МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

УМАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет економіки, підприємництва та інформаційних технологій

Кафедра інформаційних технологій

Допущено до захисту

"_____" _____ 2026 р.

Зав. кафедри, к.т.н., доцент

_____ **Ліщук Р.І.**

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

РОЗРОБКА МОДУЛЮ ДЛЯ АДАПТИВНОГО НАЛАШТУВАННЯ РІВНЯ СКЛАДНОСТІ У ГРІ НА ОСНОВІ ПОВЕДІНКОВИХ ДАНИХ ГРАВЦЯ

Виконав – студент 4 курсу, групи 41-кн

освітньої програми «Комп'ютерні науки»

спеціальності 122 Комп'ютерні науки

Різник Богдан Сергійович

Науковий керівник – д.т.н., професор

Кучерук Володимир Юрійович

Умань – 2026

Зміст

Вступ 3
РОЗДІЛ 1 Основи адаптивного налаштування складності в іграх 5
1.1. Огляд сучасних засобів налаштування складності в іграх 5
1.2. Аналіз сучасних підходів до адаптивного налаштування складності 7
1.3. Постановка задачі 16
РОЗДІЛ 2 Моделювання процесу адаптивного налаштування складності. 18
2.1. Математична модель адаптивного налаштування складності 18
2.2. Архітектура модуля адаптивного налаштування складності та інтеграція з ігровими системами 24
2.3. Вибір мови програмування та середовища розробки. 28
РОЗДІЛ 3. РЕАЛІЗАЦІЯ МОДУЛЯ АДАПТИВНОГО НАЛАШТУВАННЯ СКЛАДНОСТІ 34
3.1. Реалізація збору даних. 34
3.2. Реалізація модуля адаптивного налаштування складності. 39
3.3. Тестування та аналіз результатів роботи модуля адаптивного налаштування складності 47
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 66
Додаток А Демонстраційні матеріали роботи модуля**Помилка! Закладку не визначено.**
Додаток В Лістинг програмного забезпечення .....**Помилка! Закладку не визначено.**

ВСТУП

Актуальність теми. Сучасні комп’ютерні ігри характеризуються зростанням складності ігрових систем, варіативністю поведінки користувачів та необхідністю підтримання збалансованого рівня виклику протягом ігрової сесії. Особливо актуальною ця проблема є для ігрових систем, у яких результат проходження залежить не лише від рівня підготовки гравця, а й від поточного стану, накопичених ресурсів, тривалості сесії та прийнятих ризикових рішень.

Традиційний підхід до налаштування складності за допомогою фіксованих рівнів, таких як «легкий», «середній» або «важкий», не завжди забезпечує належний баланс між доступністю та складністю. Користувачі з різним рівнем підготовки можуть отримувати нерівномірне навантаження: для менш досвідчених гравців складність може бути надмірною, тоді як для досвідчених – недостатньою для підтримання інтересу та мотивації. У результаті виникає потреба у більш гнучких механізмах, здатних враховувати поточний стан гравця та змінювати параметри складності в реальному часі.

Метою роботи є розроблення програмного модуля адаптивного налаштування складності, який забезпечує аналіз стану гравця в реальному часі та зміну параметрів складності з урахуванням ефективності, рівня ризику, ігрового тиску, тривалості сесії та ознак можливого зловживання системою.

Об’єктом дослідження є процес адаптивного керування складністю в інтерактивних ігрових системах.

Предметом дослідження є методи, моделі та алгоритми динамічного налаштування складності, що базуються на аналізі ефективності гравця, поведінкових характеристик, рівня ризику та поточного стану ігрової сесії.

Методи дослідження включають аналіз сучасних підходів до адаптивного налаштування складності, порівняння існуючих моделей регулювання ігрового навантаження, математичне моделювання показників стану гравця, алгоритмічне проєктування програмного модуля та експериментальну перевірку його роботи в умовах реального часу.

Практичне застосування такого модуля спрямоване на підвищення стабільності рівня складності, забезпечення плавної зміни ігрового навантаження та підтримання збалансованого ігрового процесу для користувачів із різним рівнем підготовки й різними поведінковими особливостями.

Запропонований підхід може бути застосований для підвищення збалансованості ігрового процесу, зменшення різких змін складності та виявлення ситуацій, пов’язаних із навмисним впливом на систему адаптації.

РОЗДІЛ 1 ОСНОВИ АДАПТИВНОГО НАЛАШТУВАННЯ СКЛАДНОСТІ В ІГРАХ

1.1. Огляд сучасних засобів налаштування складності в іграх

Налаштування складності є одним із важливих засобів формування збалансованого ігрового досвіду. Воно визначає рівень навантаження на гравця, інтенсивність ігрових подій, частоту помилок, доступність ресурсів, поведінку противників та загальний темп проходження. Основним завданням системи складності є підтримання такого рівня виклику, за якого гра залишається достатньо доступною для менш досвідчених користувачів і водночас не втрачає інтересу для досвідчених гравців.

У сучасних комп’ютерних іграх використовуються різні засоби налаштування складності. Найпростішим із них є фіксований вибір рівня складності перед початком гри. У цьому випадку користувач самостійно обирає один із запропонованих режимів, наприклад легкий, середній або складний. Такий підхід є простим у реалізації та зрозумілим для гравця, однак він не враховує зміну навичок користувача під час проходження, різницю між окремими ігровими ситуаціями та поточний стан гравця.

Іншим поширеним засобом є ручне коригування складності в налаштуваннях гри. Воно дозволяє змінювати рівень складності під час проходження, якщо поточний режим виявляється занадто легким або надмірно складним. Перевагою такого підходу є контроль з боку користувача, проте його недоліком є відсутність автоматичної реакції системи на зміну поведінки гравця. Крім того, необхідність самостійно змінювати складність може порушувати безперервність ігрового процесу.

Окрему групу становлять засоби балансування, побудовані на заздалегідь визначених правилах. У таких системах складність змінюється відповідно до конкретних умов: кількості поразок, рівня здоров’я, часу проходження, кількості використаних ресурсів або частоти помилок. Наприклад, гра може зменшувати кількість противників після кількох невдалих спроб або збільшувати інтенсивність подій, якщо користувач демонструє стабільно високі

результати. Ігрові системи, які відповідають певним правилам, є відносно простими для реалізації, але можуть бути недостатньо гнучкими, оскільки реагують лише на обмежений набір ситуацій.

Більш складним засобом є динамічне налаштування складності, за якого параметри гри змінюються автоматично на основі аналізу поточного стану гравця. Такі системи можуть враховувати точність дій, кількість помилок, отриману та завдану шкоду, тривалість сесії, темп проходження, рівень ресурсів і поведінкові ознаки. На відміну від статичних режимів, динамічне налаштування дозволяє змінювати складність у реальному часі, не вимагаючи від користувача ручного втручання.

До сучасних засобів налаштування складності також належать системи керування інтенсивністю ігрових подій. Вони можуть змінювати частоту появи противників, кількість небезпечних ситуацій, доступність допоміжних ресурсів, агресивність штучного інтелекту або тривалість пауз між напруженими епізодами. Такі засоби спрямовані не лише на зміну формальної складності, а й на керування загальним рівнем напруження під час проходження.

Практичні приклади такого підходу можна спостерігати у відомих ігрових системах. У серії Left 4 Dead система AI Director у режимі реального часу аналізує стан і результативність гравців та відповідно впливає на темп проходження, розміщення противників, інтенсивність їх появи й доступність окремих ігрових подій [29; 30]. У Resident Evil 4 також використано приховане коригування складності, яке змінює окремі параметри проходження залежно від результативності користувача [45]. Такі приклади демонструють, що сучасні системи складності можуть керувати не лише числовими характеристиками противників, а й загальною динамікою ігрового процесу.

Важливим напрямом розвитку систем складності є використання багатофакторного аналізу. На відміну від підходів, які враховують лише один показник, багатофакторні системи поєднують кілька груп даних: результативність гравця, поточний стан, поведінкові характеристики,

тривалість сесії, рівень ризику та стабільність дій. Це дає змогу точніше визначати, чи справді гравець потребує зниження складності, чи навпаки – демонструє достатній рівень підготовки для її підвищення.

Отже, сучасні засоби налаштування складності можна поділити на статичні, ручні, засновані на правилах та динамічні. Найбільш перспективними є адаптивні системи, які працюють у реальному часі та враховують не лише результативність гравця, а й контекст ігрової сесії. Саме такі системи створюють основу для подальшого аналізу підходів до адаптивного налаштування складності та розроблення власної моделі.

1.2. Аналіз сучасних підходів до адаптивного налаштування складності

У сучасних ігрових системах можна виділити кілька основних підходів до формування складності. Вони відрізняються способом збору даних, гнучкістю, швидкістю реакції, рівнем обчислювальної складності та здатністю враховувати поведінковий контекст.

Найпростішим підходом до формування ігрової складності є статична складність. Її сутність полягає в тому, що рівень складності задається до початку проходження або на початковому етапі гри й надалі залишається незмінним. Зазвичай такий підхід реалізується через вибір одного з фіксованих режимів, наприклад «легкий», «середній» або «важкий». Кожен із цих режимів має наперед визначені параметри: кількість противників, рівень їхньої агресивності, обсяг доступних ресурсів, швидкість розвитку подій, кількість помилок, які може допустити гравець, та інші характеристики ігрового процесу (рис. 1.1).



Рисунок 1.1 – Статична складність

Перевагою статичної складності є простота реалізації та зрозумілість для користувача. Розробник заздалегідь визначає набір параметрів для кожного рівня складності, а гравець самостійно обирає найбільш прийнятний варіант відповідно до власного досвіду. Такий підхід не потребує складної системи збору даних, аналізу поведінки користувача або додаткових алгоритмів прийняття рішень. Завдяки цьому статична складність часто використовується в іграх із лінійною структурою проходження або в системах, де важливо зберегти повний контроль над балансом.

Водночас основним недоліком статичної складності є її низька здатність реагувати на зміну стану гравця. Обраний на початку рівень може виявитися невідповідним уже під час проходження. Наприклад, менш досвідчений користувач може зіткнутися з надмірним навантаженням, якщо складність була обрана неправильно або якщо окрема ігрова ситуація виявилася складнішою за очікувану. Навпаки, досвідчений гравець може швидко адаптуватися до фіксованих умов, після чого гра втрачає необхідний рівень напруження.

Другим підходом є адаптація на основі правил, за якої зміна складності відбувається відповідно до наперед визначених умов. На відміну від статичної

складності, така система вже може реагувати на окремі зміни в ігровому процесі. Проте її реакція не є повністю гнучкою, оскільки вона обмежується набором правил, заданих розробником заздалегідь.

Принцип роботи такого підходу полягає у використанні логічних умов типу «якщо - то». Наприклад, якщо гравець кілька разів поспіль зазнає поразки, система може знизити рівень складності, зменшити кількість противників, збільшити кількість доступних ресурсів або послабити інтенсивність подій. Якщо ж гравець демонструє стабільно високі результати, складність може бути підвищена шляхом збільшення кількості небезпечних ситуацій, пришвидшення темпу гри або посилення вимог до точності дій (рис. 1.2).



Рисунок 1.2 – Адаптація на основі правил

Перевагою адаптації на основі правил є відносна простота реалізації. Для її роботи не обов’язково створювати складну математичну модель або систему прогнозування стану гравця. Достатньо визначити перелік ключових показників, порогових значень і відповідних реакцій системи. Завдяки цьому

такий підхід є зрозумілим для розробника, легко тестується та може бути швидко впроваджений у базову ігрову систему.

Водночас адаптація на основі правил має низку обмежень. Насамперед вона реагує лише на ті ситуації, які були передбачені під час розроблення. Якщо поведінка гравця не відповідає заданим умовам, система може не змінити складність навіть тоді, коли це було б доцільно. Наприклад, гравець може не програвати, але постійно перебувати у критичному стані, витрачати багато ресурсів або грати занадто пасивно. Якщо такі випадки не описані окремими правилами, система не зможе коректно оцінити ситуацію, оскільки її рішення залежать від заздалегідь сформованої бази правил (рис. 1.3).

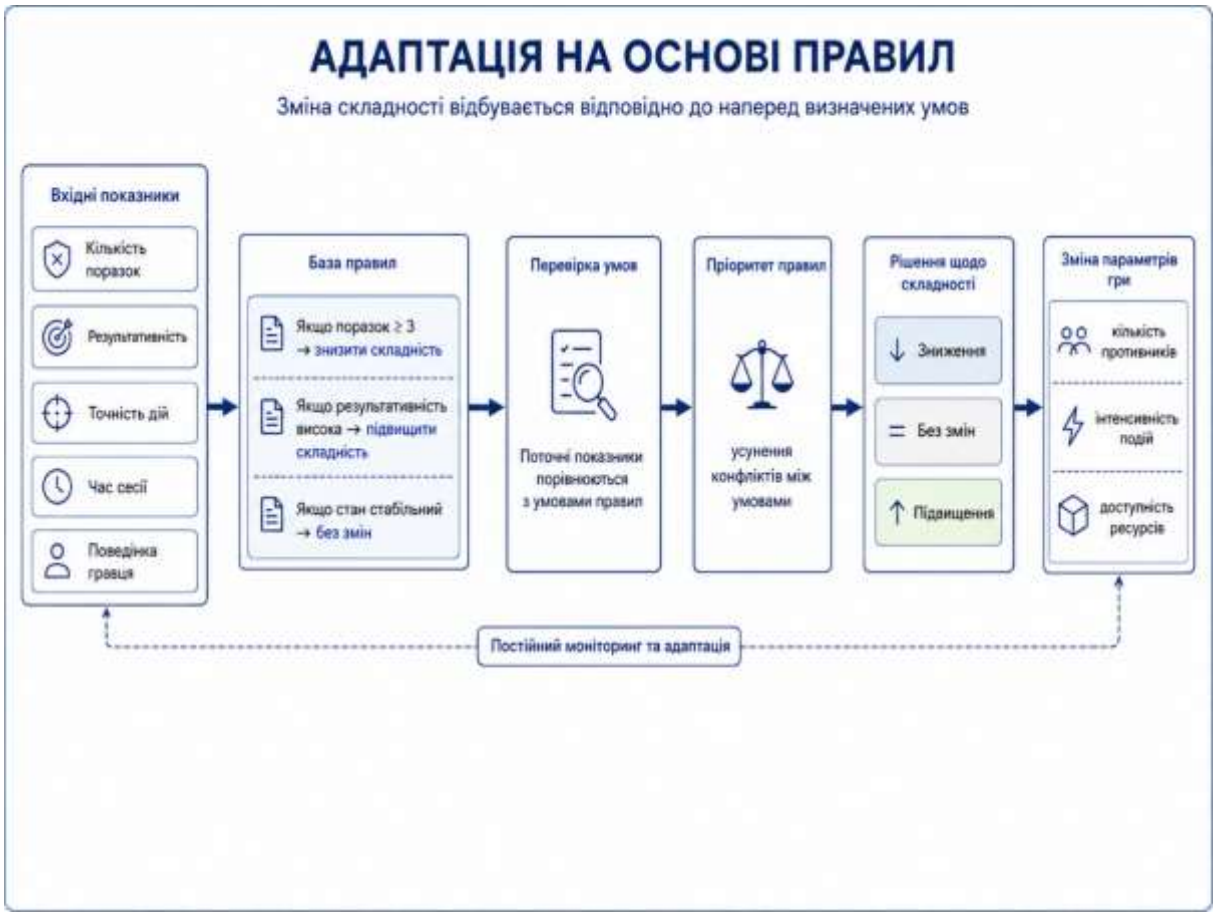


Рисунок 1.3 – Структурна схема адаптації складності на основі правил

Іншим недоліком є можливість передбачуваності для користувача. Якщо гравець розуміє, за якими умовами система знижує складність, він може навмисно впливати на ці показники. Наприклад, користувач може спеціально програвати, діяти неефективно або штучно знижувати власні результати, щоб

отримати полегшені умови проходження. Через це системи на основі правил можуть бути вразливими до зловживань, особливо якщо вони оцінюють лише один або два показники.

На відміну від систем на основі правил, цей підхід дає змогу точніше оцінювати поточний стан користувача, оскільки враховує не одну окрему подію, а сукупність показників. Наприклад, якщо гравець має низьку точність, часто отримує шкоду та витрачає багато часу на проходження, система може зробити висновок про надмірну складність і поступово знизити її. Якщо ж користувач демонструє високу результативність, швидко проходить ігрові етапи та рідко помиляється, складність може бути підвищена для підтримання необхідного рівня виклику (рис. 1.4).



Рисунок 1.4 – Структурна схема адаптації складності на основі продуктивності

На рис. 1.4 подано загальну ідею адаптації складності на основі продуктивності гравця. Система аналізує кількісні показники результативності та відповідно змінює рівень складності: у разі низької продуктивності

складність може знижуватися, за стабільних результатів залишатися без змін, а за високої продуктивності – підвищуватися.

Принцип роботи такого підходу полягає у постійному зборі телеметричних даних і перетворенні їх на узагальнену оцінку стану гравця. Окремі показники можуть мати різну вагу залежно від їхнього значення для конкретної ігрової системи. Наприклад, у динамічних іграх більшу роль можуть відігравати точність, швидкість реакції та кількість отриманої шкоди, тоді як у стратегічних або тактичних системах важливішими можуть бути темп ухвалення рішень, ефективність використання ресурсів і кількість помилкових дій (рис. 1.5).



Рисунок 1.5 – Структурна схема адаптації складності за показниками продуктивності гравця

На рис. 1.5 наведено структурну схему такого підходу. Вхідні показники гравця збираються під час ігрової сесії, після чого система виконує їх обробку, формує оцінку результативності та приймає рішення щодо зниження, підвищення або збереження поточного рівня складності.

Перевагою адаптації на основі продуктивності є вища точність порівняно зі статичною складністю та простими системами на основі правил. Такий підхід дозволяє враховувати фактичні дії користувача, а не лише вибраний рівень складності або окремі події. Завдяки цьому система може краще підтримувати баланс між надмірною легкістю та надмірною складністю.

Водночас цей підхід має певні обмеження. Насамперед він значною мірою залежить від правильного вибору показників. Якщо система аналізує лише бойову ефективність або кількість помилок, вона може неправильно інтерпретувати ситуацію. У такому випадку підвищення складності може погіршити баланс ігрового процесу.

Четвертим підходом є прогнозна та комбінована адаптація складності. Прогнозна адаптація використовує попередні значення показників для передбачення майбутнього стану гравця. Наприклад, система може не чекати критичного падіння здоров’я, серії помилок або різкого зниження результативності, а заздалегідь пом’якшити складність на основі динаміки зміни показників. Такий підхід є більш гнучким порівняно з адаптацією лише за поточними значеннями, однак потребує складнішої реалізації, згладжування даних і захисту від помилкових реакцій на короткочасні коливання (рис. 1.6).



Рисунок 1.6 – Комбінована модель адаптивного налаштування складності

Запропонований у роботі підхід доцільно розглядати як комбіновану модель адаптивного налаштування складності, що розширює ідею прогнозної адаптації.

На відміну від підходів, які враховують лише продуктивність гравця або окремі події, запропонована модель поєднує аналіз результативності, рівня ризику, ігрового тиску, тривалості сесії та ознак потенційного зловживання системою. Це дозволяє приймати рішення щодо зміни складності не лише за фактом погіршення або покращення результатів, а й з урахуванням ширшого контексту ігрової ситуації. У табл. 1.1 використано скорочення DDA – Dynamic Difficulty Adjustment.

Таблиця 1.1 – Порівняння підходів до налаштування складності

Критерій порівняння	Статична складність	Системи на основі правил	DDA за продуктивністю	Запропонована модель
Урахування стану гравця	1	2	4	4
Урахування ризику	1	1	2	5
Урахування поведінкового контексту	1	2	3	5
Захист від зловживань	1	1	2	5
Плавність зміни складності	1	2	3	4
Придатність до роботи в реальному часі	5	5	4	5
Простота реалізації	5	4	3	1
Масштабованість	2	3	4	5
Сумарна оцінка	17	20	25	37

Оцінювання виконано за шкалою від 1 до 5, де 1 - низький рівень відповідності критерію, 5 - високий рівень відповідності критерію.

Як видно з табл. 1.1, статична складність має найнижчу сумарну оцінку - 17 балів. Це пояснюється тим, що вона майже не враховує поточний стан гравця, рівень ризику, поведінковий контекст і не має механізмів захисту від зловживань. Водночас цей підхід залишається простим у реалізації та добре придатним до роботи в реальному часі, оскільки не потребує складних обчислень або постійного аналізу даних.

Системи на основі правил отримали 20 балів. Вони є гнучкішими порівняно зі статичною складністю, оскільки можуть змінювати параметри гри за наперед визначеними умовами. Проте такі системи реагують лише на обмежений набір ситуацій, заданих розробником, тому недостатньо враховують ризик, поведінковий контекст і можливість навмисного впливу на адаптивний механізм.

DDA за продуктивністю має вищу сумарну оцінку - 25 балів. Цей підхід краще враховує фактичні результати гравця, зокрема точність, кількість помилок, завдану шкоду, втрати здоров’я та час проходження. Завдяки цьому система може точніше реагувати на зміну результативності користувача. Однак така модель зазвичай обмежено враховує ризик, контекст сесії та ознаки потенційного зловживання.

Найвищу сумарну оцінку отримала запропонована модель - 37 балів. Її перевага полягає у поєднанні кількох груп показників: стану гравця, рівня ризику, поведінкового контексту, плавності зміни складності, придатності до роботи в реальному часі та масштабованості. Окремою перевагою є наявність захисту від зловживань, що дозволяє зменшити ризик навмисного заниження показників гравцем для штучного спрощення проходження.

Водночас запропонована модель має нижчу оцінку за критерієм простоти реалізації, оскільки потребує складнішої структури, обробки більшої кількості показників, згладжування даних і перевірки поведінкових ознак. Проте цей недолік компенсується вищою точністю адаптації та кращою здатністю враховувати комплексний стан ігрової сесії.

1.3. Постановка задачі

В умовах, де ігровий процес потребує гнучкого регулювання складності, важливим є не лише вибір початкового рівня навантаження, а й можливість його зміни відповідно до стану гравця під час сесії. Аналіз сучасних підходів показав, що статична складність є простою в реалізації, однак не враховує зміну навичок і стану користувача. Системи на основі правил здатні реагувати на окремі події, проте залежать від наперед визначених умов. Адаптація на основі продуктивності гравця є точнішою, але зазвичай недостатньо враховує ризик, поведінковий контекст і можливість зловживання системою.

У зв’язку з цим більш перспективним є використання комбінованої моделі адаптивного налаштування складності, яка поєднує кілька груп показників. Така модель повинна враховувати не лише результативність гравця, а й рівень ризику, ігровий тиск, тривалість сесії, зміну поведінки та ознаки навмисного впливу на адаптивний механізм. Це дозволяє приймати рішення щодо зміни складності не за одним окремим параметром, а на основі комплексної оцінки поточного стану ігрової сесії.

Для реалізації поставленої задачі необхідно визначити набір вхідних показників, які можуть характеризувати стан гравця. До таких показників належать точність дій, кількість помилок, завдана шкода, втрати здоров’я, тривалість сесії, рівень ризику, поведінкова активність та інші кількісні або контекстні характеристики. На основі цих даних система повинна формувати узагальнену оцінку стану гравця та визначати доцільний напрям зміни складності.

Окрему увагу необхідно приділити проблемі стабільності адаптації. Якщо складність змінюється занадто різко або часто, це може негативно впливати на сприйняття ігрового процесу. Тому модель повинна передбачати механізми згладжування змін, щоб адаптація відбувалася поступово й не створювала відчуття штучного втручання. Крім того, доцільним є використання елементів прогнозування, які дозволяють враховувати не лише поточний стан, а й тенденцію зміни показників.

Важливою частиною поставленої задачі є також захист від зловживань. У системах адаптивного налаштування складності існує ризик, що гравець може навмисно погіршувати власні показники для штучного зниження рівня складності. Наприклад, користувач може діяти пасивно, часто помилятися або навмисно втрачати ресурси. Тому модель повинна мати механізми виявлення таких ситуацій і не допускати некоректного зниження складності у випадках, коли погіршення показників має штучний характер.

Таким чином, поставлена задача полягає у розробленні комбінованої моделі адаптивного налаштування складності, яка працює в режимі реального часу, враховує кількісні та поведінкові показники гравця, оцінює ризик і тиск ігрової сесії, забезпечує плавну зміну складності та містить механізми захисту від зловживань.

Отже, для реалізації поставленої задачі необхідно виконати такі завдання:

1. Проаналізувати сучасні підходи до налаштування складності в комп’ютерних іграх та визначити їхні переваги й обмеження.

2. Розробити систему показників для оцінювання стану гравця, що враховує продуктивність, рівень ризику, ігровий тиск, тривалість сесії та поведінкові особливості.

3. Спроєктувати комбіновану модель адаптивного налаштування складності з використанням згладжування змін, часової ескалації та механізмів захисту від зловживань.

4. Провести перевірку роботи запропонованої моделі в умовах реального часу, порівняти її з фіксованими режимами складності та проаналізувати вплив на стабільність і збалансованість ігрового процесу.

РОЗДІЛ 2 МОДЕЛЮВАННЯ ПРОЦЕСУ АДАПТИВНОГО НАЛАШТУВАННЯ СКЛАДНОСТІ

2.1. Математична модель адаптивного налаштування складності

Проєктування модуля адаптивного налаштування складності передбачає створення математичної моделі, яка дає змогу оцінювати стан гравця в режимі реального часу та на основі цієї оцінки визначати доцільний рівень складності. Основною особливістю запропонованої моделі є багатофакторний підхід: складність формується не за одним окремим показником, а на основі сукупності параметрів, що характеризують продуктивність гравця, рівень ризику, ігровий тиск, тривалість сесії та ознаки можливої маніпуляції системою.

У межах моделі всі основні показники приводяться до нормалізованого діапазону від 0 до 1. Це дозволяє об’єднувати різні за природою величини: точність, кількість помилок, завдану шкоду, отриману шкоду, тривалість сесії, цінність накопичених ресурсів та поведінкові характеристики.

Для обмеження значень використовується функція нормалізації:

$$C_{01(x)} = \min(1, \max(0, x)) \tag{2.1}$$

де $C_{\{01\}(x)}$ – значення показника, обмежене діапазоном від 0 до 1;

(x) – поточне значення показника.

Загальний рівень складності можна подати як функцію кількох інтегральних показників:

$$\mathrm{D} = \mathrm{f}(\mathrm{P}, \mathrm{R}, \mathrm{T}, \mathrm{E}, \mathrm{A}) \tag{2.2}$$

де (D) –поточний рівень складності;

(P) –показник продуктивності гравця;

(R) –показник ризику;

(T) –показник ігрового тиску;

(E) –показник часової ескалації;

(A) – показник підозрілої поведінки або можливого зловживання системою.

Показник продуктивності гравця (P) характеризує ефективність дій користувача під час ігрової сесії. Він формується на основі бойової ефективності, тривалості виживання, цінності накопичених ресурсів і стабільності поведінки.

Бойова ефективність може бути подана так:

$$C = \mathrm{C}_{01}\left(0{,}14K + 0{,}45Acc + 0{,}20C_{01}\left(\frac{Dmg_{(out)}}{450}\right)\right) \quad (2.3)$$

де (C) – бойова ефективність;

(K) – кількість успішних дій або перемог над противниками;

Acc – точність дій гравця;

$Dmg_{\{out\}}$ – обсяг завданої шкоди.

Показник тривалості сесії нормалізується за формулою:

$$\mathrm{S} = \mathrm{C}_{01\left(\frac{\mathrm{t}}{420}\right)} \quad (2.4)$$

де (S) – нормалізована тривалість сесії;

(t) – поточний час сесії в секундах.

Показник цінності накопичених здобичі для оцінки продуктивності визначається так:

$$L_p = \mathrm{C}_{01\left(\frac{L}{900}\right)} \quad (2.5)$$

де $\mathrm{L_p}$ – нормалізована цінність накопиченої здобичі;

(L)– поточна цінність здобичі.

Показник стабільності поведінки враховує отриману шкоду та бездіяльність гравця:

$$Q = C_{01\left(1-0{,}60P_d-\frac{I}{180}\right)} \quad (2.6)$$

де (Q) – показник стабільності;

$\mathrm{P_d}$ – тиск від отриманої шкоди;

(I) – час бездіяльності гравця.

Тиск від отриманої шкоди визначається як:

$$\mathrm{P_d} = \mathrm{C}_{01}\left(\frac{\mathrm{Dmg}_{in}}{100}\right) \quad (2.7)$$

де $Dmg_{\{in\}}$ – обсяг отриманої шкоди.

Тоді попереднє значення продуктивності гравця можна подати так:

$$P_{raw} = 0{,}45C + 0{,}15S + 0{,}25L_p + 0{,}15Q \quad (2.8)$$

Отже, продуктивність гравця залежить не лише від точності або кількості успішних дій, а й від тривалості сесії, накопичених здобичі і стабільності поведінки.

Показник ігрового тиску (T) відображає поточне навантаження на гравця. Він враховує отриману шкоду, тривалість сесії, ризиковану поведінку та накопичені ресурси:

$$\mathrm{T_{raw}} = 0{,}55\mathrm{P_d} + 0{,}15\mathrm{C}_{01\left(\frac{\mathrm{t}}{360}\right)} + \\ +0{,}2\mathrm{B_r} + 0{,}1\mathrm{C}_{01\left(\frac{\mathrm{L}}{1000}\right)} \quad (2.9)$$

де $\mathrm{T_{\{raw\}}}$ – попереднє значення ігрового тиску;

$\mathrm{B_r}$ – показник ризикованої поведінки;

(L) – цінність накопичених здобичі.

Показник ризику (R) характеризує рівень потенційних втрат і важливість поточної ситуації. Він враховує цінність здобичі, тривалість сесії та ризиковану поведінку:

$$R_{raw} = 0{,}55C_{01\left(\frac{L}{750}\right)} + 0{,}25C_{01\left(\frac{t}{420}\right)} + 0{,}2B_r \quad (2.10)$$

де $\mathrm{R_{\{raw\}}}$ – попереднє значення ризику;

(L) – цінність накопичених здобичі;

(t) – тривалість сесії;

$\mathrm{B_r}$ – показник ризикованої поведінки.

Для уникнення різких змін складності використовується експоненційне згладжування. Воно застосовується до основних інтегральних показників: продуктивності, ризику та ігрового тиску.

Згладжене значення визначається за формулою:

$$X_t = 0{,}88X_{t-1} + 0{,}12X_{raw} \quad (2.11)$$

де $\mathrm{X_t}$ – згладжене значення показника в поточний момент часу;

$\mathrm{X_{\{t-1\}}}$– попереднє згладжене значення;

$X_{\{raw\}}$ – нове розраховане значення показника;

$0{,}12$ – коефіцієнт згладжування.

Таким чином, одинична помилка гравця або короткочасне покращення результатів не призводить до миттєвої зміни складності. Модель реагує на загальну тенденцію, а не на випадкові коливання.

Часовий тиск або ескалація сесії визначається через нормалізоване значення часу:

$$E = C_{01\left(\frac{t}{480}\right)} \tag{2.12}$$

де (E) – показник часової ескалації;

(t) – поточна тривалість сесії.

Ескалація дає змогу поступово підвищувати складність зі збільшенням тривалості сесії. Це необхідно для того, щоб ігровий процес не залишався статичним і підтримував необхідний рівень напруження протягом проходження.

На основі згладжених показників формується цільове значення складності:

$$D_{target} = C_{01(0{,}22+0{,}38P+0{,}30R+0{,}14E-0{,}18T)} \tag{2.13}$$

де $D_{\{target\}}$ – цільовий рівень складності;

(P) – згладжений показник продуктивності;

(R) – згладжений показник ризику;

(E) – показник часової ескалації;

(T) – згладжений показник ігрового тиску.

Логіка формули полягає в тому, що висока продуктивність, високий ризик і збільшення тривалості сесії підвищують цільову складність. Водночас високий ігровий тиск зменшує її, оскільки надмірне навантаження на гравця може потребувати пом’якшення умов.

Після визначення цільового значення складність не змінюється миттєво, а поступово наближається до нього:

$$D_t = \mathrm{MoveTowards}\left(D_{t-1}, D_{target}, \Delta t * 0{,}05\right) \tag{2.14}$$

де D_t – поточне значення складності;

$D_{\{t-1\}}$ – попереднє значення складності;

$D_{\{target\}}$ – цільове значення складності;

Δt – проміжок часу між оновленнями;

$0{,}05$ – коефіцієнт швидкості зміни складності.

Це забезпечує інерційність моделі: складність змінюється плавно, без різких стрибків, що позитивно впливає на стабільність ігрового процесу.

Окремим елементом моделі є показник підозрілої поведінки (A), який використовується для захисту від зловживань. Його призначення полягає у виявленні ситуацій, коли гравець може навмисно погіршувати власні результати для штучного зниження складності.

До таких ситуацій можуть належати: тривала бездіяльність за відсутності результативних дій, рання поразка без активної участі, отримання значної шкоди без завдання шкоди у відповідь, а також часті спроби завершити сесію в короткий проміжок часу.

Узагальнено показник підозри можна подати як суму окремих умов:

$$A_{raw} = A_{idle} + A_{early} + A_{passive} + A_{repeat} \quad (2.15)$$

де $A_{\{idle\}}$ – підозра на бездіяльність;

$A_{\{early\}}$ – підозра на навмисну ранню поразку;

$A_{\{passive\}}$ – підозра на пасивне отримання шкоди;

$A_{\{repeat\}}$ – підозра на повторювані дії для обходу системи.

Для стабільності цей показник також змінюється поступово:

$$A_t = \mathrm{MoveTowards}\left(A_{t-1}, C_{01(A_{raw})}, 0{,}08\right) \quad (2.16)$$

Якщо показник підозрілої поведінки перевищує визначений поріг, система блокує зниження складності:

$$A_t \geq 0{,}45 \Rightarrow D_{target} = \max\left(D_{target}, D_{t-1}\right) \quad (2.17)$$

Це означає, що за наявності ознак зловживання модель не дозволяє цільовому рівню складності стати нижчим за поточний. Такий механізм зменшує ймовірність ситуацій, у яких гравець навмисно погіршує показники для отримання спрощених умов.

Розрахований рівень складності може впливати на параметри ігрового процесу. Наприклад, на його основі можна визначати інтенсивність появи противників, допустиму кількість активних загроз, агресивність штучного інтелекту, інтервали між подіями та доступність здобичі.

Загальний тиск середовища можна подати як:

$$G = C_{01(0,65D+0,45R-0,25T)} \tag{2.18}$$

де (G) – узагальнений тиск середовища;

(D) – поточний рівень складності;

(R) – рівень ризику;

(T) – ігровий тиск на гравця.

На основі цього значення можуть визначатися похідні параметри, наприклад:

$$N = \text{round}\big(lerp(3,9,G)\big) \tag{2.19}$$

$$I_s = lerp(6,5; 2,6; G) \tag{2.20}$$

$$M = lerp(1,0; 1,6; G) \tag{2.21}$$

де (N) – допустима кількість активних загроз;

(I_s) – інтервал між появою нових подій або противників;

(M) – множник агресивності;

(G) – узагальнений тиск середовища.

Отже, математична модель адаптивного налаштування складності працює в нормалізованому діапазоні від 0 до 1 і складається з кількох основних компонентів: нормалізації вхідних показників, обчислення продуктивності, ризику та ігрового тиску, експоненційного згладжування, формування цільової складності, поступового переходу до неї та перевірки ознак зловживання. На відміну від простої адаптації за продуктивністю, запропонована модель враховує ширший контекст ігрової сесії, зокрема ризик, тривалість, тиск на гравця та поведінкові ознаки. Це дозволяє забезпечити більш стабільне й обґрунтоване регулювання складності в режимі реального часу.

2.2. Архітектура модуля адаптивного налаштування складності та інтеграція з ігровими системами

Архітектура модуля адаптивного налаштування складності побудована за компонентним принципом. Такий підхід передбачає поділ системи на окремі функціональні блоки, кожен із яких відповідає за конкретний етап обробки даних: збір показників, аналіз стану гравця, визначення рівня складності, перевірку ознак зловживання та передачу результатів до ігрових підсистем.

У загальному вигляді модуль адаптивного налаштування складності складається з таких основних блоків (рис. 2.1).

1. Блок збору телеметрії;
2. блок аналізу продуктивності, ризику та ігрового тиску;
3. блок часової ескалації;
4. блок виявлення підозрілої поведінки;
5. блок формування поточного стану складності;
6. блок інтеграції з ігровими системами;
7. блок візуалізації та відлагодження.



Рисунок 2.1 – Загальна архітектура модуля адаптивного налаштування складності

Першим етапом роботи модуля є збір телеметричних даних. До таких даних належать бойові показники, точність дій, обсяг завданої та отриманої шкоди, тривалість сесії, цінність накопичених ресурсів, рівень активності гравця та інші поведінкові характеристики. Зібрані показники не використовуються окремо, а передаються до аналітичного блоку для подальшої обробки.

Аналітичний блок виконує перетворення вхідних даних у декілька узагальнених оцінок. Основними з них є показник продуктивності гравця, показник ризику та показник ігрового тиску. Показник продуктивності характеризує ефективність дій користувача. Показник ризику відображає потенційну цінність втрат або важливість поточного стану сесії. Показник ігрового тиску використовується для визначення того, наскільки інтенсивним є навантаження на гравця в поточний момент. (рис. 2.2).

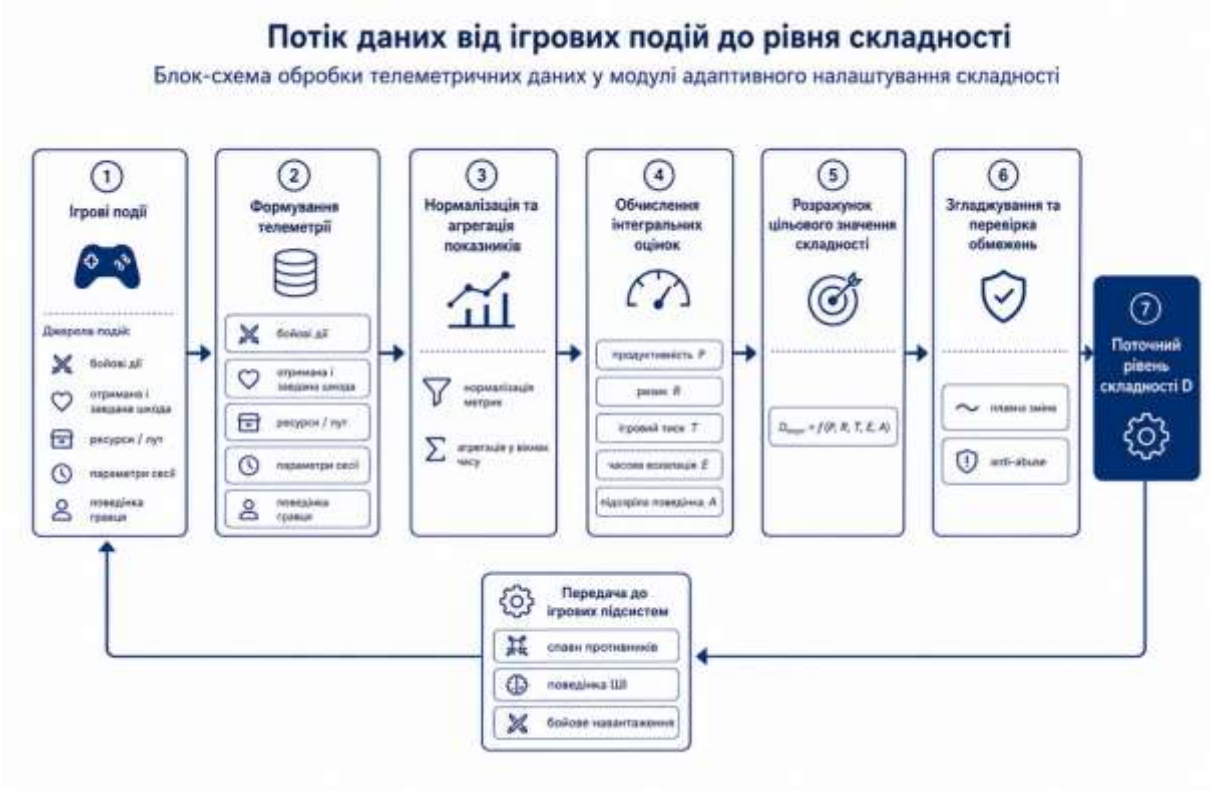


Рисунок 2.2 – Потік даних у модулі адаптивного налаштування складності

Окремим компонентом архітектури є блок часової ескалації. Його призначення полягає у поступовому підвищенні напруження залежно від тривалості ігрової сесії. Це дає змогу уникнути ситуації, коли після тривалого перебування в активній фазі гри складність залишається незмінною. Ескалація не замінює основну модель адаптації, а доповнює її, додаючи часовий фактор до загального розрахунку складності.

Важливу роль у структурі модуля відіграє блок захисту від зловживань. Він аналізує поведінкові ознаки, які можуть свідчити про навмисне заниження результатів гравцем. До таких ознак належать тривала бездіяльність, пасивне отримання шкоди, повторювані помилки без активних дій або інші ситуації, у яких користувач може намагатися штучно знизити складність.

Результати роботи блоків аналізу не використовуються ізольовано. Вони поєднуються в межах єдиного процесу прийняття рішення, де кожен показник впливає на складність залежно від свого значення та ролі в поточній сесії. Наприклад, висока продуктивність або значна цінність накопичених ресурсів можуть сприяти підвищенню складності, тоді як надмірний ігровий тиск або ознаки нестабільної поведінки можуть обмежувати різке збільшення навантаження. Завдяки цьому модуль формує не миттєву реакцію на окрему подію, а узгоджену оцінку поточного стану гри.

Після обробки всіх вхідних даних формується поточний стан адаптивної складності. Його доцільно розглядати як узагальнений набір параметрів, що містить не лише числове значення складності, а й додаткові характеристики: рівень продуктивності, ризику, ігрового тиску, етап ескалації, стан захисту від зловживань та параметри впливу на ігрове середовище. Такий підхід дозволяє передавати іншим підсистемам не окремі розрізнені значення, а цілісний опис поточного стану адаптації. (рис. 2.3).



Рисунок 2.3 – Формування поточного стану адаптивної складності

Інтеграція модуля з ігровими системами здійснюється через передачу розрахованих параметрів до підсистем, що відповідають за ігрове навантаження. Насамперед це стосується системи появи противників, поведінки штучного інтелекту, інтенсивності подій, бойового тиску та потенційної винагороди. Завдяки цьому модуль не лише розраховує складність, а й впливає на практичні параметри ігрового процесу.

Система появи противників може використовувати поточний рівень складності для зміни кількості активних загроз, інтервалу між появою нових противників, співвідношення типів противників і загальної інтенсивності подій. За низької складності такі параметри можуть залишатися помірними, тоді як зі зростанням ризику або продуктивності гравця система поступово підвищує тиск.

Поведінка штучного інтелекту також може адаптуватися до поточного стану складності. Наприклад, можуть змінюватися дальність виявлення гравця, швидкість переслідування, тривалість пошуку, схильність до відступу,

агресивність і частота бойових дій. Такий підхід дозволяє впливати не лише на кількість противників, а й на якість їхньої поведінки. (рис. 2.4).



Рисунок 2.4 – Інтеграція модуля адаптивної складності з ігровими системами

Окремо в архітектурі передбачено блок візуалізації та відлагодження. Він потрібен для контролю роботи системи під час тестування. Такий блок може відображати поточний рівень складності, значення продуктивності, ризику, тиску, етап ескалації, стан блокування зниження складності та інші службові показники. Наявність візуалізації спрощує перевірку коректності роботи моделі та дозволяє аналізувати, чому система ухвалила певне рішення.

2.3. Вибір мови програмування та середовища розробки.

Для реалізації модуля адаптивного налаштування складності необхідно обрати таке середовище розробки, яке дозволяє працювати з ігровими об’єктами в режимі реального часу, збирати телеметричні дані, обробляти події, змінювати параметри ігрового середовища та інтегрувати алгоритми адаптації з іншими підсистемами гри. Оскільки запропонований модуль повинен не лише

виконувати математичні розрахунки, а й впливати на поведінку ігрових систем, важливими критеріями вибору є підтримка компонентної архітектури, зручність реалізації логіки в реальному часі, наявність інструментів для роботи з об’єктами сцени, підтримка подій та можливість швидкого тестування.

Для розроблення такого модуля можуть використовуватися різні середовища та мови програмування. Найбільш доцільно розглянути Unity з мовою C#, Unreal Engine з C++ або Blueprints, Godot із GDScript або C#, а також Python-орієнтоване середовище. Кожен із цих варіантів має власні переваги та обмеження залежно від поставленого завдання.

Unity є одним із найбільш придатних середовищ для реалізації модуля адаптивного налаштування складності, оскільки підтримує компонентний підхід, роботу з об’єктами сцени, подієву взаємодію між системами та швидке прототипування ігрової логіки. Мова C# добре підходить для побудови модульної архітектури, оскільки підтримує об’єктно-орієнтоване програмування, класи, структури даних, колекції, події та математичні операції, необхідні для обробки телеметричних показників.

Unreal Engine має потужні можливості для створення складних ігрових систем, високоякісної графіки та продуктивних проєктів. Проте використання C++ підвищує складність розробки, особливо для невеликого дослідницького модуля. Візуальна система Blueprints є зручною для швидкого створення логіки, але для реалізації багатофакторної математичної моделі, системи згладжування та антизловживання вона може бути менш зручною порівняно з текстовим кодом.

Godot є легшим і доступнішим середовищем, яке також підтримує компонентну логіку через вузли та дозволяє швидко створювати прототипи. Однак у контексті реалізації складного DDA-модуля Unity має перевагу завдяки ширшій екосистемі, зручнішій інтеграції C#, більшій кількості готових інструментів і кращій придатності до демонстраційного ігрового проєкту середнього масштабу.

Python може бути корисним для математичного моделювання, аналізу даних або створення окремих дослідницьких прототипів. Проте для інтеграції з ігровими системами в режимі реального часу він є менш придатним. Python не є типовою основною мовою для реалізації ігрової логіки в Unity або Unreal Engine, тому його використання ускладнило б інтеграцію адаптивної системи з об’єктами сцени, поведінкою противників, подіями та параметрами ігрового середовища.

Під час вибору середовища розробки враховувалася не лише можливість реалізації окремих алгоритмів, а й здатність обраної платформи забезпечити повну інтеграцію модуля з ігровим процесом. Для DDA-модуля важливо, щоб система могла отримувати дані від ігрових подій, обробляти їх у реальному часі, змінювати параметри противників, передавати значення до підсистем появи об’єктів та забезпечувати візуальний контроль результатів тестування. Тому середовище розробки повинно підтримувати компонентну структуру, роботу з об’єктами сцени, подіями, інтерфейсом користувача та засобами налагодження.

Для узагальнення вибору середовища розробки було порівняно кілька можливих варіантів: Unity з мовою C#, Unreal Engine із C++ та Blueprints, Godot із GDScript або C#, а також Python як окреме дослідницьке середовище. Порівняння виконано за критеріями придатності до ігрової логіки в реальному часі, підтримки компонентної архітектури, зручності реалізації математичної моделі, інтеграції з ігровими об’єктами, складності освоєння та придатності до демонстраційного ігрового проєкту. Узагальнені результати наведено в табл. 2.1.

Таблиця 2.1 – Порівняння середовищ і мов програмування для реалізації DDA-модуля

Критерій порівняння	Unity + C#	Unreal Engine + C++ / Blueprints	Godot + GDScript / C#	Python-орієнтоване середовище
Придатність до ігрової логіки в реальному часі	Висока	Висока	Середня / висока	Низька / середня
Підтримка компонентної архітектури	Висока	Висока	Висока	Обмежена
Зручність реалізації математичної моделі	Висока	Середня	Середня	Висока
Зручність інтеграції з ігровими об’єктами	Висока	Висока	Середня	Низька
Складність освоєння	Середня	Висока	Низька / середня	Низька
Швидкість прототипування	Висока	Середня	Висока	Висока, але поза ігровою інтеграцією
Підтримка подієвої взаємодії	Висока	Висока	Середня	Потребує додаткової реалізації
Придатність для демонстраційного ігрового проєкту	Висока	Середня / висока	Середня	Низька
Масштабованість рішення	Висока	Висока	Середня	Обмежена для ігрового середовища
Доцільність для реалізації DDA-модуля	Найбільш доцільна	Доцільна, але складніша	Можлива, але менш зручна	Доцільна лише для окремих розрахунків

З табл. 2.1 видно, що Unity у поєднанні з мовою C# є найбільш збалансованим варіантом для реалізації модуля адаптивного налаштування складності. Це середовище забезпечує достатній рівень продуктивності, підтримує роботу з ігровими об’єктами в реальному часі, має зручну компонентну структуру та дозволяє реалізовувати як математичну логіку, так і інтеграцію з ігровими підсистемами.

Вибір Unity також обумовлений можливістю швидкої перевірки роботи модуля в умовах ігрової сцени. Для системи адаптивної складності важливо не лише розрахувати числове значення складності, а й перевірити, як воно впливає на параметри ігрового процесу: кількість активних загроз, інтенсивність подій, поведінку штучного інтелекту, бойове навантаження та доступність ресурсів. Unity дозволяє вико нувати таку перевірку без необхідності створення окремого інструментарію для візуалізації та тестування.

Мова C# була обрана через її тісну інтеграцію з Unity та придатність до побудови структурованої програмної архітектури. Вона дозволяє описувати окремі частини системи у вигляді класів і компонентів, працювати з колекціями, подіями, структурами даних і математичними функціями. Це є важливим для реалізації системи, що постійно збирає телеметрію, обчислює інтегральні показники, згладжує значення та приймає рішення щодо зміни складності.

Компонентний підхід Unity також добре відповідає структурі запропонованого DDA-модуля. Окремі частини системи можуть відповідати за збір телеметрії, аналіз продуктивності, оцінку ризику, визначення ігрового тиску, часову ескалацію, перевірку підозрілої поведінки та передачу результатів до ігрових підсистем. Такий поділ спрощує тестування, дає змогу змінювати окремі компоненти без повної перебудови системи та підвищує гнучкість подальшого розвитку модуля.

Крім того, C# є придатною мовою для реалізації алгоритмів, що працюють у реальному часі. У межах DDA-модуля необхідно регулярно оновлювати значення показників, застосовувати нормалізацію, виконувати згладжування,

перевіряти порогові умови та поступово змінювати параметри складності. Такі завдання зручно реалізовувати засобами C#, оскільки мова має достатню виразність, підтримує об’єктно-орієнтоване проєктування та дозволяє створювати зрозумілу структуру коду.

Отже, для реалізації програмного модуля адаптивного налаштування складності обрано середовище Unity та мову програмування C#. Такий вибір є доцільним, оскільки поєднує можливість швидкого прототипування, підтримку компонентної архітектури, зручну інтеграцію з ігровими системами, достатню продуктивність для роботи в реальному часі та засоби для реалізації математичної моделі адаптації складності.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ МОДУЛЯ АДАПТИВНОГО НАЛАШТУВАННЯ СКЛАДНОСТІ

3.1. Реалізація збору даних.

Першим етапом реалізації модуля адаптивного налаштування складності є створення підсистеми збору телеметричних даних. Її основним призначенням є фіксація показників ігрової сесії, які надалі використовуються для оцінювання стану гравця, розрахунку інтегральних параметрів адаптації та прийняття рішення щодо зміни рівня складності.

На відміну від статичних систем складності, у яких параметри задаються до початку проходження та не змінюються під час гри, адаптивна система потребує постійного оновлення інформації про поточний стан користувача. Саме тому в межах реалізації було передбачено збір даних у режимі реального часу. До таких даних належать бойові показники, точність дій, обсяг завданої та отриманої шкоди, цінність накопичених ресурсів, тривалість сесії, рівень бездіяльності, кількість спроб евакуації та окремі поведінкові характеристики користувача.

Збір даних організовано за компонентним принципом. Окремі групи показників фіксуються незалежно одна від одної, після чого передаються до системи аналізу адаптивної складності. Такий підхід дозволяє розділити відповідальність між різними частинами системи та спростити подальшу перевірку коректності роботи кожного елемента.

У межах підсистеми збору даних можна виділити кілька основних груп телеметрії. Перша група пов’язана з бойовою активністю користувача. Вона містить дані про кількість успішних бойових дій, точність, завдану шкоду, отриману шкоду та загальну ефективність взаємодії з противниками. Ці показники використовуються для подальшого визначення продуктивності гравця.

Друга група даних описує ресурсний стан та рівень ризику. До неї належить цінність накопичених ресурсів, а також дані, що відображають потенційні втрати гравця у випадку невдалого завершення сесії. Такі показники

важливі для запропонованої моделі, оскільки складність повинна враховувати не лише бойову ефективність, а й контекст ризику.

Третя група телеметрії пов’язана з часовими параметрами сесії. До неї належать тривалість проходження, часовий тиск, поточний етап сесії та інші значення, які характеризують розвиток ігрового процесу в часі. Ці дані використовуються для реалізації часової ескалації, за якої напруження поступово збільшується зі зростанням тривалості сесії.

Окремо збираються поведінкові показники користувача. Вони дозволяють оцінювати не лише результативність гравця, а й характер його дій. До таких показників належать час бездіяльності, повторюваність поведінки, ризиковані дії, спроби евакуації та інші ознаки, які можуть вказувати на особливості стилю проходження або можливе навмисне заниження результатів.

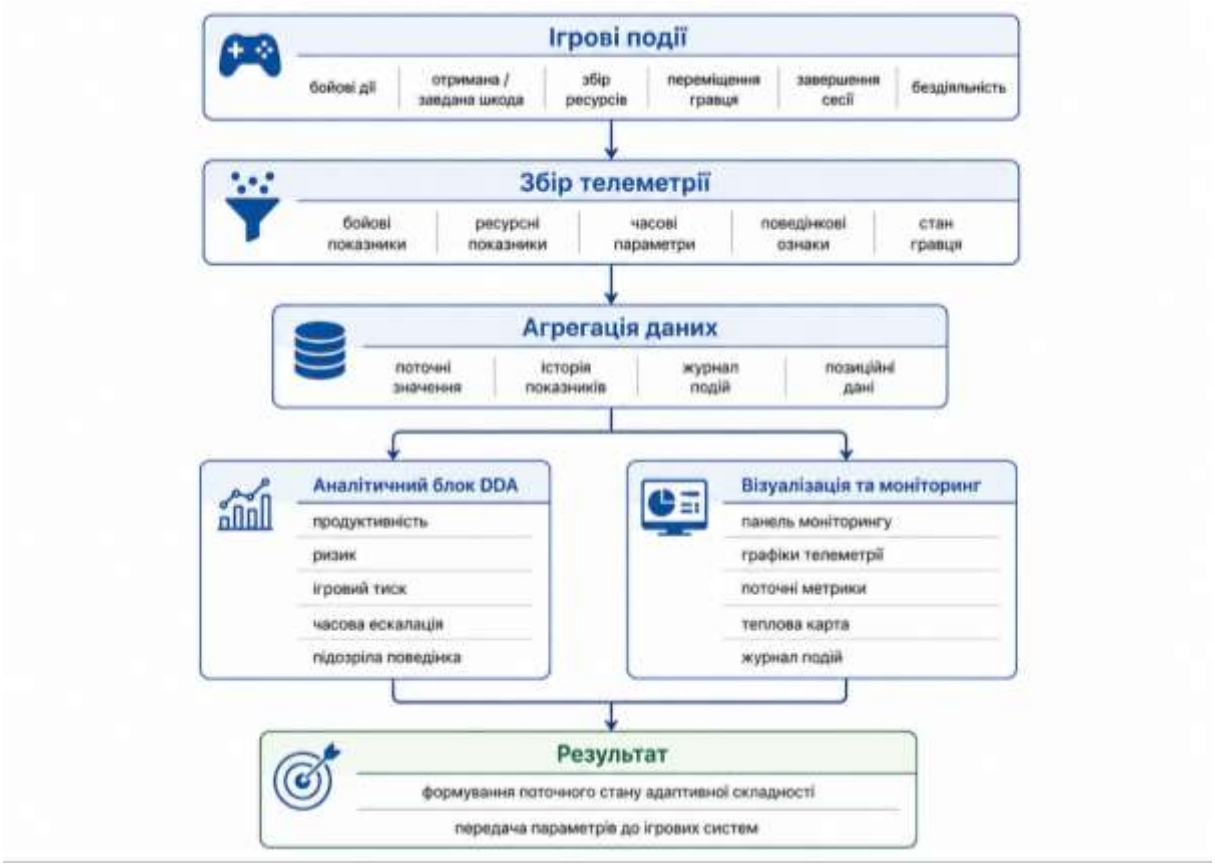


Рисунок 3.1 – Збір і візуалізація телеметричних даних у модулі адаптивної складності

На рис. 3.1 подано загальну схему збору та використання телеметричних даних у модулі адаптивної складності. Ігрові події формують окремі групи

показників, які проходять етап збору й агрегації. Після цього дані паралельно використовуються аналітичним блоком DDA для розрахунку продуктивності, ризику, ігрового тиску, часової ескалації та підозрілої поведінки, а також передаються до системи моніторингу для візуальної перевірки роботи модуля.

Для перевірки коректності збору та обробки даних реалізовано панель моніторингу адаптивної системи. Вона об’єднує перелік активних підсистем, поточні метрики складності, графіки телеметрії, стан системи керування сесією, теплову карту активності користувача та журнал подій. Такий підхід дозволяє спостерігати за роботою модуля в режимі реального часу та перевіряти, як зміна поведінки користувача впливає на інтегральні показники адаптації.

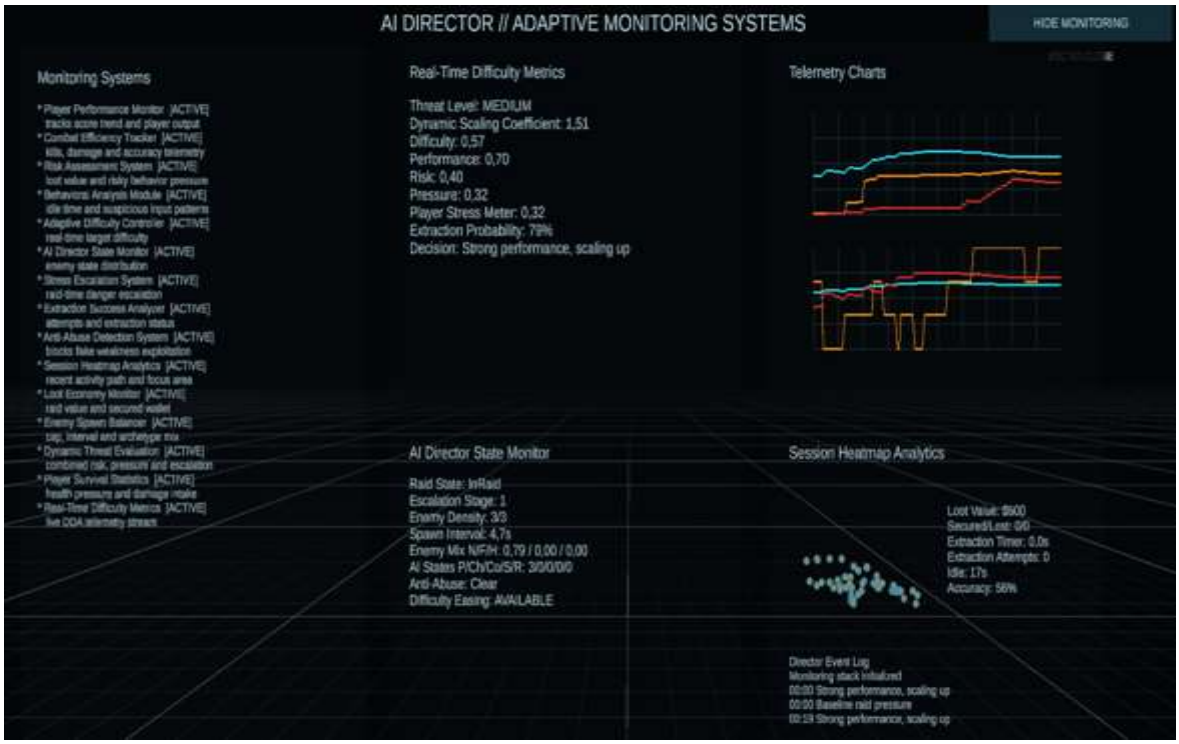


Рисунок 3.2 – Панель моніторингу системи адаптивного налаштування складності

На рис. 3.2 наведено загальний вигляд панелі моніторингу, що відображає активні підсистеми аналізу, поточні значення складності, графіки телеметрії, стан керування сесією, теплову карту активності та журнал подій. Панель використовується для тестування роботи модуля та перевірки коректності передачі даних між окремими підсистемами.

Важливою частиною реалізації є графічне відображення телеметрії. Воно дозволяє аналізувати не лише поточні значення показників, а й їх зміну в часі. Це необхідно для перевірки плавності адаптації, виявлення різких стрибків складності та аналізу реакції системи на зміну поведінки гравця.

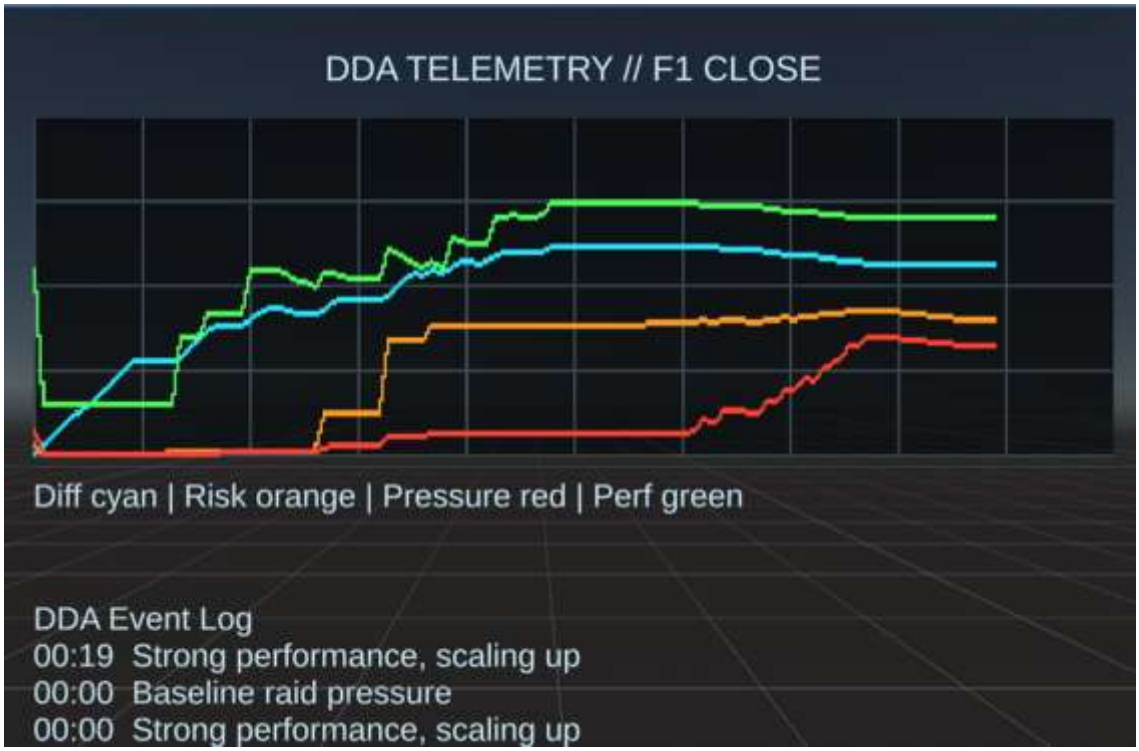


Рисунок 3.3 – Графічне відображення телеметрії DDA в реальному часі

На рис. 3.3 показано графічне представлення зміни основних інтегральних показників у часі. Окремими лініями відображаються рівень складності, ризик, ігровий тиск і продуктивність гравця. Така візуалізація дозволяє оцінити стабільність роботи адаптивної системи та перевірити, чи не відбувається надмірно різких змін параметрів.

Окрім графіків, під час тестування важливо мати доступ до поточних числових значень адаптивної системи. Саме ці значення дозволяють швидко оцінити, у якому стані перебуває модель, яке рішення вона ухвалила та які показники найбільше вплинули на зміну складності.



Рисунок 3.4 – Відображення поточних метрик адаптивної складності

На рис. 3.4 наведено приклад відображення поточних значень адаптивної системи. Серед них подано рівень загрози, коефіцієнт динамічного масштабування, поточну складність, продуктивність гравця, рівень ризику, ігровий тиск, показник стресу, ймовірність успішного завершення сесії та поточне рішення системи. Такий блок дозволяє відстежувати логіку роботи адаптивного модуля безпосередньо під час виконання ігрової сесії.

Для аналізу поведінкової активності користувача реалізовано просторову візуалізацію у форматі теплової карти. Такий інструмент дозволяє визначити, у яких зонах гравець перебуває найчастіше, як змінюється характер його переміщення та чи спостерігаються ознаки пасивної або нетипової поведінки.

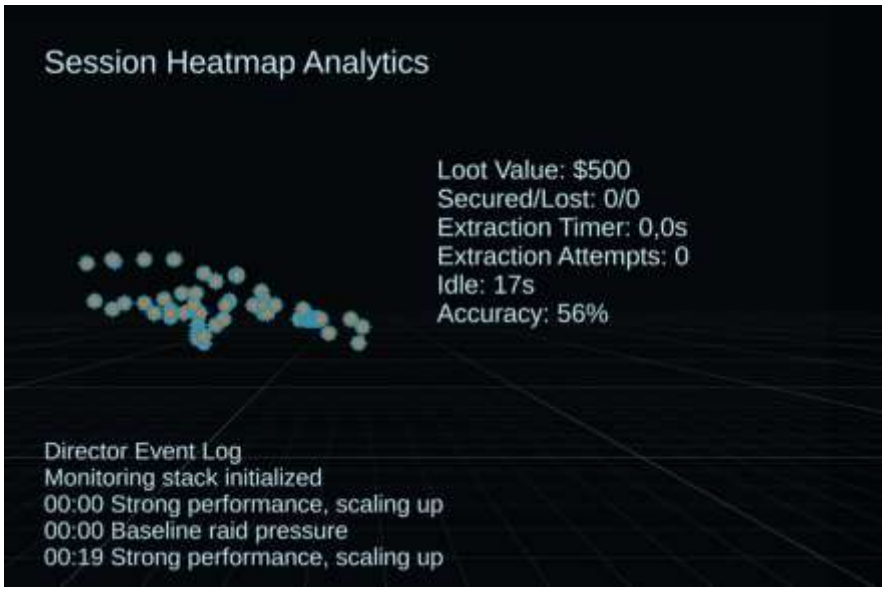


Рисунок 3.5 – Візуалізація поведінкової активності користувача у форматі теплової карти

На рис. 3.5 показано теплову карту активності користувача, яка відображає просторовий розподіл переміщень під час сесії. Додатково виводяться цінність накопичених ресурсів, кількість спроб евакуації, час бездіяльності, точність дій і журнал подій системи адаптації. Отримані дані можуть використовуватися для аналізу поведінкової активності, оцінки ризику та перевірки коректності роботи модуля адаптивного налаштування складності.

Окрему роль у підсистемі збору даних відіграє журнал подій. Він фіксує ключові рішення адаптивної системи, зокрема підвищення або зниження складності, базовий тиск сесії, реакцію на високу продуктивність, а також інші службові повідомлення. Наявність такого журналу підвищує інтерпретованість роботи системи, оскільки дозволяє бачити не лише числові значення, а й причини, через які було прийнято певне рішення.

Таким чином, у межах реалізації збору даних було створено підсистему, яка фіксує бойові, ресурсні, часові та поведінкові показники користувача. Зібрані дані використовуються для формування поточного стану адаптивної складності, розрахунку інтегральних показників, виявлення підозрілої поведінки та передачі результатів до ігрових підсистем. Панель моніторингу, графіки телеметрії, поточні метрики та теплова карта активності забезпечують можливість перевірки роботи системи в режимі реального часу.

3.2. Реалізація модуля адаптивного налаштування складності.

Після реалізації збору телеметричних даних наступним етапом є безпосередня реалізація модуля адаптивного налаштування складності. Його призначення полягає в обробці зібраних показників, розрахунку цільового рівня складності, перевірці умов зловживання системою та передачі результатів до ігрових підсистем.

Модуль реалізовано як окрему систему, що працює в режимі реального часу та виконує повний цикл адаптації під час ігрової сесії. У межах одного циклу оновлення система отримує актуальні телеметричні дані, виконує аналіз

продуктивності гравця, оцінює рівень ризику та ігрового тиску, враховує часову ескалацію, перевіряє ознаки підозрілої поведінки та формує поточний стан адаптивної складності.

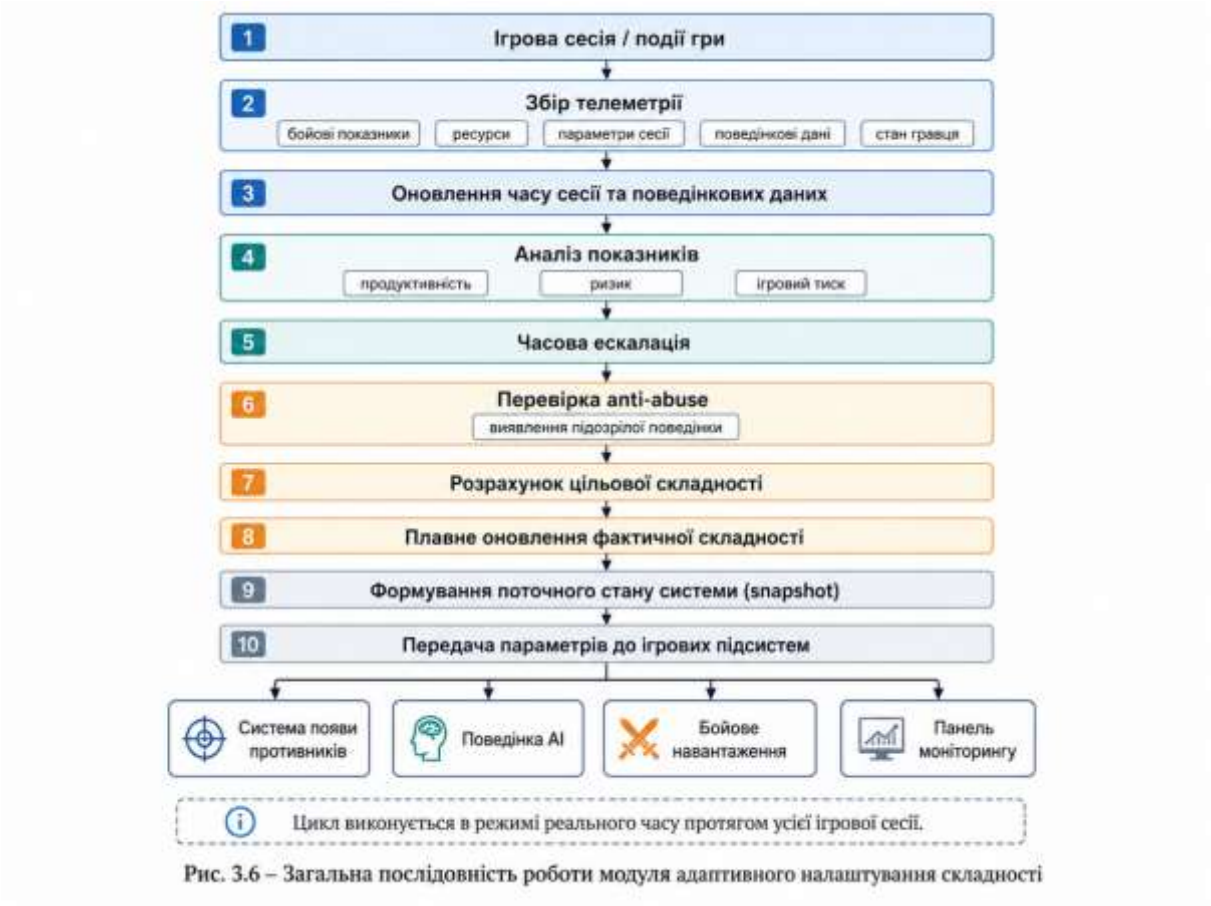


Рис. 3.6 – Загальна послідовність роботи модуля адаптивного налаштування складності

Рисунок 3.6 – Загальна послідовність роботи модуля адаптивного налаштування складності

Центральним елементом реалізації є цикл оновлення, який виконується під час роботи ігрової сесії. У цьому циклі система послідовно виконує такі операції: збільшує тривалість сесії, оновлює поведінкову телеметрію, розраховує інтегральні показники, перевіряє умови зловживання, визначає цільову складність, оновлює параметри появи противників і формує поточний стан системи.

Приклад фрагмента реалізації основного циклу оновлення наведено нижче:

```
private void Update()
{
   EnsureMonitoringUI();
   raidTime += Time.deltaTime;
```

```
    GameEvents.RaiseRaidTick(raidTime);

    behaviorCollector.Tick(Time.deltaTime);
    escalationManager.Tick(raidTime);
    performanceAnalyzer.Analyze(combatCollector, lootCollector, sessionCollector,
healthCollector, behaviorCollector);
    antiAbuseManager.Analyze(combatCollector, sessionCollector,
behaviorCollector);

    targetDifficulty = CalculateDifficultyTarget();
    spawnDirector.Tick(targetDifficulty, escalationManager.Stage,
performanceAnalyzer.RiskScore, performanceAnalyzer.PressureScore, raidTime);
    BuildSnapshot(targetDifficulty);
```

Наведений фрагмент демонструє, що модуль працює не як одноразовий розрахунок, а як постійно активна система. Кожне оновлення ігрової сесії супроводжується повторним аналізом стану гравця та коригуванням службових параметрів адаптації.

Розрахунок цільової складності базується на поєднанні кількох інтегральних показників. До них належать продуктивність гравця, рівень ризику, часовий тиск та поточне навантаження на гравця. Такий підхід дозволяє уникнути ситуації, коли складність змінюється лише за одним параметром, наприклад за кількістю перемог або поразок.

Фрагмент розрахунку цільового рівня складності має такий вигляд:

```
float target =
    0.22f +
    performanceAnalyzer.PerformanceScore * 0.38f +
    performanceAnalyzer.RiskScore * 0.3f +
    escalationManager.TimePressure * 0.14f -
    performanceAnalyzer.PressureScore * 0.18f;

return Mathf.Clamp01(target);
```

У цьому розрахунку продуктивність, ризик і часовий тиск підвищують цільову складність, тоді як надмірний тиск на гравця зменшує її. Це дозволяє зробити адаптацію більш збалансованою: система може підвищувати складність для результативного гравця, але не повинна робити це надто різко, якщо він уже перебуває під значним навантаженням.

```
private float CalculateDifficultyTarget()
{
    float target =
        0.22f +
        performanceAnalyzer.PerformanceScore * 0.38f +
        performanceAnalyzer.RiskScore * 0.3f +
        escalationManager.TimePressure * 0.14f -
        performanceAnalyzer.PressureScore * 0.18f;

    if (antiAbuseManager.BlocksDifficultyReduction)
    {
        target = Mathf.Max(target, Snapshot.Difficulty);
    }

    decisionReason = BuildDecisionReason(target);
    return Mathf.Clamp01(target);
}
```

Рисунок 3.7 – Приклад реалізації розрахунку цільового рівня складності

Після розрахунку цільового значення фактична складність не змінюється миттєво. Для цього використовується поступовий перехід до нового значення. Такий підхід потрібний для уникнення різких стрибків складності, які могли б порушити стабільність ігрового процесу.

Плавне оновлення складності реалізовано таким чином:

```
Difficulty = Mathf.MoveTowards(
   Snapshot.Difficulty,
   targetDifficulty,
   Time.deltaTime * 0.05f
);
```

Завдяки цьому фактичне значення складності поступово наближається до цільового. Якщо цільова складність різко змінюється через коливання показників гравця, система не застосовує це значення миттєво, а згладжує перехід у часі.

Для централізованої передачі поточного стану системи формується спеціальна структура даних. Вона містить не лише поточний рівень складності, а й ключові інтегральні показники, службові параметри та результати роботи окремих механізмів адаптації. Такий підхід дозволяє іншим підсистемам

отримувати узгоджений стан DDA без прямої залежності від усіх внутрішніх компонентів.

Формування поточного стану системи можна подати таким фрагментом:

```
Snapshot = new DifficultySnapshot
{
   Difficulty = Mathf.MoveTowards(
      Snapshot.Difficulty,
      targetDifficulty,
      Time.deltaTime * 0.05f
   ),

   PerformanceScore = performanceAnalyzer.PerformanceScore,
   PressureScore = performanceAnalyzer.PressureScore,
   RiskScore = performanceAnalyzer.RiskScore,
   EscalationStage = escalationManager.Stage,

   SpawnMultiplier = spawnDirector.SpawnMultiplier,
   ActiveEnemyCap = spawnDirector.ActiveEnemyCap,
   SpawnInterval = spawnDirector.SpawnInterval,
   AggressionMultiplier = spawnDirector.AggressionMultiplier,

   AntiAbuseState = antiAbuseManager.State,
   DifficultyReductionBlocked = antiAbuseManager.BlocksDifficultyReduction
};
```

Цей фрагмент демонструє, що поточний стан адаптивної системи містить як аналітичні показники, так і практичні параметри впливу на ігрове середовище. До нього входять складність, продуктивність, ризик, тиск, етап ескалації, параметри появи противників, агресивність і стан захисту від зловживань.

Для централізованої передачі поточного стану системи формується спеціальна структура даних. Вона містить основні інтегральні показники, поточний рівень складності, етап ескалації та стан механізму захисту від зловживань. Повна структура також може містити додаткові параметри, що передаються до ігрових підсистем, зокрема параметри появи противників, інтервалів та агресивності.

```
Snapshot = new DifficultySnapshot
{
    Difficulty = Mathf.MoveTowards(Snapshot.Difficulty, targetDifficulty, Time.deltaTime * 0.05f),
    PerformanceScore = performanceAnalyzer.PerformanceScore,
    PressureScore = performanceAnalyzer.PressureScore,
    RiskScore = performanceAnalyzer.RiskScore,
    EscalationStage = escalationManager.Stage,
    SpawnMultiplier = spawnDirector.SpawnMultiplier,
    ActiveEnemyCap = spawnDirector.ActiveEnemyCap,
    SpawnInterval = spawnDirector.SpawnInterval,
```

Рисунок 3.8 – Фрагмент формування поточного стану адаптивної системи

На рис. 3.8 наведено скорочений фрагмент формування поточного стану адаптивної системи, що містить ключові значення для подальшої роботи модуля: складність, продуктивність, ризик, ігровий тиск, етап ескалації та стан блокування зниження складності. Окремо в модулі реалізовано механізм захисту від зловживань, який виявляє підозрілу поведінку гравця та блокує автоматичне зниження складності.

Фрагмент реалізації блокування зниження складності наведено на рисунку 3.9.

```
if (antiAbuseManager.BlocksDifficultyReduction)
{
    target = Mathf.Max(target, Snapshot.Difficulty);
}
```

Рисунок 3.9 – Блокування зниження складності механізмом захисту від зловживань

У разі виявлення підозрілої поведінки система не дозволяє автоматично знизити рівень складності. Для цього цільове значення складності порівнюється з поточним, після чого обирається більше з двох значень. Така логіка дозволяє уникнути ситуацій, коли гравець свідомо створює умови для полегшення проходження, наприклад через бездіяльність або навмисне погіршення показників.

У разі активації механізму захисту система не дозволяє знизити цільову складність нижче поточного рівня. Це запобігає штучному спрощенню проходження через бездіяльність або навмисне погіршення бойових показників.

Після формування цільової складності модуль передає результати до систем, які безпосередньо впливають на ігровий процес. Одним із таких напрямів є керування появою противників. На основі складності, ризику та тиску формується тиск появи противників, після чого визначаються кількість активних противників, інтервал їх появи та загальна агресивність.

Приклад передачі результатів адаптації до системи появи противників:

```
float spawnPressure = Mathf.Clamp01(
   targetDifficulty * 0.65f +
   riskScore * 0.45f -
   pressureScore * 0.25f
);

ActiveEnemyCap = Mathf.RoundToInt(
   Mathf.MoveTowards(
      ActiveEnemyCap,
      targetCap,
      Time.deltaTime * 0.35f
   )
);

SpawnInterval = Mathf.MoveTowards(
   SpawnInterval,
   targetInterval,
   Time.deltaTime * 0.18f
);

AggressionMultiplier = Mathf.Lerp(1f, 1.6f, spawnPressure);
```

Наведений фрагмент показує, що значення складності, ризику та ігрового тиску використовуються для формування тиску появи противників. На його основі змінюються кількість активних противників, інтервал їх появи та рівень агресивності.

Крім системи появи противників, результати адаптації використовуються для зміни поведінки штучного інтелекту. Для цього формується ефективна складність, яка враховує поточну складність, ризик, етап ескалації та коефіцієнт пом’якшення тиску.

Приклад передачі результатів у систему штучного інтелекту:

```
float effectiveDifficulty = Mathf.Clamp01(
    snapshot.Difficulty * 0.55f +
    snapshot.RiskScore * 0.35f +
    stagePressure -
    pressureRelief
);

moveSpeed = baseMoveSpeed *
    Mathf.Lerp(0.92f, 1.16f, effectiveDifficulty) *
    aggression;

detectionRange = Mathf.Lerp(15f, 29f, effectiveDifficulty);
combatPressure = Mathf.Lerp(0.55f, 0.95f, effectiveDifficulty);
```

У результаті адаптивна система впливає на швидкість пересування противників, дальність виявлення гравця, бойовий тиск і загальну інтенсивність взаємодії. Це дозволяє змінювати складність не лише кількісно, через кількість противників, а й якісно – через їхню поведінку.

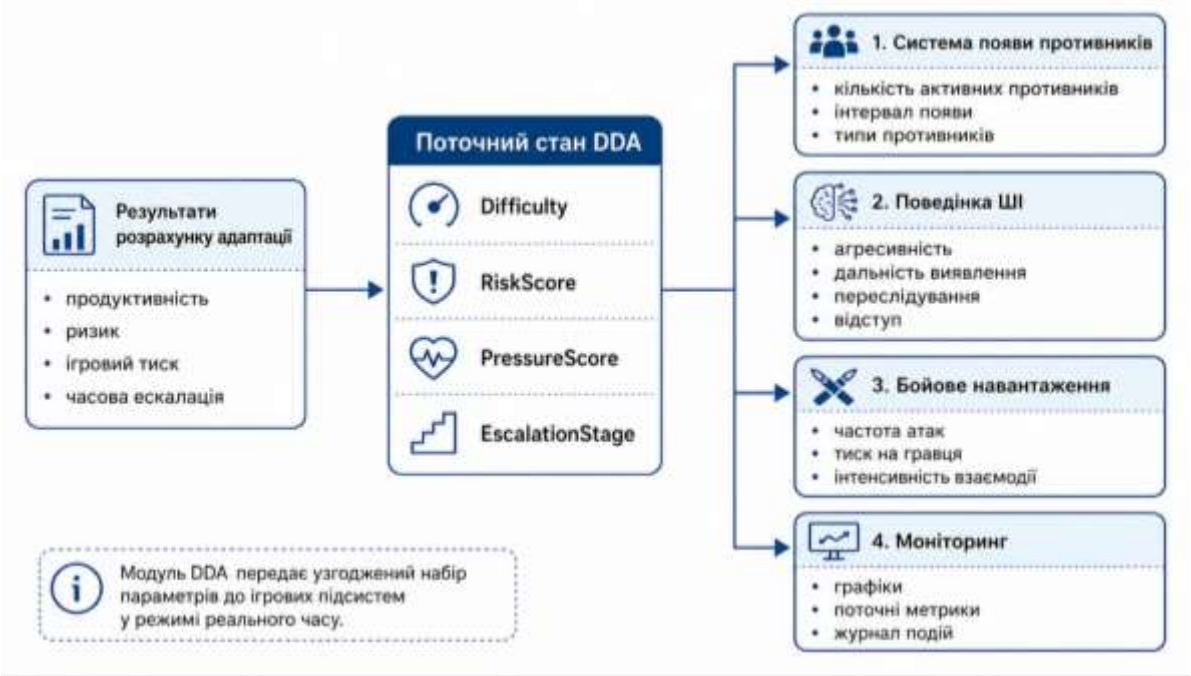


Рисунок 3.10 – Передача параметрів адаптації до ігрових підсистем

Таким чином, реалізований модуль поєднує збір телеметрії, аналіз стану гравця, розрахунок цільової складності, плавне оновлення фактичного значення, перевірку зловживань і передачу результатів до ігрових систем. Це дозволяє модулю працювати в режимі реального часу та впливати на ключові параметри ігрового процесу без різких чи необґрунтованих змін складності.

3.3. Тестування та аналіз результатів роботи модуля адаптивного налаштування складності

Метою тестування є перевірка працездатності запропонованого модуля в умовах реальної ігрової сесії та порівняння його роботи з фіксованими рівнями складності. Основну увагу приділено коректності оновлення телеметричних показників, реакції системи на продуктивність гравця, зміні рівня ризику, впливу ігрового тиску, роботі механізму захисту від зловживань та передачі результатів до ігрових підсистем.

Для формування контрольної бази було використано три статичні режими складності: легкий, середній і важкий. У цих режимах основні параметри ігрового середовища залишаються незмінними протягом сесії та не залежать від продуктивності, ризику, тиску або поведінки гравця. Адаптивний режим, навпаки, змінює параметри в реальному часі.

Таблиця 3.1 – Параметри контрольних статичних режимів

Параметри	**Легкий**	**Середній**	**Важкий**
Рівень складності	0,25	0,50	0,75
Максимальна кількість противників	3	5	7
Інтервал появи, сек	6,5	4,8	3,3
Множник агресивності	0,95	1,10	1,30
Частка звичайних противників	1,00	0,85	0,65
Частка швидких противників	0,05	0,25	0,45
Частка важких противників	0,00	0,05	0,18

Тестування проводилося в єдиній демонстраційній сцені з увімкненою панеллю моніторингу.

Сцена містила керованого персонажа, противників різних типів, ресурси для збирання та зону завершення сесії. Для всіх режимів використовувалися однакові початкові умови, ігрове середовище та правила завершення рейду. Загальний вигляд демонстраційної сцени та основні елементи ігрового процесу наведено на рис. А.1–А.4. Приклад ігрового процесу під час роботи адаптивного режиму наведено на рис. А.6.

Під час кожного тестового сценарію фіксувалися поточні значення складності, продуктивності, ризику, ігрового тиску, стану механізму захисту від зловживань, інтервалу появи противників, коефіцієнта динамічного масштабування та текстового пояснення рішення системи.

Для забезпечення однакових умов порівняння спочатку було визначено параметри контрольних статичних режимів складності. Вони задавали фіксований рівень складності, максимальну кількість активних противників, інтервал їх появи, множник агресивності та співвідношення типів противників.

Окремо було сформовано набір тестових сценаріїв для перевірки реакції адаптивної системи на різні умови ігрової сесії. До таких умов належали базовий стан системи, висока продуктивність гравця, підвищений рівень ризику, значний ігровий тиск і підозріла поведінка користувача. Завдяки цьому тестування охоплювало не лише порівняння зі статичними режимами, а й перевірку окремих механізмів модуля: оновлення показників продуктивності, ризику та тиску, блокування штучного заниження складності, а також передачу сформованих параметрів до ігрових підсистем. Перелік використаних сценаріїв наведено в табл. 3.2.

Таблиця 3.2 – Сценарії тестування модуля адаптивного налаштування складності

№	Сценарій	Мета тестування	Дії під час тесту	Очікувана реакція системи
1	Базова сесія	Перевірити запуск DDA-модуля та стабільність початкових показників	Запустити сцену, увімкнути моніторинг, не створювати екстремальних умов	Метрики оновлюються, складність залишається близькою до базового рівня
2	Висока продуктивність	Перевірити реакцію системи на результативні дії гравця	Активно знищувати противників, підтримувати високу точність, мінімізувати отриману шкоду	Performance зростає, Difficulty поступово підвищується
3	Високий ризик	Перевірити вплив цінності накопичених ресурсів на складність	Зібрати значну кількість ресурсів і завершити сесію з високою сумою збереженого луту	Risk зростає, система підвищує або підтримує небезпеку
4	Високий ігровий тиск	Перевірити реакцію системи на підвищене навантаження гравця	Перебувати під атакою, отримувати шкоду, не накопичувати значний лут	Pressure зростає, складність не повинна різко підвищуватися
5	Підозріла поведінка	Перевірити роботу механізму захисту від зловживань	Після активного забігу запустити нову сесію та тривалий час залишатися без активних дій	Anti-abuse активується, зниження складності блокується

У табл. 3.2 наведено основні сценарії, які були використані для перевірки роботи DDA-модуля. Вони охоплюють як стандартну поведінку системи, так і ситуації з підвищеною продуктивністю, високим ризиком, значним ігровим тиском та підозрілою поведінкою користувача.

Таблиця 3.3 – Контрольовані показники під час тестування

Показник	**Призначення**	**Очікувана поведінка під час тестування**
Difficulty	Поточний рівень складності	Змінюється поступово, без різких стрибків
Performance	Оцінка продуктивності гравця	Зростає при високій точності та активних бойових діях
Risk	Оцінка ризику	Зростає при накопиченні ресурсів або підвищенні цінності прогресу
Pressure	Оцінка ігрового тиску	Зростає при отриманні шкоди та високому навантаженні
Dynamic Scaling Coefficient	Узагальнений коефіцієнт масштабування	Збільшується при зростанні складності або ризику
Decision	Пояснення рішення системи	Має відповідати поточному сценарію тестування
Anti-Abuse	Стан захисту від зловживань	Активується при підозрілій або пасивній поведінці
Spawn Interval	Інтервал появи противників	Зменшується при підвищенні тиску середовища
Spawn Cap	Ліміт активних противників	Може збільшуватися при підвищенні складності

Контрольовані показники подано у табл. 3.3. Їх було обрано таким чином, щоб оцінити не лише числову зміну складності, а й причини цієї зміни, зокрема продуктивність, ризик, тиск, стан захисту від зловживань та параметри, які передаються до ігрових підсистем.

Таблиця 3.4 – Фактичні результати тестування DDA-модуля

№	Сценарій	Зафіксовані значення	Реакція системи	Результат
1	Базова сесія	36 с; Difficulty 0,29; Performance 0,16; Risk 0,02; Pressure 0,01; Spawn Interval 5,7 с	Decision: Baseline raid pressure. Система працює у базовому режимі, anti-abuse не активований	Успішно
2	Висока продуктивність	1 хв 39 с; Difficulty 0,50; Performance 0,63; Risk 0,06; Pressure 0,07; Accuracy 94%; Spawn Interval 5,2 с	Decision: Strong performance, scaling up. Складність підвищується через високу продуктивність	Успішно
3	Високий ризик	1 хв 29 с; Secured Loot $2550; Difficulty 0,64; Performance 0,64; Risk 0,60; Pressure 0,16; Spawn Interval 4,0 с	Decision: High loot/risk raises danger. Система підвищила небезпеку через значну цінність ресурсів	Успішно
4	Високий ігровий тиск	Близько 2 хв; Difficulty 0,35; Performance 0,49; Risk 0,05; Pressure 0,56; Spawn Interval 6,1 с	У журналі зафіксовано Player under pressure, easing. Складність не зросла різко при високому тиску	Успішно
5	Підозріла поведінка	Другий рейд ~1 хв 23 с; Idle 58 с; Difficulty 0,29; Performance 0,13; Risk 0,04; Pressure 0,03	Anti-Abuse: AFK/FakeWeakness Block; Decision: Anti-abuse blocks easing; Difficulty Easing: BLOCKED	Успішно

Фактичні результати тестування наведено у табл. 3.4. В усіх сценаріях модуль демонстрував очікувану поведінку: у базовій сесії складність залишалася стабільною, при високій продуктивності та значному накопиченні ресурсів підвищувалася, а при підозрілій поведінці автоматичне зниження складності блокувалося.

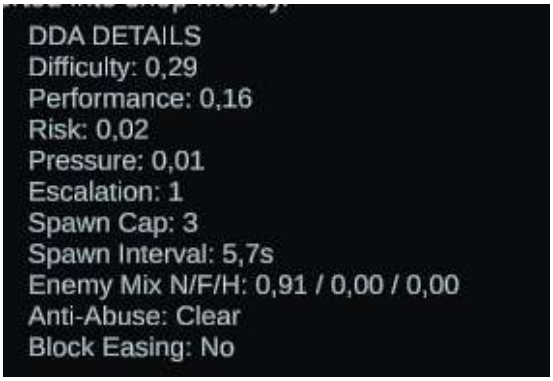


Рисунок 3.11 – Стан DDA-модуля під час базової ігрової сесії

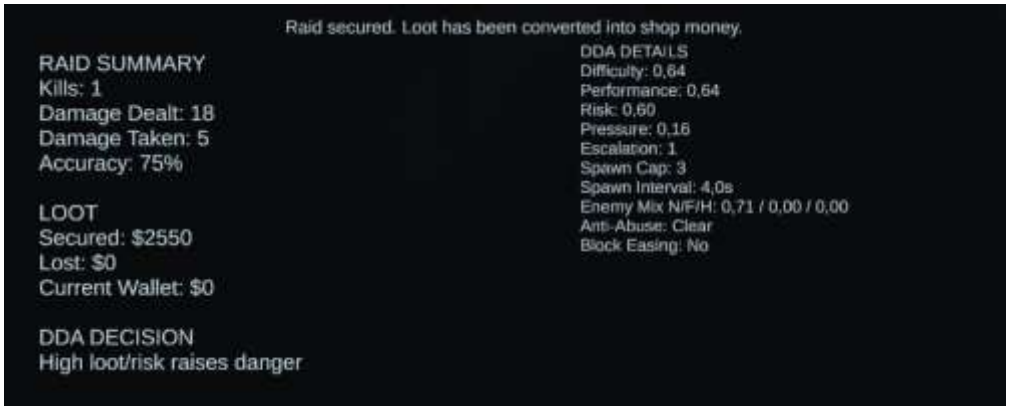


Рисунок 3.12 – Реакція системи на високий ризик через накопичення ресурсів



Рисунок 3.13 – Спрацювання механізму захисту від зловживань під час AFK-поведінки

Таблиця 3.5 – Зміна параметрів ігрових підсистем залежно від стану DDA

Сценарій	Difficulty	Dynamic Scaling	Spawn Cap	Spawn Interval	Threat Level	Висновок
Базова сесія	0,29	1,26	3	5,7 с	LOW	Система працює у базовому режимі
Висока продуктивність	0,50	1,45	3	5,2 с	MEDIUM	Складність і масштабування зросли
Високий ризик	0,64	1,58	3	4,0 с	MEDIUM	Небезпека підвищена через ресурсний ризик
Високий тиск	0,35	1,31	3	6,1 с	LOW	Система не підвищує навантаження різко
Anti-abuse	0,29	1,26	3	5,7 с	LOW	Зниження складності заблоковано

Дані табл. 3.5 показують, що DDA-модуль впливає не лише на числовий рівень складності, а й на параметри ігрових підсистем: інтервал появи противників, коефіцієнт динамічного масштабування та загальне навантаження. У сценарії високого тиску система не підвищує складність різко, що підтверджує роботу механізму пом’якшення.

Після перевірки коректності основних механізмів адаптивного модуля було проведено порівняльне тестування. Його метою стало зіставлення роботи запропонованого адаптивного режиму з трьома контрольними статичними режимами складності: легким, середнім і важким. На відміну від адаптивного режиму, параметри контрольних режимів залишалися незмінними протягом усієї ігрової сесії та не залежали від продуктивності, ризику, ігрового тиску або поведінки гравця.

Порівняльне тестування проводилося в однаковій демонстраційній сцені з використанням тих самих правил керування, початкового стану персонажа, розташування ресурсів, складу доступних противників та умов завершення сесії. Такий підхід дав змогу зменшити вплив сторонніх чинників і зосередити порівняння на відмінностях між фіксованим та адаптивним способами формування складності.

Для кожного режиму складності було передбачено проведення п’яти ігрових сесій. Загалом порівняльне тестування охоплювало двадцять рейдів: по п’ять для легкого, середнього, важкого та адаптивного режимів. Перед початком кожної сесії відновлювалися однакові початкові умови: запас здоров’я, спорядження персонажа, розміщення ресурсів, склад доступних противників і стан ігрової сцени.

Під час проходження використовувалася однакова загальна стратегія: збирання доступних ресурсів, участь у бойових зіткненнях і спроба успішного завершення рейду. Тривалість сесії не обмежувалася штучно, оскільки час проходження належав до показників, за якими оцінювалася робота кожного режиму.

Результати завершених сесій автоматично записувалися до файлу у форматі CSV. Для порівняння фіксувалися результат рейду, його тривалість, кількість знищених і створених противників, точність, обсяг завданої та отриманої шкоди, цінність збережених ресурсів, середня кількість активних противників, середній інтервал їх появи та середній множник агресивності. Для адаптивного режиму додатково враховувалися середній і максимальний рівні складності, показники продуктивності, ризику, ігрового тиску та кількість спрацювань механізму захисту від зловживань.

Таблиця 3.6 – Критерії порівняльного тестування режимів складності

Показник	**Призначення**
Результат сесії	Визначення кількості успішно завершених і невдалих рейдів
Тривалість сесії	Оцінювання часу, необхідного для проходження рейду
Кількість знищених противників	Характеристика бойової результативності гравця
Загальна кількість створених противників	Оцінювання фактичного бойового навантаження протягом сесії
Кількість звичайних, швидких і важких противників	Визначення фактичного складу створених противників
Точність дій гравця	Оцінювання ефективності використання зброї
Завдана шкода	Характеристика результативності бойових дій
Отримана шкода	Оцінювання рівня небезпеки та навантаження на гравця
Цінність збережених ресурсів	Визначення результативності проходження з урахуванням механіки ризику й винагороди
Середня кількість активних противників	Оцінювання інтенсивності одночасного бойового навантаження
Середній інтервал появи противників	Характеристика частоти виникнення нових загроз
Середній множник агресивності	Оцінювання інтенсивності поведінки противників
Середній рівень складності	Визначення фактичного рівня навантаження в адаптивному режимі
Максимальний рівень складності	Фіксація найбільшого рівня складності, досягнутого протягом адаптивної сесії
Кількість блокувань зниження складності	Перевірка роботи механізму захисту від зловживань

Наведені показники дали змогу порівнювати режими не лише за встановленим рівнем складності, а й за фактичним впливом на перебіг ігрової сесії. Для статичних режимів основні параметри залишалися незмінними, тоді як в адаптивному режимі вони могли змінюватися відповідно до поточного стану гравця. Середній і максимальний рівні складності, а також кількість блокувань її зниження використовувалися лише під час аналізу адаптивного режиму. Для забезпечення однакових умов було визначено стандартний

сценарій порівняльного рейду. На початку кожної сесії персонаж мав повний запас здоров’я, однакове спорядження та однакову кількість боєприпасів. Розташування ресурсів, противників, точки появи персонажа та зони завершення рейду залишалося незмінним.

Під час проходження гравець рухався за однаковим маршрутом через основні ділянки демонстраційної сцени. У межах рейду необхідно було зібрати доступні на маршруті ресурси, брати участь у бойових зіткненнях із противниками, які перешкоджали проходженню, та досягти зони завершення сесії. Навмисне очікування, уникнення всіх бойових взаємодій, штучне отримання шкоди або навмисне погіршення показників не допускалися.

Сесію вважали успішною після досягнення зони завершення рейду зі збереженням зібраних ресурсів. У разі втрати всього запасу здоров’я фіксувалося невдале завершення. Тривалість проходження не обмежувалася, оскільки час сесії використовувався як один із показників порівняння.

Після завершення рейду результати автоматично записувалися до файлу CSV. Перед початком наступної сесії стан сцени, здоров’я персонажа, спорядження, ресурси та лічильники телеметрії повністю скидалися. Вибраний режим складності змінювався лише перед початком нової серії тестів.

Для всіх порівняльних рейдів використовувався однаковий маршрут через основні ділянки демонстраційної сцени, зони розміщення ресурсів і бойових зіткнень. Під час проходження гравець збирав ресурси вздовж маршруту, знищував противників, які перешкоджали руху, та завершував сесію в точці евакуації. Відхилення від маршруту й навмисне очікування додаткових противників не допускалися. Маршрут проходження демонстраційної сцени під час порівняльного тестування наведено на рис. А.5.

Під час кожного порівняльного рейду використовувалися однакові початкові умови: повний запас здоров’я, незмінне спорядження та однакова кількість боєприпасів. На визначеному маршруті розміщувалося шість точок здобичі. Гравець збирав доступні ресурси та знищував противників, які атакували персонажа або перешкоджали проходженню, не створюючи

додаткових бойових зіткнень навмисно. Зазвичай протягом одного рейду знищувалося від одного до двох противників.

Сесію завершували після досягнення точки евакуації або втрати всього запасу здоров’я. Перед кожним новим рейдом стан сцени, персонажа та телеметричні показники скидалися до початкових значень. Режими тестувалися почергово, що дало змогу зменшити вплив поступового звикання до маршруту.

У табл. 3.7 наведено узагальнені результати двадцяти порівняльних рейдів – по п’ять сесій для легкого, середнього, важкого та адаптивного режимів. Для кожного режиму розраховано середні значення тривалості проходження, бойової результативності, точності, отриманої шкоди, кількості створених і активних противників, інтервалу їх появи, множника агресивності та рівня складності.

Таблиця 3.7 – Порівняння результатів статичних та адаптивного режимів складності

Показник	**Легкий**	**Середній**	**Важкий**	**Адаптивний**
Кількість сесій	5	5	5	5
Успішне завершення, %	100	100	100	100
Середня тривалість сесії, с	19,02	20,49	24,04	22,84
Середня кількість знищених противників	1,2	2,0	3,6	2,8
Середня точність, %	63,73	69,54	51,31	61,60
Середня отримана шкода	0	0	21,0	0
Середня кількість створених противників	3,4	4,8	7,8	5,4
Середня кількість активних противників	1,47	1,81	2,34	1,57
Середній інтервал появи противників, с	6,50	4,80	3,30	4,45
Середній множник агресивності	0,95	1,10	1,30	1,32
Середня цінність збережених ресурсів	1400	1400	1400	1400
Середній рівень складності	0,25	0,50	0,75	0,45
Середній максимальний рівень складності	0,25	0,50	0,75	0,65

Дані табл. 3.7 демонструють послідовне збільшення бойового навантаження між статичними режимами. У легкому режимі середня кількість створених противників становила 3,4, у середньому – 4,8, а у важкому – 7,8. Одночасно інтервал їх появи скоротився з 6,50 до 3,30 с, а середня тривалість проходження збільшилася з 19,02 до 24,04 с. Лише у важкому режимі було зафіксовано отримання шкоди, середнє значення якої становило 21 одиницю.

Адаптивний режим за більшістю показників зайняв проміжне положення між середнім і важким рівнями. У ньому в середньому створювалося 5,4 противника, інтервал їх появи становив 4,45 с, а тривалість проходження – 22,84 с. Середній рівень складності дорівнював 0,45, проте в окремі моменти він підвищувався в середньому до 0,65. Це підтверджує, що модуль не підтримував одне фіксоване значення, а змінював навантаження відповідно до перебігу сесії.

Усі рейди були завершені успішно, а середня цінність збережених ресурсів у кожному режимі становила 1400 одиниць. Водночас адаптивний режим забезпечив більше бойове навантаження порівняно із середнім режимом, але не призвів до отримання гравцем шкоди. Отримані значення характеризують результати саме проведеної серії тестів і залежать від демонстраційної сцени, маршруту та дій одного тестувальника.

За результатами порівняльного тестування встановлено, що статичні режими забезпечують передбачуване, але незмінне навантаження, яке послідовно зростає від легкого до важкого рівня. Адаптивний режим формував змінний рівень складності та за фактичними показниками розташувався переважно між середнім і важким режимами. При цьому система підвищувала інтенсивність появи противників без різкого збільшення отриманої шкоди або погіршення результату проходження. Це підтверджує здатність запропонованого модуля регулювати навантаження в реальному часі, хоча остаточна оцінка його переваг потребує врахування обмеженого масштабу проведеного експерименту.

Для узагальнення результатів тестування доцільно розглянути окремо функціональні сценарії роботи модуля та результати порівняння адаптивного режиму з фіксованими рівнями складності.

Проведене тестування охоплювало два взаємопов’язані етапи. На першому етапі перевірялася коректність окремих механізмів адаптивного модуля в сценаріях базового стану, високої продуктивності, підвищеного ризику, значного ігрового тиску та підозрілої поведінки. На другому етапі роботу адаптивного режиму було порівняно з легким, середнім і важким статичними режимами за однакових умов проходження.

У базовому сценарії система підтримувала невисокий рівень складності та не виконувала необґрунтованого підвищення навантаження на початку сесії. Це свідчить про коректну роботу початкового стану модуля та відсутність різкої реакції за недостатньої кількості телеметричних даних.

У сценарії високої продуктивності зростання результативності гравця спричинило підвищення рівня складності та скорочення інтервалу появи противників. Отриманий результат підтверджує здатність системи реагувати на високу точність, завдану шкоду та успішні бойові дії, поступово збільшуючи інтенсивність ігрового процесу.

Під час перевірки сценарію високого ризику основним чинником зміни складності стала значна цінність накопичених ресурсів. Система підвищила рівень небезпеки навіть за невеликої кількості бойових дій. Така реакція відповідає механіці ризику й винагороди, оскільки можливі втрати гравця зростають разом із цінністю зібраної здобичі.

У сценарії підвищеного ігрового тиску система не продовжувала різко збільшувати навантаження, незважаючи на наявність активних противників і отриману шкоду. Інтервал появи противників збільшився, а рівень складності залишився нижчим, ніж у сценаріях високої продуктивності та ризику. Це підтверджує роботу механізму пом’якшення, призначеного для запобігання надмірному навантаженню на гравця.

Окремо було перевірено механізм захисту від зловживань. Після результативної сесії було розпочато новий рейд, у якому гравець тривалий час не виконував активних дій. Система розпізнала підозрілу поведінку та заблокувала автоматичне зниження складності. Таким чином, навмисне погіршення показників не призвело до штучного спрощення подальшого проходження.

Результати функціонального тестування підтвердили, що основні складові комбінованої моделі працюють відповідно до передбаченої логіки. Модуль реагує не на один окремий показник, а враховує продуктивність, ризик, ігровий тиск, тривалість сесії та поведінкові ознаки. Зміна складності при цьому передається до ігрових підсистем через кількість активних противників, інтервал їх появи, склад противників і параметри агресивності.

Для оцінювання поведінки модуля порівняно зі статичним підходом було проведено двадцять рейдів – по п’ять для легкого, середнього, важкого та адаптивного режимів. Усі сесії виконувалися за однаковим маршрутом, із незмінним спорядженням, шістьма точками здобичі та однаковими умовами завершення.

Статичні режими продемонстрували передбачуване зростання навантаження. У легкому режимі середня кількість створених противників становила 3,4, у середньому – 4,8, а у важкому – 7,8. Одночасно середній інтервал їх появи зменшувався від 6,50 с у легкому режимі до 3,30 с у важкому. Середня тривалість проходження також зросла від 19,02 до 24,04 с. Отримання гравцем шкоди було зафіксовано лише у важкому режимі, де середнє значення становило 21 одиницю.

Адаптивний режим за більшістю фактичних показників розташувався між середнім і важким статичними режимами. Середня кількість створених противників становила 5,4, середній інтервал їх появи – 4,45 с, а середня тривалість проходження – 22,84 с. Середній рівень складності дорівнював 0,45, проте максимальне значення протягом окремих сесій у середньому досягало

0,65. Це показує, що система не підтримувала постійне значення складності, а змінювала його відповідно до поточного перебігу рейду.

У всіх двадцяти сесіях було виконано успішну евакуацію та збережено по 1400 одиниць ресурсів. Водночас адаптивний режим створив більше бойове навантаження, ніж середній статичний режим, але не призвів до отримання шкоди гравцем. Отже, у межах проведених тестів модуль збільшував інтенсивність проходження без переходу до рівня навантаження, характерного для важкого режиму.

Перевагою статичних режимів є передбачуваність: їхні параметри відомі до початку проходження та залишаються незмінними. Це спрощує налаштування, тестування й розуміння умов гравцем. Водночас такий підхід не реагує на зміну результативності або поточного стану користувача. Важкий режим продовжує підтримувати високе навантаження незалежно від отриманої шкоди, тоді як легкий і середній режими не здатні підвищити інтенсивність за високої продуктивності.

Адаптивний режим має складнішу структуру та залежить від правильного вибору вагових коефіцієнтів, порогів і швидкості згладжування. Його поведінка є менш передбачуваною, а тестування потребує аналізу більшої кількості показників. Проте він здатний автоматично змінювати параметри ігрового процесу без ручного перемикання режиму й ураховувати ширший контекст сесії.

Проведене тестування має певні обмеження. Воно виконувалося в межах однієї демонстраційної сцени, за участю одного тестувальника та з використанням однакового маршруту. Кількість сесій була достатньою для первинного порівняння, але не охоплює всі можливі стилі гри й поведінкові ситуації. Крім того, результати залежать від конфігурації сцени, характеристик противників, розміщення ресурсів і тривалості маршруту.

Загалом результати функціонального та порівняльного тестування підтвердили працездатність розробленого модуля. Система коректно реагує на зміну продуктивності, ризику й ігрового тиску, виконує плавне регулювання

складності та протидіє навмисному погіршенню показників. У порівнянні зі статичними режимами адаптивна модель формує змінне навантаження, яке в проведених рейдах переважно відповідало проміжному рівню між середньою та важкою складністю. Це підтверджує можливість використання запропонованого модуля для регулювання складності в режимі реального часу.

ВИСНОВКИ

У межах кваліфікаційної роботи бакалавра досліджено проблему адаптивного налаштування складності в комп’ютерних іграх та розроблено програмний модуль для регулювання ігрового навантаження в режимі реального часу; лістинг основних компонентів модуля наведено в додатку В.

У межах першого завдання було проаналізовано сучасні підходи до налаштування складності в комп’ютерних іграх. Розглянуто статичну складність, системи на основі правил, адаптацію за продуктивністю гравця та комбіновані моделі адаптивного налаштування складності. Встановлено, що статичні режими є простими й передбачуваними, але не реагують на зміну стану гравця. Системи на основі правил здатні змінювати параметри за наперед визначеними умовами, проте мають обмежену гнучкість. Адаптація лише за продуктивністю точніше враховує результати гравця, однак недостатньо враховує ризик, контекст сесії та поведінкові особливості.

У межах другого завдання було розроблено систему показників для оцінювання стану гравця. До неї включено бойову ефективність, точність, обсяг завданої та отриманої шкоди, тривалість сесії, цінність накопичених ресурсів, рівень бездіяльності, ризиковану поведінку та інші телеметричні характеристики. На їх основі формуються інтегральні показники продуктивності, ризику та ігрового тиску, що дає змогу оцінювати стан гравця не за одним параметром, а з урахуванням сукупності чинників.

У межах третього завдання було спроєктовано комбіновану модель адаптивного налаштування складності. Для стабілізації реакції системи використано нормалізацію показників, експоненційне згладжування та поступову зміну цільового рівня складності. Часова ескалація забезпечує поступове підвищення інтенсивності зі збільшенням тривалості сесії. Окремо реалізовано механізм захисту від зловживань, який аналізує бездіяльність, ранні поразки, пасивне отримання шкоди та повторювані дії, спрямовані на штучне зниження складності.

Розроблений модуль реалізовано в середовищі Unity мовою C#. Його архітектура передбачає окремі компоненти збору телеметрії, аналізу стану гравця, часової ескалації, виявлення підозрілої поведінки, формування поточного стану складності та передачі результатів до ігрових підсистем. Розраховані параметри впливають на кількість активних противників, інтервал їх появи, склад противників, агресивність і поведінку штучного інтелекту.

У межах четвертого завдання було проведено функціональне та порівняльне тестування розробленого модуля. Функціональні сценарії підтвердили, що система коректно реагує на високу продуктивність, збільшення ризику, підвищений ігровий тиск і підозрілу поведінку. За високої продуктивності складність та інтенсивність появи противників зростали, за значного ризику підвищувалася небезпека втрати накопичених ресурсів, а за високого тиску система уникала різкого збільшення навантаження. У випадку тривалої бездіяльності механізм захисту від зловживань блокував автоматичне зниження складності.

Для порівняння адаптивного підходу з фіксованими рівнями складності було проведено двадцять рейдів: по п’ять для легкого, середнього, важкого та адаптивного режимів. Статичні режими продемонстрували послідовне збільшення навантаження від легкого до важкого рівня. Адаптивний режим за більшістю показників розташувався між середнім і важким режимами. Середня кількість створених противників у ньому становила 5,4, середній інтервал їх появи – 4,45 с, а середня тривалість проходження – 22,84 с. Середній рівень складності дорівнював 0,45, тоді як максимальне значення протягом сесій у середньому досягало 0,65.

Усі порівняльні рейди завершилися успішною евакуацією, а середня цінність збережених ресурсів у кожному режимі становила 1400 одиниць. Водночас адаптивний режим забезпечив більше бойове навантаження порівняно із середнім статичним режимом, але не призвів до отримання гравцем шкоди. Це свідчить про здатність модуля підвищувати інтенсивність ігрового

процесу без переходу до надмірного рівня навантаження, характерного для важкого режиму.

Разом із тим статичні режими залишаються простішими в реалізації, тестуванні та сприйнятті користувачем. Адаптивний підхід потребує більшої кількості параметрів, правильного добору вагових коефіцієнтів і порогів, а його поведінка є менш передбачуваною. Проведене тестування також обмежувалося однією демонстраційною сценою, одним маршрутом, одним тестувальником і невеликою кількістю сесій, тому отримані результати не охоплюють усі можливі стилі гри та ігрові ситуації.

Отже, поставлену мету роботи досягнуто, а визначені завдання виконано. Розроблений модуль забезпечує збір і аналіз телеметричних даних, формування комплексної оцінки стану гравця, плавне регулювання параметрів складності та захист від навмисного впливу на систему. Результати тестування підтвердили можливість використання запропонованої комбінованої моделі для адаптивного керування складністю в режимі реального часу. Подальше вдосконалення доцільно спрямувати на розширення кількості тестових сцен, залучення більшої кількості користувачів, автоматичне налаштування вагових коефіцієнтів і перевірку роботи модуля в інших жанрах комп’ютерних ігор.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hunicke R. AI for Dynamic Difficulty Adjustment in Games. Proceedings of the AAAI Workshop on Challenges in Game Artificial Intelligence. 2005.

2. Hunicke R. The Case for Dynamic Difficulty Adjustment in Games. Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology. 2005.

3. Fisher N. Dynamic Difficulty Adjustment in Games: Concepts, Techniques and Applications. IntechOpen, 2025.

4. Vang C. The Impact of Dynamic Difficulty Adjustment on Player Experience in Video Games. Scholarly Horizons: University of Minnesota, Morris Undergraduate Journal. 2022.

5. Andrade G., Ramalho G., Santana H., Corruble V. Challenge-Sensitive Action Selection: An Application to Game Balancing. Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology. 2005.

6. Missura O., Gärtner T. Player Modeling for Intelligent Difficulty Adjustment. Proceedings of the 12th International Conference on Discovery Science. 2009.

7. Jennings-Teats M., Smith G., Wardrip-Fruin N. Polymorph: Dynamic Difficulty Adjustment Through Level Generation. Proceedings of the Workshop on Procedural Content Generation in Games. 2010.

8. Lopes R., Bidarra R. Adaptivity Challenges in Games and Simulations: A Survey. IEEE Transactions on Computational Intelligence and AI in Games. 2011.

9. Yannakakis G. N., Togelius J. Experience-Driven Procedural Content Generation. IEEE Transactions on Affective Computing. 2011.

10. Yannakakis G. N., Togelius J. Artificial Intelligence and Games. Cham: Springer, 2018.

11. Csikszentmihalyi M. Flow: The Psychology of Optimal Experience. New York: Harper & Row, 1990.

12. Chen J. Flow in Games. MFA Thesis. University of Southern California, 2007.

13. Sweetser P., Wyeth P. GameFlow: A Model for Evaluating Player Enjoyment in Games. Computers in Entertainment. 2005.

14. Cowley B., Charles D., Black M., Hickey R. Toward an Understanding of Flow in Video Games. Computers in Entertainment. 2008.

15. Nacke L. E., Lindley C. A. Affective Ludology, Flow and Immersion in a First-Person Shooter: Measurement of Player Experience. Loading... The Journal of the Canadian Game Studies Association. 2008.

16. Koster R. A Theory of Fun for Game Design. 2nd ed. Sebastopol: O'Reilly Media, 2013.

17. Schell J. The Art of Game Design: A Book of Lenses. 3rd ed. Boca Raton: CRC Press, 2019.

18. Fullerton T. Game Design Workshop: A Playcentric Approach to Creating Innovative Games. 4th ed. Boca Raton: CRC Press, 2018.

19. Adams E. Fundamentals of Game Design. 3rd ed. Berkeley: New Riders, 2014.

20. Salen K., Zimmerman E. Rules of Play: Game Design Fundamentals. Cambridge: MIT Press, 2004.

21. Millington I., Funge J. Artificial Intelligence for Games. 2nd ed. Boca Raton: CRC Press, 2009.

22. Buckland M. Programming Game AI by Example. Plano: Wordware Publishing, 2005.

23. Bourg D. M., Seemann G. AI for Game Developers. Sebastopol: O'Reilly Media, 2004.

24. Rabin S. Game AI Pro: Collected Wisdom of Game AI Professionals. Boca Raton: A K Peters/CRC Press, 2013.

25. Rabin S. Game AI Pro 2: Collected Wisdom of Game AI Professionals. Boca Raton: CRC Press, 2015.

26. Rabin S. Game AI Pro 3: Collected Wisdom of Game AI Professionals. Boca Raton: CRC Press, 2017.

27. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. 4th ed. Hoboken: Pearson, 2020.

28. Sutton R. S., Barto A. G. Reinforcement Learning: An Introduction. 2nd ed. Cambridge: MIT Press, 2018.

29. Booth M. The AI Systems of Left 4 Dead. Valve, 2009.

30. Valve. Replayable Cooperative Game Design: Left 4 Dead. Game Developers Conference, 2009.

31. Tozour P. The Evolution of Game AI. In: Rabin S. AI Game Programming Wisdom. Hingham: Charles River Media, 2002.

32. Champandard A. J. AI Game Development: Synthetic Creatures with Learning and Reactive Behaviors. Berkeley: New Riders, 2003.

33. Schwab B. AI Game Engine Programming. 2nd ed. Boston: Course Technology, 2008.

34. Gregory J. Game Engine Architecture. 3rd ed. Boca Raton: CRC Press, 2018.

35. Nystrom R. Game Programming Patterns. Genever Benning, 2014.

36. Seif El-Nasr M., Drachen A., Canossa A. Game Analytics: Maximizing the Value of Player Data. London: Springer, 2013.

37. Medler B. Player Dossiers: Analyzing Gameplay Data as a Reward. Game Studies. 2011.

38. El-Nasr M. S., Drachen A., Canossa A. Game Analytics: The Basics. In: Game Analytics. Springer, 2013.

39. Unity Technologies. MonoBehaviour. Unity Scripting API. Electronic resource. Accessed: 01.06.2026.

40. Unity Technologies. Mathf.Clamp01. Unity Scripting API. Electronic resource. Accessed: 01.06.2026.

41. Unity Technologies. Mathf.MoveTowards. Unity Scripting API. Electronic resource. Accessed: 01.06.2026.

42. Unity Technologies. Time.deltaTime. Unity Scripting API. Electronic resource. Accessed: 01.06.2026.

43. Unity Technologies. Input System. Unity Manual. Electronic resource. Accessed: 01.06.2026.

44. Unity Technologies. AI Navigation. Unity Manual. Electronic resource. Accessed: 01.06.2026.

45. Saltsman A. Game Changers: Dynamic Difficulty. Game Developer. 2009. Electronic resource. Accessed: 09.06.2026.