УДК 004.75

О. О. Складнюк; В. П. Майданюк, канд. техн. наук, доц.;
І. Р. Арсенюк, канд. техн. наук, доц.

ГРАФІЧНИЙ РЕДАКТОР ДЛЯ РЕДАГУВАННЯ ВІДЕОІГОР З РОЗШИРЕНИМИ ФУНКЦІОНАЛЬНИМИ МОЖЛИВОСТЯМИ

У статті виконано аналітичний огляд методів та засобів для редагування сцен у 2D-відеоіграх. Обґрунтовано актуальність створення подібного інструменту з урахуванням поширеності інді-розробок і потреби в інтуїтивно зрозумілих редакторах. Проаналізовано технічні аспекти реалізації функціоналу, таких як інтерактивне переміщення об'єктів (drag-and-drop), редагування властивостей об'єктів сцени у реальному часі, управління шарами та компонентами сцени. Розглянуто підходи до збереження структури сцени у форматі JSON, що дозволяє легко експортувати та імпортувати дані про ігрові рівні. Особливу увагу приділено використанню 2D-графічної бібліотеки Pixi.js, як рушія для рендерингу сцени, а також принципам побудови архітектури на основі об'єктно-орієнтованого підходу. Здійснено аналіз використання подієво-орієнтованої моделі для забезпечення взаємодії між компонентами редактора. Окремо проаналізовано перспективи розширення функціоналу редактора, зокрема реалізацію сітки для вирівнювання об'єктів, систему undo/redo, підтримку кількох сцен, та інструменти для створення складніших ігрових механік. Визначено напрями подальших досліджень, зокрема створення адаптивної системи плагінів для розширення функцій редактора. Основним науково-практичним результатом дослідження є визначення ключових компонентів, потрібних для побудови ефективного редактора 2D-сцен, а також реалізація базової версії редактора, що може бути використана як основа для подальшого розвитку. Основними перевагами запропонованого редактору є модульність інтерфейсу, легка масштабованість проєкту, простота розробки та підтримки, збереження сцен у JSON, наявність системи шарів, drag-and-drop редагування, базова підтримку undo/redo, інтеграція з Pixi.js як візуальним рушієм, платформнонезалежність. Практична цінність роботи полягає у можливості використання розробленого інструменту для побудови прототипів та ігрових рівнів без необхідності написання коду, що сприяє прискоренню процесу розробки 2D-ігор, дозволяючи розробникам зосередитися на ігровій логіці та дизайні, а не на технічних деталях реалізації сцени. Запропонований редактор може бути інтегрований у навчальні програми для ознайомлення студентів із принципами побудови ігрових середовищ та графічного програмування.

Ключові слова: *графічний редактор, відеоігри, сцени, 2D, drag-and-drop, Pixi.js, JSON, undo/redo, шари, об'єктно-орієнтоване програмування.*

Вступ

У сучасну епоху ігрової індустрії все більше людей отримують доступ до інструментів створення власних відеоігор. Зокрема, зростає популярність так званих інді-розробок (англ. *indie*, скорочено від *independent* – створення ігор командами або окремими розробниками, що не залежать від великих видавців), які потребують простих, ефективних та інтуїтивно зрозумілих інструментів для створення та редагування ігрових сцен. Багато з таких проєктів мають обмежені ресурси, тому виникає потреба у графічних редакторах, які дозволяють створювати та змінювати 2D-сцени без потреби у глибоких знаннях програмування. Одним із перспективних рішень є створення універсального редактора, що підтримує візуальне редагування, організацію об'єктів сцени та збереження її структури у зручному форматі, придатному для імпорту в ігровий рушій [1].

Графічний дизайн дуже важливий для створення мультимедіа. Він має широкий спектр застосування у таких сферах, як реклама, книги, веб-сайти тощо. Гарні дизайни, які є візуально приємними та чітко доносять інформацію, значною мірою залежать від досвіду дизайнерів. Останнім часто потрібно створювати той самий дизайн у різних розмірах або переналаштовувати наявні дизайни на нові розміри, щоб вони відповідали різним роздільним

здатностям дисплеїв, наприклад, від веб-сайтів до екранів мобільних телефонів, що вимагає багато людської праці та інтелекту. Інструменти автоматизації для створення дизайнів у потрібних розмірах допоможуть дизайнерам і значно скоротять час циклу проєктування.

Першим кроком до візуалізації реальних дизайнів є створення графічних макетів шляхом визначення розташування та розмірів елементів дизайну. Традиційні методи машинного навчання, такі як генеративно-змагальні мережі (GAN) [2, 3], використовуються для створення природних цифрових зображень у піксельному просторі [4 – 7]. Однак ці методи не підходять для синтезу графічних дизайнів, які є не пікселями, а макетами редагованих елементів дизайну. Нещодавно було запропоновано LayoutGAN [8] для безпосереднього генерування геометричних параметрів графічних елементів замість пікселів зображення з випадкових величин. Це демонструє можливість генерування параметризованих графічних макетів за допомогою нейронних мереж, але такий підхід не здатний виконувати практичні завдання, такі як автоматичний графічний дизайн та ретаргетинг, коли надаються певні дані користувача. Такі дані містять певні елементи дизайну та їх відповідні атрибути на основі контенту. Ці атрибути вводять додаткові просторові обмеження, яких необхідно дотримуватися під час генерації макета.

Мета

Метою статті є розширення функціональних можливостей графічного редактору для 2D-сцен, визначення архітектурних рішень і ключових компонентів для реалізації функціоналу, необхідного для редагування ігрових рівнів, а також побудова відповідної базової версії інструменту, що може бути використана у процесі інді-розробки відеоігор.

Для досягнення мети слід розв'язати такі задачі:

– здійснити аналітичний огляд сучасних інструментів та підходів до створення графічних редакторів для редагування сцен у 2D-відеоіграх;

– визначити архітектурні принципи, на яких базується побудова редакторів;

– розглянути рушії, використовувані в редакторах у контексті інді-розробки відеоігор.

– розглянути та проаналізувати переваги та недоліки популярних редакторів для редагування відеоігор;

– створити власну версію графічного редактору для 2D-сцен з розширенням функціональних можливостей та здійснити його тестування.

Обґрунтування актуальності розробки

Попит на прості та доступні інструменти для створення 2D-відеоігор постійно зростає. Згідно з аналітичними дослідженнями, кількість інді-розробників та незалежних ігрових студій упродовж останніх років значно збільшилася. Наприклад, платформи подібна до Steam щороку приймають тисячі нових інді-ігор. Одночасно з цим зростає потреба у редакторах, які дозволяють створювати ігрові сцени без написання коду. Водночас на ринку присутні як складні професійні рішення, так і досить примітивні інструменти, що не задовольняють потреби більшості користувачів. Це свідчить про наявність ніші для розробки редактора, що поєднує функціональність і простоту використання. Актуальність проблеми також підтверджується зростанням використання бібліотек на зразок Pixi.js, що дозволяють створювати високопродуктивні браузерні 2D-додатки [8].

Отже, все очевиднішим є той факт, що, у зв'язку зі швидким розвитком технологій ігрової індустрії та зростанням вимог до якості продуктів, постає необхідність у створенні висококласних інструментів для редагування ігрових сцен. Графічний редактор для 2D-відеоігор надає можливість не лише створювати сцени, а й впроваджувати ефекти, що забезпечить більш інтерактивний та захоплюючий ігровий процес. Сутність дослідження полягає у розробці інструментів, що, на відміну від наявних аналогів, дозволять гнучко працювати з різними елементами ігрових сцен.

Аналіз функціональних можливостей графічного редактора для редагування десктопних відеоігор

У типовому редакторі сцен для 2D-ігор мають місце такі основні компоненти:

1. Сцена та об'єкти зберігаються у форматі JSON, що дозволяє легко імпортувати/експортувати дані та інтегрувати з рушієм.
2. Сцена підтримує ієрархічну структуру, що дозволяє створювати вкладені об'єкти, групи, шари з різними рівнями видимості.
3. Інтерфейс drag-and-drop, що забезпечує зручне переміщення, масштабування та обертання об'єктів мишею у редакторі [9].
4. Підтримка історії змін, що дозволяє користувачеві повертати попередні стани сцени (Undo/redo).
5. Панель властивостей дозволяє змінювати позицію, розмір, текстуру, поведінку об'єктів тощо.

Під час розробки редактора часто використовують бібліотеки на зразок Pixi.js для рендерингу сцени, React або Vue для побудови інтерфейсу користувача (User Interface, UI), Zustand або Redux – для керування станом. У складніших рішеннях можливе використання WebAssembly для пришвидшення обчислень, а також Canvas/WebGL для рендерингу великої кількості об'єктів у сцені без втрати продуктивності.

Сьогодні багато популярних ігрових рушіїв, таких як Unity та Unreal Engine, пропонують розробникам потужні інструменти для редагування сцен та об'єктів. Ці інструменти дозволяють змінювати візуальні та функціональні аспекти гри без необхідності переписувати код. Unity – це потужний і дуже популярний багатоплатформний рушій для створення комп'ютерних ігор, а також інтерактивних 2D і 3D-додатків. Він надає розробникам усе необхідне для створення ігор. Саме тому його використовують фахівці з усього світу, і багато хто з них вважає Unity номером 1 у світі розробки ігор [10]. Принцип роботи в Unity передбачає ряд особливостей, які розробники мають врахувати під час створення гри. Нижче розглянемо основні з них детальніше.

1. Першою особливістю є створення сцени – простору, де розміщуються та взаємодіють об'єкти гри. Їх може бути декілька на кожному рівні. В Unity можна створювати та редагувати сцени, визначати компоненти оточення, задавати вигляд та рівні освітлення, камери і т. ін.
2. Ще однією важливою особливістю є додавання основних будівельних блоків гри – об'єктів (таких як персонажі, предмети, перешкоди тощо), а також визначити їх взаємодію та поведінку.
3. Інша особливість передбачає застосування скриптів та кодування. За допомогою скриптів можна створювати поведінку об'єктів та керувати ігровою логікою, а за допомогою коду – унікальні функції, визначити умови перемоги або поразки, обробляти введення користувача тощо. Доцільно також зазначити, що вся робота з Unity здійснюється мовою програмування C#.
4. На етапі, коли гра створена, варто її ретельне тестування на різних пристроях та платформах, що допомагає виявити помилки, покращити геймплей та оптимізувати продуктивність. Це в кінцевому підсумку, дозволить забезпечити ефективне функціонування гри на різних пристроях.
5. На останньому кроці гра публікується на вибраних платформах, таких як комп'ютери, мобільні пристрої, ігрові приставки та інше. Unity надає ряд корисних інструментів для експорту та публікації гри на різних платформах, що дозволяє досягти широкої аудиторії гравців [2].

А тепер розглянемо особливості ігрового рушія Unreal Engine. Цей інструмент для створення ігор і сцен із використанням 3D-моделей, розроблений компанією Epic Games, має такі особливості:

1. Він надає потужні засоби для створення дивовижних графічних ефектів: високоякісне освітлення, рендеринг (формування зображення або відео з тривимірної сцени) та підтримку віртуальної реальності.
2. Дозволяє досить реалістично моделювати об'єкти, відтворювати їх рух і взаємодію з навколишнім середовищем.
3. Може бути використаний для розробки ігор на різних платформах: комп'ютерах, консолях, мобільних пристроях і у віртуальній реальності.
4. Він використовує мову програмування C++, а також має систему розширень та графічний інтерфейс для того, щоб створити гру, не знаючи кодингу [11].

Огляд та аналіз аналогів

Tiled Map Editor – це потужний, безкоштовний та відкритий редактор карт для 2D-ігор, який підтримує різні типи карт: ортогональні, ізометричні та гексагональні. Загальний вигляд інтерфейсу даного редактору наведено на рис. 1. Основні можливості Tiled Map Editor такі:

– підтримка кількох шарів тайлів та об'єктів, що дозволяє створювати досить складні сцени [12];

– можливість додавати прямокутники, еліпси, полігони та інші об'єкти з індивідуальними властивостями, що фактично надає гнучкості об'єктним шарам;

– можливість створення та повторного використання шаблонів для стандартних елементів;

– підтримка експорту у різні формати: TMX (XML), JSON, Lua та інші, що забезпечує сумісність із багатьма ігровими рушіями.

LDtk (Level Designer Toolkit) – це сучасний редактор рівнів, створений автором гри Dead Cells. Він поєднує простоту використання та досить високу ефективність [13]. Загальний вигляд інтерфейсу цього редактору наведено на рис. 2.

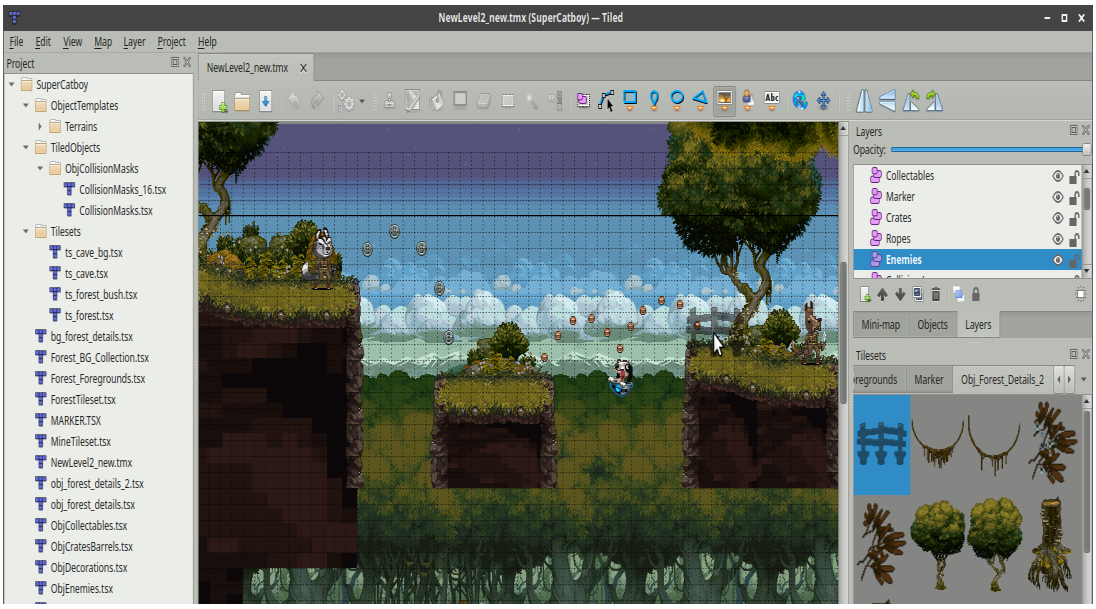

Рис. 1. Інтерфейс редактора Tiled Map Editor

Основні особливості LDtk такі:

– наявність сучасного, інтуїтивно зрозумілого та зручного інтерфейсу користувача;

– наявність системи автоматичного розміщення тайлів (auto-tiling) на основі правил, що значно пришвидшує процес створення рівнів;

– генерування PNG-файлів для візуалізації та JSON-файлів для даних, що полегшує інтеграцію з ігровими рушіями.

– підтримка кастомних полів, що забезпечується можливістю додавання власних полів до об'єктів для зберігання додаткової інформації.

Рис. 2. Інтерфейс редактора LDtk

Ogmo Editor – легкий, відкритий редактор рівнів, розроблений для інді-розробників [14]. Загальний вигляд його інтерфейсу наведено на рис. 3. Основними перевагами редактора є:

– передбачено проєктно-орієнтований підхід, де всі налаштування зберігаються у проєкті, що забезпечує узгодженість між рівнями;

– передбачено підтримку різних типів шарів (тайлових, сіткових, об'єктних та декоративних) для гнучкого дизайну рівнів;

– передбачено зручний формат для інтеграції з багатьма ігровими рушіями (експорт у JSON);

– простота використання завдяки наявності інтуїтивно зрозумілого інтерфейсу, що дозволяє швидко освоїтися навіть новачкам.

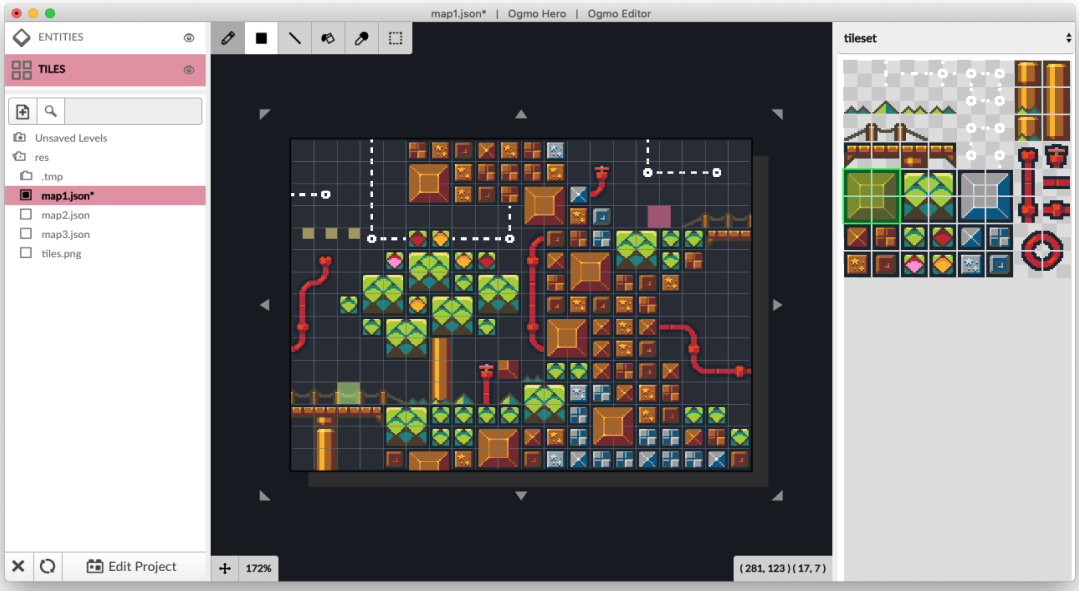

Рис. 3. Інтерфейс редактора Ogmo Editor

Вищенаведені аналоги графічних редакторів для 2D-ігор мають спільний набір базових функцій, таких як підтримка шарів, drag-and-drop інтерфейс, збереження сцен у форматі JSON та система undo/redo. Ці функції створюють зручне середовище для візуального редагування ігрових сцен без потреби у прямому втручанні у код. Завдяки використанню

таких бібліотек, як Pixi.js, і сучасних підходів до управління станом (наприклад, Zustand), забезпечується гнучкість і масштабованість редакторів. У поєднанні з можливістю інтеграції з рушіями або власними пайплайнами розробки, ці редактори стають незамінними інструментами, особливо у сфері інді-розробки.

Попри наявності широкого вибору редакторів для 2D-ігор, усі розглянуті інструменти мають певні обмеження: складність інтеграції користувацьких об'єктів (Tiled), вузька спеціалізація та низька гнучкість (LDtk), відсутність ієрархії та обмеженість налаштувань (Ogmo Editor). Саме тому існує нагальна потреба у створенні редактора, що поєднує гнучкість налаштувань, інтуїтивно зрозумілий інтерфейс та просту інтеграцію з ігровими рушіями. Усунення цих недоліків і стало основою розробки власного редактора.

Розробка власного редактору Editor

Для досягнення мети було створено власний графічний редактор, що має загальний інтерфейс, наведений на рис. 4. Його порівняння з аналогами наведено у таблиці 1.

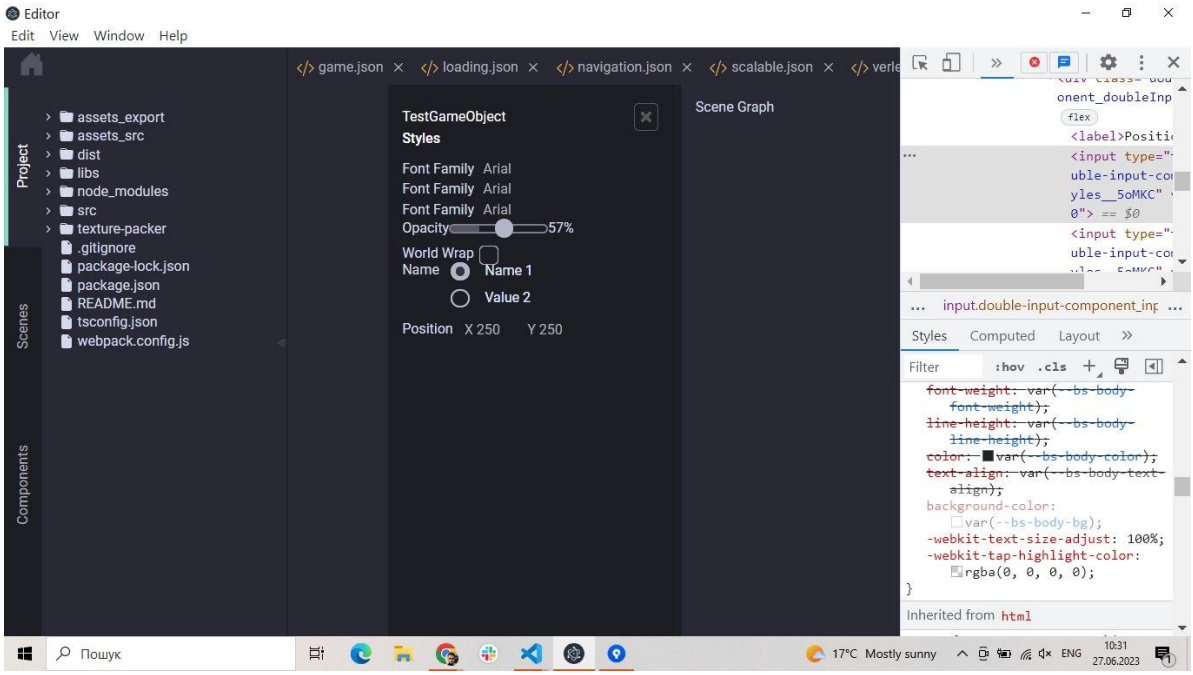

Рис. 4. Прототип інтерфейсу власного редактору

Цей графічний редактор поєднує гнучкість сучасних веб-технологій з можливістю запуску на десктопі завдяки використанню React.js для побудови інтерфейсу та Electron для створення кросплатформного десктопного застосунку. Редактор надає функції створення власних об'єктів з можливістю вставляти їх у обхідну сцену. Основними перевагами цієї розробки є:

– використання компонентного підходу React дозволяє легко масштабувати проєкт та підтримувати розширюваність (наприклад, додавання нових панелей, типів об'єктів тощо);

– завдяки вебстеку розробка виконується значно швидше порівняно з C++ або C#, що використовуються в аналогах (Tiled, LDtk);

– сцена формується як структура з об'єктами, що легко експортується у формат JSON для подальшої інтеграції з ігровим рушієм;

– висока гнучкість архітектури, що досягається шляхом реалізації системи шарів, drag-and-drop редагування, базової підтримки undo/redo, інтеграції з Pixi.js як візуальним рушієм.

– платформонезалежність, оскільки Electron дозволяє запускати редактор на Windows, macOS та Linux без необхідності специфічної компіляції.

Таблиця 1

Порівняння власного редактору з аналогами

Редактор	Платформа	Мова / Технології	Основні особливості	Недоліки
Tiled	Десктоп (Qt)	C++	Підтримка тайлів, інтеграція з рушіями, JSON/TSX формати.	Важка інтеграція кастомних об'єктів, складна система плагінів .
LDtk	Десктоп (MonoGame)	C#	Тайли, автошари, вбудовані правила генерації рівнів.	Вузька орієнтація на платформери, менша гнучкість щодо об'єктів.
Ogmo Editor	Десктоп (Electron)	TypeScript	Проста структура, легке збереження в JSON, орієнтований на інді.	Відсутність візуальної ієрархії, обмежена кількість налаштувань.
Власна розробка Editor	Десктоп (Electron)	React.js + Pixi.js	Система шарів, drag-and-drop, JSON, undo/redo, відкритість до розширення.	Потребує подальшого розвитку (анімації, інтеграція з рушіями, GUI для подій).

Висновки

У роботі здійснено аналітичний огляд сучасних інструментів та підходів до створення графічних редакторів для редагування сцен у 2D-відеоіграх. Здійснено порівняльний аналіз переваг, недоліків та обмежень ряду популярних графічних редакторів, що дозволило обґрунтувати вибір архітектурного рішення для створення власного інструменту.

Визначено архітектурні принципи, на яких базується побудова ефективних редакторів, зокрема використання об'єктно-орієнтованого підходу, системи шарів та модульного інтерфейсу. Проаналізовано рушії, використовувані у редакторах, особливу увагу приділено Pixi.js як базовому рушію у контексті інді-розробки.

На основі виконаних досліджень створено власну версію графічного редактору для 2D-сцен з розширеними функціональними можливостями, який успішно протестовано та представлено як прототип для подальшого вдосконалення.

СПИСОК ЛІТЕРАТУРИ

1. Statista. Share of game developers working on 2D games worldwide. 2023. URL: https://www.statista.com/statistics/1250457/2d-vs-3d-games-indie-developers.

2. Generative Adversarial Nets / I. Goodfellow et al. Advances in Neural Information Processing Systems 27, 2014. P. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

3. Karras T., Aila T., Samuli Laine S., Lehtinen J. Progressive Growing of GANs for Improved Quality, Stability, and Variation. *International Conference on Learning Representations (ICLR)*. 2018. P. 1–26. URL: https://doi.org/10.48550/arXiv.1710.10196.

4. Radford A., Metz L., Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434*. 2015. URL: https://arxiv.org/abs/1511.06434.

5. StackGAN++: Realistic image synthesis with stacked generative adversarial networks / H. Zhang et al. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2018. №41 (8). P. 1947–1962. URL: https://doi.org/10.1109/TPAMI.2018.2856256.

6. Brock A., Donahue J., Simonyan K. Large scale GAN training for high fidelity natural image synthesis. *arXiv:1809.11096*. 2018. URL: https://arxiv.org/abs/1809.11096.

7. Karras T., Laine S., Aila T. A style-based generator architecture for generative adversarial networks *arXiv:1812.04948*. 2018. URL: https://arxiv.org/abs/1812.04948.

8. LayoutGAN: Generating graphic layouts with wireframe discriminators / J. Li et al. *In Proc. 7th International Conference on Learning Representations (ICLR),* New Orleans, LA, USA. 2019. URL: https://arxiv.org/abs/1901.06767.

9. Pixi.js Examples. URL: https://pixijs.io/examples/.

10. Mozilla Developer Network. HTML Drag and Drop API. URL: https://developer.mozilla.org/en-US/docs/Web/API/HTML_Drag_and_Drop_API.

11. Ігровий движок Unity: що це таке та на що він здатний? URL: https://itproger.com/ua/news/igrovoy-dvizhok-unity-razbiraemsya-chto-k-chem#google_vignette.

12. Не соромно запитати: як працює Unreal Engine. URL: https://skvot.io/uk/blog/ne-soromno-zapitati-yak-pracyuye-unreal-engine.
13. Lindeijer Th. Tiled: A flexible level editor. URL: https://www.mapeditor.org.
14. Bénard S. LDtk – Level Designer Toolkit. URL: https://ldtk.io.
15. Ogmo Editor. Open-source level editor for 2D games. URL: https://ogmo-editor-3.github.io.

Стаття надійшла до редакції 27.04.2025.
Стаття пройшла рецензування 16.06.2025.

Складанюк Олексій Олегович – аспірант кафедри програмного забезпечення**.**

Майданюк Володимир Павлович – канд. тех. наук, доцент кафедри комп'ютерних наук**.**

Арсенюк Ігор Ростиславович – канд. тех. наук, доцент кафедри комп'ютерних наук, e-mail: air@vntu.edu.ua**.**

Вінницький національний технічний університет.