Панасюк Андрій Сергійович магістр з інженерії програмного забезпечення, Одеса, Україна

Пригожев Олександр Сергійович, кандидат технічних наук, доцент, доцент кафедри інженерії програмного забезпечення Національного університету «Одеська політехніка», Одеса, Україна https://orcid.org/0000-0001-8532-8897

Panasyuk Andriy, Master of science in software engineering, Odesa, Ukraine

Prygozhev Oleksandr, Candidate of technical sciences, Assosiate Professor, Associate Professor in Software Engeneering Department Odessa Polytechnic National University, Odessa, Ukraine https://orcid.org/0000-0001-8532-8897

ПРОГРАМНА СИСТЕМА ДЛЯ СИМУЛЯЦІЇ ФІЗИЧНИХ ЕФЕКТІВ У МЕХАНІЦІ ІГОР

SOFTWARE SYSTEM FOR SIMULATING PHYSICAL EFFECTS IN GAME MECHANICS

Abstract

The aim of the work is to reduce the time for developing and implementing physical effects in game projects by creating a software system that will provide modeling of key physical phenomena in an interactive environment.

The development methods are based on Python as the main programming language, as well as on the use of the Django framework. The code for integrating physical simulations can also be generated in JavaScript, Python and C# using the Three.js, Chart.js, etc. libraries. The PostgreSQL relational database

is used to store data. The StarUML software development tool was used to design the PhysiXLab system.

As a result of the work, a software system for modeling physical phenomena was developed, which allows you to create simulations, obtain real-time visualization, and generate code in Python, JavaScript, and C#.

Keywords: software system, physical simulation, Python, JavaScript, C#, NumPy, SciPy, Three.js, Chart.js, PostgreSQL database, StarUML.

Анотація

Метою роботи є скорочення часу на розробку і впровадження фізичних ефектів у ігрові проєкти шляхом створення програмної системи, яка забезпечить моделювання ключових фізичних явищ у інтерактивному середовищі.

Методи розробки базуються на Python як основній мові програмування, а також на використанні фреймворку Django. Код для інтеграції фізичних симуляцій також може бути генерований на мовах JavaScript, Python та C# із використанням бібліотек Three.js, Chart.js та інших. Для зберігання даних використовується реляційна СУБД PostgreSQL. Для проєктування системи PhysiXLab використано інструмент розробки програмного забезпечення StarUML.

Як результат роботи розроблено програмну систему для моделювання фізичних явищ, яка дозволяє створювати симуляції, отримувати візуалізацію в реальному часі та генерувати код мовами Python, JavaScript і C#.

Ключові слова: програмна система, фізична симуляція, Python, JavaScript, C#, NumPy, SciPy, Three.js, Chart.js, СУБД PostgreSQL, StarUML.

Постановка проблеми. У сучасному світі ігрова індустрія посідає одну з провідних позицій серед найбільш динамічних і прибуткових сфер інформаційних технологій. Щороку цей ринок зростає, залучаючи мільйони користувачів і генеруючи мільярдні доходи. Популярність ігор обумовлена їх здатністю пропонувати інтерактивний досвід, який поєднує в собі розваги, виклики та емоційне залучення. Розробка таких ігор вимагає від фахівців міждисциплінарного підходу, який включає елементи мистецтва, дизайну, програмування, а також застосування новітніх технологій. Одним із найбільш важливих аспектів у цьому процесі є створення реалістичних фізичних симуляцій, які дозволяють іграм виглядати правдоподібно і природно.

Фізичні ефекти є невіддільною частиною сучасних ігор, оскільки вони забезпечують логічну та візуальну реалістичність. Від способу падіння персонажа з висоти до складних симуляцій взаємодії об'єктів – фізика відіграє ключову роль у формуванні ігрового світу. Її використання робить ігри більш захоплюючими, сприяє створенню інтерактивного середовища, яке відчувається «живим» і наближеним до реального світу. Наприклад, симуляція води, диму, вітру або зіткнень об'єктів дозволяє гравцям не лише спостерігати за процесами, але й активно взаємодіяти з ними, що значно підвищує рівень занурення.

Симуляція фізичних ефектів у механіці ігор забезпечує відтворення реалістичної поведінки об'єктів у просторі з урахуванням законів фізики: таких як гравітація, тертя, пружність, зіткнення та інші фундаментальні явища [1]. Це дозволяє створювати сцени, які логічно відповідають очікуванням користувача, і тим самим зміцнювати його емоційний зв’язок із грою. Наприклад, якщо гравець підкидає об’єкт у повітря, його рух і падіння повинні виглядати природно. Такі ефекти підвищують довіру до ігрового світу та роблять його більш привабливим для гравців.

Крім того, фізика в іграх використовується не тільки для реалізації реалістичної поведінки персонажів або об'єктів, а й для створення унікальних ігрових механік, які додають глибини і різноманіття ігрового процесу. Наприклад, у деяких іграх гравцям необхідно використовувати фізичні закони для вирішення головоломок, таких як балансування об'єктів, створення мостів або керування потоками води [2]. У шутерах фізика часто використовується для моделювання траєкторій куль і гранат, а в спортивних симуляторах – для передбачення руху м'яча або поведінки гравців на полі. Такі механіки створюють нові виклики та способи взаємодії, що сприяє тривалому інтересу до гри.

Таким чином, фізична симуляція є важливою складовою сучасної ігрової індустрії. Її застосування дозволяє поєднати реалістичність, захоплюючий ігровий досвід і можливість створювати нові, унікальні ігрові світи. Завдяки фізиці ігри перетворюються з простих розваг на інтерактивні середовища, які пропонують гравцям безмежні можливості для дослідження, навчання та розваг.

Аналіз останніх досліджень і публікацій. Розробка системи для симуляції фізичних ефектів у механіці ігор вимагає детального аналізу існуючих програмних аналогів, які використовуються для вирішення подібних завдань. Розглянемо існуючі бібліотеки та рушії фізики.

Box2D є однією з найпопулярніших бібліотек для 2D-фізики. Вона широко використовується у мобільних іграх, головоломках і платформерах завдяки своїй ефективності й простоті інтеграції [3]. Бібліотека пропонує базові фізичні моделі, такі як гравітація, зіткнення, тертя, а також обробку пружних і непружних взаємодій. Основними перевагами Box2D є висока продуктивність і легкість у використанні. Проте її можливості обмежені роботою виключно з 2D-середовищем, що унеможливлює використання для складних 3D-симуляцій.

Bullet Physics – це потужний інструмент для моделювання фізики у 3D [4]. Ця бібліотека використовується в багатьох комерційних ігрових рушіях, таких як Unreal Engine, і пропонує широкий набір функцій для моделювання динаміки твердих тіл, зіткнень, пружних і непружних взаємодій, а також моделювання тканин і рідин . Однією з переваг Bullet є її модульність, яка дозволяє адаптувати бібліотеку під конкретні потреби. Водночас, висока складність налаштування може стати перешкодою для менш досвідчених розробників.

Matter.js – це JavaScript-бібліотека для 2D-фізики, орієнтована на використання в вебзастосунках [5]. Вона надає широкий спектр інструментів для роботи з динамічними об'єктами, включаючи обробку гравітації, зіткнень і тертя. Основна перевага Matter.js – її інтеграція з вебтехнологіями, що забезпечує легку роботу у браузері без потреби у встановленні додаткового програмного забезпечення. Проте бібліотека має обмеження у продуктивності для складних сцен з великою кількістю об'єктів.

Ammo.js є портом Bullet Physics для JavaScript, що дозволяє виконувати фізичні симуляції у 3D-просторі безпосередньо у браузері [6]. Завдяки використанню WebAssembly, бібліотека забезпечує високу продуктивність і підтримує складні моделі, такі як деформації, рідини та взаємодії твердих тіл. Однак складність у налаштуванні й великий розмір бібліотеки можуть бути перешкодою для використання в простих проєктах.

Незважаючи на широкий вибір інструментів, більшість існуючих рішень має певні обмеження: залежність від конкретних платформ чи рушіїв (наприклад, Bullet Physics потребує додаткових налаштувань для використання у вебсередовищі); складність інтеграції та налаштування для розробників із базовими знаннями; високі вимоги до апаратного забезпечення, особливо для складних симуляцій; брак гнучкості для

налаштування специфічних механік і функцій. Отже, існує проблема подальшого розвитку цих засобів.

Мета статті. Метою роботи є скорочення часу на розробку і впровадження фізичних ефектів у ігрові проєкти шляхом створення програмної системи, яка забезпечить моделювання ключових фізичних явищ у інтерактивному середовищі.

Виклад основного матеріалу дослідження. Інформаційна модель системи симуляції фізичних ефектів у механіці ігор відображає основні компоненти системи та взаємозв'язки між ними (рис. 1).

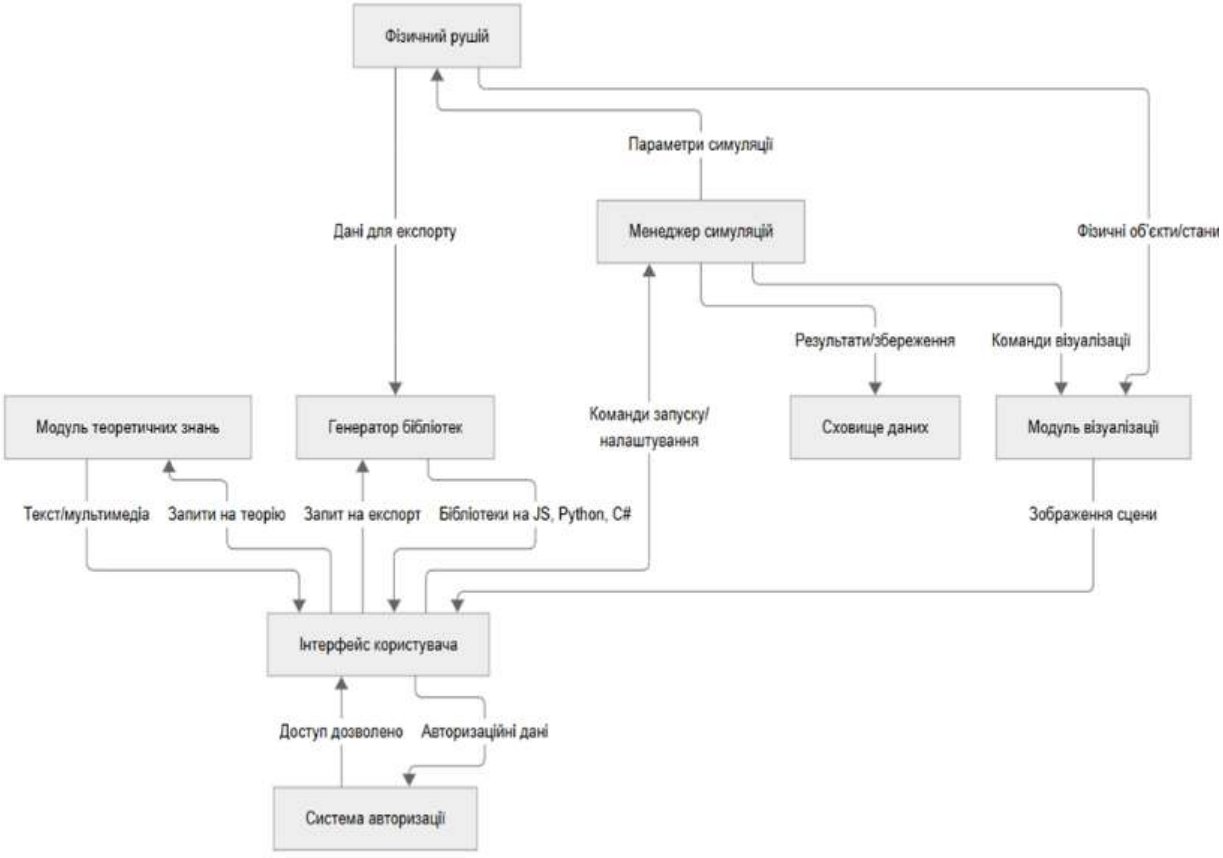


Рис. 1 Інформаційна модель системи симуляції

Центральною ланкою взаємодії є Інтерфейс користувача (UI), який забезпечує комунікацію між користувачем і внутрішніми модулями.

Користувач ініціює команди запуску симуляції та налаштування параметрів, які надходять до Менеджера симуляцій. Цей компонент відповідає за координацію симуляційного процесу: передає параметри до фізичного рушія (PhysicsEngine), керує візуалізацією через модуль Renderer та зберігає результати у сховище даних.

Реалістичність симуляцій у комп'ютерних іграх багато в чому залежить від того, наскільки точно в них змодельовано фізичні явища. Відтворення гравітації, тертя, зіткнень, обертання та інших ефектів дозволяє занурити гравця у світ, де об'єкти поводяться звично і передбачувано, а ігрова механіка будується на логічній та зрозумілій основі. Щоб забезпечити таку поведінку, кожен фізичний ефект описується відповідною фізичною моделлю, яка формалізує явище з використанням законів механіки, динаміки або опору середовища.

При цьому враховується, які з параметрів є змінними, а які – фіксованими протягом симуляції, а також чи підлягає ефект спрощенню без втрати правдоподібності. Такі рішення приймаються на етапі проектування, щоб зберегти баланс між візуальною достовірністю та продуктивністю.

Ця модель потім адаптується до програмної реалізації, що передбачає трансляцію математичних рівнянь та формул у логіку програмного коду. Конкретні параметри, такі як маса, швидкість, прискорення чи коефіцієнт тертя, реалізуються у вигляді змінних, тоді як закони взаємодії виражаються через функції, які оновлюють стан об'єктів на кожному кроці часу. Структури даних дозволяють описати складні об'єкти, зберігати їхні фізичні характеристики та координати, а також організовують обчислення з ефективним доступом до властивостей.

Окремо потрібно розглянути алгоритми для виявлення зіткнень. Ці алгоритми зазвичай використовується для запуску реакцій, таких як зміна швидкості, обертання або знищення об'єктів. Найпоширенішими методами

є Axis-Aligned Bounding Box (AABB) і перевірка точок зіткнення.

Алгоритм AABB використовується для перевірки перетинів між прямокутними об'єктами та включає наступні кроки: для кожного об'єкта визначаються межі (мінімальні та максимальні координати по кожній осі); виконується перевірте, чи перетинаються ці межі у просторі; якщо межі перекриваються, відбувається зіткнення.

Після виявлення зіткнення необхідно визначити, як об'єкти взаємодіють між собою. Реакція на зіткнення базується на фізичних принципах, таких як збереження імпульсу та енергії. Залежно від типу об'єктів і їхніх фізичних властивостей, реакція може включати зміну швидкості, обертання або руйнування.

Алгоритм реакції з урахуванням пружності включає наступні кроки: використовується закон збереження імпульсу для розрахунку нових швидкостей; застосовується коефіцієнт пружності (e) для визначення швидкості після зіткнення; оновлюються швидкості об'єктів відповідно до розрахованих значень.

Якщо об'єкти мають обертання, реакція включає зміну кутової швидкості. Це залежить від точки дотику та моменту інерції об'єктів.

Система має два основні типи акторів: користувач і розробник. Користувач взаємодіє із симуляціями, налаштовує параметри, переглядає результати та вивчає теоретичні основи фізичних процесів. Розробник відповідає за створення, оновлення та видалення симуляцій і теоретичних модулів, забезпечуючи актуальність і функціональність системи. Діаграма варіантів використання системи PhysiXLab наведена на рис. 2.

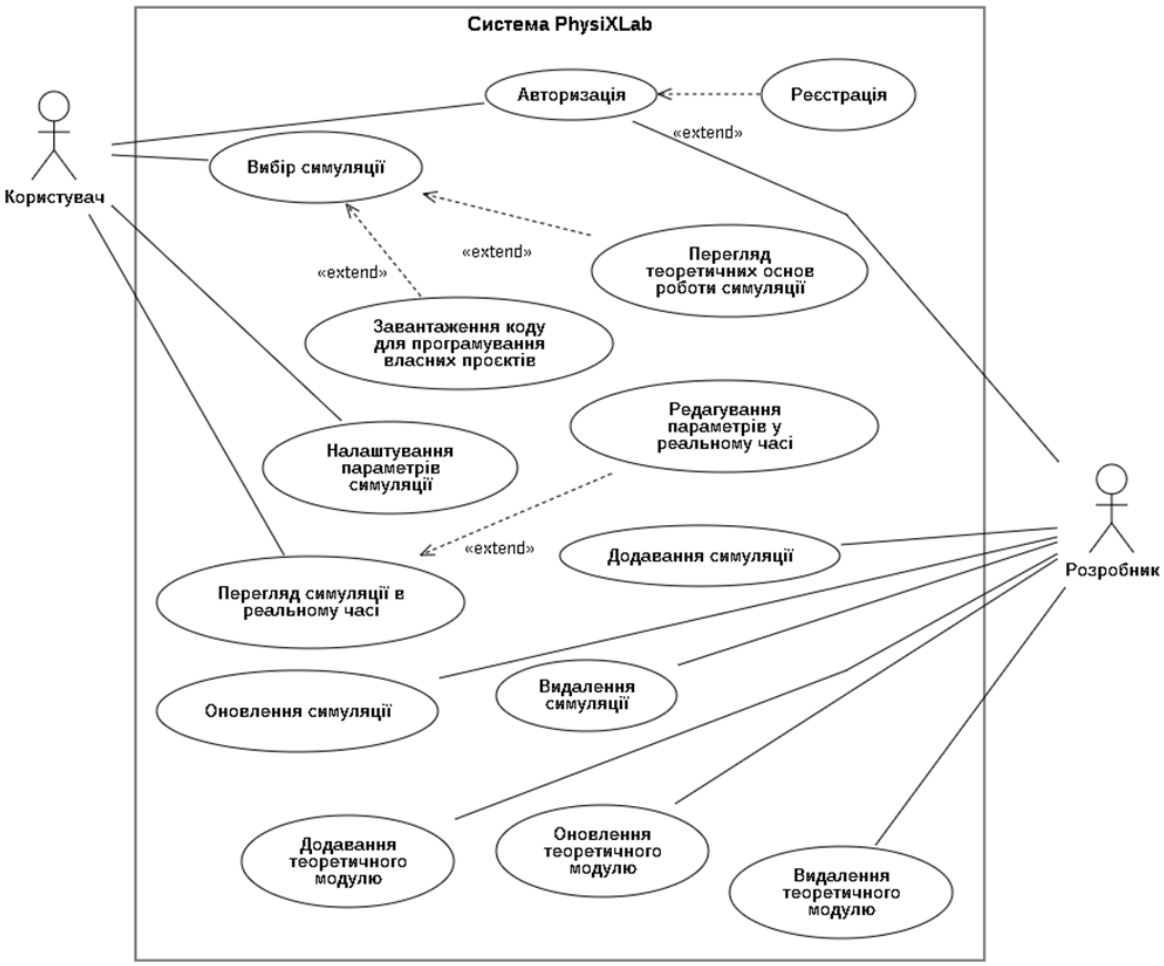


Рис. 2 Діаграма варіантів використання

Архітектура системи PhysiXLab з використанням шаблонів «Декоратор» і «Стратегія» представлена на рис. 3.

Користувач взаємодіє із системою через симуляції, які можуть налаштовуватися за допомогою параметрів та відображати результати. Додатково користувач має доступ до теоретичних модулів, що забезпечують розуміння фізичних принципів. Розробник, у свою чергу, управляє цими симуляціями та теоретичними модулями, додаючи, оновлюючи чи видаляючи їх.

Симуляції реалізують шаблон «Декоратор». Базова симуляція (BaseSimulation) надає основні функції запуску та відображення результатів.

Ця базова структура може бути розширена через декоратори, які додають специфічні функції, наприклад, покращення графіки чи інтеграцію складніших фізичних моделей. SimulationDecorator об'єднує додаткову функціональність, а конкретні реалізації, такі як AdvancedSimulation, додають конкретні поліпшення.

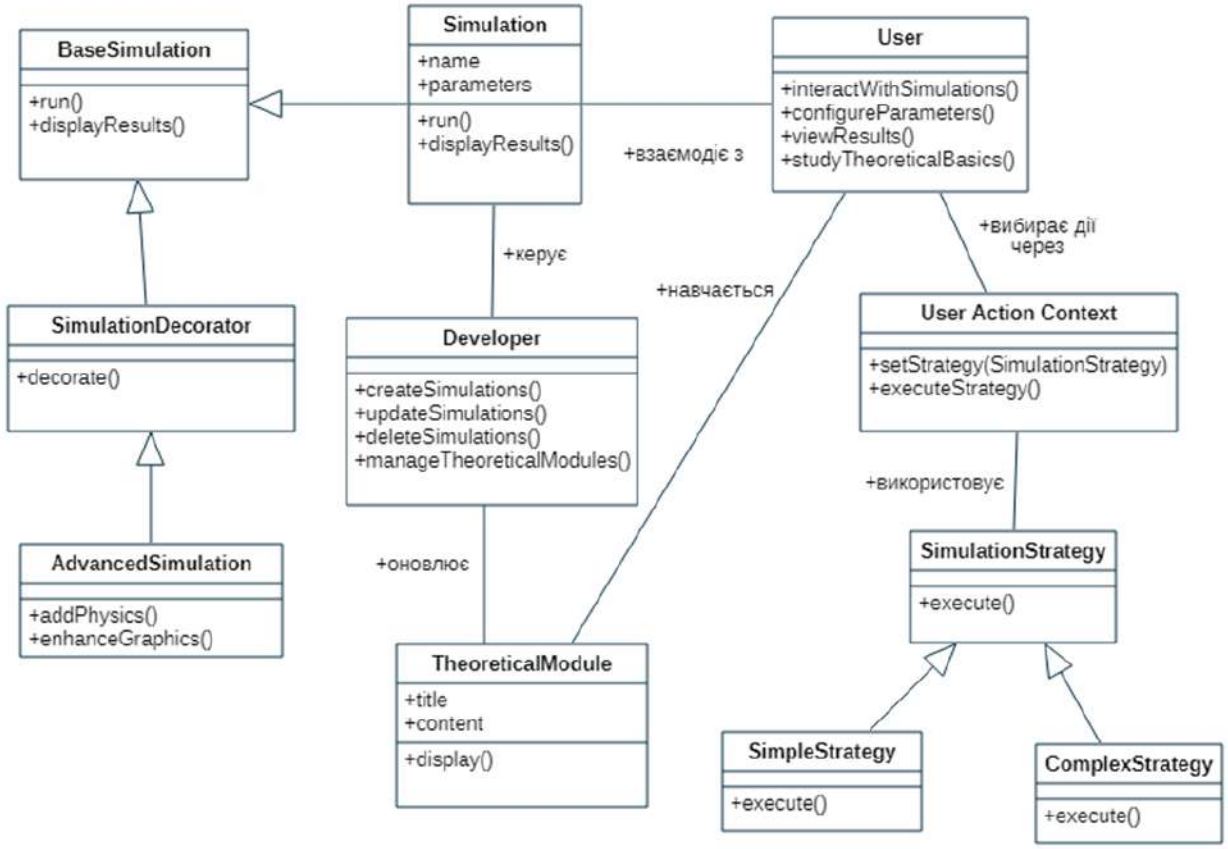


Рис. 3 Діаграма концептуальних класів

Шаблон «Стратегія» реалізовано через класи SimulationStrategy, SimpleStrategy та ComplexStrategy. Контекст користувача (User Action Context) забезпечує механізм вибору та використання стратегії, яка визначає спосіб взаємодії користувача із симуляціями. Наприклад, проста стратегія може використовуватись для базових дій (вибір параметрів), а складна стратегія – для багатокрокових або оптимізованих сценаріїв.

Таким чином, система PhysiXLab використовує ці шаблони для забезпечення гнучкості та масштабованості, дозволяючи розробляти

складні взаємодії та розширювати функціональність без необхідності зміни базової архітектури.

Експериментальне тестування програмної системи було проведено відповідно до мети скорочення часу на створення та впровадження фізичних ефектів у ігрові проєкти. Для перевірки досягнення мети роботи було проведено експеримент,який полягав у порівнянні часу, необхідного для розробки фізичних ефектів за допомогою традиційного підходу, і часу, витраченого на виконання тих самих завдань із використанням системи PhysiXLab. Учасники експерименту, поділені на дві групи, отримали завдання створити кілька фізичних ефектів, зокрема моделювання гравітації, тертя та пружних зіткнень. Учасниками експерименту були розробники з базовими знаннями фізики та досвідом роботи з інструментами для моделювання фізичних явищ. Вони мали вміти використовувати обрану мову програмування (JavaScript, Python або C#) для інтеграції фізичних ефектів у свої проєкти. Перша група виконувала завдання за допомогою звичайних інструментів, з якими вони працювали раніше (табл. 1).

Таблиця 1 – Результати експерименту для першої групи учасників

Учасник	Моделювання ефекту	Мова інтеграції	Час виконання (хв)
Учасник 1	Обертання	JavaScript	45.23
Учасник 2	Обертання	Python	44.77
Учасник 3	Рух по траєкторії	JavaScript	60.12
Учасник 4	Зіткнення	C#	54.88
Учасник 5	Гравітація	Python	50.10
Учасник 6	Тертя	C#	48.55
Учасник 7	Обертання	Python	43.90
Учасник 8	Рух по траєкторії	JavaScript	57.45

Друга група використовувала функціонал системи PhysiXLab, включаючи інструменти для автоматичного налаштування параметрів і візуалізації результатів у реальному часі (табл. 2).

Таблиця 2 – Результати експерименту для другої групи учасників

Учасник	Моделювання ефекту	Мова інтеграції	Час виконання (хв)
Учасник 1	Обертання	Python	31.05
Учасник 2	Рух по траєкторії	JavaScript	41.40
Учасник 3	Зіткнення	C#	37.95
Учасник 4	Гравітація	Python	34.50
Учасник 5	Тертя	C#	33.12
Учасник 6	Обертання	Python	30.36
Учасник 7	Рух по траєкторії	JavaScript	37.02

Результати експерименту дозволили оцінити, наскільки система PhysiXLab сприяє скороченню часу розробки фізичних ефектів, а також її вплив на зручність і якість роботи розробників.

Середнє арифметичне часу виконання завдань для першої групи (без використання системи PhysiXLab) становить 50.63 хвилин, тоді як для другої групи (з використанням системи PhysiXLab) середнє значення становить 34.49 хвилин. Таким чином, система PhysiXLab скоротила середній час виконання завдань на 31% порівняно з першою групою.

Висновки У рамках роботи виконано розробку програмної системи PhysiXLab для моделювання фізичних явищ у інтерактивному середовищі. Використання цієї системи дозволяє скоротити час розробки фізичних ефектів на 31% порівняно з традиційними підходами, що значно підвищує ефективність інтеграції фізичних симуляцій у проєкти різного типу.

Список використаних джерел

1. Вовк Л., Недавній Д. Фізика в іграх. Створення ігрового двигуна на основі фізичних процесів. Цифрова платформа: інформаційні технології в соціокультурній сфері. 2023, том 6, № 1. – С. 107–120.

2. GameDesigning – «Don’t Forget Physics in Your Game». URL: https://www.gamedesigning.org/learn/game-physics/ (дата звернення: 01.06.2026).

3. Box2D: a 2D Physics Engine for Games. URL: https://box2d.org/ (дата звернення: 01.06.2026).

4. Bullet Real-Time Physics Simulation. URL: https://pybullet.org/wordpress/ (дата звернення: 01.06.2026).

5. Matter.js. URL: https://brm.io/matter-js/ (дата звернення: 01.06.2026).

6. Ammo.js. URL: https://github.com/kripken/ammo.js (дата звернення: 01.06.2026).

References

1. Вовк Л., Недавній Д. Фізика в іграх. Створення ігрового двигуна на основі фізичних процесів. Цифрова платформа: інформаційні технології в соціокультурній сфері. 2023, том 6, № 1. – С. 107–120.

2. GameDesigning – «Don’t Forget Physics in Your Game». URL: https://www.gamedesigning.org/learn/game-physics/ (дата звернення: 01.06.2026).

3. Box2D: a 2D Physics Engine for Games. URL: https://box2d.org/ (дата звернення: 01.06.2026).

4. Bullet Real-Time Physics Simulation. URL: https://pybullet.org/wordpress/ (дата звернення: 01.06.2026).

5. Matter.js. URL: https://brm.io/matter-js/ (дата звернення: 01.06.2026).

6. Ammo.js. URL: https://github.com/kripken/ammo.js (дата звернення: 01.06.2026).