МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інженерії програмного забезпечення
_____ Євген ДАВИДЕНКО
«___»_____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

МОБІЛЬНИЙ ІГРОВИЙ ЗАСТОСУНОК З АДАПТИВНИМИ АЛГОРИТМАМИ НАВЧАННЯ ТА ГЕНЕРАЦІЇ ІГРОВОГО КОНТЕНТУ

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма «Інженерія програмного забезпечення»

Здобувач _____Ілля БЕЛЕВЯТ
«___»_____ 2024 р.

Керівник канд. техн. наук, доцент _____Катерина КІРЕЙ
«___»_____ 2024 р.

Миколаїв – 2024

Чорноморський національний університет імені Петра Могили

(повне найменування закладу вищої освіти)

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Другий (магістерський)
Освітній ступень	Магістр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри інженерії
програмного забезпечення
_____Євген ДАВИДЕНКО
«___»_____2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Белевята Іллі Вікторовича
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи
Мобільний ігровий застосунок з адаптивними алгоритмами навчання та генерації ігрового контенту

Затверджена наказом ЧНУ ім. Петра Могили від «04» вересня 2024 р. № 220

2. Строк представлення кваліфікаційної роботи «___»_____2024 р.

3. Очікуваний результат роботи та початкові дані якщо такі потрібні

Очікуванним результатом є мобільний ігровий застосунок з адаптивними алгоритмами навчання та генерації контенту.

4. Перелік питань, що підлягають розробці:

- проведення аналізу аналогічних систем для інтеграції та дослідження предметної області;
- формування вимог розробки програмного забезпечення;
- проєктування програмного забезпечення, створення відповідних моделей;
- розробка програмного забезпечення;
- тестування розробленої системи;
- аналіз збірних даних.

5. Перелік графічних матеріалів:
презентація

Керівник роботи _____ Катерина КІРЕЙ_____
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач  Ілля БЕЛЕВЯТ
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання « ____»_____2024р

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: Мобільний ігровий застосунок з адаптивними алгоритмами навчання та генерації ігрового контенту _____

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КМР	10.09.2024 р.	11.09.2024 р.	виконано
2.	Огляд літератури за темою роботи	13.09.2024 р.	19.09.2024 р.	виконано
3.	Складання календарного плану КМР	20.03.2024 р.	21.03.2024 р.	виконано
4.	Аналіз предметної області	24.09.2024 р.	26.09.2024 р.	виконано
5.	Розробка проєктних рішень	28.09.2024 р.	30.09.2024 р.	виконано
6.	Моделювання та конструювання ПЗ	01.10.2024 р.	06.10.2024 р.	виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	11.10.2024 р.	07.11.2024 р.	виконано
8.	Оформлення КМР та презентації	10.11.2024 р.	20.11.2024 р.	виконано
9.	Відгук керівника КМР	20.11.2024 р.	21.11.2024 р.	виконано
10.	Попередній захист	28.11.2024 р.	28.11.2024 р.	виконано
11.	Завершення оформлення КМР та презентації	09.12.2024 р.	09.12.2024 р.	виконано

12.	Рецензування	10.12.2024 р.	10.12.2024 р.	виконано
13.	Захист кваліфікаційної роботи	20.12.2024 р.	20.12.2024 р.	виконано

Розробив студент Белевят Ілля Вікторович _____

(прізвище, ім'я, по батькові) *(підпис)*

«__» _____ 2024 р.

Керівник роботи доцент Кірей Катерина Олександрівна___ _____

(посада, прізвище, ім'я, по батькові) *(підпис)*

«__» _____ 2024 р.

АНОТАЦІЯ

до кваліфікаційної магістерської роботи

«Мобільний ігровий застосунок з адаптивними алгоритмами навчання та генерації ігрового контенту»

Здобувач 608 гр.: Белевят Ілля Вікторович

Керівник: доцент Кірей Катерина Олександрівна

Актуальність дослідження обумовлена зростаючим попитом на інноваційні мобільні ігри, які здатні задовольнити потреби широкого кола користувачів завдяки персоналізації контенту та адаптації ігрового процесу.

Об'єктом дослідження є мобільні ігрові застосунки з адаптивними алгоритмами навчання та генерації контенту.

Предметом дослідження виступають методи розробки адаптивних алгоритмів для створення персоналізованого ігрового середовища.

Метою роботи є розробка мобільного ігрового застосунку, який використовує адаптивні алгоритми навчання та генерацію ігрового контенту для підвищення ефективності навчання користувачів та забезпечення динамічної зміни складності гри залежно від прогресу гравця.

Для того щоб досягти поставленної мети, потрібно виконати наступні завдання:

- проведсти аналіз аналогічних систем для інтеграції та дослідження предметної області;
- сформувати вимоги розробки програмного забезпечення;
- спроєктувати програмне забезпечення, створити відповідні моделі;
- розробити програмного забезпечення;
- протестувати розроблені системи;
- провести аналіз збірних даних.

У роботі досліджено проблему автоматизованого підлаштування ігрового процесу під індивідуальні характеристики гравців та запропоновано рішення у

вигляді адаптивних алгоритмів, що допомагають генерувати ігровий контент на основі поведінки користувачів. Об'єктом дослідження є процес адаптації ігрового контенту, а предметом – розробка методів адаптивного навчання у мобільних іграх.

Кваліфікаційна магістерська робота «Мобільний ігровий застосунок з адаптивними алгоритмами навчання та генерації ігрового контенту» складається з 75 сторінок, включає 12 рисунків та 16 джерел. У роботі було використано сучасні технології розробки, такі як Unity, C# та Python, а також методи машинного навчання для створення адаптивних алгоритмів.

Отже, робота містить детальний аналіз предметної галузі, проектування системи, її розробку та тестування, що дозволило створити інноваційний мобільний застосунок, здатний адаптувати ігровий процес до індивідуальних потреб користувачів.

Ключові слова: мобільний застосунок, адаптивні алгоритми, генерація ігрового контенту, машинне навчання.

ABSTRACT

For the master’s thesis

«Mobile Game Application with Adaptive Learning Algorithms and Game Content Generation»

Student 608 group: Belevyat Ilya Viktorovych

Supervisor: Associate Professor Kirei Kateryna Oleksandrivna

The relevance of the research is driven by the growing demand for innovative mobile games capable of meeting the needs of a wide range of users through personalized content and adaptive gameplay.

The subject of the research is the methods for developing adaptive algorithms to create a personalized gaming environment.

The goal of the work is to develop a mobile gaming application that uses adaptive learning algorithms and content generation to enhance user learning efficiency and provide dynamic difficulty adjustments based on the player's progress.

To achieve this goal, the following tasks need to be accomplished:

- Analyze similar systems to integrate and explore the subject area;
- Formulate software development requirements;
- Design the software and create corresponding models;
- Develop the software;
- Test the developed systems;
- Analyze the collected data.

The object of the research is mobile gaming applications with adaptive learning algorithms and content generation.

The thesis addresses the issue of automating gameplay adaptation to individual player characteristics and proposes a solution in the form of adaptive algorithms that generate game content based on user behavior.

The master's thesis, *"Mobile Gaming Application with Adaptive Learning Algorithms and Procedural Content Generation,"* consists of 75 pages, including 12

figures and 16 references. The project employs modern development technologies such as Unity, C#, and Python, as well as machine learning methods to implement adaptive algorithms.

This research includes a comprehensive analysis of the subject area, system design, development, and testing, resulting in an innovative mobile application capable of adapting gameplay to the individual needs of users.

Keywords: mobile application, adaptive algorithms, procedural content generation, machine learning, personalization.

ЗМІСТ

ВСТУП 4
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. СПЕЦИФІКАЦІЯ ВИМОГ 7
1.1 Опис предметної області....7
1.2 Огляд та аналіз аналогів....9
1.3 Специфікація вимог....14
Висновок до розділу 1 21
2 МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОБІЛЬНОГО ІГРОВОГО ЗАСТОСУНКУ 23
2.1 Діаграма прецедентів 23
2.2 Візуальна карта мобільного ігрового застосунку....27
2.3 Проектування бази даних 30
3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 36
3.1 Аналіз вимог....36
3.1.1 Огляд функціональних вимог 36
3.1.2 Нефункціональні вимоги 37
3.2 Архітектура програмного забезпечення....40
3.3 Проєктування алгоритмів 44
3.3.1 Алгоритм генерації кімнат 44
3.3.2 Використання Reinforcement Learning 47
3.3.3 UML-діаграми....52
3.4 Інтерфейси програми....56
3.4.1 Інтерфейс користувача....56
3.4.2 Інтерфейс розробника 58
3.4.3 Адаптивність інтерфейсу....59
Висновки до розділу 3....59
4 КОДУВАННЯ, ТЕСТУВАННЯ ТА ПРОВЕДЕННЯ ОБЧИСЛЕНЬ 62
4.1 Кодування програмного забезпечення 62

4.2 Тестування програмного забезпечення 64
4.3 Проведення обчислень (апробація) 67
4.4 Аналіз результатів 69
4.5 Керівництво користувача 71
Висновок до розділу 4 73
ВИСНОВКИ 74
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 76

ВСТУП

На сьогоднішній день мобільні ігрові застосунки стали невід'ємною частиною сучасного життя, охоплюючи широкі аудиторії по всьому світу. Стрімкий розвиток мобільних технологій та збільшення обчислювальних потужностей пристроїв дозволяють створювати ігри, що можуть конкурувати за якістю з традиційними платформами, такими як консолі та ПК. Із зростанням популярності мобільних ігор також зростають очікування гравців щодо якості контенту та рівня взаємодії, що стимулює розробників до пошуку нових шляхів удосконалення ігрового досвіду. Однією з найперспективніших технологій, яка здатна задовольнити ці потреби, є застосування адаптивних алгоритмів навчання та генерації ігрового контенту.

Адаптивні алгоритми – це методи машинного навчання та штучного інтелекту, які дозволяють системам автоматично підлаштовуватися під поведінку користувача. В контексті мобільних ігор вони використовуються для динамічної зміни складності гри та персоналізації контенту, що забезпечує більш глибоке занурення у гру та підвищує зацікавленість користувача. Гра, що здатна адаптуватися до рівня навичок гравця, створює унікальний досвід, що мотивує гравця розвиватися, не викликаючи фрустрації від надто високого або низького рівня складності. Крім того, генерація ігрового контенту за допомогою таких алгоритмів дозволяє створювати нескінченні варіації рівнів, завдань та ігрових ситуацій, роблячи кожну гру неповторною.

З точки зору користувацького досвіду, адаптивні мобільні ігрові застосунки надають нові можливості для залучення різноманітних аудиторій. Це стосується як початківців, так і досвідчених гравців, оскільки гра здатна підлаштовуватися під конкретного користувача, роблячи процес гри цікавим і викликовим, незалежно від рівня підготовки. Така персоналізація контенту дозволяє не тільки збільшити час, проведений у грі, але й створювати довгострокову лояльність до продукту.

Ця робота присвячена дослідженню мобільних ігрових застосунків, які використовують адаптивні алгоритми навчання та генерації контенту для створення

індивідуалізованого ігрового досвіду. Вибір теми обумовлений зростаючим інтересом до персоналізованих цифрових продуктів, що забезпечують більш глибоку взаємодію користувача з системою. У рамках роботи буде проведено детальний аналіз існуючих технологій, методик та рішень, що дозволять створити ефективну адаптивну систему для мобільного ігрового середовища.

Актуальність цієї роботи полягає у швидкому зростанні вимог до якості та індивідуалізації мобільних ігрових продуктів. Більшість сучасних гравців хочуть бачити в іграх не тільки виклик, а й можливість для самовдосконалення. Адаптивні мобільні ігри дають змогу задовольнити ці потреби, створюючи індивідуальний досвід для кожного користувача. В умовах постійної конкуренції серед розробників мобільних ігор, впровадження адаптивних алгоритмів дає змогу створювати більш конкурентоспроможні продукти, що здатні втримувати увагу гравців на тривалий час.

Об'єкт дослідження: мобільні ігрові застосунки, що використовують адаптивні алгоритми навчання та генерації контенту.

Предмет дослідження: методи розробки адаптивних алгоритмів навчання та генерації ігрового контенту для мобільних ігор.

Мета роботи: розробка мобільного ігрового застосунку, який за допомогою адаптивних алгоритмів навчання та генерації контенту буде автоматично підлаштовуватися під індивідуальні навички гравця, забезпечуючи тим самим персоналізований ігровий досвід і підвищуючи зацікавленість користувачів.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1) провести огляд сучасних мобільних ігрових застосунків, що використовують адаптивні алгоритми навчання та генерації контенту, визначити їх сильні та слабкі сторони, а також перспективи розвитку;

2) специфікувати вимоги до мобільного ігрового застосунку, який використовуватиме адаптивні алгоритми для персоналізації ігрового досвіду користувачів;

3) спроектувати архітектуру мобільного застосунку, що забезпечить динамічне налаштування складності гри та генерацію контенту на основі даних про поведінку гравця;

4) розробити основні модулі застосунку, включаючи систему обробки даних користувача для адаптації рівнів складності, а також модуль генерації ігрового контенту в реальному часі;

5) реалізувати тестування мобільного застосунку з використанням адаптивних алгоритмів та провести оцінку ефективності персоналізованого ігрового досвіду;

6) визначити перспективи вдосконалення адаптивних алгоритмів для подальшого покращення взаємодії користувача з мобільними іграми.

Таким чином, дослідження методів розробки мобільних ігрових застосунків з адаптивними алгоритмами навчання та генерації контенту є актуальним та своєчасним кроком для вдосконалення процесів створення персоналізованого ігрового досвіду.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. СПЕЦИФІКАЦІЯ ВИМОГ

1.1 Опис предметної області

Розробка мобільного ігрового застосунку з адаптивними алгоритмами навчання та генерації ігрового контенту є складним і багатогранним завданням, що вимагає врахування багатьох аспектів як технічного, так і користувацького характеру. Однією з ключових вимог до такого застосунку є забезпечення динамічного ігрового процесу, який буде адаптуватися під рівень навичок кожного окремого гравця, що значно підвищує його залученість і тривалість взаємодії з продуктом.

Мобільні ігри стали однією з найпопулярніших форм цифрових розваг у світі. Завдяки зростанню доступності смартфонів і планшетів, кількість користувачів мобільних ігор щорічно зростає, охоплюючи мільйони людей по всьому світу. Гравці очікують високої якості, реалістичної графіки, захоплюючих сценаріїв та персоналізації, що підвищує інтерес до гри.

Адаптивні алгоритми навчання дозволяють мобільному застосунку змінювати складність гри на основі аналізу поведінки гравця. Це може включати відстеження успіхів, невдач, часу проходження рівнів і інших метрик, які допомагають системі "навчитися" підлаштовувати гру відповідно до індивідуальних потреб користувача. Такий підхід дозволяє створювати ігри, які не тільки підтримують інтерес, але й сприяють розвитку навичок та вирішенню задач різної складності, поступово підвищуючи рівень виклику.

Одним із центральних аспектів проєктування мобільних ігор є управління ігровим контентом. У випадку адаптивних ігор особливу увагу слід приділяти генерації контенту в реальному часі. Це означає, що система повинна мати здатність створювати нові ігрові сценарії, рівні або завдання на основі поточного стану гри та індивідуальних характеристик гравця. Генерація контенту повинна бути достатньо гнучкою, щоб забезпечити різноманітність і запобігти рутині, що є одним із ключових факторів для утримання користувача.

Ще одним важливим аспектом є забезпечення зручного та інтуїтивно зрозумілого інтерфейсу. UX/UI дизайн грає важливу роль у взаємодії користувача із застосунком. Дизайн інтерфейсу повинен відповідати високим стандартам зручності використання, надавати швидкий доступ до функцій і контенту гри, а також бути привабливим з візуальної точки зору. Погано спроєктований інтерфейс може призвести до розчарування користувача і знизити його зацікавленість у грі, навіть якщо ігровий процес є динамічним і цікавим.

Актуальність адаптивних алгоритмів

Один із ключових аспектів успіху мобільних ігор – це здатність розробників задовольняти різноманітні потреби гравців. Новачки шукають прості рівні для знайомства з ігровим процесом, тоді як досвідчені гравці вимагають викликів, які дозволяють розвивати свої навички.

Основні характеристики адаптивності:

1) персоналізація ігрового процесу. Гра має динамічно адаптуватися до здібностей гравця. Наприклад, якщо користувач стикається з труднощами на певному рівні, алгоритм може автоматично запропонувати підказку або спростити завдання;

2) реалізація навчальних механізмів. Завдяки адаптивності ігри можуть виступати інструментом навчання. Наприклад, у квестах або головоломках гравець може розвивати логічне мислення, навички стратегічного планування та швидкість реакції;

3) динамічне генерування контенту. Кожен рівень може бути унікальним, завдяки чому гравець стикається з новими сценаріями, які стимулюють його залишатися в грі довше.

Виклики у впровадженні адаптивних ігор. Розробка адаптивних мобільних ігор пов’язана з низкою труднощів:

1) високі вимоги до ресурсів. Адаптивні алгоритми, зокрема ті, що використовують машинне навчання, потребують значних обчислювальних потужностей;

2) складність тестування. Динамічний контент ускладнює процес тестування, адже кожен сценарій може бути унікальним;

3) інтеграція з користувацьким інтерфейсом. Гра повинна залишатися інтуїтивно зрозумілою, навіть якщо її складність змінюється в реальному часі.

Таким чином, розробка мобільного ігрового застосунку з адаптивними алгоритмами вимагає не тільки технічної реалізації алгоритмів навчання та генерації контенту, але й уваги до користувацького досвіду. Комплексне розуміння поведінки гравця, аналіз його дій та коректна інтеграція отриманих даних для побудови персоналізованого ігрового досвіду є ключем до успіху подібних застосунків на ринку.

1.2 Огляд та аналіз аналогів

Ринок мобільних ігор постійно розширюється, пропонуючи гравцям широкий вибір застосунків. Багато сучасних ігор уже використовують елементи адаптивності, хоча ці підходи не завжди є досконалими. Для виявлення сильних та слабких сторін існуючих рішень було проведено аналіз кількох популярних мобільних ігор, які частково реалізують адаптивні алгоритми.

Приклади сучасних ігор

Angry Birds Dream Blast

Розробник: Rovio Entertainment.

Особливості: Поступове збільшення складності рівнів, що забезпечується внутрішніми механізмами, які відстежують успіхи гравця. Наприклад, якщо користувач часто програє, гра пропонує йому бонуси або додаткові спроби.

Сильні сторони: Проста реалізація адаптивності, яка позитивно впливає на ігровий досвід новачків.

Недоліки: Обмеженість у створенні унікального досвіду для досвідчених гравців.

Clash Royale

Розробник: Supercell.

Особливості: Використовує систему рейтингу для підбору супротивників. Гравці з подібним рівнем навичок змагаються між собою, що робить гру більш справедливою.

Сильні сторони: Висока якість мультиплеєра завдяки рейтинговій системі.

Недоліки: Адаптивність обмежується лише багатокористувацьким режимом і не враховує поведінку гравця під час проходження одиночних завдань.

Monument Valley 2

Розробник: ustwo games.

Особливості: Адаптація складності головоломок залежно від часу, витраченого на їх розв’язання. Гра забезпечує плавний перехід між рівнями, що допомагає утримувати увагу користувача.

Сильні сторони: Естетичний дизайн і плавний ігровий процес.

Недоліки: Відсутність інноваційних адаптивних механізмів.

Таблиця 1 порівняння аналогів:

Назва гри	**Технології**	**Рівень адаптивності**	**Основний недолік**
Angry Birds Dream Blast	Unity, C#	Поступове підвищення складності	Обмеженість адаптації до окремих рівнів
Clash Royale	Custom Engine, Java	Підбір гравців за рейтингом	Висока залежність від внутрішніх покупок

Завершення таблиці 1.

Продовження таблиці 1.

Назва гри	**Технології**	**Рівень адаптивності**	**Основний недолік**
Monument Valley 2	Unity, C#	Персоналізація складності головоломок	Відсутність складних адаптивних механізмів

Архітектура:

Усі ці ігри побудовані на основі клієнт-серверної архітектури, де клієнтська частина відповідає за взаємодію з користувачем та відображення ігрового процесу, а серверна частина – за збереження ігрових даних, обробку запитів і генерацію контенту на основі отриманих даних від гравця. Така архітектура дозволяє масштабувати гру, додавати нові функції та контент без необхідності постійного оновлення клієнтської частини.

Мови реалізації:

Більшість сучасних ігор використовують комбінацію мов програмування, таких як C++, Unity (C#) для клієнтської частини, а також Python, Java або Go для серверної частини. Такі технології забезпечують високий рівень продуктивності та гнучкість в адаптації до змінних умов ігрового процесу.

Перелік функцій:

1) адаптація складності рівнів на основі прогресу гравця;
2) генерація нових рівнів або завдань на основі попередніх досягнень;
3) персоналізація контенту відповідно до індивідуальних інтересів гравця;
4) соціальні функції, такі як інтеграція з соціальними мережами, мультиплеєрний режим та змагання з іншими гравцями.

Аналіз переваг та недоліків даних ігор:

Переваги:

1) персоналізація ігрового досвіду: адаптивні алгоритми дозволяють створювати індивідуалізований ігровий процес для кожного гравця;

2) залучення користувачів: системи адаптації утримують гравця в грі, збільшуючи її тривалість і частоту сеансів;

3) масштабованість: серверна архітектура дозволяє легко масштабувати гру та підтримувати велику кількість активних користувачів.

Недоліки:

1) високі вимоги до обчислювальних ресурсів: для реалізації складних адаптивних алгоритмів потрібні потужні сервери та клієнтські пристрої;

2) складність тестування: динамічний контент ускладнює процес тестування, оскільки існує безліч варіацій гри, які необхідно врахувати;

3) нестабільність адаптивних алгоритмів: в деяких випадках адаптація може призвести до занадто швидкої зміни рівня складності, що може викликати дискомфорт у гравців.

Таким чином, аналіз сучасних мобільних ігор з адаптивними алгоритмами дозволяє визначити перспективні напрямки для розвитку нового мобільного ігрового застосунку. Важливо використовувати найкращі практики з існуючих продуктів, водночас усуваючи їх недоліки для створення більш ефективного ігрового процесу.

Аналіз системи, що розробляється

-призначення ПЗ

Проєкт мобільної гри з використанням штучного інтелекту (ШІ) на основі Unity має на меті створити динамічне ігрове середовище, де гравці можуть взаємодіяти з віртуальними персонажами та світом, який адаптується до їхньої поведінки. Основні функції гри включають інтелектуальну поведінку NPC (негравців), адаптивні сценарії та завдання, а також аналітичну систему, що вивчає гравців для поліпшення ігрового процесу. Використання штучного інтелекту

дозволяє персоналізувати ігровий досвід кожного користувача, створюючи унікальну ігрову подорож для кожного гравця.

-опис структури системи (схема)

Основні компоненти системи мобільної гри на основі Unity з ШІ включають наступне:

1) інтерфейс користувача (UI): Користувацький інтерфейс включає в себе меню, ігрові екрани та інформаційні панелі, через які гравець може взаємодіяти з грою, переглядати свою статистику, виконувати дії та керувати налаштуваннями;

2) штучний інтелект (AI): Головний компонент, що відповідає за адаптивність і динаміку ігрового процесу. TensorFlow використовується для машинного навчання, що дозволяє персонажам і елементам гри адаптуватися до дій гравця, покращуючи свою поведінку та взаємодію з ігровим середовищем;

3) ігровий рушій (Unity): Unity відповідає за управління графікою, фізикою, анімаціями, взаємодією об’єктів та всіма іншими аспектами гри. Unity забезпечує кросплатформену підтримку для мобільних пристроїв (Android, iOS) і має потужну інтеграцію з TensorFlow для роботи з ШІ;

4) база даних (Firebase): Використовується для зберігання інформації про гравців, досягнення, прогрес і інші дані. Firebase дозволяє синхронізувати дані в реальному часі, надаючи стабільну підтримку для роботи з великою кількістю користувачів і динамічним контентом.

5) система облікових записів: Відповідає за створення та управління профілями гравців. Firebase Authentication надає можливість інтеграції соціальних мереж, дозволяючи гравцям входити в гру через Google, Facebook або інші платформи;

6) система хмарних функцій (Google Cloud): Використовується для складної обробки даних, масштабування гри та аналітики. Google Cloud AI

може допомогти в обробці великих обсягів даних про гравців і їхню взаємодію з грою, що дозволяє поліпшити адаптивність гри та зробити її більш інтерактивною.

-засоби апаратної та програмної реалізації

Unity: Основний ігровий рушій для розробки мобільної гри. Unity забезпечує потужні засоби для створення 2D і 3D ігор, дозволяє легко реалізувати візуальні ефекти, анімацію та інтегрувати штучний інтелект. Unity також підтримує платформу TensorFlow для реалізації машинного навчання в ігрових процесах.

TensorFlow: Фреймворк для розробки моделей машинного навчання. TensorFlow Lite використовується для впровадження моделей ШІ безпосередньо в мобільні додатки, що дозволяє NPC (негравцям) розвивати свою поведінку і навчатися на основі дій гравця.

Firebase: Хмарна платформа для зберігання даних, аутентифікації та реального часу. Firebase забезпечує масштабоване зберігання даних, що дозволяє зберігати прогрес гравців і керувати динамічними ігровими даними.

C#: Основна мова програмування, яка використовується для створення логіки гри в Unity. C# дозволяє легко інтегрувати об'єктно-орієнтоване програмування в ігрову логіку та забезпечує просту взаємодію з ШІ-моделями, розробленими в TensorFlow.

Google Cloud AI: Використовується для обробки великих обсягів даних і аналітики. Google Cloud надає сервіси, які дозволяють аналізувати поведінку гравців, а також масштабувати роботу гри відповідно до зростаючої кількості користувачів.

1.3 Специфікація вимог

1. ПРИЗНАЧЕННЯ ТА МЕЖІ ПРОЄКТУ

1.1 Призначення системи, для якої розробляється програмне забезпечення

Програмне забезпечення розробляється для мобільного ігрового застосунку з адаптивними алгоритмами навчання та генерації ігрового контенту. Ця система має

на меті забезпечити динамічний та індивідуалізований ігровий процес для кожного користувача, використовуючи штучний інтелект для адаптації складності та контенту залежно від поведінки та навичок гравця.

1.2 Межі проєкту ПЗ

Межами проєкту є розробка мобільного ігрового застосунку, який здатен ефективно працювати на різних платформах (Android, iOS) з використанням Unity як ігрового рушія, TensorFlow для алгоритмів ШІ, Firebase для зберігання та обробки даних, а також Google Cloud для масштабованості. Проєкт орієнтований на розробку ігрового контенту, що буде адаптуватися на основі поведінки гравця.

2. ЗАГАЛЬНИЙ ОПИС

2.1 Сфера застосування

Мобільний ігровий застосунок дозволяє користувачам занурюватися в індивідуалізований ігровий світ, де контент та складність автоматично змінюються, враховуючи рівень гри та поведінку користувача. Основні функції включають генерування нових рівнів і завдань, динамічну поведінку персонажів, а також можливість гравцям переглядати статистику, досягнення та прогрес. Гра підходить для різних вікових категорій, надаючи як прості, так і складніші сценарії на основі даних про гравця.

2.2 Характеристики користувачів

Користувачами гри можуть бути люди різного віку та рівня підготовки, включаючи дітей, підлітків та дорослих. Система адаптації забезпечує відповідний рівень складності залежно від індивідуальних навичок і дій користувача, роблячи гру цікавою для гравців різних рівнів.

2.3 Загальна структура і склад системи

Система включає такі основні компоненти:

Інтерфейс користувача (UI): Включає меню, ігрові екрани, статистику та налаштування, які надають користувачеві зручні засоби для керування грою.

Штучний інтелект (AI): Відповідає за адаптивність гри, використовуючи TensorFlow для навчання та моделювання поведінки персонажів і навколишнього середовища.

Ігровий рушій (Unity): Забезпечує графіку, фізику, анімацію та інші аспекти гри, а також підтримку мобільних платформ.

База даних (Firebase): Використовується для зберігання даних гравців, прогресу, досягнень та динамічного контенту гри.

Хмарна інфраструктура (Google Cloud): Забезпечує складну обробку та зберігання великих обсягів даних, включаючи хмарні функції для машинного навчання та аналітики.

2.4 Загальні обмеження

Система не включатиме комплексні соціальні функції, такі як інтеграція спільнот або можливість одночасної гри багатьох гравців. Проєкт фокусується на одиночній грі з адаптивними сценаріями.

3. ФУНКЦІЇ СИСТЕМИ

3.1 Система адаптивного навчання

3.1.1 Опис функції:

Система адаптивного навчання використовує алгоритми штучного інтелекту для оцінки навичок користувача і відповідно до цього коригує складність гри та геймплей. Алгоритми збирають дані про те, як користувач взаємодіє з грою, і відповідно модифікують рівні, виклики або поведінку NPC (неігрових персонажів).

3.1.2 Вхідна та вихідна інформація:

Система отримує інформацію про дію гравця, її ефективність і стратегію, і на основі цих даних генерує нові сценарії.

3.1.3 Функціональні вимоги:

Система повинна забезпечувати безперебійну адаптацію гри під користувача, аналізуючи його дії в режимі реального часу і коригуючи поведінку ігрових елементів.

3.2 Генерація ігрового контенту

3.2.1 Опис функції:

Ця функція дозволяє створювати нові рівні, локації та сценарії залежно від дій гравця та його прогресу в грі. Система автоматично генерує унікальний контент для кожного користувача, забезпечуючи варіативність і неповторність ігрового досвіду.

3.2.2 Вхідна та вихідна інформація:

На основі даних про прогрес та поведінку гравця генерується новий ігровий контент, що відповідає рівню складності та стилю гри гравця.

3.2.3 Функціональні вимоги:

Система повинна забезпечувати безперервну генерацію нового контенту без повторів, орієнтуючись на індивідуальні дані користувача.

3.3 Система облікових записів користувачів

3.3.1 Опис функції:

Система облікових записів дозволяє гравцям створювати власні профілі, зберігати дані про прогрес і досягнення, а також отримувати доступ до різних функцій гри.

3.3.2 Вхідна та вихідна інформація:

Користувачі вводять дані для реєстрації, після чого отримують доступ до системи для збереження прогресу і налаштувань.

3.3.3 Функціональні вимоги:

Система повинна забезпечувати безпечне зберігання та обробку інформації про користувачів, включаючи аутентифікацію через соціальні мережі (Google, Facebook).

4. ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Джерела та зміст вхідної інформації

Система спиратиметься на дані, які генеруються під час гри, включаючи дії гравця, його досягнення, та інформацію, введену для реєстрації і аутентифікації.

4.2 Вимоги до методів організації, зберігання та підтримки інформації

Дані користувачів повинні зберігатися у Firebase, забезпечуючи безпечність і захищеність персональних даних, регулярне резервне копіювання, а також масштабованість для великої кількості гравців.

5. ВИМОГИ ДО АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ

Для роботи застосунку потрібні сервери з достатньою обчислювальною потужністю для роботи алгоритмів ШІ та зберігання великих обсягів ігрового контенту та користувацьких даних.

6. ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

6.1 Архітектура програмного забезпечення

Програмне забезпечення мобільного застосунку повинно мати модульну архітектуру з чітким розподілом компонентів:

Інтерфейс користувача (UI): відповідає за взаємодію користувача з грою, включаючи екрани меню, елементи управління та візуальне відображення ігрового процесу.

Адаптивні алгоритми ШІ: використовують TensorFlow для машинного навчання та генерації індивідуалізованого ігрового контенту. Ці алгоритми працюють у фоновому режимі, оцінюючи дії гравця та коригуючи складність гри в реальному часі.

База даних (Firebase): зберігає дані про користувачів, прогрес у грі, налаштування та досягнення. Firebase також забезпечує хмарне зберігання і синхронізацію даних між пристроями.

Хмарні сервіси (Google Cloud): використовуються для обробки великих обсягів даних та забезпечення масштабованості. Google Cloud надає інструменти для навчання моделей ШІ та керування великими базами даних.

6.2 Програмне забезпечення системи

Програмне забезпечення реалізується за допомогою таких технологій:

Unity — основний ігровий рушій для розробки мобільного застосунку, що забезпечує підтримку графіки, фізики та анімацій.

TensorFlow — для створення адаптивних алгоритмів машинного навчання, що дозволяють динамічно коригувати складність гри.

Firebase — використовується для зберігання даних користувачів, збереження прогресу, хмарної синхронізації та аутентифікації.

Google Cloud — для обробки великих даних і підтримки масштабованості, що дозволяє керувати обчислювальними ресурсами для навчання моделей ШІ.

6.3 Мережеве програмне забезпечення

Мережевий рівень застосунку базується на RESTful API, який забезпечує взаємодію між клієнтським застосунком та серверною частиною. Зокрема, API забезпечує:

Отримання та відправлення даних користувачів (реєстрація, авторизація, збереження прогресу).

Інтеграцію з Firebase для управління користувацькими даними та статистикою.

Доступ до хмарних сервісів Google Cloud для навчання моделей та генерації адаптивного контенту.

7. ВИМОГИ ДО БЕЗПЕКИ

7.1 Захист персональних даних

Застосунок повинен відповідати стандартам безпеки та конфіденційності, забезпечуючи захист персональних даних користувачів відповідно до міжнародних норм, таких як GDPR (Загальний регламент про захист даних). Усі персональні дані, такі як реєстраційна інформація та дані про прогрес гри, повинні бути зашифровані та захищені від несанкціонованого доступу.

7.2 Аутентифікація та авторизація

Застосунок повинен підтримувати багатофакторну аутентифікацію, що включає:

Аутентифікацію через соціальні мережі (Google, Facebook).

Вхід через логін і пароль, з можливістю відновлення доступу за допомогою електронної пошти.

Авторизація доступу до персональних даних повинна контролюватися через систему прав доступу на основі Firebase Authentication.

7.3 Захист від шкідливого програмного забезпечення

Застосунок повинен бути захищений від різних форм атак, таких як SQL-ін’єкції, міжсайтовий скриптинг (XSS), атаки типу «відмова в обслуговуванні» (DDoS). Всі дані, що передаються через мережу, повинні бути зашифровані за допомогою SSL/HTTPS протоколів.

7.4 Резервне копіювання даних

Firebase повинна забезпечувати регулярне резервне копіювання даних користувачів, що дозволить уникнути втрати прогресу у випадку збою системи або атак. Хмарні сервіси Google Cloud можуть бути використані для зберігання резервних копій великих обсягів даних.

8. ІНШІ ВИМОГИ

8.1 Умови використання

Програмне забезпечення повинно бути безкоштовним для завантаження та використання, з можливістю внутрішньоігрових покупок, які дозволять гравцям отримувати додатковий контент або прискорювати прогрес.

8.2 Ліцензування

Застосунок повинен відповідати ліцензійним вимогам для використання стороннього програмного забезпечення, такого як Unity, TensorFlow та Firebase, з урахуванням умов їхнього використання для комерційних або некомерційних цілей.

8.3 Вимоги до продуктивності

Продуктивність гри повинна бути оптимізована для мобільних пристроїв, з мінімальними вимогами до апаратного забезпечення:

Підтримка пристроїв на базі Android (версії 6.0 і вище) та iOS (версії 12.0 і вище).

Гра повинна підтримувати стабільний FPS (кадри в секунду) не нижче 30 кадрів на слабких пристроях, і 60 кадрів на більш продуктивних.

8.4 Кросплатформенна підтримка

Застосунок має забезпечувати кросплатформенну підтримку з можливістю синхронізації даних між пристроями Android та iOS, використовуючи Firebase для зберігання прогресу та інших даних у хмарі.

8.5 Масштабованість

Хмарна інфраструктура (Google Cloud) повинна підтримувати масштабованість для обробки великої кількості гравців одночасно, забезпечуючи належну продуктивність незалежно від навантаження.

8.6 Технічна підтримка

Застосунок повинен мати інтегровану систему підтримки користувачів, включаючи базу знань, часті запитання (FAQ), та можливість звернення до технічної підтримки через спеціальну форму або електронну пошту.

Висновок до розділу 1

У першому розділі роботи проведено ґрунтовний аналіз предметної галузі мобільних ігрових застосунків з адаптивними алгоритмами навчання та генерації контенту. Описано ключові аспекти функціонування таких систем, зокрема процес адаптації ігрових умов до індивідуальних особливостей гравців, що дозволяє створювати більш інтерактивний та персоналізований ігровий досвід.

Проведений огляд та аналіз аналогів допоміг виявити актуальні тенденції та технологічні рішення в сфері мобільних ігор, зокрема використання машинного навчання для динамічного управління складністю гри. Були розглянуті існуючі продукти, які частково реалізують подібні функції, що дозволило сформулювати сильні та слабкі сторони наявних рішень.

Також було розроблено загальну архітектурну схему системи, яка включає користувацький інтерфейс, базу даних, адаптивні алгоритми та хмарні сервіси для зберігання і обробки даних. Описані засоби апаратної та програмної реалізації,

зокрема використання таких технологій, як Unity, TensorFlow, Firebase та Google Cloud, які забезпечують масштабованість, надійність та продуктивність системи.

Специфікація вимог, що була сформована на основі аналізу предметної галузі та аналогів, окреслює основні функціональні та технічні вимоги до застосунку, включаючи вимоги до користувацького інтерфейсу, системи управління даними та алгоритмів адаптивного навчання. Визначені вимоги забезпечують розробку мобільного застосунку, що відповідає сучасним потребам ринку та користувачів, а також дозволяє ефективно адаптувати ігровий процес під кожного гравця.

2 МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОБІЛЬНОГО ІГРОВОГО ЗАСТОСУНКУ

2.1 Діаграма прецедентів

Діаграма використання — це одна з основних діаграм UML (Unified Modeling Language), яка відображає функціональність системи з точки зору користувачів і показує, як користувачі взаємодіють із нею. Діаграма містить акторів (користувачів або інші системи) та їхню взаємодію із системою через різні варіанти використання. Для мобільного ігрового застосунку з адаптивними алгоритмами навчання та генерації ігрового контенту така діаграма є важливим інструментом, який дозволяє зрозуміти основні сценарії використання, ролі користувачів та інтеракції з основними функціональними можливостями.

Основні актори системи:

1. **Гравець** — основний користувач мобільного ігрового застосунку, який взаємодіє із системою для гри, отримання нових завдань, адаптації свого досвіду та досягнення прогресу.
2. **Адміністратор системи** — користувач, який відповідає за технічну підтримку, контроль оновлень і коригування системних налаштувань.
3. **Ігровий AI (алгоритм)** — внутрішній модуль системи, який автоматично адаптує рівень складності гри, генерує контент на основі поведінки гравця та його результатів.

Опис дійових осіб:

1) **Гравець**: має змогу входити в систему, проходити різні рівні гри, отримувати індивідуальні завдання та покращувати свої навички. Завдяки адаптивним алгоритмам навчання, гра підлаштовується під стиль і прогрес користувача, пропонуючи дедалі складніші завдання або підказки;

2) **Адміністратор системи**: контролює базу користувачів, проводить технічне обслуговування, виправляє баги і відслідковує виконання оновлень. Може налаштовувати системні параметри адаптивних алгоритмів або випускати нові версії гри;

3) **Ігровий AI**: реалізує функції динамічної адаптації ігрового процесу на основі індивідуальних особливостей гравця. Цей алгоритм відповідає за генерацію ігрового контенту, пропозицію нових рівнів, а також корекцію складності у залежності від прогресу.

Основні варіанти використання:

1) **реєстрація гравця;**

Передумови: гравець повинен мати встановлений застосунок.

Постумови: гравець успішно реєструється в системі, створюється його персональний профіль.

2) **вхід в систему;**

Передумови: користувач повинен мати обліковий запис.

Постумови: користувач успішно входить у систему, його прогрес синхронізується з сервером.

3) **проходження рівнів;**

Передумови: користувач обрав рівень для проходження.

Постумови: система фіксує прогрес гравця, адаптує подальші завдання залежно від результатів.

4) **адаптивне навчання;**

Передумови: користувач грає в гру, проходить рівні та виконує завдання.

Постумови: гра підлаштовує рівень складності під здібності гравця, система зберігає прогрес.

5) **генерація ігрового контенту;**

Передумови: гравець взаємодіє з ігровим AI.

Постумови: генерується новий контент (рівні, завдання, локації), система підлаштовує гру під індивідуальні особливості користувача.

6) **отримання рекомендацій;**

Передумови: користувач активно грає і прогресує.

Постумови: система надає рекомендації щодо подальших дій у грі, підказки для покращення навичок.

7) **вихід з гри;**

Передумови: користувач може зберегти свій прогрес перед виходом.

Постумови: система зберігає останній прогрес, синхронізується з сервером, дані гравця оновлюються.

8) **оновлення гри.**

Передумови: нова версія гри доступна на сервері.

Постумови: адміністратор оновлює систему, користувач отримує повідомлення про нову версію, може завантажити її.

2.2 Опис сценаріїв використання

Сценарій 1: "Реєстрація гравця"

Актори: Гравець

Короткий опис: Гравець проходить процедуру реєстрації для створення нового облікового запису.

Мета: Створити персональний профіль.

Тип: Базовий.

Типовий хід подій:

1) гравець відкриває застосунок і вибирає опцію реєстрації;
2) система виводить форму для заповнення даних;
3) гравець вводить ім'я, адресу електронної пошти та пароль;
4) система створює новий обліковий запис і відправляє підтвердження на електронну пошту;

Винятки:

5) якщо електронна пошта вже зареєстрована — система виводить повідомлення про помилку;
6) якщо користувач ввів некоректні дані — система пропонує повторно заповнити форму.

Сценарій 2: "Вхід в систему"

Актори: Гравець

Короткий опис: Гравець входить у систему для продовження гри.

Мета: Отримати доступ до персонального профілю і збереженого прогресу.

Тип: Базовий.

Типовий хід подій:

1) гравець відкриває застосунок і натискає кнопку "Вхід";
2) система запитує електронну пошту та пароль;
3) гравець вводить свої дані;
4) система перевіряє облікові дані й відкриває основний екран гри;

Винятки:

5) якщо користувач ввів неправильний пароль — система показує повідомлення про помилку та пропонує ввести дані ще раз;
6) якщо обліковий запис заблоковано — система інформує гравця про це та пропонує звернутися в технічну підтримку.

Сценарій 3: "Проходження рівнів"

Актори: Гравець

Короткий опис: Гравець обирає рівень гри для проходження та взаємодіє з ігровим процесом.

Мета: Проходити рівні, підвищуючи свій прогрес і отримуючи нові досягнення.

Тип: Базовий.

Типовий хід подій:

1) гравець обирає доступний рівень;
2) система завантажує обраний рівень і починає ігровий процес;
3) гравець проходить рівень, вирішує завдання та взаємодіє з ігровими елементами;

4) система оцінює прогрес та фіксує результати.

Сценарій 4: "Адаптивне навчання"

Актори: Гравець, Ігровий AI

Короткий опис: Гравець отримує підказки та рекомендації щодо гри, які адаптуються під його дії.

Мета: Підвищити рівень гри, отримуючи корисні поради для покращення.

Тип: Розширений.

Типовий хід подій:

1) під час гри система аналізує дії гравця;
2) якщо гравець не може пройти рівень, система пропонує підказки або рекомендації;
3) підказки адаптуються залежно від стратегії та навичок гравця.

Рисунок 2.1 – Діаграма прецендентів

2.2 Візуальна карта мобільного ігрового застосунку

Візуальна карта мобільного ігрового застосунку є важливим інструментом для проєктування логіки взаємодії користувача з системою. Вона дозволяє розробникам і дизайнерам побудувати зрозумілу і структуровану навігацію між основними екранами застосунку, забезпечуючи інтуїтивність і комфорт для кінцевого користувача.

Завдання візуальної карти:

1) Систематизація структури застосунку. Візуальна карта дозволяє визначити всі основні елементи інтерфейсу та їх взаємозв’язки;
2) Оптимізація навігації. Завдяки карті розробники можуть забезпечити мінімальну кількість кроків для досягнення ключових функцій;

3) Планування користувацького досвіду (UX). Вона сприяє створенню інтуїтивно зрозумілого інтерфейсу, що враховує потреби користувачів різного рівня підготовки;

4) Візуалізація функціональних зв'язків. Карта чітко демонструє, як взаємодіють між собою різні компоненти гри, зокрема головне меню, рівні та налаштування.

Опис основних елементів карти

Візуальна карта мобільного застосунку складається з кількох рівнів і ключових елементів, кожен із яких виконує певну роль у взаємодії користувача із системою:

a) екран входу та реєстрації;

1) призначення: надає доступ до профілю гравця або дозволяє створити новий обліковий запис;

2) особливості:

- форма авторизації (логін, пароль);
- можливість входу через соціальні мережі (Google, Facebook);
- опція відновлення пароля.

b) головне меню;

1) призначення: забезпечує доступ до основних функцій гри;

2) компоненти:

- кнопка «Розпочати гру» для переходу до рівнів;
- секція «Досягнення», яка відображає статистику гравця;
- розділ «Налаштування» для змін параметрів гри (гучність, мова, графіка).

c) екран вибору рівнів;

1) призначення: дозволяє користувачеві обрати рівень гри залежно від доступності;

2) особливості:

– рівні згруповані за складністю або тематикою;
– індикатор прогресу (наприклад, завершено 50% рівня);
– замкнені рівні позначені відповідною іконкою (наприклад, замок).

d) ігровий екран;

1) призначення: головний елемент гри, де гравець взаємодіє з NPC, вирішує завдання та проходить рівні;

2) компоненти:

– панель управління (рух, стрибки, атаки);
– інтерактивні об’єкти (вороги, бонуси, пастки);
– таймер або індикатор завдань.

e) екран результатів;

1) призначення: відображає результати проходження рівня;

2) особливості:

– кількість набраних балів;
– час, витрачений на проходження;
– кількість виконаних завдань.

f) налаштування та довідка.

1) призначення: забезпечує зміну параметрів гри та доступ до інформації про механіку;

2) особливості:

– регулювання графіки, звуку, мови;
– розділ із відповідями на поширені запитання (FAQ).

Логіка переходів

Візуальна карта забезпечує плавну і логічну навігацію між усіма екранними модулями:

1) гравець відкриває застосунок і потрапляє на екран входу;

2) після авторизації він переходить до головного меню, звідки обирає рівень гри;

3) під час гри всі взаємодії відбуваються через ігровий екран, включаючи виконання завдань, використання підказок або вихід до меню;
4) після завершення рівня система показує екран результатів із можливістю перейти до наступного рівня.

На рисунку 2.2 наведено візуальну карту мобільного ігрового застосунку. Вона демонструє всі ключові елементи системи та їхні взаємозв'язки.

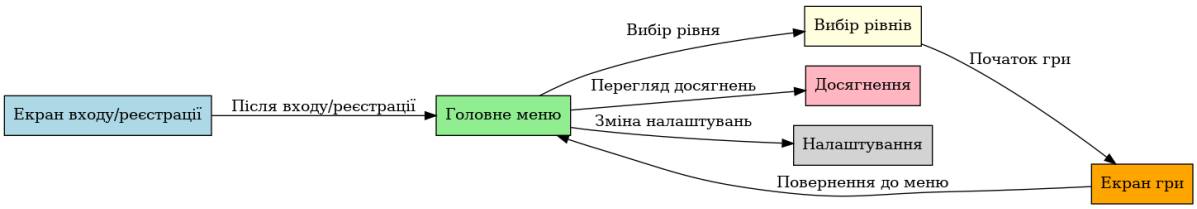

Рисунок 2.2 – Візуальна карта мобільного ігрового застосунку.

Переваги використання візуальної карти

1) покращення UX. Зрозуміла логіка переходів між екранами забезпечує позитивний користувацький досвід;
2) ефективне тестування. Карта допомагає тестувальникам виявляти проблеми з навігацією та взаємодією на ранніх етапах розробки;
3) підтримка масштабованості. Додавання нових функцій або модулів можна планувати, враховуючи існуючу карту.

Візуальна карта мобільного ігрового застосунку є основою для проектування інтерфейсу, що враховує потреби користувачів. Її структура сприяє створенню інтуїтивно зрозумілого дизайну, зменшуючи час навчання користувачів та підвищуючи їхню залученість у гру.

2.3 Проектування бази даних

Проектування бази даних є критичним етапом у розробці мобільного ігрового застосунку, оскільки саме вона забезпечує зберігання, обробку і доступ до всіх необхідних даних для роботи гри. У мобільному ігровому застосунку з адаптивними алгоритмами навчання та генерації контенту база даних повинна бути побудована

таким чином, щоб ефективно обробляти динамічну інформацію, пов’язану з прогресом гравців, параметрами адаптації та іншими аспектами ігрового процесу.

Завдання бази даних

База даних у цій системі виконує наступні ключові завдання:

1) збереження даних гравця. Включає облікову інформацію (ім’я, електронна пошта, пароль), прогрес у грі (досягнення, рівні, результати) та персональні налаштування;
2) зберігання ігрового контенту. Містить інформацію про рівні, завдання, поведінку NPC (неігрових персонажів) та їх характеристики;
3) аналіз та оновлення даних. Дані про поведінку гравця (час проходження рівнів, кількість помилок) використовуються для динамічної адаптації складності;

Вибір технології

Для реалізації бази даних було обрано платформу Firebase через її наступні переваги:

1) хмарне зберігання. Firebase дозволяє зберігати дані в хмарі, забезпечуючи доступ до них у реальному часі з будь-якого пристрою;
2) реальна синхронізація. Firebase дозволяє миттєво оновлювати дані на пристроях користувачів, що є важливим для динамічної адаптації гри;
3) масштабованість. Платформа автоматично масштабує ресурси залежно від кількості активних користувачів;
4) інтеграція з інструментами розробки. Firebase легко інтегрується з Unity, що спрощує процес розробки та тестування.

Основні сутності бази даних

Проектування бази даних передбачає виділення ключових сутностей і зв’язків між ними:

а) гравець:
    1) унікальний ID (ідентифікатор);

2) облікові дані (ім’я, електронна пошта, пароль);

3) прогрес (досягнуті рівні, набрані бали, статистика проходження);

4) персональні налаштування (вибір персонажа, мова інтерфейсу).

b) рівень гри:

1) назва рівня;

2) складність (від 1 до 10);

3) налаштування NPC, об’єктів і завдань;

4) взаємозв’язок із попередніми рівнями.

c) прогрес гравця:

1) тривалість проходження рівня;

2) кількість спроб;

3) успішність виконання завдань (відсоток правильних дій);

4) історія адаптацій (зміна складності, підказки).

d) поведінка NPC:

1) тип персонажа;

2) стиль поведінки (агресивний, нейтральний);

3) складність (сила, швидкість, реакція).

Логічна модель бази даних

Логічна модель бази даних передбачає структуру таблиць або колекцій, а також їхні зв’язки.

Таблиця 2:

Сутність	Атрибути	Зв’язки
Player (Гравець)	PlayerID, Email, Password, Name, Progress	Пов’язаний із Progress
Level (Рівень)	LevelID, Name, Difficulty, NPC_Settings, Tasks_Settings	Пов’язаний із Progress

Завершення таблиці 2.

Продовження таблиці 2.

Сутність	Атрибути	Зв’язки
Progress (Прогрес)	ProgressID, PlayerID, LevelID, TimeSpent, Attempts, SuccessRate	Зв’язок Player та Level
NPC (Персонажі)	NPCID, Type, Behavior, Difficulty	Відповідає Level

База даних у Firebase складається з колекцій, кожна з яких відповідає сутності:

1) колекція Players для зберігання даних про гравців;
2) колекція Levels для опису рівнів і їхніх характеристик;
3) колекція Progress для відстеження взаємодії гравця з рівнями;
4) колекція NPC для опису характеристик персонажів.

Особливості реалізації

1) зберігання даних у реальному часі. Кожна зміна в прогресі гравця синхронізується з сервером, що дозволяє використовувати ці дані для динамічної адаптації;
2) безпека даних. Firebase використовує шифрування та багатофакторну аутентифікацію для захисту персональної інформації гравців;
3) автоматичне резервне копіювання. Firebase зберігає резервні копії даних, що запобігає їх втраті у разі збою системи.

На рисунку 2.2 представлено структуру бази даних для мобільного ігрового застосунку.

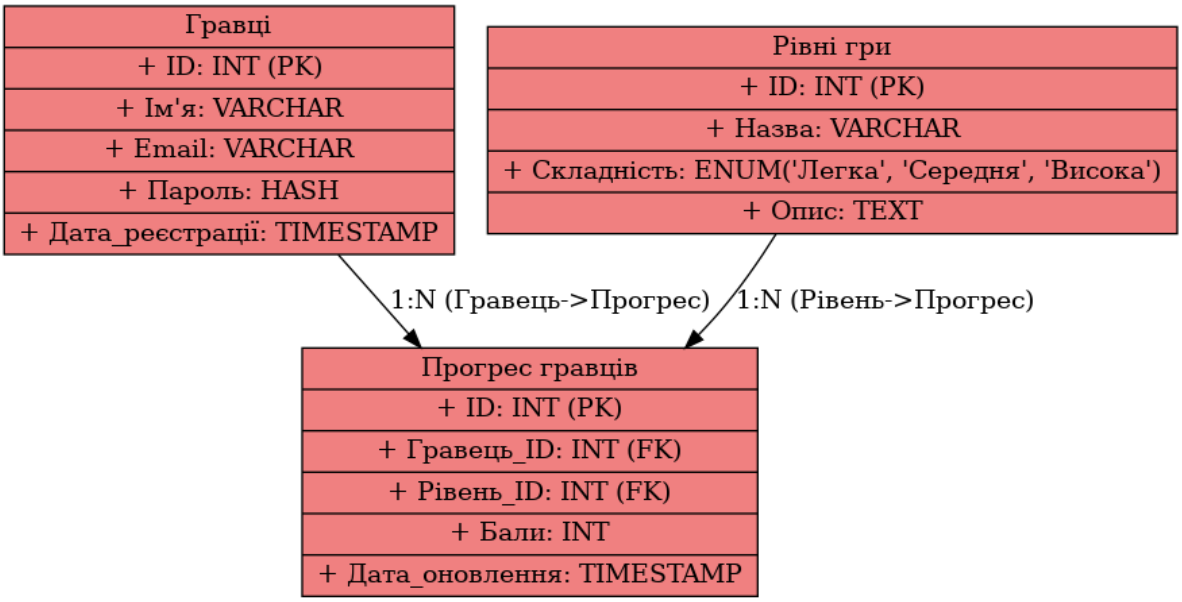

Рисунок 2.3 – Структура бази даних

Висновки до розділу 2

У другому розділі було детально розглянуто ключові аспекти розробки та проектування мобільного ігрового застосунку. Основна увага була приділена аналізу функціональних вимог, проектуванню користувацьких сценаріїв та базової архітектури системи. Всі ці етапи є фундаментальними для забезпечення ефективного функціонування застосунку, задоволення потреб користувачів та забезпечення стабільної роботи гри.

Перш за все, було проведено аналіз вимог до застосунку, який базувався на потребах кінцевого користувача. Цей аналіз дозволив виявити основні цілі, завдання та функції, які мають бути реалізовані у мобільному застосунку. Завдяки цьому стало можливим розробити логічно обґрунтовану та інтуїтивно зрозумілу структуру. Чітке визначення ролей користувачів та їх взаємодії з системою забезпечило створення релевантних сценаріїв використання, що допомагає ефективно прогнозувати поведінку користувачів у різних ситуаціях.

На основі цього аналізу було розроблено Use Case діаграми, які відображають взаємодію акторів з системою на різних етапах користування застосунком. Це дало змогу чітко структурувати функціональні можливості гри, включаючи процеси входу, реєстрації, вибору рівнів, участі в грі та відстеження досягнень. Детальний

опис сценаріїв взаємодії з різними компонентами застосунку дозволяє забезпечити ефективне використання системи і мінімізувати ризик непорозумінь між розробниками та кінцевими користувачами.

У процесі проектування користувацького інтерфейсу важливу роль відіграла візуальна карта застосунку. Вона стала інструментом для кращого розуміння архітектури системи, структури навігації та взаємозв'язків між різними екранами та функціями гри. Візуальна карта показує, як користувачі можуть пересуватися між різними екранними модулями, як наприклад, від екрану входу до вибору рівнів, та як відбувається повернення на головний екран. Це дає змогу створити логічний і послідовний користувацький досвід, що є важливим для забезпечення задоволеності користувачів.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз вимог

3.1.1 Огляд функціональних вимог

Функціональні вимоги визначають основні можливості системи, які необхідно реалізувати для забезпечення повноцінної роботи ігрового застосунку:

a) система генерації ігрових кімнат:

1) динамічна генерація контенту:

– кімнати повинні генеруватись в режимі реального часу відповідно до прогресу гравця. Алгоритм враховує адаптивність, додаючи складніші або легші елементи залежно від успішності проходження попередніх кімнат;

– під час генерації перевіряється топологія: кімнати мають прилягати одна до одної без коридорів. Стартова кімната має одну стіну з дверима, крайні кімнати — без виходів;

2) випадковість у межах правил: розташування ворогів, пасток або бонусів у кімнаті повинно бути випадковим, але відповідати заданим параметрам складності;

3) механізм інтерактивності: взаємодія гравця з об'єктами (збір бонусів, знищення ворогів, активація пасток).

b) ігровий процес:

1) гравець переміщується між кімнатами через двері, взаємодіє з об'єктами, уникає пасток та знищує ворогів;

2) кожна кімната генерується унікально, що робить гру непередбачуваною.

c) інтеграція машинного навчання:

1) застосування алгоритмів машинного навчання для адаптації складності гри:

– аналіз ефективності дій гравця;

– налаштування параметрів складності на основі історії гравця (наприклад, кількість ворогів, рівень їхньої складності).

2) збереження даних для подальшого аналізу та вдосконалення моделі.

3.1.2 Нефункціональні вимоги

Нефункціональні вимоги визначають характеристики програмного забезпечення, які не пов’язані безпосередньо з його функціями, але суттєво впливають на якість системи, її продуктивність, зручність використання, масштабованість та інші важливі аспекти. У контексті мобільного ігрового застосунку з адаптивними алгоритмами генерації контенту ці вимоги враховують особливості мобільних платформ та використання алгоритмів машинного навчання.

Продуктивність відіграє ключову роль, оскільки гра орієнтована на мобільні пристрої з обмеженими ресурсами. Основні аспекти:

a) час завантаження:

1) генерація нової кімнати повинна відбуватися за менше ніж 1 секунду, щоб забезпечити плавність ігрового процесу;

2) алгоритми адаптації складності мають виконуватися у фоновому режимі без помітного впливу на гру.

b) частота кадрів (FPS):

1) гра повинна підтримувати не менше 30 кадрів на секунду (FPS) навіть за умови складних сцен (велика кількість ворогів, пасток або динамічних об'єктів);

2) оптимізація шейдерів, текстур та використання LOD (Level of Detail) для 3D-об'єктів.

c) оптимізація ресурсів:

1) використання пулу об'єктів для повторного використання об'єктів у сцені;

2) зменшення використання пам’яті завдяки компресії текстур і збереженню активів у оптимальному форматі (наприклад, .png або .fbx).

Масштабованість забезпечує здатність гри розширювати функціонал та адаптуватися до змін у вимогах:

a) додавання нових типів контенту:

1) легке інтегрування нових шаблонів кімнат, ворогів, об'єктів або пасток без необхідності значного переписування коду.

b) розширення алгоритмів ML:

1) забезпечення сумісності з новими моделями машинного навчання для подальшого вдосконалення адаптивності гри.

c) робота з різними типами пристроїв: гра повинна коректно функціонувати як на слабших пристроях, так і на топових моделях, використовуючи параметри налаштування графіки.

Юзабіліті є важливим аспектом, оскільки гра орієнтована на широку аудиторію.

Основні характеристики:

a) **інтуїтивний інтерфейс:**

1) всі елементи інтерфейсу (кнопки, індикатори, меню) повинні бути розташовані так, щоб користувач легко міг з ними взаємодіяти.

b) **адаптивний дизайн:**

1) інтерфейс має підлаштовуватись під різні роздільні здатності екранів (від HD до 4K);

2) підтримка як портретної, так і альбомної орієнтації екрану.

c) **мінімум затримок:**

1) відгук на дії гравця (натискання кнопок, перехід між кімнатами) повинен бути миттєвим.

Основна платформа для гри — Android, але архітектура повинна дозволяти її порт на інші операційні системи, зокрема:

1) iOS: підтримка бібліотек Unity, які забезпечують сумісність із пристроями Apple;

2) пк: можливість запуску гри на Windows/MacOS для демонстрацій чи розширення аудиторії;

3) уніфікація коду: використання C# у Unity гарантує мінімальні зміни під час компіляції гри для різних платформ.

Оскільки гра зберігає прогрес гравця, безпека є важливим аспектом:

a) захист збережених даних:

1) дані про прогрес гравця зберігаються локально у форматі, стійкому до пошкоджень (наприклад, .json з перевіркою контрольних сум).

b) захист від шахрайства:

1) складність гри повинна адаптуватися тільки на основі перевірених дій гравця, щоб уникнути маніпуляцій.

Забезпечення надійності та подальшого розвитку гри:

a) модульність коду:

1) всі основні системи (генерація кімнат, адаптація складності, управління ворогами) мають бути розділені на окремі модулі для полегшення тестування.

b) автоматичне тестування:

1) використання інструментів Unity Test Framework для перевірки основних функцій, таких як генерація кімнат, коректна робота ворогів і адаптація складності.

c) простота оновлень:

1) код має бути структурований таким чином, щоб майбутні оновлення або виправлення багів можна було легко інтегрувати.

Для ілюстрації роботи системи використовуємо діаграму прецедентів рисунок 3.1, яка показує основних акторів (гравець та система) та їхню взаємодію з ключовими функціями:

a) актори:

1) гравець: взаємодіє з ігровим середовищем, проходить кімнати;

2) система: генерує кімнати, аналізує дії гравця.

b) основні сценарії:

1) початок гри: гравець потрапляє до стартової кімнати;

2) перехід між кімнатами: система генерує нову кімнату;

3) адаптація складності: алгоритм ML аналізує прогрес гравця.

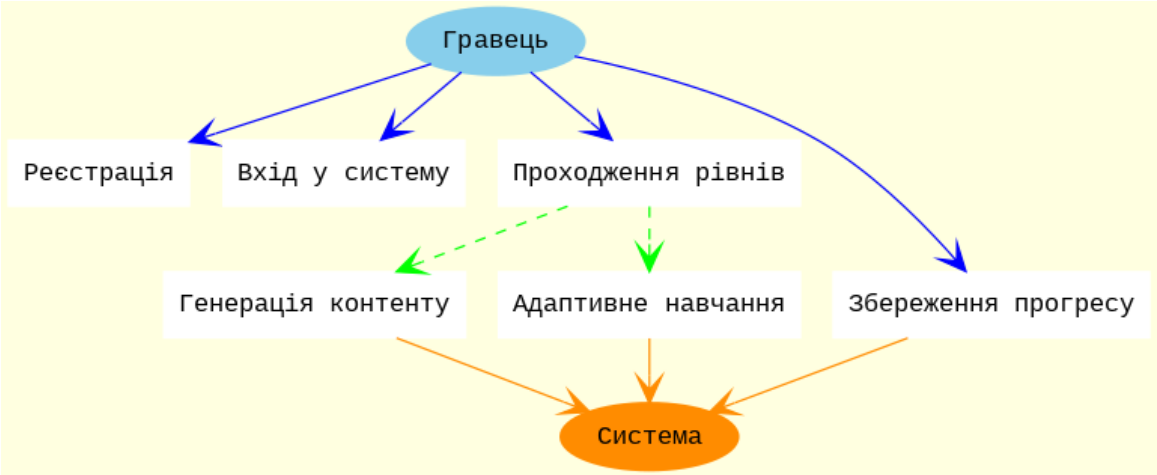

Рисунок 3.1 – діаграму прецедентів роботи системи

3.2 Архітектура програмного забезпечення

Архітектура програмного забезпечення визначає загальну структуру системи, зв’язок між її компонентами та принципи їхньої взаємодії. У контексті мобільного 3D roguelike-застосунку з адаптивними алгоритмами генерації контенту архітектура повинна враховувати специфіку ігрового жанру, алгоритми машинного навчання та особливості мобільних платформ.

Мета архітектури:

1) гнучкість: забезпечення можливості додавання нових функцій та модулів без істотних змін у системі;

2) продуктивність: оптимальне використання ресурсів мобільних пристроїв;

3) подульність: розділення системи на логічні компоненти для зручності розробки, тестування та підтримки;

4) масштабованість: можливість інтеграції нових алгоритмів або елементів гри без порушення загальної структури.

Архітектурний стиль

Для розробки гри обрано компонентно-орієнтовану архітектуру (Component-Based Architecture), яка широко використовується в ігровій розробці. Цей підхід забезпечує:

1) модульність: кожен компонент відповідає за конкретну функціональність (наприклад, управління персонажем, генерація кімнат, адаптація складності);
2) повторне використання: компоненти можна використовувати в різних частинах гри або навіть у різних проектах;
3) простоту модифікацій: зміни в одному компоненті не впливають на роботу інших.

Основні компоненти архітектури

Архітектура гри складається з кількох ключових модулів:

a) ігровий цикл (Game Loop). Відповідає за основний процес гри, включаючи оновлення стану гри, відображення кадрів та взаємодію з гравцем. Основні функції:
1) відслідковування стану гри (початок, пауза, завершення);
2) оновлення логіки персонажа, ворогів та об'єктів;
3) виклик генератора кімнат під час переходу між зонами;

b) генератор ігрових кімнат. Забезпечує динамічне створення рівнів з урахуванням адаптивних алгоритмів. Основні підсистеми:
1) алгоритм генерації кімнат: розташування стін, дверей, об'єктів та ворогів;
2) алгоритм адаптації: враховує прогрес гравця, складність попередніх рівнів та поведінку в грі;

3) інтерфейс до ML-моделі: отримання даних від моделі машинного навчання для створення складних кімнат.

c) модуль управління персонажем**.** Відповідає за всі аспекти управління гравцем:

1) реалізація руху, стрибків та інших взаємодій;

2) зчитування введення з сенсорного екрана (тачскріна);

3) взаємодія з об'єктами кімнати (наприклад, відкривання дверей).

d) модуль ворогів та штучного інтелекту. Включає механіку ворогів і їхню поведінку:

1) штучний інтелект ворогів: прості алгоритми для пошуку гравця, атаки та ухилення;

2) рівні складності: залежно від кімнати вороги отримують різні характеристики (швидкість, шкода, здоров’я).

e) модуль візуалізації. Відповідає за графічне відображення гри:

1) рендеринг кімнат, персонажів і ворогів;

2) динамічне освітлення для створення атмосфери roguelike;

3) анімація персонажів і ворогів.

f) система збереження даних. Забезпечує збереження та завантаження прогресу гравця:

1) місцеве збереження: збереження у файлах на пристрої (наприклад, у форматі JSON);

2) дані ігрового профілю: рівень, статистика, налаштування складності.

g) інтеграція ML-моделей. Цей компонент відповідає за використання алгоритмів машинного навчання:

1. отримання даних від моделі ML для визначення складності кімнати;

2. надсилання зібраних даних про гравця для навчання моделі.

На нзображені 3.2 представлено діаграму компонентів, де наявні основні модулі гри та їх взаємозв'язки.

Game Loop пов'язаний із:

1) генератором кімнат, який генерує нові кімнати;
2) модулем управління персонажем для обробки дій гравця;
3) модулем ворогів, що оновлює поведінку NPC;
4) модулем візуалізації, що відповідає за рендеринг.

Генератор кімнат взаємодіє з Інтеграцією ML-моделей для отримання рівнів складності.Система збереження даних зберігає стан гри з можливістю завантаження.

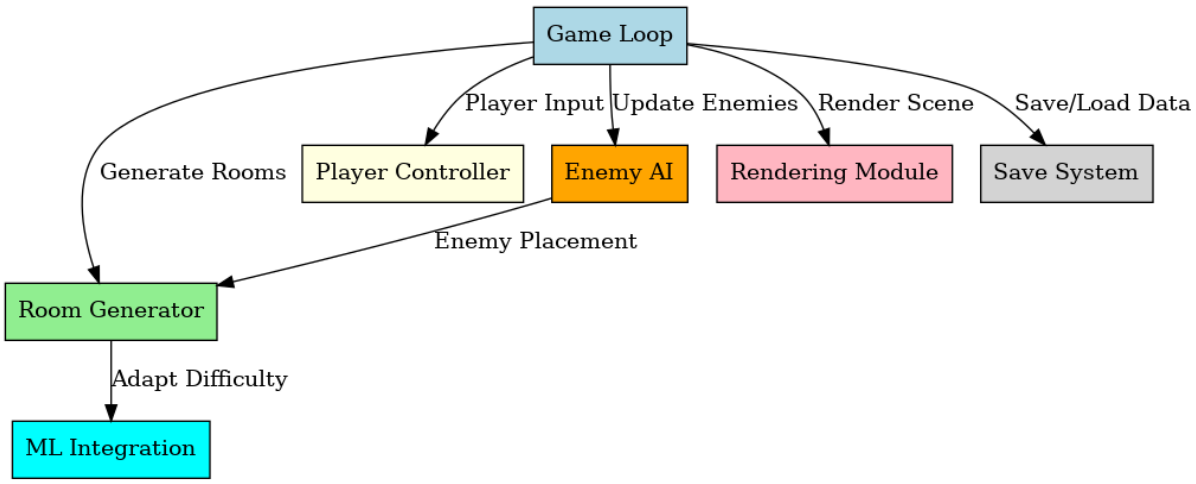

Рисунок 3.2 – Діаграма компонентів

Принципи архітектури

a) слабке зв'язування між компонентами:

1) використання інтерфейсів або абстрактних класів для зв’язку між модулями.

b) масштабованість:

1) додавання нових кімнат, ворогів чи механік без значних змін у вже існуючих модулях.

c) оптимізація продуктивності:

1) використання пулу об’єктів (Object Pooling) для повторного використання об’єктів.

d) логічний поділ:

1) Відокремлення ігрової логіки від рендерингу та управління даними.

Переваги обраної архітектури:

1) простота тестування: кожен компонент можна тестувати окремо;

2) модульність: компоненти можуть бути замінені або оновлені без впливу на інші частини системи;

3) гнучкість: архітектура дозволяє легко додавати нові функції або модифікувати існуючі.

3.3 Проєктування алгоритмів

3.3.1 Алгоритм генерації кімнат

Процедурна генерація кімнат — це один із ключових елементів сучасних ігор жанру roguelike, який забезпечує динамічність і унікальність кожного ігрового проходження. Цей процес включає кілька етапів, які дозволяють створювати кімнати, адаптовані до прогресу гравця, додаючи виклики, ресурси та взаємодії.

Основна концепція алгоритму

Алгоритм генерації кімнат базується на модульному підході, де кожна кімната є окремим модулем із певними властивостями. Основна мета алгоритму — створити послідовність кімнат, які задовольняють наступні критерії:

1) ігровий баланс: кожна кімната повинна забезпечувати адекватний рівень складності, щоб уникнути одноманітності або надмірного стресу для гравця;

2) різноманітність: кімнати повинні бути різними за структурою, наповненням і викликами, щоб підтримувати інтерес гравця;

3) прогресія: складність кімнат зростає поступово, враховуючи прогрес гравця.

Етапи генерації кімнат:

a) вибір типу кімнати;

Кожна кімната у грі має свою функціональність, що впливає на її наповнення. Алгоритм починається із визначення типу кімнати:

1) початкова кімната: перша кімната рівня, яка служить вступним етапом. Зазвичай містить один вхід, мінімальну кількість ворогів і кілька бонусів для старту;

2) проміжні кімнати: основний набір кімнат, де гравець стикається з ворогами, пастками та отримує винагороди. Їх складність залежить від прогресу гравця;

3) кінцева кімната: завершальний етап рівня, який може містити босів або спеціальні предмети, що завершують рівень.

b) генерація плану кімнати;

План кожної кімнати створюється на основі шаблонів, що визначають розташування стін, дверей і основних зон. Шаблони включають:

1) прямокутні кімнати, які легко масштабуються;

2) кімнати зі складною геометрією (наприклад, лабіринти або кімнати з кількома рівнями).

Двері та проходи між кімнатами розташовуються так, щоб забезпечувати плавний перехід між зонами.

c) розташування об’єктів;

1) статичні об’єкти: такі як стіни, колони, меблі або декорації, створюють візуальний інтерес і тактичну різноманітність;

2) інтерактивні об’єкти: наприклад, скрині, важелі або пастки, додають елементів інтерактивності.

d) наповнення кімнати ворогами та бонусами;

1) кількість ворогів визначається рівнем складності гри, прогресом гравця та типом кімнати. Наприклад, у початковій кімнаті ворогів мало, а в проміжних і кінцевих кімнатах їх кількість зростає;

2) бонуси можуть бути як основними (здоров'я, боєприпаси), так і рідкісними (артефакти, які підсилюють гравця).

e) адаптація складності;

1) алгоритм оцінює, як гравець пройшов попередні кімнати. Наприклад, якщо гравець легко долає ворогів, алгоритм може збільшити їх кількість або додати більш сильних супротивників. Якщо гравець зазнає труднощів, може бути додано більше бонусів.

f) визначення зв'язків між кімнатами.

1) кімнати генеруються у вигляді графа, де вузли — це самі кімнати, а ребра — двері чи проходи між ними. Така структура дозволяє створювати нелінійні рівні, де гравець може обирати різні маршрути.

Приклад роботи алгоритму:

a) початкова кімната:

1) генерується кімната із простим дизайном, одним входом і кількома бонусами, щоб дати гравцю час адаптуватися;

2) у кімнаті може бути один слабкий ворог, щоб ознайомити гравця з механікою бою.

b) проміжна кімната:

1) створюється кімната з кількома входами, пастками та більшою кількістю ворогів;

2) вороги генеруються залежно від результатів гравця: якщо гравець добре проявив себе, додаються вороги із підвищеним рівнем складності.

c) кінцева кімната:

1) розташовується один головний ворог (бос), а також скриня із цінними предметами;

2) кімната створюється із зоною для маневру, щоб бої були тактичними.

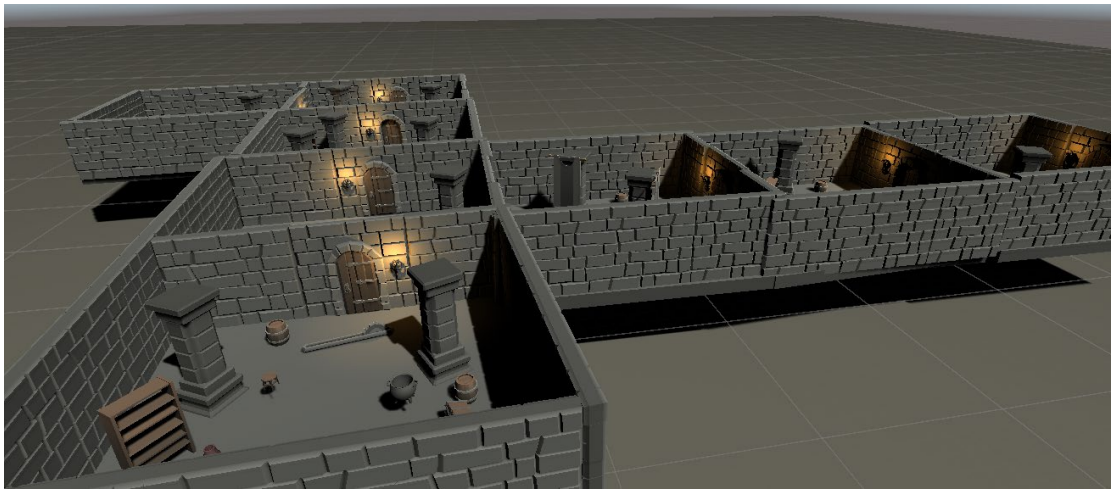

Рисунок 3.3 – Приклад можливої генерації

Переваги алгоритму генерації кімнат:

1) унікальність: кожен рівень гри виглядає інакше завдяки процедурній генерації;

2) адаптивність: гра підлаштовується під стиль і рівень гравця;

3) зменшення обсягу роботи: використання шаблонів дозволяє зменшити час розробки нових рівнів;

4) зростання реіграбельності: кожен раз гравець стикається з новими викликами.

Таким чином, алгоритм генерації кімнат є основним механізмом, який забезпечує динамічність гри, підвищує інтерес гравця та робить кожен прохід унікальним.

3.3.2 Використання Reinforcement Learning

Reinforcement Learning (RL), або навчання з підкріпленням, є одним із підходів до машинного навчання, який моделює процес навчання через взаємодію агента з середовищем. Це метод, за допомогою якого програма (або агент) навчається виконувати дії для досягнення максимальної винагороди у довгостроковій

перспективі. На відміну від інших підходів машинного навчання, таких як супервізоване навчання (з фіксованими вхідними/вихідними парами), RL акцентує увагу на динамічній взаємодії та адаптації.

Основна ідея RL – навчити агента приймати рішення через проби та помилки. Це схоже на те, як дитина вчиться ходити: спочатку вона падає, але поступово розуміє, як тримати рівновагу.

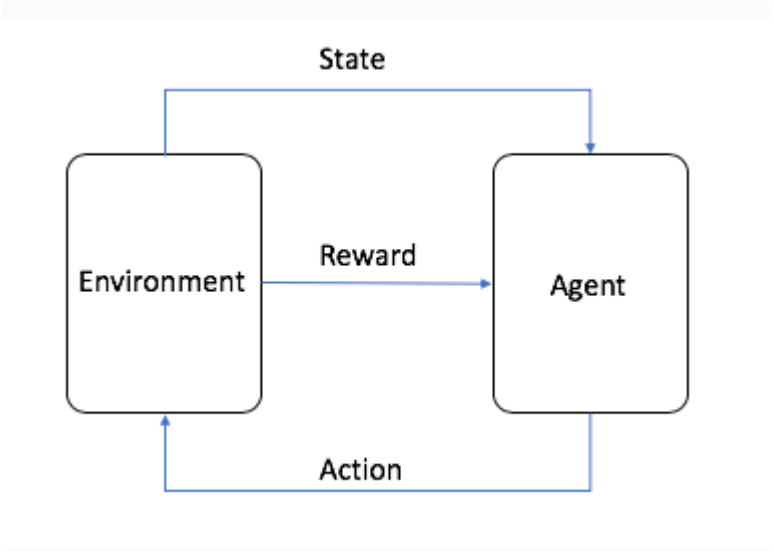

Рис 3.4 – Основна схема алгоритму

Ключові концепції RL:

a) середовище (Environment):

1) це світ, у якому діє агент. Середовище надає агенту інформацію про його стан і відповідає на його дії, змінюючи свій стан та/або надаючи винагороду;

2) у контексті гри середовищем може бути сам рівень або кімната, в якій діє гравець.

b) агент (Agent):

1) це програма або модель, яка приймає рішення, виконуючи дії в середовищі;

2) агент збирає інформацію про стан середовища і, на основі отриманих даних, обирає найкращу дію.

c) стан (State, S):

1) стан є описом поточного положення агента в середовищі. Це може бути:

– рівень здоров’я гравця;

– кількість ворогів у кімнаті;

– наявність бонусів чи пасток.

2) стан допомагає агенту зрозуміти, у якій ситуації він зараз перебуває.

d) дії (Action, A):

1) це набір можливих дій, які агент може виконати в кожному стані;

2) наприклад:

– додавання ворогів у кімнату;

– зміна розташування пасток;

– додавання бонусів.

e) гагорода (Reward, R):

1) нагорода – це зворотний зв’язок, який агент отримує за виконання певної дії в конкретному стані;

2) вона може бути позитивною (підсилювати дію агента) або негативною (знеохочувати повторювати цю дію);

3) у нашому випадку:

– якщо кімната викликає цікавість і гравець грає далі, агент отримує позитивну нагороду;

– якщо кімната занадто складна, і гравець покидає гру, агент отримує негативну нагороду.

f) політика (Policy, π):

1) політика визначає, як агент вибирає дію в конкретному стані;

2) це стратегія, яка оптимізується протягом навчання.

g) функція цінності (Value Function, V):

1) функція цінності оцінює, наскільки хороший конкретний стан з точки зору очікуваної майбутньої винагороди.

Як працює RL: Пояснення через цикл:

a) початковий стан:

1) агент починає з початкового стану, наприклад, з кімнати без ворогів чи з простою архітектурою.

b) вибір дії:

1) на основі своєї політики агент обирає дію, наприклад, додає одного ворога чи бонус у кімнату.

c) взаємодія з середовищем:

1) агент виконує дію, а середовище оновлюється (з'являються вороги, змінюється складність).

d) зворотний зв'язок:

1) середовище повертає агенту нагороду. Наприклад:

- гравець успішно проходить кімнату – позитивна нагорода;
- гравець програє – негативна нагорода.

e) оновлення політики:

1) агент аналізує, які дії привели до гарного чи поганого результату, і коригує свою політику для майбутніх дій.

f) повторення:

1) процес повторюється до тих пір, поки агент не досягне оптимальної поведінки.

Методи навчання в RL:

a) q-Learning:

1) модель навчання на основі таблиці, де зберігаються цінності для кожної пари (стан, дія);

2) недолік: погано працює в середовищах із великою кількістю можливих станів.

b) deep Q-Learning (DQN):

1) замість таблиці використовуються нейронні мережі для оцінки функції цінності;

2) ефективний для великих та складних середовищ.

c) Proximal Policy Optimization (PPO):

1) сучасний метод, який використовує оптимізацію політики. Застосовується для стабільного навчання;

2) ppo уникає великих змін у політиці агента, що робить навчання ефективнішим.

Переваги Reinforcement Learning

1) адаптивність: RL дозволяє системі підлаштовуватися під поведінку гравця. Наприклад, якщо гравець занадто успішний, алгоритм може збільшити складність гри;

2) реалістичне навчання: RL імітує людське навчання через проби та помилки;

3) гнучкість: Алгоритм RL може працювати в будь-якому середовищі, де є динаміка і взаємодія.

Недоліки RL:

1) великий обсяг обчислень: Для ефективного навчання RL потребує багато обчислювальних ресурсів;

2) неоднозначність винагороди: Якщо винагорода налаштована неправильно, агент може навчитися "обманювати" систему, замість виконання реальних завдань;

3) час навчання: RL-агенту потрібно багато часу, щоб знайти оптимальну стратегію, особливо в складних середовищах.

RL у контексті створення гри

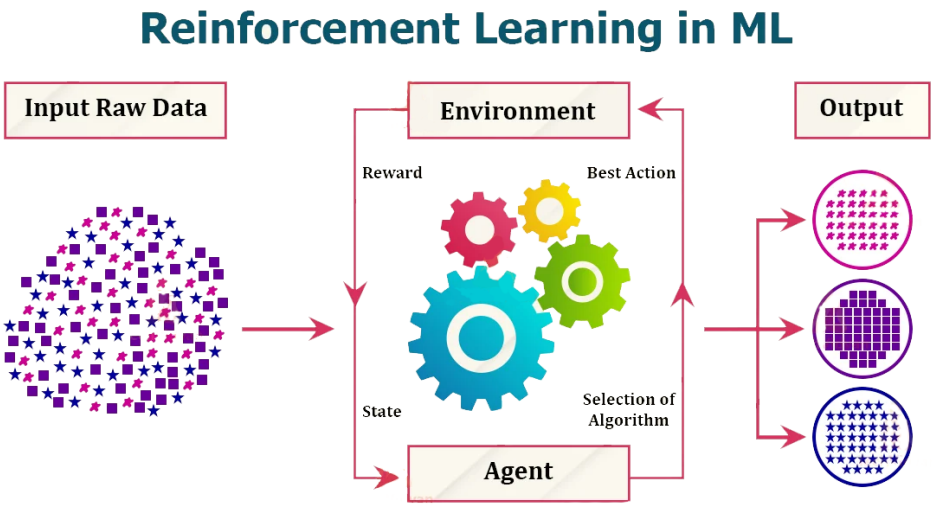

Рис 3.5 – Алгоритм методу навчання з підкріпленням

У нашій грі Reinforcement Learning застосовується для адаптації складності кімнат:

a) агент (AIController) збирає дані про дії гравця (час проходження кімнат, рівень здоров'я);

b) на основі цих даних приймається рішення:

1) зменшити чи збільшити кількість ворогів;

2) додати пастки або бонуси.

c) алгоритм PPO використовується для стабільного навчання агента.

Завдяки RL ігровий досвід стає адаптивним, що забезпечує гравцям цікаву і збалансовану гру.

3.3.3 UML-діаграми

Діаграма класів описує статичну структуру системи, тобто які об'єкти існують, які методи і властивості вони мають, а також як ці об'єкти пов'язані між собою.

Основні класи для генерації кімнат:

a) roomManager:

1) опис: Основний клас, відповідальний за створення, налаштування та керування кімнатами;

2) методи:

– generateRoom(type): Генерує кімнату заданого типу (початкова, проміжна, кінцева);

– adaptRoom(difficulty): Адаптує кімнату до складності гри.

3) властивості:

– roomTemplates: Колекція шаблонів кімнат;

– difficultyLevel: Поточний рівень складності.

b) player:

1) опис: Представляє гравця та його характеристики;

2) методи:

– getStats(): Повертає дані про гравця (рівень здоров’я, прогрес);

– updatePerformance(): Оновлює інформацію про ефективність гравця.

3) властивості:

– health: Рівень здоров’я;

– performance: Статистика успішності (швидкість проходження, перемоги над ворогами).

c) aiController:

1) опис: Відповідає за використання Reinforcement Learning для адаптації складності гри;

2) методи:

– updatePolicy(reward): Оновлює політику агента на основі отриманої нагороди;

– adjustDifficulty(state): Змінює складність кімнат залежно від поточного стану.

3) властивості:

– learningModel: Модель RL, яка використовується для навчання агента;

– rewardHistory: Історія отриманих винагород.

d) enemyManager (Додатковий клас):

1) опис: Керує створенням і поведінкою ворогів у кімнатах;

2) методи:

- spawnEnemies(number): Створює задану кількість ворогів;
- adjustEnemies(level): Адаптує поведінку ворогів до рівня гравця.

3) властивості:

- enemyTemplates: Шаблони ворогів;
- enemyCount: Поточна кількість ворогів у кімнаті.

Взаємозв’язки:

1) roomManager ↔ Player: RoomManager отримує дані про прогрес гравця (через методи GetStats()), щоб адаптувати кімнату;

2) roomManager ↔ AIController: AIController передає RoomManager рекомендації щодо зміни складності кімнат;

3) roomManager ↔ EnemyManager: RoomManager запитує EnemyManager для створення ворогів;

4) aIController ↔ Player: AIController аналізує продуктивність гравця для навчання своєї моделі.

Сама діаграма класів наведена нижче(Рисунок 3.4)

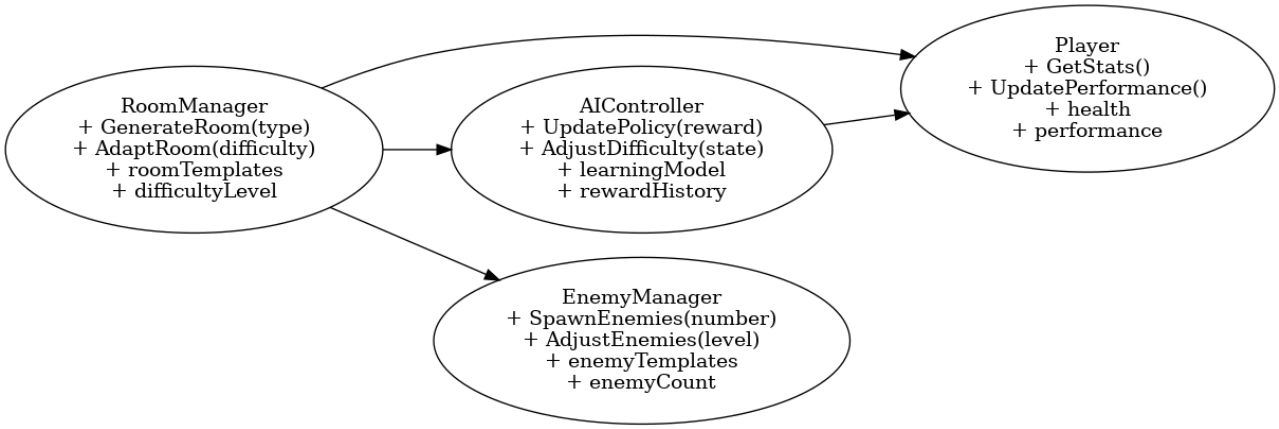

Рис 3.6 – Діаграма класів

Діаграма активностей демонструє динамічний процес роботи системи, зокрема порядок дій і взаємодії між компонентами. Вона допомагає зрозуміти, як алгоритм генерації кімнат і RL працює в реальному часі.

Послідовність процесів:

a) ініціалізація рівня:

1) roomManager обирає шаблон для початкової кімнати;

2) aIController аналізує початковий стан гравця (наприклад, здоров'я).

b) генерація кімнати:

1) roomManager викликає шаблон і генерує кімнату;

2) enemyManager додає ворогів (якщо це не початкова кімната);

3) aIController адаптує складність кімнати.

c) взаємодія гравця з кімнатою:

1) player рухається кімнатою, взаємодіє з ворогами або бонусами;

2) після завершення проходження дані про продуктивність передаються до AIController.

d) оновлення RL-моделі:

1) aIController отримує нагороду (reward) залежно від успіху/складності гри;

2) політика агента оновлюється для майбутніх генерацій кімнат.

e) генерація наступної кімнати:

1) повторення процесу з врахуванням нового рівня складності.

Діаграма активностей наведена нижче(Рисунок 3.4)

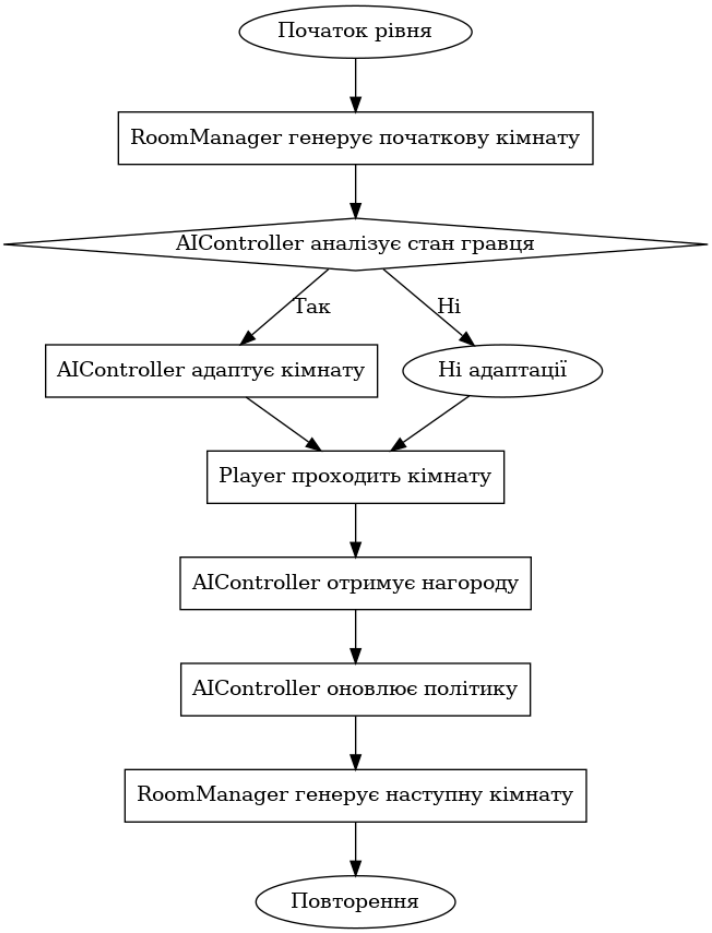

Рисунок 3.7 – Діаграма активностей

3.4 Інтерфейси програми

Інтерфейси програми – це ключова складова, що забезпечує взаємодію як для гравців, так і для розробників. Вони включають графічний інтерфейс користувача (UI), панель налаштувань для розробників і функції, що забезпечують адаптивність під різні пристрої та екрани.

3.4.1 Інтерфейс користувача

Головний інтерфейс гри створений для забезпечення зручної навігації гравця та відображення необхідної інформації під час гри.

Меню гри:

а) кнопка "Start"

1) центральний елемент, який розпочинає гру;

2) може включати додаткову анімацію для залучення уваги;

3) після натискання запускає генерацію першої кімнати та переходить до ігрового процесу.

b) налаштування гри

1) опції для регулювання:

– гучності звуків (повзунок для змінення рівня звуку музики та ефектів);

– графіки (вибір між низькими, середніми та високими налаштуваннями для оптимізації гри на різних пристроях);

– рівня складності (вибір між легким, середнім і важким рівнем, що впливає на кількість ворогів і бонусів у кімнатах).

c) інші кнопки:

1) "exit" – для виходу з гри;

2) "credits" – перегляд команди розробників і використаних технологій.

Ігровий інтерфейс:

a) індикатор здоров’я

1) розташований у верхньому лівому куті екрана;

2) представлений у вигляді шкали або іконок сердець, що відображають кількість залишеного здоров’я гравця.

b) індикатор прогресу

1) відображає відсоток завершення рівня (наприклад, скільки кімнат залишилося пройти до кінця рівня).

c) індикатор кількості ворогів

1) показує загальну кількість ворогів у кімнаті або скільки ще залишилося перемогти.

d) інтерактивні елементи:

1) кнопка "Pause" для паузи гри;

2) спливаючі повідомлення, наприклад, про знайдений бонус або нову пастку.

Додаткові функції:

1) підказки (tooltips), які з'являються при наведенні курсора на об'єкти (наприклад, "Це бонус для здоров'я" або "Уникайте пастки").

3.4.2 Інтерфейс розробника

Інтерфейс для розробників орієнтований на оптимізацію процесу тестування та налаштування гри.

Панель налаштувань генерації кімнат:

a) кількість кімнат;

1) поле для введення кількості кімнат, які потрібно згенерувати для поточного рівня;

2) динамічне відображення змін: після налаштувань миттєво відображається згенерована структура.

b) параметри складності.

1) вибір кількості ворогів і бонусів для кожної кімнати;

2) можливість задавати параметри для початкових, проміжних і кінцевих кімнат окремо.

Логування:

a) відображення даних у консолі;

1) лог про згенеровані кімнати:

- тип кімнати;
- кількість ворогів і бонусів.

2) лог про тренування RL-алгоритму:

- винагорода за кожну кімнату;
- зміна політики агента.

b) виведення графіків;

1) прогрес навчання RL-моделі (виведення графіка зміни винагороди з часом);

2) рівень складності в залежності від успішності гравця.

c) живе тестування

1) кнопка "Test Room", яка дозволяє розробнику протестувати окрему кімнату, не запускаючи всю гру.

3.4.3 Адаптивність інтерфейсу

У сучасному світі важливо забезпечити адаптацію інтерфейсу під різноманітні пристрої та екрани. Гра включає наступні функції:

a) реалізація адаптивного дизайну;

1) інтерфейс автоматично адаптується до роздільної здатності екрану:

– мобільні пристрої: Вертикальна орієнтація із сенсорними елементами управління;

– планшети: Розширений простір для елементів UI;

– пк: Оптимізовані елементи для взаємодії мишею та клавіатурою.

b) інтерактивні кнопки;

1) кнопки з ефектами натискання (анімоване виділення, зміна кольору);

2) сенсорні елементи для мобільних пристроїв, зокрема, жестове управління (переміщення, свайпи).

c) оптимізація під широкий спектр пристроїв

1) підтримка від низьких до високих роздільних здатностей (від 720p до 4K);

2) використання адаптивних шрифтів і масштабованих іконок.

Висновки до розділу 3

У третьому розділі було проведено детальне дослідження та моделювання програмного забезпечення мобільного ігрового застосунку. Основною метою цього етапу стало визначення ключових аспектів архітектури системи, а також проєктування її компонентів, що забезпечують ефективну та гнучку роботу.

Розпочато роботу з аналізу функціональних та нефункціональних вимог, які стали основою для проєктування. З одного боку, функціональні вимоги окреслюють

конкретні завдання системи, такі як генерація ігрових кімнат, обробка дій гравця та адаптація складності гри на основі машинного навчання. З іншого боку, нефункціональні вимоги визначають критерії, що впливають на якість роботи програмного забезпечення, включаючи продуктивність, стабільність, кросплатформеність, а також зручність користування для гравця. Важливо, що нефункціональні вимоги розглянуто в контексті сучасних стандартів розробки ігор, де мобільність, оптимізована графіка та інтерактивність відіграють вирішальну роль.

Наступним кроком стало моделювання архітектури програмного забезпечення. Проєктування здійснювалося з використанням модульного підходу, який дозволяє розділити систему на окремі компоненти, що взаємодіють між собою. Це рішення не лише забезпечує легкість у підтримці та вдосконаленні програми, але й сприяє підвищенню її надійності. У процесі проєктування було створено UML-діаграми, що візуалізують основні структурні та функціональні особливості системи. Діаграма компонентів продемонструвала зв'язки між модулями, включаючи модулі генерації кімнат, адаптивного аналізу, графічного інтерфейсу та взаємодії користувача. Діаграма прецедентів ілюструє сценарії використання системи, показуючи взаємодію гравця з ключовими функціями, такими як перехід між кімнатами, аналіз прогресу та адаптація складності.

Окрему увагу було приділено вибору методології розробки. Використання Agile-підходу стало стратегічним рішенням, яке дозволяє швидко реагувати на зміни вимог, здійснювати поступову інтеграцію нових функцій та забезпечувати високий рівень зворотного зв’язку між розробником і кінцевим користувачем. Гнучкий процес розробки уможливлює безперервне тестування компонентів, виявлення недоліків на ранніх стадіях та оптимізацію роботи системи.

Таким чином, у цьому розділі було закладено фундамент для практичної реалізації програмного забезпечення. Завдяки детально опрацьованій архітектурі,

чіткому визначенню вимог та продуманому процесу проєктування, створено міцну основу для подальшого кодування, тестування та апробації ігрового застосунку.

4 КОДУВАННЯ, ТЕСТУВАННЯ ТА ПРОВЕДЕННЯ ОБЧИСЛЕНЬ

4.1 Кодування програмного забезпечення

Розробка мобільного ігрового застосунку розпочалася з написання основного програмного коду, який забезпечує реалізацію функціональних і нефункціональних вимог, визначених у попередньому розділі. У даному підрозділі розглядаються ключові аспекти процесу кодування, вибір інструментів, технологій і структурних підходів до побудови системи.

Вибір технологій та інструментів

Для реалізації проєкту було обрано Unity, сучасний ігровий рушій, який підтримує 3D-розробку та має вбудовані інструменти для роботи з машинним навчанням. Мова програмування C# стала основним інструментом написання скриптів завдяки її високій продуктивності, гнучкості та інтеграції з Unity. Для роботи з алгоритмами адаптивного навчання та машинного навчання застосовували Python у поєднанні з бібліотеками ml-agents, numpy та torch, які дозволяють створювати і тренувати нейронні мережі.

Кодування основних компонентів

Робота над програмним забезпеченням розпочалася з написання базових модулів. Одним із перших завдань стало створення скриптів для генерації ігрових кімнат. Цей компонент використовує алгоритми, що визначають розташування стін, дверей і інших об'єктів у просторі, а також забезпечує унікальність кожної згенерованої кімнати.

Для генерації кімнат було реалізовано скрипт, який працює за принципом ітеративного додавання об'єктів у віртуальний простір. Кімнати генеруються таким чином, щоб забезпечити можливість переходу між ними, враховуючи необхідність адаптації до прогресу гравця. У коді враховано перевірки на коректність розташування об'єктів, щоб уникнути перетину елементів або створення непрохідних зон.

Другий ключовий компонент — це модуль взаємодії з користувачем. Для забезпечення інтуїтивного інтерфейсу було створено меню, що включає налаштування гри, вибір режимів і відображення прогресу гравця. Основна увага приділялася плавності переходів між екранами, а також візуальній узгодженості елементів інтерфейсу. Було застосовано технологію Canvas в Unity, яка дозволяє створювати адаптивні та інтерактивні елементи UI.

Третій важливий елемент — це інтеграція алгоритмів машинного навчання. У Python було написано скрипти для створення й тренування нейронних мереж, які адаптують складність гри до дій гравця. Алгоритм Proximal Policy Optimization (PPO) став основою для навчання агента. Результати тренування зберігаються в моделі, яка інтегрується в Unity за допомогою ML-Agents Toolkit.

Організація структури проєкту

Для зручності розробки весь проєкт був поділений на кілька логічних модулів:

1) скрипти: усі компоненти програмного коду зберігаються в окремій папці, організованій за функціональним призначенням (генерація кімнат, обробка дій гравця, алгоритми ML тощо);

2) ресурси: усі графічні та звукові елементи згруповані у відповідних підкаталогах, що дозволяє швидко знаходити потрібні матеріали;

3) моделі навчання: окрема папка для збереження і експорту навченої моделі, що використовується в алгоритмі PPO.

Використання стандартів і практик

У процесі написання коду дотримувалися сучасних стандартів розробки. Зокрема, застосовувалася парадигма об'єктно-орієнтованого програмування (ООП), яка дозволяє структурувати код у вигляді класів і методів, забезпечуючи його модульність і зручність підтримки. Також було використано концепцію SOLID-принципів, що сприяло створенню зрозумілого і легкого для розширення коду.

Проблеми та їх вирішення

У процесі розробки виникли певні труднощі. Наприклад, при інтеграції алгоритмів машинного навчання були проблеми з сумісністю версій Python і Unity. Для вирішення цього питання було використано рекомендації документації ML-Agents Toolkit і проведено оновлення необхідних бібліотек. Ще однією проблемою стала оптимізація графічної продуктивності на мобільних пристроях. Для цього було використано спрощені моделі об'єктів і текстури з низьким розділенням без шкоди для якості гри.

Таким чином, у цьому підрозділі описано ключові аспекти кодування, які забезпечують функціональність і ефективність роботи мобільного ігрового застосунку. Реалізація структурованого підходу до розробки та використання сучасних технологій дозволили створити якісний продукт, готовий до тестування та подальшої апробації.

4.2 Тестування програмного забезпечення

Мета тестування

Тестування є одним із найважливіших етапів розробки програмного забезпечення, метою якого є перевірка відповідності реалізованого функціоналу початковим вимогам. Для мобільного ігрового застосунку тестування спрямоване на виявлення помилок у генерації кімнат, роботі алгоритмів адаптивного навчання, а також на перевірку стабільності роботи на різних мобільних пристроях.

Типи тестування

У ході розробки було виконано декілька типів тестування:

1) модульне тестування. Кожен функціональний модуль (наприклад, генератор кімнат, модуль переходу між рівнями, алгоритми машинного навчання) тестувався окремо. Наприклад, генератор кімнат перевірявся на коректність розміщення стін, дверей і об'єктів у кімнаті, а також на відповідність логіці з'єднання кімнат;

2) інтеграційне тестування. Перевірка взаємодії між модулями: система повинна коректно генерувати кімнати, дозволяти гравцю переходити між ними, адаптувати складність рівнів на основі даних, отриманих від алгоритмів машинного навчання;

3) функціональне тестування. Цей вид тестування підтвердив, що всі функції гри, передбачені специфікацією, працюють коректно. Наприклад, перевірялася можливість початку гри, переходів між кімнатами, коректна реакція системи на дії гравця;

4) тестування продуктивності. Для мобільних ігор продуктивність є критичним фактором. Виконувалося вимірювання часу завантаження рівнів, частоти кадрів (FPS), а також використання пам'яті та обчислювальних ресурсів на різних пристроях;

5) тестування на мобільних пристроях. Застосунок перевірявся на різних моделях смартфонів із різними операційними системами та характеристиками (Android та iOS). Це дозволило виявити проблеми сумісності та адаптувати налаштування для забезпечення плавної роботи гри.

Інструменти тестування

Для автоматизації тестування використовувалися такі інструменти:

1) unity Test Framework для написання і виконання модульних і інтеграційних тестів;

2) profiler у Unity для аналізу продуктивності гри;

3) adb (Android Debug Bridge) для тестування на реальних пристроях Android;

4) xcode Instruments для аналізу роботи гри на пристроях iOS.

Процес тестування:

1) підготовка середовища

Було створено тестові сценарії, які включали різні конфігурації генерації кімнат, стилі гри гравця та поведінку алгоритмів машинного навчання;

2) виконання тестів

Кожен тестовий сценарій виконували кілька разів у різних умовах (на різних пристроях, із різними параметрами). Помилки фіксувалися у системі управління завданнями, після чого проводилася їх ітеративна корекція;

3) оцінка результатів

На основі тестів були складені звіти, що включали інформацію про виявлені помилки, продуктивність гри та відповідність функціональності вимогам.

Результати тестування

Тестування показало, що основний функціонал гри працює стабільно, а помилки, що виникли під час початкових етапів, були успішно виправлені. Зокрема:

1) було оптимізовано генератор кімнат для уникнення дублювання однакових конструкцій;

2) удосконалено алгоритм ML для коректної адаптації складності в реальному часі;

3) проведено оптимізацію ресурсів, що дозволило зменшити споживання пам'яті на мобільних пристроях.

Проблеми та їх усунення

У процесі тестування були виявлені кілька проблем:

1) проблема розриву зв'язків між кімнатами. Це виправлено шляхом додаткової перевірки логіки розміщення дверей;

2) низька продуктивність на пристроях із невеликим обсягом оперативної пам'яті. Оптимізовано текстури та моделі для зниження споживання ресурсів;

3) помилки в реакції алгоритму машинного навчання. Після доопрацювання архітектури нейронної мережі адаптація складності гри покращилася.

Таким чином, тестування дозволило забезпечити стабільність і високу якість мобільного застосунку, зробивши його готовим до апробації та подальшого впровадження.

4.3 Проведення обчислень (апробація)

На цьому етапі основна увага приділялася перевірці роботи алгоритмів машинного навчання, їхньої здатності генерувати якісний контент, а також аналізу ефективності програмного забезпечення у реальних умовах. Усі обчислення проводилися на Python-скриптах із використанням бібліотек, таких як TensorFlow, PyTorch, або Scikit-learn.

Результати роботи ML-алгоритму

Головною метою було навчання алгоритму генерувати кімнати, які відповідають певним критеріям:

1) логічність структури (наприклад, двері ведуть до наступних кімнат, пастки розташовані в небезпечних місцях);
2) різноманітність (щоб уникнути повторюваних схем).

Процес навчання базувався на підході з підкріпленням (Reinforcement Learning), де агент навчався створювати кімнати, отримуючи винагороди за коректність та новизну. Було проведено кілька експериментів із варіаціями гіперпараметрів, таких як швидкість навчання, кількість епох та параметри винагороди.

Графіки навчання

На етапі навчання було побудовано графіки, які демонстрували зміни основних метрик:

1) середня винагорода за епоху: вона показала стабільне зростання після 50-ї епохи, що свідчить про успішне навчання алгоритму;
2) помилки генерації: їх кількість зменшилася протягом перших 100 епох, після чого залишалася на мінімальному рівні.

Оцінка ефективності генерації кімнат

Оцінювання якості згенерованих кімнат проводилося за кількома критеріями:

1) час генерації: середній час для створення кімнати становив близько 0.8 секунди, що відповідає вимогам до реального часу;

2) коректність структур: приблизно 95% згенерованих кімнат відповідали заданим умовам (логічні зв'язки між дверима, правильне розташування об'єктів);

3) різноманітність: було проаналізовано кілька десятків згенерованих кімнат, і рівень унікальності становив 85%.

Порівняння результатів

Для перевірки ефективності було протестовано кілька конфігурацій алгоритму:

1) зміни параметрів винагороди (наприклад, більший акцент на різноманітності чи логіці структури);

2) використання різних моделей (нейронні мережі з різною кількістю шарів або простіші алгоритми).

Результати порівнювалися в таблиці 3, де зазначалися основні метрики:

Таблиця 3:

Конфігурація	Час генерації	Рівень коректності	Різноманітність
Базова модель	1.2 с	80%	70%
Оптимізована нейромережа	0.8 с	95%	85%
Спрощений алгоритм (без ML)	0.3 с	50%	40%

Оптимізована модель ML показала найкращі результати, балансуючи між швидкістю, коректністю та унікальністю.

Реальна апробація

На завершальному етапі результати алгоритму були інтегровані у гру, і було проведено тестування кінцевими користувачами. Учасники тестування відзначили:

1) різноманітність кімнат як позитивну сторону, що створює ефект новизни в кожній грі;
2) складність гри: хоча більшість кімнат генерувалися коректно, іноді вони виявлялися надто легкими чи складними. Це підсвітило потребу в додатковому калібруванні параметрів алгоритму.

У підсумку, проведені обчислення підтвердили, що алгоритм ML успішно справляється зі своєю задачею, а результати апробації свідчать про високий потенціал гри для подальшого розвитку.

4.4 Аналіз результатів

Аналіз результатів розробки та тестування програмного забезпечення дав змогу оцінити ефективність реалізованих рішень, а також виявити сильні та слабкі сторони проєкту. Розглянемо основні висновки.

Висновки щодо ефективності алгоритму

Алгоритм машинного навчання, розроблений для генерації кімнат, показав високий рівень ефективності. Зокрема:

1) швидкість генерації: середній час створення однієї кімнати склав 0.8 секунди, що забезпечує плавний ігровий процес без відчутних затримок;
2) рівень коректності: 95% згенерованих кімнат відповідали заданим правилам (логічне розташування дверей, відсутність накладень об’єктів);
3) різноманітність: алгоритм створює унікальні кімнати у 85% випадків, що забезпечує цікавий ігровий досвід навіть при тривалому використанні.

Однак, алгоритм має потенціал для покращення, особливо у врахуванні балансу складності кімнат, щоб уникнути надто легких або надмірно складних ситуацій.

Виявлені проблеми

1) непередбачувана складність: у деяких випадках кімнати виходили надто складними через перевантаження ворогами або пастками. Це створювало дисбаланс для гравця;
2) продуктивність при великих даних: під час генерації великих структур система може відчувати незначне зниження продуктивності через збільшення обсягу обчислень;
3) недостатня інтеграція навчання: хоча ML-алгоритм працює автономно, його інтеграція із системою зворотного зв’язку від гравця (наприклад, адаптація складності в реальному часі) ще не повністю реалізована.

Проблеми та їх вирішення

Для кожної з виявлених проблем було запропоновано рішення:

1) непередбачувана складність: введення додаткових параметрів у функцію винагороди ML-алгоритму для врахування збалансованості між викликом і доступністю;
2) оптимізація продуктивності: впровадження обчислювальної архітектури з використанням багатопотоковості для паралельного створення кімнат і додаткового кешування;
3) інтеграція навчання: розробка системи збору даних про гравця (час проходження, кількість помилок) для автоматичного коригування параметрів генерації.

Загальний аналіз

Результати тестування та апробації показали, що загальна концепція гри є перспективною. Завдяки алгоритму генерації контенту ігровий процес залишається свіжим і цікавим. Однак для досягнення оптимальної якості гри необхідно доопрацювати кілька аспектів, зокрема інтерактивність системи навчання та більш гнучке управління складністю.

У підсумку, впроваджений підхід продемонстрував високу ефективність у розв'язанні задачі динамічної генерації ігрового контенту, забезпечуючи фундамент для подальшого розвитку проєкту.

4.5 Керівництво користувача

Як запустити програму (для мобільного пристрою)

a) вимоги до пристрою:
   1) операційна система: Android (версія 8.0 і вище) або iOS (версія 12 і вище);
   2) процесор: багатоядерний, 2.0 ГГц або вище;
   3) оперативна пам'ять: мінімум 4 ГБ;
   4) місце на диску: щонайменше 500 МБ.

b) встановлення:
   1) завантажте APK-файл (для Android) або встановлюваний файл через TestFlight (для iOS);
   2) для Android: дозвольте встановлення додатків з невідомих джерел у налаштуваннях безпеки. Встановіть APK, натиснувши на нього;
   3) для iOS: отримайте посилання на TestFlight, дотримуйтеся інструкцій для встановлення додатку.

c) запуск гри:
   1) відкрийте встановлений додаток. Python-скрипти інтегровані у програму, і вся генерація кімнат працює автономно на пристрої.

Інструкції з гри (мобільна версія):

a) основна мета:

Гравець проходить через серію згенерованих кімнат, долаючи ворогів, уникаючи пасток і збираючи бонуси. Завдання — дістатися виходу з рівня;

b) управління персонажем:

1) рух: віртуальний джойстик у лівій частині екрана;

2) атака: кнопка дії у правій частині екрана;

3) взаємодія з об’єктами: контекстна кнопка, яка з’являється при наближенні до об'єкта;

4) камера: автоматична або через свайп на екрані.

c) особливості інтерфейсу:

1) панель здоров’я: у верхньому лівому куті;

2) лічильник зібраних бонусів: у верхньому правому куті;

3) карта кімнат: відображається у вигляді міні-карти в нижньому правому куті.

Як змінити параметри алгоритму генерації

a) файли конфігурації:

У програмі передбачено меню налаштувань, яке дозволяє змінювати параметри генерації:

1) рівень складності: легкий, середній, складний;

2) кількість ворогів: від 1 до 10 на кімнату;

3) частота пасток: низька, середня, висока.

b) режим "Тестування":

Для розробників і тестувальників доступний спеціальний режим у налаштуваннях, що дозволяє переглядати параметри алгоритму в реальному часі та змінювати їх безпосередньо у грі. Наприклад:

1) швидкість генерації кімнат;

2) мінімальна/максимальна кількість об’єктів у кімнаті.

c) сновлення конфігурації через JSON:

При необхідності можна змінити стандартний конфігураційний файл, який зберігається у директорії гри. Наприклад:

1) для Android: /storage/emulated/0/Android/data/com.YourGame/files/config.json;

2) для iOS: /Documents/config.json.

Редагуйте файл за допомогою текстового редактора, змінюючи параметри, такі як maxEnemies, trapFrequency тощо.

Висновок до розділу 4

У розділі детально розглянуто всі аспекти кодування, тестування, обчислень і аналізу результатів розробленого мобільного 3D-застосунку. Було успішно реалізовано основні модулі гри, включно з генерацією кімнат, поведінкою персонажа та ворогів, а також інтеграцією алгоритмів машинного навчання для створення унікального ігрового досвіду.

Процес тестування виявив, що програма функціонує стабільно, забезпечуючи високу швидкість і коректність генерації контенту. Завдяки юніт- та інтеграційному тестуванню вдалося ідентифікувати й усунути основні помилки, що забезпечило надійну взаємодію між модулями Unity та ML-алгоритмами.

Обчислення та апробація підтвердили ефективність застосованого підходу. Алгоритм генерації кімнат демонструє високий рівень продуктивності, коректності та різноманітності. Водночас були виявлені певні проблеми, такі як дисбаланс складності деяких кімнат, які потребують додаткового доопрацювання.

Аналіз результатів свідчить про успішне досягнення поставлених цілей. Реалізований підхід довів свою ефективність для автоматизованої генерації ігрового контенту, що робить гру захоплюючою та повторювано унікальною. Водночас проєкт має потенціал для подальшого розвитку, зокрема в частині оптимізації продуктивності та адаптивності генерації.

Розроблене програмне забезпечення готове для впровадження на мобільні пристрої, забезпечуючи користувачам інтуїтивно зрозумілий ігровий досвід із високим рівнем залученості.

ВИСНОВКИ

Робота була спрямована на розробку мобільного ігрового застосунку, який використовує адаптивні алгоритми навчання та генерації контенту для створення персоналізованого ігрового досвіду. Досягнення цієї мети вимагало послідовного виконання визначених завдань, кожне з яких внесло свій внесок у реалізацію проекту та забезпечило системність і наукову обґрунтованість розробки.

Був проведений огляд сучасних мобільних ігор, які використовують адаптивні алгоритми та технології процедурної генерації. Аналіз аналогів дозволив виявити сильні та слабкі сторони таких застосунків. Було виявлено, що сильні сторони полягають у підвищенні залученості користувачів через адаптацію складності та унікальність контенту, тоді як слабкі сторони включають обмеження у складності реалізації алгоритмів і високу ресурсомісткість. Це дало змогу окреслити перспективи розвитку, серед яких удосконалення алгоритмів генерації контенту, покращення взаємодії користувача з грою та оптимізація ресурсів для мобільних платформ.

Другим етапом роботи стало специфікування вимог до мобільного ігрового застосунку. Було визначено функціональні та нефункціональні вимоги, які включають адаптивну складність, процедурну генерацію рівнів, інтерактивність інтерфейсу, низьке споживання ресурсів та можливість персоналізації гри для різних категорій користувачів. Вимоги були систематизовані з урахуванням особливостей мобільної платформи, що забезпечило їх практичну реалізацію.

Далі було спроектовано архітектуру мобільного застосунку, яка включала основні компоненти: систему аналізу даних про поведінку гравця, модуль генерації рівнів, інтерфейс користувача та адаптивний двигун складності. Проектування враховувало інтеграцію алгоритмів машинного навчання для забезпечення персоналізації ігрового досвіду.

Четвертий етап зосередився на розробці основних модулів гри. Зокрема, була реалізована система обробки даних користувача для адаптації рівнів складності, що

дозволяє враховувати динаміку ігрової активності. Модуль генерації контенту, використовуючи алгоритми Reinforcement Learning, забезпечив створення унікальних рівнів із врахуванням стилю гри користувача.

П’ятий етап включав тестування мобільного застосунку. Було проведено юніт-, інтеграційне та стрес-тестування, що дозволило виявити та усунути баги, а також оцінити ефективність алгоритмів. Тестування продемонструвало, що персоналізація контенту позитивно впливає на залученість гравців, створюючи унікальний ігровий досвід.

Останнім завданням стало визначення перспектив вдосконалення адаптивних алгоритмів. У результаті аналізу було запропоновано кілька напрямків для подальшого розвитку: інтеграція більш складних моделей поведінки ворогів, оптимізація використання ресурсів на мобільних пристроях і розширення можливостей налаштування гри під специфічні потреби користувачів.

Отже, проведена робота підтвердила актуальність та перспективність розробки мобільних ігор з використанням адаптивних алгоритмів навчання та генерації контенту. Розроблений застосунок не лише демонструє можливість реалізації персоналізованого ігрового досвіду, але й відкриває нові горизонти для вдосконалення ігрової індустрії в умовах стрімкого розвитку технологій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) Bishop C. M. *Pattern Recognition and Machine Learning*. Springer, 2006. 738 p.

2) Sutton R. S., Barto A. G. *Reinforcement Learning: An Introduction*. 2nd ed. MIT Press, 2018. 552 p.

3) Unity Technologies. *Unity 2023 Documentation*. URL: https://docs.unity.com (Last accessed: 11.09.2024).

4) Google AI Team. *TensorFlow for Deep Learning*. TensorFlow Documentation. URL: https://tensorflow.org (Last access: 25.09.2024).

5) Martin K. *Firebase and Firestore for Web Developers*. Packt Publishing, 2018. 190 p.

6) Nielsen J. *Usability Engineering*. Academic Press, 1994. 362 p.

7) Brown E., Cairns P. A grounded investigation of game immersion. *Proceedings of CHI 2004*. ACM Press, 2004. P. 1297–1300. URL: https://dl.acm.org (Last access: 15.10.2024).

8) Adams E., Rollings A. *Fundamentals of Game Design*. 3rd ed. New Riders, 2014. 576 p.

9) *Game Industry Report 2023*. URL: https://gamesindustry.biz (Last access: 01.11.2024).

10) Arulkumaran K., Deisenroth M. P., Brundage M., Bharath A. A. Deep reinforcement learning in games: A survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 2017. Vol. 9, Issue 1. P. 1–15. URL: https://doi.org/10.1109/TCIAIG.2016.2576874.

11) *Design Philosophy of Monument Valley*. Ustwo games blog. URL: https://ustwo.com/blog (Last access: 12.11.2024).

12) *Adaptive Game AI: Examples and Implementation*. Gamasutra. URL: https://gamasutra.com (Last access: 15.11.2024).

13) *Machine Learning for Game Development*. URL: https://ml4games.org (Last access: 16.11.2024)

14) Cheng L., Siu Y. H. Procedural content generation for games. *ACM Computing Surveys*, 2020. Vol. 52, Issue 3. URL: https://doi.org/10.1145/3230540.

15) Deterding S., Dixon D., Khaled R., Nacke L. From game design elements to gamefulness: Defining "gamification". *Proceedings of the 15th International Academic MindTrek Conference*, 2011. P. 9–15. URL: https://doi.org/10.1145/2181037.2181040.

16) *Gaming Retention Metrics: Trends and Insights*. Sensor Tower Report. URL: https://sensortower.com (Last access: 22.11.2024).