МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра «Системний аналіз та обчислювальна математика»
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

Бакалавра
(ступінь вищої освіти)

на тему: Системний аналіз та розробка агентно-орієнтованого середовища для моделювання динаміки ринку цифрових активів
(назва теми)

Виконав(ла): студент(ка) 4 курсу, групи КНТ-812

Спеціальності 124 Системний аналіз
(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інтелектуальні технології та прийняття рішень у складних системах

ВЛАСЕНКО Д. А.
(ПРІЗВИЩЕ та ініціали)

Керівник ТЕРЕЩЕНКО Е. В.
(ПРІЗВИЩЕ та ініціали)

Рецензент ЛОЗОВСЬКА Л.І.
(ПРІЗВИЩЕ та ініціали)

2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
Кафедра «Системний аналіз та обчислювальна математика»
Ступінь вищої освіти Бакалавр
Спеціальність 124 Системний аналіз
(код і найменування)

Освітня програма (спеціалізація) Інтелектуальні технології та прийняття рішень у складних системах
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри Терещенко Еліна Валентинівна

« 23 » квітня 2026 року

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ВЛАСЕНКО Даміра Андрійовича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) «Системний аналіз та розробка агентно-орієнтованого середовища для моделювання динаміки ринку цифрових активів»

керівник проєкту (роботи) к. ф.-м. н., доцент кафедри системного аналізу та обчислювальної математики ТЕРЕЩЕНКО Еліна Валентинівна

(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «23» квітня 2026 року № 196

2. Строк подання студентом проєкту (роботи) 05.06.2026

3. Вихідні дані до проєкту (роботи) Результати симуляцій; ттехнічна документація Steam; наукові публікації та дослідження з теми моделювання фінансових ринків.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

- Аналіз предметної області та огляд науково-методичної літератури за темою дослідження. Обґрунтування методу моделювання.
- Системний аналіз та проєктування середовища дослідження.
- Програмна реалізація моделі ринку цифрових активів.
- Проведення серії обчислювальних експериментів, збір статистичних даних та аналіз результатів моделювання.

- Дослідження впливу ключових параметрів моделі на ринкову динаміку, узагальнення результатів і формулювання висновків.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів) Презентація обсягом 20 слайдів.

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
Розділи 1-3	Терещенко Е.В.	02.03.2026	08.06.2026
Норконтроль	Широкорад Д.В.	04.05.2026	

7. Дата видачі завдання «2» березня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Огляд наукової літератури	3 тижні	виконано
2	Аналіз предметної області	1 тиждень	виконано
3	Програмна реалізація моделі ринку цифрових активів	4 тижні	виконано
4	Проведення серії обчислювальних експериментів та їх аналіз	2 тижні	виконано
5	Формулювання висновків щодо впливу ключових параметрів моделі	1 тиждень	виконано
6	Оформлення пояснювальної записки до роботи	2 тижні	виконано

Студент(ка) _____ **Дамір ВЛАСЕНКО**
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ **Еліна ТЕРЕЩЕНКО**
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Дипломна робота: 77 стор., 40 рис., 13 табл., 21 джерел.

У дипломній роботі розглянуто задачу моделювання динаміки ринку цифрових активів Steam Community Market методом агентно-орієнтованого моделювання. Проведено серію симуляцій із різними параметрами для виявлення їхнього впливу.

Об’єкт дослідження — ринок цифрових активів Steam Community Market.

Предмет дослідження — закономірності ціноутворення, динаміки цін, розподілу добробуту між учасниками ринку, обсягів торгів і суми комісійних зборів залежно від значення комісії та структури учасників ринку.

Метою роботи — системний аналіз та розробка параметризованого середовища агентно-орієнтованого моделювання для дослідження впливу ринкових механізмів, правил і структури учасників на динаміку ціни цифрових предметів CS2, обсяги торгів і прибуток, зароблений з комісії платформи.

Актуальність зумовлена закритістю досліджуваної платформи для зовнішнього аналізу, гетерогенністю учасників ринку та відсутністю обчислювальних моделей або досліджень щодо впливу ринкових механізмів платформи на ціноутворення, ліквідність та обсяг комісійних зборів.

Практична цінність результатів полягає в розробленому програмному забезпеченні — моделі реального ринку з продажу цифрових активів, що дозволяє досліджувати структуру системи, тестувати різні нові підходи чи зміни в правилах і механізмах системи, будувати графіки з візуалізацією результатів.

Роботу також було представлено на Всеукраїнському конкурсі студентських наукових робіт.

АГЕНТНО-ОРІЄНТОВАНА МОДЕЛЬ; STEAM COMMUNITY MARKET; CS2; КОМІСІЯ; TRADE-LOCK; ЛІКВІДНІСТЬ; ЕЛАСТИЧНІСТЬ; МОНТЕ-КАРЛО; ДРОП; СИМУЛЯЦІЇ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

АОМ — скорочення від агентно-орієнтована модель.

Грош. од. — скорочення від «грошові одиниці».

Дроп (англ. drops) — внутрішньоігрова винагорода, видана за певну активність.

Лутбокс (англ. loot box) — віртуальний предмет, контейнер, що містить випадковий внутрішньоігровий предмет.

Матчинг (англ. matching) — механізм, що узгоджує протилежні замовлення за певним правилом.

Ордер (англ. order) — замовлення на купівлю чи продаж віртуального актива.

Скін (англ. skin) — в контексті гри візуально-косметичний предмет, який змінює зовнішній вигляд ігрового предмета.

CS2 — відеогра, шутер від першої особи, розроблена компанією Valve.

ROI (Return on Investment) — фінансовий показник, що вимірює прибутковість активу.

Steam — платформа, що надає послуги з цифрової дистрибуції комп’ютерних ігор.

Steam Community Market — офіційний ринок Steam для продажу чи купівлі внутрішньоігрових предметів.

ВСТУП

Ринки цифрових активів є відносно новим і динамічним об'єктом дослідження на перетині інформаційних систем та економіки. Steam — найбільша у світі платформа з дистрибуції комп'ютерних ігор, розроблена компанією Valve Corporation. Щодня на платформі одночасно грає понад 38 мільйони користувачів. За даними проєкту SteamDB щомісяця виходить близько двох тисяч нових ігор на Steam. Це не просто цифровий сервіс, а окрема екосистема для гравців різних вподобань. Платформа давно вийшла за межі цифрового магазину, що спеціалізується лише на продажі комп’ютерних ігор. Вона являє собою окрему екосистему з соціальними функціями, майстернею ігрового контенту та магазином з продажу техніки, розробленою Valve (а саме портативні консолі, VR-шолом, ігрові контролери).

Головною особливістю цієї платформи є ринок спільноти (Steam Community Market). Він дозволяє гравцям торгувати внутрішньоігровими предметами за кошти гаманця Steam. Предмети, що існують виключно у віртуальному світі гри, мають реальну економічну цінність.

Найбільшою економікою в Steam є косметичні предмети CS2: ігрова зброя з різними текстурами (скіни), наліпки, ігрові персонажі, контейнери з предметами. Counter-Strike 2 — тактичний шутер від першої особи, розроблений Valve, — найпопулярніша франшиза компанії з понад 26-річною історією. Станом на 7 травня 2025 року загальна капіталізація предметів CS2 склала п'ять мільярдів доларів США за даними ресурсу Pricempire. Водночас ринок CS2 характеризується стабільним зростанням попри нестабільність на традиційних фінансових ринках. На відміну від них, він не залежить від макроекономічних і політичних чинників. На його динаміку впливають: настрій гравців, частота та якість ігрових оновлень.

Ринок предметів CS2 є унікальним об’єктом дослідження завдяки специфічним внутрішнім механізмам і правилам появи предметів, що принципово відрізняють його від класичних фінансових ринків. Зокрема: комісія платформи, втримування предметів протягом семи днів після отримання, система щотижневої випадкової винагороди за активну гру та інші обмеження. Сукупність цих правил формує замкнену ринкову екосистему зі своїми закономірностями, дослідження яких дозволяє виявити якісний вплив окремих параметрів на поведінку системи в цілому.

РОЗДІЛ 1 АНАЛІЗ ЛІТЕРАТУРИ ТА МЕТОДІВ ДОСЛІДЖЕННЯ

1.1 Цифрові активи та ринок Steam Community Market

Розвиток цифрових технологій зумовив появу принципово нового класу активів — цифрових віртуальних предметів, що обертаються існують всередині відеоігор. На відміну від традиційних фінансових інструментів, такі активи не мають фізичного втілення, однак мають реальну грошову вартість, прикріплену за ними, ціноутворення якої обумовлене попитом і пропозицією. Ринок скінів Counter-Strike 2 (CS2) є одним із найбільших і найліквідніших у сегменті ігрових віртуальних предметів: за даними Bloomberg, у березні 2025 року він досяг нового абсолютного рекорду ринкової капіталізації [1].

Торговельний майданчик Steam Community Market, запущений компанією Valve Corporation у 2013 році, виступає центральним середовищем для купівлі та продажу предметів ігор Steam [2]. Важливою особливістю платформи є фіксована комісія на рівні 5% від суми кожної угоди та комісія розробника гри (10% — для ігор Valve), яка стягується з продавця в момент здійснення транзакції [2]. Ця ставка є незмінною з моменту запуску майданчика, що принципово обмежує можливості її емпіричного дослідження методами, що потребують спостереження за варіаціями у реальних даних.

Вартість предметів залежить від рідкості, зовнішнього вигляду, стану предмета. Розподіл цін є вкрай нерівномірним: більшість предметів торгується за доступною ціною у великій кількості кожного дня, тоді як унікальні та рідкісні предмети можуть сягати тисяч доларів і майже не з’являються на ринку. Сервіс CSROI [3] дає змогу оцінити математичне очікування вартості вмісту контейнера і порівняти його з ринковою ціною самого контейнера — у переважній більшості випадків відкриття є невигідним, а ринкова ціна контейнера вища за очікуваний виграш з урахуванням комісій. Відкриття

контейнерів є одним з основних шляхів отримання предметів в економіці CS2. Окрім отримання комісійного збору від продажу новостворених предметів на торгівельному майданчику Steam, компанія-розробник додатково заробляє оціночно 1 мільярд доларів США щороку, продаючи ключі для відкриття контейнерів.

1.2 Гетерогенність учасників ринку цифрових предметів

Протягом існування ринку Steam Community Market на ньому створилися принципово особливі типи учасників, мотивації яких суттєво відрізняються. Lehdonvirta [4] у дослідженні ринків віртуальних предметів виділяє три базові класи мотивації: функціональну (предмет покращує ігровий досвід), соціальну (демонстрація статусу іншим гравцям) та економічну (очікування зростання вартості). Ці мотивації формують якісно різні поведінкові патерни.

Дослідження Thorhauge і Nielsen [5] аналізує роль скін-ставок і торгівлі в економіці ігрових платформ і показує, що навіть серед фінансово мотивованих учасників існують суттєві відмінності: короткострокові спекулянти (flipper-traders), що заробляють на спредах між купівлею і продажем, та довгострокові інвестори, що утримують предмети роками в очікуванні зростання цін.

Yuan [6] у дослідженні факторів, що визначають ціни скінів CS:GO, виявляє, що рідкість і категорія предмета пояснюють значну частку цінової варіації, однак ринок все ще схильний до шумових коливань, що генеруються поведінкою некваліфікованих учасників. Окрім цього, на ціноутворення предметів впливає популярність гри, а деякі типи предметів залежать від ігрових оновлень. Це показує, що ринкова ціна визначається не лише об'єктивними характеристиками предмета, а й психологічними та

контекстуальними факторами, що підкреслює доцільність моделювання поведінки ринку, описуючи логіку користувачів, механізми платформи та умови появи предметів. Такий дизайн відповідає підходам, запропонованим у роботі Hommes [7], де гетерогенність агентів розглядається як ключовий чинник формування реалістичної ринкової динаміки.

1.3 Агентно-орієнтоване моделювання

Агентно-орієнтоване моделювання (АОМ) є обчислювальним підходом до вивчення складних систем, у якому поведінка системи досліджується через взаємодію автономних агентів — програмних об'єктів із власними правилами поведінки, станом і здатністю до взаємодії із середовищем та між собою. На відміну від рівняннєвих підходів (диференціальні рівняння, системна динаміка), АОМ дозволяє формуватися поведінці системи знизу вгору через індивідуальні дії агентів.

Macal і North [8] у своєму базовому туторіалі з АОМ визначають три обов'язкові характеристики агента: автономність (здатність діяти незалежно від зовнішнього управління), соціальність (здатність взаємодіяти з іншими агентами) та адаптивність (здатність змінювати поведінку залежно від стану середовища). Типи агентів у розробленій моделі відповідають цим характеристикам: кожен агент самостійно приймає рішення на основі доступної ринкової інформації та власного стану (баланс, інвентар, тип).

Tesfatsion і Judd [9] систематизують методологію агентно-орієнтованих обчислювальних економік (ACE — Agent-Based Computational Economics) і підкреслюють, що цей підхід особливо ефективний для моделювання ринків, де агенти є гетерогенними. Як описано раніше, саме така ситуація характерна для досліджуваної системи Steam Community Market: постійний приплив нових предметів через активність гравців, непостійна кількість активних агентів та

їхня відмінність і неповторність унеможливлюють застосування класичних рівноважних моделей, моделей що описують систему у стані балансу та рівноваги.

Farmer і Foley [10] у своїй роботі у журналі Nature аргументують необхідність переходу від агрегованих макроекономічних моделей до АОМ для вивчення реальних економічних систем, зокрема фінансових ринків. Автори зазначають, що стандартні економічні моделі базуються на нереалістичних припущеннях про раціональність агентів і досягнення рівноваги, тоді як АОМ здатне відтворити спостережувані феномени, як цінові бульбашки, — без попереднього опису такої поведінки.

Орієнтиром для моделювання взято роботу Todd та ін. [11] у огляді якої систематизують наявні підходи агентно-орієнтованого моделювання і виділяють ключові характеристики успішних АОМ фінансових ринків: реалістична мікроструктура ринку (механізм матчингу заявок), гетерогенність агентів за типом стратегії та рівнем інформованості, ендогенне ціноутворення (ціна формується всередині системи, а не задається ззовні). Розроблена модель відповідає всім трьом критеріям: реалізовано повноцінну книгу заявок з пріоритетом за ціною і часом, чотири типи агентів із різними стратегіями, ціна формується як результат фактичних угод.

1.4 Метод Монте-Карло

Оскільки агентно-орієнтована модель є стохастичною — поведінка агентів містить випадкові компоненти та шум — одиничний запуск симуляції не є достовірним і не може бути основою для статистичних висновків. Для отримання надійних оцінок параметрів застосовується метод Монте-Карло: багаторазовий запуск симуляції з різними початковими станами генератора випадкових чисел і усереднення результатів.

1.5 Порівняльний аналіз методів моделювання

Класичні статистичні методи (описова статистика, кореляційний аналіз) дозволяють охарактеризувати наявні дані, але не надають інструментів для аналізу інших сценаріїв. Оскільки комісія на ринку Steam була постійною та жодного разу не змінювалася, класичні методи не здатні відповісти на запитання «що відбудеться, якщо комісію змінити».

Сюди ж відносяться методи регресійного аналізу, що потребують варіації в ключовій незалежній змінній. Відсутність історичної зміни значення комісії на платформі Steam не дає змогу кількісно оцінити ефект комісії на поведінку ринку.

Методи машинного навчання здатні виявляти складні нелінійні залежності між характеристиками предмета і його ринковою ціною, однак є безрезультативними для аналізу впливу внутрішніх механізмів платформи. Як зазначалося раніше, Guede-Fernández та ін. [12] застосовують алгоритми машинного навчання для прогнозування цін предметів CS2 і досягають обмеженої точності, що підтверджує складність і непередбачуваність цього ринку.

Методи системної динаміки виражають поведінку системи у вигляді диференціальних рівнянь, що унеможливлює моделювання гетерогенної поведінки окремих учасників. Така модель могла б відтворити загальну динаміку цін, але не здатна відтворити відмінності між різними типами агентів і їх взаємодію.

В результаті було обрано поєднання методу агентно-орієнтованого моделювання та Монте-Карло, що забезпечує реалізацію поставленої мети дослідження системи. Порівняльний аналіз методів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння методів моделювання ринку цифрових активів

Метод аналізу	Гетерогенність агентів	Аналіз «What If?»	Моделювання контрфактуальних умов	Відтворення структури ринку
Описова статистика	Не враховується	Неможливий	Неможливе	Неможливе
Регресійний аналіз	Не враховується	Обмежений	Потребує варіації даних	Неможливе
Аналіз часових рядів	Не враховується	Обмежений	Неможливе	Неможливе
Машинне навчання	Не враховується	Обмежений	Обмежене	Неможливе
Системна динаміка	Не враховується	Частковий	Обмежене	Неможливе
АОМ	Повністю враховується	Необмежений	Повністю реалізоване	Можливе

Метод агентно-орієнтованого моделювання є сучасним і одним із найкращих підходів для симуляції поведінки складних соціальних систем, таких як ринок спільноти Steam. У роботі Axtel [13] він розглядається як один із фундаментальних, що дозволяє досліджувати емерджентні властивості системи через поведінку агентів.

РОЗДІЛ 2 ФІЗИЧНЕ МОДЕЛЮВАННЯ

2.1 Загальна архітектура моделі

Розроблене програмне забезпечення являє собою агентно-орієнтовану модель ринку спільноти Steam, реалізовану мовою програмування Python [14]. Існує готове рішення для побудови агентно-орієнтованих моделей мовою програмування Python – фреймворк MESA [15], однак у даній роботі, розроблено модель з повного нуля з метою точного відтворення унікальної мікроструктури ринку Steam. Середовище для досліджень побудоване за принципом модульної архітектури: кожен незалежний аспект системи виділений в окремий модуль. Таке розмежування спрощує тестування та подальший розвиток моделі, підтримує чистоту коду. Кожен модуль відповідає виключно за власну логіку. Склад модулів програмного середовища наведено у таблиці 2.1.

Таблиця 2.1 – Структура модулів розробленої моделі

Модуль	Призначення
models.py	Визначення типів даних і структур: NewType-аліаси, датакласи Sale, Order, MarketHashName, AgentType та ін.
market.py	Середовище симуляції, відповідає за відповідне узгодження замовлень, розрахунок комісії, тимчасового блокування предметів
drop_generator.py	Відповідає за генерацію предметів всередині моделі
agents.py	Абстрактний клас агентів і чотири підкласи
metrics.py	Методи аналізу: медіана ціни, середньозважена ціна, обсяг торгів, сукупна комісія

constants.py	Глобальні константи: ONE_CENT, ONE_DOLLAR, MIN_PRICE, MIN_FEE та ін.

Сувора статична типізація на рівні всього проєкту забезпечується за допомогою вбудованого інструменту Pylance у найстрогішому режимі. Це дозволяє виявити приховані помилки на етапі розробки та покращує читабельність коду.

Додатковий код, що надає готові рішення з візуалізації, знаходиться у модулі plots.py пакета visualization, створеному в тій самій директорії проєкту.

Часова структура моделі є дискретною: кожен крок симуляції відповідає одній дії агента. Це також не гарантує, що дія завершиться успішно, адже агент може не мати достатнього балансу, кількість предметів для продажу або на ринку будуть відсутні предмети для купівлі. За замовчуванням одна симуляційна доба триває 1000 кроків. На кожному кроці обирається випадково один агент з усієї популяції, що діятиме. Таким чином, апроксимується асинхронна поведінка учасників ринку, що відповідає реальній природі цифрових торговельних платформ.

2.2 Модуль моделей даних (models.py)

Модуль models.py відповідає за опис ключових сутностей симуляції. В ньому визначено основні моделі, типи та структури даних. Далі наведено основні з них.

2.2.1 Ідентифікатори та енумератори

За допомогою `typing.NewType` створено три нові типи даних для опису унікальних ідентифікаторів (див. рис. 2.1), які використовуються як ключі в

словниках. `OrderID` – унікальний ідентифікатор замовлення, `AgentID` – унікальний ідентифікатор агента, а `MarketHashName` – унікальний ринковий ідентифікатор предмета, що відповідає внутрішній номенклатурі Steam і використовується як ключ у всіх словниках ринку. На відміну від alias, статичний аналізатор побачить, що це новостворений тип, і повідомить про помилку, якщо буде передано int замість `OrderID`. Хоча Python є динамічно типізованою мовою та ця сувора типізація не впливає на виконання коду, вона суттєво покращує його читабельність та розробку.

```
OrderID = NewType("OrderID", int)
AgentID = NewType("AgentID", int)
MarketHashName = NewType("MarketHashName", str)
```

Рисунок 2.1 – Зображення нових типів

`OrderType` – структура типу `Enum`, визначає лише два можливих типи замовлення: `Buy` та `Sell`.

`AgentType` – успадковує структуру `StrEnum`, що дозволяє використовувати значення енумератора безпосередньо як рядкові ключі словників і читати їх при збереженні результатів як звичайні рядки замість посилання на об’єкт (див. рис. 2.2).

```
class AgentType(StrEnum):  33 usages  Damir
    NOVICE = 'Novice'
    TRADER = 'Trader'
    INVESTOR = 'Investor'
    FARMER = 'Farmer'
```

Рисунок 2.2 – Енумератор типів агентів

Додатково для опису категорії певного предмета було створено `ItemCategory`. Він описує тип предмета: контейнер, скін, наліпка чи інше. `ItemRarity` відтворює дев'ять рівнів рідкості предметів в економіці CS2. `WeaponExterior` описує п'ять основних градацій зносу якості зброї, відповідно до внутрішньоігрових назв.

2.2.2 Ієрархія класів предметів

Абстрактний клас `MarketItem` є базою для всіх предметів, що можуть торгуватися на ринку (див. рис. 2.3). Він описує їх загальні атрибути, такі як name, rarity, category, market_hash_name. Атрибут `market_hash_name` є унікальним ідентифікатором предмета та оновлюється при його створенні, за замовчуванням він рівний назві предмета. Використання (`frozen=True`) у декораторі забезпечує незмінність даних після створення.

```python
@dataclass(frozen=True, slots=True)  27 usages  Damir *
class MarketItem(ABC):
    """
    Base abstract class for all tradable items in the simulation.

    Provides general attributes shared by every item type.
    `market_hash_name` is a canonical string identifier that matches
    Steam's internal naming scheme (used as a key in dictionaries).
    """
    name: str
    rarity: ItemRarity
    category: ItemCategory
    market_hash_name: MarketHashName = field(init=False)

    def __post_init__(self):  Damir
        object.__setattr__(self, __name: 'market_hash_name', self.name)
```

Рисунок 2.3 – Абстрактний клас MarketItem

`MarketItem` наслідується іншими класами, що описують вузьку логіку різних типів предметів: `Container, WeaponSkin.`

2.2.3 Структури інвентаря та транзакцій

Клас `InventoryItem` зберігає запис про предмет в інвентарі агента: посилання на об'єкт item, кількість quantity та крок unlock_step, після якого предмет стає доступним для продажу.

Клас `Order` описує активне замовлення у книзі ордерів: type (OrderType), item, price, quantity, agent_id, step (крок розміщення) і автоматично генерований id через лічильник `itertools.count()`. Замовлення може змінюватися з часом, а отже, прапорець frozen тут не використовується (див. рис. 2.4).

```
@dataclass(slots=True)  13 usages  Damir
class Order:
    type: OrderType
    item: MarketItem
    price: int
    quantity: int
    agent_id: AgentID
    step: int
    id: OrderID = field(default_factory=lambda: OrderID(next(_order_id)))
```

Рисунок 2.4 – Модель замовлення

Клас `Sale` фіксує завершену угоду, а тому використовує (`frozen=True`) (див. рис. 2.5). Використання (`slots=True`) в датакласі дозволяє оптимізувати код, не створювати динамічний словник для кожного об'єкта, зменшити споживання оперативної пам'яті в умовах багатьох паралельних симуляцій та пришвидшити звертання за атрибутами об'єкта. Інформація про угоду зберігається в полях: item, price, total_fee, quantity, buyer_id, seller_id, step, id.

```python
@dataclass(frozen=True, slots=True)  7 usages  Damir
class Sale:
    item: MarketItem
    price: int
    total_fee: int
    quantity: int
    buyer_id: AgentID
    seller_id: AgentID
    step: int
    id: int = field(default_factory=lambda: next(_sale_id))
```

Рисунок 2.5 – Модель продажа

Ціни та суми з комісії зберігаються як int (цілі числа в центах), а не float. Дане рішення є свідомим, адже арифметика цілих чисел є точною та детермінованою, коли для чисел з плаваючою комою притаманна похибка при округленні. Використання типу даних int замість `Decimal` забезпечує ту саму точність при значно меншій вартості обчислення. Дана перевага є суттєвою при виконанні великої кількості симуляцій.

2.2.4 Псевдоніми складних типів

Наприкінці модуля визначено TypeAlias-псевдоніми, що описують складні типи структур в одному місці та позбавляють від повторів анотацій у інших модулях (див. рис. 2.6).

```python
SalesHistory: TypeAlias = DefaultDict[MarketHashName, list[Sale]]
AgentMarketHistory: TypeAlias = DefaultDict[AgentID, list[Sale]]
AgentBuyOrderIndex: TypeAlias = dict[AgentID, dict[MarketHashName, OrderID]]
```

Рисунок 2.6 – Псевдоніми складних типів

2.3 Модуль ринкового середовища (market.py)

Клас `Market` є центральним компонентом моделі, середовищем симуляції. Він відтворює внутрішню бізнес-логіку Steam Community Market (див. рис. 2.7), разом із його правилами та механізмами: зберігає список ордерів, виконує матчинг замовлень, розраховує комісію з кожної угоди, оновлює баланси агентів, передає предмети між інвентарями, застосовує торгові блокування та зберігає повну історію продажів. Повний перелік атрибутів наведено у таблиці 2.2.

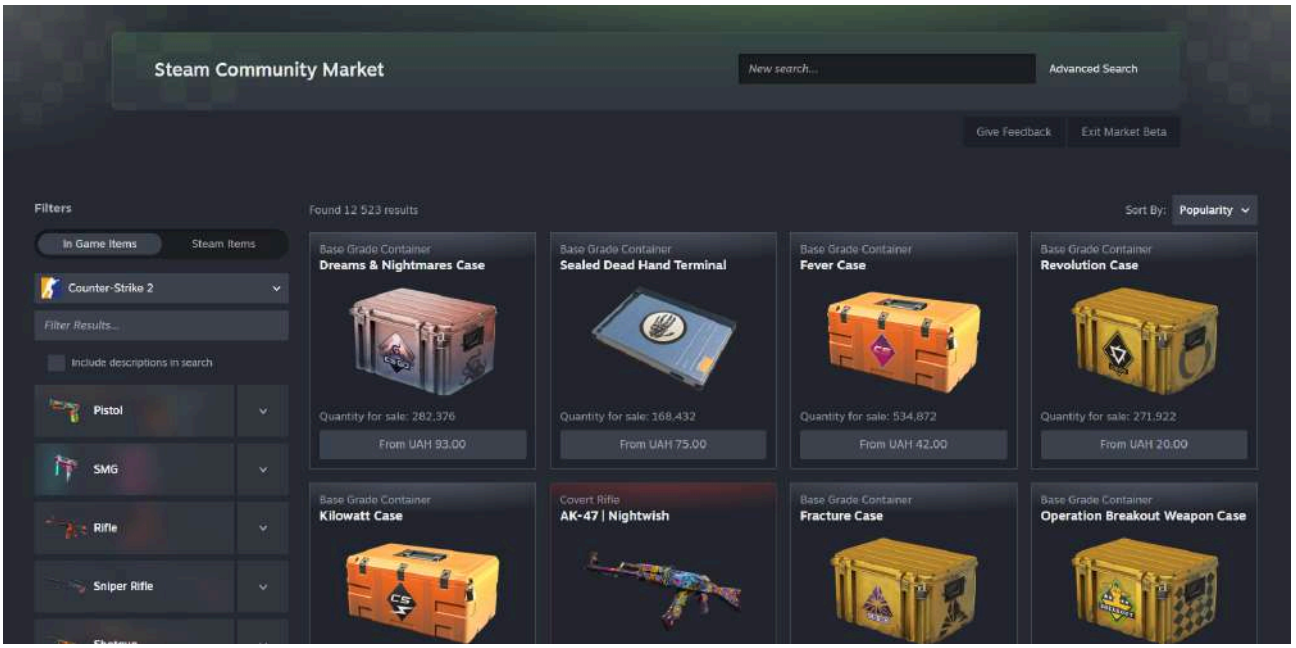

Рисунок 2.7 – Інтерфейс головної сторінки торговельного майданчика Steam Community Market

Таблиця 2.2 – Атрибути класу Market

Атрибут	Призначення
market_fee	Значення комісії платформи (частка від суми угоди)
steps_per_day	Кількість кроків, що відповідає одному дню симуляції
trade_lock_period	Тривальість торгового блокування (у симульованих днях)
lock_on_purchase	Прапорець, що визначає застосування тимчасового блокування до предметів куплених на ринку
max_balance	Максимально допустимий баланс агента
current_step	Поточний крок симуляції
agents	Реєстр усіх агентів
buy_orders	Книга ордерів на купівлю
sell_orders	Книга ордерів на продаж
agent_buy_orders_idx	Індекс замовлень на купівлю по агентам
items_map	Реєстр предметів із активними замовленнями на продаж
sales_history	Список усіх продажів по предметах
agent_purhases	Список купівель агента
agent_sales	Список продажів агента

При ініціалізації класу параметр `max_balance` отримує значення у доларах і одразу конвертує його у центи множенням на константу `ONE_DOLLAR`. Це відповідає загальній архітектурі моделі та більшості фінансових систем, де грошові одиниці представляються цілими числами в центах.

2.3.1 Структура книги ордерів

Книга ордерів є найбільш навантаженою структурою даних у моделі, адже вона безперервно поповнюється новими записами, ордери оновлюються при кожному матчингу та видаляються при їх виконанні. Тому було вирішено для реалізації використовувати структуру типу SortedList із бібліотеки sortedcontainers [16]. Порівняльний аналіз альтернатив наведено у таблиці 2.3.

Таблиця 2.3 – Порівняння різних структур даних для реалізації книги ордерів

Структура даних	Вставка	Видалення	Пошук
list	$O(n)$	$O(n)$	$O(logn)$ (якщо відсортований)
heapq	$O(logn)$	$O(n)$	$O(n)$ (будь-який елемент)
PriorityQueue	$O(logn)$	$O(n)$	$O(n)$ (будь-який елемент)
SortedList	$O(logn)$	$O(logn)$	$O(logn)$ (bisect)

Як видно з таблиці 2.3, SortedList є єдиною структурою, яка одночасно забезпечує $O(logn)$ складність операції для вставки, видалення та бінарного пошуку за довільним критерієм. Стандартний list потребує складності $O(logn)$ для вставки у вже відсортованому списку та повторного сортування після цього $O(nlogn)$ за допомогою `list.sort()`. Структури heapq та PriorityQueue є корисними для додавання чи видалення елементів на кінцях (мінімальний чи максимальний елемент), тому видалення довільного елемента (наприклад, при виконанні ордера або скасуванні агентом) займає $O(n)$.

Ключова функція сортування легко задається при ініціалізації відсортованого списку (SortedList). Для ордерів на покупку:

`SortedList(key=lambda o: (-o.price, o.step))`, де o – екземпляр ордеру (Order).

Від'ємне значення ціни забезпечує сортування за її спаданням – замовлення з найвищою ціною є найпершим. Для однакових значень ціни відбувається сортування за другим параметром – кроком розміщення. Для ордерів на продаж – навпаки, сортування відбувається за зростанням ціни, першим замовлення стоїть найдешевше:

```
SortedList(key=lambda o: (o.price, o.step))
```

Таке архітектурне рішення відповідає внутрішньому механізму матчингу ордерів на Steam Market, детальніше про нього у наступному розділі.

Додатково створено допоміжний індекс `agent_buy_orders_idx`, який зберігає ордери на предмети відповідно до кожного агента. Завдяки ньому перевірка дубліката замовлення відбувається без лінійного пошуку списком усіх замовлень.

Conn та ін. [17] запропонували у своїй роботі стохастичну модель динаміки книги ордерів, на основі незалежних пуассонівських процесів, що відтворює короткострокову поведінку ринку без явних припущень. На відміну від цього підходу, розроблена модель описує поведінку агентів явно, що в результаті дозволяє досліджувати їх вплив на динаміку ринку та формування замовлень.

2.3.2 Алгоритм матчингу ордерів

Матчинг ордерів реалізовано у двох приватних методах: `_get_matching_sell_orders()` та `_get_matching_buy_orders()`. Обидва методи використовують бінарний пошук по відсортованому списку замовлень для отримання зрізу ордерів, що задовольняють ціні.

Метод `_get_matching_sell_orders()` знаходить усі ордери на продаж із ціною, що не перевищує задану ціну купівлі. Для цього створюється штучний dummy-ордер із відповідною ціною, після чого викликається метод

бінарного пошуку правої межі зрізу (`bisect_right`). Отриманий зріз ордерів гарантує, що кожен із них є дешевшим за ціну купівлі.

Метод `_get_matching_buy_orders()` аналогічно знаходить ордери, що задовольняють ціні продажу предмета, а отриманий зріз замовлень додатково сортується за зростанням кроку (step). Це гарантує, що результат задовільняючих замовлень відсортований від найдавнішого. Така реалізація повністю відтворює виконання замовлень на купівлю за часовим пріоритетом. Відповідно до опису узгодження ордерів на купівлю в довідці Steam Community FAQ: «Тому тепер, якщо є кілька пропозицій для щойно виставленого предмета, буде вибрано найстарішу з них» (див. рис. 2.8).

Рисунок 2.8 – Алгоритм вибору замовлень на купівлю Steam Community FAQ

Обидва методи приймають параметр exclude_agent_id, що запобігає самоторгівлі: агент не може виконати власне замовлення. Це відтворює реальне обмеження на ринку спільноти Steam.

Приватні методи матчингу ордерів використовуються публічними методами buy() та sell(), для купівлі чи продажу предмета відповідно. Загальний алгоритм методу buy():

1. Перевірка на дублікат ордерів купівлі через agent_buy_orders_idx;
2. Перевірка достатності балансу покупця;
3. Отримання відповідних ордерів на продаж через _get_matching_sell_orders();

4. Для кожного знайденого ордера обчислюється максимально допустима кількість через метод _max_receivable_quantity() із урахуванням обмеження на максимальний баланс продавця агентів;

5. Розрахунок комісії, оновлення балансів, передача предметів у інвентар відповідного покупці з застосуванням Trade-lock, фіналізація продажу у історії;

6. У випадку якщо невдалося повністю купити бажану кількість предметів – створюється замовлення на купівлю із нею.

Метод sell() працює симетрично: спочатку вилучаються предмети з інвентаря продавця, отримуються відповідні замовлення на купівлю, що задовольняють ціні продажу, виконуються угоди та створюється замовлення на продаж із залишковою кількістю.

2.3.3 Розрахунок комісії та обмеження балансу

Комісія розраховується з кожної угоди за допомогою метода calculate_fee() за формулою (2.1):

$$F = max\left(\lfloor O \cdot f_{market} \rfloor, F_{min}\right), \quad (2.1)$$

де F – загальна сума комісії з угоди;

O – загальна вартість замовлення;

f_{market} – ринкова комісія;

F_{min} – мінімально можлива комісія (константа);

$\lfloor \rfloor$ – операція округлення числа до цілої частини.

Метод _max_receivable_quantity() забезпечує обмеження максимального балансу продавця. Якщо угода призведе до перевищення продавцем

визначеного ліміту max_balance, кількість проданих предметів буде автоматично зменшена до допустимого рівня. Це дає змогу моделювати реальне обмеження гаманця Steam (максимальний баланс — 2 000 доларів США). У разі перевищення ліміту ордер видаляється зі списку за допомогою методу _remove_buy_order().

2.3.4 Торгове блокування Trade-lock

На ринку спільноти Steam діє обмеження щодо обміну та продажу предметів. Після отримання предмети неможливо обміняти чи продати протягом 7 днів. Ця система створена з метою збереження та убезпечення безпеки предметів користувачів, захисту та називається Trade-lock.

Крок розблокування предмета обчислюється методом `calculate_unlock_step()` за формулою (2.2), якщо параметр `is_trade_lock = True`, якщо ні, то `unlock_step` рівний 0:

$$S_{unlock} = S_{current} + T_{lock} \cdot N_{steps} \quad (2.2)$$

де S_{unlock} – результуючий крок розблокування предмета;

$S_{current}$ – поточний крок симуляції;

T_{lock} – тривалість періоду блокування;

N_{steps} – кількість симуляційних кроків, що відповідають одному дню.

При `lock_on_purchase = True` кожен придбаний на ринку предмет отримує застосування блокування. Аналогічно блокується винагорода, отримана генератором предметів.

2.3.5 Публічні методи ринку для взаємодії агентів

Окрім попередньо згаданих методів, клас Market надає агентам повний набір методів для взаємодії та отримання стану ринку:

- `get_item_buy_orders()` — повертає активні замовлення на купівлю певного предмета;
- `get_item_sell_orders()` — повертає активні замовлення на продаж певного предмета;
- `get_item_recent_sales()` — повертає список останніх продажів для предмета;
- `get_available_items()` — повертає список доступних до купівлі предметів, з опціональним фільтром категорії (ItemCategory);
- `get_agent_orders()` — повертає всі активні ордери агента, відфільтровані за типом;
- `get_agent_purchases()` та `get_agent_sales()` — повертають торгову історію агента.

Скасування ордерів реалізовано через методи `cancel_buy_order()` та `cancel_sell_order()`. При скасуванні ордера на купівлю метод _remove_buy_order() видаляє відповідний об’єкт зі списку замовлень та очищає запис у допоміжному індексі `agent_buy_orders_idx`, що відновлює можливість агентам розміщати нові замовлення на ті самі предмети. А при скасуванні ордера на продаж – предмети повертаються в інвентар агента.

Таким чином, клас Market реалізує повний цикл ринкових транзакцій Steam Community Market: від валідації даних, перевірки умов, до вірної передачі ігрових предметів і запису історії продажів. Обмеження та відтворення внутрішньої бізнес-логіки Steam Community Market повністю реалізовані.

2.4 Модуль генератора предметів (drop_generator.py)

2.4.1 Джерела появи предметів в економіці CS2

Станом на 2026 рік в ігровій економіці CS2 є кілька основних джерел появи предметів. Детально про кожен з них:

Weekly Care Package — механізм заохочення користувачів до активної гри [18]. Представляє собою регулярну щотижневу винагороду, де гравцям при підвищенні рівня акаунта пропонується на вибір два предмети з чотирьох випадкових варіантів (див. рис. 2.9). Запропонована нагорода генерується випадковим чином, а параметри та ймовірності визначаються розробником гри. Кожен гравець обмежений лише однією винагородою на тиждень для одного облікового запису (акаунта). Обмеження спадає щовівторка о 18:00 за тихоокеанським часом (PT). Окремі категорії користувачів використовують автоматизацію з метою зловживання цією системою та масового отримання безкоштовних предметів.

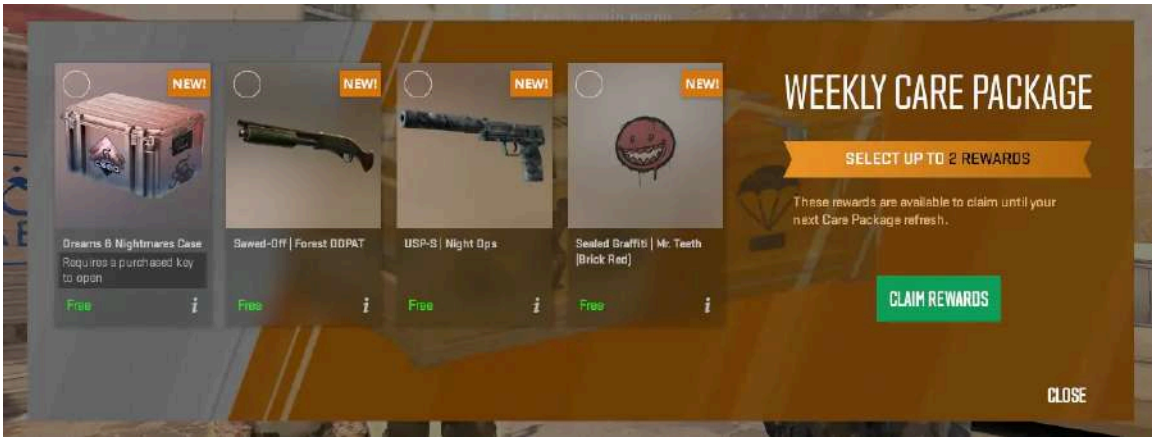

Рисунок 2.9 – Вікно щотижневої винагороди

Контейнери (Container) — лутбокси, предмети, що відкриваються за допомогою окремого ключа або безкоштовно та видають випадковий предмет однієї з рідкостей із зафіксованими ймовірностями відповідно до рідкості.

Можуть бути кейсами зі зброєю або капсулами з наліпками чи нашивками (див. рис. 2.10). Розподіл ймовірностей був офіційно оприлюднений компанією Valve через вимоги законодавства КНР щодо розкриття шансів у лутбоксах.

Рисунок 2.10 – Вміст контейнера та ймовірності випадіння предметів

Контракти обмінів (Trade-Up Contract) — ігровий інструмент, доступний усім гравцям, що дозволяє обміняти 10 предметів однієї рідкості з різних колекцій — на один випадковий предмет вищої рідкості з однієї з використаних колекцій. Після підтвердження операції використані предмети видаляються назавжди (див. рис. 2.11).

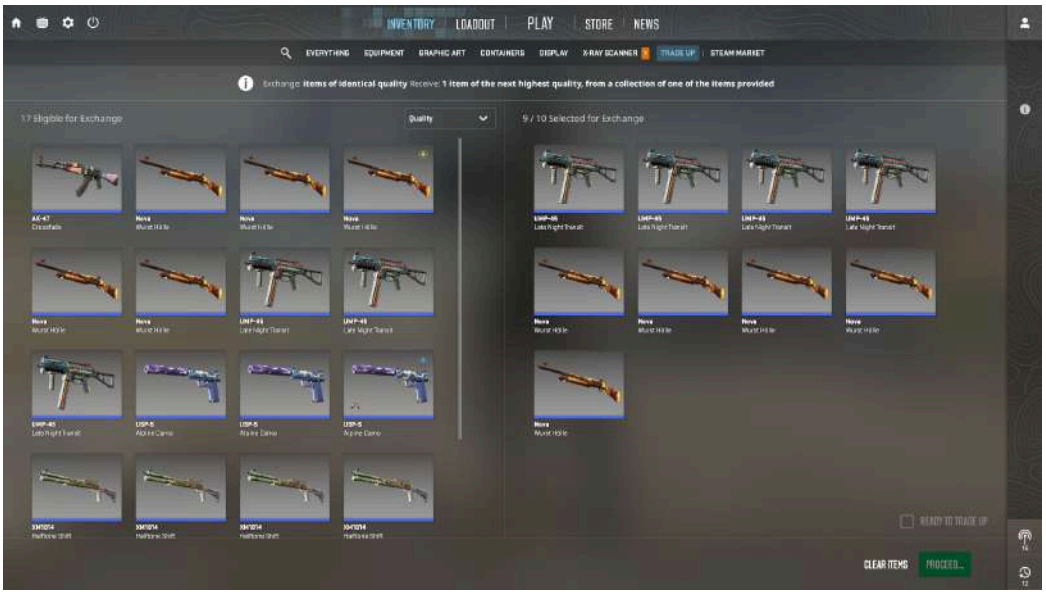

Рисунок 2.11 – Вікно контракту обмінів

Арсенал (Armory) — внутрішньоігровий магазин, де предмети купуються за зірки — окрему валюту, що накопичується під час активної гри після придбання перепустки магазину (Armory Pass) за реальні гроші (див. рис. 2.12).

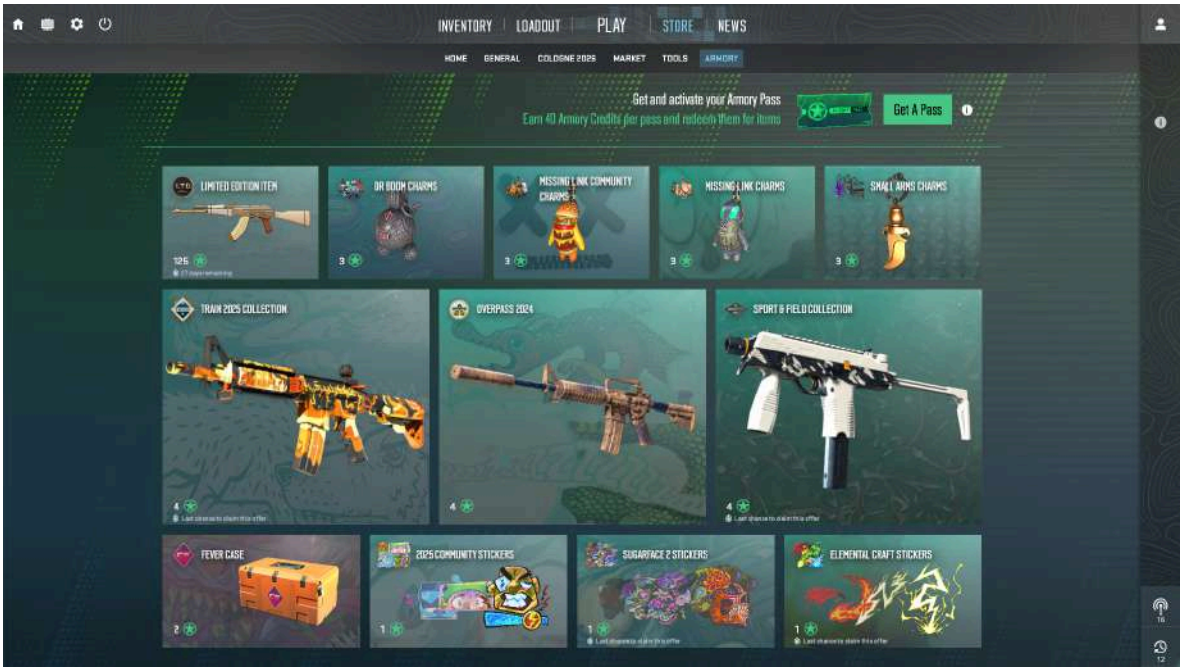

Рисунок 2.12 – Інтерфейс магазину “Armory”

Термінал (Terminal) — інструмент, що пропонує гравцеві 5 варіантів випадкових предметів для прямої купівлі у розробника (див. рис. 2.13).

Рисунок 2.13 – Вікно купівлі предмета в терміналі

Магазин турніру (Major Shop) — найновіша система. Раніше гравці мали можливість придбати в меню гри капсули з наліпками чи нашивками команд, що присвячені поточному професійному турніру. З 22 травня 2026 року в грі було оновлено пропозицію в магазині турніру. Відтепер він дозволяє купувати наліпки команд чи автографи професійних гравців напряму без механіки випадковості та азартної складової (див. рис. 2.14).

Рисунок 2.14 – Вікно магазину сезонного турніру (Major Shop)

2.4.2 Обґрунтування обраного джерела для моделювання

Для реалізації в моделі було обрано саме систему щотижневої винагороди (Weekly Care Package), адже вона є детермінованою з чітко визначеними правилами появи предметів, що базуються на ймовірностях випадіння. Додатково, модель контейнера вже повністю реалізує метод для його відкриття та генерації результату.

Системи арсеналу, терміналів і турнірного магазину не були обрані через непублічність та недостатню повноту даних щодо ціноутворення чи формули розрахунку валюти магазину Armory. Контракти обмінів потребували б окремої важкої реалізації з генерацією зносу предмета (float value) на основі вхідних предметів.

2.4.3 Реалізація класу DropGenerator

DropGenerator призначений для імітації системи щотижневої винагороди. Він використовує спеціальний атрибут класу Python — “`__slots__`”, що зменшує споживання пам’яті та прискорює доступ до полів класу. При ініціалізації класу переданий список агентів перетворюється на словник за ідентифікаторами агентів. Такий підхід забезпечує O(1)-доступ до екземпляра агента під час розподілу винагороди. А словник `items_drop_pool` одразу розпаковується у два паралельних списки, для оптимізації обчислень. Повний перелік атрибутів наведено у таблиці 2.3.

Таблиця 2.4 – Атрибути класу DropGenerator

Атрибут	Призначення
agents	Масив агентів, що
market	Посилання на екземпляр ринку
items_drop_pool	Список предметів і їх ймовірності
base_drop_chance	Частка агентів, що отримають винагороду щодня
reset_day	Індекс дня тижня скидання ліміту (0-пн, 1-вт, ..., 6-нд)
max_drops_per_week	Максимум предметів на один аккаунт
trade_lock_on	Застосування торгового блокування до виданих предметів
_eligible	Множина ID агентів, що мають право на дроп цього тижня
_items_list	Масив предметів доступних до отримання
_weights_list	Масив числових ваг (ймовірностей) для кожного предмета
total_drops_count	Лічильник загальної кількості виданих предметів за симуляцію

2.4.4 Алгоритм методу tick()

Єдиним публічним методом генератора предметів є метод *tick*(), що викликається на кожному кроці симуляції та імітує видачу щотижневої винагороди агентам, за такою послідовністю:

1. Перш за все метод перевіряє, чи відповідає поточний крок завершенню симуляційної доби. Якщо остача при діленні поточного кроку на кількість кроків, що дорівнює одній симуляційній добі, не дорівнює нулю, то умова не виконується та виконання методу завершується.

2. На початку кожної нової доби викликається метод `_is_reset_day()`. Якщо поточний день відповідає індексу дня оновлення обмеження щотижневої винагороди, то список придатних агентів оновлюється. З цього моменту кожен агент може отримати нову винагороду.

3. Наступним чином викликається метод `_calculate_winners_count()` для обчислення кількості переможців — агентів, що мають отримати винагороду.

4. Відбір переможців. Обрані унікальні ідентифікатори агентів вилучаються з масиву агентів-претендентів.

5. Розподіл винагороди в інвентарі агентів через `_distribute_items_to_winners()`.

Метод `_distribute_items_to_winners()` обчислює єдиний `unlock_step` (крок, коли всі ці предмети будуть розблоковані) для всіх переможців на цьому кроці та застосовує його до кожного виданого предмета. Кількість предметів для конкретного агента обчислюється за формулою (2.3):

$$N = n_{max} \cdot n_{acc}, \tag{2.3}$$

де n_{max} — максимальна кількість предметів на тиждень;

n_{acc} — кількість акаунтів агента, за замовчуванням дорівнює одному.

Такий спрощений підхід повністю відтворює систему щотижневих дропів в економіці Counter-Strike 2 та дає можливість для аналізу за допомогою симуляцій із різними параметрами. Детальніше в розділі експериментів.

2.5 Модуль агентів (agents.py)

На реальному ринку спільноти Steam одночасно взаємодіють учасники з принципово різними цілями, горизонтами планування та схильністю до ризику. Для відтворення гетерогенності учасників у моделі реалізовано чотири спеціалізованих типи агентів, кожен з яких успадковує абстрактний клас Agent. Гетерогенність досягається шляхом випадкової генерації агентів із різними типами, особистими параметрами тощо.

Базовий клас `Agent` визначає спільний стан для всіх типів агентів і реалізує їхні основні методи, що забезпечують керування інвентарем (додавання та видалення предметів), продаж, а також імітацію відкриття контейнерів. Зокрема, метод `is_impulsive()` реалізує стохастичний алгоритм прийняття рішення на основі генерації випадкової величини, розподіленої рівномірно в інтервалі [0, 1). Метод повертає логічне значення `True`, якщо згенероване число є меншим за індивідуальний показник імпульсивності агента. При цьому константа `IMPULSIVITY_UNDERESTIMATION` використовується для калібрування інтенсивності нераціональних дій агентів у симуляції. Повний перелік атрибутів наведено у таблиці 2.4.

Таблиця 2.5 – Атрибути класу Agent

Атрибут	Призначення
id	Унікальний ідентифікатор агента
market	Посилання на екземпляр ринку
balance	Баланс агента
type	Тип агента
impulsivity	Показник імпульсивності агента
inventory	Інвентар агента
containers_opened	Кількість відкритих контейнерів

`NoviceAgent` відтворює поведінку пересічного, ірраціонального користувача ринку, який не зацікавлений у отриманні прибутку та часто діє імпульсивно. Його роль у симуляції полягає в генерації ринкового шуму. Агент прагне отримати миттєвий результат, тому швидко позбавляється предметів, без спроби почекати й продати дорожче. Gode та Sunder [19] показали, що навіть агент із “нульовим інтелектом”, забезпечують реалістичну ринкову динаміку.

Логіка купівлі: агент отримує список доступних контейнерів, що торгуються на ринку, та обирає один випадковим чином. Далі він обирає кількість, що бажає придбати, та виконує купівлю. Після успішної купівлі він негайно відкриває ці контейнери для отримання їх вмісту.

Логіка продажу: агент обирає предмет і випадкову кількість розблокованих одиниць зі свого інвентарю. Наступним чином він знижує ціну, щоб точно продати свої предмети. Розмір знижки є випадковим числом від одного цента до одного долара.

`TraderAgent` здійснює короткострокові спекулятивні операції на основі аналізу тенденцій ринку. На відміну від `NoviceAgent`, він аналізує усі предмети з активними замовленнями на продаж і обирає найбільш потенційні для перепродажу. Особливістю цього агента є те, що він може взагалі не знайти такого предмета, пропустивши свою дію. Цей підхід робить виконання його дій більш ресурсномісткими за інших агентів. Агент має додатковий параметр толерантності до ризику — `risk_tolerance`, що впливає на прийняття ризикованої дії. Агент є раціональним, а тому під час розрахунку прибутковості предметів враховує комісію.

`InvestorAgent` імітує поведінку довгострокового інвестора, учасника, що купує активи, сподіваючись на майбутнє зростання ціни. Він заздалегідь виділяє фіксовану частку балансу під конкретну інвестицію. У випадку невиправдання інвестиції він швидко розпродає предмети. `InvestorAgent` має особистий параметр `risk_tolerance`, який впливає на частку капіталу,

витраченого для інвестування, ціну та кількість, а також на цільовий рівень прибутку. Враховує комісію при спробі реалізації своїх інвестицій.

`FarmerAgent` моделює вузький клас учасників ринку, що систематично отримують велику кількість предметів зловживаючи ігровими механіками за допомогою великої кількості ігрових акаунтів (див. рис. 2.15). Така поведінка негативно впливає на ціни предметів і погіршує ігровий процес для звичайних гравців. `FarmerAgent` має індивідуальний атрибут `number_of_accounts`, що визначає кількість ігрових акаунтів для одного агента. Особливістю цього агента, є те, що він намагається продати предмети партіями у межах медіани, щоб запобігти різкому обвалу ціни та шоку зі сторони інших учасників ринку. При спрацюванні імпульсивної дії, що моделює страх блокування акаунтів, `FarmerAgent` викликає метод панічного продажу предметів `_panic_sell()`. Він намагається швидко продати усі предмети з інвентаря за активними замовленнями на купівлю, щоб гарантувати швидкість продажу, без спроби максимізувати суму з продажу.

Рисунок 2.15 – Демонстрація зловживання користувачами ігровою механікою щотижневої винагороди [20]

РОЗДІЛ 3 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ ТА ЇХ АНАЛІЗ

Для кількісного дослідження впливу ключових параметрів ринку проведено серію незалежних експериментів методом Монте-Карло. Загальні параметри конфігурації, незмінні для всіх трьох серій, наведено у таблиці 3.1.

Таблиця 3.1 – Загальні параметри симуляцій

Параметр	Значення	Призначення
number_of_simulations	500	Кількість симуляцій
number_of_agents	1 000	Чисельність популяції (кількість агентів)
number_of_steps	90 000	Кількість симульованих кроків
steps_per_day	1 000	Кількість кроків в день
trade_lock_period	7	Період блокування предметів
lock_on_purchase	True	Застосування блокування для куплених предметів
max_wallet_balance	100 000	Обмеження гаманця агента
base_drop_chance	0,2	Частка агентів що отримують дроп
max_drops_per_week	1	Максимальна кількість предметів на тиждень для одного акаунта
trade_lock_on	True	Застосування блокування на щотижневу винагороду

Агенти генеруються однаковим чином за допомогою методу generate_agents() відповідно до визначених налаштувань генерації. Особисті параметри є випадковими, а. баланси згенеровані за нормальним розподілом. Налаштування генерації агентів, їхніх балансів і ваг наведено в таблиці 3.2.

Таблиця 3.2 – Загальні налаштування генерації агентів

Параметр	Значення	Призначення
number_of_agents	1 000	Чисельність популяції (кількість агентів)
min_balance	0	Мінімальний баланс агента
max_balance	2000	Максимальний баланс агента
mean_balance	650	Середнє значення баланса
std_dev_balance	300	Стандартне відхилення баланса
agent_weights	dict	Словник із вагами типів агентів
min_number_of_accounts	1	Мінімальна кількість акаунтів
max_number_of_accounts	1000	Максимальна кількість акаунтів
mean_number_of_accounts	100	Середнє значення кількості акаунтів
std_dev_account_numbers	50	Стандартне відхилення кількості акаунтів

Проведено три незалежних експерименти з різним складом агентів. У межах кожного експерименту було проведено п’ять тестів, що відрізняються умовами торгового блокування (Trade-lock). У кожному тесті значення ринкової комісії варіюється від 0,1 до 0,9 із кроком 0,1. Таким чином, загальна кількість умов у рамках одного експерименту становить 5 × 9 = 45, а загальна кількість симуляцій — 45 × 500 = 22 500 на один експеримент.

Умови торгового блокування для п’яти тестів у межах одного експерименту:

1. Тест 1 — Без блокування (`trade_lock_period = 0`): предмети одразу доступні до продажу; моделює ринок без інституційних обмежень на ліквідність;
2. Тест 2 — Без блокування на купівлю (`lock_on_purchase = False`), блокування протягом 7 діб зберігається лише для предметів виданих генератором, придбані на ринку предмети негайно доступні для перепродажу;
3. Тест 3 — Повне блокування, 7 діб (базові умови Steam Community Market): торгове блокування тривалістю 7 симульованих днів застосовується до всіх предметів — як придбаних на ринку (`lock_on_purchase = True`), так і отриманих через систему щотижневої винагороди. Ці умови є максимально наближеними до реальних правил платформи;
4. Тест 4 — Повне блокування протягом 15 діб: посилені обмеження порівняно з реальним Steam (`trade_lock_period = 15`);
5. Тест 5 — Повне блокування протягом 30 діб: гранично жорсткі умови утримання предметів (`trade_lock_period = 30`).

Склад популяції агентів для кожного експерименту задається словником `AGENT_WEIGHTS`, значення якого для трьох різних експериментів структури агентів наведено в таблиці 3.3.

Таблиця 3.3 – Конфігурації складу популяції агентів

Експеримент	Новачок	Трейдер	Інвестор	Фермер
№1	40 %	20 %	30 %	10 %
№2	10 %	70 %	10 %	10 %
№3	10 %	10 %	70 %	10 %

Метриками оцінювання для кожної умови є середнє значення та стандартне квадратичне відхилення (СКВ) трьох показників: загальної кількості угод (avg_sales), середньозваженої ціни (avg_price) та сукупного доходу платформи від комісії (avg_total_fee).

3.1 Експеримент 1. Результати та аналіз

Популяція агентів відповідає схожій збалансованій структурі учасників ринку спільноти Steam: `NoviceAgent` — 40%, `TraderAgent` — 20%, `InvestorAgent` — 30%, `FarmerAgent` — 10%. Проведено 5 тестів із різними налаштуваннями торгового блокування. А в межах кожного тесту — по 500 симуляцій для різних значень комісії. Зведені результати для базової конфігурації при повному торговому блокуванні 7 днів (Тест 3) наведено у таблиці 3.4.

Таблиця 3.4 – Результати Експерименту 1 (Тест 3)

market_fee	Сер. к-ть угод	СКВ угод	Сер. ціна	Сер. дохід від комісії
0,10	34 833	3 005	6,03	113 056
0,20	34 270	2 974	5,31	197 745
0,30	33 676	3 241	4,67	260 057
0,40	32 425	3 300	3,94	292 724
0,50	30 985	3 229	3,00	280 283
0,60	29 982	2 533	2,32	261 811
0,70	29 272	1 899	1,89	248 460
0,80	28 941	1 665	1,70	255 652
0,90	28 704	1 726	1,64	275 465

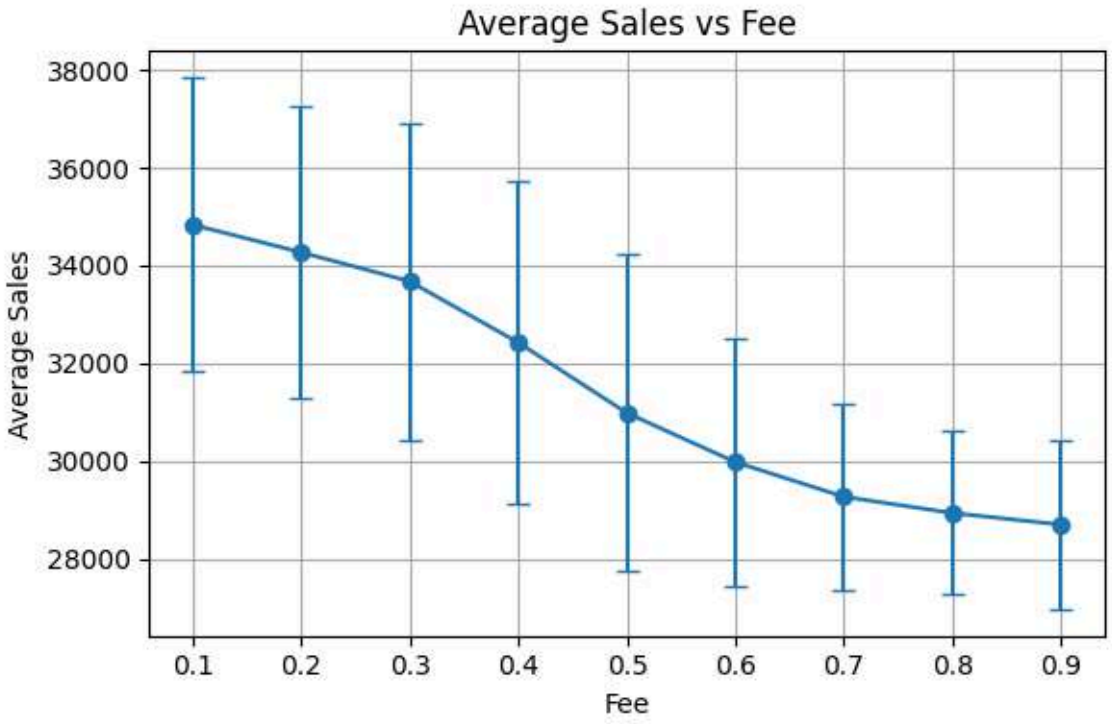

Рисунок 3.1 – Тест 3. Залежність середньої кількості продажів від значення комісії

Рисунок 3.2 – Тест 3. Залежність середньої ціни від значення комісії

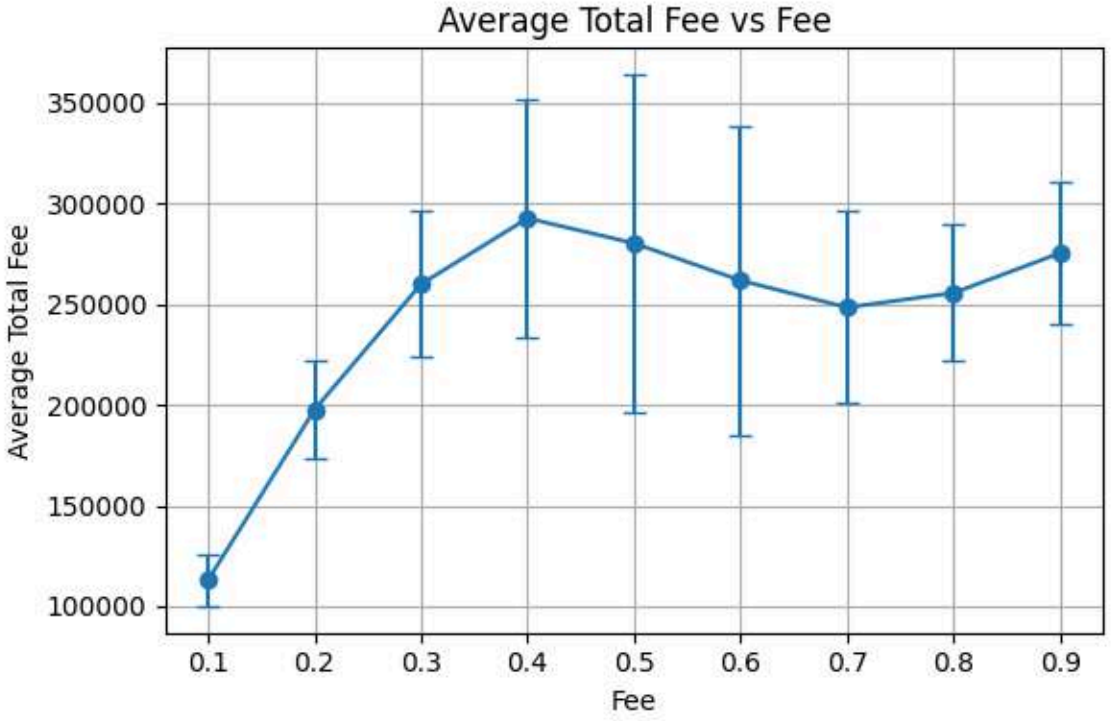

Рисунок 3.3 – Тест 3. Залежність сукупного доходу платформи від ставки комісії

Кількість угод монотонно зменшується зі збільшенням значення комісії: середнє значення кількості угод зменшилось на 17,6% з 34 833 при 10% комісії до 28 704 при 90% (див. рис. 3.1). В умовах замкненої системи, без симульованого додаткового грошового потоку балансу агентів в агентів з часом меншає коштів, щоб купляти предмети. А зі збільшенням значення комісії стягування коштів стає більш стрімким. Покупна спроможність агентів падає, це видно на зменшенні середньої ціни зі збільшенням комісії. Але навіть при найвищому показнику комісії на ринку зберігається якась активність у результаті дешевих цін на предмети, агенти можуть дозволити більшу кількість до купівлі. Це свідчить про нееластичну природу попиту, що підтверджується аналізом еластичності (див. рис. 3.4).

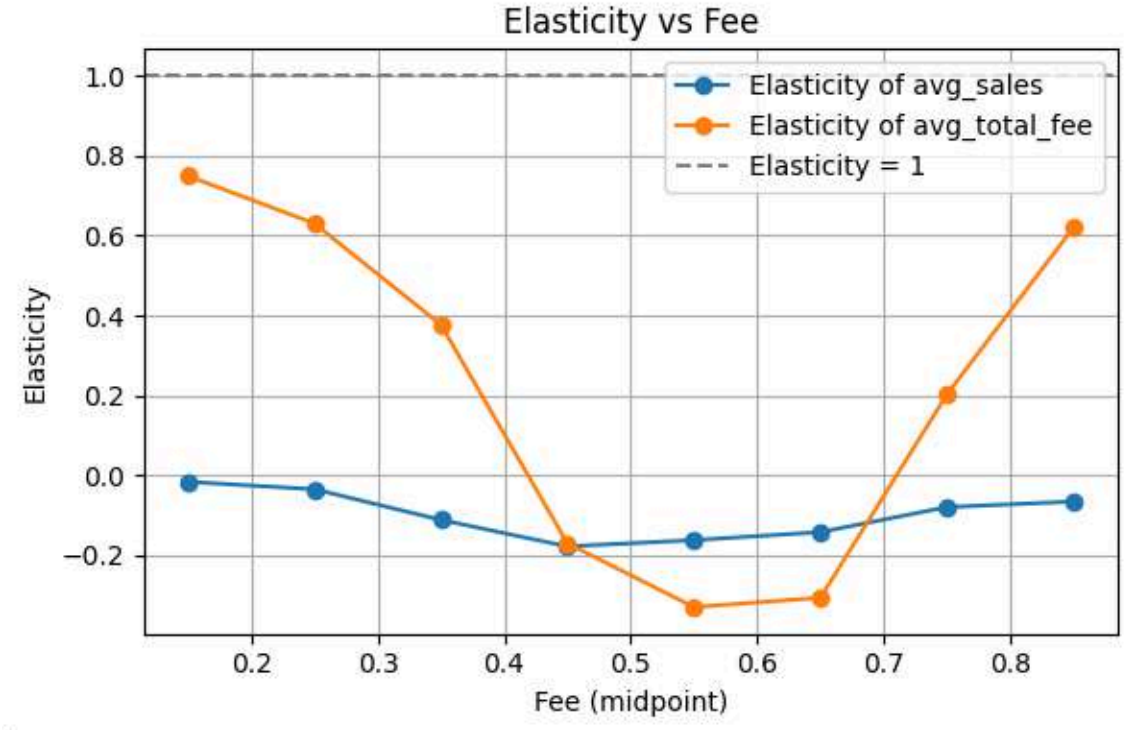

Рисунок 3.4 – Тест 3. Залежність сукупного доходу платформи від ставки комісії

Дохід платформи від комісії демонструє немонотонну залежність від самої ставки збору з вираженим максимумом при 40%: 292 724 гр. од. (див. рис. 3.3). При подальшому зростанні ставки дохід знижується до 248 460 гр. од. при комісії 70%, після чого зростає незначною мірою. Такий результат є проявом

ефекту, аналогічного кривій Лаффера: надмірна ставка збору настільки скорочує обсяг продажів і середню ціну, що сукупний дохід з комісії зменшується попри зростання збору для кожної угоди.

Визначення оптимальної ставки комісії досягнуто методом зваженої утиліти. Оскільки максимізація ліквідності (avg_sales) та максимізація доходу платформи (avg_total_fee) є конфліктуючими цілями, задача зводиться до двокритеріальної оптимазації, яка скаляризується до однокритеріальної через лінійну комбінацію нормованих критеріїв за формулою (3.1):

$$U(fee) = w \cdot f_1(fee) + (1 - w) \cdot f_2(fee) \quad (3.1)$$

де $f_1(fee)$ — нормована середня кількість продажів;

$f_2(fee)$ — нормований дохід від комісії;

w — ваговий коефіцієнт пріоритету.

Нормування за формулою (3.2) приводить обидва критерії до єдиного виміру [0;1] незалежно від їхніх абсолютних величин.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (3.2)$$

де x — поточне значення показника (avg_sales або avg_total_fee);

x_{min} — мінімальне значення показника по всьому діапазону досліджуваних значень комісії;

x_{max} — максимальне значення показника по всьому діапазону досліджуваних значень комісії;

x_{norm} — нормоване значення показника у діапазоні [0;1].

Аналіз проводиться для трьох значень w: 0,2 — пріоритет доходу платформи; 0,5 — баланс пріоритетів; 0,8 — пріоритет більшої кількості продажів.

Результати свідчать, що оптимальна ставка знаходиться у діапазоні 30-40% (див. рис. 3.5).

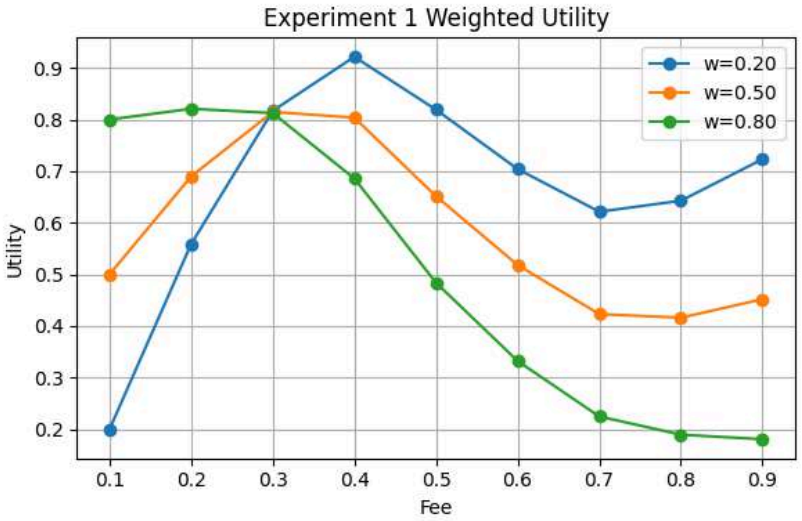

Рисунок 3.5 – Тест 3. Зважені утиліти

З рисунків 3.6 та 3.7 видно, що торгове блокування (Trade-lock) має якісно відмінний вплив на два показники: збільшення тривалості блокування одночасно підвищує середню ціну та знижує кількість угод. Довший період блокування утримує вищі ціни на предмети. Для найдовшого періоду блокування (30 днів) зі збільшенням комісії монотонно зменшується середня ціна, а при відсутності втримання предметів — вона зменшується стрімко, зі збільшенням значення комісії, а потім плавніше.

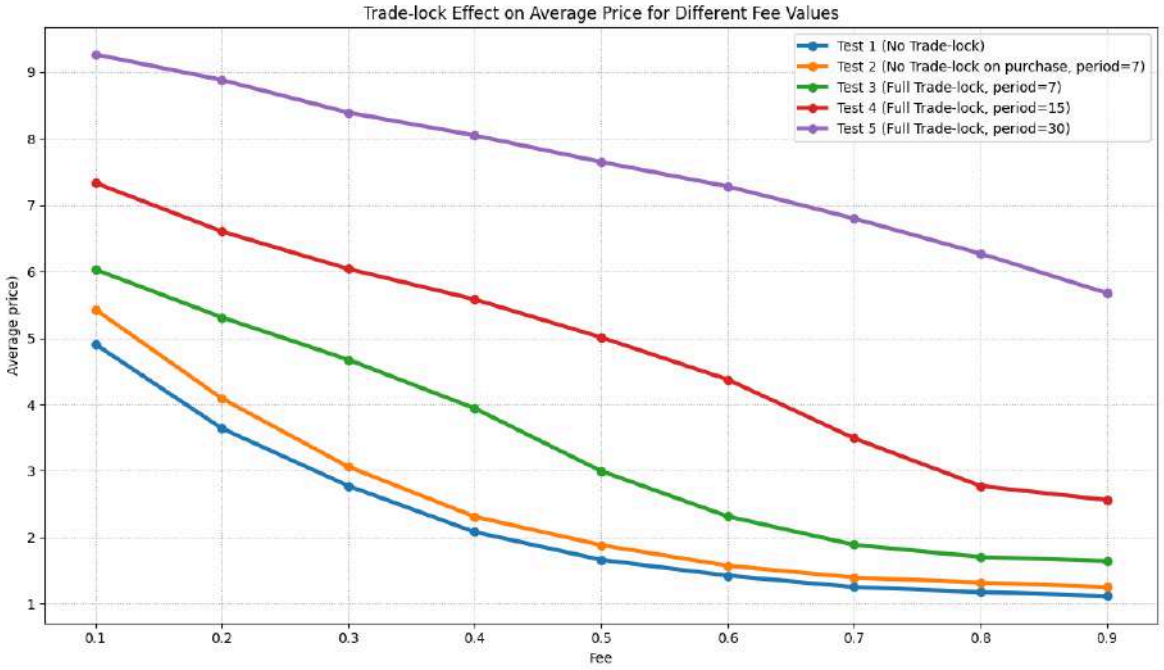

Рисунок 3.6 – Експеримент №1. Вплив Trade-lock на середню ціну для різних значень комісії

Повністю відсутнє блокування забезпечує найбільшу кількість продажів для всього діапазону значень комісії, тоді як 30-денне блокування — найменшу. Показово, що при 30-денному періоді блокування залежність кількості угод від значення комісії практично відсутня — крива є горизонтальною. При надзвичайно жорстоких обмеженнях, ефект змінив комісії на ліквідність стає другорядним порівняно зі зменшеною пропозицією: більшість предметів ще недоступна для продажу, а ціна не встигла впасти до таких рівнів, що дозволяли агентам купувати велику кількість предметів.

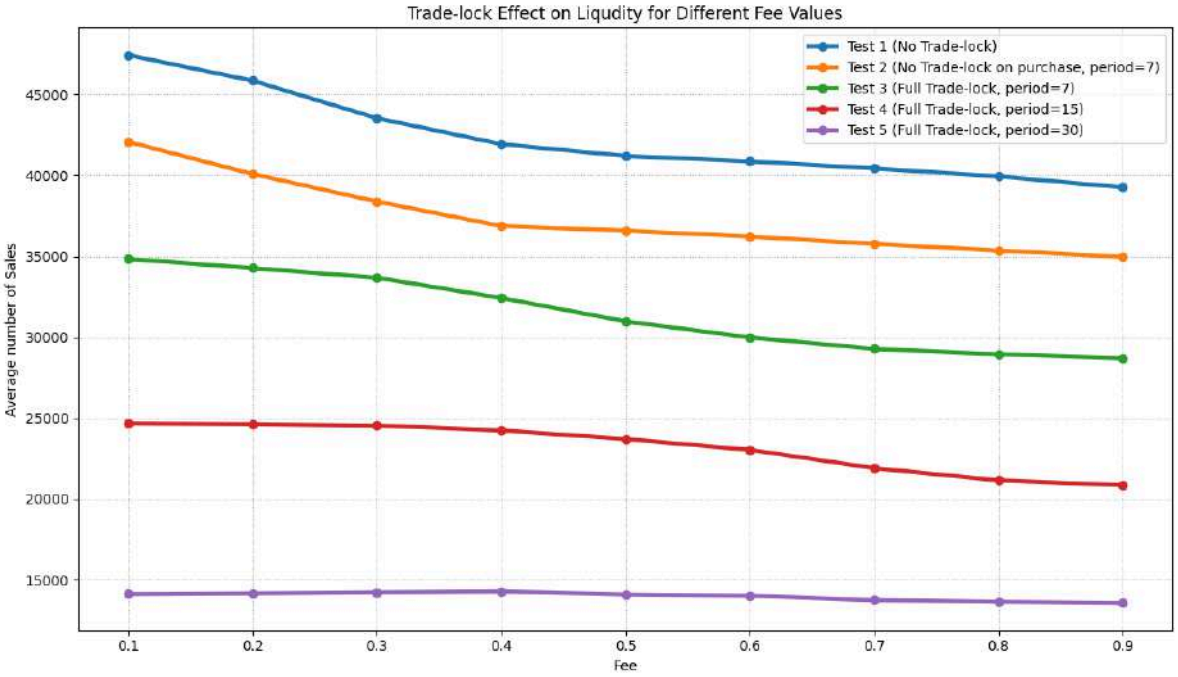

Рисунок 3.7 – Експеримент №1. Вплив Trade-lock на ліквідність для різних значень комісії

За результатами Експерименту 1 з налаштуваннями агентів основаними на умовах ринку спільноти Steam отримано такі висновки:

- кількість угод та середня ціна монотонно зменшуються зі збільшенням рівня комісії демонструючи сповільнення темпу та нелінійну залежність;
- дохід платформи не прямопропорційний від ставки комісії, але від серії факторів (кількості угод і середньої ціни), а тому досягає свого максимуму при комісії рівній 40% описуючи нелінійну залежність аналогічно кривій Лаффера;
- за допомогою зваженої утиліти визначено діапазон оптимального значення комісії від 30% до 40%, незалежно від пріоритету платформи;
- торгове блокування помірно зменшує ліквідність та суттєво втримує значення середньої ціни.

3.2 Експеримент 2

Ті ж самі налаштування симуляцій, окрім складу агентів: `NoviceAgent` — 10%, `TraderAgent` — 70%, `InvestorAgent` — 10%, `FarmerAgent` — 10%. Тепер на ринку перебільшують агенти-трейдери. Зведені результати для заданої конфігурації та повному торговому блокуванні 7 днів (Тест 3) наведено у таблиці 3.5.

Таблиця 3.5 – Результати Експерименту 2 (Тест 3)

market_fee	Сер. к-ть угод	СКВ угод	Сер. ціна	СКВ ціни	Сер. дохід від комісії	СКВ доходу
0,10	37 097	4 184	4,80	0,872	116 314	19 973
0,20	35 156	4 878	3,97	0,890	185 529	41 854
0,30	33 840	5 135	3,37	0,819	226 964	57 088
0,40	33 840	4 763	2,83	0,724	247 397	65 464
0,50	30 957	4 230	2,46	0,579	261 174	61 957
0,60	29 773	3 573	2,17	0,506	268 632	62 259
0,70	28 493	2 727	1,91	0,368	265 570	51 571
0,80	27 734	2 336	1,74	0,340	269 927	54 521
0,90	27 391	2 105	1,64	0,330	284 101	56 504

Порівнюючи отримані результати з попереднім експериментом, видно, що середня ціна по всьому діапазону значень комісії нижча за попередню структуру ринку. Зниження ціни пояснюється структурою агентів. Кількість продажів монотонно спадає зі зростанням комісії, при високих значеннях комісії, частина угод перестає бути прибутковою, через що заробіток платформи з комісії став меншим. Пояснюється це нижчими середніми цінами. Значення оптимальної комісії залишилося на тому самому рівні.

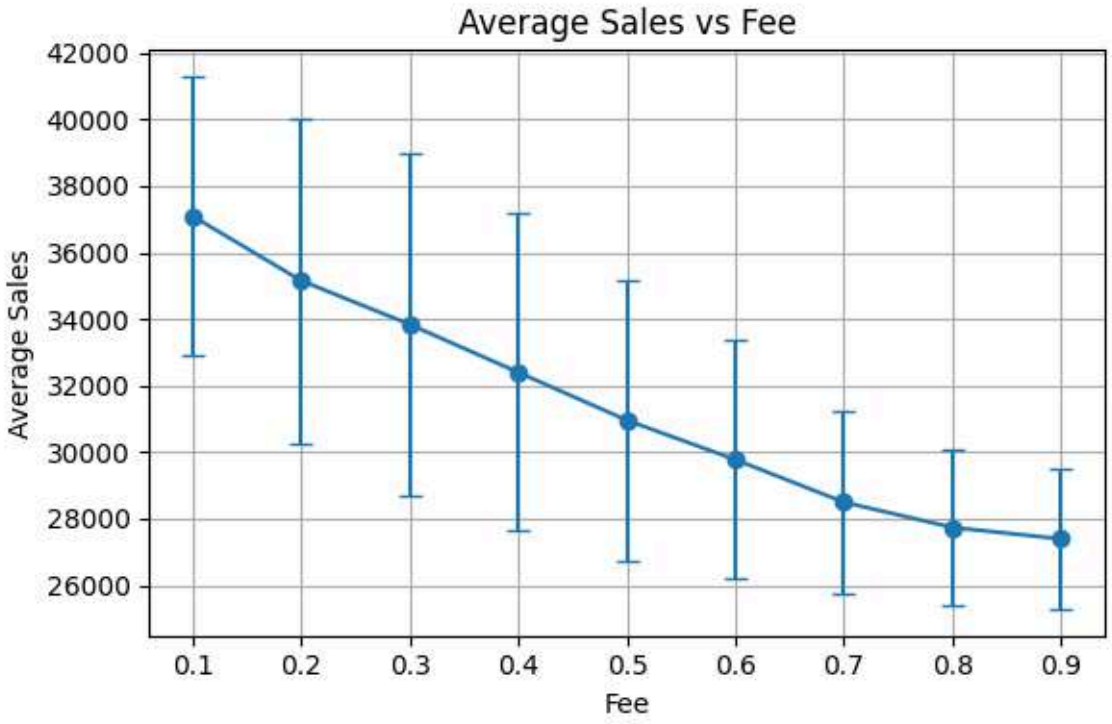

Рисунок 3.8 – Тест 3. Залежність середньої кількості продажів від значення комісії

Рисунок 3.9 – Тест 3. Залежність середньої ціни від значення комісії

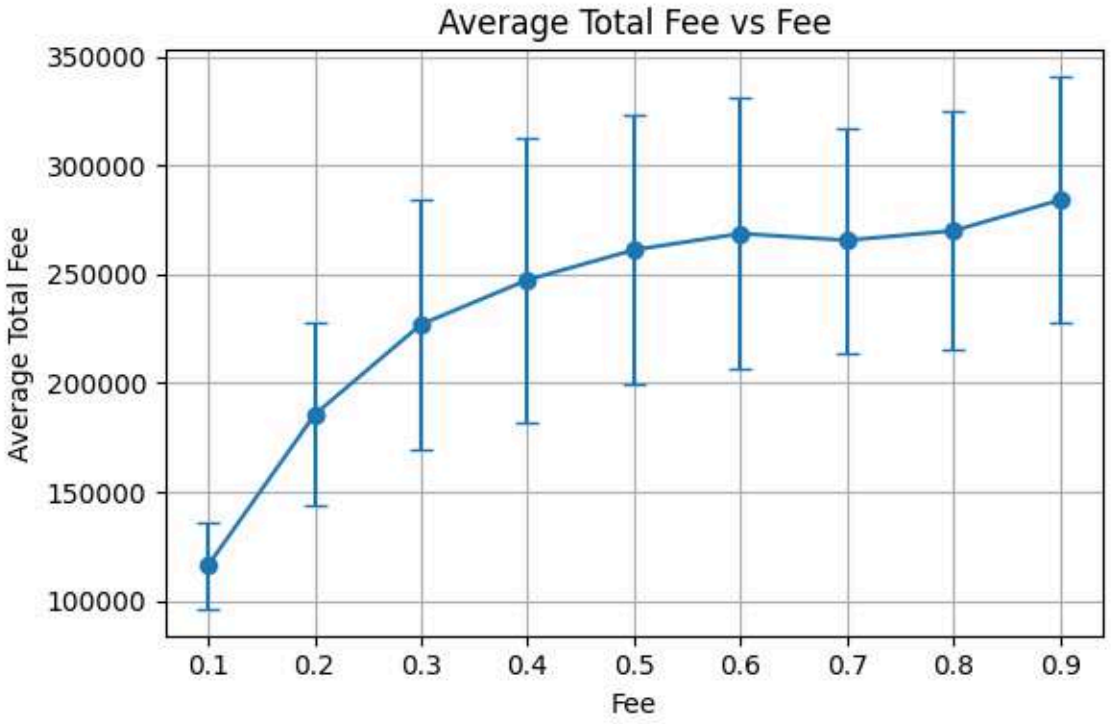

Рисунок 3.10 – Тест 3. Залежність сукупного доходу платформи від ставки комісії

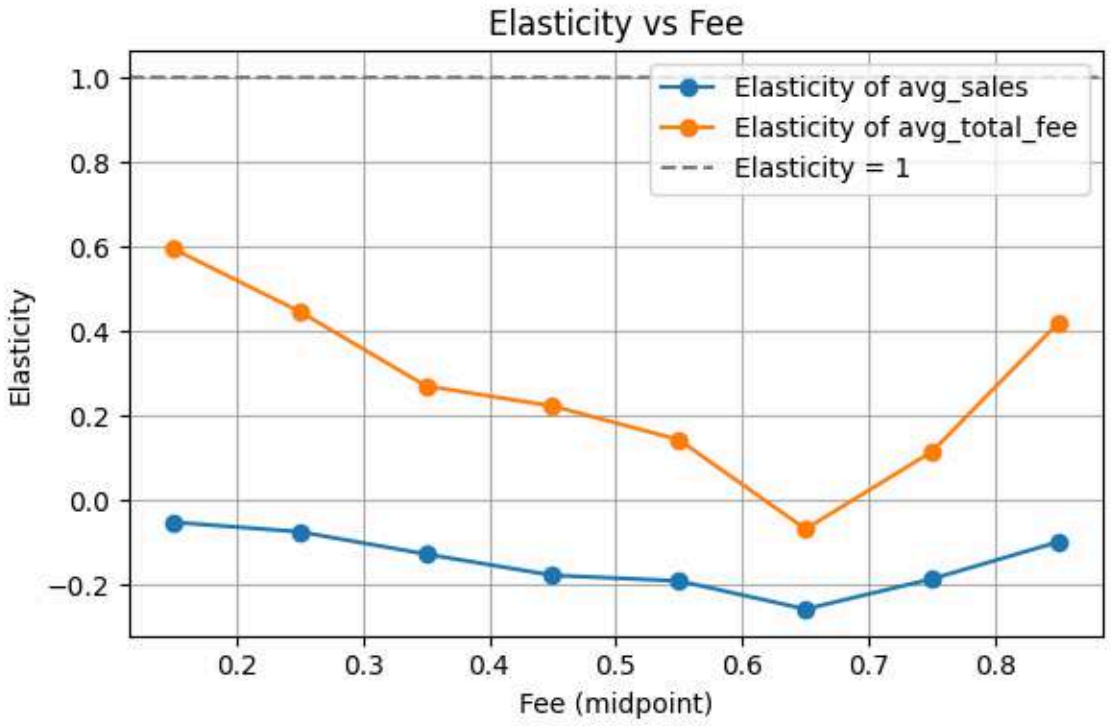

Рисунок 3.11 – Тест 3. Залежність сукупного доходу платформи від ставки комісії

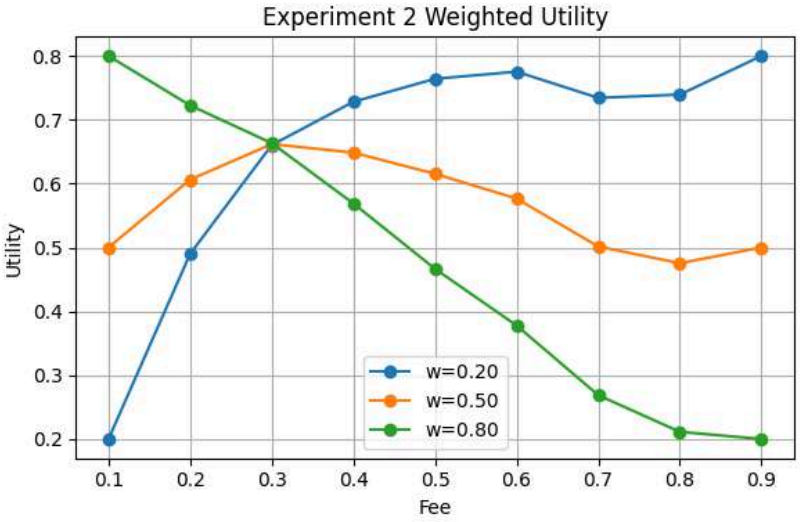

Рисунок 3.12 – Тест 3. Зважені утиліти

Тенденції впливу тимчасового торгового блокування (Trade-lock) на ціну та ліквідність повторюються аналогічно попередньому експерименту. Збільшення тривалості блокування втримує середню ціну через дефіцит пропозиції.

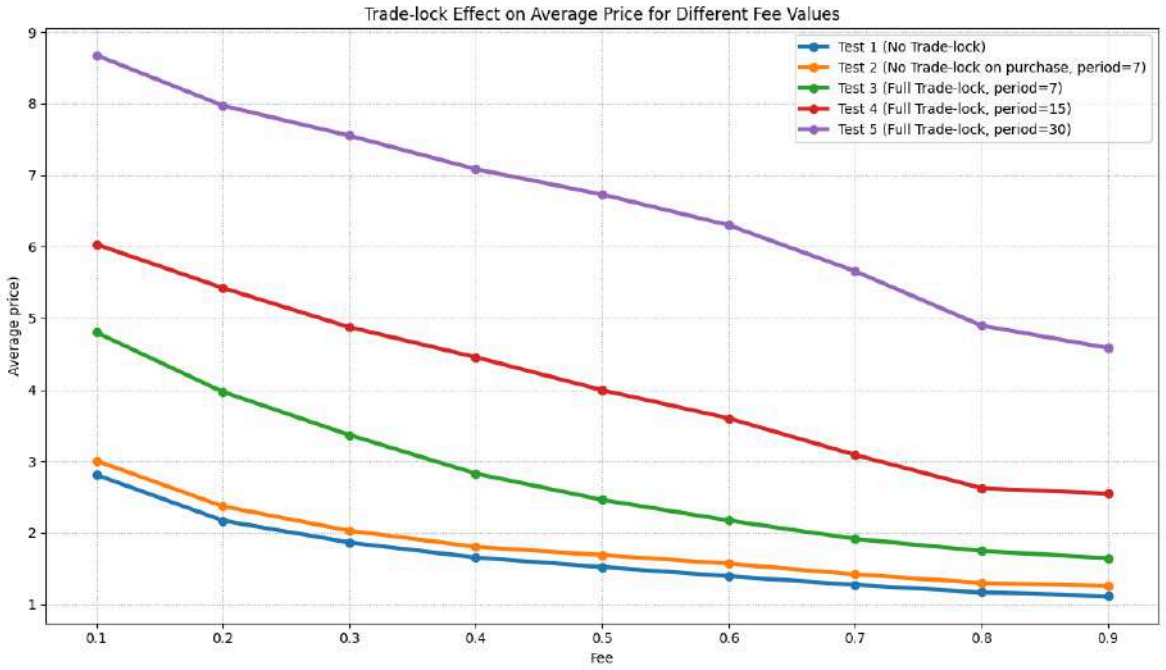

Рисунок 3.13 – Експеримент №2. Вплив Trade-lock на середню ціну для різних значень комісії

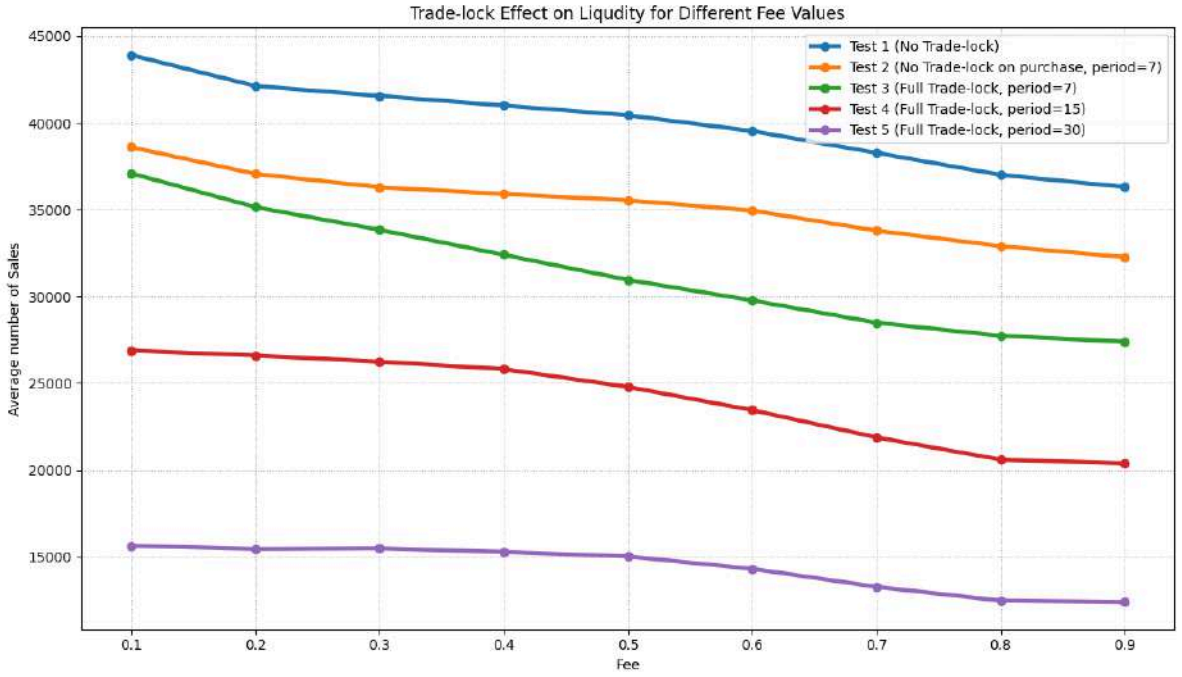

Рисунок 3.14 – Експеримент №2. Вплив Trade-lock на ліквідність для різних значень комісії

Переважання трейдерів підвищує ліквідність ринку при низьких значеннях комісії, але систематично знижує середню ціну предметів порівняно з попереднім експериментом. Саме обчислення симуляцій було довшим за попереднє, через складність типу агента. Трейдери аналізують ринок, перевіряють ціни на кожен предмет з метою знайти вигідну пропозицію, часто використовують автоматизовані системи, що впливає на роботоспроможність серверної інфраструктури.

3.3 Експеримент 3

Нові налаштування складу агентів: `NoviceAgent` — 10%, `TraderAgent` — 10%, `InvestorAgent` — 70%, `FarmerAgent` — 10%. В цій структурі ринку переважають інвестори. Зведені результати для заданої конфігурації та повному торговому блокуванні 7 днів (Тест 3) наведено у таблиці 3.6.

Таблиця 3.6 – Результати Експерименту 3 (Тест 3)

market_fee	Сер. к-ть угод	СКВ угод	Сер. ціна	СКВ ціни	Сер. дохід від комісії	СКВ доходу
0,10	36 633	3 121	5,28	0,719	102 053	5 931
0,20	36 232	3 104	4,74	0,679	180 969	13 878
0,30	35 311	2 957	4,35	0,541	246 507	9 587
0,40	34 288	2 680	4,01	0,488	299 998	9 871
0,50	34 018	2 490	3,74	0,478	344 497	18 291
0,60	34 091	2 446	3,53	0,423	384 951	10 665
0,70	34 440	2 356	3,35	0,439	420 612	23 732
0,80	34 896	2 388	3,20	0,381	455 957	10 786
0,90	34 924	2 513	3,05	0,392	486 326	25 205

Даний експеримент продемонстрував якісну відмінну поведінку порівняно з попередніми. Середня кількість продажів при комісії рівній 10% склала 36 633 угоди, де далі спадає до мінімуму 34 018 при 50% комісійних і зростає до 34 924 при 90% комісії. Інвестори зберігають попит на предмети на всьому діапазоні значень комісії. Стандартне квадратичне відхилення кількості продажів є рівномірним і зберігається біля 3 000. Саме значення комісії впливає на прибутковість їх інвестицій у майбутньому. Середня ціна монотонно зменшується при збільшенні комісії. Вона все ще є вищою за конфігурацію

трейдерів, але нижчою за конфігурацію експерименту №1. Інвестори підтримують вищі ціни предметів шляхом постійної закупівлі та утримування їх в інвентарі. Ринок з переважанням інвесторів є найстабільнішим за ціною.

Дохід платформи з комісії монотонно зростає, що пояснюється поєднанням високих цін і відносно стабільної кількості продажів при високих значеннях комісії, адже інвестори лише купують предмети та втримують потік пропозиції спричинений фермерами. Через що оптимальне значення комісії змістилося вправо та знаходиться в діапазоні від 30% до 40%.

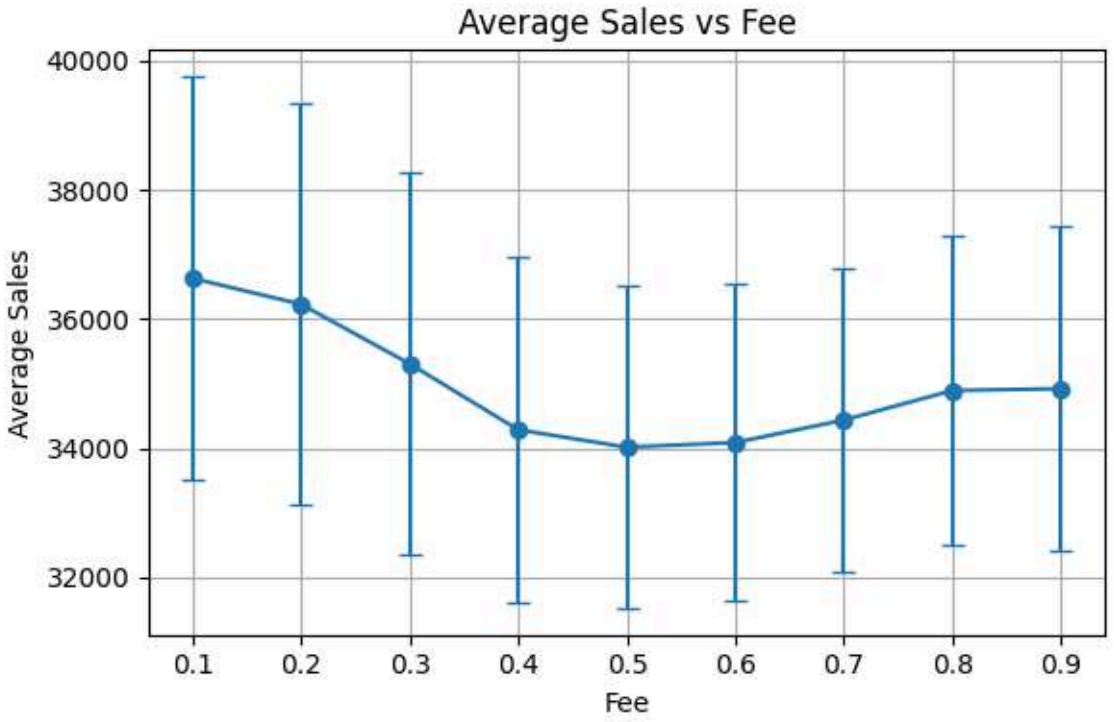

Рисунок 3.15 – Тест 3. Залежність середньої кількості продажів від значення комісії

Рисунок 3.16 – Тест 3. Залежність середньої ціни від значення комісії

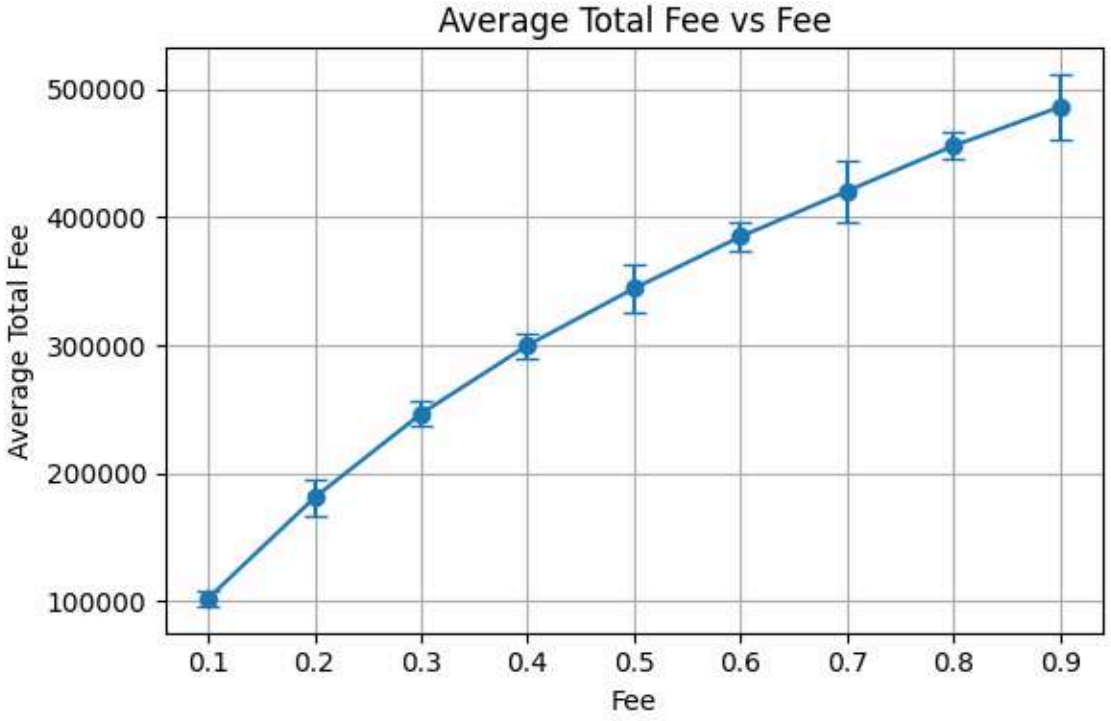

Рисунок 3.17 – Тест 3. Залежність сукупного доходу платформи від ставки комісії

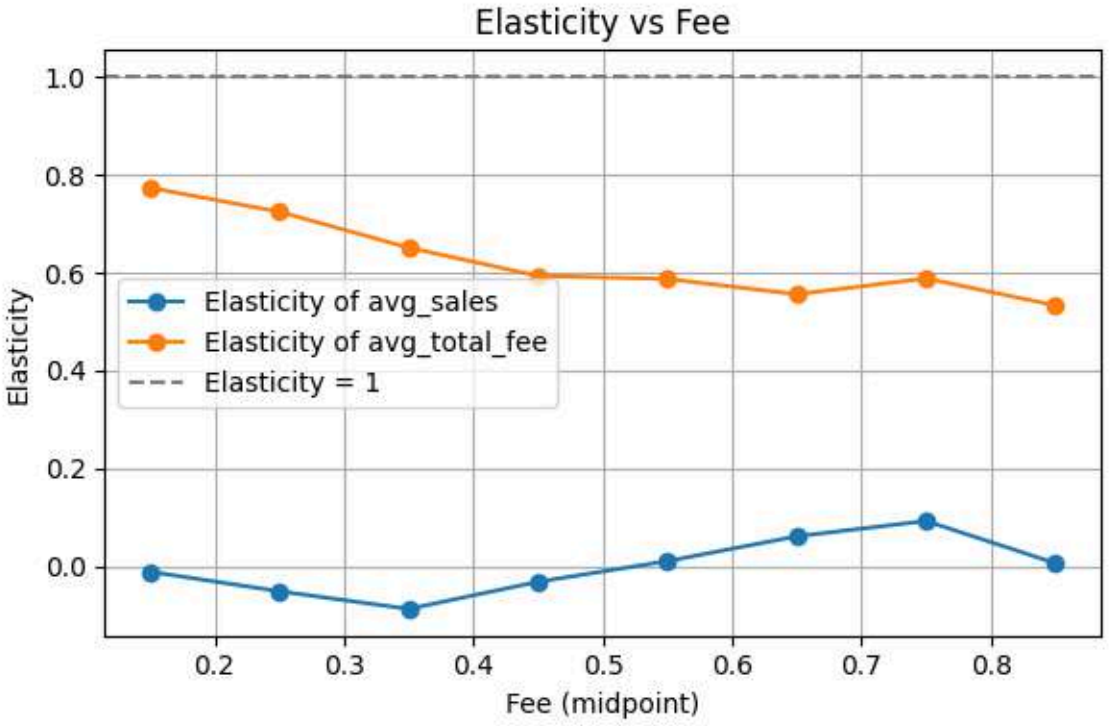

Рисунок 3.18 – Тест 3. Залежність сукупного доходу платформи від ставки комісії

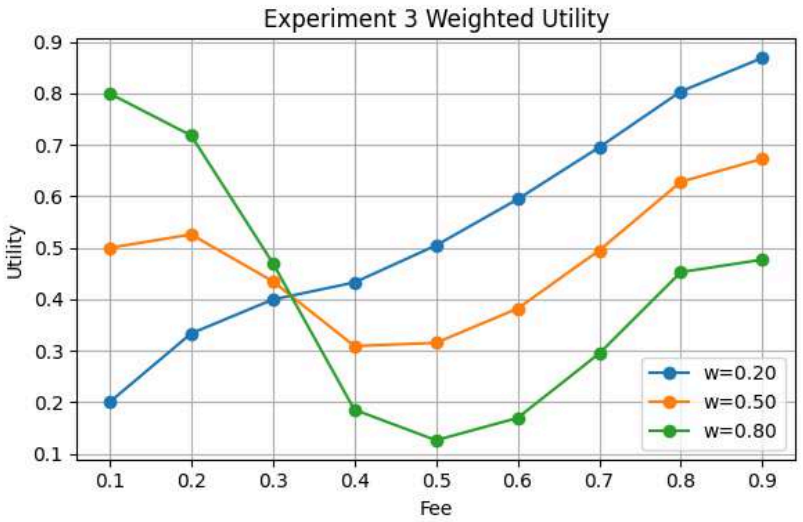

Рисунок 3.19 – Тест 3. Зважені утиліти

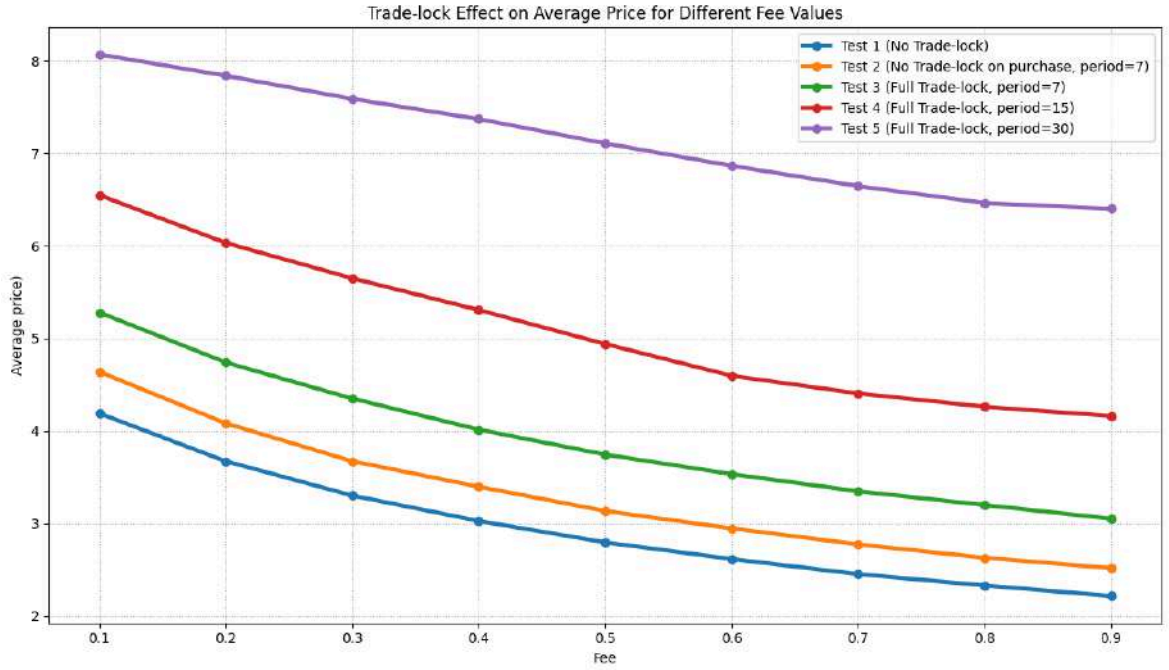

Рисунок 3.20 – Експеримент №3. Вплив Trade-lock на середню ціну для різних значень комісії

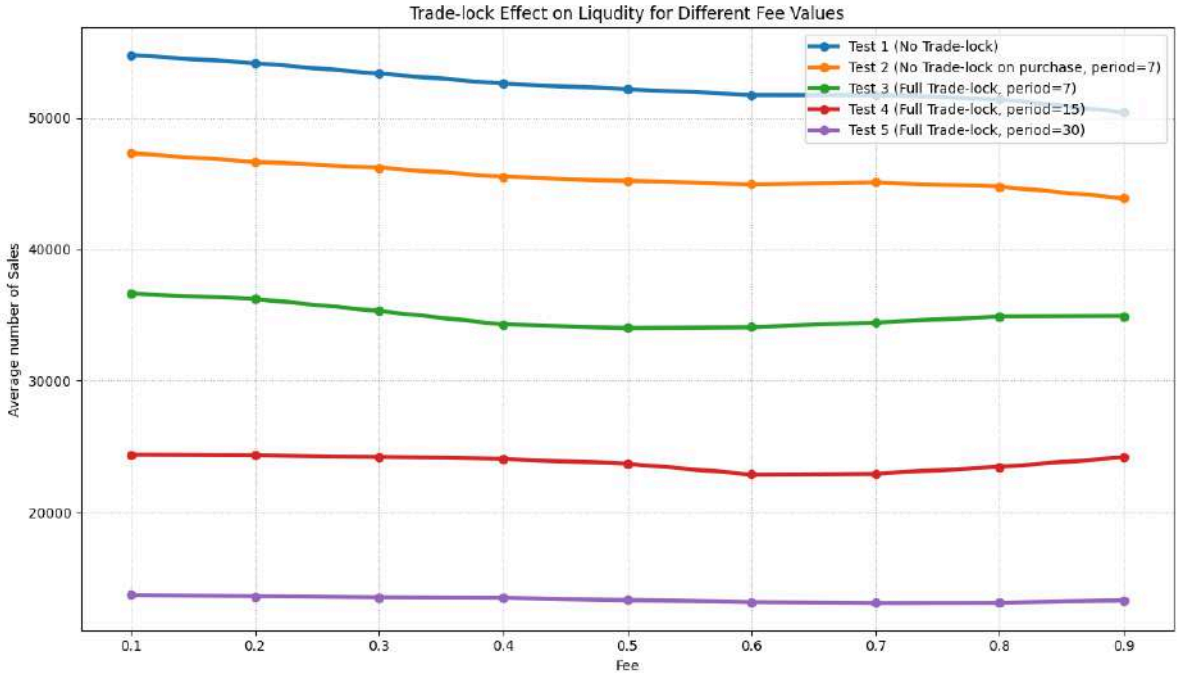

Рисунок 3.21 – Експеримент №3. Вплив Trade-lock на ліквідність для різних значень комісії

Даний експеримент продемонстрував якісну відмінну поведінку порівняно з попередніми. Середня кількість продажів при комісії рівній 10% склала 36 633 угоди, де далі спадає до мінімуму 34 018 при 50% комісійних і зростає до 34 924 при 90% комісії. Інвестори зберігають попит на предмети на всьому діапазоні значень комісії. Стандартне квадратичне відхилення кількості продажів є рівномірним і зберігається біля 3 000. Саме значення комісії впливає на прибутковість їх інвестицій у майбутньому. Середня ціна монотонно зменшується при збільшенні комісії. Вона все ще є вищою за конфігурацію трейдерів, але нижчою за конфігурацію експерименту №1. Інвестори підтримують вищі ціни предметів шляхом постійної закупівлі та утримування їх в інвентарі. Ринок з переважанням інвесторів є найстабільнішим за ціною.

Дохід платформи з комісії монотонно зростає, що пояснюється поєднанням високих цін і відносно стабільної кількості продажів при високих значеннях комісії, адже інвестори лише купують предмети та втримують потік пропозиції, спричинений фермерами. Через що оптимальне значення комісії змістилося вправо та знаходиться в діапазоні від 30% до 40%. А тимчасове блокування мало такий самий ефект, як і в попередніх експериментах.

3.4 Вплив ймовірності винагороди на динаміку ринку

У роботі Csaszar [21] підтверджується, що структурні зміни ринку, структура агентів, характеристики предметів і умови їх появи мають значний вплив на їх ціну, тому було цікаво дослідити вплив ймовірності отримання винагороди в створеній моделі. Для дослідження впливу параметра `base_drop_chance` (ймовірності отримання щотижневої винагороди) використав структуру агентів для Експеримента 1 і зафіксовану ставку комісії рівну 15%. Результати 500 симуляцій для кожного з десяти значень `base_drop_chance` наведено у таблиці 3.7.

Таблиця 3.7 – Вплив ймовірності отримання винагороди.

base_drop_chance	Сер. к-ть угод	Сер. ціна	Сер. дохід від комісії
0,1	81 849	4,49	288 648
0,2	111 091	2,59	250 618
0,3	125 679	2,14	242 328
0,4	131 911	1,96	238 524
0,5	135 582	1,89	237 405
0,6	136 150	1,83	234 274
0,7	137 107	1,82	234 208
0,8	136 398	1,77	229 985
0,9	136 621	1,75	227 785
1,0	136 415	1,77	232 054

Залежність кількості угод від `base_drop_chance` зображено на рисунку (див. рис. 3.22): у діапазоні від 0,1 до 0,4 спостерігається стрімке зростання — з 81 849 до 131 911 угод (+61,2 %), тоді як між 0,5 та 1,0 приріст становить лише 833 угоди (+0,6 %). Точка практичного насичення досягається в діапазоні значень 0,4–0,6:, при чому подальше збільшення притоку предметів через генератор предметів не призводить до зростання кількості продажів. Причина — основна кількість предметів отримується агентами у перші дні після оновлення щотижневого ліміту. Параметр `base_drop_chance` визначає частку агентів, що отримають винагороду кожного дня. Після отримання агент не може повторно отримати її до оновлення обмеження. При високих значеннях переважна більшість предметів потрапляє на ринок у перші дні після скидання обмеження, що призводить до стрімкого падіння ціни в результаті

одномоментного приросту пропозиції. При низьких значеннях дроп рівномірно розподіляється протягом тижня.

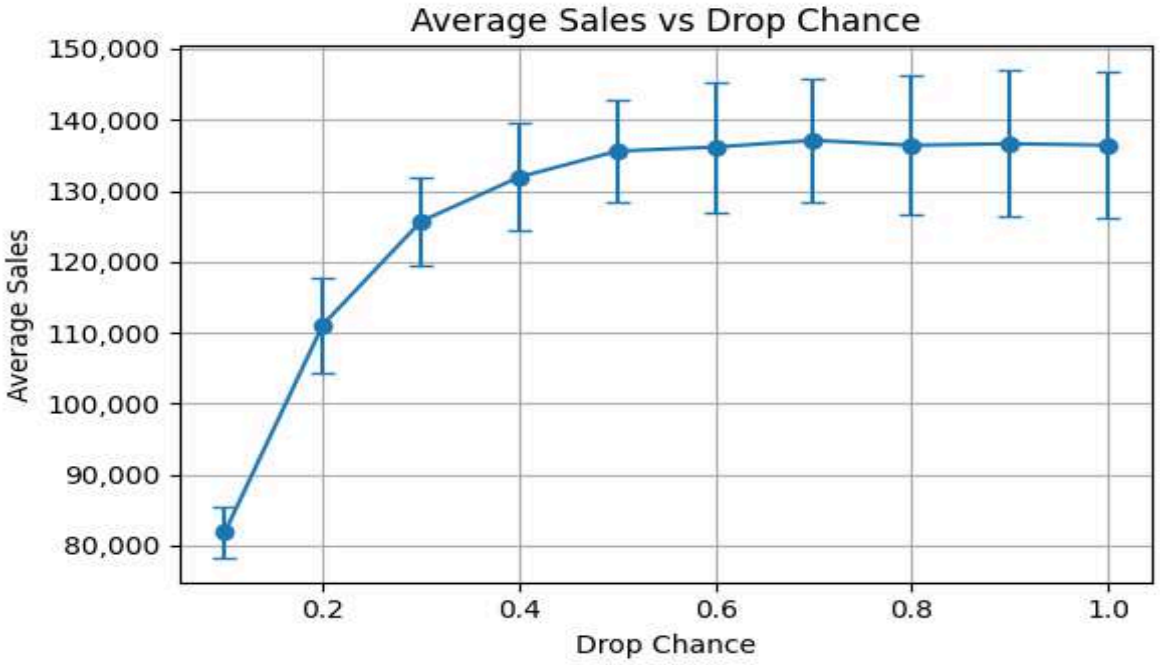

Рисунок 3.22 – Залежність середньої кількості продажів від ймовірності щотижневої винагороди

Середня ціна монотонно знижується з 4,49 до 1,77 гр. од. — більший притік предметів на ринок швидше тисне ціну вниз (див. рис. 3.23). При цьому зниження середньої ціни також сповільнюється зі зменшенням пропозиції на ринку після `base_drop_chance` рівного 0,5. Між значенням 0,5 та 1,0 ціна змінилась лише на -6,3%.

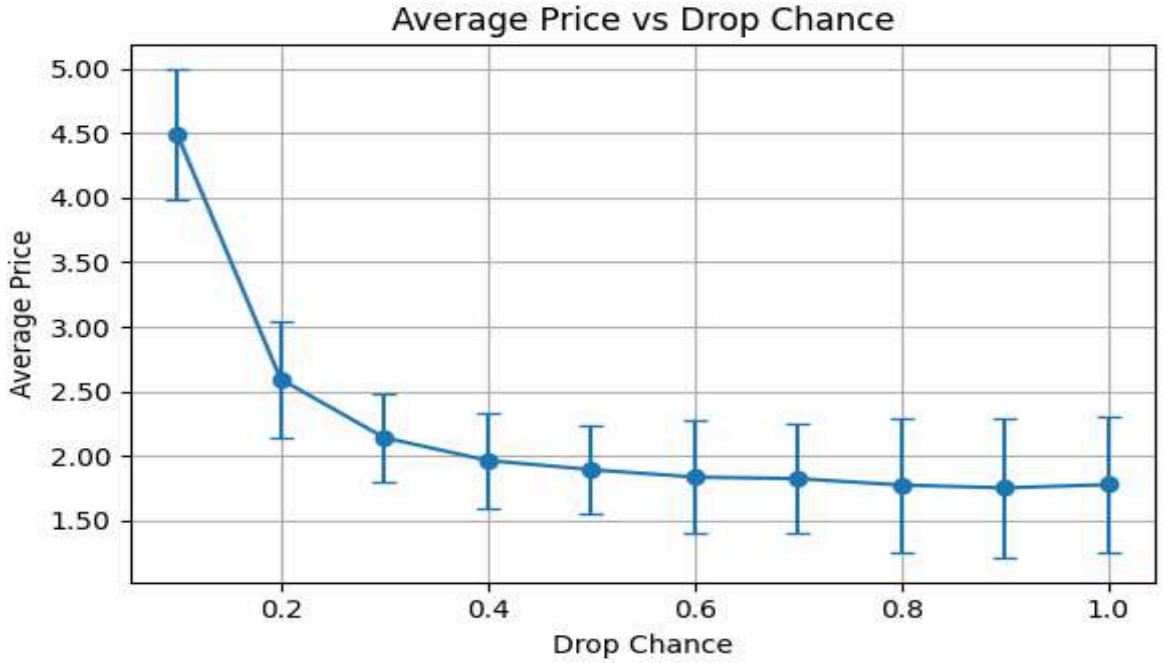

Рисунок 3.23 – Залежність середньої ціни предмета від ймовірності щотижневої винагороди

Дохід платформи від комісії демонструє спадну тенденцію: з 288 648 при base_drop_chance = 0,1 до 227 785 при 0,9 (−21,1 %), попри зростання кількості угод у 1,7 рази. Це пояснюється тим, що ціна падає швидше ніж зростає обсяг угод — добуток (кількість × ціна), тобто сукупний оборот ринку, скорочується. Іншими словами, більша кількість дешевших угод приносить платформі менший дохід від комісії, ніж менша кількість дорогих.

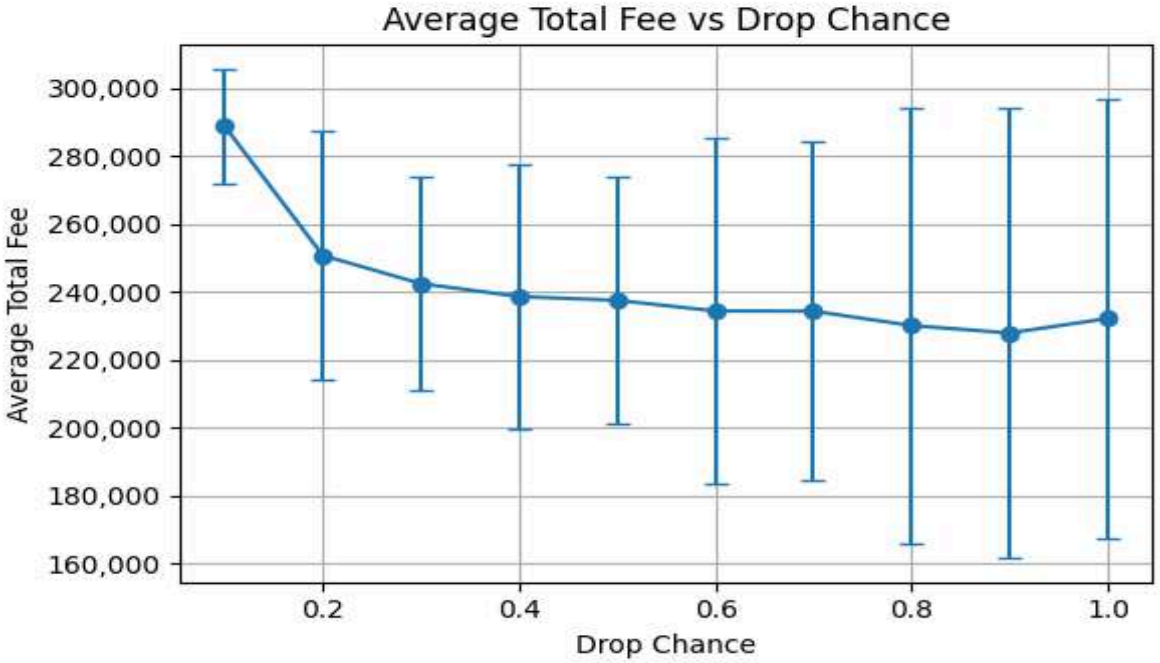

Рисунок 3.24 – Залежність комісійного прибутку платформи від ймовірності щотижневої винагороди

Таким чином, з позиції оператора платформи низька ймовірність дропу є фінансово вигіднішою: вона підтримує вищі ціни і вищий дохід від комісії, хоча і за рахунок меншої ліквідності ринку. Реальна система Steam встановлює base_drop_chance значно нижче насичення, що відповідає лівій зоні кривої — де кожен додатковий відсоток охоплення гравців відчутно збільшує активність ринку. Аналіз зваженої утиліти (див. рис. 3.25) демонструє оптимальне значення ймовірності отримання винагороди рівне 0,2. Переводячи з мови моделі, це означає, що слід балансувати складністю отримання винагороди гравцями для збалансованого потрапляння пропозиції на ринок.

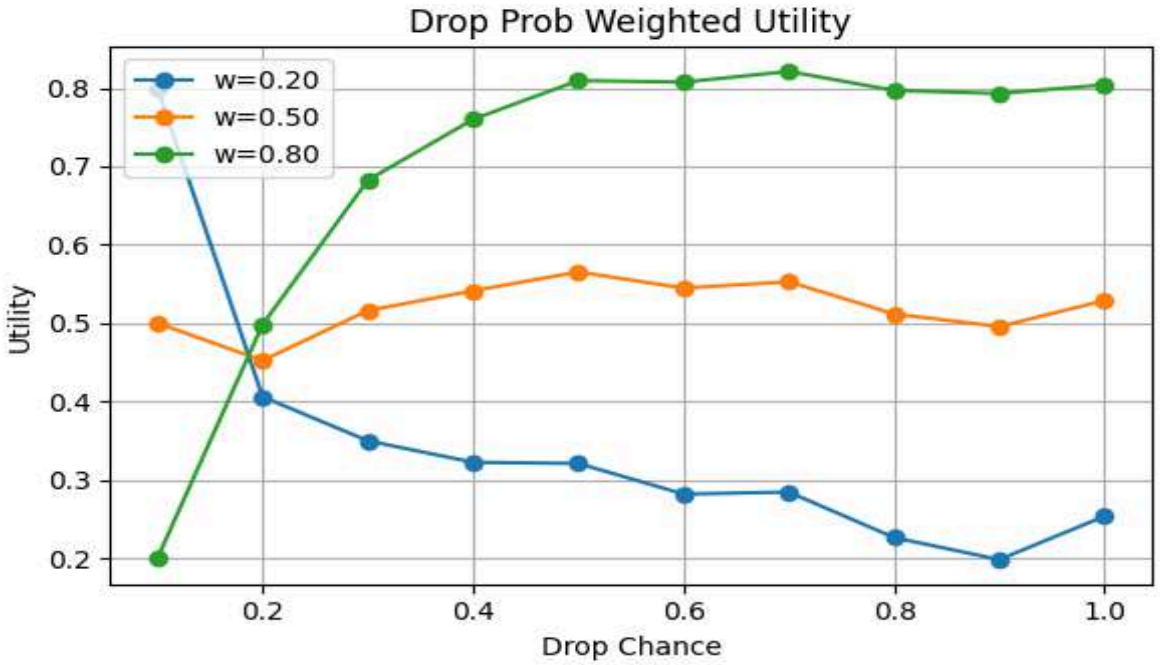

Рисунок 3.25 – Оптимальне значення ймовірності отримання щотижневої винагороди

ВИСНОВКИ

У дипломній роботі розроблено агентно-орієнтовану модель ринку цифрових активів Steam Community Market, а саме економіки предметів CS2. Модель відтворює ключові механізми платформи: збереження списку замовлень та узгодження за пріоритетом ціни та часу, систему торгового блокування (Trade-lock), щотижневу винагороду та обчислення комісії з кожного продажу. Реалізовано чотири основні типи агентів, кожен із власною стратегією поведінки. У ході розробки профілював модель та навчився основним методам оптимізації коду та використанню найбільш ефективних структур. Проведено серію експериментів з 500 симуляцій методом Монте-Карло. За результатами цих обчислень сформульовано наступні висновки.

1. Середня ціна предметів монотонно знижується зі збільшенням комісії в усіх конфігураціях ринку, що пояснюється вилученням коштів з учасників у замкненій системі: зменшується купівельна спроможність покупців, а продавці змушені знижувати ціну. Варто зауважити, що це додатково стимулює користувачів до поповнення гаманців акаунтів — прямого прибутку компанії.

2. Кількість продажів є нееластичною до збільшення комісії: зростання комісії з 10% до 90% спричинило скорочення кількості продажів лише на 17%, для збалансованої структури агентів. Зниження ціни дозволяє покупцям купувати більший обсяг предметів.

3. Залежність суми комісійних зборів платформи від рівня комісії є немонотонно зростаючою до певного значення, після якого стабілізується або знижується внаслідок падіння ціни та обсягу продажів.

4. Вплив структури агентів на ринкову динаміку є якісно значущим і важливим. При домінуванні трейдерів ринок характеризується вищою активністю і сильнішою реакцією ціни на зміну комісії. При домінуванні інвесторів спостерігається вищий і стійкіший рівень цін завдяки постійному підтримуючому попиту, кількість угод знижується повільно. Водночас ці два

типи агентів навантажують пропускну здатність серверів, через постійний аналіз ринку та використання автоматизованих систем. Користувачі типу фермер є дестабілізуючим фактором: масований продаж предметів чинить систематичний тиск на ціну. Отже, розробнику важливо впроваджувати заходи щодо балансування структури учасників і протидії дестабілізуючим факторам, спричиненим певними типами користувачів.

5. Система торгового блокування (Trade-lock), окрім забезпечення безпеки предметів від зловмисників, також виконує роль регулятора пропозиції: вона відтерміновує вихід предметів на ринок, тим самим підтримуючи ціну на вищому рівні протягом довшого періоду часу. Проте надмірне збільшення цього параметра має негативний ефект, гальмуючи циркуляцію предметів. Поточний семиденний період є компромісом між безпекою, стабільністю ринку і ліквідністю.

6. Окремі експерименти показали, що обмеження балансу агентів на рівні 2 000 гр. од. (відповідно до реального ліміту Steam) спричиняє дефіцит пропозиції: агенти, які досягли максимуму балансу, не можуть продовжувати продаж і змушені або купувати предмети або витрачати кошти.

7. На підставі результатів усіх трьох серій експериментів рекомендованим значенням комісії є діапазон від 10% до 30%. На початковому етапі розвитку платформи виправданим є нижнє значення цього діапазону для залучення учасників; після досягнення стабільного зростання — можна розглянути підвищення в межах помірного рівня з метою фінансування збільшених витрат на серверну інфраструктуру. Оптимальний період торгового блокування становить 4–7 діб. Отримані результати не враховують психологічного впливу рівня комісії.

8. Для заданої економічної моделі властиве виконання базових економічних принципів ціноутворення та «грошового розподілу Больцмана».

9. Отримані результати мають практичну цінність для формування комісійної політики цифрових торгових платформ. Розроблена модель є

гнучкою та масштабованою: вона може бути використана як інструмент симуляцій із реальними параметрами або як основа для подальших досліджень динаміки ринків цифрових активів.

10. Було успішно досягнуто поставленої задачі. У ході роботи я навчився планувати архітектуру агентно-орієнтованої моделі на основі реального ринку з продажу віртуальних активів, тестувати та оптимізовувати модель.

ПЕРЕЛІК ПОСИЛАНЬ

1. D'Anastasio C. Market for 'Counter-Strike 2' Digital Items Hits New High [Електронний ресурс]. Bloomberg. 2025. 7 March. URL: https://www.bloomberg.com/news/articles/2025-03-07/market-for-counter-strike-2-digital-items-hits-all-time-high (дата звернення: 16.04.2025).
2. Community Market FAQ [Електронний ресурс]. Valve Corporation. URL: https://help.steampowered.com/uk/faqs/view/61F0-72B7-9A18-C70B (дата звернення: 01.06.2026).
3. CSROI [Електронний ресурс]. URL: https://csroi.com (дата звернення: 20.07.2025).
4. Lehdonvirta V. Virtual Item Sales as a Revenue Model: Identifying Attributes that Drive Purchase Decisions. *Electronic Commerce Research.* 2009. Vol. 9, No. 1. P. 97–113. DOI: https://doi.org/10.1007/s10660-009-9028-2 (дата звернення: 01.05.2026).
5. Thorhauge A. M., Nielsen R. K. L. Epic, Steam, and the role of skin-betting in game (platform) economies. *Journal of Consumer Culture.* 2021. Vol. 21, No. 1. P. 52–67. DOI: https://doi.org/10.1177/1469540521993929 (дата звернення: 20.04.2025).
6. Yuan M. CS: GO Skins Market Impact Factors Analysis. Advances in Economics Management and Political Sciences. 2024. Vol. 82. P. 1–11. DOI: https://doi.org/10.54254/2754-1169/82/20230630 (дата звернення: 15.03.2025).

7. Hommes C. Heterogeneous Agent Models in Economics and Finance. SSRN Electronic Journal. 2005. DOI: https://doi.org/10.2139/ssrn.742384 (дата звернення: 01.05.2026).

8. Macal C. M., North M. J. Tutorial on agent-based modelling and simulation. Journal of Simulation. 2010. Vol. 4, No. 3. P. 151–162. DOI: https://doi.org/10.1057/jos.2010.3 (дата звернення: 19.04.2025).

9. Tesfatsion L., Judd K. L. (eds.). Handbook of Computational Economics. Vol. 2: Agent-Based Computational Economics. Eds. L. Tesfatsion, K. L. Judd. Amsterdam, North-Holland/Elsevier, 2006. 904 p.

10. Farmer J. D., Foley D. The economy needs agent-based modelling. *Nature*. 2009. Vol. 460, No. 7256. P. 685–686. DOI: https://doi.org/10.1038/460685a (дата звернення: 01.05.2026).

11. Todd A., Beling P., Scherer W., Yang S. Y. Agent-Based Financial Markets: A Review of the Methodology and Domain. *Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. Athens, Greece, 2016. P. 1–8. DOI: https://doi.org/10.1109/SSCI.2016.7850016 (дата звернення: 01.05.2026).

12. Guede-Fernández F., Wagle Y., Dias P. та ін. Artificial intelligence for algorithmic trading digital assets: evidence from the Counter-Strike 2 skin market. *Frontiers in Artificial Intelligence*. 2025. Vol. 8. DOI: https://doi.org/10.3389/frai.2025.1702924 (дата звернення: 01.05.2026).

13. Axtell R. L., Farmer J. D. Agent-Based Modeling in Economics and Finance: Past, Present, and Future. *Journal of Economic Literature*. 2025. Vol. 63, No.

1. P. 197–287. DOI: https://doi.org/10.1257/jel.20221319 (дата звернення: 01.05.2026).

14. Python Software Foundation. *Python Language Reference*. Version 3.12. [Електронний ресурс]. 2026. URL: https://docs.python.org/3.12/ (дата звернення: 20.07.2025).

15. Kazil J., Masad D., Crooks A. Utilizing Python for Agent-Based Modeling: The Mesa Framework. *Social, Cultural, and Behavioral Modeling: 13th International Conference SBP-BRiMS 2020, Washington, DC, USA, October 18–21, 2020, Proceedings. Lecture Notes in Computer Science.* 2020. Vol. 12268. P. 308–317. DOI: https://doi.org/10.1007/978-3-030-61255-9_30

16. Jenks G. *Python Sorted Containers: SortedList, SortedDict, SortedSet.* [Електронний ресурс] Version 2.4.0. 2021. URL: https://grantjenks.com/docs/sortedcontainers/ (дата звернення: 21.09.2025).

17. Cont R., Stoikov S., Talreja R. A Stochastic Model for Order Book Dynamics. *Operations Research*. 2010. Vol. 58, No. 3. P. 549–563. DOI: https://doi.org/10.1287/opre.1090.0780 (дата звернення: 01.05.2026).

18. How The Weekly CS2 Drop System Works. *CSGOSKINS.gg*. [Електронний ресурс]. URL: https://csgoskins.gg/blog/how-the-weekly-csgo-drop-system-works (дата звернення: 20.07.2025).

19. Gode D. K., Sunder S. Allocative Efficiency of Markets with Zero-Intelligence Traders: Market as a Partial Substitute for Individual Rationality. *Journal of*

Political Economy. 1993. Vol. 101, No. 1. P. 119–137. DOI: https://doi.org/10.1086/261868 (дата звернення: 01.05.2026).

20. Gabe Follower [@gabefollower]. [Ілюстрація технічної організації ферми облікових записів у Counter-Strike 2]. *Допис у мережі X/Twitter*. 5 травня 2024. URL: https://x.com/gabefollower/status/1787144419453374653 (дата звернення: 20.05.2026).

21. Csaszar B. *A Fixed Effects Panel Analysis of Skin Prices in Counter-Strike: Understanding Value in a Virtual Economy*. Thesis for: Economics and Business Administration. Aarhus University, 2025. DOI: https://doi.org/10.13140/RG.2.2.31567.39840 (дата звернення: 05.05.2026).

ДОДАТОК А
QR-код із посиланням на GitHub репозиторій

Рисунок 1. QR-код із посиланням на GitHub репозиторій