мода (MyMod), який узгоджується з mods.toml.

3. Поле workQueue

private Collection<SimpleEntry<Runnable, Integer>> workQueue = new java.util.concurrent.ConcurrentLinkedQueue<>().

Це потокобезпечна черга (ConcurrentLinkedQueue), яка зберігає пари SimpleEntry.

Кожен SimpleEntry містить Runnable – об’єкт, що представляє завдання для виконання (наприклад, код для анімації атаки).

Integer – кількість ігрових тіків (одиниць часу в грі, ≈20 тіків = 1 секунда), протягом яких завдання відкладається.

Призначення: Використовується для планування дій, пов’язаних із предметами мода (наприклад, затримка між ударами зброї).

4. Конструктор Main()

IEventBus eventBus = MinecraftForge.EVENT_BUS;: Отримує автобус подій Forge, який обробляє усі події в грі.

eventBus.register(this);: Реєструє клас Main як слухача подій, дозволяючи методам із @SubscribeEvent реагувати на них.

Registry.init();: Викликає метод для ініціалізації реєстрів (ймовірно, де реєструються предмети, блоки чи рендери, включаючи твої JSON-моделі зброї).

Призначення: Ініціалізує мод при завантаженні, підключаючи його до ігрового циклу та реєструючи контент.

5. Метод queueServerWork(int tick, Runnable action)

Параметри:

tick – кількість тіків затримки перед виконанням.

action – завдання типу Runnable (наприклад, код для активації ефекту зброї).

Логіка: Додає нову пару (SimpleEntry) до workQueue, де action асоціюється з tick.

Призначення: Дозволяє відкладати виконання коду в грі. Наприклад, якщо ти створюєш зброю з перезарядкою, це може планувати дію через 20 тіків (1 секунда).

6. Метод tick(TickEvent.ServerTickEvent)

@SubscribeEvent: Позначка, яка вказує, що метод слухає подію ServerTickEvent (оновлення сервера).

Умова if (event.phase == TickEvent.Phase.END): Виконується лише наприкінці кожного серверного тику, уникаючи дублювання.

Логіка:

Створює тимчасовий список actions із копії workQueue.

Для кожного елемента в actions:

Отримує action (завдання) і tick (залишок часу).

Якщо tick <= 0, виконує action.run() і видаляє елемент із черги.

Інакше зменшує tick на 1, готуючись до наступного тику.

Призначення: Обробляє заплановані завдання, наприклад, анімацію атаки чи ефект зброї, синхронізуючи їх із ігровим часом.

7. Статичний блок <clinit>

LOGGER = LogUtils.getLogger();: Ініціалізує логер при завантаженні класу.

workQueue = new java.util.concurrent.ConcurrentLinkedQueue<>();: Ініціалізує чергу завдань.

Призначення: Гарантує, що ресурси доступні до першого використання класу.

Що цей код робить у цілому?

Ініціалізація: При запуску мода він реєструє себе в системі подій Forge і викликає Registry.init() для завантаження всіх предметів (зброї) і їхніх моделей.

Планування завдань: Дозволяє відкладати виконання коду (наприклад, ефекти зброї чи анімації) із заданою затримкою через queueServerWork.

Обробка подій: Кожні 50 мілісекунд (1 тик) перевіряє чергу і виконує готові завдання, видаляючи їх після завершення.

3.5 Налаштування Mods.toml

Файл mods.toml – це конфігураційний файл у форматі TOML (Tom's Obvious, Minimal Language), який використовується в модифікаціях для Minecraft на платформі Forge. Він не містить виконуваного коду, але є ключовим для того, щоб мод коректно завантажився, визначив свої залежності та був розпізнаний грою.

```
    modLoader="javafml"   #mandatory   loaderVersion="[47,)"
#mandatory     license="ARR     (Assets),     MIT     (Code)"
#issueTrackerURL="https://change.me.to.your.issue.tracker.e
xample.invalid/" #optional [[mods]] #mandatory modId="MyMod"
#mandatory version="1.12.2" #mandatory displayName="Tugkan's
Weaponry                  Mod"                  #mandatory
#updateJSONURL="https://change.me.example.invalid/updates.j
son"                                                #optional
#displayURL="https://change.me.to.your.mods.homepage.exampl
e.invalid/"  #optional  #logoFile="examplemod.png"  #optional
authors="TugkanDeMan,Ivan-Khar"                     #optional
displayTest="MATCH_VERSION"              description='''''
[[dependencies.MyMod]]  #optional  modId="forge"  #mandatory
mandatory=true  #mandatory  versionRange="[47,)"  #mandatory
ordering="NONE"      side="BOTH"      [[dependencies.MyMod]]
modId="minecraft"                                 mandatory=true
versionRange="[1.20.1,1.21)"   ordering="NONE"   side="BOTH"
[[dependencies.MyMod]]     #optional     modId="bettercombat"
mandatory=true  #mandatory  versionRange="[1.7,)"  #mandatory
ordering="AFTER"        side="BOTH"        #[features.MyMod]
#openGLVersion="[3.2,)"
```

Детальний розбір:

1. Глобальні налаштування

modLoader="javafml": Вказує, що мод використовує Forge Mod Loader (FML), частину платформи Forge, яка забезпечує завантаження модів у Minecraft.

loaderVersion="[47,)": Визначає діапазон версій Forge, сумісних із модом (від 47 включно до найновіших). Версія 47 відповідає Forge для Minecraft 1.20.1 і вище.

license="ARR (Assets), MIT (Code)": Вказує ліцензії: ARR для активів (наприклад, JSON-моделі, текстури) і MIT для коду. Це важливо для юридичної документації, зокрема для диплома.

2. Секція [[mods]] (опис мода)

modId="MyMod": Унікальний ідентифікатор мода, який збігається з анотацією @Mod("MyMod") у Main.java. Це ключ, за яким гра розпізнає твій мод.

version="ForStudy": Версія мода.

displayName="My Mod": Назва мода, яка відображається в грі (наприклад, у списку модів).

authors="Me": Вказує авторів (тебе й, ймовірно, співавтора), що є важливим для визнання внеску в дипломі.

displayTest="MATCH_VERSION": Вказує, що версії мода на клієнті й сервері мають збігатися. Це типово для модів із клієнтськими та серверними компонентами, як твоя зброя.

description="""": Опис , у нашому випадку пустий.

Необхідні, але закоментовані поля (наприклад, issueTrackerURL, logoFile) можна додати для повноти (наприклад, логотип MyMod_logo.png).

3. Секція [[dependencies.MyMod]] (залежності)

Залежність від Forge:

modId="forge", versionRange="[47,)", ordering="NONE", side="BOTH": Мод залежить від Forge версії 47 або вище, і ця залежність діє як для клієнта, так і для сервера.

Залежність від Minecraft:

modId="minecraft", versionRange="[1.20.1,1.21)", ordering="NONE", side="BOTH": Мод сумісний із Minecraft від 1.20.1 до 1.21 (не включно).

Залежність від BetterCombat:

modId="bettercombat", versionRange="[1.7,)", ordering="AFTER", side="BOTH": Мод залежить від мода BetterCombat версії 1.7 або вище і завантажується після нього. Це вказує на інтеграцію з бойовими механіками.

Призначення: Забезпечує, що мод працюватиме лише за наявності цих залежностей, уникаючи збоїв.

4. Секція [features.MyMod] (додаткові вимоги)

Зараз закоментована (#openGLVersion="[3.2,)"), але може вказувати на мінімальні вимоги до графіки (наприклад, OpenGL 3.2). Це необов’язково, але може бути додано для оптимізації.

Що робить mods.toml у цілому?

Ідентифікація мода: Повідомляє Forge про існування мода MyMod, його назву та версію.

Сумісність: Визначає, з якими версіями Forge, Minecraft і BetterCombat мод сумісний, запобігаючи конфліктам.

Завантаження: Забезпечує правильний порядок завантаження (наприклад, після BetterCombat) і перевірку версій (MATCH_VERSION).

Документація: Надає метадані (авторів, ліцензію), які корисні для користувачів.

Зв’язок із Main.java

modId="MyMod" узгоджується з @Mod("MyMod"): Ідентификатор у mods.toml збігається з анотацією в Main.java, що дозволяє Forge пов’язати конфігурацію з класом.

Завантаження: mods.toml визначає умови (версії, залежності), а Main() у Main.java виконує ініціалізацію (наприклад, Registry.init()), реєструючи предмети.

Події: displayTest="MATCH_VERSION" вимагає синхронізації, а tick у Main.java забезпечує серверну логіку, що працює в обох середовищах (клієнт і сервер).

Залежності: BetterCombat, зазначений у mods.toml, може впливати на поведінку предметів, яку queueServerWork і tick обробляють.

3.6 Файл локалізації en_us.json

Цей файл – це таблиця локалізації у форматі JSON, яка зберігається зазвичай у папці src/main/resources/assets/MyMod/lang/en_us.json. Його основна мета – надавати людинозрозумілі назви та описи для ігрових елементів, доданих твоїм модом, замінюючи технічні ідентифікатори (наприклад, MyMod:wooden_mace) на зручні для гравців тексти (наприклад, "Wooden Mace").

Ключі та значення:

Кожен запис має формат "ключ": "значення", де ключ – це унікальний ідентифікатор із префіксом MyMod (твій Mod ID), а значення – текст, який відображається в грі.

Приклади:

"MyMod.itemgroup.weapons": "TugkanDeMan's Weaponry" – назва групи предметів для зброї.

"item.MyMod.wooden_club_mace": "Wooden Club" – назва предмета "Wooden Club".

"item.MyMod.the_bloodthirster_claymore.description": "Absorbs the enemy health in a pinch" – опис для предмета "The Bloodthirster".

Категорії:

Групи предметів:

"MyMod.itemgroup.weapons": "TugkanDeMan's Weaponry" – групує зброю в творчому режимі.

"MyMod.itemgroup.armors": "TugkanDeMan's Armory" – групує броню.

Зброя:

Мечис (наприклад, "item.MyMod.katana": "Katana", "item.MyMod.the_bloodthirster_claymore": "The Bloodthirster").

Булави та молоти (наприклад, "item.MyMod.wooden_mace": "Wooden Mace", "item.MyMod.steel_hammer": "Steel Hammer").

Кулаки/рукавиці (наприклад, "item.MyMod.terrapotta_fist": "Terrapotta Fist").

Ножі (наприклад, "item.MyMod.netherite_knife": "Netherite Knife").

Броня:

Комплект Brightsteel (шолом, нагрудник тощо).

Інструменти:

Кирки, лопати, сокири (наприклад, "item.MyMod.copper_pickaxe": "Copper Pickaxe").

Блоки та матеріали:

Блоки (наприклад, "block.MyMod.ancient_debris": "Ancient Debris").

Інготи та уламки (наприклад, "item.MyMod.titanite_ingot": "Titanite Ingot").

Зачарування:

"enchantment.MyMod.levitating": "Levitating" – нове зачарування.

Описи:

Деякі предмети мають додаткові описи (наприклад, "item.MyMod.katana.description": "Grants the strength of the wind"), які відображаються в підказках (tooltip).

Локалізація: Коли гравець наводить курсор на предмет у грі, Minecraft шукає відповідний ключ у файлі локалізації (наприклад, item.MyMod.wooden_mace) і відображає значення ("Wooden Mace").

Групи предметів: Ключі типу MyMod.itemgroup.weapons визначають вкладки у творчому режимі, де сортуються предмети.

Інтеграція: Файл зв’язаний із кодом через реєстрацію предметів у Registry.init() (з Main.java), де кожному предмету присвоюється ID, який збігається з ключами в JSON.

3.7 Приклад коду предмету Katana

Це один з наших предметів створений у BlockBench. Цей JSON-файл – це опис 3D-моделі предмета "Katana" для Minecraft, який використовується для рендерингу в грі.


```
{
  "credit": "Made with Blockbench",
  "texture_size": [32, 32],
  "textures": {
    "0": "MyMod:item/katana",
    "particle": "MyMod:item/katana"
  },
  "elements": [
    {
      "from": [6.5, -12, 7.5],
      "to": [9.5, 20, 8.5],
      "rotation": {"angle": 0, "axis": "y", "origin": [7.50191, 6.625, 8]},
      "faces": {
        "north": {"uv": [1.5, 0, 0, 16], "texture": "#0"},
        "east": {"uv": [1.5, 0, 1, 16], "texture": "#0"},
        "south": {"uv": [0, 0, 1.5, 16], "texture": "#0"},
        "west": {"uv": [0.5, 0, 0, 16], "texture": "#0"},
        "up": {"uv": [7, 5.5, 8.5, 5], "texture": "#0"},
        "down": {"uv": [7, 5.5, 8.5, 6], "texture": "#0"}
      }
    },
    {
      "from": [6.5, -14, 7.5],
      "to": [7.5, -12, 8.5],
      "rotation": {"angle": 0, "axis": "y", "origin": [7.50191, 6.625, 8]},
      "faces": {
        "north": {"uv": [8.5, 0, 8, 1], "texture": "#0"},
        "east": {"uv": [8.5, 3, 8, 4], "texture": "#0"},
        "south": {"uv": [8.5, 2, 8, 3], "texture": "#0"},
        "west": {"uv": [8.5, 1, 8, 2], "texture": "#0"},
        "up": {"uv": [8, 6.5, 8.5, 6], "texture": "#0"},
        "down": {"uv": [8, 6.5, 8.5, 7], "texture": "#0"}
      }
    },
    {
      "from": [20.68364, -7.96143, 7.499],
      "to": [22.08564, -5.13443, 8.501],
```

```
        "rotation": {"angle": -45, "axis": "z", "origin": [7.50191, 6.625, 8]},
        "faces": {
            "north": {"uv": [6.5, 7, 6, 8.5], "texture": "#0"},
            "east": {"uv": [6.5, 7, 6, 8.5], "texture": "#0"},
            "south": {"uv": [6.5, 7, 6, 8.5], "texture": "#0"},
            "west": {"uv": [7, 7, 6.5, 8.5], "texture": "#0"},
            "up": {"uv": [8, 7.5, 8.5, 7], "texture": "#0"},
            "down": {"uv": [8, 7.5, 8.5, 8], "texture": "#0"}
        }
    },
    {
        "from": [7.10411, 21, 7.60221],
        "to": [8.60411, 31.5, 9.10221],
        "rotation": {"angle": 45, "axis": "y", "origin": [7.50191, 6.625, 8]},
        "faces": {
            "north": {"uv": [5, 0, 4, 5.5], "texture": "#0"},
            "east": {"uv": [6, 5.5, 5, 11], "texture": "#0"},
            "south": {"uv": [5, 5.5, 4, 11], "texture": "#0"},
            "west": {"uv": [6, 0, 5, 5.5], "texture": "#0"},
            "up": {"uv": [6, 5, 7, 4], "texture": "#0"},
            "down": {"uv": [6, 5, 7, 6], "texture": "#0"}
        }
    },
    {
        "from": [5.85411, 20, 6.35221],
        "to": [9.85411, 21, 10.35221],
        "rotation": {"angle": 45, "axis": "y", "origin": [7.50191, 6.625, 8]},
        "faces": {
            "north": {"uv": [12.5, 6, 10, 6.5], "texture": "#0"},
            "east": {"uv": [12.5, 5.5, 10, 6], "texture": "#0"},
            "south": {"uv": [12.5, 5, 10, 5.5], "texture": "#0"},
            "west": {"uv": [12.5, 6.5, 10, 7], "texture": "#0"},
            "up": {"uv": [10, 2.5, 12.5, 0], "texture": "#0"},
            "down": {"uv": [10, 2.5, 12.5, 5], "texture": "#0"}
        }
    }
],
"display": {
    "thirdperson_righthand": {
        "rotation": [180, -90, 0],
        "translation": [0, 10, 1.5],
        "scale": [0.75, 0.75, 0.75]
    },
    "thirdperson_lefthand": {
```

```
            "rotation": [-15, -90, 0],
            "translation": [-1.75, -12.5, 2.5],
            "scale": [0.75, 0.75, 0.75]
        },
        "firstperson_righthand": {
            "rotation": [145, -90, 0],
            "translation": [0, 6.25, -1.5],
            "scale": [0.63, 0.63, 0.63]
        },
        "firstperson_lefthand": {
            "rotation": [-104.9, -76.72, -29.08],
            "translation": [1, -4, 10],
            "scale": [0.81, 0.81, 0.81]
        },
        "ground": {
            "rotation": [0, 0, 150],
            "translation": [0, 9, 0],
            "scale": [0.6, 0.6, 0.6]
        },
        "gui": {
            "rotation": [-30, -15, 135],
            "translation": [0.25, 0.25, 0],
            "scale": [0.45, 0.45, 0.45]
        },
        "fixed": {
            "translation": [0, -15, 0],
            "scale": [1.5, 1.5, 1.5]
        }
    }
}
```


Він визначає геометрію моделі, текстури, ротацію та позиціювання в різних ігрових ситуаціях (наприклад, у руках гравця чи на землі). Проаналізуємо код докладніше:

- Структура та вміст.
- Метадані.

credit": "Made with Blockbench": Вказує, що модель створена в Blockbench, популярному інструменті для моделювання в Minecraft.

texture_size": [32, 32]: Розмір текстури моделі (32x32 пікселі), що є стандартним для предметів у Minecraft.

Текстури

**textures":

"0": "MyMod:item/katana" – основна текстура моделі, взята з assets/MyMod/textures/item/katana.png.

"particle": "MyMod:item/katana" – текстура для частинок (наприклад, ефектів руйнування), також використовує katana.png.

Елементи моделі (elements)

Модель складається з п’яти елементів (кубоїдів), кожен із яких описує частину катани (лезо, рукоятка тощо):

Елемент 1: [6.5, -12, 7.5] до [9.5, 20, 8.5] – основна частина леза, із ротацією навколо осі Y і текстурою #0 (katana.png).

Елемент 2: [6.5, -14, 7.5] до [7.5, -12, 8.5] – верхівка леза.

Елемент 3: [20.68364, -7.96143, 7.499] до [22.08564, -5.13443, 8.501] – частина з ротацією -45° навколо осі Z (можливо, гарда).

Елемент 4: [7.10411, 21, 7.60221] до [8.60411, 31.5, 9.10221] – рукоятка з ротацією 45° навколо осі Y.

Елемент 5: [5.85411, 20, 6.35221] до [9.85411, 21, 10.35221] – основа рукоятки.

Кожен елемент має faces (північ, схід, південь тощо) із UV-координатами, які визначають, як текстура накладається на геометрію.

3.8 Створені класи

Скріншоти показують дерево проєкту з кореневим пакетом MyMod, який є моїм робочим простором створеним у IntelliJ IDEA. Пакет client містить клієнтські ресурси, які відповідають за рендеринг і інтерфейс.models:BrightsteelArmorModelBoots.class, BrightsteelArmorModelChestplate.class, тощо.

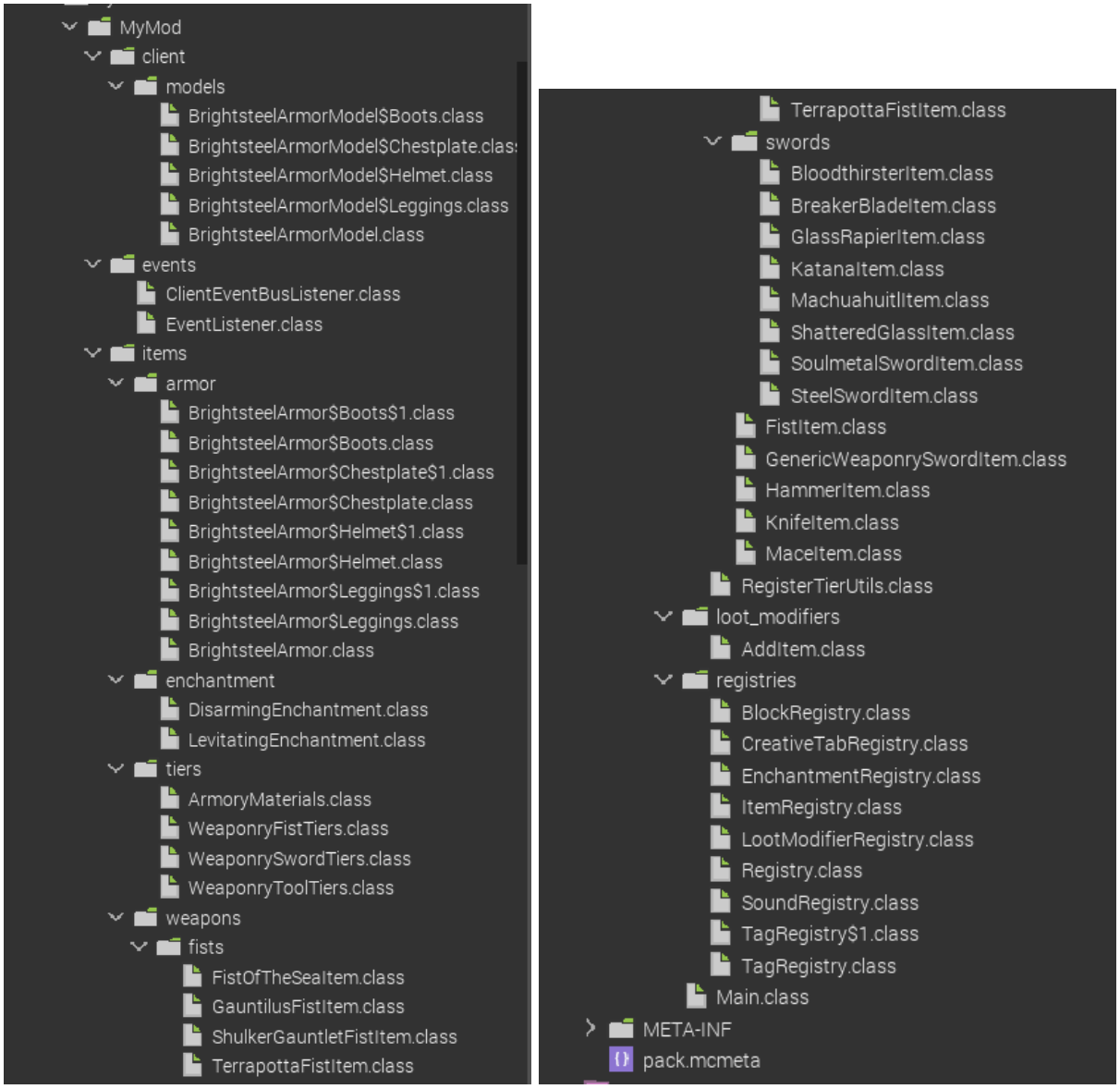


Рисунок 2.4 – Структура класів модифікації

Класи моделей для броні Brightsteel (шолом, нагрудник, легінги, чоботи). Вони визначають 3D-моделі для рендерингу броні в грі, аналогічно до JSON-моделі катани, яку ти надіслав раніше. Ці класи реалізують кастомний рендеринг броні, використовуючи API Forge (наприклад, ModelLoader).

Пакет events містить обробники подій.

ClientEventBusListener.class: Слухає клієнтські події (наприклад, рендеринг або завантаження текстур).

EventListener.class: Загальний слухач подій, можливо, для серверних чи обох типів подій.

Призначення: Ці класи інтегрують логіку мода з ігровим циклом, подібно до tick у Main.java.

Пакет items містить класи предметів, доданих модом.

armor:

BrightsteelArmorBoots$1.class, BrightsteelArmorChestplate$1.class, тощо: Внутрішні класи або версії броні Brightsteel.

BrightsteelArmor.class: Базовий клас для броні Brightsteel, ймовірно, успадкований від ArmorItem.

enchantment:

DisarmingEnchantment.class, LevitatingEnchantment.class: Класи для кастомних зачарувань, згадані в локалізації (наприклад, "Levitating").

tiers:

ArmoryMaterials.class, WeaponryFistTiers.class, WeaponrySwordTiers.class, WeaponryToolTiers.class: Класи, що визначають матеріали та рівні предметів (наприклад, витривалість, шкода), використовувані в Tier Forge.

weapons:

fists:

FistOfTheSeaItem.class, GauntilusFistItem.class, ShulkerGauntletFistItem.class, TerrapottaFistItem.class: Класи для кулаків/рукавиць, згадані в локалізації.

swords:

BloodthirsterItem.class, BreakerBladeItem.class, GlassRapierItem.class, KatanaItem.class, MachuahuitlItem.class, ShatteredGlassItem.class, SoulmetalSwordItem.class, SteelSwordItem.class, FistItem.class, GenericWeaponrySwordItem.class, HammerItem.class, KnifeItem.class, MaceItem.class, RegisterTiersUtil.class: Класи для мечів, молотів, ножів тощо, включаючи катану.

Ці класи реалізують логіку предметів (наприклад, атаки, ефекти), а їхні моделі (наприклад, katana.json) визначають зовнішній вигляд.

Пакет loot_modifiers

AddItem.class: Клас для модифікації луту, додає предмети (наприклад, "Titanite Ingot") у випадкові дропи.

Пакет registries

Містить класи для реєстрації вмісту мода.

BlockRegistry.class, CreativeTabRegistry.class, EnchantmentRegistry.class, ItemRegistry.class, LootModifierRegistry.class, Registry.class, SoundRegistry.class, TagRegistry$1.class, TagRegistry.class, MainRegistry.class: Класи для реєстрації блоків, предметів, зачарувань, луту, тегів і звуків.

Призначення: Registry.class – ключовий, викликаний у Main.java через Registry.init(), де реєструються всі предмети, блоки та інше.

Файл Main.class

Основний клас мода, аналогічний Main.java, який ми аналізували. Він ініціалізує мод і запускає реєстрацію.

3.9 Пакування моду

Щоб перевірити наш мод , ми повинні використати команди gradl. Gradle – це система автоматизації побудови (build automation system), яка використовується для компіляції, тестування, упаковки та розгортання програмного забезпечення, включаючи моди для Minecraft. Це відкрите програмне забезпечення, написане на Groovy та Kotlin, яке стало популярним завдяки своїй гнучкості, швидкості та інтеграції з різними інструментами розробки, такими як IntelliJ IDEA чи Eclipse. Gradle автоматично завантажує і керує бібліотеками. Gradle використовує кешування, щоб уникати повторної компіляції незмінених файлів, що прискорює розробку. Він розбиває процес побудови на окремі задачі, такі як clean (очищення), build (збірка) чи runClient (тестування). Кожна задача може залежати від інших, створюючи ланцюжок дій.

У консолі вводимо команду gradlew build, це збере мод у файл .json. Далі пишемо gradlew runClient це запустить гру якцо код зроблено вірно (рисунок 2.5) (рисунок 2.6) .

```
Executing Gradle task: build
Build info: MCreator 2023.4.52316, forge-1.20.
Loaded APIs: geckolib

> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE
> Task :jar
> Task :downloadMcpConfig
> Task :extractSrg UP-TO-DATE
> Task :createMcpToSrg
> Task :reobfJar
> Task :jarJar SKIPPED
> Task :reobfJarJar SKIPPED
> Task :assemble
> Task :compileTestJava NO-SOURCE
> Task :processTestResources NO-SOURCE
> Task :testClasses UP-TO-DATE
> Task :test NO-SOURCE
> Task :check UP-TO-DATE
> Task :build

BUILD SUCCESSFUL in 10s
7 actionable tasks: 4 executed, 3 up-to-date
Task completed in 10 seconds
```

Рисунок 2.5 – Успішна збірка моду

Рисунок 2.6 – Головне меню гри

3.10 Тест моду у грі

Перевіримо чи бачe гра наш мод. Натискаємо у головному меню кнопку Моди і шукаємо наш мод (рисунок 2.7).

Рисунок 2.7 – Папка модів у грі

Перевіремо тепер наші предмети в грі, створюємо новий світ у режимі креативу (рисунок 2.8).

Рисунок 2.8 – добавлена в гру Катана

Коли наводимсоь на неї курсором , наш опис теж працює (рисунок 2.9)

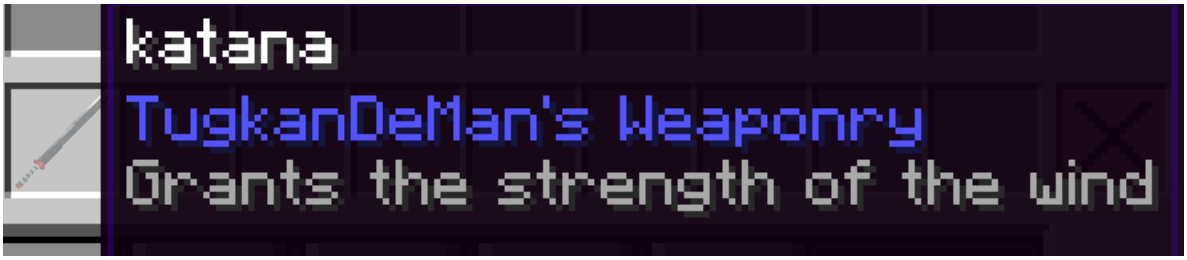


Рисунок 2.9 – Перевірка працездатності опису

А також перевіримо бойову систему, яку описували ще в першому розділі. Ми можемо увімкнути вид від третього лиця та побачити що персонаж тримає меч як самурай (рисунок 2.10). У ванільному майнкрафті мечь тримається по іншому а також ми бачимо ванильну модельку меча (Рисунок 2.11).

Рисунок 2.11 – Не стандартна механіка.

Рисунок 2.12 – Стандартна механіка

Перевіримо також бойову систему (Рисунок 2.13).

Натискаємо ПКМ та бачимо , що у нас зьявились эффекти від меча .

Рисунок 2.13 – Унікальні властивості меча

Висновки по розділу

Мод демонструє значні переваги порівняно з ванільними механіками та предметами. На відміну від багатьох аналогів, що пропонують лише косметичні оновлення чи обмежений асортимент зброї, наш мод вводить унікальну колекцію зброї, зокрема катану, яка поєднує японський стиль із кастомними механіками атаки. Ця зброя, створена з використанням API Forge, включає інноваційні ефекти, такі як "сила вітру", що перевершує стандартні мечі за глибиною ігрового впливу. Нові типи зброї, такі як молоти, кулаки та ножі, разом із інтеграцією з модом BetterCombat, забезпечують унікальні виклики та тактичні можливості, вирізняючи twm серед сучасних збройових модів.

РОЗДІЛ 4 ОХОРОНА ПРАЦІ

4.1 Регулювання питань охорони праці на законодавчому рівні

Охорона праці та виробнича безпека – це галузь, яка в першу чергу фокусується на безпеці, здоров'ї та добробуті працівників на робочому місці [9].

У багатьох частинах світу норми охорони праці вимагають, щоб роботодавці приділяли особливу увагу профілактиці проблем зі здоров'ям та безпекою на робочих місцях.

За останні кілька десятиліть охорона праці та виробнича безпека значно покращилися, що зрештою призвело до позитивних результатів. Навіть високоризикові види діяльності стали безпечнішими, і були введені безпечніші методи експлуатації небезпечного обладнання. У більшості випадків охорона праці та виробнича безпека часто забезпечуються поєднанням правових та виконавчих норм. Роботодавці також вимагають від своїх працівників практикувати саморегулювання.

Коли компанії впроваджують стандарти з охорони праці та виробничої безпеки, це дозволяє працівникам виконувати свої функції у більш безпечному та захищеному середовищі, вільному від будь-яких серйозних небезпек. Без належних стандартів ризик отримання травм працівниками був би значно вищим.

У багатьох випадках роботодавці не почуваються відповідальними за управління безпекою на робочому місці та можуть повністю ігнорувати охорону праці та питання виробничої безпеки. Це може призвести до швидкого зростання рівня виробничого травматизму в організації.

Працівник відіграє ключову роль у запобіганні виробничим травмам і захворюванням. Працівники повинні бути обережні і вживати необхідних запобіжних заходів, щоб забезпечити своє власне здоров'я та безпеку, а також здоров'я та безпеку колег, які можуть опинитися у зоні впливу небезпеки.

Працівники користуються правами, передбаченими законодавством України про охорону праці: право знати, право брати участь та право відмовитися від небезпечної роботи. Також у працівників є обов'язки з охорони свого здоров'я та безпеки, а також здоров'я та безпеки колег.

Працівники мають право бути поінформованим про відомі або прогнозовані небезпеки на робочому місці, а також на отримання інформації, інструкцій, навчання та нагляду, необхідних для захисту свого здоров'я та безпеки.

Реалізація цього положення потребує використання відповідних методів спілкування для всіх працівників, включаючи працівників з особливими потребами. До таких методів належать шрифт Брайля, великий шрифт, аудіозаписи, мова жестів та усне спілкування.

Крім того, працівникам надається право доступу до звітів уряду або роботодавця, які стосуються здоров'я та безпеки співробітників, через профспілку або представника трудового колективу з охорони праці.

Працівники мають право брати участь у виявленні та усуненні проблем, пов'язаних з охороною праці та виробничою безпекою.

Працівники мають право відмовитися від роботи, якщо у них є підстави вважати, що:

- робоче місце становить для них небезпеку;

- використання або експлуатація машини або устаткування становить небезпеку для них чи іншого співробітника;

- виконання діяльності становить небезпеку для них чи іншого співробітника.

Працівник відповідно до законодавства України про охорону праці зобов’язаний:

- використовувати всі захисні обладнання, пристрої та одяг, які надаються роботодавцем та призначені для захисту працівників;

- дотримуватись процедур, що стосуються здоров'я та безпеки співробітників;

- дотримуватись усіх інструкцій роботодавця щодо охорони праці та безпеки процесів;

- повідомляти роботодавця про будь-які речі або обставини, які можуть становити небезпеку для працівників або будь-якої іншої особи на робочому місці;

- повідомляти роботодавця про всі нещасні випадки на виробництві, професійні захворювання або інші небезпечні події, які призвели до травми.

4.2 Виявлення потенційних небезпек стосовно об’єкту проектування

У дипломній роботі розглянута розробка модифікації для гри Minecraft мовою Java з використанням API Forge. Розробкою займаються працівники IT0-індустрії, чиє робоче місце може представляти собою типовий офіс.

Аналіз умов праці розробника у сфері інформаційних технологій передбачає розгляд специфіки їхньої діяльності, виявлення небезпечних та шкідливих виробничих факторів, а також оцінку потенційних наслідків їхнього впливу на здоров'я та працездатність [10].

Загальна характеристика умов праці IT-фахівця:

- місце роботи: найчастіше офісне приміщення чи віддалений (домашній) формат;

- тип праці: розумовий, з тривалою роботою за комп'ютером;

- тривалість та режим роботи: 8 та більше годин на день, можливі понаднормові години роботи, особливо у період дедлайнів.

Таблиця 3.1 – Аналіз умов праці на робочому місці розробника

Фактор	Опис впливу	Наслідки впливу
Одноманітна поза (сидяча)	Тривале перебування за робочим столом	Порушення постави, остеохондроз, варикоз, гіподинамія
Освітленість	Недостатня або надмірна яскравість екрану та приміщення	Втома очей, головний біль, погіршення зору
Шум	Робота в open-space, шум вентиляції, офісної техніки	Хронічний стрес, зниження концентрації
Температурний режим	Кондиціювання, протяги, перегрів ноутбука	Простудні захворювання, дискомфорт
Виділення від оргтехніки	Лазерні принтери, старі комп'ютери можуть виділяти озон, формальдегіди	Алергічні реакції, подразнення слизових
Біологічні фактори (відсутні безпосередньо)	Можлива дія у разі неякісного прибирання приміщення або контролю стану систем кондиціювання	Алергени, спори грибків, пліснява
Високе когнітивне навантаження	Постійне прийняття рішень, концентрація, програмування	Розумова перевтома, виснаження
Жорсткі терміни, дедлайни	Тиск замовників/менеджерів проекту	Стрес, тривожність, вигоряння
Нерегламентований робочий день	Робочі дзвінки/листи у неробочий час	Порушення work-life balance, хронічна втома
Обмежене живе спілкування (особливо при віддаленій роботі)	Ізоляція, відчуття самотності	Соціальна дезадаптація, депресія

Можливі наслідки тривалого впливу зазначених факторів:

- порушення зору (астенопія, синдром сухого ока);

- проблеми з опорно-руховим апаратом (спина, шия, кисті рук – синдром зап'ясткового каналу);

- підвищений ризик серцево-судинних захворювань (через гіподинамію).

- синдром професійного вигоряння (burnout);

- тривожні розлади, депресія;

- порушення сну, зниження мотивації.

4.3 Дослідження ризику реалізації потенційних небезпек на об’єкті проектування та розробка заходів щодо їх попередження

Оцінка ризиків є важливим аспектом забезпечення та підтримання безпеки на робочому місці, оскільки вона допомагає виявляти та контролювати потенційні небезпеки, які можуть завдати шкоди співробітникам. Це проактивний підхід до запобігання нещасним випадкам та підтримці здорового робочого середовища.

Оцінка ризику зазвичай включає такі етапи:

- визначення небезпеки: потрібно оглянути робоче місце та виявити потенційні джерела шкоди, такі як обладнання, хімічні речовини або робочі процеси;

- визначення переліку осіб, які можуть наражатись на небезпеки: потрібно визначити працівників та інших осіб, які можуть зазнавати впливу виявлених небезпек;

- оцінка ризиків: оцінка ймовірності і серйозності шкоди, заподіяної кожною небезпекою, та розставлення пріоритетів на основі рівня ризику;

- реалізація заходи контролю: розробка та впровадження відповідних стратегій для мінімізації чи усунення виявлених ризиків;

- моніторинг та огляд: регулярна оцінка ефективності заходів контролю та оновлення оцінки ризику за необхідності.

Результати оцінки ризиків слід використовувати при розробці ініціатив щодо забезпечення безпеки на робочому місці:

- розробка плану безпеки: можна використовувати результати оцінки ризиків для створення комплексного плану безпеки, в якому будуть розглянуті виявлені небезпеки та описані необхідні заходи контролю;

- навчання працівників: співробітники повинні бути поінформовані про небезпеки, які є на їхньому робочому місці, та про відповідні заходи, які слід вжити для мінімізації ризику;

- взаємодія зі співробітниками: інформування співробітників про результати оцінки ризиків та заохочення їх участі у поточних зусиллях щодо забезпечення безпеки;

- постійне вдосконалення: можна використовувати оцінку ризику як відправну точку для постійного покращення стану безпеки на робочому місці та регулярно переглядати й оновлювати оцінку, щоб відобразити зміни на робочому місці та виявити нові небезпеки.

Таблиця 3.2 – Результат використання матриці оцінки ризиків

Робота в одноманітній позі				
Визначення категорії серйозності небезпеки		Визначення рівня ймовірності небезпеки		Індекс ризику небезпеки
Вид, категорія	Опис	Вид, рівень	Опис	
II – критична	Порушення постави, остеохондроз, варикоз, гіподинамія	А – часта подія	Робота в одноманітній позі часто зустрічається у досліджуваному типі робіт	2А – неприпустимий (надмірний) рівень ризику

Для управління фактором монотонної (сидячої) пози розробника розумно застосовувати комплекс заходів, спрямованих на мінімізацію статичного навантаження, підтримання правильної постави та включення руху в робочий день.

Таблиця 3.3 – Рекомендації із управління фактором одноманітної пози

Елемент	Рекомендації
Крісло	Регульована спинка та підлокітники, підтримка попереку, висота – так, щоб стопи стояли на підлозі
Стіл	Рівень ліктів має збігатися з рівнем поверхні столу
Монітор	Верхня межа екрану на рівні очей, відстань – 50-70 см
Підставка під ноги	При необхідності – для зняття навантаження з ніг
Метод	Зміст
Правило 20–20–20	Кожні 20 хвилин дивитися на 20 секунд на об'єкт на відстані 20 футів (~6 м)
Метод “помідора” (Pomodoro)	25 хвилин роботи, а потім 5 хвилин перерва. Після 4 циклів – довга перерва (15-30 хвилин)
Розминки	Встати, пройтися, зробити розминку для шиї, плечей, спини та ніг кожні 1–2 години
Інструмент	Приклад
Таймери та нагадування	Додатки: Stretchly, Workrave, Time Out (macOS)
Смарт-годинники/браслети	Нагадують рухатися при тривалому сидінні
Змінні пози	Альтернація сидячого та стоячого положення – за допомогою регульованого столу

Фактор монотонної пози не слід недооцінювати, оскільки він може призвести до серйозних проблем зі здоров'ям. Однак він піддається ефективному управлінню, якщо впровадити в робочі процеси прості звички, покращити ергономіку та підвищити загальну рухову активність працівників.

Висновки по розділу

У розділі з охорони праці приділено увагу аналізу ролі роботодавця та працівника організації у досягненні спільних зусиль із забезпечення безпечного робочого середовища та відповідних вимогам законодавства вимог з охорони праці.

Окрім того, у розділі проведено аналіз умов праці розробника як представника ІТ-індустрії, визначено основні небезпечні та шкідливі виробничі фактори, які впливають на працівників, наслідки їх впливу.

На основі отриманих результатів у розділі проведено оцінку ризику однієї з найбільш розповсюджених небезпек – робота в одноманітній позі – та наведено ряд рекомендацій, дотримання яких дозволить підвищити рівень безпеки праці працівника.

ВИСНОВКИ

Мод демонструє значні переваги порівняно з ванільними механіками та предметами. На відміну від багатьох аналогів, що пропонують лише косметичні оновлення чи обмежений асортимент зброї, наш мод вводить унікальну колекцію зброї, зокрема катану, яка поєднує японський стиль із кастомними механіками атаки. Ця зброя, створена з використанням API Forge, включає інноваційні ефекти, такі як "сила вітру", що перевершує стандартні мечі за глибиною ігрового впливу. Нові типи зброї, такі як молоти, кулаки та ножі, разом із інтеграцією з модом BetterCombat, забезпечують унікальні виклики та тактичні можливості, вирізняючи twm серед сучасних збройових модів.

Наш мод вирізняється високим рівнем новизни завдяки інтеграції катани та інших кастомних предметів, які не просто розширюють арсенал, а додають оригінальні механіки взаємодії. Катана, розроблена з унікальною 3D-моделлю та анімаціями, створеними в Blockbench, пропонує гравцям нові способи ведення бою, включаючи спеціальні ефекти та затримки атаки, реалізовані через Java-код. Введення нових матеріалів (наприклад, Brightsteel) і зачарувань (наприклад, Levitating) забезпечує безпрецедентні можливості для кастомізації та стратегії, що є значним кроком уперед у розвитку збройових модифікацій для Minecraft.

Практичне значення полягає в здатності оживляти ігровий процес Minecraft, додаючи різноманітність у бої та залучаючи як досвідчених, так і нових гравців. Включення катани та інших унікальних предметів забезпечує нові способи дослідження та творчості, подовжуючи інтерес до гри. Сумісність із BetterCombat і стабільна продуктивність, досягнута завдяки оптимізованому коду, роблять мод цінним доповненням для спільноти, яке не порушує загальної стабільності гри та легко інтегрується з іншими модифікаціями.

Подальші оновлення можуть включати розширення асортименту зброї з новими механіками (наприклад, комбіновані атаки чи ефекти отруєння), а також додавання кастомних броньових наборів із динамічними властивостями. Інтеграція інтерактивних елементів, таких як унікальні вороги чи квести, пов’язані з катанами

та іншими предметами, може ще більше збагатити ігровий досвід. Крім того, створення детальної документації та навчальних матеріалів сприятиме залученню нових розробників до спільноти моддерів Minecraft.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mojang Studios. *Minecraft: Official Website*. – [Електронний ресурс] - Режим доступу: https://www.minecraft.net

2. GitHub – MinecraftForge. *Minecraft Forge Source Code*. – [Електронний ресурс] - Режим доступу: [Електронний ресурс] - Режим доступу: https://github.com/MinecraftForge/MinecraftForge

3. Blockbench. *3D Modeling Tool for Minecraft and More*. – [Електронний ресурс] - Режим доступу: https://www.blockbench.net

4. How to start modding - Modder Support [Електронний ресурс] - Режим доступу: https://forums.minecraftforge.net/topic/78290-how-to-start-modding/

5. Forge Documentation [Електронний ресурс] - Режим доступу: https://mcforge.readthedocs.io/.

6. Cubicoder's Minecraft Forge Tutorials [Електронний ресурс] - Режим доступу: https://cubicoder.wordpress.com/

7. Jabelar's Minecraft Forge Modding Tutorials: Welcome [Електронний ресурс] - Режим доступу: http://jabelarminecraft.blogspot.com/2014/06/welcome.html

8. Gradle User Guide [Електронний ресурс] - Режим доступу: https://docs.gradle.org.

9. Modding-Resources/dev_pins.md at master [Електронний ресурс] - Режим доступу: https://github.com/MinecraftModDevelopment/Modding-Resources/blob/master/dev_pins.md

10. Programmers Wanted For Helpful Villagers Mod [Електронний ресурс] - Режим доступу: https://www.minecraftforum.net/forums/mapping-and-modding-java-edition/minecraft-mods/mods-discussion/2527538-programmers-wanted-for-helpful-villagers-mod

11. Закон України «Про охорону праці». [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/laws/show/2694-12.

12. Safety Statement and Risk Assessment. – Режим доступу: https://www.hsa.ie/eng/topics/managing_health_and_safety/safety_statement_and_risk_assessment/.

13. Наказ «Про затвердження Класифікаційних ознак надзвичайних ситуацій» затверджений Міністерством внутрішніх справ України від 06.08.2018 № 658 та зареєстрований Міністерством юстиції України від 28 серпня 2018 р. № 969/32421 [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/laws/show/z0969-18#Text

14. Порядок розслідування та обліку нещасних випадків, професійних захворювань та аварій на виробництві затверджений постановою Кабінету Міністрів України від 17 квітня 2019 року №337 [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/laws/show/337-2019-%D0%BF#Text

15. Кодекс цивільного захисту України [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/laws/show/5403-17#Text