регулярно оцінювати їх ефективність, а також враховувати різноманітність потреб студентів та доступ до нових розробок.

Обговорення використання віртуальної та доповненої реальності в освіті вказує на їхній потенціал для трансформації окремих підходів у навчанні. Ці технології можуть збагатити освітній досвід, надаючи студентам інноваційні методи навчання та дозволяючи їм візуалізувати складні концепції. Ефективна інтеграція VR та AR у навчальні програми може зробити викладання більш інтерактивним та ефективним, відкриваючи нові перспективи для освітньої сфери, але вона пов'язана з проблемами, такими як забезпечення доступності обладнання та розробка якісного освітнього контенту.

На закінчення слід зазначити, що AR-бібліотеки володіють різними програмними характеристиками і функціональними можливостями, за допомогою яких удосконалюються методики викладання. Навчання з використанням VR та AR забезпечує інтерактивність, симуляцію та створення віртуальних навчальних матеріалів для більш ефективного навчання.

Список використаних джерел

1. Грод Ін. М., Безверхній Є. І. Дослідження можливостей існуючих застосунків для створення об'єктів доповненої реальності. *Підготовка майбутніх учителів фізики, хімії, біології та природничих наук в контексті вимог Нової української школи : з*бірник тез доповідей VI Міжнародної науково-практичної конференції (Тернопіль, 23-24 травня 2024 року). Тернопіль. С. 293–297.

2. Diegmann P., Schmidt-Kraepelin M. S., van den Eynden and Basten D. Benefits of Augmented Reality in Educational Environments – A Systematic Literature Review. *Proceedings of the 12th International Conference on Wirtschaftsinformatik,* 2015. P. 1542–1556.

ОСОБЛИВОСТІ ВИКОРИСТАННЯ РУШІЯ GODOT ТА C# ДЛЯ РОЗРОБКИ ІГРОВИХ ЗАСТОСУНКІВ

Мельник Петро Петрович
здобувач другого (магістерського) рівня вищої освіти зі спеціальності Комп'ютерні науки,
Тернопільський національний педагогічний університет імені Володимира Гнатюка,
melnyk_pp@fizmat.tnpu.edu.ua
Василенко Ярослав Пилипович
викладач кафедри інформатики та методики її навчання,
Тернопільський національний педагогічний університет імені Володимира Гнатюка,
yava@fizmat.tnpu.edu.ua

Розвиток ігрової індустрії значно прискорився завдяки появі потужних та водночас доступних рушіїв, що дозволяють розробникам створювати високоякісні ігрові продукти з меншими затратами часу та ресурсів. Одним із таких рушіїв є Godot, який виділяється своєю відкритістю, гнучкістю та підтримкою кількох мов програмування, зокрема C#.

Godot пропонує багатофункціональне середовище для створення як 2D, так і 3D-ігор, має простий у використанні редактор та потужну систему вузлів, що значно спрощує розробку. Використання C# як мови програмування у Godot

відкриває перед розробниками можливості інтеграції з уже існуючими проєктами на платформі.NET, що є важливим фактором для професійних розробників.

Зростаючий попит на ігровий контент, а також необхідність ефективних та економічних інструментів розробки роблять дослідження можливостей Godot та C# надзвичайно актуальним. Хоча рушій Godot поступово набуває популярності, він ще не настільки широко використовується, як інші рушії, наприклад Unity або Unreal Engine. Водночас поєднання Godot та C# дає можливість розробникам скористатися всіма перевагами сучасного програмування, зокрема об'єктно-орієнтованим підходом, продуктивністю та масштабованістю.

Дослідження цієї теми допоможе розробникам краще зрозуміти особливості використання Godot у зв'язці з C#, а також оцінити його потенціал для створення конкурентоспроможних ігрових застосунків.

Методами дослідження є *аналіз першоджерел та документації* – вивчення офіційної документації Godot, матеріалів щодо використання C# у цьому рушії, наукових статей та технічних ресурсів; *порівняльний аналіз* – зіставлення можливостей Godot+C# з іншими популярними зв'язками рушіїв та мов програмування (Unity+C# або Unreal Engine+Blueprints); вивчення та аналіз готових ігрових продуктів, створених з використанням Godot+C#, для виявлення загальних підходів, кращих практик та можливих вдосконалень; *емпіричний метод* (практичний експеримент) – розробка прототипу ігрового застосунку для перевірки можливостей рушія та оцінки продуктивності; *метод тестування* – перевірка працездатності, продуктивності, оптимізації та стабільності роботи застосунку; *метод моделювання* – створення архітектури гри, побудова ігрових механік та систем на основі C# у середовищі Godot; *метод спостереження та аналізу результатів* – оцінка ефективності використання Godot та C# шляхом аналізу результатів тестування та порівняння з очікуваними характеристиками; *метод узагальнення та висновків* – формулювання висновків щодо доцільності використання Godot+C# для розробки ігрових застосунків.

Godot – це багатоплатформний відкритий ігровий рушій, що підтримує 2D та 3D графіку. Він надає розробникам інтуїтивно зрозумілий інтерфейс та потужні інструменти для створення ігрових проєктів різної складності. Однією з ключових особливостей Godot є його система вузлів, яка дозволяє будувати складні сцени шляхом комбінування простих компонентів.

Починаючи з версії 3.0, Godot інтегрував підтримку мови C#, що дозволяє розробникам використовувати переваги цієї мови, такі як статична типізація, об'єктно-орієнтоване програмування та багата стандартна бібліотека. Для роботи з C# у Godot необхідно встановити відповідну версію рушія з підтримкою.NET та налаштувати середовище розробки, наприклад, Visual Studio або Visual Studio Code.

GDScript є власною мовою сценаріїв Godot, спеціально розробленою для швидкої та легкої інтеграції з рушієм. Дослідження [2] показує, що GDScript забезпечує кращий доступ до основних функцій рушія, тоді як C# демонструє вищу продуктивність та гнучкість у великих проєктах. Вибір між C# та GDScript залежить від потреб проєкту та досвіду розробника.

Розробка ігор у Godot з використанням C# включає кілька ключових етапів:

Налаштування проєкту. Створення нового проєкту з підтримкою C#, налаштування середовища розробки та підключення необхідних бібліотек.

Розробка ігрової логіки. Створення скриптів на C# для визначення поведінки ігрових об'єктів, обробки подій та взаємодії між компонентами.

Взаємодія з вузлами. Отримання доступу до вузлів сцени, їх властивостей та методів через скрипти на C#.

Тестування та налагодження. Використання вбудованих інструментів Godot для тестування ігрового процесу та налагодження коду.

Детальний опис цих етапів подано в [1].

Практичний приклад створення гри на Godot з використанням C# можна знайти у курсі «Learn 2D Game Development with Godot 4.3 and C# from Scratch»на платформі Udemy [3], в якому є покрокові інструкції щодо розробки гри, починаючи від налаштування проєкту до реалізації ігрової механіки та публікації готового продукту.

Аналізуючи процес розробки ігрового застосунку можна виділити такі аспекти: архітектура вузлів у Godot дозволяє ефективно організовувати код та структуру гри; основні механіки (рух, взаємодія з об'єктами, фізика) реалізуються через скрипти C#, які інтегруються безпосередньо з вузлами сцени.

Практичні рекомендації розробки ігрового застосунку на основі Godot + C#:

1. Налаштування середовища розробки. Використовуйте Godot 4 з підтримкою.NET (скачати можна з офіційного сайту). Встановіть.NET SDK та налаштуйте середовище розробки (Visual Studio, VS Code або Rider). Використовуйте C# 10 або вище, щоб отримати доступ до сучасних можливостей мови.

2. Архітектура проєкту. Використовуйте патерн вузлів (Node System): кожен об'єкт має бути окремим вузлом, що спрощує розробку та налагодження. Розділяйте логіку на окремі класи та компоненти для полегшення тестування та модифікації. Використовуйте автозавантажувані (autoload) скрипти для менеджерів станів (наприклад, для збереження даних гравця).

3. Робота з C# у Godot. Використовуйте partial-класи та інтерфейси, щоб краще структурувати код. Мінімізуйте використання `await` в ігрових циклах, оскільки це може вплинути на продуктивність. Використовуйте `Signals` для взаємодії між вузлами, замість безпосереднього виклику методів. Дотримуйтеся асинхронного підходу при роботі з ресурсами (завантаження текстур, звуків тощо).

4. Графіка та анімація. Використовуйте TileMap для побудови рівнів у 2D-іграх. Використовуйте AnimationPlayer для складних анімацій персонажів. Оптимізуйте спрайти та текстури, уникаючи непотрібних прозорих пікселів.

5. Фізика та керування персонажем. Для руху використовуйте `CharacterBody2D` або `CharacterBody3D` замість `KinematicBody`, оскільки вони мають оновлену фізику в Godot 4. Використовуйте `MoveAndSlide()` для реалізації плавного руху персонажів. Враховуйте Delta Time (`delta`) для узгодженості швидкості руху незалежно від FPS.

6. Оптимізація продуктивності. Мінімізуйте виклики `GetNode()` – краще кешувати вузли в полях класу. Використовуйте Object Pooling для повторного

використання об’єктів (наприклад, снарядів або ворогів). Обмежуйте використання `process()` для важких обчислень, виносячи їх у окремі методи або `Timers`. Використовуйте Multi-threading (`Thread`, `Task`) для обробки фонових процесів.

7. Робота з користувацьким інтерфейсом (UI). Використовуйте Control Nodes для створення гнучкого UI. Адаптуйте інтерфейс під різні розміри екранів за допомогою `Container` та `Anchor`. Використовуйте `Theme` для стандартизації UI-елементів (кнопки, панелі, шрифти).

8. Обробка введення (Input System). Використовуйте Input Map для налаштування управління замість жорстко закодованих клавіш. Враховуйте підтримку контролерів та сенсорного управління, якщо гра орієнтована на мобільні пристрої.

9. Збереження даних. Використовуйте JSON або Binary файли для збереження прогресу гравця. Для складних збережень використовуйте SQLite або Godot FileSystem API. Враховуйте шифрування даних, якщо потрібно зберігати чутливу інформацію.

10. Дебагінг та тестування. Використовуйте Godot Debugger для відстеження помилок у коді. Виводьте важливу інформацію через `GD.Print()` під час розробки. Використовуйте Unit-тести (`GUT – Godot Unit Test`) для перевірки логіки. Перевіряйте продуктивність через Profiler, щоб оптимізувати ресурси.

11. Експорт гри. Налаштуйте автоматичне збирання (CI/CD) для тестування гри на різних платформах. Використовуйте Godot Export Templates для створення кросплатформених версій. Оптимізуйте розмір пакету шляхом видалення зайвих ресурсів перед експортом.

Дотримуючись цих рекомендацій, можна значно покращити продуктивність, якість коду та зручність роботи з Godot + C# при розробці ігрових застосунків.

Практичне значення одержаних результатів у даному дослідженні полягає в тому, що отримані результати дозволяють створити робочий прототип гри з використанням Godot та C#, що може бути основою для подальшого розвитку та комерційного використання. Практична реалізація підтверджує ефективність цього підходу для інді-розробників та малих студій. Дослідження демонструє, як Godot та C# можна ефективно використовувати для швидкої розробки ігор із застосуванням об’єктно-орієнтованого підходу. Висвітлені рекомендації щодо вибору C# чи GDScript допоможуть розробникам обрати найкраще рішення для їхніх проєктів. Отримані результати можуть бути використані у навчальних курсах та посібниках з розробки ігор, зокрема у програмах університетів або онлайн-курсах. Код, розроблений під час дослідження, може слугувати шаблоном для початківців у Godot + C#.

Результати дослідження допоможуть ігровим розробникам, студентам, викладачам та ентузіастам краще зрозуміти особливості використання Godot + C# та ефективно застосовувати їх у власних проєктах.

Використання відкритого рушія Godot та мови C# сприяє розвитку незалежної розробки ігор та створенню економічно ефективних ігрових продуктів без ліцензійних обмежень. Godot у поєднанні з C# є ефективним інструментом для розробки ігрових застосунків. Рушій Godot забезпечує інтуїтивно зрозуміле середовище, потужні можливості для 2D та 3D графіки та підтримку багатьох

платформ. Інтеграція мови C# дозволяє розробникам використовувати об'єктно-орієнтований підхід, покращену продуктивність та багатий функціонал стандартної бібліотеки.NET.

У дослідженні розроблено робочий прототип гри, що підтвердило можливість ефективного використання Godot + C# для створення ігор. Дослідження надає практичні рекомендації щодо оптимізації коду, вибору між C# та GDScript, а також найкращих практик роботи з рушієм. Отримані результати можуть бути використані у навчальних матеріалах, курсах з ігрової розробки та для подальших наукових досліджень.

Рушій Godot у поєднанні з C# є перспективним вибором для розробки як інді-ігор, так і комерційних проєктів, забезпечуючи баланс між продуктивністю, простотою використання та відкритою екосистемою. Отримані висновки можуть бути основою для подальших досліджень щодо продуктивності C# у Godot, порівняння з іншими рушіями або впровадження додаткових технологій, таких як штучний інтелект чи багатокористувацькі системи.

Список використаних джерел

1. Документація Godot: Основи C#. URL: https://docs.godotengine.org/uk/4.x/tutorials/scripting/c_sharp/c_sharp_basics.html (available at: 25.03.2025).

2. Learn 2D Game Development with Godot 4.3 and C# from Scratch. URL: https://www.udemy.com/course/learn-2d-game-development-godot-43-c-from-scratch/?couponCode=UDEAFFHP12025 (available at: 25.03.2025).

3. Sharp Coder Blog. URL: https://uk.sharpcoderblog.com/blog/essential-techniques-for-game-development-in-godot#google_vignette (available at: 25.03.2025).

4. Thorn A. Scripting with C# in Godot: Common Tasks. In: Moving from Unity to Godot. Apress, 5. Berkeley. CA. 2020 (available at: 25.03.2025).

5. Vuorela J. «Differences Between C# and GDScript in Godot Game Engine.» Bachelor's thesis, Tampere University of Applied Sciences. URL: https://www.theseus.fi/bitstream/handle/10024/874264/Vuorela_Jesse.pdf?sequence=2 (available at: 25.03.2025).

ЦИФРОВІ ІДЕНТИФІКАТОРИ ЯК ІНСТРУМЕНТ АКАДЕМІЧНОЇ ВИДИМОСТІ: МЕТОДИЧНІ АСПЕКТИ

Мінтій Ірина Сергіївна
кандидат педагогічних наук, провідний науковий співробітник відділу відкритих освітньо-наукових інформаційних систем,
Інститут цифровізації освіти НАПН України,
mintii@iitlt.gov.ua

Вакалюк Тетяна Анатоліївна
доктор педагогічних наук, завідувач кафедри інженерії програмного забезпечення,
Державний університет «Житомирська політехніка»,
tetianavakaliuk@gmail.com

Цифрові ідентифікатори стали невід'ємною частиною сучасного наукового середовища, забезпечуючи дослідникам видимість у науковому просторі. Особливої актуальності це питання набуває для молодих науковців, зокрема аспірантів, які тільки починають формувати свою академічну репутацію. Правильне використання таких інструментів як Google Scholar, ORCID,