УДК 514.18

МОДЕЛЬ АДАПТИВНОЇ СИСТЕМИ ЕФЕКТІВ РУЙНУВАННЯ МАТЕРІАЛІВ ПРИ СТВОРЕННІ ІГРОВОГО СЕРЕДОВИЩА НА БАЗІ UNREAL ENGINE

DOI: 10.33842/2313-125X-2026-29-276-285
Сімонова О.Г., канд. техн. наук,
xoro37@ukr.net, ORCID: 0009-0005-4992-4970,
Федченко Г.В., канд. техн. наук,
anna-fedchenko@ukr.net, ORCID: 0000-0003-0690-6017
Охотська О.В.,
lenaohotskaya@gmail.com, ORCID: 0009-0005-8052-9661
Омельченко Я.В.,
Yana.Omelchenko@infiz.khpi.edu.ua
Національний технічний університет «Харківський політехнічний інститут» (м. Харків, Україна)

В роботі досліджується актуальна тема, що зумовлена зростаючими вимогами до візуальної якості та продуктивності в інтерактивних середовищах. У сучасному геймдеві важливо забезпечити правдоподібність фізичних процесів – зокрема, ефектів руйнування – без перевищення обчислювального бюджету кадру. Надмірне навантаження на систему призводить до зниження FPS, що критично для ігрового досвіду. Об'єктом дослідження є інтерактивне 3D-середовище з динамічними ефектами. Предметом дослідження є методи моделювання, симуляції та оптимізації ефектів руйнування в рушії Unreal Engine 5 [1]. Наукова новизна роботи полягає в створенні математичної моделі для оцінки обчислювальної складності візуальних ефектів та її інтеграції у рушій Unreal Engine для забезпечення продуктивності при високій візуальній якості.

Практичне значення дослідження визначається питаннями реалістичності й фізичної правдоподібності ігрових світів, що набуває особливої ваги. Гравці очікують не лише красиву графіку, але й правдоподібну поведінку середовища, у якому вони взаємодіють. Один із ключових елементів цієї взаємодії – руйнування об'єктів: чи то розбита скляна вітрина, проламана дерев'яна дошка, або уламки після вибуху. Відображення фізичних властивостей матеріалів у реальному часі – складне технічне завдання, особливо коли мова йде про інтерактивність, тобто можливість гравця безпосередньо впливати на руйнування об'єктів. Це включає як візуальні, так і фізичні наслідки: анімацію розлому, появу уламків, частинок, зміну геометрії об'єкта та його стану у грі. У реальному світі руйнування – результат перевищення меж міцності матеріалу. У віртуальному – це необхідність імітувати це перевищення

швидко, переконливо, і без втрат у продуктивності. З огляду на високі обчислювальні витрати, що пов'язані з симуляцією фізики, у більшості випадків потрібен компроміс між точністю моделювання та швидкодією. Особливо це критично для складних сцен з великою кількістю динамічних об'єктів, як у бойових або екшен-іграх.. Результатом дослідження стала реалізація системи адаптивного керування ефектами руйнування з урахуванням обмежень реального часу, а також створення повноцінного віртуального середовища із 3D-персонажем, сценами та візуальними ефектами, виконаними у Blender [2] і Unreal Engine [3].

Ключові слова: Unreal Engine 5, ефекти руйнування, геймдизайн, тривимірне моделювання, математична модель, Niagara

Постановка проблеми. Симуляція фізичних процесів у реальному часі – одна з найскладніших задач в інтерактивному 3D-середовищі. На відміну від наукових або інженерних симуляцій, які можуть тривати годинами на потужних комп'ютерах, відеогра повинна стабільно функціонувати в межах жорстких обчислювальних обмежень. Ефекти руйнування створюють додаткове навантаження: об'єкт потрібно візуально модифікувати (розділити на уламки), надати кожному з фрагментів фізичні властивості (маса, інерція, колізії), запустити візуальні ефекти (частинки, пил, освітлення), синхронізувати все це з логікою гри. Якщо на сцені одночасно руйнуються кілька об'єктів – це може легко перевищити допустимий обчислювальний бюджет кадру. Особливо складною є ситуація у масштабних сценах, де присутні десятки або сотні потенційно руйновних елементів: меблі, стіни, вікна, декоративні об'єкти. У таких випадках ризик втрати продуктивності або навіть аварійного завершення програми зростає в рази. Щоб уникнути подібних проблем, у сучасному геймдеві широко використовуються наближені моделі замість повноцінної симуляції.

Саме тому виникає потреба у математичній моделі, яка не просто імітує фізику, а ще й контролює ресурсне навантаження. Вона повинна оцінювати кожен ефект за кількома параметрами (візуальна складність, фізика, логіка) і приймати рішення: запускати повний ефект, спростити його або відкласти до наступного кадру.

Аналіз останніх досліджень і публікацій. В останні роки значна увага приділяється покращенню реалізму візуалізацій через використання передових алгоритмів Так, щоб підтримувати комфортні 60 кадрів на секунду (FPS), рушій має завершити всі розрахунки – фізику, графіку, логіку, UI, звук – за 16.67 мс на кожен кадр [4]. Через обмеження продуктивності, більшість ігрових рушіїв застосовують спрощені моделі [5], які не обчислюють сили, напруження чи деформації в реальному часі. Натомість використовується система тригерів [6] – наприклад, Raycast-перевірка зіткнення, взаємодія з Trigger Volume або аналіз швидкості об'єкта перед ударом. Коли гравець чинить надмірний тиск, активується

ефект деструкції. У грі це реалізується через запуск Niagara-частинок, що імітують розрив [7]. Усе це створює відчуття реалістичності при мінімальному використанні ресурсів.

Проблеми та перспективи розвитку. Основними проблемами у сфері 3D моделювання залишаються високі вимоги до аппаратного та программного забезпечення. Перспективи розвитку спрямовані на оптимізацію ефектів руйнування та моделювання цих процесів. Це є новими технологічними рішеннями для майбутнього розвитку галузі.

Формулювання цілей статті. Метою роботи є створення математичної моделі для оцінки обчислювальної складності візуальних ефектів та її інтеграція у рушій Unreal Engine для забезпечення продуктивності при високій візуальній якості.

Основна частина. У відеоіграх, на відміну від інженерного моделювання, мета симуляції фізичних процесів полягає не в досягненні точності, а у створенні переконливого візуального результату за мінімального використання ресурсів. Одним з таких процесів є руйнування об'єктів – динамічна зміна стану матеріалів у відповідь на дії гравця або події у грі. У реальному світі руйнування матеріалів відбувається внаслідок перевищення межі міцності під дією зовнішніх навантажень. Для його моделювання застосовуються метод скінченних елементів (FEM), дискретних елементів (DEM) або молекулярної динаміки (MD). Проте використання цих методів у геймдеві є недоцільним через їхню надмірну обчислювальну складність і неможливість виконання в реальному часі. Ігрові рушії, зокрема Unreal Engine, замінюють точну симуляцію на систему правил і тригерів, які дозволяють створити ілюзію фізики. Основними параметрами, які враховуються при реалізації системи руйнування, є порогова міцність, тип матеріалу, стан після руйнування, Оптимізація цих параметрів повинна створювати відчуття реалістичності при мінімальному використанні ресурсів.

Реалізація ефектів руйнування в інтерактивних середовищах потребує балансу між правдоподібністю та ефективністю. Оскільки точна симуляція в реальному часі надто витратна, розробники ігор застосовують різноманітні підходи, кожен з яких має свої особливості, переваги та обмеження. Вибір методу залежить від платформи, типу гри, масштабу сцени, а також від ролі об’єкта в ігровому процесі.

Гра реалізується з виглядом від третьої особи, візуальна складова моделі персонажа є ключовою – гравець постійно бачить тіло героя під час руху, атаки та взаємодії з середовищем. Головного персонажа гри було створено в середовищі Blender (рис. 1).

Рис. 1. Готова модель в режимі Material Preview Front і Back сторони

Створення ландшафту проводилося в середовищі Unreal Engine 5 (рис. 2).

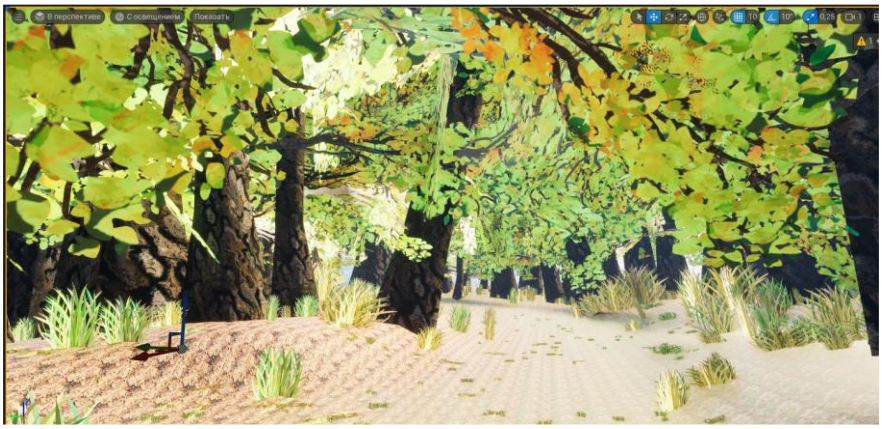

Рис. 2. Готова частина карти ландшафту

Оскільки ключовим елементом є демонстрація ефектів руйнування, постала необхідність створити базову систему взаємодії з об'єктами сцени.

Перший крок – створення нового C++ класу, який отримав назву CharAliza, оскільки він описує поведінку персонажа та всі пов’язані з ним дії. Після генерації класу в середовищі Visual Studio автоматично відкриваються два файли: CharAliza.cpp і CharAliza.h. У файлі CharAliza.cpp (рис. 3) відбувається безпосередня реалізація логіки. В конструкторі створюються компоненти камери, встановлюється обертання відповідно до миші та вмикається оновлення кожного кадру.

У функції SetupPlayerInputComponent() задаються прив’язки клавіш: WASD – для руху, мишка – для огляду, E – для взаємодії.

Функція Interact() відповідає за просте трасування в напрямку погляду гравця. Якщо промінь влучає у якийсь об'єкт – перевіряється, чи можна з ним взаємодіяти, і далі виконується дія (наприклад, знищення

об’єкта через Destroy()). Також використано візуалізацію лінії променя (DrawDebugLine) для зручності тестування.

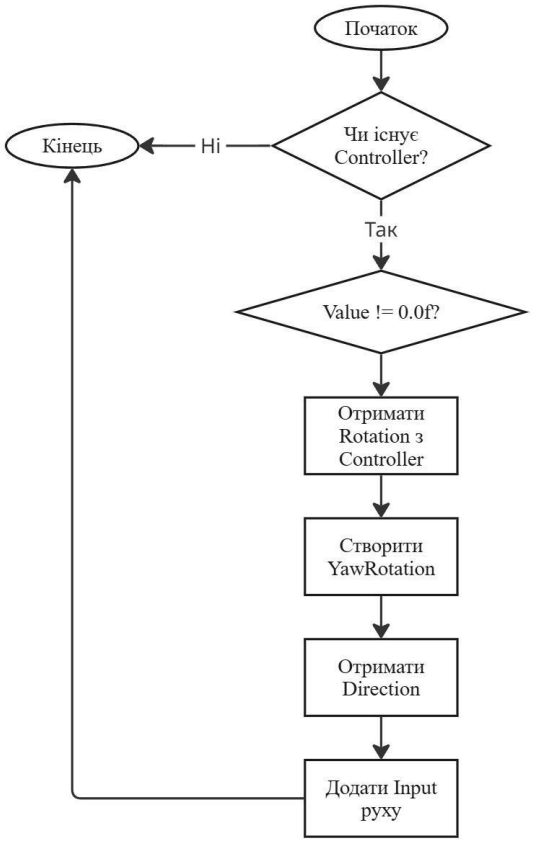

Рис. 3. Блок-схема CharAliza.cpp

Файл заголовку (CharAliza.h) містить оголошення компонентів, які відповідають за камеру та стрілку, а також сигнатури функцій для переміщення (вперед/назад і вліво/вправо), повороту огляду мишкою та виклику взаємодії. Камера прив’язана до стрілки, яка, в свою чергу, прикріплена до тіла персонажа – така схема дозволяє м’яко слідкувати за гравцем у стилі від третьої особи.

Загалом цей клас дозволяє створити базову систему взаємодії з елементами сцени — і саме через неї реалізується демонстрація ефектів руйнування в Unreal Engine 5.

Після створення усіх необхідних елементів гри, наступним етапом є налаштування ігрового режиму та забезпечення можливості управління персонажем одразу після запуску гри.

У процесі розробки інтерактивних середовищ – таких як відеоігри, тренажери або віртуальні презентації – моделювання фізичних властивостей об'єктів є важливим чинником для підвищення рівня реалізму. Особливо значущою є поведінка матеріалів під час механічного впливу: деформації, розтріскування, розриву та повного руйнування.

У класичних фізичних симуляціях (наприклад, інженерних системах) такі процеси моделюються з високою точністю – методом скінченних

елементів (Finite Element Method), дискретних елементів (Discrete Element Method) або молекулярної динаміки. Проте ці методи є вкрай ресурсомісткими і не можуть бути застосовані в реальному часі у рамках стандартного ігрового рушія.

Настурне завдання – створити наближену до реальності модель руйнування, яка буде працювати в режимі реального часу з обмеженим обчислювальним бюджетом. Нехай маємо віртуальне середовище, у якому певні об'єкти (наприклад, дерев'яна балка, скляна вітрина або паперовий лист) підлягають руйнуванню внаслідок дій користувача або внутрішньоігрових подій.

Кожен ефект руйнування (надалі – ефект) повинен задовольняти наступним умовам:

1) активуватися лише у визначених умовах, пов’язаних з силовим впливом або тригерами;

2) оцінюватися за критеріями навантаження на систему (тобто наскільки він "дорогий" для комп’ютера);

3) автоматично або вручну підлягати адаптації: використовувати спрощений варіант, відкладатись або взагалі скасовуватись при дефіциті ресурсів;

4) підлягати програмному контролю з боку ігрового рушія.

У моделі, яка икористовується в ігровому рушії Unreal Engine 5, необхідно врахувати такі фактори:

- фізичні властивості матеріалу: чи є він крихким, еластичним, пластичним тощо;
- візуальні наслідки: уламки, пил, анімація;
- логіка: зміна стану об'єкта, вплив на геймплей;
- обчислювальна вартість: наскільки багато ресурсів потрібно на рендеринг і обчислення ефекту;
- пріоритет виконання: чи є ефект критично важливим, чи його можна тимчасово ігнорувати.

У середньостатистичній грі кадр рендериться протягом 16.67 мс (для 60 FPS). Якщо в момент запуску ефекту виконується надто багато розрахунків або рендеринг занадто складний – продуктивність падає, ігровий процес сповільнюється, візуально з’являються «фрізи» або затримки.

Модель повинна не лише правильно описувати поведінку матеріалу, але й бути здатною динамічно реагувати на навантаження. Реалістичне відображення процесів руйнування у відеоіграх потребує не лише візуальної достовірності, а й ефективного використання ресурсів обчислювальної системи. Щоб забезпечити стабільну частоту кадрів, необхідно створити адаптивну модель, яка дозволяє оцінювати ефекти за ступенем їх впливу на продуктивність та керувати їх активацією в залежності від поточного навантаження.

Концепція моделі базується на розбитті кожного ефекту на три

основні компоненти:

- G (Geometry) – геометричне навантаження: кількість полігонів, фізичні об’єкти, модифікація сітки;
- V (Visuals) – візуальні ефекти: частинки, шейдери, освітлення, прозорі матеріали;
- L (Logic) – логічна складність: скрипти, події, обробка колізій, звукові ефекти.

Кожному ефекту призначаються числові ваги відповідно до цих параметрів. Це дозволяє порівнювати ефекти між собою та приймати рішення про їх запуск, спрощення або відкладення в рамках одного кадру. Загальна вартість і-го ефекту C_i розраховується за формулою (1).

$$C_i = \alpha \cdot G_i + \beta \cdot V_i + \gamma \cdot L_i, \quad (1)$$

де:

G_i, V_i, L_i – ваги по кожному з трьох параметрів для i–го ефекту;

α, β, γ– вагові коефіцієнти (налаштовуються під конкретне середовище експериментально)

Загальна умова запуску наведена в формулі (2).

$$\sum_{i=1}^{n} C_i \le B, \quad (2)$$

де B – ресурсний бюджет одного кадру (наприклад, 16.67 мс на кадр).

Загалом цей клас дозволяє створити базову систему взаємодії з елементами сцени – і саме через неї реалізується демонстрація ефектів руйнування в Unreal Engine 5. Після створення усіх необхідних елементів гри, наступним етапом є налаштування ігрового режиму та забезпечення можливості управління персонажем одразу після запуску гри.

Для цього потрібно створити Blueprint-клас, наприклад, BP_CharAliza, що наслідує від класу GameModeBase. У налаштуваннях цього класу слід перейти до вкладки Class Defaults (Параметри класу) та в полі Default Pawn Class (Клас персонажа за замовчуванням) вказати створений C++ клас CharAliza. Це забезпечує автоматичне використання власного персонажа з усім необхідним функціоналом при запуску гри.

Для впровадження адаптивної системи ефектів руйнування на основі моделі G/V/L було реалізовано власну архітектуру керування ефектами у рушії Unreal Engine 5. Головна мета – забезпечити можливість динамічного контролю за запуском ефектів відповідно до поточного навантаження сцени. Система побудована на класичній патерні "менеджера", який щокадрово перевіряє чергу запланованих ефектів і приймає рішення: активувати ефект, відкласти його або відмовитися від запуску. Основні компоненти:

- менеджер ефектів (UDestructionManager) – головний клас, який керує обробкою ефектів;
- черга ефектів (QueuedEffects) — список запитів, сформованих у процесі гри;
- оцінка вартості — обчислення навантаження на кадр через функцію

ComputeCost;

- механізм обмеження — перевірка, чи загальна вартість не перевищує бюджет кадру (16.67 мс для 60 FPS);
- активація / відкладення – ефект або запускається, або переноситься на наступний кадр.

Розглянемо реалізацію логіки основного класу UDestructionManager (Менеджер ефектів). Він містить метод Tick(), який виконується кожен кадр та оцінює сумарне навантаження і приймає рішення щодо запуску кожного з ефектів у черзі (рис. 4).

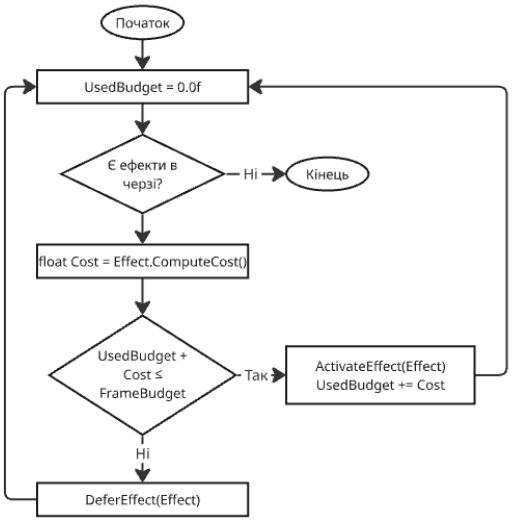

Рис. 4. Блок-схема виконання функції Tick() у системі керування ефектами руйнування

Цей механізм дозволяє уникати перевантаження під час запуску великої кількості ефектів одночасно. Під час події в грі (наприклад, коли гравець наступає на дерев'яну дошку), система формує ефект руйнування дерев'яної дошки із відповідними параметрами:

```
FDestructionEffect WoodCrackEffect;
WoodCrackEffect.GeometryWeight = 6.0f;        // Fracture Mesh + колізії
WoodCrackEffect.VFXWeight = 3.0f;              // Частинки пилу
WoodCrackEffect.LogicWeight = 1.0f; // Звук, анімація
DestructionManager->EnqueueEffect(WoodCrackEffect);
```

Цей ефект потрапляє в чергу і буде оброблений на найближчому кадрі залежно від доступного бюджету продуктивності.

Тестування включало всебічну перевірку системи керування ефектами руйнування.

Висновки. В рамках роботи було проведено аналітичний огляд сучасних підходів до реалізації ефектів руйнування в інтерактивних застосунках, визначено їх ключові переваги та обмеження. На основі цього аналізу сформовано формальну математичну модель оцінки ефектів за

трьома компонентами: геометричне навантаження, візуальні ефекти та логічна складність. Запропонована модель дозволяє адаптивно керувати запуском ефектів залежно від обчислювального бюджету кадру, що сприяє підвищенню загальної продуктивності системи.

Окрему увагу приділено створенню 3D-моделі персонажа у середовищі Blender із подальшою інтеграцією в рушій Unreal Engine 5. Враховано вимоги до анімаційної підготовки, топології та сумісності з системами рендерингу й фізики.

Отримані результати демонструють можливість ефективного поєднання високої якості візуалізації з оптимальним використанням обчислювальних ресурсів, що є актуальним завданням у сучасній інді-розробці, геймінгу та симуляційних системах.

Література

1. Epic Games. Unreal Engine 5 Documentation [Electronic resource]. URL: https://docs.unrealengine.com/5.0/en-US/ (accessed: 22.05.2026).
2. Blender Foundation. Blender Manual [Electronic resource]. URL: https://docs.blender.org/manual/en/latest/ (accessed: 22.05.2026).
3. Smith J. Game Physics Engine Development. Boca Raton : CRC Press, 2020. 432 p.
4. NVIDIA Developer Blog [Electronic resource]. URL: https://developer.nvidia.com/blog/ (accessed: 22.05.2026).
5. Бондаренко С. В. Фізичне моделювання в ігрових рушіях. Харків : ХНУРЕ, 2021. 184 с.
6. Epic Games. Unreal Engine Documentation [Electronic resource]. URL: https://docs.unrealengine.com/ (accessed: 22.05.2026).
7. Epic Games. Unreal Engine Niagara VFX Quick Start [Electronic resource]. URL: https://docs.unrealengine.com/Niagara (accessed: 22.05.2026).

MODEL OF AN ADAPTIVE SYSTEM OF MATERIAL DESTRUCTION EFFECTS WHEN CREATING A GAME ENVIRONMENT BASED ON UNREAL ENGINE

Olha Simonova, Hanna Fedchenko, Olena Okhotska, Yana Omelchenko

The work explores a topical topic driven by increasing demands for visual quality and performance in interactive environments. In modern game development, it is important to ensure the plausibility of physical processes – in particular, destruction effects – without exceeding the computational budget of the frame. Excessive load on the system leads to a decrease in FPS, which is critical for the gaming experience. The object of the study is an interactive 3D environment with dynamic effects. The subject of the study is methods for modeling, simulating, and optimizing destruction effects in the Unreal Engine 5 engine [1]. The scientific novelty of the work lies in the creation of a mathematical model for assessing the computational complexity of visual effects and its integration into the Unreal Engine engine to ensure performance with

high visual quality.

The practical significance of the study is determined by the issues of realism and physical plausibility of game worlds, which is gaining particular importance. Players expect not only beautiful graphics, but also the believable behavior of the environment in which they interact. One of the key elements of this interaction is the destruction of objects: whether it is a broken glass display case, a cracked wooden board, or fragments after an explosion. Displaying the physical properties of materials in real time is a difficult technical task, especially when it comes to interactivity, that is, the ability of the player to directly influence the destruction of objects. This includes both visual and physical consequences: animation of a fracture, the appearance of fragments, particles, changing the geometry of the object and its state in the game. In the real world, destruction is the result of exceeding the strength limits of the material. In the virtual world, it is the need to simulate this excess quickly, convincingly, and without loss of performance. Given the high computational costs associated with physics simulation, in most cases a compromise is required between modeling accuracy and speed. This is especially critical for complex scenes with a large number of dynamic objects, such as in fighting or action games. The result of the research was the implementation of a system for adaptive control of destruction effects taking into account real-time constraints, as well as the creation of a full-fledged virtual environment with a 3D character, scenes and visual effects made in Blender [2] and Unreal Engine [3].

Keywords: Unreal Engine 5, destruction effects, game design, three-dimensional modeling, mathematical model, Niagara.

Referenses

1. Epic Games. Unreal Engine 5 Documentation. URL: https://docs.unrealengine.com/5.0/en-US/ (accessed: 22.05.2026) [In English].
2. Blender Foundation. Blender Manual. URL: https://docs.blender.org/manual/en/latest/ (accessed: 22.05.2026) [In English].
3. Smith, J. (2020). Game physics engine development. Boca Raton: CRC Press, 432 p. [In English].
4. NVIDIA Developer Blog. URL: https://developer.nvidia.com/blog/ (accessed: 22.05.2026) [In English].
5. Bondarenko, S. V. (2021). Physical modeling in game engines. Kharkiv: KhNURE, 184 p. [In Ukrainian].
6. Epic Games. Unreal Engine Documentation. URL: https://docs.unrealengine.com/ (accessed: 22.05.2026) [In English].
7. Epic Games. Unreal Engine Niagara VFX Quick Start. URL: https://docs.unrealengine.com/Niagara (accessed: 22.05.2026) [In English].

Матеріал надійшов до редакції 26.04.2026

Прийнято до друку 13.05.2026 р.