Міністерство освіти і науки України

Національний технічний університет

«Дніпровська політехніка»

Інститут електроенергетики

Факультет інформаційних технологій

Кафедра Інформаційних технологій та комп'ютерної інженерії

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційна робота ступеня бакалавра

Студента Сердюка Юрія Віталійовича

академічної групи 126-20-1

спеціальності 126 «Інформаційні системи та технології»

за освітньо-професійною програмою «Інформаційні системи та технології»

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

Керівник	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	доц. Сергєєва К.Л.			
розділів:				

Рецензент				

Нормоконтролер	проф. Коротенко Г.М.			

Дніпро
2024

ЗАТВЕРДЖЕНО:
завідувач кафедри
Інформаційні технології
та комп'ютерної інженерії

_____ _____
(підпис) (прізвище)
«___» _____ 20 ___ року

ЗАВДАННА

на кваліфікаційну роботу

ступеня бакалавра

студенту Сердюк Ю.В. **академічної групи** 126-20-1

спеціальності 126 «Інформаційні системи та технології»

за освітньо-професійною програмою «Інформаційні системи та технології»

на тему «Розробка модифікації для гри Minecraft мовою Java з використанням API Fabric

затверджену наказом ректора НТУ «Дніпровська політехніка» від 23.05.2024 № № 469-с

Розділ	**Зміст**	**Термін виконання**
Розділ 1	Огляд гри Minecraft та його можливості	
Розділ 2	Проект модифікації для Minecraft	

Завдання видано _____ Сергєєва К. Л.
(підпис керівника) (прізвище, ініціали)

Дата видачі _____

Дата подання до екзаменаційної комісії _____

Прийнято до виконання _____ Сердюк Ю. В.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 58 сторінок, 21 ілюстрацій, 1 додаток, 6 джерел.

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

Мета роботи: мета цього проекту — покращити ігровий досвід Minecraft шляхом створення комплексної модифікації, яка вводить нові виміри з унікальними біомами, ресурсами та елементами ігрового процесу. Цей мод спрямований на стимулювання дослідження, творчості та стратегічного геймплея.

Вступ: розробка мода Mystic Dimensions відповідає зростаючому попиту в спільноті Minecraft на свіжий та інноваційний контент. Оскільки гравці постійно шукають нові виклики та середовища, цей мод представляє різні виміри, які збагачують ігровий процес, пропонуючи різноманітні біоми та унікальні пригоди. Це вдосконалення пожвавлює інтерес серед давніх гравців і залучає нову аудиторію, гарантуючи, що Minecraft залишається привабливим. Розширюючи ігровий всесвіт, Mystic Dimensions робить внесок у ширшу модифікаційну спільноту, демонструючи потенціал контенту, створеного користувачами, для значного подовження терміну служби та задоволення від популярних ігор.

Розділ 1: перший розділ містить огляд поточного стану Minecraft та існуючих обмежень у грі, таких як відсутність різноманітності біомів, структур і ресурсів. Це підкреслює потребу в новому контенті для підтримки інтересу гравців і покращення ігрового досвіду.

Розділ 2: У другому розділі детально описано дизайн і реалізацію мода Mystic Dimensions. Він охоплює створення унікальних біомів із чіткими візуальними та тематичними елементами, створення нових вимірів і розробку власних ресурсів і предметів. Мод спрямований на бездоганну інтеграцію з існуючою механікою Minecraft, пропонуючи нові враження від гри.

Практична значущість: практична значущість роботи полягає в її внеску в спільноту модифікаторів Minecraft, надаючи комплексну модифікацію, яка покращує ігровий досвід. Розроблене технологічне рішення може бути реалізоване в електронних бізнес-середовищах (електронна комерція та інші суб’єкти господарювання).

Ключові слова: МІСТИЧНІ ВИМІРИ, МОДИНГ, БІОМИ, РЕСУРСИ, ГЕЙМПЛЕЙ, СУМІСНІСТЬ, ОПТИМІЗАЦІЯ, MINECRAFT, ДОСЛІДЖЕННЯ, ТВОРЧІСТЬ, МОДИФІКАЦІЯ, ГЕНЕРАЦІЯ РЕЛЬЄФУ.

ABSTRACT

The explanatory note: 58 pages, 21 illustrations, 1 appendice, 6 sources.

Object of development: The object of development is the Mystic Dimensions modification for Minecraft, which aims to expand the game's universe by introducing new dimensions, biomes, resources, and challenges.

Objective of the work: The goal of this project is to enhance the Minecraft gaming experience by creating a comprehensive modification that introduces new dimensions with unique biomes, resources, and gameplay elements. This mod aims to stimulate exploration, creativity, and strategic gameplay.

Introduction: The development of the Mystic Dimensions mod addresses a growing demand within the Minecraft community for fresh and innovative content. As players continually seek new challenges and environments, this mod introduces distinct dimensions that enrich gameplay, offering diverse biomes and unique adventures. This enhancement revitalizes interest among long-time players and attracts new audiences, ensuring Minecraft remains engaging. By expanding the game's universe, Mystic Dimensions contributes to the broader modding community, showcasing the potential of user-generated content to significantly extend the lifespan and enjoyment of popular games.

Chapter 1: The first chapter provides an overview of the current state of Minecraft and the existing limitations within the game, such as the lack of variety in biomes, structures, and resources. It highlights the need for new content to maintain player interest and enhance the gaming experience.

Chapter 2: The second chapter details the design and implementation of the Mystic Dimensions mod. It covers the creation of unique biomes with distinct visual and thematic elements, the construction of new dimensions, and the development of custom resources and items. The mod aims to integrate seamlessly with Minecraft’s existing mechanics while offering new gameplay experiences.

Practical significance: The practical significance of the work lies in its contribution to the Minecraft modding community by providing a comprehensive modification that enhances the gaming experience. The technological solution developed can be implemented in electronic business environments (e-commerce and other economic entities).

Keywords: MYSTIC DIMENSIONS, MODDING, BIOMES, RESOURCES, GAMEPLAY, COMPATIBILITY, OPTIMIZATION, MINECRAFT, EXPLORATION, CREATIVITY, MODIFICATION, TERRAIN GENERATION.

ЗМІСТ

ВСТУП

Розробка мода Mystic Dimensions була розпочата через постійно зростаючуй попит у спільноті Minecraft на свіжий контент. Оскільки гравці шукають нові виклики та ігровий досвід, ця модифікація представляє новий вимір, який збагачує ігровий процес, пропонуючи новий біом та унікальні пригоди. Цей аддон додасть інтерес давнім гравцям і залучатиме нову аудиторію, до Minecraft. Розширюючи всесвіт гри, Mystic Dimensions значно подовжує тривалість гри та задоволення від неї.

Метою проекту Mystic Dimensions є створення комплексної та захоплюючої модифікації для Minecraft, яка збагачує ігровий процес. Ця модифікація має на меті представити унікальний вимір. Гравці зустрінуться з новими ландшафтами, ресурсами та викликами, створеними для стимулювання дослідження та творчості. Мод пропонує різноманітні умови навколишнього середовища та тематичні елементи, підвищуючи як естетичну привабливість, так і стратегічний геймплей. Крім того, він прагне підтримувати сумісність із існуючою ігровою механікою та іншими модифікаціями, забезпечуючи повну інтеграцію, що дозволяє гравцям налаштовувати свою гру, як заманеться. Цей проект прагне розширити горизонти Minecraft, надаючи гравцям нескінченні можливості для відкриттів і пригод.

Mystic Dimensions зосереджується на створенні нових вимірів у Minecraft. У дизайні надаватиметься пріоритет естетичному сполученню зі стилем Minecraft, одночасно вводячи нові елементи. Крім того, мод включає власні ресурси та предмети, які гравці зможуть знаходити та використовувати, покращуючи ігровий процес завдяки новим рецептам крафта та будівельним матеріалам. Кінцева мета - повноцінно інтегрувати нові виміри в існуючий ігровий світ, забезпечуючи сумісність і покращуючи загальний досвід гравця.

Дизайн і реалізація: це включає в себе розробку унікальних біомів з чіткими візуальними та тематичними елементами, гарантуючи, що кожен

вимір пропонує свіже середовище для дослідження гравців. Основа для нових вимірів включатиме алгоритми створення рельєфу, які створюватимуть різноманітні ландшафти, адаптовані до естетики кожного біому.

Створення ресурсів: будуть представлені ексклюзивні ресурси, такі як руди та будівельні матеріали, що розширить можливості для творчості та будівництва гравців. Нові предмети, інструменти та обладнання, що використовують ці ресурси, заохочуватимуть гравців досліджувати та повноцінно працювати з вмістом мода.

Тестування та оптимізація: для виявлення та виправлення помилок буде проведено широке тестування в різних ігрових сценаріях, що забезпечить плавну та приємну роботу для користувачів.

РОЗДІЛ 1
АНАЛІЗ СТАНУ ОБЛАСТІ РІШЕННЯ ЗАДАЧІ

Преамбула

Спільнота Minecraft, постійно розвивається і вигадує нові ідеї завдяки модифікаціям для гри. Так як спільнота Minecraft продовжує рости та впроваджувати інновації, це створює потреби в аналізі та дослідженні багатогранного всесвіту гри.

Незмінна популярність Minecraft охоплює покоління, приваблюючи мільйони гравців у всьому світі своїм поєднанням дослідження, створення та пригод. Але коли гравці глибше заглиблюються в механізми та можливості гри, вони стикаються як з можливостями, так і з проблемами. Поява модифікацій, та інструменнтів для їх створення, очолюваних спільнотою гри, відображає бажання звернути увагу відображає межі їх бажання розширити потенціал Mincraft, щоб покращити загальний досвід гравця.

Ми починаємо аналіз, для того щоб зрозуміти складність ігрового середовища Minecraft. Метою є визначення ключові сфери, де Minecraft може розвиватися та вдосконалюватися, а також запропонувати інноваційні рішення та підходи для вирішення цих проблем. Заглиблюючись у такі теми, як різноманітність ігрового процесу, оптимізація, залучення нових гравців.

1.1. Відсутність різноманітності у «Vanilla» Minecraft

У «Vanilla», тобто у Minecraft без модифікацій, гравці часто стикаються з відчуттям повтору через обмежену різноманітність різних аспектів гри.

1.1.1. Біоми та структури

«Vanilla» Minecraft надає гравцям дещо обмежений діапазон біомів і структур для дослідження (рис.1.). Біоми, відкрита зона в ігровому світі, яка

визначає ландшафт, рослинність і клімат регіону. Існують різноманітні біоми, від теплих лісів до холодної тундри, але порівняно з простором потенційного природного середовища загальний вибір може здатися обмеженим [1].

Подібним чином і структури, розкидані по всьому світу, вони є цікавими точками з потенційною здобиччю, які гравці можуть досліджувати. Ці споруди включають села, населені NPC, храми, що наповнені пастками та скарбами, а також підземелля, де живуть небезпечні моби та цінна здобич. Однак кількість унікальних споруд обмежена, і гравці можуть зустрічати один і той же тип структур багато разів в одному світі.

Відсутність різноманітності в біомах і структурах дає гравцю відчуття повторюваності та передбачуваності світу гри з часом. Крім того, обмежена різноманітність може призвести до відчуття ідентичності в кожній грі, зменшуючи відчуття відкриття чогось нового, яке повинно викликати дослідження світу гри.

Вирішення цієї проблеми у ванільному Minecraft може включати впровадження ширшого спектру біомів і структур, для покращення ігрового процесу і повертаючи відчуття чогось нового у грі. Урізноманітнюючи структури, з якими стикаються гравці, гра може забезпечити більш захоплюючий досвід та спонукати гравців шукати щось нове.

Рис. 1.1. Біоми

1.1.2. Різновиди мобів

«Vanilla» Minecraft має величезну різноманітність істот, які населяють світ гри, цей може здатися трохи обмеженою після різних довгих ігрових сесій (рис. 2.). Хоча Minecraft має широкий спектр істот, від пасивних тварин, таких як вівці та корови, до ворожих монстрів, таких як зомбі і скелети, але гравці можуть багато разів стикатися з одним і тим же типом істот, особливо у знайомих біомах [2].

Гравці можуть швидко звикнути до їх дій і здібностей, зменшуючи труднощі від зустрічі з черговим мобом. Крім того, обмежена різноманітність може сприяти відчуттю одноманітності в грі, оскільки гравці стикаються з однаковими загрозами, незалежно від їхнього місця розташування чи прогресу в грі.

Розширення різноманітності мобів в Minecraft передбачає впровадження нових типів істот з унікальною поведінкою і здібностями. Вони живуть у певних біомах та вимірах, додаючи глибини екосистемі гри, заохочуючи гравця досліджувати світ, щоб зустрітися з ними. Це збільшить загальну різноманітність і захоплення від ігрового процесу.

Рис. 1.2. Моби

1.1.3. Дефіцит ресурсів

У «Vanilla» Minecraft деякі ресурси можуть бути дефіцитними або обмеженими в певних біомах, і гравці можуть відчувати себе обмеженими у творчих зусиллях. Ресурси Minecraft включають широкий спектр матеріалів, які використовуються в крафті та будівництві: руди, рослини та інші рідкісні предмети. Хоча багато ресурсів у більшості своїй легкодоступні, деякі може бути важче отримати.

Брак ресурсів створює труднощі для гравців, особливо для тих, хто хоче розпочати амбітні будівельні проекти чи створити рідкісних предметів. Гравці можуть відчути, що вони обмежені доступністю певних матеріалів, що змушує їх дуже далеко мандрувати або торгувати з іншими гравцями щоб отримати те, що їм потрібно. Крім того, дефіцит ресурсів може сприяти нерівності в ігровому досвіді, так як гравцям у багатих на ресурси біомах може бути легше збирати потрібні їм ресурси, ніж гравцям у більш бідних на них.

Усунення дефіциту ресурсів в Minecraft передбачає пере балансування розподілу ресурсів, щоб забезпечити більш справедливий розподіл між біомами та вимірами. Це може включати коригування рівня появи дефіцитних ресурсів, впровадження нових методів отримання ресурсів або надання альтернативних методів отримання дефіцитних ресурсів.

Зменшуючи дефіцит ресурсів і забезпечуючи більшу доступність ресурсів, Minecraft може допомогти гравцям розкрити свій творчій потенціал і досягати цілей, не відлучаючи себе обмеженими в доступності ресурсів.

1.2. Складність і крива навчання

У Vanilla Minecraft ігрова механіка та система забезпечують баланс між доступністю для новачків і глибини для досвідчених гравців. Однак певні

аспекти гри можуть створювати труднощі для гравців, особливо з точки зору розуміння та оволодіння різними її тонкощами.

1.2.1. Система крафту

У ванільному Minecraft система крафту забезпечує гравцям основу для створення різних предметів, інструментів і блоків, використовуючи ресурси з ігрового світу, і ця система проста та універсальна, що дозволяє гравцям комбінувати різні інгредієнти в сітці ремесл для створення певного рецепту.

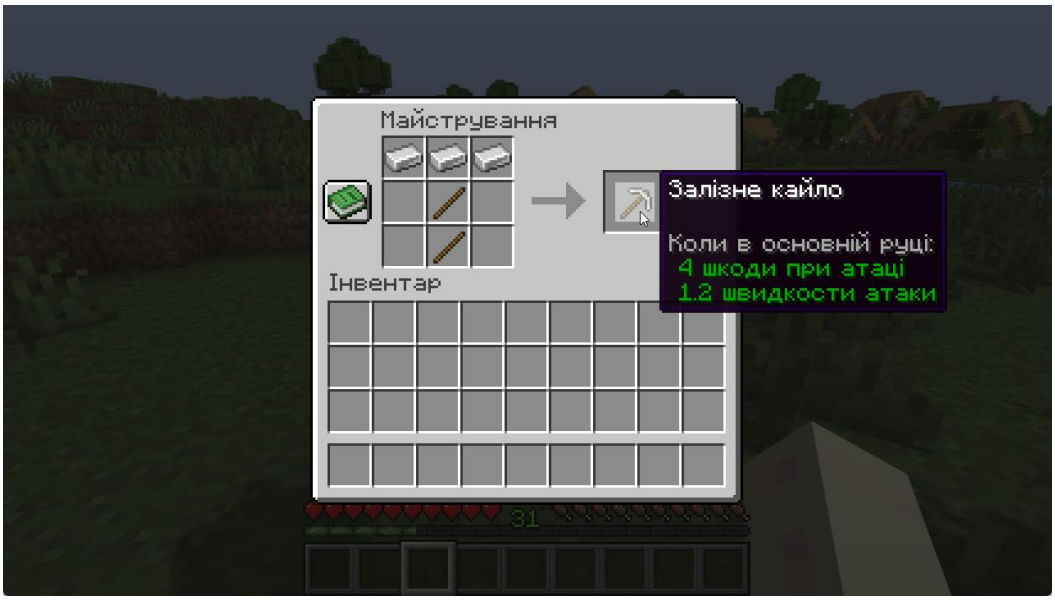

Рис. 1.3. Система крафту

Основні особливості системи крафту:

- Крафт на основі рецептів: гравці можуть виготовляти предмети, розміщуючи ресурси на сітці крафта 3x3 відповідно до попередньо визначених рецептів. Ці рецепти визначають компонування та комбінацію інгредієнтів, необхідних для створення кожного предмета (рис. 3.).

- Інтуїтивно зрозумілий інтерфейс: інтерфейс крафта дозволяє гравцям натискати на предмет у сітці крафта, щоб створити новий.
- Прогрес і дослідження: крафт відіграє одну із головних ролей у прогресі та дослідженні світу, оскільки гравці повинні збирати ресурси з навколишнього середовища, щоб створювати кращі інструменти, броню та зброю.
- Глибина та складність: хоча система крафту Minecraft є інтуїтивно зрозумілою та доступною, деякі гравці додають глибину та складність за допомогою модифікацій, для покращення ігрового досвіду, наприклад додавання складніших рецептів, які потребують більшої кількості матеріалів та кроків для створення предмету.
- Спеціалізація та налаштування: ще одна область для вдосконалення може включати використання спеціальних ремісничих станцій і верстаків, які дозволяють гравцям виготовляти певні типи предметів ефективніше або з унікальними бонусами. Це дозволяє гравцям експериментувати з різними техніками та стратегіями крафта.
- Інтеграція з іншими ігровими системами: розширена інтеграція системи крафту з іншими ігровими системами, такими як приготування їжі або редстоун механізми, це може надати гравцям більше можливостей експериментувати та досліджувати.

1.2.2. Ігрова механіка

Ігрова механіка у ванільному Minecraft включає в себе широкий спектр систем і функцій, які сприяють загальному ігровому досвіду. Ці механізми включають редстоун механізми, зачарування, приготування зілля, бої тощо. Незважаючи на те, що це забезпечує глибину та складність гри, вони також можуть поставити завдання гравцям, особливо початківцям, щодо їх розуміння.

Основні ігрові механізми в Minecraft:

- Редстоун це універсальний матеріал, який гравці можуть використовувати для створення складних, механізмів і автоматизованих систем у ігровому світі. Від простих дверей і пасток до детальних машин і схем Redstone Engineering пропонує безмежні можливості для творчості [5].
- Зачаровування: Зачаровування дозволяє гравцям надати своїм інструментам, зброї або броні різноманітні ефекти. поєднуючи зачаровувальні книги та різні предмети. Це може забезпечити такі бонуси, як захист, гострота, ефективність, тощо (рис.4.).
- Приготування зілля: приготування зілля дозволяє гравцям готувати різні зілля з різними ефектами, зокрема зцілення, невидимість, вогнестійкість тощо. Комбінуючи різні інгредієнти в зілляварильному стенді та додаючи палаючий порошок як паливо, гравці можуть варити зілля, які допоможуть їм досліджувати та виживати у світі гри.
- Бій: Бойовий механізм Minecraft передбачає нанесення урону ворожим мобам та іншим гравцівцям за допомогою різноманітної зброї. Гравці можуть використовувати мечі, луки та інші інструменти, щоб захистити себе від шкоди.

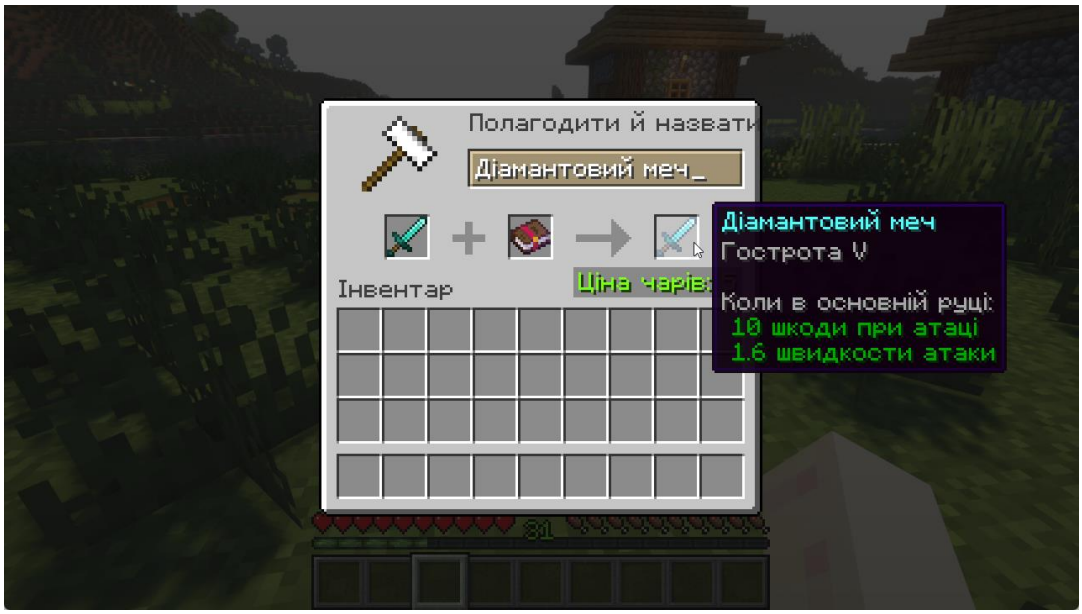

Рис. 1.4. Механіка зачарування

Питання, пов'язані з покращенням:

- Доступність і вказівки: Доступність і вказівки: одним із напрямків удосконалення ігрової механіки Minecraft є надання більш чітких вказівок і навчальних посібників для гравців, особливо новачків, щоб допомогти їм зрозуміти та освоїти гру. Це включає в себе навчальні посібники в грі, підказки або інтерактивні посібники, які пояснюють ключові поняття та механізми простим і доступним способом.
- Збалансованість і складність: збалансування складності ігрових механік, щоб гарантувати, що вони пропонують глибину та виклик, не перевантажуючи гравців, є важливим для забезпечення задоволення від ігрового процесу. Поступове впровадження механік та надання можливостей для експериментів може допомогти гравцям поступово звикнути до гри.

Загалом, покращення ігрової механіки в Minecraft передбачає досягнення балансу між доступністю та глибиною, надаючи гравцям можливість навчатися, експериментувати та впроваджувати інновації в насиченому та динамічному середовищі гри-пісочниці.

1.2.3. Розмірне дослідження

Дослідження просторів в Minecraft дає гравцям можливість вийти за межі звичайного світу гри та дослідити додаткові виміри, такі як Незер та Ендер світ. Хоча ці виміри пропонують унікальні виклики та досвід, вони не завжди можуть повністю задовольнити гравців, які хотіли би досліджувати більш різноманітні та захоплюючі виміри.

Основні розміри Minecraft:

- Незер: Незер — це ворожий вимір, наповнений небезпечною місцевістю, ворожими мобами та унікальними ресурсами, такими як

Кварц та світлокамінь. Гравці можуть отримати доступ до незеру, побудувавши портал за допомогою обсидіану та активувавши його вогнем (рис. 5.).

Рис. 1.5. Незер

- Ендер світ: Ендер світ - це таємничий вимір, населений Ендер Драконом і Ендерменами. Гравці після перемоги над Ендер Драконом можуть побачити кінцеві титри гри, що насправді не є кінцем гри, так як після можна далі мандрувати світом енду по віддаленим летючим островам (рис. 6.).

Рис. 1.6. Ендер світ

Питання, пов'язані з покращенням ігрового процесу:

- Розширення існуючих вимірів: Один із варіантів покращення вимірів у Minecraft полягає в тому, щоб розширити існуючі виміри, такі як Незер та Ендер світ, представивши гравцю новий світ, що включає додавання нових типів біомів, ресурсів, істот та порталу до цього виміру, або створити унікальні біоми у вже існуючих світах, а також цікаві структури, які гравці можуть досліджувати.
- Ендгейм контент: Покращеня дослідження просторів і ендгейм контенту забезпечує додаткове бажання гравцям досліджувати світи Minecraft. Це може включати впровадження нових випробувань і нагород, пов’язаних із дослідженням просторів, зокрема рідкісних артефактів, потужнї броні та доступу до ексклюзивного вмісту та функцій.

Удосконалення вимірів в Minecraft включають в себе додавання нових біомів, вимірів і покращення вмісту ендгейму, щоб надати гравцям більш різноманітний, захоплюючий і корисний досвід.

1.3. Проблеми з продуктивністю та оптимізацією

У модифікованому Minecraft дуже важливо забезпечити оптимальну продуктивність і сумісність з модифікаціями, щоб надати гравцям плавний і приємний ігровий досвід [7].

1.3.1. Сумісність модів

Модифікації часто додають нові функції, вміст і механіки в гру, але іноді вони можуть конфліктувати один з одним, що призводить до нестабільності та проблем з продуктивністю [6].

Конфлікти модифікацій можуть спричинити збої у роботі гри та. Ці конфлікти можуть бути викликані дублюванням функцій, конфліктами коду або несумісними залежностями. Саме тому сумісність модів є одним із ключових аспектів при їх розробці, для того, щоб гарантувати бездоганний ігровий досвід. Це передбачає забезпечення того, щоб різні моди добре працювали разом, без конфліктів або проблем, які можуть порушити ігровий процес.

Щоб вирішити проблеми сумісності модів, модери та розробники пакетів модів часто приймають різні варіанти вирішення проблем. Можна створювати патчі сумісності або додаткові модифіцації, які роз’єднують конфлікти між модифікаціями. Ці патчі та додаткові модифікації можуть регулювати поведінку одного або більше модів, щоб забезпечити їх сумісну роботу.

Іншим підходом є створення паку модифікацій, що дозволяє розробникам паків ретельно вибирати та тестувати моди, які доповнюють один одного та мінімізують конфлікти. Пакети модів зазвичай включають коригування та налаштування їх конфігурації , щоб гарантувати безперебійну взаємодію модів і забезпечити сумісний ігровий досвід.

Співпраця спільноти також відіграє важливу роль у покращенні сумісності модів. Гравці діляться своїм досвідом і рішеннями на форумах, вікі

та інших платформах спільноти, щоб допомогти у вирішенні проблем із сумісністю, з якими стикаються інші.

Загалом, сумісність модів допомагає підтримувати активна спільнота розробників і дозволяє гравцям насолоджуватися різноманітністю контенту, який вони пропонують, не стикаючись з дратівливими технічними проблемами.

1.3.2. Використання пам'яті

Коли мова заходить про модифікований Minecraft, використання пам’яті викликає велике занепокоєння, особливо для гравців, які мають слабкі системи або використовують велику кількість модів. Модифікації збільшують обсяг використовуємої пам’яті грою та можуть призвести до проблем із продуктивністю, таких як затримка, приривання та навіть збої.

Для вирішення проблеми з використанням великої кількості пам’яті є кілька варіантів, якмим можуть скористатися гравці:

- Вибір модів: одним із найефективніших способів керування використанням пам’яті є ретельний вибір модифікацій які буде використовувати гравець. Вибір спрощених модів, для оптимізації гри, може зменшити навантаження на пам’ять і покращити загальну стабільність.
- Конфігурація моду: багато модифікацій пропонують параметри конфігурації, які дозволяють гравцям регулювати різні параметри для оптимізації їх роботи. Точне налаштування таких параметрів, як відстань відтворення, швидкість відтворення об’єктів і графічні ефекти, допоможуть зменшити використання пам’яті та підвищити продуктивність.
- Оптимізація паку модифікацій: розробники паків модифікацій налаштовують їх для ефективної спільної роботи, і, забезпечуючи економічне використання ресурсомістких режимів, більшість

модпакетів також можуть містити моди для підвищення продуктивності та додаткові моди, які допомагають зменшити використання пам’яті та покращити загальну стабільність.

- Розподіл пам’яті: Ви можете налаштувати параметри розподілу пам’яті Minecraft, для виділення більшого чи меншого обсягу пам’яті для вашої гри. Дуже важливо знайти правильний баланс. Виділення занадто великого обсягу пам’яті може призвести до фрагментації пам’яті та проблем із продуктивністю, але виділення занадто малого обсягу пам’яті може призвести до збоїв та інших помилок.
- Аргументи запуску Java: Ви також можете використовувати аргументи запуску Java для оптимізації використання пам’яті та продуктивності. Такі налаштування, як алгоритм збирання сміття, розмір стека та пріоритет потоку, покращує загальну продуктивність і стабільність.

Загалом, керування використанням пам’яті в модифікованому Minecraft вимагає поєднання ретельного вибору модів, оптимізації та налаштування системи. Використовуючи все це, гравці можуть насолоджуватися більш плавною та стабільною грою навіть при великій кількості модифікацій.

1.3.3. Відстань промальовки

Відстань промальовки — це відстань, на якій гра відтворює сутності та чанки у ігровому світі. У Minecraft гравці можуть регулювати параметр відстані промальовки, щоб контролювати, як далеко вони можуть бачити в грі. Встановлення високої відстані промальовки дозволяє гравцям бачити об’єкти та місцевість, які знаходяться далеко від їх місцезнаходження, тоді як встановлення низької відстані промальовки обмежує видимість для найближчих об’єктів і місцевості [8].

Регулювання відстані промальовки відіграє важливу роль у продуктивності та споживанні пам’яті (рис. 7.). Це може збільшити

використання системних ресурсів і вплинути на продуктивність гри, особливо під час роботи слабких систем або великої кількості модів. Це може спричинити затримку, заїкання або зниження частоти кадрів.

З іншого боку, встановлення меншої відстані відтворення зменшує навантаження на систему, обмежуючи відстань, на якій відображаються сутності та чанки. Хоча це допомагає підвищити продуктивність і стабільність гри, це також може бути менш захоплюючим ігровим досвідом, оскільки віддалені об’єкти можуть бути невидимі для гравця.

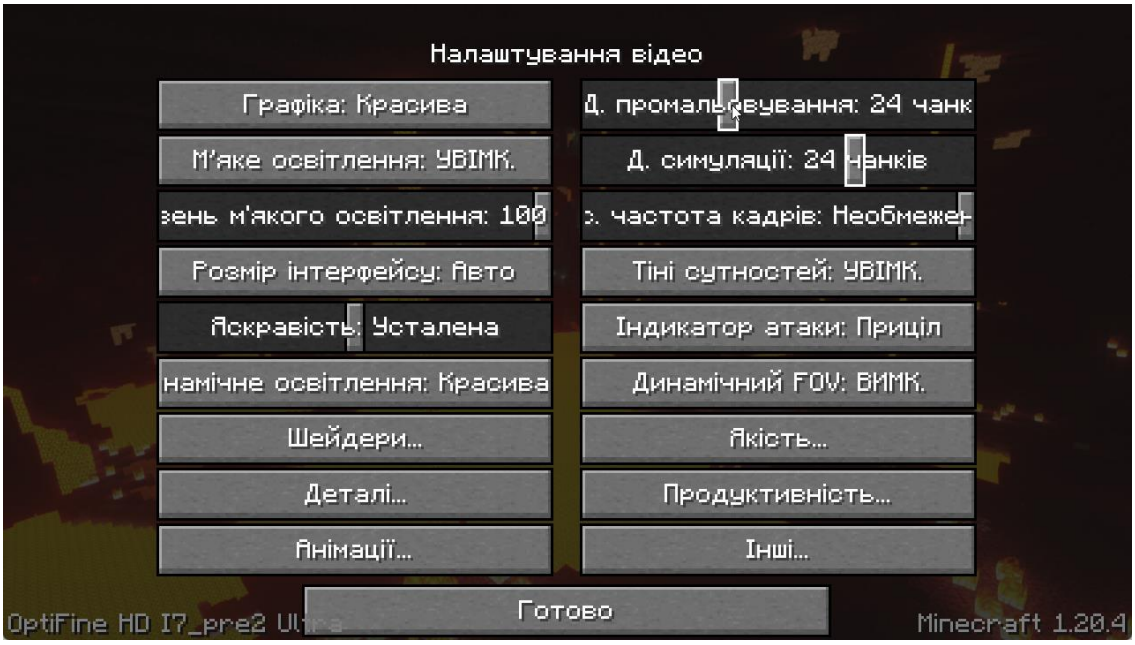

Рис. 1.7. Налаштування промальовки чанків

Налаштовуючи відстань промальовки в Minecraft, важливо знайти правильний баланс між візуальною якістю та продуктивністю. Гравцям, які використовують слабкі системи або мають проблеми з продуктивністю, можливо, знадобиться зменшити налаштування відстані промальовки, щоб покращити продуктивність гри. З іншого боку, гравці, які використовують більш продуктивні системи, можуть збільшити це налаштування, та насолоджуватися більшою дальністю видимості і зануренням.

Крім того, гравці із модифікаціями в Minecraft можуть зіткнутися з додатковими труднощами в оптимізації відстаней промальовки через підвищену складність для системи і вимоги до запуску кількох модів одночасно. У цих випадках вибір модифікацій, конфігурація та оптимізація є ще важливішими для забезпечення плавної та приємної гри.

1.3.4. API Fabric

Fabric API — це фреймворк для створення і запуску модифікацій для Minecraft, який надає набір інструментів і утиліт розробникам модів для більш ефективної розробки. Він служить основою, на якій можна створювати модифікації, пропонуючи низку функцій і можливостей для спрощення процесу модифікації.

Одним із ключових аспектів Fabric API є його легкий і модульний дизайн. На відміну від інших модифікаційних фреймворків, таких як Forge, Fabric API зосереджується на простоті та продуктивності, дозволяючи розробникам модів створювати легкі моди, які мінімально впливають на продуктивність гри.

API Fabric надає різноманітні інструменти та утиліти для спрощення типових завдань при розробці. До них входять API для взаємодії з різними аспектами гри, такими як блоки, елементи, сутності та події. Розробники модів можуть використовувати ці API, щоб створювати новий вміст, змінювати наявні функції та додавати власну поведінку до гри.

На додаток до своїх API, Fabric API також містить низку утиліт і допоміжних класів для допомоги розробникам модів. Ці утиліти вирішують такі завдання, як керування конфігурацією, мережевий зв’язок і створення інтерфейсу користувача, що дозволяє розробникам модів зосередитися на реалізації своїх ідей, а не винаходити колесо.

Модульна архітектура Fabric API дозволяє розробникам модів змішувати та поєднувати різні компоненти відповідно до своїх потреб. Ця

гнучкість дозволяє розробникам створювати модифікації, адаптовані до конкретних умов гри, незалежно від того, додають вони новий вміст, покращують наявні функції чи змінюють ігрову механіку.

Загалом, Fabric API відіграє зараз дуже важливу роль у спільноті модів Minecraft, надаючи легку, гнучку та ефективну структуру для розробників модів для створення модів. Його орієнтація на простоту, продуктивність і модульність робить його популярним вибором як для новачків, так і для досвідчених розробників модів.

1.4. Залучення гравців і довголіття

Забезпечення залученості і тривалого інтересу до гри має вирішальне значення для підтримки живої та активної бази гравців у Minecraft.

1.4.1. Контент ендгейму

Контент гри Minecraft стосується дій, викликів і нагород, доступних гравцям після досягнення важливих етапів і цілей у грі. Він створений, для забезпечення постійної мотивації та участі гравців, які досягли пізніх стадій гри.

У ванільному Minecraft основною метою зазвичай є перемога над Драконом Ендера, що вказує на те, що основна історія гри завершена. Але далі цього моменту гравці не можуть переслідувати чіткі цілі чи завдання, що може призвести до зменшення інтересу.

Щоб вирішити цю проблему, розробники модів часто пропонують додатковий ендгейм вміст, щоб надати гравцям постійні виклики та нагороди.

- Битви з босами: гравцям пропонуються нові боси та могутні вороги, яких потрібно перемогти, кожен з яких має унікальні здібності

та механізми. Ці боси можуть запропонувати цінні нагороди, такі як рідкісні предмети, ресурси та обладнання.

- Підземелля: часто розробники додають нові підземелля, щоб гравці могли досліджувати та перемагати поодинці чи з іншими гравцями. Ці підземелля можуть містити складних ворогів, головоломки та пастки, що дає гравцям захоплюючий та складни ігровий доствід.
- Дослідження та відкриття: створення нових біомів та вимірів, структур для дослідження гравцям. Ці біоми можуть надати гравцям унікальні завдання, ресурси та можливості перевірити свої навички.
- Система прогресування: нові системи прогресування та механіка, які дозволяють гравцям продовжувати рости та розвивати своїх персонажів і світи навіть після досягнення кінця гри, що включає розблокування нових здібностей, модернізацію обладнання.

Розробники модифікацій можуть надавати різноманітний і захоплюючий ендгейм контент, забезпечуючи нові виклики та досвід для гравців, щоб мотивувати і далі проводити час у грі та отримувати задоволення навіть після завершення основної історії. Це допоможе подовжити життя гри та підтримувати базу активних гравців.

1.4.2. Взаємодія спільноти

Взаємодія спільноти в Minecraft серед гравців за допомогою різних засобів, включаючи багатокористувацьку гру, форуми та створення контенту, орієнтованого на гравця. Йдеться про створення можливостей для гравців спілкуватися один з одним, співпрацювати та ділитися своїм досвідом, збагачувати загальний ігровий досвід і зміцнювати спільноту.

Одним із основних способів, якими Minecraft заохочує взаємодію спільноти, є використання багатокористувацької гри. Гравці можуть приєднуватися до серверів, щоб взаємодіяти з іншими гравцями в спільному світі, працювати разом над проектами, брати участь у різних подіях і у дружніх

змаганнях. Сервери у грі пропонують широкий спектр досвіду, від виживання та творчого будівництва до міні-ігор, рольових ігор, і задовольняє інтереси та вподобання різних гравців.

На додаток до ігрового процесу для кількох гравців чат, полегшує спілкування та взаємодію між гравцями. Це дозволяює гравцям координувати свою діяльність у грі, та дружити. Форуми спільноти, сервери Discord та інші онлайн-платформи ще більше сприяють спілкуванню та сумісній роботі з іншими гравцями, забезпечують простір для дискусій, обміну соїми ідеями та організації подій та заходів у самій грі.

Створення контенту, орієнтованого на гравця, є ще одним важливим аспектом взаємодії спільноти в Minecraft. Зокрема, спільнота міні-ігор відіграє ключову роль у розширенні та збагаченні досвіду Minecraft, створюючи новий контент, функції та ігрові механіки. Розробники діляться своїми твореннями зі спільнотою через такі платформи, як CurseForge і Modrinth, а гравці знаходять моди, які покращують гру.

Окрім модифікацій, розробники сприяють спільноті за допомогою інших форм створення контенту, таких як створення та обмін користувацькими картами, створення навчальних посібників, створення фан-арту та анімації, а також стріми або створення ігрових відео. Це все допомагає надихати та взаємодіяти з іншим гравцям, заохочуючи творчість, співпрацю та почуття приналежності до спільноти Minecraft.

Загалом взаємодія спільноти є фундаментальним аспектом досвіду Minecraft, що дозволяє гравцям зв’язуватися та співпрацювати один з одним у жвавій та інклюзивній спільноті, надаючи можливості для гри на кількох гравців, сприяючи спілкуванню та співпраці та підтримуючи створення контенту, орієнтованого на гравця. Це збагачує ігровий досвід і підтримує сильне почуття єдності, що забезпечує довголіття гри та її подальший успіх.

1.4.3. Реграбельність

Реграбельність Minecraft означає здатність гри надавати різноманітний і захоплюючий досвід, що спонукає гравців грати в гру кілька разів. Йдеться про отримування гравцями нових і захоплюючих ігрових можливостей щоразу, коли вони заходять у новий світ, із збереженням динамічного, веселого та корисного досвіду протягом тривалого часу.

Одним із важливих факторів, що сприяють реграбельності Minecraft, є процедурна система генерації світу. У грі створюються випадкові світи, кожен новий світ унікальний і надає гравцям різні виклики, ресурси та можливості для його дослідження. Ця випадковість дозволяє гравцям досліджувати, експериментувати та відкривати для себе щось нове в кожній грі.

В Minecraft немає сюжету як такого, що дозволяє гравцям формувати власні історії та досвід у грі. Незалежно від того, чи ви будуєте величезні споруди, чи створюєте союзи з іншими гравцями, Minecraft дає свободу гри та нескінченні можливості для творчості, дослідження та самовираження.

Модифікації також підвищують зручність гри в Minecraft, розширюючи та налаштовуючи вміст і механізми гри. Завдяки новим біомам, істотам, предметам, ігровій механіці та тисячам доступних режимів - це дозволяє гравцям постійно досліджувати та експериментувати з новим вмістом, роблячи кожну гру свіжою та захоплюючою. Це чудовий спосіб отримати уявлення про те, що відбувається у вашому житті.

Загалом відтворюваність Minecraft підтримується його процедурним світом, динамічними подіями, орієнтованими на гравця сюжетними лініями та великою спільнотою модів. Надаючи гравцям інший ігровий досвід, який можна налаштувати, Minecraft дозволить гравцям продовжувати насолоджуватися грою та досліджувати нескінченні можливості творчості, досліджень і пригод у найближчі роки.

1.5. Творчість і налаштування

Природа пісочниці Minecraft дозволяє гравцям виявляти свою творчість і налаштовувати гру по-різному.

1.5.1. Будівельні інструменти

Інструменти для будівництва Minecraft охоплюють низку механік, які дозволяють гравцям створювати детальні та креативні споруди в ігровому світі. Це необхідно, для того щоб розкрити креативність гравця та дозволити йому виразити себе.

По суті, будівництво в Minecraft передбачає просто розміщення ігрових блоків з яких складається весь світ гри. Гравці проектують і будують усе: від простих будинків і замків до складних скульптур і піксель-артів, використовуючи різноманітні блоки, такі як дерево, камінь, скло та різні декоративні блоки.

Інструменти для створення структур покращують цей процес, надаючи гравцям додаткові функції, зручність і гнучкість. Деякі з основних будівельних інструментів і функцій у Minecraft:

- Вибір і розміщення блоків: гравці можуть вибирати блоки зі свого інвентарю та розміщувати їх у ігровому світі. Уміння правильно розміщувати блоки дозволяє гравцям легко створювати детальні та складні будівлі.
- Будівельні матеріали: Minecraft пропонує широкий вибір будівельних матеріалів, кожен з яких має свої особливості та текстуру. Гравці можуть експериментувати з різними матеріалами, щоб створити потрібний вигляд.
- Інструменти творчого режиму: у творчому режимі гравці мають доступ до всіх матерівалів які присутні у грі, це полегшує створення структур а також у гравця в творчому режімі є можливість польоту. Це дає можливість гравцеві зосередитися на будівлях, не турбуючись про труднощі збору матеріалів або виживання.

Загалом будівельні інструменти відіграють ключову роль у розширенні можливостей гравців втілювати свої творчі бачення в Minecraft. Незалежно від того, будуєте ви прості будівлі, складні гігантські споруди, ці інструменти надають гравцям інструменти для створення та проектування в своїх світах нескінченними способами, заохочуючи творчість, дослідження та самовираження в спільноті Minecraft.

1.5.2. Кастомізація

Кастомізація в Minecraft стосується різних варіантів, якими гравці можуть використовувати, щоб змінювати свій ігровий досвід відповідно до своїх уподобань та переваг. Ці параметри включають широкий спектр плагінів і модифікацій, які дозволяють гравцям налаштовувати все, від візуальних елементів гри до ігрової механіки та вмісту.

Одним з основних варіантів налаштування в Minecraft є використання пакетів ресурсів. За допомогою пакету ресурсів гравці можуть змінювати текстуру, звук і музику гри та створювати власні візуальні стилі та атмосферу. Гравці мають широкий вибір пакетів ресурсів, створених спільнотою, від реалістичних текстур до стилізованих тем та інше.

Крім того, гравці можуть використовувати шейдери для подальшого покращення візуальних ефектів у грі. Шейдери дозволяють гравцям мати приголомшливі та захоплюючі середовища з динамічним освітленням і реалістичними ефектами води, тіні, відображення та інші візуальні ефекти гри. Шейдери можуть значно покращити естетичну привабливість гри та надати гравцям візуально привабливий ігровий досвід (рис. 8.).

Рис. 1.8. З шейдерами та без

Окрім візуальної кастомізації, гравці також можуть різними способами змінити механіку та вміст гри. Моди - це, мабуть, найпоширеніший спосіб налаштувати Minecraft і дозволити гравцям додавати нові функції, ігрову механіку та контент до гри. Модифікації можуть представляти все: від нових біомів, істот і предметів до абсолютно нових вимірів, систем прогресування та режимів гри, надаючи гравцям нескінченні можливості для гри та експериментів.

Загалом параметри налаштування відіграють важливу роль у покращенні досвіду Minecraft, дозволяючи гравцям виражати себе, персоналізувати ігровий досвід та досліджувати новий та захоплюючий контент. Незалежно від того, створюєте ви власні текстури, експериментуєте з шейдерами чи переходите у світ модів, Minecraft надає гравцям безліч можливостей для налаштування та самовираження, гарантуючи, що досвід кожного гравця буде унікальним і адаптованим до його власних уподобань.

Висновок

Модифікація «Mystic Dimensions» базується на всебічному аналізі ігрового середовища Minecraft, та спрямований на покращення ігрового

досвіду Minecraft завдяки поєднанню, креативності та прискіпливої уваги до деталей. Ця модифікація не тільки усуває виявлені недоліки гри, а також збагачує ігровий досвід гравців Minecraft.

Буде впроваджений новий вимір, наповнений новими біомами, такими як ефірні таємничі ліси, кришталеві рівнини та таємничі ефірні болота, наповнених власними унікальними ресурсами, структурами і мобами. У цих вимірах, надається гравцям можливість різноманітних зустрічей та цілі, для відчуття відкриття і прогресу.

В основі нововведень мода лежить додавання містичних кристалів, універсального ресурсу, який не тільки розширює стратегічний вибір гравця, але й поглиблює занурення, посилюючи відчуття прогресу, діючи як точка опори для створення предметів, занурених у потужні інструменти, броню та унікальні предмети. Також, включення унікальних споруд, таких як загадкові таємничі вежі та небезпечні кришталеві печери, пропонує гравцям нові цілі, наповнені викликами та нагородами. Поява нових мобів, включаючи грізного таємничого Вартового і відважного Кришталевого Голема, додасть елемент небезпеки і азарту в пригоди гравця і наповнить ігровий світ життям і динамкою.

РОЗДІЛ 2
ПРОЕКТНІ РІШЕННЯ

2.1. Створення та налаштування проекту модифікації

Для розробки модифікації була обрана **Java**, так як це основна мова на якій розробляють модифікації для Minecraft. **API Fabric** був обраний, через свою зручність, під час процесу розробки. Також для зручності в налаштуванні генерації біомів, використовувується додатковий **API TerraBlender.**

2.1.1. Створення шаблону модифікації

Генерування шаблону, з параметрами генерації даних на офіційному сайті **Fabric API**.

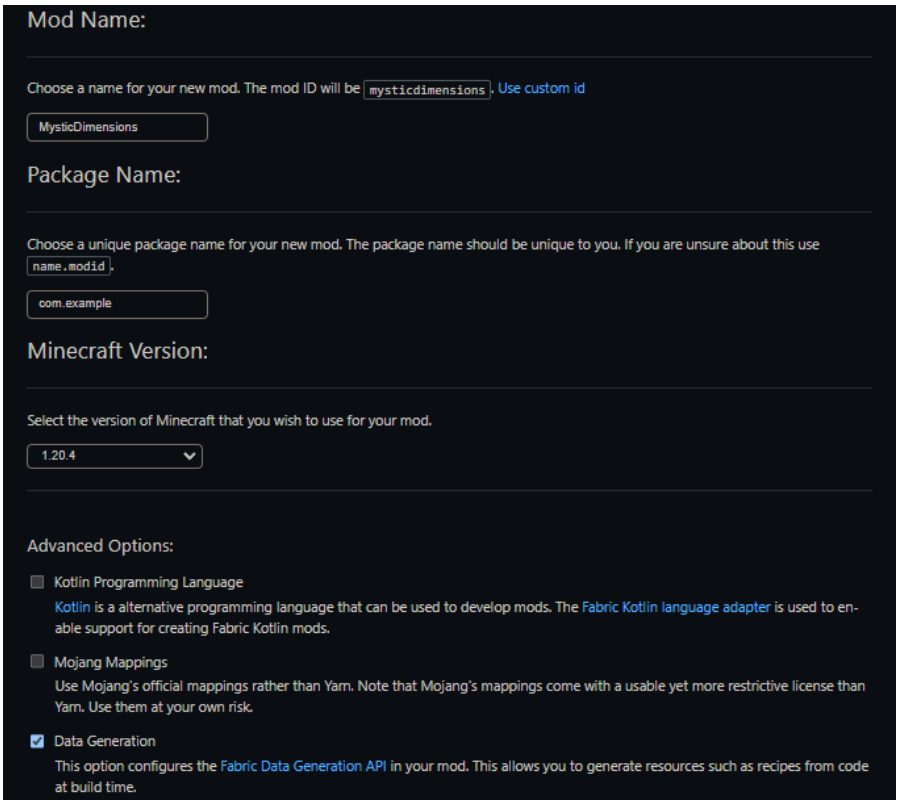

Рис. 2.1 Генерування шаблону модифікації

2.1.2. Першочергове налаштування проекту

Налаштування залежностей у файлі fabric.mod.json

```
{
  "schemaVersion": 1,
  "id": "mysticdimensions",
  "version": "${version}",
  "name": "MysticDimensions",
  "description": " Ця модифікація була створена для дипломної роботи студентом НТУ 'ДП' ",
  "authors": [
    "Сердюк Юрій"
  ],
  "contact": {
    "homepage": "https://fabricmc.net/",
    "sources": "https://github.com/FabricMC/fabric-example-mod"
  },
  "license": "MIT",
  "icon": "assets/mysticdimensions/icon.png",
  "environment": "*",
  "entrypoints": {
    "main": [
      "com.example.mysticdimensions.MysticDimensions"
    ],
    "client": [
      "com.example.mysticdimensions.MysticDimensionsClient"
    ],
    "fabric-datagen": [
      "com.example.mysticdimensions.MysticDimensionsDataGenerator"
    ]
  },
  "mixins": [
    "mysticdimensions.mixins.json"
  ],
  "depends": {
    "fabricloader": ">=0.15.11",
    "minecraft": "~1.20.4",
    "java": ">=17",
    "fabric-api": "*"
  },
  "suggests": {
    "another-mod": "*"
  }
}
```

Збірка всіх вихідних даних та конфігурація проекту за допомогую **Gradle**

```
PS C:\Users\brick4ik\Desktop\diplom\mysticdimensions-template-1.20.4> ./gradlew genSources
```

Рис 2.2 Збірка використовуючи **Gradle**

2.1.3. Першочергове налаштування у коді головного класу модифікації

Оголошення змінної MOD_ID у головному класі MysticDimensions

```
public class MysticDimensions implements ModInitializer {
    public static final String MOD_ID = "mysticdimensions";
    public static final Logger LOGGER = LoggerFactory.getLogger(MOD_ID);

    @Override
    public void onInitialize() {

        LOGGER.info("Hello Fabric world!");
    }
}
```

2.1.4. Перший перевірочний запуск за допомогою Gradle

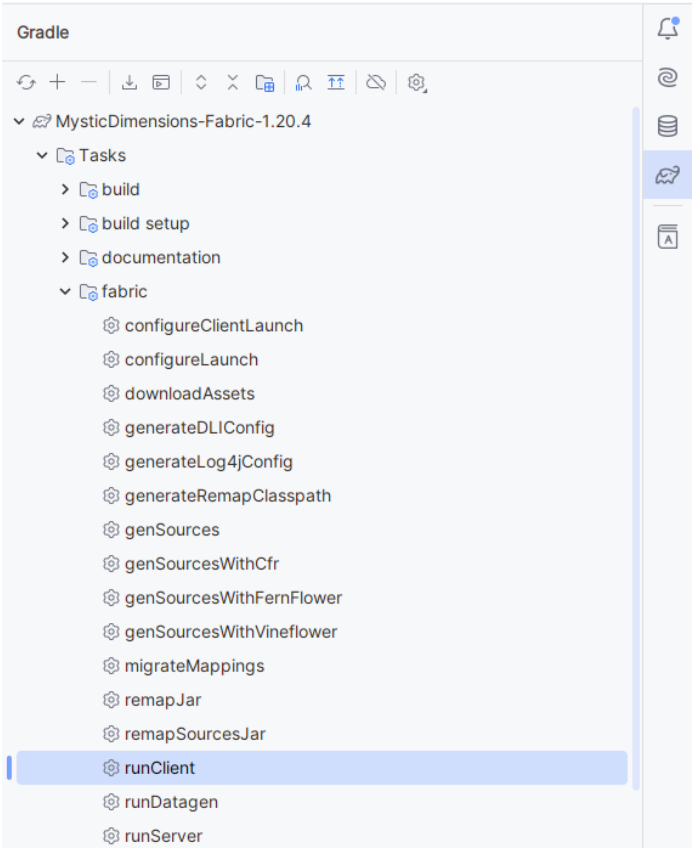

Рис 2.3 Запуск через runClient

Рис 2.4 Головне меню Minecraft

2.2. Додавання кастомних блоків та предметів до гри

Додавання блоків, предметів, а також створення групи моду у кретивному режимі гри.

2.2.1. Створення класу ModBlocks

Клас ModBlocks для створення кастомних блоків був створений за каталогом: com\example\mysticdimensions\block

```
public class ModBlocks {
```

Реєстрація всіх кастомних блоків типу Block:

```
public static final Block MYSTIC_CRYSTAL_BLOCK = registerBlock("mystic_crystal_block",
        new Block(FabricBlockSettings.copyOf(Blocks.AMETHYST_BLOCK).luminance(10)));
public static final Block MYSTIC_STONE = registerBlock("mystic_stone",
        new Block(FabricBlockSettings.copyOf(Blocks.DEEPSLATE)));
public static final Block MYSTIC_COBBLESTONE = registerBlock("mystic_cobblestone",
        new Block(FabricBlockSettings.copyOf(Blocks.COBBLED_DEEPSLATE)));
public static final Block PLANKS_MYSTIC = registerBlock("planks_mystic",
        new Block(FabricBlockSettings.copyOf(Blocks.CRIMSON_PLANKS)));
```

```
public static final Block MYSTIC_DUST_ORE = registerBlock("mystic_dust_ore",
        new Block(FabricBlockSettings.copyOf(Blocks.DEEPSLATE_LAPIS_ORE).luminance(5)));
public static final Block MYSTIC_DIRT = registerBlock("mystic_dirt",
        new Block(FabricBlockSettings.copyOf(Blocks.DIRT)));
```

Реєстрація блоку кастомної деревени типу PillarBlock:

```
public static final Block MYSTIC_WOOD = registerBlock("mystic_wood",
        new PillarBlock(FabricBlockSettings.copyOf(Blocks.ACACIA_LOG)));
```

Реєстрація блоку кастомної трави типу GrassBlock:

```
public static final Block MYSTIC_GRASS_BLOCK = registerBlock("mystic_grass_block",
        new GrassBlock(FabricBlockSettings.copyOf(Blocks.GRASS_BLOCK)));
```

Реєстрація блоку кастомного листя типу LeavesBlock:

```
public static final Block MYSTIC_LEAVES = registerBlock("mystic_leaves",
        new LeavesBlock(FabricBlockSettings.copyOf(Blocks.CHERRY_LEAVES)));
```

Реєстрація Block для блоків:

```
    private static Block registerBlock(String name, Block block) {
        registerBlockItem(name, block);
        return Registry.register(Registries.BLOCK, new Identifier(MysticDimensions.MOD_ID, name), block);
    }
```

Реєстрація BlockItem для блоків:

```
    private static Item registerBlockItem(String name, Block block) {
        return Registry.register(Registries.ITEM, new Identifier(MysticDimensions.MOD_ID, name),
                new BlockItem(block, new FabricItemSettings()));
    }
}
```

2.2.2. Створення json файлів для моделей блоків, та стану блоків

Всі json моделі блоків обов’язково повинні міститись в каталозі: resources\assets\mysticdimensions\models\block.

Всі json файли стану блоків обов’язково повинні міститись в каталозі: resources\assets\mysticdimensions\blockstates.

Всі Json файли моделі предмету для блока, задля корректного відображення блоку у руці обов’язково повинен міститись в каталозі: resources\assets\mysticdimensions\models\item

Json файл моделі блоку з однаковою текстурою з усіх боків:

```
{
    "parent": "block/cube_all",
    "textures": {
        "all": "mysticdimensions:block/mystic_crystal_block"
```

```
    }
}
```

Json файл моделі предмета для цього блоку:

```
{
  "parent": "mysticdimensions:block/mystic_crystal_block"
}
```

Json файл стану цього блоку:

```
{
  "variants": {
    "": {
      "model": "mysticdimensions:block/mystic_crystal_block"
    }
  }
}
```

Json файл моделі блоку деревини:

```
{
  "parent": "block/cube_column",
    "textures": {
      "end": "mysticdimensions:block/mystic_wood_top",
      "side": "mysticdimensions:block/mystic_wood"
    }
  }
```

Json файл стану блоку деревини:

```
{
  "variants": {
    "axis=x": {
      "model": "mysticdimensions:block/mystic_wood_horizontal",
      "x": 90,
      "y": 90
    },
    "axis=y": {
      "model": "mysticdimensions:block/mystic_wood"
    },
    "axis=z": {
      "model": "mysticdimensions:block/mystic_wood_horizontal",
      "x": 90
    }
  }
}
```

У файлі моделі блока є декілька типів надання текстур блоку зокрема:

cube_all - всі сторони блоку мають однакову текстуру

cube_collum - Блок має дві текстури з боків, та згори та знизу.

2.2.3 Надання текстур Блокам

Для надання текстур блокам треба помістити текстури в каталог: resources\assets\mysticdimensions\textures\block.

Текстура обов’язково повинна мати правильну назву наприклад «Кристальний блок» - «mystic_crystal_block.png»

Рис 2.5 Кастомі блоки з наданими їм текстурами

2.2.4. Створення класу ModItemGroups

Клас ModItemGroups має призначення створити каталог предметів та блоків з моду у креативному режимі гри:

```
public class ModItemGroups {
    public static final ItemGroup MYSTICDIMENSIONS_GROUP = Registry.register(Registries.ITEM_GROUP,
        new Identifier(MysticDimensions.MOD_ID, "mysticdimensions"),
        FabricItemGroup.builder().displayName(Text.translatable("itemgroup.mysticdimensions"))
            .icon(() -> new ItemStack(ModItems.MYSTIC_CRYSTAL)).entries((displayContext, entries) -> {

                entries.add(new ItemStack(ModBlocks.MYSTIC_DIRT));
                entries.add(new ItemStack(ModBlocks.MYSTIC_GRASS_BLOCK));
                entries.add(new ItemStack(ModBlocks.MYSTIC_CRYSTAL_BLOCK));
                entries.add(new ItemStack(ModBlocks.MYSTIC_WOOD));
                entries.add(new ItemStack(ModBlocks.PLANKS_MYSTIC));
                entries.add(new ItemStack(ModBlocks.MYSTIC_STONE));
                entries.add(new ItemStack(ModBlocks.MYSTIC_COBBLESTONE));
                entries.add(new ItemStack(ModBlocks.MYSTIC_DUST_ORE));
                entries.add(new ItemStack(ModBlocks.MYSTIC_LEAVES));
```

```
        }).build());
    }
```

Рис 2.6 Кастомний каталог предметів у креативному режимі гри

2.2.5. Створення класу ModFuel

Клас ModFuel створенний для того, щоб надати властивості предмету, чи блоку бути пальним у пічці для переплавки, чи приготування чогось у світі гри:

```
public class ModFuel {

    public static void registerModFuel() {
        FuelRegistry.INSTANCE.add(ModBlocks.MYSTIC_WOOD, 480);
        FuelRegistry.INSTANCE.add(ModBlocks.PLANKS_MYSTIC, 120);
    }
}
```

Надання блокам деревени, та дощок властивості пального для пічки.

2.2.6. Створення класу ModItems

Клас ModItems для створення кастомних предметів був створений за каталогом: com\example\mysticdimensions\item

Реєстрація нових предметів, та створення класу:

```
public class ModItems {

    public static final Item MYSTIC_DUST = registerItem("mystic_dust",
            new Item(new FabricItemSettings()));
    public static final Item PIECE_MYSTIC_DUST = registerItem("piece_mystic_dust",
            new Item(new FabricItemSettings()));
    public static final Item ETHEREAL_ESSENCE = registerItem("ethereal_essence",
            new Item(new FabricItemSettings()));
    public static final Item GUARDIAN_ESSENCE = registerItem("guardian_essence",
            new Item(new FabricItemSettings()));
    public static final Item MYSTIC_INGOT = registerItem("mystic_ingot",
            new Item(new FabricItemSettings()));
    public static final Item MYSTIC_CRYSTAL = registerItem("mystic_crystal",
            new Item(new FabricItemSettings()));
    public static final Item MYSTIC_CRYSTAL_SHARD = registerItem("mystic_crystal_shard",
            new Item(new FabricItemSettings()));

    private static void addItemsToIngridientItemGroup(FabricItemGroupEntries entries) {
    }

    private static Item registerItem(String name, Item item) {
        return Registry.register(Registries.ITEM, new Identifier(MysticDimensions.MOD_ID, name), item);
    }
}
```

Створені предмети, треба також додати до класу ModItemGroups:

```
entries.add(new ItemStack(ModItems.MYSTIC_DUST));
entries.add(new ItemStack(ModItems.PIECE_MYSTIC_DUST));
entries.add(new ItemStack(ModItems.MYSTIC_MIX));
entries.add(new ItemStack(ModItems.ETHEREAL_ESSENCE));
entries.add(new ItemStack(ModItems.GUARDIAN_ESSENCE));
entries.add(new ItemStack(ModItems.MYSTIC_INGOT));
entries.add(new ItemStack(ModItems.MYSTIC_CRYSTAL));
entries.add(new ItemStack(ModItems.MYSTIC_CRYSTAL_SHARD));
```

2.2.7. Створення класу MysticMixItem

Клас MysticMixItem створений, для надання особливих можливостей предмету «Містична Суміш», який надає гравцю при використанні ефект нічного бачення на кілька секунд, а також генерує частинки для більш красивого ефекту при використанні. Клас викликається при використанні

предмета гравцем, після використання накладує на всі предмети цього типу кулдаун, надаючи гравцю ефект нічного бачення.

Клас знаходиться в каталозі: com\example\mysticdimensions\item\custom

```
public class MysticMixItem extends Item {
    private static final int COOLDOWN_DURATION = 128;
    public MysticMixItem(Settings settings) {
        super(settings);
    }

    @Override
    public TypedActionResult<ItemStack> use(World world, PlayerEntity user, Hand hand) {
        if (!world.isClient) {
            if (!user.getItemCooldownManager().isCoolingDown(this)) {
            user.addStatusEffect(new StatusEffectInstance(StatusEffects.NIGHT_VISION, 85, 0));
            user.getItemCooldownManager().set(this, COOLDOWN_DURATION);
            user.getStackInHand(hand).decrement(1);
                spawnParticles((ServerWorld) world, user);
            }
        }

        return TypedActionResult.success(user.getStackInHand(hand));

    }
    private void spawnParticles(ServerWorld world, PlayerEntity player) {
        for (int i = 0; i < 20; i++) {
            double X = world.random.nextGaussian() * 0.02;
            double Y = world.random.nextGaussian() * 0.02;
            double Z = world.random.nextGaussian() * 0.02;
            world.spawnParticles(ParticleTypes.GLOW_SQUID_INK,
                    player.getX(), player.getY() + 1.0, player.getZ(),
                    1, X, Y, Z, 0.15);
        }
    }
}
```

Цей клас використовується при створенні предмету «Містична суміш» у класі ModItems, де додатково вказано .maxCount, тобто обмеження на кількість предметів в одному слоті інвентаря:

```
public static final Item MYSTIC_MIX = registerItem("mystic_mix",
        new MysticMixItem(new FabricItemSettings().maxCount(16)));
```

2.2.8. Створення класу ModToolMaterial

Клас ModToolMaterial використовується для надання матеріалу кастомним інструментам. Цей клас знаходиться в каталозі: com\example\mysticdimensions\item. Він корегує їхні характеристики, а також використовується для їх ремонту, впливає на швидкість добування блоків

інструментами яким буде наданий матеріал з цього класу, на міцність, на кількість дамагу, який буде нанесений, цими інструментами, а також на якість зачарувань, які можуть бути накладені через зачарувальний стіл:

```
public enum ModToolMaterial implements ToolMaterial{
    MYSTIC(MiningLevels.DIAMOND,
            650, 5f, 0, 26,
    () -> Ingredient.ofItems(ModItems.MYSTIC));

    private final int miningLevel;
    private final int ItemDurability;
    private final float miningSpeed;
    private final float attackDamage;
    private final int enchantability;
    private final Supplier<Ingredient> repairIngredient;

    ModToolMaterial(int miningLevel, int itemDurability, float miningSpeed, float attackDamage, int enchantability,
Supplier<Ingredient> repairIngredient) {
        this.miningLevel = miningLevel;
        this.ItemDurability = itemDurability;
        this.miningSpeed = miningSpeed;
        this.attackDamage = attackDamage;
        this.enchantability = enchantability;
        this.repairIngredient = repairIngredient;
    }

    @Override
    public int getDurability() {
        return this.ItemDurability;
    }

    @Override
    public float getMiningSpeedMultiplier() {
        return this.miningSpeed;
    }

    @Override
    public float getAttackDamage() {
        return this.attackDamage;
    }

    @Override
    public int getMiningLevel() {
        return this.miningLevel;
    }

    @Override
    public int getEnchantability() {
        return this.enchantability;
    }

    @Override
    public Ingredient getRepairIngredient() {
        return this.repairIngredient.get();
    }
}
```

Інструменти також додані до класу ModItems, а також вказано, який саме матеріал відповідає цьому класу:

```
public static final Item MYSTIC_PICKAXE = registerItem("mystic_pickaxe",
        new PickaxeItem(ModToolMaterial.MYSTIC, 4, 1.2f, new FabricItemSettings()));
public static final Item MYSTIC_AXE = registerItem("mystic_axe",
        new AxeItem(ModToolMaterial.MYSTIC, 10, 0.9f, new FabricItemSettings()));
public static final Item MYSTIC_SHOVEL = registerItem("mystic_shovel",
        new ShovelItem(ModToolMaterial.MYSTIC, 6, 1, new FabricItemSettings()));
public static final Item MYSTIC_SWORD = registerItem("mystic_sword",
        new SwordItem(ModToolMaterial.MYSTIC, 9, 1.6f, new FabricItemSettings()));
public static final Item MYSTIC_HOE = registerItem("mystic_hoe",
        new HoeItem(ModToolMaterial.MYSTIC, 4, 4, new FabricItemSettings()));

public static final ItemConvertible MYSTIC = MYSTIC_INGOT;
private static void addItemsToIngridientItemGroup(FabricItemGroupEntries entries){}
```

2.2.9. Створення класу ModArmorMaterial

Клас ModArmorMaterial використовується для створення матеріалу Броні, та привласнює матеріалу характеристики міцності, захисту, опір відкидуванню:

```
public enum ModArmorMaterial implements ArmorMaterial {
    MYSTIC("mystic", 2, 4,
            19, 10, 2, () -> Ingredient.ofItems(ModItems.MYSTIC));
    ;

    private final String name;
    private final int DurabilityMultiplayer;
    private final int Protection;
    private final int Enchantability;
    private final float Toughness;
    private final float KnockbackResistance;
    private final Supplier<Ingredient> repairIngredient;

    private static final int [] BASE_DURABILITY = {11, 16, 15, 13};

    ModArmorMaterial(String name, int durabilitymutiplayer, int protection, int enchantability, float toughness, float
knockbackResistance, Supplier<Ingredient> repairIngredient) {
        this.name = name;
        this.DurabilityMultiplayer = durabilitymutiplayer;
        this.Protection = protection;
        this.Enchantability = enchantability;
        this.Toughness = toughness;
        this.KnockbackResistance = knockbackResistance;
        this.repairIngredient = (Supplier<Ingredient>) repairIngredient;
    }
    @Override
    public int getDurability(ArmorItem.Type type)
    {return BASE_DURABILITY[type.ordinal()] * this.DurabilityMultiplayer;
    }
    @Override
    public int getProtection(ArmorItem.Type type)
    {return this.Protection;
    @Override
    public int getEnchantability()
    {return this.Enchantability;
    }
    @Override
    public SoundEvent getEquipSound()
    {return null;
    }
```

```
    @Override
    public Ingredient getRepairIngredient()
    {return this.repairIngredient.get();
    }
    @Override
    public String getName()
    {return "";
    @Override
    public float getToughness()
    {return this.Toughness;
    }
    @Override
    public float getKnockbackResistance()
    {return this.KnockbackResistance;
    }
}
```

Також сама броня реалізована у класі ModItems:

```
public static final Item MYSTIC_HELMET = registerItem("mystic_helmet",
        new ArmorItem(ModArmorMaterial.MYSTIC, ArmorItem.Type.HELMET, new FabricItemSettings()));
public static final Item MYSTIC_CHESTPLATE = registerItem("mystic_chestplate",
        new ArmorItem(ModArmorMaterial.MYSTIC, ArmorItem.Type.CHESTPLATE, new FabricItemSettings()));
public static final Item MYSTIC_LEGGINS = registerItem("mystic_leggins",
        new ArmorItem(ModArmorMaterial.MYSTIC, ArmorItem.Type.LEGGINGS, new FabricItemSettings()));
public static final Item MYSTIC_BOOTS = registerItem("mystic_boots",
        new ArmorItem(ModArmorMaterial.MYSTIC, ArmorItem.Type.BOOTS, new FabricItemSettings()));
```

2.2.10. Json файли моделей кастомних предметів

Для правильного відображення інструментів у руці гравця, використовується «parent» = «item/handheld» на прикладі топора:

```
{
  "parent": "item/handheld",
  "textures": {
    "layer0": "mysticdimensions:item/mystic_axe"
  }
}
```

Для звичайних предметів використовується «parent» = «minecraft:item», на прикладі містичного кристалу:

```
{
  "parent": "minecraft:item/generated",
  "textures": {
    "layer0": "mysticdimensions:item/mystic_crystal"
  }
}
```

2.2.11. Створення рецептів крафту

Всі рецепти крафту зберігаються у Json файлах обов’язково повинні знаходитись в каталозі: resources\data\mysticdimensions\recipes. Існує декілька різних типів крафту наприклад безформенний «crafting_shapeless» на прикладі «Уламків містичного кристалу» обирається тип крафту, категорія в якій буде знаходитись цей крафт, обирається предмет, який буде використовуватись у крафті, та результат крафту:

```
{
  "type": "minecraft:crafting_shapeless ",
  "category": "misc",
  "ingredients": [
    {
      "item": "mysticdimensions:mystic_crystal"
    }
  ],
  "result": {
    "item": "mysticdimensions:mystic_crystal_shard",
    "count": 2
  }
}
```

Також існує фігурний крафт на прикладі «Містичної суміщі», також обирається тип крафту, та ми обираємо паттерн для нього, після чого вказуються змінні, яким привласнюється один із предметів, після чого вказується результат крафту, та кількість предметів, які були зроблені:

```
{
  "type": "minecraft:crafting_shaped",
  "pattern": [
    "#",
    "C",
    "#"
  ],
  "key": {
    "#": {
      "item": "mysticdimensions:piece_mystic_dust"
    },
    "C": {
      "item": "mysticdimensions:mystic_crystal_shard"
    }
  },
  "result": {
    "item": "mysticdimensions:mystic_mix",
    "count": 1
  }
}
```

Переплавка, також є крафтом та реалізується схожим чином, але трохи інакше, на прикладі крафту «Містичного злитку», вказується тип крафту,

категорія, час переплавки, кількість отриманого досвіда, за одну переплавку, та предмети, з якого складаєтсья крафт, та результат крафту:

```
{
  "type": "minecraft:smelting",
  "category": "misc",
  "cookingtime": 200,
  "experience": 2.0,
  "group": "mysticdimensions",
  "ingredient": {
    "item": "mysticdimensions:mystic_dust"
  },
  "result": "mysticdimensions:mystic_ingot"
}
```

2.2.12. Таблиці здобичі блоків

Всі таблиці здобичи для блоків у Json файлах обов’язково зберігаються в каталозі: resources\data\mysticdimensions\loot_tables\blocks. Приклад звичайної здобичі з блоків «Блоку містичного кристалу» Вказується тип де тут це блок, вказано, що він випаде, якщо переживе ігровий вибух, вказаний предмет, який випадає з блока, та їх кількість:

```
{
  "type": "minecraft:block",
  "pools": [
    {
      "bonus_rolls": 0.0,
      "conditions": [
        {
          "condition": "minecraft:survives_explosion"
        }
      ],
      "entries": [
        {
          "type": "minecraft:item",
          "name": "mysticdimensions:mystic_crystal_block"
        }
      ],
      "rolls": 1.0
    }
  ]
}
```

2.2.13. Надання текстур предметам

Текстури для предметів додаються аналогічно з блоками, але текстури для предметів обов’язково повинні знаходитись в каталозі:

resources\assets\mysticdimensions\textures\item, та також мати назву аналогічну до назви предмета, якому вони належать.

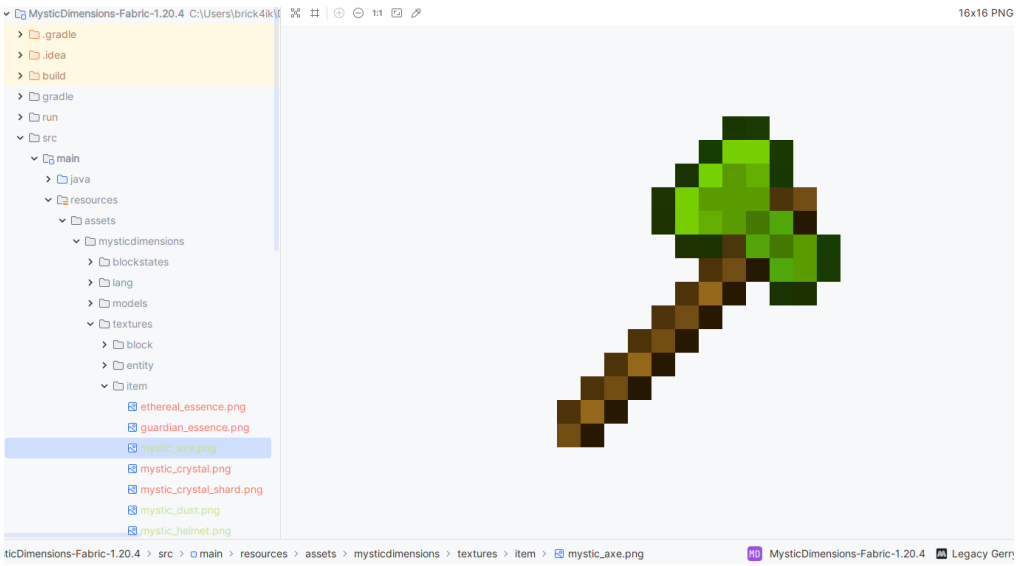

Рис 2.7 Кастомна текстура, та шлях до каталогу

2.3. Додавання кастомних мобів до гри

Додавання мобів, та налаштування їх поведінки.

2.3.1. Створення класу ModEntities

Клас ModEntities відповідає за реєстрацію сутностей у моді. Реєструється «Містичний кам’яний охоронець», типу монстр розміром в 1 блок. Клас знаходиться в каталозі com\example\mysticdimensions\entity:

```
public class ModEntities {
    public static final EntityType<MysticStoneGuardianEntity> MYSTICSTONEGUARDIAN =
Registry.register(Registries.ENTITY_TYPE,
            new Identifier(MysticDimensions.MOD_ID, "mystic_stone_guardian"),
            FabricEntityTypeBuilder.create(SpawnGroup.MONSTER, MysticStoneGuardianEntity::new)
                    .dimensions(EntityDimensions.fixed(1f, 1f)).build());
}
```

2.3.2. Створення класу MysticStoneGuardianEntity

Клас MysticStoneGuardianEntity, створений для надання сутності «Містичний кам’яний охоронець», характеристик та характеру поведінки.

Кількість здоров'я моба, швидкість пересування, кількість захисту, швидкість у польоті та скільки шкоди може нанести гравцеві. Моб є агресивним до гравця.

Також в класі прописані ефекти, які викидає з себе моб:

```
public class MysticStoneGuardianEntity extends IronGolemEntity {
    public MysticStoneGuardianEntity(EntityType<? extends IronGolemEntity> entityType, World world) {
        super(entityType, world);
    }

    @Override
    protected void initGoals() {
        this.goalSelector.add(0, new SwimGoal(this));
        this.goalSelector.add(1, new WanderAroundGoal(this, 1.0D));

        this.goalSelector.add(2, new AttackGoal(this));
        this.goalSelector.add(3, new MeleeAttackGoal(this, 1.0D, true));

        this.goalSelector.add(4, new LookAtEntityGoal(this, PlayerEntity.class, 3));
        this.goalSelector.add(5, new LookAroundGoal(this));
        this.targetSelector.add(6, new ActiveTargetGoal<>(this, PlayerEntity.class, true));
    }

    public static DefaultAttributeContainer.Builder createMysticStoneGuardianAttributes() {
        return MobEntity.createMobAttributes()
                .add(EntityAttributes.GENERIC_MAX_HEALTH, 40)
                .add(EntityAttributes.GENERIC_MOVEMENT_SPEED, 0.5)
                .add(EntityAttributes.GENERIC_ARMOR, 2)
                .add(EntityAttributes.GENERIC_FLYING_SPEED, 1)
                .add(EntityAttributes.GENERIC_ATTACK_DAMAGE, 13.5f);

    }

    @Override
    public void tick() {
        super.tick();

        World world = this.getWorld();

        if (world != null) {
            double offsetX = this.random.nextGaussian() * 0.07;
            double offsetY = this.random.nextGaussian() * 0.07;
            double offsetZ = this.random.nextGaussian() * 0.07;

            double posX = this.getX();
            double posY = this.getY() + this.getHeight();
            double posZ = this.getZ();

            world.addParticle(ParticleTypes.SMOKE, posX, posY, posZ, offsetX, offsetY, offsetZ);
        }
    }
}
```

2.3.3. Реалізація моделі кастомного моба

Для створення моделі, було використано ПЗ blockbench, яке створене спеціально, для реалізації подібного роду моделей для Minecraft.

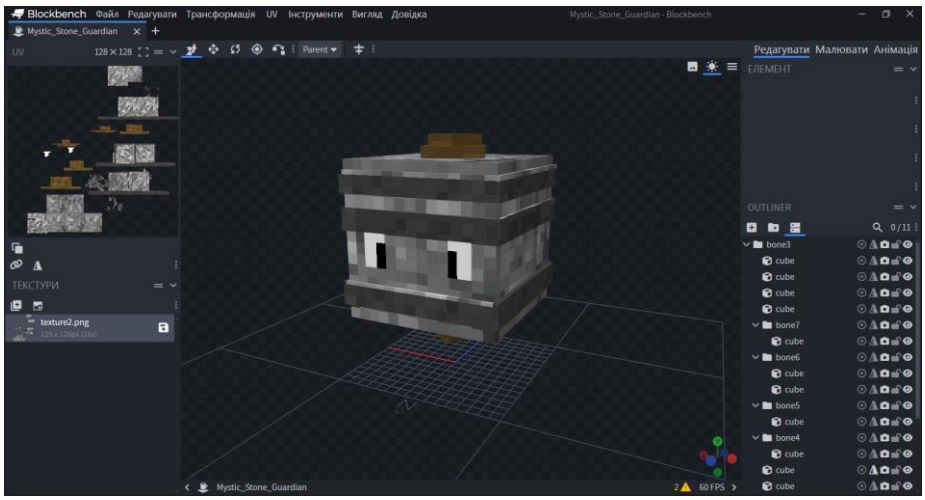

Рис 2.8 Створена модель у середовищі моделювання blockbench

Модель еспортується у вигляді Java файлу:

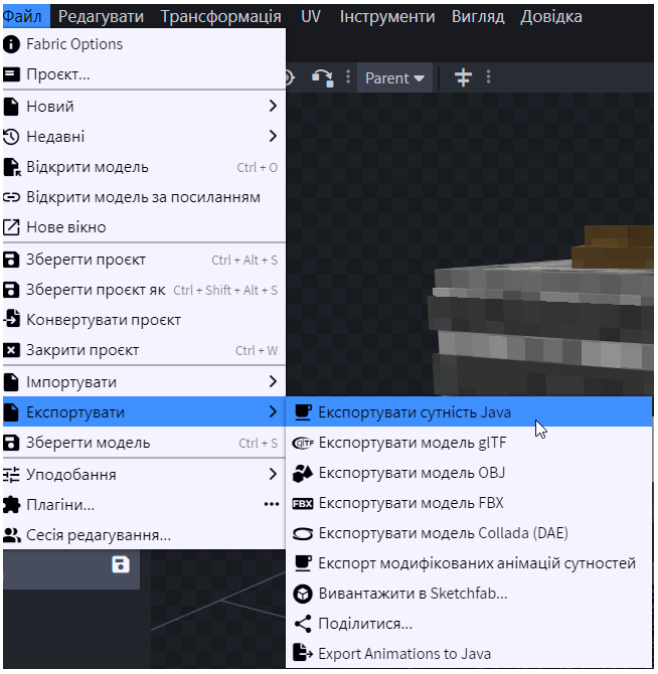

Рис 2.9 Експорт моделі з blockbench

Модель із середини виглядає приблизно так:

```
public class MysticStoneGuardianModel <T extends MysticStoneGuardianEntity> extends SinglePartEntityModel<T> {  4 usages  new *
    private final ModelPart bone3;  3 usages

    public MysticStoneGuardianModel(ModelPart root) {  1 usage  new *
        this.bone3 = root.getChild( name: "bone3");
    }
    public static TexturedModelData getTexturedModelData() {  1 usage  new *
        ModelData modelData = new ModelData();
        ModelPartData modelPartData = modelData.getRoot();

        ModelPartData bone3 = modelPartData.addChild( name: "bone3", ModelPartBuilder.create().uv( textureX: 15,  textureY: 96).cuboid( offset
        .uv( textureX: 66,  textureY: 58).cuboid( offsetX: -14.4F,  offsetY: -9.45F,  offsetZ: -7.3F,  sizeX: 14.8F,  sizeY: 2.2F,  sizeZ: 14.7F, new Dila
        .uv( textureX: 68,  textureY: 23).cuboid( offsetX: -14.3F,  offsetY: -19.15F,  offsetZ: -7.3F,  sizeX: 14.6F,  sizeY: 2.0F,  sizeZ: 14.6F, new Di
        .uv( textureX: 61,  textureY: 79).cuboid( offsetX: -14.3F,  offsetY: -15.95F,  offsetZ: -7.2F,  sizeX: 14.6F,  sizeY: 2.0F,  sizeZ: 14.5F, new Dil
        .uv( textureX: 27,  textureY: 64).mirrored().cuboid( offsetX: -12.0F,  offsetY: -14.0F,  offsetZ: -7.2F,  sizeX: 2.0F,  sizeY: 3.0F,  sizeZ: 0.2F
        .uv( textureX: 45,  textureY: 61).cuboid( offsetX: -4.0F,  offsetY: -14.0F,  offsetZ: -7.2F,  sizeX: 2.0F,  sizeY: 3.0F,  sizeZ: 0.2F, new Dilati

        ModelPartData bone7 = bone3.addChild( name: "bone7", ModelPartBuilder.create().uv( textureX: 84,  textureY: 44).cuboid( offsetX: -2.7F,

        ModelPartData bone6 = bone3.addChild( name: "bone6", ModelPartBuilder.create(), ModelTransform.pivot( pivotX: -7.0F,  pivotY: -7.3

        ModelPartData cube_r1 = bone6.addChild( name: "cube_r1", ModelPartBuilder.create().uv( textureX: 66,  textureY: 45).cuboid( offsetX: -1
        .uv( textureX: 25,  textureY: 83).cuboid( offsetX: -3.7F,  offsetY: 1.3F,  offsetZ: -3.7F,  sizeX: 7.4F,  sizeY: 1.0F,  sizeZ: 7.4F, new Dilation(

        ModelPartData bone5 = bone3.addChild( name: "bone5", ModelPartBuilder.create(), ModelTransform.pivot( pivotX: -7.0F,  pivotY: -23.3

        ModelPartData cube_r2 = bone5.addChild( name: "cube_r2", ModelPartBuilder.create().uv( textureX: 42,  textureY: 71).cuboid( offsetX: -2

        ModelPartData bone4 = bone3.addChild( name: "bone4", ModelPartBuilder.create(), ModelTransform.pivot( pivotX: -7.0F,  pivotY: -25.3

        ModelPartData cube_r3 = bone4.addChild( name: "cube_r3", ModelPartBuilder.create().uv( textureX: 38,  textureY: 55).cuboid( offsetX: -1
```

Рис 2.10 частина коду експортованої моделі

Кастомна модель цього моба, була імпортована до каталогу com\example\mysticdimensions\entity\client, а також створені два класи у цьому каталозі ModModelLayers та MysticStoneGuardianRenderer.

Клас ModModelLayers призначений для реєстрації шару моделі «MYSTICSTONEGUARDIAN», для допомоги у рендерингу сутності, для відображення у світі:

```
public class ModModelLayers {
   public static final EntityModelLayer MYSTICSTONEGUARDIAN =
         new EntityModelLayer(new Identifier(MysticDimensions.MOD_ID, "mystic_stone_guardian"), "main");
}
```

Клас MysticStoneGuardianRenderer забезпечує правильне відображення моделі та текстури сутності в Minecraft Розширюючи клас MobEntityRenderer, і також відповідаючи за рендеринг сутності «MYSTICSTONEGUARDIAN», з моделлю MysticStoneGuardianModel:

```
public class MysticStoneGuardianRenderer extends MobEntityRenderer<MysticStoneGuardianEntity,
MysticStoneGuardianModel<MysticStoneGuardianEntity>> {
   public static final Identifier TEXTURE =
         new Identifier(MysticDimensions.MOD_ID, "textures/entity/mystic_stone_guardian.png");
   public MysticStoneGuardianRenderer(EntityRendererFactory.Context context) {
      super(context, new MysticStoneGuardianModel<>(context.getPart
            (ModModelLayers.MYSTICSTONEGUARDIAN)), 0.8f);
   }

   @Override
   public Identifier getTexture(MysticStoneGuardianEntity entity) {
      return TEXTURE;
   }
}
```

Рис 2.11 «Містичний кам’яний охоронець» у грі

2.4. Створення кастомних біомів, вимір та генерація руди

Створення нового біому, виміру та додавання генерації кастомної руди для нових подорожей грою

2.4.1. Створення кастомного біому

Для кастомного біому потрібно створити класи ModBiomes, ModOverworldRegion, ModTerrablenderAPI.

ModTerrablenderAPI – Ініціалізація TerraBlender API:

```
public class ModTerrablenderAPI implements TerraBlenderApi {
    @Override
    public void onTerraBlenderInitialized() {
        Regions.register(new ModOverworldRegion(new Identifier(MysticDimensions.MOD_ID, "overworld"), 4));

        SurfaceRuleManager.addSurfaceRules(SurfaceRuleManager.RuleCategory.OVERWORLD,
MysticDimensions.MOD_ID, ModMaterialRules.makeRules());
    }
}
```

ModBiones – Налаштування кольорів біому, звуків біому (.BiomeBuilder) глобальні налаштування генерації для світу, та глобальні налаштування біому (globalOverworldGeneration):

```
public class ModBiomes {
    public static final RegistryKey<Biome> MYSTIC_BIOME = RegistryKey.of(RegistryKeys.BIOME,
            new Identifier(MysticDimensions.MOD_ID, "mystic_biome"));

    public static void bootstrap(Registerable<Biome> context) {
        context.register(MYSTIC_BIOME, mysticBiome(context));
    }

    public static void globalOverworldGeneration(GenerationSettings.LookupBackedBuilder builder) {
        DefaultBiomeFeatures.addLandCarvers(builder);
        DefaultBiomeFeatures.addAmethystGeodes(builder);
        DefaultBiomeFeatures.addDungeons(builder);
        DefaultBiomeFeatures.addMineables(builder);
        DefaultBiomeFeatures.addSprings(builder);
        DefaultBiomeFeatures.addFrozenTopLayer(builder);
    }

    public static Biome mysticBiome(Registerable<Biome> context) {
        SpawnSettings.Builder spawnBuilder = new SpawnSettings.Builder();
        spawnBuilder.spawn(SpawnGroup.CREATURE, new
SpawnSettings.SpawnEntry(ModEntities.MYSTICSTONEGUARDIAN, 2, 3, 5));

        spawnBuilder.spawn(SpawnGroup.CREATURE, new SpawnSettings.SpawnEntry(EntityType.WOLF, 5, 4, 4));

        DefaultBiomeFeatures.addFarmAnimals(spawnBuilder);
        DefaultBiomeFeatures.addBatsAndMonsters(spawnBuilder);

        GenerationSettings.LookupBackedBuilder biomeBuilder =
                new
GenerationSettings.LookupBackedBuilder(context.getRegistryLookup(RegistryKeys.PLACED_FEATURE),
                        context.getRegistryLookup(RegistryKeys.CONFIGURED_CARVER));

        globalOverworldGeneration(biomeBuilder);
        DefaultBiomeFeatures.addMossyRocks(biomeBuilder);
        DefaultBiomeFeatures.addDefaultOres(biomeBuilder);
        DefaultBiomeFeatures.addExtraGoldOre(biomeBuilder);

        biomeBuilder.feature(GenerationStep.Feature.VEGETAL_DECORATION,
VegetationPlacedFeatures.TREES_PLAINS);
        DefaultBiomeFeatures.addForestFlowers(biomeBuilder);
        DefaultBiomeFeatures.addLargeFerns(biomeBuilder);

        DefaultBiomeFeatures.addDefaultMushrooms(biomeBuilder);
        DefaultBiomeFeatures.addDefaultVegetation(biomeBuilder);

        return new Biome.Builder()
```

```
            .precipitation(true)
            .downfall(0.4f)
            .temperature(0.7f)
            .generationSettings(biomeBuilder.build())
            .spawnSettings(spawnBuilder.build())
            .effects((new BiomeEffects.Builder())
                .waterColor(0xe82e3b)
                .waterFogColor(0x6342C7)
                .skyColor(0x30c918)
                .grassColor(0x7f03fc)
                .foliageColor(0xd203fc)
                .fogColor(0x22a1e6)
                .music(new MusicSound(SoundEvents.MUSIC_OVERWORLD_BADLANDS, 12000, 24000, true))
                .build())
            .build();
    }
}
```

ModOverworldRegion - Відповідає за визначення та модифікацію біомів у стантартному світі Minecraft:

```
public class ModOverworldRegion extends Region {
    public ModOverworldRegion(Identifier name, int weight) {
        super(name, RegionType.OVERWORLD, weight);
    }

    @Override
    public void addBiomes(Registry<Biome> registry, Consumer<Pair<MultiNoiseUtil.NoiseHypercube,
RegistryKey<Biome>>> mapper) {
        super.addBiomes(registry, mapper);
        this.addModifiedVanillaOverworldBiomes(mapper, modifiedVanillaOverworldBuilder -> {
            modifiedVanillaOverworldBuilder.replaceBiome(BiomeKeys.FOREST, ModBiomes.MYSTIC_BIOME);
        });
    }
}
```

2.4.2. Створення нового кастомного виміру

Клас ModDimensions реєструє новий вимір:

```
public class ModDimensions {
    public static final RegistryKey<World> MYSTIC_DIMENSION_KEY = RegistryKey.of(RegistryKeys.WORLD,
            new Identifier(MysticDimensions.MOD_ID, "mystic_dimension"));
    public static final RegistryKey<DimensionType> MYSTIC_DIMENSION_TYPE_KEY =
            RegistryKey.of(RegistryKeys.DIMENSION_TYPE, MYSTIC_DIMENSION_KEY.getValue());

    public static void register() {
        MysticDimensions.LOGGER.debug("Registering ModDimensions for " + MysticDimensions.MOD_ID);
    }
}
```

Для налаштування виміру створюються два Json файли які повинні називатись так само як і сам вимір «mystic_dimension».

В каталозі resources\data\mysticdimensions\dimension файл для налаштування генерації виміру:

```
{
  "type": "mysticdimensions:mystic_dimension",
  "generator": {
    "type": "minecraft:noise",
    "settings": "minecraft:floating_islands",
    "seed": 0,
    "biome_source": {
      "type": "minecraft:fixed",
      "biome": "mysticdimensions:mystic_biome"
    }
  }
}
```

В каталозі resources\data\mysticdimensions\dimension_type файл для налаштування ефектів у вимірі, його висоти та рейту спавну мобів:

```
{
  "ultrawarm": false,
  "natural": true,
  "coordinate_scale": 1,
  "ambient_light": 0.1,
  "has_skylight": true,
  "has_ceiling": false,
  "infiniburn": "#minecraft:infiniburn_overworld",
  "logical_height" : 320,
  "has_raids": false,
  "respawn_anchor_works": true,
  "bed_works": true,
  "piglin_safe": false,
  "height": 320,
  "min_y": 0,
  "effects": "minecraft:overworld",
  "monster_spawn_block_light_limit": 8,
  "monster_spawn_light_level": 1
}
```

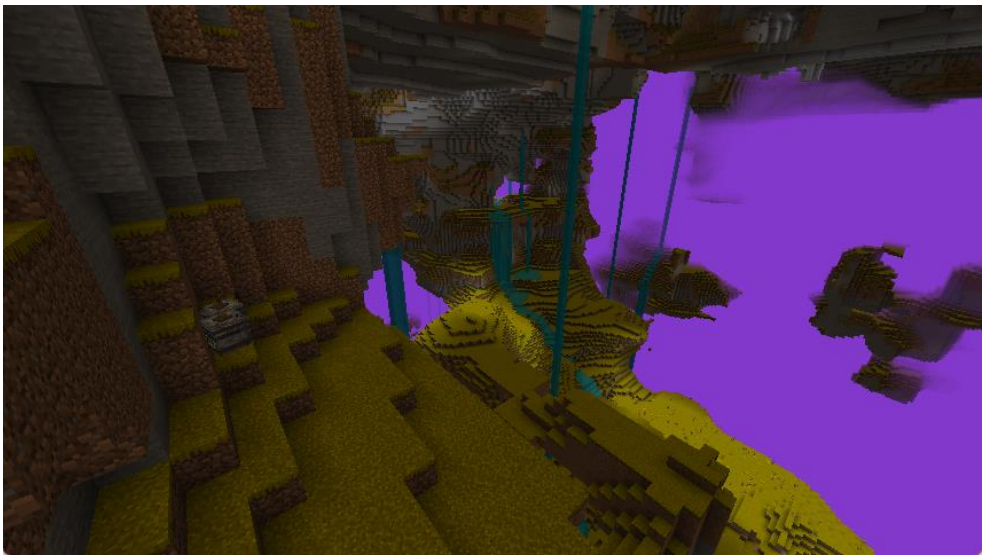

Рис 2.12 Сгенерований вимір

2.4.3 Генерація кастомної руди

Для додавання руди, яка сама генерується у світі створюються декілька класів ModConfiguredFeatures – цей клас відповідає за визначення та реєстрацію налаштованих особливостей для генерації руд у світі:

```
public class ModConfiguredFeatures {
    public static final RegistryKey<ConfiguredFeature<?, ?>>
            MYSTIC_DUST_ORE_KEY = registerKey("mystic_dust_ore");

    public static void boostrap
            (Registerable<ConfiguredFeature<?, ?>> context) {
        RuleTest stoneReplacables =
                new TagMatchRuleTest(BlockTags.STONE_ORE_REPLACEABLES);


        List<OreFeatureConfig.Target> overworldRubyOres =
                List.of(OreFeatureConfig.createTarget(stoneReplacables,
                        ModBlocks.MYSTIC_DUST_ORE.getDefaultState()));


        register(context, MYSTIC_DUST_ORE_KEY, Feature.ORE,
                new OreFeatureConfig(overworldRubyOres, 14));

    }


    public static RegistryKey<ConfiguredFeature<?, ?>> registerKey(String name) {
        return RegistryKey.of
                (RegistryKeys.CONFIGURED_FEATURE,
                        new Identifier(MysticDimensions.MOD_ID, name));
    }

    private static <FC extends FeatureConfig, F
            extends Feature<FC>> void register
            (Registerable<ConfiguredFeature<?, ?>> context,

             RegistryKey<ConfiguredFeature<?, ?>> key, F feature, FC configuration) {
        context.register(key, new ConfiguredFeature<>(feature, configuration));
    }
}
```

ModOrePlacement – цей клас відповідає за визначення модифікаторів розміщення руд у світі:

```
public class ModOrePlacement {
    public static List<PlacementModifier> modifiers
            (PlacementModifier countModifier, PlacementModifier heightModifier) {
        return List.of(countModifier, SquarePlacementModifier.of(),
                heightModifier, BiomePlacementModifier.of());
    }

    public static List<PlacementModifier> modifiersWithCount
            (int count, PlacementModifier heightModifier) {
        return modifiers(CountPlacementModifier.of(count), heightModifier);
    }
```

```
    public static List<PlacementModifier> modifiersWithRarity
            (int chance, PlacementModifier heightModifier) {
        return modifiers
                (RarityFilterPlacementModifier.of(chance), heightModifier);
    }
}
```

ModPlace Features – цей клас відповідає за визначення та реєстрацію розміщених функцій для генерації руди в світі:

```
public class ModPlacedFeatures {
    public static final RegistryKey<PlacedFeature>
            MYSTIC_DUST_ORE_PLACED_KEY = registerKey("mystic_dust_ore_placed");

    public static void boostrap(Registerable<PlacedFeature> context) {
        var configuredFeatureRegistryEntryLookup =
                context.getRegistryLookup(RegistryKeys.CONFIGURED_FEATURE);

        register(context, MYSTIC_DUST_ORE_PLACED_KEY,
                configuredFeatureRegistryEntryLookup.getOrThrow
                        (ModConfiguredFeatures.MYSTIC_DUST_ORE_KEY),
                ModOrePlacement.modifiersWithCount(12, // Veins per Chunk
                        HeightRangePlacementModifier.uniform
                                (YOffset.fixed(-1), YOffset.fixed(85))));
    }

    public static RegistryKey<PlacedFeature> registerKey(String name) {
        return RegistryKey.of(RegistryKeys.PLACED_FEATURE,
                new Identifier(MysticDimensions.MOD_ID, name));
    }

    private static void register
            (Registerable<PlacedFeature> context, RegistryKey<PlacedFeature> key,
             RegistryEntry<ConfiguredFeature<?, ?>> configuration,
                        List<PlacementModifier> modifiers) {
        context.register(key, new PlacedFeature(configuration, List.copyOf(modifiers)));
    }
```

ModOreGeneration – цей клас відповідає за додавання функцій генерації руди до біомів у світі.

```
public class ModOreGeneration {
    public static void generateOres() {
        BiomeModifications.addFeature(BiomeSelectors.foundInOverworld(),
                GenerationStep.Feature.UNDERGROUND_ORES,
                ModPlacedFeatures.MYSTIC_DUST_ORE_PLACED_KEY);
    }
}
```

Рис 2.13 Генерація руд у новому вимірі

ВИСНОВКИ

Оцінка одержаних результатів відносно аналогів

Мод Mystic Dimensions для Minecraft вносить значні покращення навіть в порівнянні з існуючими модифікаціями, надаючи унікальні розміри, різноманітні біоми та нові враження від гри. На відміну від багатьох інших модів, які пропонують поступові зміни або косметичні вдосконалення, Mystic Dimensions фундаментально змінює взаємодію гравця з ігровим світом. Цей мод додає глибини та різноманітності, перевершуючи аналогічні модифікації за масштабом та креативністю. Нові біоми, кожен з яких має відмінні екосистеми та ресурси, створюють нові виклики та можливості для дослідження, відрізняючи Mystic Dimensions від своїх сучасників.

Ступінь новизни

Mystic Dimensions демонструє, інтегрує абсолютно новий вимір у всесвіт Minecraft. Ці розміри є не просто розширенням існуючих біомів, а цілком оригінальними твореннями з унікальними умовами навколишнього середовища, флорою, фауною та геологічними утвореннями. Введення нових матеріалів і предметів для крафта ще більше відрізняє цей мод, пропонуючи гравцям безпрецедентні способи взаємодії з навколишнім середовищем і маніпулювання ним. Інноваційний підхід до побудови світу та механіки ігрового процесу є значним кроком вперед у модифікації Minecraft.

Практичне значення результатів

Практичне значення Mystic Dimensions полягає в його здатності підвищувати довговічність і відтворюваність Minecraft. Впроваджуючи нові виміри, біоми та ресурси, мод надає гравцям нескінченні можливості для дослідження та творчості. Це не тільки пожвавлює інтерес серед давніх гравців, але й залучає нову аудиторію, забезпечуючи постійну взаємодію з грою. Крім того, сумісність мода з існуючою ігровою механікою та іншими

модифікаціями робить його цінним доповненням для гравців, які хочуть розширити свій досвід Minecraft без шкоди для продуктивності чи стабільності.

Прогнозні припущення про подальший розвиток розроблення

Майбутній розвиток Mystic Dimensions може бути досить продуктивним. Подальші оновлення можуть представити більш складні екосистеми, динамічні погодні системи та інтерактивні NPC, щоб збагатити досвід гравців.

ЛІТЕРАТУРА

1. Майнкрафт вікі «Біоми» [Електронний ресурс]. – Режим доступу: https://minecraft.fandom.com/wiki/Biome.
2. Майнкрафт вікі «Моби» [Електронний ресурс]. – Режим доступу: https://minecraft.fandom.com/wiki/Mob
3. Майнкрафт вікі «Структури» [Електронний ресурс]. – Режим доступу: https://minecraft.fandom.com/wiki/Structure
4. Майнкрафт вікі «Крафт» [Електронний ресурс]. – Режим доступу: https://minecraft.wiki/w/Crafting
5. Майнкрафт вікі «Редстоун пил» [Електронний ресурс]. – Режим доступу: https://minecraft.fandom.com/wiki/Mob
6. Майнкрафт вікі «Кінець» [Електронний ресурс]. – Режим доступу: https://minecraft.fandom.com/wiki/The_End

Додаток А
Відомість матеріалів квадіфікаційної роботи

		Позначення	Найменування	Кільк. аркушів	Примітка
1					
2			Документація		
3					
4		**ІСТ.КР 24._.ДА.ПЗ**	Пояснювальна записка	61	
5					
6			Презентація	15	
7					
8			Диск CD з презентацією	1	

					ІТКІ.КР 24._.ДА.ПЗ
Зм.	Ар-куш	№ докум	Підпис	Дата	

Розроб.		Ю.В.Сердюк			**Матеріал кваліфікаційної роботи**	Літ.	Аркуш	Арку-шів
Керівник		К. Л. Сергєєва				Н	1	1
Рецензкнт						НТУ «ДП», 12; 126-20-1		
Н.контр.		Г.М. Коротенко						
Зав.каф.		В. В. Гнатушенко						