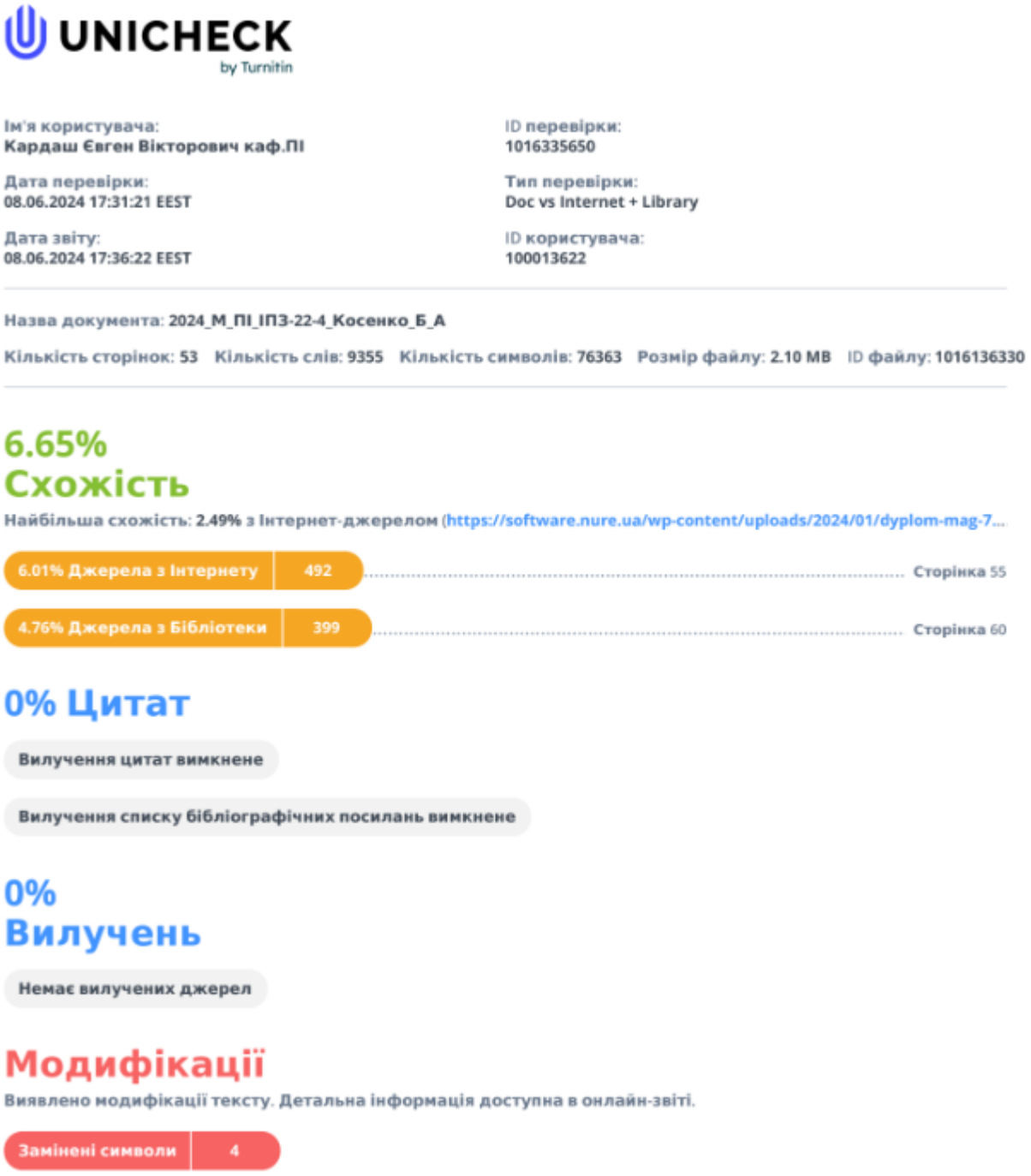
ДОДАТОК А

Результати перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Кардаш Євген Вікторович каф.ПІ

ID перевірки:
1016335650

Дата перевірки:
08.06.2024 17:31:21 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
08.06.2024 17:36:22 EEST

ID користувача:
100013622

Назва документа: 2024_М_ПІ_ІПЗ-22-4_Косенко_Б_А

Кількість сторінок: 53 Кількість слів: 9355 Кількість символів: 76363 Розмір файлу: 2.10 MB ID файлу: 1016136330

6.65%
Схожість

Найбільша схожість: 2.49% з Інтернет-джерелом (https://software.nure.ua/wp-content/uploads/2024/01/dyplom-mag-7...

6.01% Джерела з Інтернету 492 Сторінка 55

4.76% Джерела з Бібліотеки 399 Сторінка 60

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 4

ДОДАТОК Б

Скан-копії тез з XXVII міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті»

УДК 004.514

ДОСЛІДЖЕННЯ МЕТОДІВ ОПТИМІЗАЦІЇ ТА ПРОЕКТУВАЛЬНИХ РІШЕНЬ ДЛЯ СТВОРЕННЯ СКЛАДНИХ ІГРОВИХ ПРОГРАМНИХ СИСТЕМ

Косенко Б.А.
Науковий керівник – к.т.н., доц. Мар'їн С.О.
Харківський національний університет радіоелектроніки, каф. ПІ, м. Харків, Україна
e-mail: borys.kosenko@nure.ua

This research is dedicated to the exploration of optimization methods and design solutions in the development of complex gaming software systems. The research aims to investigate contemporary trends in the field of game system development and identify optimal optimization strategies to ensure high performance and gameplay quality. The Unity development environment will be utilized for this research, given its extensive built-in tools for project exploration and analysis. The robust customization capabilities of Unity also allow for the development of new tools. Additionally, the research plans to examine how various design solutions impact the development speed and fault tolerance of the gaming system.

Ігрова індустрія, нинішній гігант із оборотом у 200 мільярдів доларів, можливо, є сьогодні одним із найважливіших та інноваційних секторів технологій. Його важливість для культури, соціальних мереж та розваг неможливо недооцінити. Термін «індустрія розваг» більше не використовується тільки для Голлівуду та кіноіндустрії, оскільки ігри тепер надають одну з найбільш захоплюючих форм розваг більш ніж трьом мільярдам людей по всьому світу. Сучасна ігрова індустрія, створена постійно зростаючими ігровими компаніями, розширює межі технологій, створюючи передові та захоплюючі розваги.

Зі збільшенням кількості гравців зростає і конкуренція у сфері розробки ігор, що у свою чергу сприяє появі нових ігрових програм. В наш час практично всі можливі ігрові механіки вже були створені кимось і реалізовані в одному з тисяч проектів. Однак, просто надати якісну гру вже не достатньо для завоювання користувальницької уваги.

Сьогодні гравців необхідно дивувати, пропонуючи їм щось унікальне та захоплююче. Це призводить до того, що ігри стають все більш масштабними та амбітними у своїх ідеях та реалізації. Збільшується складність як процесу розробки, а й подальшої підтримки таких гігантських ігрових проектів.

Розробка та підтримка АА та ААА проектів є викликом для багатьох розробників, особливо в умовах зростаючої конкуренції та вимогливих смаків гравців[1]. Відсутність якісної оптимізації може призвести до невдалого випуску гри на ринок.

Отже оптимізація продуктивності ігрового проекту — один із вирішальних факторів для масштабування, оскільки різні пристрої мають різні апаратні можливості та обмеження. Продуктивність означає, наскільки плавно та ефективно працює гра, без затримок, збоїв та розрядки акумулятора. Щоб оптимізувати продуктивність гри необхідно використовувати різні методи, такі як скорочення кількості викликів малювання, оптимізація розміру та якості ресурсів, використання ефективних алгоритмів та структур даних, а також реалізація кешування та об'єднання в пули[2]. Також необхідно використовувати інструменти профілювання для вимірювання та аналізу продуктивності проекту на різних платформах та виявлення потенційних вузьких місць та проблем.

Підтримка таких проектів також є значущим аспектом. Після випуску розробники повинні надавати постійні оновлення, виправляти помилки та взагалі удосконалювати гру, щоб утримати та залучати аудиторію[3].

Швидкість впровадження нових механік та вирішення наявних помилок залежить від обраного комплексу проектних рішень. Це охоплює всі частини проекта: створення та налаштування ігрових об'єктів, їх управління, впровадження залежностей, підхід до розподілу бізнес-логіки, управління ігровим циклом та станами, складність ієрархії класів, абстракції ігрових об'єктів, отримання доступу до інших компонентів у об'єкті чи сцені, складність компонентів ігрових об'єктів, характеристики об'єктів у грі, модифікації, доступ до тимчасових та постійних даних, серіалізація даних та їх безпека, передача даних між об'єктами у сцені та між сценами, покриття тестами та інше.

При цьому вибір одних рішень може перекривати можливість використання інших. Наприклад, якщо бізнес-логіка буде в ігрових компонентах, прикріплених до об'єктів на сцені, то це викличе труднощі для покриття такої логіки автоматичними текстами. У такому разі доведеться розділити логіку та візуальну модель, що у свою чергу збільшить час, необхідний на впровадження нових механік, але зменшить ризик виникнення помилок.

Також при проектуванні потрібно розробляти рішення (правила, підходи, гайдлайни) не тільки для ігрової логікі, але і для всього, з чим проект пов'язаний: назва файлів і папок, структура зберігання ассетів, структура збірок, утиліти для розробки, тестування, робота із системою контролю версії, білд проекту та його деплой, робота зі сторами та виливка хотфіксів, моніторинг метрик, помилок та реагування на них[4].

Розробка додаткових сервісів та шарів абстракції, кастомних утиліт для редактора, покриття тестами може зайняти значний час розробки та окупити його лише у довгостроковій перспективі. Іноді цілі команди розробників займаються проектуванням заради проектування там, де це зовсім не потрібно забуваючи, що в першу чергу вони розробляють гру.

Саме тому важливо знаходити баланс між дійсно необхідними проектними рішеннями та тими, що на даний момент не несуть практичної користі.

Отже, грамотний підхід до проектування продукту може сприяти швидкому впровадженню нових механік, уникненню основної проблеми швидкої розробки - появі великої кількості багів. У такому випадку розробники матимуть можливість витратити більше часу на вдосконалення ігрових механік проекту.

Оптимізація продуктивності має безпосередній вплив на фінансовий успіх проекту, оскільки вона визначає кількість пристроїв, на яких гра зможе оптимально функціонувати. Профайлінг проекту, хоча і є тривалим і складним процесом, є необхідною складовою розробки, яку необхідно враховувати впродовж всього проекту, а не залишати на останній момент.

Якісно розроблений оптимізаційний план із систематичним усуненням вузьких місць у проекті буде значно ефективнішим, ніж хаотичні виправлення. Це сприятиме вдосконаленню продукту та забезпечить його стабільну роботу, що відіграє ключову роль у забезпеченні задоволення користувачів та, в кінцевому рахунку, фінансовому успіху проекту.

Список використаних джерел

1. Jason Gregory. Game Engine Architecture. 3rd edition. CRC Press, 2018. 1240 p.
2. Andrew Rollings, Dave Morris. Game Architecture and Design. 2rd edition. New Riders, 2004. 926 p.
3. Julian Gold. Object-oriented Game Development. Pearson Education, 2004. 426p.
4. Daniel Sánchez, Crespo Dalmau. Core Techniques and Algorithms in Game Programming. New Riders, 2004. 854 p.

ДОДАТОК В

Слайди презентації

Дослідження методів оптимізації та проектувальних рішень для створення складних ігрових програмних систем

Косенко Б.А. , ІПЗм-22-4
Науковий керівник: доцент, Мар'їн С.О.
18 червня 2024

Дослідження

Актуальність: На даний момент ігрова індустрія стрімко розвивається, зростають вимоги до складності та якості ігор. Оптимізація ігрових програмних систем стає критично важливою для забезпечення високої продуктивності, особливо при роботі з великими обсягами даних та складною графікою.

Напрям дослідження: Оцінка ефективності застосування сучасних методів оптимізації рендерингу великої кількості об'єктів на сцені.

Об'єкт дослідження: Ігрові програмні системи, що використовують великий обсяг об'єктів і даних, з акцентом на оптимізацію їхньої продуктивності та ефективності за допомогою передових технологій, таких як ECS, Jobs, Burst, та GPU Instancing.

2

Огляд літератури (аналогів)

Одним із найпотужніших методів оптимізації застосовуваних у моєму дослідженні є стек технологій Unity, орієнтованих на дані (DOTS).

Так як, в цілому, це досить сучасна технологія, найактуальнішу інформацію найкраще отримувати з офіціальної документації, наприклад https://unity.com/ru/dots або Unity Technologies (2021).

Незважаючи на це, з загальними принципами архітектурного шаблону ECS можна ознайомитись у цій книзі «Data-oriented design: software engineering for limited resources and short schedules by Mr Richard Fabian (2018)»

Застосовуваний у роботі технологічний стек все ще перебуває у розробці, тому багато інформації з відкритих джерел вже застаріли і неактуіальні.

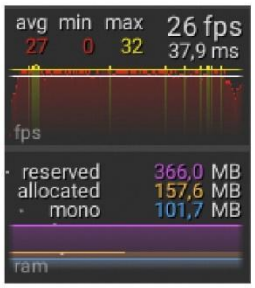
Постановка задачі

Проблема полягає в недостатній продуктивності ігрових програмних систем при роботі з великою кількістю об'єктів у реальному часі. Це зумовлено обмеженнями традиційних об'єктно-орієнтованих підходів до програмування, що призводить до надмірного навантаження системи.

Необхідно дослідити та впровадити сучасні методи оптимізації, які дозволять ефективно управляти великими обсягами даних та покращити загальну продуктивність ігрових систем.

Від результатів дослідження очікується значне підвищення частоти кадрів, суттєве зниження часу обробки одного кадру та оптимізація споживання пам’яті.

Методологія



Було проведено емпіричне дослідження, а саме практична реалізація та тестування різних методів оптимізації на прикладі ігрової сцени, що складається з великої кількості об'єктів.

Після тестування був проведений порівняльний аналіз у якому ми зіставили результати продуктивності для різних підходів та технологій з метою визначення їхньої ефективності.

В роботі використовувалися наступні технології: Unity Data-Oriented Technology Stack, GPU Instancing, Jobs System, Burst Compiler.

Інструменти для виміру продуктивности: Unity Profiler та Stats Monitor & Debugger.

Архітектура системи для проведення експериментального дослідження

Main Application відповідає за основний ігровий цикл.

ECS (Entity Component System) організовує ігрову логіку шляхом розділення даних (Entities та Components) та поведінки (Systems).

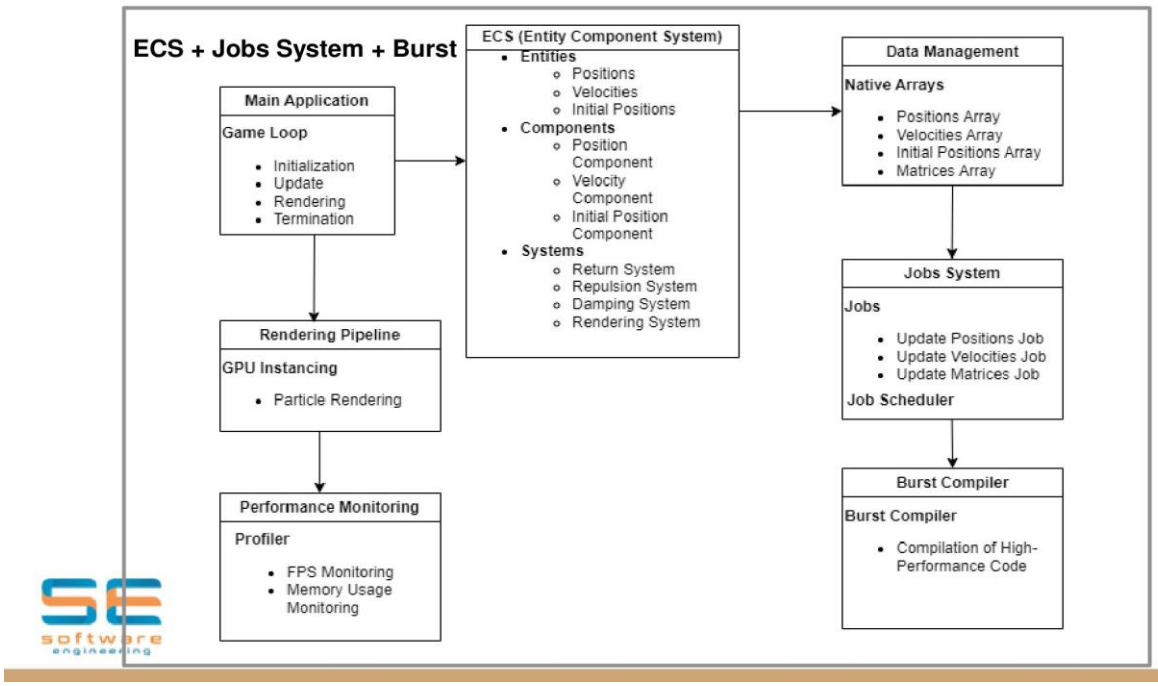
Jobs System відповідає за паралельне виконання завдань.

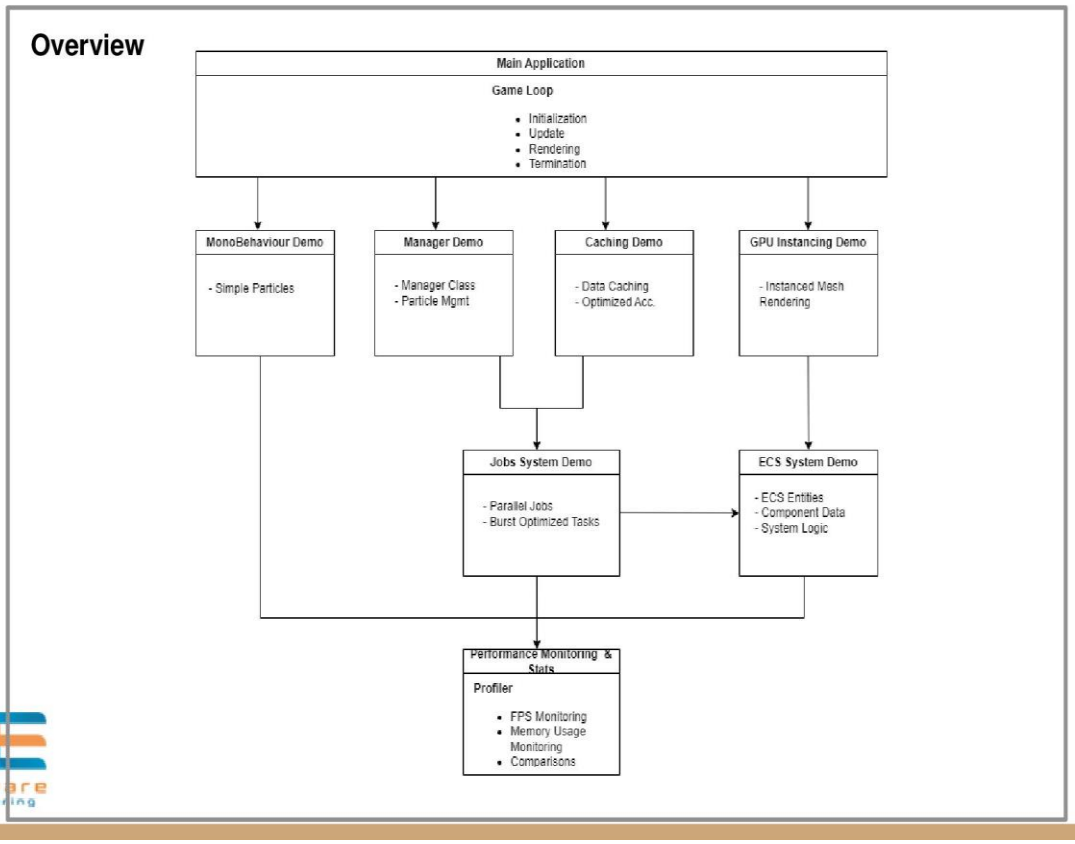
Burst Compiler забезпечує високопродуктивну компіляцію коду.

Rendering Pipeline використовує GPU Instancing для ефективного рендерингу великої кількості об'єктів на графічному процесорі.

Data Management організовує зберігання і управління даними за допомогою Native Arrays.

Performance Monitoring включає використання профайлера для моніторингу продуктивності системи в реальному часі.


ECS + Jobs System + Burst
Main Application
Game Loop
Initialization
Update
Rendering
Termination
ECS (Entity Component System)
Entities
Positions
Velocities
Initial Positions
Components
Position Component
Velocity Component
Initial Position Component
Systems
Return System
Repulsion System
Damping System
Rendering System
Data Management
Native Arrays
Positions Array
Velocities Array
Initial Positions Array
Matrices Array
Jobs System
Jobs
Update Positions Job
Update Velocities Job
Update Matrices Job
Job Scheduler
Burst Compiler
Burst Compiler
Compilation of High-Performance Code
Rendering Pipeline
GPU Instancing
Particle Rendering
Performance Monitoring
Profiler
FPS Monitoring
Memory Usage Monitoring
7


Overview
Main Application
Game Loop
Initialization
Update
Rendering
Termination
MonoBehaviour Demo
- Simple Particles
Manager Demo
- Manager Class
- Particle Mgmt
Caching Demo
- Data Caching
- Optimized Acc.
GPU Instancing Demo
- Instanced Mesh Rendering
Jobs System Demo
- Parallel Jobs
- Burst Optimized Tasks
ECS System Demo
- ECS Entities
- Component Data
- System Logic
Performance Monitoring & Stats
Profiler
FPS Monitoring
Memory Usage Monitoring
Comparisons
8

Опис програмного забезпечення, що було використано у дослідженні

Після визначення вимог до продуктивності системи було спроектовано систему, що включає:

- C# + Unity з фреймворком DOTS, що включає ECS, Jobs System та Burst Compiler;
- для виміру продуктивности Unity Profiler та Stats Monitor & Debugger.

Були розроблені основні компонентів системи (Entities, Components, Systems). Проведена інтеграція системи завдань для паралельної обробки. Використано Burst Compiler для підвищення продуктивності та реалізована система рендерингу за допомогою GPU Instancing.

У якості середовища розробки був використаний Rider.

Зміст проведеного експерименту

Методи: індивідуальні MonoBehaviours, спільний менеджер, кешування, GPU Instancing, Jobs System, ECS .

Вхідні дані: від 20000 до 290000 об’єктів, початкові позиції, швидкість та стан об’єктів, положення курсору для симуляції взаємодії.

Критерії: FPS, Frame Time, Memory Usages.

Послідовність: ініціалізація - запуск експерименту - збір даних - аналіз результатів.

Вимірювання: Profiler та Custom Performance Monitoring.

Результати експерименту

Результати тестування за різних сценаріїв. Метрикою продуктивності виступає середня частота кадрів

Сценарій тестування	Кількість об'єктів - 20000	Кількість об'єктів - 40000	Кількість об'єктів - 100000
Індивідуальні MonoBehaviours	75	15	6
Спільний менеджер	90	26	17
Кешування	95	30	20
GPU Instansing	300	55	37
GPU Instansing + Jobs + Burst	600	110	70
ECS + Jobs + Burst	650	270	160

Результати експерименту

Результати тестування двох реалізацій Mono та IL2CPP. Метрикою продуктивності виступає середня частота кадрів

Кількість об'єктів	Середній FPS, Mono	Середній FPS, IL2CPP
30000	115	115
60000	113	114
100 000	72	107
200 000	32	48
290 000	23	34

Аналіз отриманих результатів

Проведений експеримент повністю відповідає зазначеним цілям, оскільки дозволив отримати детальні результати для кожного методу оптимізації, який ми використовували. Зібрані дані свідчать про значні відмінності в продуктивності між цими методами.

Яскравий приклад: впровадження GPU Instancing надає **215.79%** приріст продуктивності у порівнянні з загальним менеджером та кешованими даними при 20000 об'єктів.

Загалом впровадження ECS + Jobs + Burst надає **2566.67%** приріст продуктивності у порівнянні з індивідуальними MonoBehaviours при 100000 об'єктів.

Також ми провели тестування двох реалізацій Mono та IL2CPP. На підставі отриманих даних можливо зробити висновок, що, в рамках цього з експерименту, IL2CPP виграє у Mono приблизно на **30%**.

Отримані результати підтверджують ефективність сучасних методів оптимізації, таких як ECS та Jobs System, для обробки великих обсягів даних у ігрових системах. Це сприяє подальшому розвитку та впровадженню цих підходів у ігровій індустрії, зокрема у великих проектах з високими вимогами до продуктивності.

Overview	Total	Self	Calls	GC Alloc	Time ms	Self ms
▼ PlayerLoop	89.9%	0.2%	2	400 B	39.36	0.12
▼ SimulationSystemGroup	83.4%	0.0%	1	96 B	36.50	0.00
▼ UpdateFunction.Invoke()	83.4%	0.0%	1	96 B	36.50	0.00
▼ Default World Unity.Entities.SimulationSystemGroup	83.4%	0.0%	1	96 B	36.49	0.02
▶ Default World ECS.Systems.SpriteSheetRenderer	82.8%	81.7%	1	96 B	36.24	35.75

Публікація результатів

УДК 004.514

ДОСЛІДЖЕННЯ МЕТОДІВ ОПТИМІЗАЦІЇ ТА ПРОЕКТУВАЛЬНИХ РІШЕНЬ ДЛЯ СТВОРЕННЯ СКЛАДНИХ ІГРОВИХ ПРОГРАМНИХ СИСТЕМ

Косенко Б.А.

Науковий керівник – к.т.н., доц. Мар'їн С.О.

Харківський національний університет радіоелектроніки, каф. ПІ, м. Харків, Україна

e-mail: borys.kosenko@nure.ua

This research is dedicated to the exploration of optimization methods and design solutions in the development of complex gaming software systems. The research aims to investigate contemporary trends in the field of game system development and identify optimal optimization strategies to ensure high performance and gameplay quality. The Unity development environment will be utilized for this research, given its extensive built-in tools for project exploration and analysis. The robust customization capabilities of Unity also allow for the development of new tools. Additionally, the research plans to examine how various design solutions impact the development speed and fault tolerance of the gaming system.

Ігрова індустрія, величезний гігант із оборотом у 200 мільярдів доларів, можливо, є сьогодні одним із найвражливіших та інноваційних секторів технологій. Його важливість для культури, соціальних мереж та розваг неможливо недооцінити. Термін «індустрія розваг» більше не використовується тільки для Голлівуду та кіноіндустрії, оскільки ігри тепер надають одну з найбільш захоплюючих форм розваг більш ніж трьом мільярдам людей по всьому світу. Сучасна ігрова індустрія, створена постійно зростаючими ігровими компаніями, розширює межі технологій, створюючи передові та захоплюючі розваги.

Зі збільшенням кількості гравців зростає і конкуренція у сфері розробки ігор, що у свою чергу спричиняє появі нових ігрових програм. В наш час практично всі можливі ігрові механіки вже були створені кимось і реалізовані в одному з тисяч проектів. Однак, просто видати якісну гру вже не достатньо для завоювання користувальницької уваги.

Сьогодні гравцям необхідно дивувати, пропонуючи їм щось унікальне та захоплююче. Це призводить до того, що ігри стають все більш масштабними та амбітними у своїх ідеях та реалізації. Збільшується складність як процесу розробки, а й подальшої підтримки таких гігантських ігрових проектів.

Розробка та підтримка AA та AAA проектів є викликом для багатьох розробників, особливо в умовах зростаючої конкуренції та вимогливих смаків гравців[1]. Відсутність якісної оптимізації може призвести до невдалого випуску гри на ринок.

Отже оптимізація продуктивності ігрового проекту — один із вирішальних факторів для масштабування, оскільки різні пристрої мають різні апаратні можливості та обмеження. Продуктивність означає, наскільки плавно та ефективно працює гра, без затримок, збоїв та розрядки акумулятора. Щоб оптимізувати продуктивність гри необхідно використовувати різні методи, такі як скорочення кількості викликів малювання, оптимізація розміру та якості ресурсів, використання ефективних алгоритмів та структур даних, а також реалізація кешування та об'єднання в пули[2]. Також необхідно використовувати інструменти профілювання для вимірювання та аналізу продуктивності проекту на різних платформах та виявлення потенційних вузьких місць та проблем.

Підтримка таких проектів також є значущим аспектом. Після випуску розробники новинки надавати постійні оновлення, виправляти помилки та взагалі удосконалювати гру, щоб утримати та залучати аудиторію[3].

Швидкість впровадження нових механік та вирішення наявних помилок залежить від обраного комплексу проектних рішень. Це охоплює всі частини проекта: створення та налаштування ігрових об'єктів, їх управління, впровадження залежностей, підхід до розподілу бізнес-логіки, управління ігровим циклом та станами, складність ієрархії класів, абстракції ігрових об'єктів, отримання доступу до інших компонентів у об'єкті чи спеці, складність компоненти ігрових об'єктів, характеристики об'єктів у грі, модифікації, доступ до тимчасових та постійних даних, серіалізація даних та їх безпека, передача даних між об'єктами у спеці та між спеціами, покриття тестами та інше.

При цьому вибір одних рішень може перекривати можливість використання інших. Наприклад, якщо бізнес-логіка буде в ігрових компонентах, прикріплених до об'єктів на спеці, то це викличе труднощі для покриття такої логіки автоматичними тестами. У такому разі доведеться розділити логіку та візуальну модель, що у свою чергу збільшить час, необхідний на впровадження нових механік, але зменшить ризик виникнення помилок.

Також при проектуванні потрібно розробляти рішення (правила, підходи, гайдлайни) не тільки для ігрової логіки, але з для всього, з чим проект пов'язаний: назва файлів і папок, структура зберігання ассетів, структура збірок, утиліти для розробки, тестування, робота із системою контролю версій, білд проекту та його деплой, робота зі сторами та виливка хотфіксів, моніторинг метрик, помилок та реагування на них[4].

Розробка додаткових сервісів та шарів абстракції, кастомних утиліт для редакторів, покриття тестами може займати значний час розробки та окупити його лише у довгостроковій перспективі. Іноді цілі команди розробників займаються проектуванням заради проектування там, де не зовсім не потрібно забуваючи, що в першу чергу вони розробляють гру.

Саме тому важливо знаходити баланс між дійсно необхідними проектними рішеннями та тими, що на даний момент не несуть практичної користі.

Отже, грамотний підхід до проектування продукту може сприяти швидкому впровадженню нових механік, уникненню основної проблеми швидкої розробки - появі великої кількості багів. У такому випадку розробники матимуть можливість витратити більше часу на вдосконалення ігрових механік проекту.

Оптимізація продуктивності має безпосередній вплив на фінансовий успіх проекту, оскільки вона визначає кількість пристроїв, на яких гра зможе оптимально функціонувати. Профайлінг проекту, хоча і є тривалим і складним процесом, є необхідною складовою розробки, яку необхідно враховувати впродовж всього проекту, а не залишати на останній момент.

Якісно розроблений оптимізаційний план із систематичним усуненням вузьких місць у проекті буде значно ефективнішим, ніж хаотичні виправлення. Це скоротить вдосконалення продукту та забезпечить його стабільну роботу, що відіграє ключову роль у забезпеченні задоволення користувача та, в кінцевому рахунку, фінансовому успіху проекту.

Список використаних джерел

1. Jason Gregory. Game Engine Architecture. 3rd edition. CRC Press, 2018. 1240 p.
2. Andrew Rollings, Dave Morris. Game Architecture and Design. 2nd edition. New Riders, 2004. 926 p.
3. Julian Gold. Object-oriented Game Development. Pearson Education, 2004. 426p.
4. Daniel Sánchez, Crespo Dalmau. Core Techniques and Algorithms in Game Programming. New Riders, 2004. 854 p.

XXVII МІЖНАРОДНИЙ МОЛОДІЖНИЙ ФОРУМ «РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ»

Підсумки

Наше дослідження підтверджує доцільність використання Unity DOTS у масштабних проектах з жанрової спрямованостю RTS, MMO, симулятори та економічні стратегії.

Загалом можна оптимізувати як ігрову логіку (керування великою кількістю юнітів/будівель, обробка поведінки великих груп NPC), так і окремі ігрові компоненти (рендерингу великої кількості дерев та рослин без значного зниження продуктивності, оптимізація обчислення фізичних взаємодій між об'єктами, реалізація складних руйнувань будівель).

Варто враховувати що при застосуванні даних технологій у реальному проекті ефект від їх впровадження буде значно нижчим, тому що ці проекти часто мають набагато складнішу архітектуру та більше залежностей між компонентами, ніж демонстраційні приклади.

Також результат від впровадження цих технологій значною мірою залежить від складності обчислень та кількості об'єктів на сцені.

Можливий розвиток досліджень: аналіз впливу на різних апаратних платформах та дослідження комбінованих підходів.

ДОДАТОК Г

Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимоги ДСТУ 3008:2015

1

Експертний висновок результатів перевірки кваліфікаційної роботи

студент (посада) | програмної інженерії (кафедра) | ІПЗм-22-4 (група)

Косенко Борис Андрійович
(прізвище, ім'я, по батькові)

Зауваження

Пункт ДСТУ 3008-2015	Зміст пункту	Сторінка кваліфікаційної роботи
1	2	3
	7.1 Загальні положення	
	7.3 Нумерація сторінок звіту	
	7.4 Нумерація розділів, підрозділів, пунктів, підпунктів	
	7.5 Рисунки	
	7.6 Таблиці	
	7.7 Переліки	
	7.8 Примітки	
	7.9 Виноски	
	7.10 Формули та рівняння	
	7.11 Посилання	
	7.13 Список авторів	
	7.14 Скорочення та умовні познаки	
	7.15 Додатки	
Методичні вказівки до виконання кваліфікаційної роботи магістра… ЗАТВЕРДЖЕНО кафедрою ІП протокол №10 11.01.2022	Відсутній перелік посилань на наукові праці викладачів кафедри	

Експерт _____ (підпис) Олена ОЛІЙНИК (прізвище, ініціали)

09.06.2024

ДОДАТОК Д

Виміри продуктивності під час тестування

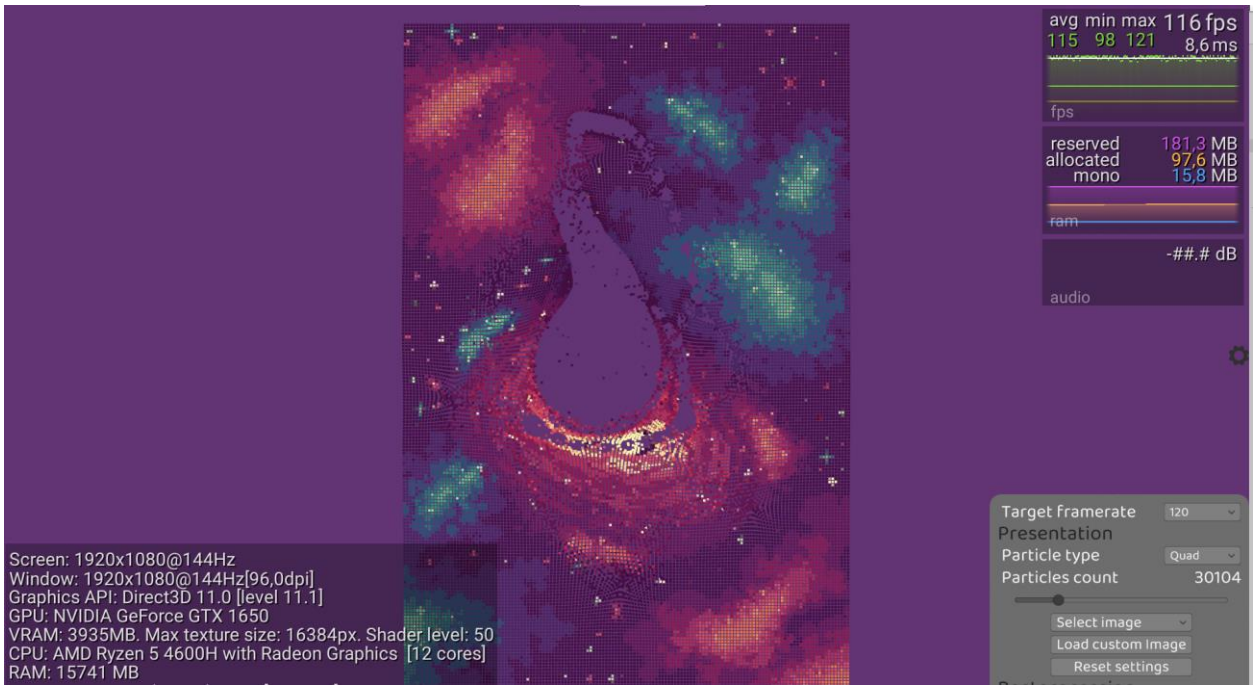


Рисунок Д.1 – Тестування продуктивності, сбірка Mono 30 000 об'єктів (виконано самостійно)

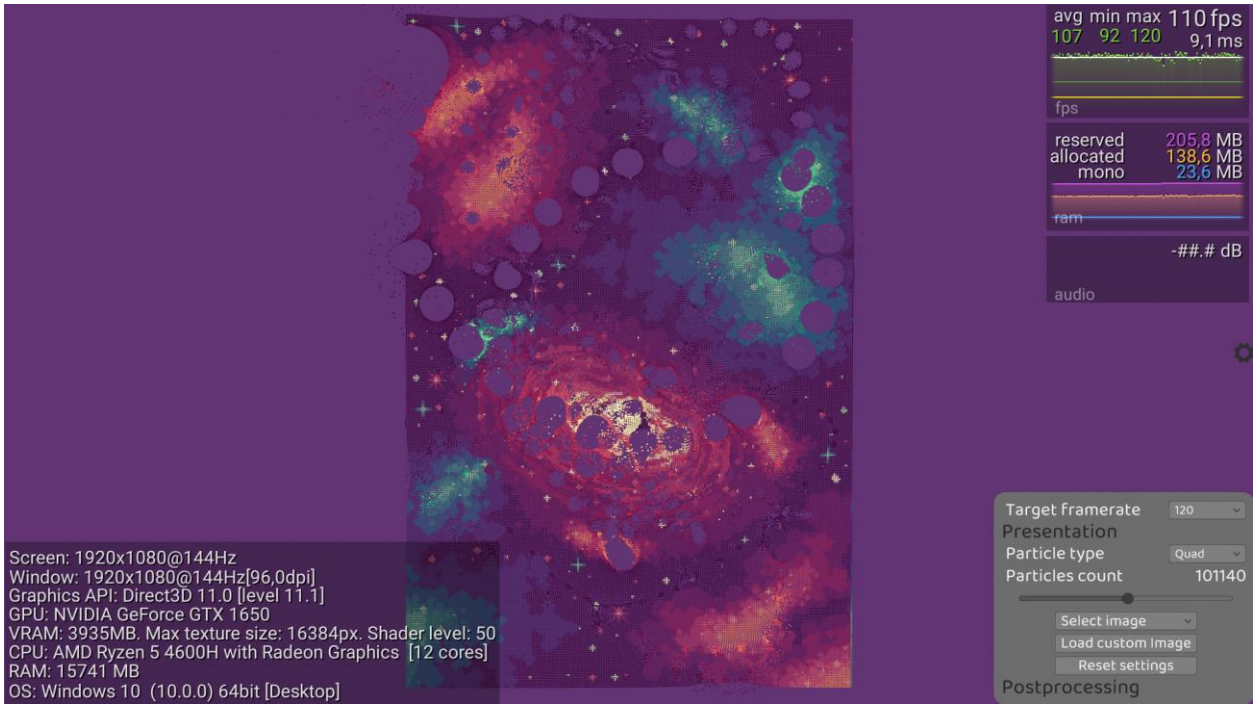


Рисунок Д.2 – Тестування продуктивності, сбірка IL2CPP 100 000 об'єктів (виконано самостійно)

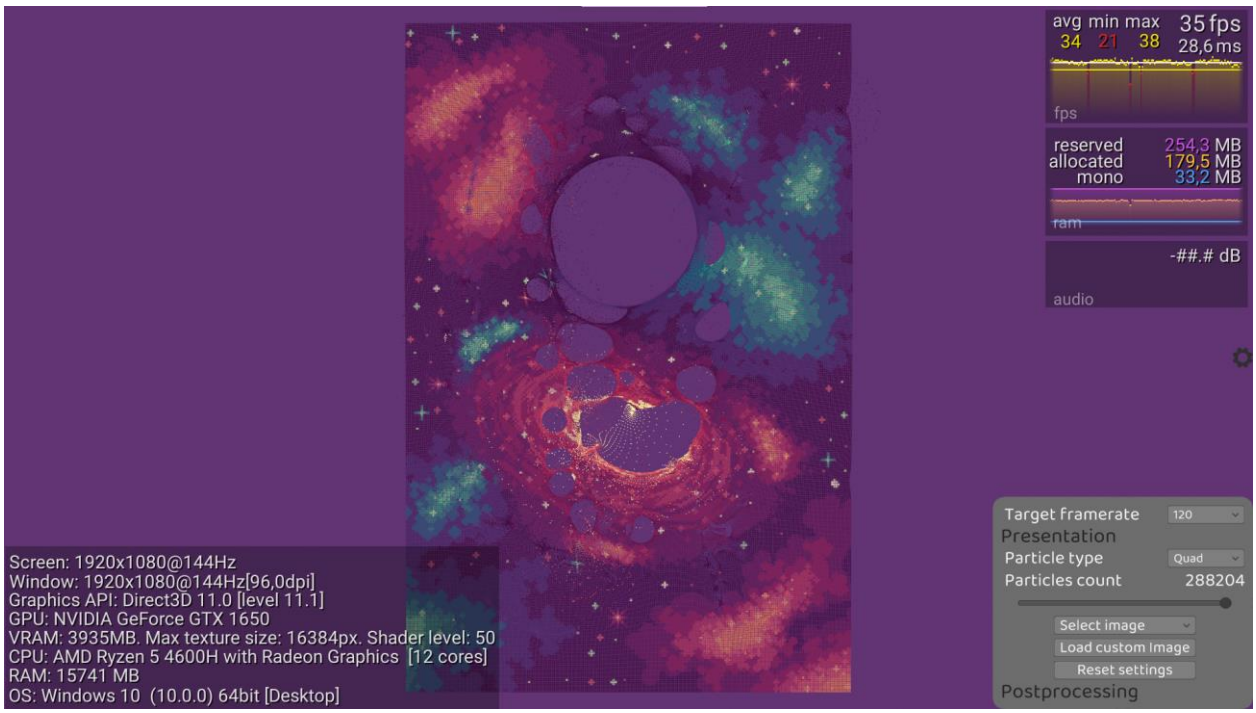


Рисунок Д.3 – Тестування продуктивності, сбірка IL2CPP 300 000 об'єктів (виконано самостійно)

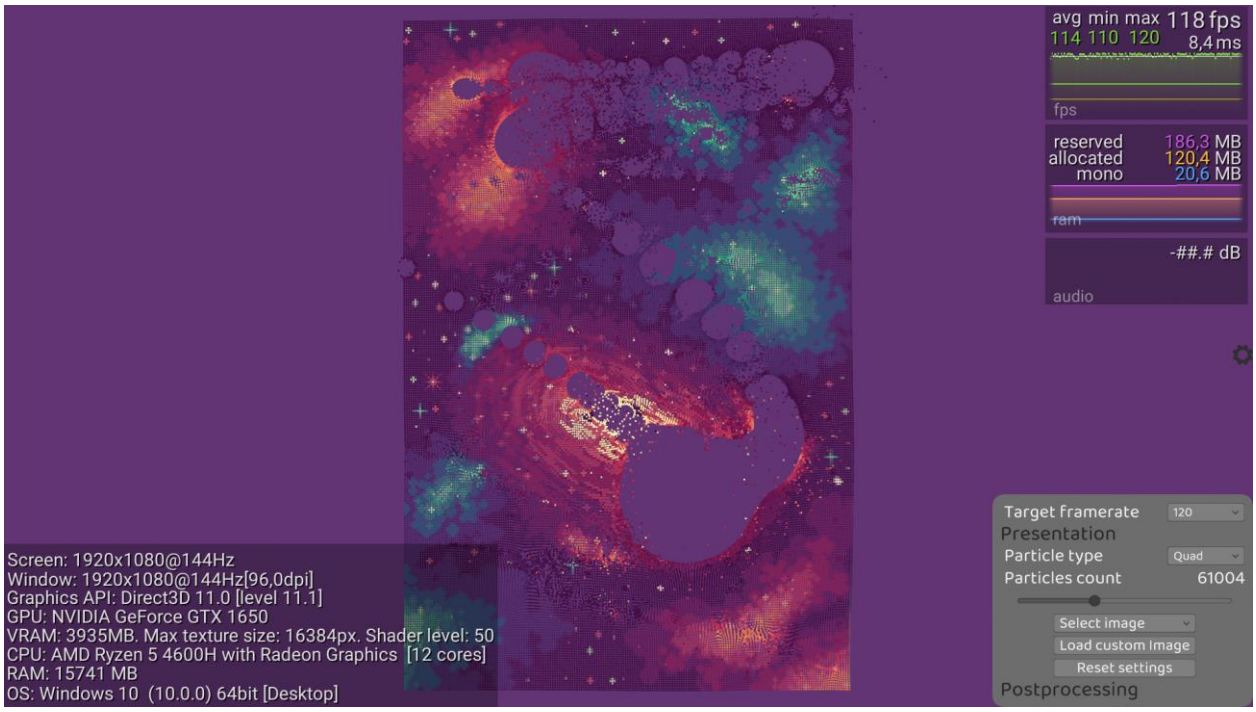


Рисунок Д.4 – Тестування продуктивності, сбірка Mono 60 000 об'єктів (виконано самостійно)