УДК 004.8,004.415.2,004.056
DOI https://doi.org/10.36994/2788-5518-2025-02-10-10

АВТОМАТИЗАЦІЯ ТЕСТУВАННЯ ІГРОВИХ МЕХАНІК ЗА ДОПОМОГОЮ ШТУЧНОГО ІНТЕЛЕКТУ

Завгородня Г. А., *к.т.н., доц., Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна. https://orcid.org/0000-0001-8523-1761, annzavgorodnya@gmail.com*

Завгородній В. В., *д.т.н., проф., Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна. https://orcid.org/0000-0002-8347-7183, zavgorodniivalerii@gmail.com*

Анотація. *У сучасній ігровій індустрії складність відеоігор постійно зростає: великі відкриті світи, численні варіанти взаємодій, нелінійні сценарії, процедурно-генерований контент та інтерактивні елементи стають нормою. Це створює суттєві виклики для забезпечення якості, оскільки вручну перевірити всі можливі комбінації дій гравця, виявити аномалії в механіках, оцінити баланс ігрових систем або визначити проблеми продуктивності стає дедалі складніше. У зв'язку з цим дедалі активніше впроваджуються підходи автоматизації тестування, зокрема із застосуванням методів штучного інтелекту (ШІ) та машинного навчання.*

У статті розглядається концепція застосування ШІ-агентів для автоматичного тестування ігрових механік. Зокрема, увага приділяється генерації тестових сценаріїв, імітації поведінки гравця, виявленню порушень балансу, перевірці продуктивності, регресійному тестуванню та пошуку аномалій у даних гри. Розглянуто сучасні методи, включно з підкріпленим навчанням (DRL), глибокими нейронними мережами, плануванням на основі логічних моделей (PDDL) та методами агентного моделювання, які дозволяють досліджувати ігровий простір без прямого втручання у код або модифікацій рушія гри.

На основі запропонованої методики представлено фрагмент реалізації алгоритму на Python, який включає модуль генерації тестових траєкторій із підкріпленням, аналіз логів гри та обчислення метрик покриття тестування. Показано, що використання ШІ дозволяє значно підвищити ефективність тестування: покриття тестами збільшується на ~42 %, витрати часу на тестування скорочуються на ~40 %, а кількість виявлених дефектів збільшується на ~72 % порівняно з базовими ручними методиками.

Стаття також узагальнює переваги та недоліки автоматизації тестування ігрових механік, визначає ключові виклики впровадження таких рішень у промислову практику, а також окреслює перспективи подальших досліджень, включаючи адаптацію агентів до процедурно-генерованого контенту, інтеграцію мульти-модальних даних (звук, графіка, фізика) та розвиток гібридних моделей тестування, що поєднують можливості ШІ та експертизу людини.

Ключові слова: *автоматизація тестування, ігрові механіки, штучний інтелект, машинне навчання, підкріплене навчання, регресійне тестування, QA GameDev.*

AUTOMATION OF GAME MECHANICS TESTING USING ARTIFICIAL INTELLIGENCE

Ganna Zavhorodnia*, Ph.D., Assoc. Prof., National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine. https://orcid.org/0000-0001-8523-1761, annzavgorodnya@gmail.com*

Valerii Zavhorodnii, *Dr. Sc., Prof., National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine. https://orcid.org/0000-0002-8347-7183, zavgorodniivalerii@gmail.com*

Abstract. *In the modern gaming industry, the complexity of video games is constantly increasing: large open worlds, numerous interaction options, non-linear scenarios, procedurally generated content, and interactive elements have become standard. This creates significant challenges for quality assurance, as manually verifying all possible player actions, detecting anomalies in game mechanics, evaluating system balance, or identifying performance issues becomes increasingly difficult. Consequently, approaches to test automation, particularly those leveraging artificial intelligence (AI) and machine learning, are being actively implemented.*

This paper examines the concept of using AI agents for automated testing of game mechanics. In particular, attention is given to test scenario generation, player behavior simulation, balance violation detection, performance evaluation, regression testing, and anomaly detection in game data. Modern

methods, including deep reinforcement learning (DRL), deep neural networks, planning based on logical models (PDDL), and agent-based modeling, are reviewed, enabling the exploration of the game space without direct intervention in the game code or engine modifications.

Based on the proposed methodology, a fragment of a Python implementation is presented, which includes a module for generating test trajectories using reinforcement learning, game log analysis, and calculation of test coverage metrics. It is demonstrated that the use of AI significantly improves testing efficiency: test coverage increases by approximately 42 %, testing time is reduced by ~40 %, and the number of detected defects increases by ~72 % compared to traditional manual testing methods.

The paper also summarizes the advantages and limitations of automating game mechanics testing, identifies key challenges in implementing such solutions in industrial practice, and outlines directions for further research, including adaptation of agents to procedurally generated content, integration of multi-modal data (audio, graphics, physics), and development of hybrid testing models that combine AI capabilities with human expertise.

Key words: *test automation, game mechanics, artificial intelligence, machine learning, reinforcement learning, regression testing, QA GameDev.*

Вступ. У межах сучасної розробки відеоігор індустрія стикається з небувалим зростанням складності: інтерактивність, нелінійні сценарії, процедурно-генерований контент, великі відкриті світи та онлайнові компоненти значно розширюють простір тестування, який необхідно охопити для забезпечення належної якості продукту. Традиційні підходи до тестування – ручні або скриптові сценарії – стають недостатніми, оскільки кількість можливих комбінацій дій гравця, станів системи та потенційних помилок експоненційно зростає [1; 2].

Останні дослідження демонструють, що застосування методів штучного інтелекту (ШІ) здатне суттєво підвищити ефективність тестування ігор. У роботах останніх років розглянуто можливості генерації тестових сценаріїв, моделювання поведінки гравця, виявлення помилок, автоматичної перевірки продуктивності та сумісності ігрових механік [3; 4]. Зокрема, активно досліджуються підходи із застосуванням планування дій (PDDL) для побудови регресійних тестів [5], а також агентно-орієнтовані методи, де віртуальні агенти з елементами підкріпленого навчання досліджують ігрове середовище та виявляють відхилення у логіці гри [6; 7].

Значну увагу приділено й методам, що базуються на аналізі візуальних даних, наприклад, піксель-орієнтованому тестуванню, коли агент досліджує гру, сприймаючи лише зображення екрану [8]. Такі рішення мінімізують потребу у втручанні в код ігрового движка та роблять тестування більш універсальним. Паралельно розвиваються підходи до автоматичної генерації контенту та процедурного моделювання [9, 10], що має безпосередній зв'язок із побудовою тестових середовищ і сценаріїв для ШІ-агентів.

В українському науковому просторі досліджуються питання, пов'язані з математичним моделюванням і формалізацією процесів [11; 12], що забезпечують теоретичну основу для побудови моделей тестування. Розробка методів автоматичної генерації контенту на основі процедурних алгоритмів [13; 14] сприяє створенню адаптивних тестових середовищ, у яких ШІ може ефективно перевіряти поведінку гри в умовах динамічних змін. Окремі праці присвячено створенню штучних текстур із заданими параметрами [15–17], що може бути застосовано для валідації візуальної частини ігрового рушія. Водночас методи пошуку аномалій у даних за допомогою машинного навчання [18] є ефективним інструментом для виявлення неочевидних дефектів у логах гри та телеметрії.

На основі проведеного аналізу можна стверджувати, що сучасні дослідження у сфері автоматизації тестування з використанням ШІ активно розвиваються, але питання комплексного застосування таких підходів саме до тестування ігрових механік залишається відкритим. Це визначає наукову та практичну актуальність даної роботи.

Метою дослідження є розроблення методики автоматизації тестування ігрових механік із використанням алгоритмів штучного інтелекту та демонстрація її ефективності шляхом реалізації фрагмента рішення, обчислення метрик покриття тестування, продуктивності та порівняння з базовою моделлю ручного тестування.

Виконання досліджень. У межах дослідження тестування ігрових механік гра моделюється як динамічна система, що описується множиною станів S, множиною дій A, функцією переходів $T : S \times A \to S$ та умовами-цілями $G \in S$.

Позначимо:

– $s_0 \in S$ – початковий стан гри;

– агент (тестовий бот) обирає дії $a \in A$;

– після виконання дії a у стані s система переходить у стан $s' = T(s,a)$.

Мета тестування – забезпечити покриття множини критичних сценаріїв $C \in S \times A$, таких як небезпечні комбінації дій, можливість «зламу» механіки, несправності тощо.

Для автоматичного тестування застосовано підхід *DRL*-агента *(deep reinforcement learning)*, який навчається політиці $\pi(a \mid s)$, що максимізує очікувану винагороду:

$$J(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{T}\gamma^{t} r(s_t, a_t)\right], \tag{1}$$

де $r(s,a)$ – винагорода за досягнення станів із критичних сценаріїв C;
γ – коефіцієнт дисконтування.

Розглядалися кілька алгоритмів *DRL*:

– *DQN (Deep Q-Network)* – базова модель, швидка та проста для демонстрації;
– *Double DQN* – зменшує переоцінку *Q*-значень;
– *PPO (Proximal Policy Optimization)* – стійка до великих просторів станів.

Для експерименту обрано *DQN*, як ефективний баланс між простотою і результативністю.

Тестові сценарії формуються двома способами:

– статична генерація – створення повного набору комбінацій дій у критичних точках гри;
– динамічна генерація *DRL*-агентом – агент досліджує гру самостійно, обираючи дії відповідно до винагороди, що стимулює пошук дефектів та критичних сценаріїв.

Такий підхід дозволяє охопити не лише відомі сценарії, а й непередбачувані шляхи поведінки гравця.

Для оцінки ефективності застосовано:

1. Покриття критичних сценаріїв *Coverage*:

$$Coverage = \frac{C_{28\text{Œ}2;\ 5=>}}{[C]}. \tag{2}$$

2. Прискорення тестування (*Speedup*):

$$Speedup = \frac{T_{manual} - T_{AI}}{T_{manual}} \times 100\%, \tag{3}$$

де T_{manual} – тривалість ручного тестування;
T_{AI} – тривалість із застосуванням ШІ.

3. Якість виявлення дефектів: *Precision, Recall, F1-score.*

4. Розподіл помилок за категоріями: логічні, графічні, помилки продуктивності.

Нижче наведено фрагмент коду на *Python*, що демонструє реалізацію простого агента *DQN* для автоматизованого тестування ігрових механік. Агент навчається політиці $\pi(a \mid s)$, що максимізує очікувану винагороду, пов'язану з досягненням критичних станів і виявленням дефектів.

```
import gym
import numpy as np
import random
from collections import deque
# GameEnv — оболонка гри
env = GameEnv()
state = env.reset()
# Параметри
gamma = 0.99
alpha = 0.001
batch_size = 64
# Простий DQN-агент
class DQNAgent:
    def __init__(self, state_size, action_size):
        self.state_size = state_size
        self.action_size = action_size
        self.memory = deque(maxlen=2000)
        self.q_table = np.zeros((state_size, action_size))
    def act(self, state, eps=0.1):
        if np.random.rand() < eps:
            return random.randrange(self.action_size)
        return np.argmax(self.q_table[state])
    def learn(self, batch):
        for (s, a, r, ns, done) in batch:
            target = r + gamma * np.max(self.q_table[ns]) * (not done)
            self.q_table[s, a] += alpha * (target - self.q_table[s, a])
    def remember(self, transition):
        self.memory.append(transition)
# Ініціалізація агента
```

```
agent = DQNAgent(env.state_size, env.action_size)
# Основний цикл навчання
for episode in range(1000):
    state = env.reset()
    done = False
    while not done:
        action = agent.act(state)
        next_state, reward, done, info = env.step(action)
        agent.remember((state, action, reward, next_state, done))
        state = next_state
        if len(agent.memory) > batch_size:
            batch = random.sample(agent.memory, batch_size)
            agent.learn(batch)
```

Основні аспекти наведеної реалізації:

1. Ініціалізація середовища env = GameEnv() створює умовну оболонку гри, яка дозволяє агенту взаємодіяти зі станами та виконувати дії без зміни коду ігрового рушія.

2. Параметри навчання:

– gamma = 0.99 – коефіцієнт дисконтування майбутньої винагороди;

– alpha = 0.001 – швидкість навчання;

– batch_size = 64 – розмір пакета досвіду для оновлення *Q*-таблиці.

3. Структура агента. Клас *DQNAgent* містить методи:

– act – вибір дії за *ε*-жадібною стратегією;

– learn – оновлення *Q*-значень на основі накопиченого досвіду;

– remember – збереження переходів у буфері досвіду (*memory*).

4. Агент проходить серію епізодів, поступово накопичуючи досвід і оновлюючи політику. Це дозволяє йому ефективно досліджувати простір станів та виявляти критичні сценарії, які складно перевірити вручну.

5. Аналітика та результати застосування:

– агент здатний значно розширити охоплення сценаріїв порівняно з ручним тестуванням;

– завдяки дослідженню нестандартних комбінацій дій, агент знаходить дефекти, які можуть бути пропущені тестувальниками;

– час виконання тестування скорочується, що підтверджує формули (2) та (3) для метрик *Coverage* і *Speedup*;

– агент може навчатися на різних рівнях складності та швидко підлаштовувати політику під зміни в грі.

6. Агент може пропускати дефекти, що потребують інтуїтивного оцінювання досвіду гравця; ефективність залежить від правильного формулювання винагороди та конфігурації середовища.

Наведений підхід демонструє потенціал використання штучного інтелекту для ефективного *QA GameDev* та може бути масштабований для складніших ігор і процедурно-генерованих світів.

Для відтворення експерименту було створено тестову ігрову механіку (простий платформер із 10 рівнями). Було проведено два підходи:

– базовий: ручне тестування 5 тестерами по 4 години кожен;

– ШІ-підхід: агент *DRL* працював 12 годин без перерви.

Метрики:

– покриття критичних сценаріїв: ручне – 68 %; ШІ-агент – 96 %;

– кількість виявлених дефектів: ручне – 25 од.; ШІ-агент – 43 од;

– витрачений час: ручне – 20 людино-годин; ШІ-агент – 12 годин.

Отже, застосовуючи формули (2) та (3), отримаємо:

$$Coverage = \frac{96 - 68}{68} \approx 41.2\ \%$$

$$Speedup = \frac{20 - 12}{20} \approx 40\ \%$$

У таблиці 1 наведено порівняльні результати.

Отримані результати підтверджують, що інтеграція ШІ у процес тестування ігрових механік може забезпечити значне підвищення ефективності. Проте слід зазначити низку обмежень: агент міг пропускати дефекти, які потребують інтуїтивного розуміння користувацького досвіду; також потрібна підготовка логів, інтеграція з ігровим движком, формулювання винагороди – усе це потребує ресурсу. Тому застосування гібридного підходу, який полягає у сумісному використанні людини і ШІ, залишається оптимальним.

Таблиця 1
Порівняльні результати

Показник	Ручне тестування	ШІ-тестування
Покриття сценаріїв, %	68	96
Кількість виявлених дефектів, од.	25	43
Час витрачено, г	20	12

Висновок. Автоматизація тестування ігрових механік із використанням алгоритмів штучного інтелекту є актуальним та ефективним напрямом сучасного *QA* у геймдеві. Експериментальна реалізація *DRL*-агента продемонструвала значне підвищення ефективності тестування: покриття критичних сценаріїв збільшилось на ~42 %, час виконання тестів скоротився на ~40 %, а кількість виявлених дефектів зросла на ~72 % порівняно з традиційним ручним тестуванням.

Отримані результати підтверджують переваги використання ШІ для дослідження складних ігрових просторів без прямого втручання в код або модифікацій рушія, автоматичного генерування сценаріїв, аналізу логів гри та оцінки продуктивності системи. Водночас, агент може пропускати дефекти, які потребують інтуїтивного розуміння користувацького досвіду, а якість тестування значною мірою залежить від коректного формулювання винагороди та підготовки тестового середовища.

Серед перспективних напрямів подальших досліджень можна виділити: покращення пояснюваності рішень ШІ, адаптацію агентів до процедурно-генерованого контенту, інтеграцію мультимодальних даних у процес автоматичного тестування та розвиток гібридних моделей, що поєднують машинний інтелект та людську експертизу.

Література

1. Paduraru C., Paduraru M., Stefanescu A. RiverGame – a game testing tool using artificial intelligence. *Verification and Validation (ICST'22)* : 15th IEEE Conference on Software Testing. 2022. Pp. 422–432. DOI: https://doi.org/10.1109/ICST53961.2022.00048.
2. Zhang B. et al. Enhancing human-AI collaboration in game testing. *Proceedings of SPIE*. Vol. 13550. 2025. DOI: https://doi.org/10.1117/12.3059740.
3. Prasetya I. S. W. B., Pastor F., Kifetew F., Prandi D., Shirzadehhajimahmood S., Vos T. E. J. et al. *An agent-based approach to automated game testing: An experience report. A-TEST 2022* : 13th International Workshop on Automating Test Case Design, Selection and Evaluation. 2022. DOI: https://doi.org/10.1145/3548659.3561305.
4. Alharthi S. A. Generative AI in game design: Enhancing creativity or constraining innovation? *Journal of Intelligence*. 13 (6). 60. 2025. DOI: https://doi.org/10.3390/jintelligence13060060.
5. Mastain V., Petrillo F. A behavior-driven development and reinforcement learning approach for videogame automated testing. Proceedings of the ACM/IEEE 8th International Workshop on Games and Software Engineering (GAS '24). 2024. Pp. 1–8. DOI: https://doi.org/10.1145/3643658.3643919.
6. Bergdahl J., Gordillo C., Tollmar K., Gisslén L. Augmenting automated game testing with deep reinforcement learning. arXiv preprint arXiv:2103.15819. 2021. DOI: https://doi.org/10.48550/arXiv.2103.15819.
7. Shirzadehhajimahmood S., Kifetew F., Prandi D. Automated game testing with online search agents and model-free reinforcement learning. *Software Testing, Verification & Reliability*. Wiley, 2024. DOI: https://doi.org/10.1002/stvr.70002.
8. Liu G., Cai M., Zhao L., Qin T., Brown A., Bischoff J., Liu T. Inspector: Pixel-based automated game testing via exploration, detection and investigation. IEEE Conference on Games (CoG 2022). 2022. DOI: https://doi.org/10.48550/arXiv.2207.08379.
9. Завгородній В. В., Завгородня Г. А., Валявська Н. О., Адаменко В. С., Дороговцев Є. В., Несмачний П. В. Метод автоматичної генерації контенту на основі процедурних алгоритмів. *Вчені записки ТНУ ім. В. І. Вернадського. Серія: Технічні науки*. 2022. 33 (72). № 1. 91–96. DOI: https://doi.org/10.32838/2663-5941/2022.1/15.
10. Shaker N., Togelius J., Nelson M. J. Procedural Content Generation in Games: A Textbook and an Overview of Current Research. *Springer*. 2016. DOI: https://doi.org/10.1007/978-3-319-42716-4.
11. Aminian M. et al. Mathematical methods for visualization and anomaly detection in time-series. R*oyal Society Open Science*. 2020. 7 (5). DOI: https://doi.org/10.1098/rsfs.2019.0086.
12. Завгородній В. В., Завгородня Г. А., Дроботович К. Є., Тенігін О. В., Шматко М. М. Математичне моделювання у методах формального дослідження. *Вчені записки ТНУ ім. В. І. Вернадського. Серія: Технічні науки*. 2021. 32 (71). № 6. 75–79. DOI: https://doi.org/10.32838/2663-5941/2021.6/12.
13. Beukman M. et al. Hierarchically composing level generators for the creation of complex structures. *IEEE Transactions on Games*. 2024. 16 (2). 459–469. DOI: https://doi.org/10.1109/TG.2023.3297619.
14. Завгородній В. В., Завгородня Г. А., Демченко І. В., Крамаренко К. С., Шевченко І. О., Юрченко А. В. Метод створення штучних текстур із заданими параметрами. *Вчені записки ТНУ ім. В. І. Вернадського. Серія: Технічні науки*. 2022. 33 (72). № 2. 86–90. DOI: https://doi.org/10.32838/2663-5941/2022.2/14.
15. Azimi S. et al. Anomaly analytics in data-driven machine learning. Data Mining and Knowledge Discovery, 2025. DOI: https://doi.org/10.1007/s41060-024-00593-y
16. Moharam M. H. et al. Anomaly detection using machine learning and adopted digital twin concepts in radio environments. *Scientific Reports*. 2025. 15. DOI: https://doi.org/10.1038/s41598-025-02759-5.
17. Завгородній В. В., Завгородня Г. А., Валявська Н. О., Герасименко О. О., Калюжний О. В., Степовий А. В. Пошук аномалій у даних за допомогою машинного навчання. *Вчені записки ТНУ ім. В. І. Вернадського. Серія: Технічні науки*. 2022. 33 (72). № 3. 39–43. URL: https://tech.vernadskyjournals.in.ua/journals/2022/3_2022/6.pdf
18. Souchleris K., Sidiropoulos G. K., Papakostas G. A. Reinforcement learning in game industry – review, prospects and challenges. *Applied Sciences*. 2023.13 (4). 2443. DOI: https://doi.org/10.3390/app13042443.

References

1. Paduraru C., Paduraru M., Stefanescu A. (2022). RiverGame – a game testing tool using artificial intelligence. *Verification and Validation (ICST'22)* : 15th IEEE Conference on Software Testing. Pp. 422–432. DOI: https://doi.org/10.1109/ICST53961.2022.00048.
2. Zhang B. et al. (2025). Enhancing human-AI collaboration in game testing. *Proceedings of SPIE*. Vol. 13550. DOI: https://doi.org/10.1117/12.3059740.
3. Prasetya I. S. W. B., Pastor F., Kifetew F., Prandi D., Shirzadehhajimahmood S., Vos T. E. J. et al. (2022). An agent-based approach to automated game testing: An experience report. *A-TEST 2022* : 13th International Workshop on Automating Test Case Design, Selection and Evaluation. DOI: https://doi.org/10.1145/3548659.3561305.
4. Alharthi S. A. (2025). Generative AI in game design: Enhancing creativity or constraining innovation? *Journal of Intelligence*. 13 (6). 60. DOI: https://doi.org/10.3390/jintelligence13060060.
5. Mastain V., Petrillo F. (2024). A behavior-driven development and reinforcement learning approach for videogame automated testing. Proceedings of the ACM/IEEE 8th International Workshop on Games and Software Engineering (GAS '24). Pp. 1–8. DOI: https://doi.org/10.1145/3643658.3643919.
6. Bergdahl J., Gordillo C., Tollmar K., Gisslén L. (2021). Augmenting automated game testing with deep reinforcement learning. arXiv preprint arXiv:2103.15819. DOI: https://doi.org/10.48550/arXiv.2103.15819.
7. Shirzadehhajimahmood S., Kifetew F., Prandi D. (2024). Automated game testing with online search agents and model-free reinforcement learning. *Software Testing, Verification & Reliability*. DOI: https://doi.org/10.1002/stvr.70002.
8. Liu G., Cai M., Zhao L., Qin T., Brown A., Bischoff J., Liu T. (2022). Inspector: Pixel-based automated game testing via exploration, detection and investigation. IEEE Conference on Games (CoG 2022). DOI: https://doi.org/10.48550/arXiv.2207.08379.
9. Zavhorodnii V. V., Zavhorodnia H. A., Valiavska N. O., Adamenko V. S., Dorohovtsev Ye. V., Nesmachnyi P. V. (2022). Metod avtomatychnoi heneratsii kontentu na osnovi protseduralnykh alhorytmiv [Method of automatic content generation based on procedural algorithms]. *Vcheni zapysky TNU im. V.I. Vernadskoho. Seriia: Tekhnichni nauky*. 33 (72). № 1. S. 91–96. DOI:https://doi.org/10.32838/2663-5941/2022.1/15 [in Ukrainian]
10. Shaker N., Togelius J., Nelson M. J. (2016). Procedural Content Generation in Games: A Textbook and an Overview of Current Research. *Springer*. DOI: https://doi.org/10.1007/978-3-319-42716-4.
11. Aminian M. et al. (2020). Mathematical methods for visualization and anomaly detection in time-series. *Royal Society Open Science*. 7 (5). DOI: https://doi.org/10.1098/rsfs.2019.0086.
12. Zavhorodnii V. V., Zavhorodnia H. A., Drobotovych K. Ye., Tenihin O. V., Shmatko M. M. (2021). Matematychne modeliuvannia u metodakh formalnoho doslidzhennia [Mathematical modeling in formal research methods]. *Vcheni zapysky TNU im. V.I. Vernadskoho. Seriia: Tekhnichni nauky.* 32 (71). № 6. S. 75–79. DOI: https://doi.org/10.32838/2663-5941/2021.6/12 [in Ukrainian].
13. Beukman M. et al. (2024). Hierarchically composing level generators for the creation of complex structures. *IEEE Transactions on Games*. 16 (2). S. 459–469. DOI: https://doi.org/10.1109/TG.2023.3297619.
14. Zavhorodnii V. V., Zavhorodnia H. A., Demchenko I. V., Kramarenko K. S., Shevchenko I. O., Yurchenko A. V. (2022). Metod stvorennia shtuchnykh tekstur iz zadanymy parametramy [Method for creating artificial textures with given parameters]. *Vcheni zapysky TNU im. V. I. Vernadskoho. Seriia: Tekhnichni nauky*. 33 (72). № 2. S. 86–90. DOI: https://doi.org/10.32838/2663-5941/2022.2/14 [in Ukrainian].
15. Azimi S., et al. (2025). Anomaly analytics in data-driven machine learning. Data Mining and Knowledge Discovery. DOI: https://doi.org/10.1007/s41060-024-00593-y.
16. Moharam M. H. et al. (2025). Anomaly detection using machine learning and adopted digital twin concepts in radio environments. *Scientific Reports*. 15. DOI: https://doi.org/10.1038/s41598-025-02759-5.
17. Zavhorodnii V. V., Zavhorodnia H. A., Valiavska N. O., Herasymenko O. O., Kaliuzhnyi O. V., Stepovyi A. V. (2022). Poshuk anomalii u danykh za dopomohoiu mashynnoho navchannia [Anomaly detection in data using machine learning]. *Vcheni zapysky TNU im. V. I. Vernadskoho. Seriia: Tekhnichni nauky*. 33 (72). № 3. S. 39–43. URL: https://tech.vernadskyjournals.in.ua/journals/2022/3_2022/6.pdf [in Ukrainian].
18. Souchleris K., Sidiropoulos G. K., Papakostas G. A. (2023). Reinforcement learning in game industry – review, prospects and challenges. *Applied Sciences*. 13 (4). 2443. DOI: https://doi.org/10.3390/app13042443.

Дата першого надходження статті до видання: 29.10.2025
Дата прийняття статті до друку після рецензування: 28.11.2025
Дата публікації (оприлюднення) статті: 26.12.2025