Додаток А ГРАФІЧНА ЧАСТИНА

Харківський національний університет радіоелектроніки
Кафедра ЕОМ

Кваліфікаційна робота
Другий (магістерський) рівень

Метод оптимізації ігрового додатку на платформі Unity

Автор:
Воробйов А.А.,
студ. гр. СПм-22-1

Керівник:
Фесенко Т.Г.,
проф. каф. ЕОМ

1

Мета і задачі роботи

Мета: розробка алгоритмів оптимізації шляхом застосування методу оптимізації коду.

Задачі:

- проаналізувати існуючі методи оптимізації ресурсів, графіки та коду;
- дослідити вплив оптимізації на продуктивність ігрового додатку;
- розробити метод оптимізації ігрового додатку;
- протестувати застосування удосконаленого методу оптимізації на прикладі ігрового додатку на платформі Unity.

2

Оптимізація ігрових додатків

Оптимізація ігор - це процес поліпшення продуктивності та продуктивності гри, шляхом зменшення навантаження на апаратне забезпечення комп'ютера.

а)

Приклад оптимізації ефектів ігрового додатку: а) Ефекти до оптимізації; б) Ефекти після оптимізації.

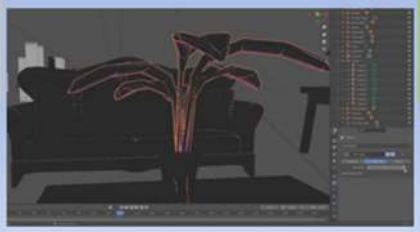
б)

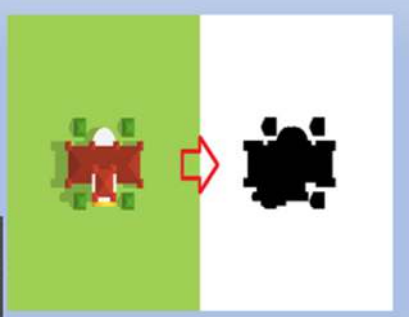
3

ОПТИМІЗАЦІЯ

МЕТОДИ ВИРІШЕННЯ ПРОБЛЕМ ПРОДУКТИВНОСТІ

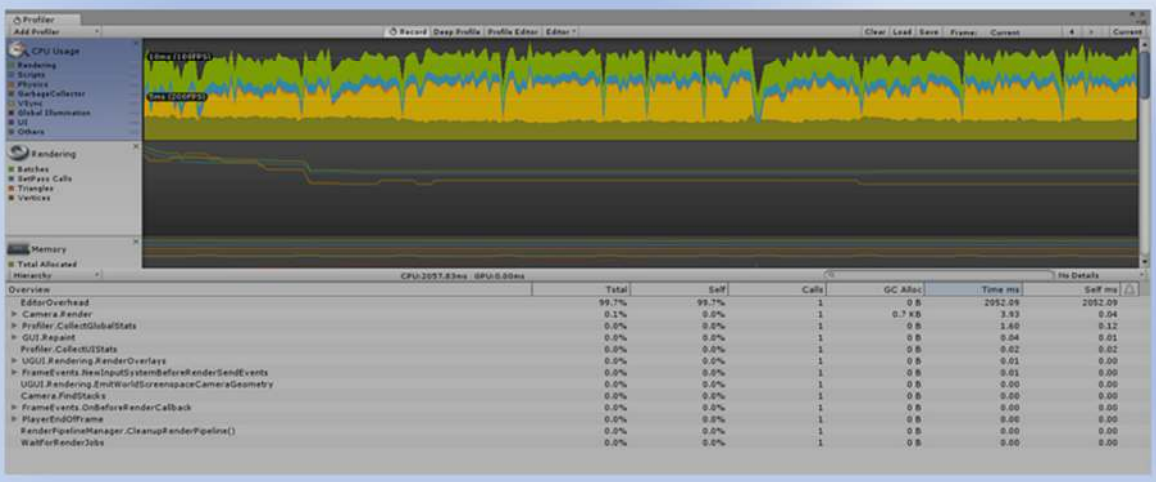
МЕТОДИ ОПТИМІЗАЦІЇ ГРАФІЧНИХ ЗОБРАЖЕНЬ ТА АНІМАЦІЇ





4

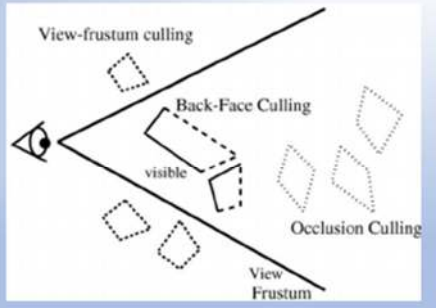
Unity Profiler – як невід'ємна частина оптимізації

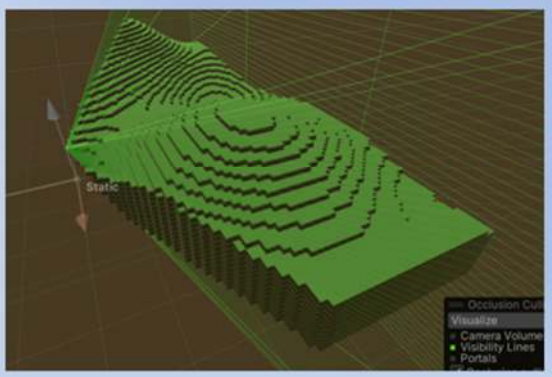


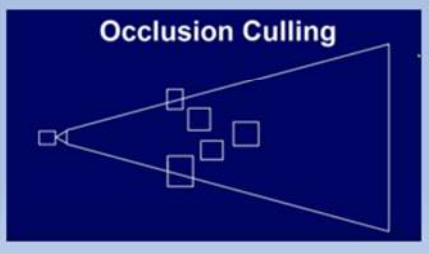
5

Вирішення проблематики рендерингу об’єкту

Occlusion Culling - це метод оптимізації графіки, який дозволяє виключати з рендерингу об'єкти, які не видно з точки зору камери.



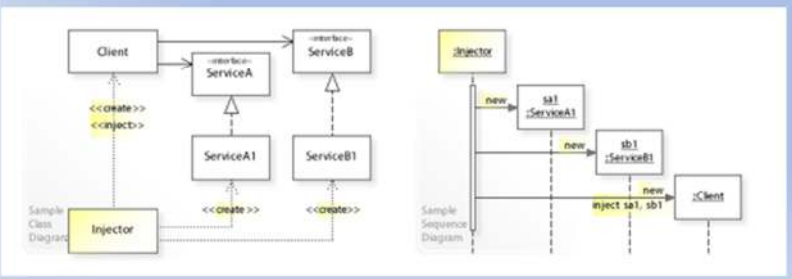




6

Оптимізація продуктивності, шляхом зменшення кількості циклів залежності.

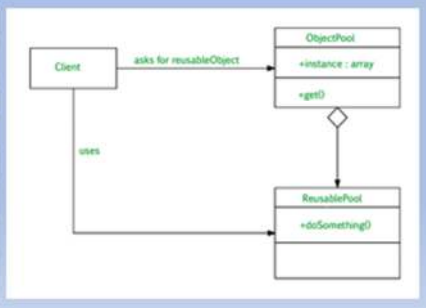
Впровадження залежностей (DI) - це метод проектування, який дозволяє об'єктам отримувати доступ до інших об'єктів, не знаючи про їхню конкретну реалізацію.

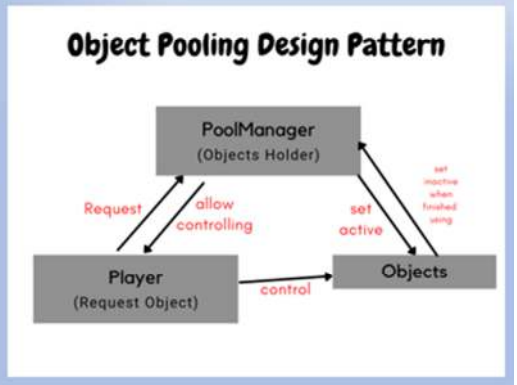


7

Оптимізація продуктивності, шляхом повторного використання вже створених об’єктів.

Метод Object Pooling допомагає оптимізувати продуктивність ігрового додатку на платформі Unity шляхом повторного використання вже створених об'єктів.

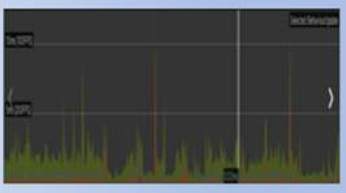




8

Кінцевий результат роботи методів

Object Polling



Результат роботи методу

(Time: 0.13 ms)

Результат роботи без роботи методу (Time: 0.95 ms)

DI

Параметр	Без DI	З DI
FPS	60	65
Середнє навантаження на CPU	100%	95%
Середнє навантаження на GPU	90%	85%

Occlusion Culling

Характеристика	Без Occlusion Culling	З Occlusion Culling
Кількість об'єктів графіки, які рендеряться	Всі об'єкти, які знаходяться в межах видимості камери	Об'єкти, які не перекриваються один з одним, або які видно з точки зору камери
Приклад	Для сцени з 100 об'єктами графіки навантаження на графічний процесор близько 100 кадрів в секунду	Для сцени з 100 об'єктами графіки навантаження на графічний процесор близько 50 кадрів в секунду
Навантаження на графічний процесор	Може бути значним	Знижується
FPS	30	60

9

Експериментальна частина

Сцена меню вибору персонажу

Головна сцена гри, з головним персонажем

10

Тестування

Характеристика	Рекомендовані вимоги	Мінімальні вимоги
Операційна система	Windows 10	Windows 7
Процесор	Intel Core i5-4590 або AMD Ryzen 5 1500X	Intel Core i3-3240 або AMD Athlon II X3 4150
Оперативна пам'ять	8 ГБ	4 ГБ
Відеокарта	NVIDIA GeForce GTX 1050 або AMD Radeon RX 570	NVIDIA GeForce GTX 650 або AMD Radeon HD 7770
Звук	Звукова карта, сумісна з DirectX 9.0c	Звукова карта, сумісна з DirectX 9.0c

Результат тестуванні гри на мінімальних вимогах

11

Висновки

- розглянуто різні аспекти оптимізації ігрових додатків, включаючи графіку, фізику, анімацію та логіку гри.
- розроблено та впроваджено комплекс практичних методів оптимізації гри. Методи оптимізації були спрямовані на зменшення впливу на ресурсоємність гри з метою підтримки гри на різних платформах.
- Результати тестування застосування удосконаленого методу оптимізації на прикладі ігрового додатку на платформі Unity підтвердили значне покращення продуктивності та якості геймплею гри.

12

Апробація результатів кваліфікаційної роботи

Воробйов А.А. Метод оптимізації ігрового додатку на платформі Unity. *Modernization of science and its influence on global processes: collection of scientific papers «SCIENTIA» with Proceedings of the IV International Scientific and Theoretical Conference, November 3, 2023*. Bern, Swiss Confederation: International Center of Scientific Research. 2023, P. 116–120.

13

Додаток Б

SECTION 18.
INFORMATION TECHNOLOGIES AND SYSTEMS

Воробйов Антон Андрійович
здобувач вищої освіти технічного факультету
Харківський національний університет радіоелектроніки, Україна
Науковий керівник: Фесенко Тетяна Григорівна
Професор кафедри електронних обчислювальних машин, доктор технічних наук
Харківський національний університет радіоелектроніки, Україна

МЕТОД ОПТИМІЗАЦІЇ ІГРОВОГО ДОДАТКУ НА ПЛАТФОРМІ UNITY

Анотація. *Предметом дослідження у статті є методи оптимізації ігор на движку Unity. Мета роботи є проведення дослідження та розробка комплексного підходу до оптимізації ігор, створених з використанням популярного движка Unity.*
Головною метою є досягнення покращення продуктивності та підвищення якості геймплею шляхом оптимізації різних аспектів гри, включаючи графіку, фізику, анімацію та логіку гри.
Результатом роботи буде розробка та впровадження комплексу практичних методів оптимізації, які дозволять забезпечити оптимальне використання ресурсів та забезпечити гладкий та стабільний геймплей. Важливим аспектом роботи є також зменшення впливу на ресурсоємність гри для підтримки гри на різних платформах, включаючи мобільні пристрої, персональні комп'ютери та інші платформи.

Вступ

Галузь комп'ютерних ігор, що включає розробку, видання та просування ігор, є однією з найшвидше розвиваючихся галузей комп'ютерних технологій і глобального сектору розваг на сьогоднішній день. Світовий ринок комп'ютерних ігор розширився і став більш глибоким і об'ємним. Протягом останніх п'яти років він зростав у середньому на 11% щорічно, і в 2021 році досягнув 192,7 мільярда доларів, випередивши інші конкуруючі галузі розважального контенту, такі як кіно та музика. Цей зріст був підсилений загальною доступністю інтернету та розповсюдженням комп'ютеризованих пристроїв, зокрема смартфонів, а також розширенням каналів постачання. У 2020 році обсяг світового ринку комп'ютерних ігор очікувалося збільшити на 18% до 160 мільярдів доларів, але пандемія COVID-19 призвела до змін, і ринок ігор зрос до 179,18 мільярда доларів. Загальний прибуток за цей період перевищив прогнози на 43 мільярди доларів до приходу пандемії. За прогнозами, ринок ігор продовжить зростати до 2025 року, середньорічно на 3,4%, майже до 211,2 мільярда доларів.

У 2022 році мобільні ігри залишаються найбільшим сегментом глобального ринку комп'ютерних ігор, складаючи 50% (або 92,2 мільярда доларів) загального обсягу, виходячи з оцінок. Існує прогноз, що цей сегмент продовжить збільшувати свою долю, оскільки кількість мобільних пристроїв перевищує кількість ПК та ігрових консолей. Ігри для консолей становлять другий за вагою сегмент з 28% ринку, що дорівнює 51,8 мільярда

доларів. Десктопні ігри загалом складають 22% ринку (40,5 мільярда доларів), включаючи браузерні онлайн-ігри (1%) та скачувані або коробкові версії (21%).Проте в 2022 році ринок ігор відзначився спадом на 4,3% в порівнянні з попереднім роком, зібравши 184,4 мільярда доларів, оскільки цей рік був періодом корекції після двох попередніх років зростання через карантин.

Сучасний світ відеоігор запрошує гравців у захопливий світ імпресивної графіки, реалістичної анімації та захоплюючої геймплею. Однак така висока якість гри часто вимагає від розробників ретельного підходу до оптимізації геймдвайсу, особливо коли йдеться про ігри, створені на платформі Unity.

Unity став одним із найпопулярніших інструментів для розробки відеоігор, і його застосування настільки розширилося, що ігри, створені на цьому движку, доступні на різних платформах – від мобільних пристроїв до особистих комп'ютерів. В цьому контексті оптимізація стає життєво важливою для розробників, які мають на меті забезпечити плавний геймплей та відмінну якість гри на різних пристроях.

Виклад основного матеріалу

1. Особливості оптимізації

Оптимізація - процес нескінченний, немає межі досконалості і немає ідеальних систем. Тому в хороших конторах оптимізацією у сенсі починають займатися ще початковому етапі виробництва, коли лише замислюють гру.

Саме тоді починають міркувати про архітектуру системи, цільові платформи, про кількість ассетів та їх перевикористання, про кількість механік та ступеня їх складності, про взаємодію механік між собою, про використовувані інструменти та технології при розробці та про якісні вимоги до гри у всіх її компонентах.

Ці рішення найсильніше вплинуть на терміни розробки та кінцевий результат, оскільки саме вони визначають обсяг робіт, вимоги до кваліфікації співробітників і наскільки сильно можна дозволити відхилитися від початково-заданого шляху.

Оптимізацію неможливо описати в одній статті. Тому зосередимося виключно на методі аналізу, що підкаже правильні шляхи оптимізації вашої конкретної гри.

Основна робота з оптимізації розпочинається під час розробки гри. І ось тут є три підходи до розробки:

- Паралельний підхід: Оптимізація починається на ранніх етапах розробки, і програмісти та художники враховують майбутні вимоги до продуктивності. Цей підхід може бути ефективним, оскільки дозволяє уникнути необхідності переробляти код або арт-об'єкти пізніше. Однак, він також може бути дорогим і трудомістким, оскільки вимагає від розробників розглядати технічні обмеження на ранніх етапах розробки.
- Послідовний підхід: Оптимізація починається після створення основної частини гри. Цей підхід може бути більш ефективним, оскільки дозволяє розробникам спочатку зосередитися на створенні захоплюючого ігрового досвіду. Однак, він також може призвести до зниження продуктивності, оскільки деякі частини гри можуть бути неоптимізовані.
- Пост-релізний підхід: Оптимізація проводиться після випуску гри. Цей підхід може бути найбільш ефективним, оскільки дозволяє розробникам отримати відгуки від гравців і визначити, які області гри потребують оптимізації. Однак, він також може призвести до затримок у випуску гри, оскільки розробникам необхідно буде переконатися, що оптимізація не призведе до збоїв або інших проблем.

Кожен з цих підходів має свої переваги та недоліки. Вибір підходу залежить від конкретних обставин розробки гри.

2. Поширені помилки при оптимізації

Почнемо з того, чого не варто робити, щоб не допустити поширених помилок. Звісно, з будь-якого правила є винятки, але новачкам в оптимізації краще уникати деяких речей:

- Дедлайн: Підвищити продуктивність в кілька днів чи навіть тижнів перед релізом неможливо, оскільки іноді потрібно змінювати системи повністю. Гра не обов'язково має досягати 60 FPS на всіх етапах розробки, але капітальні перегляди архітектури та величезну кількість роботи не варто відкладати на останній тиждень.
- Некоректні дані при включенні GPU Profiler: Увімкнення профайлера GPU може призвести до некоректних результатів на деяких платформах, тому рекомендується його вимкнути. Використання VSync призведе до великого використання системних ресурсів, перевищуючи 90%, і операції, такі як GPUProfiler.EndQueries, можуть виводити некоректні дані та спричиняти значне навантаження. Профайлер GPU корисний для глибшого аналізу ситуації, але варто включати його лише в разі, коли ви ретельно розумієте, як він працює і для яких конкретних завдань його використовувати.

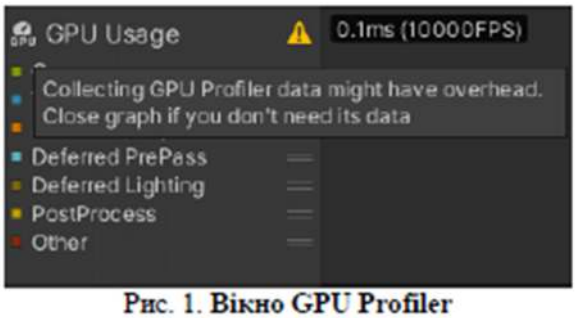


Рис. 1. Вікно GPU Profiler

- Некоректні дані під час запуску Deep Profiling: Не використовуйте опцію Deep Profile, поки не впевнетеся, коли це необхідно. Deep Profile корисно включати лише у випадку, коли є конкретна проблема з певною частиною програми, оскільки це допоможе визначити, яка частина коду викликає проблему. Проте, не слід спиратися на таймінги, отримані під час глибокого профілювання, оскільки ця опція може значно ускладнити аналіз для невеликих методів та спотворити результати. Як альтернативу або доповнення до глибокого профілювання, розміщення власних маркерів профілю може бути більш корисним підходом.

3. Методи оптимізації

3.1. Оптимізація коду

У світлі програмного розроблення, розробники часто стикаються з дилемою між продуктивністю та стабільністю. Наприклад, численні перевірки коректності вхідних даних, де найпоширенішою є перевірка на відсутність значень null, а також обробка винятків можуть використовувати значну частину обчислювальних ресурсів, і, на перший погляд, в більшості випадків, ці перевірки здаються зайвими. Однак відсутність таких перевірок може ускладнити виявлення та виправлення потенційних помилок.Також, написання коду, який був би загальним та зрозумілим для інших розробників, не завжди виходить з першої спроби. Крім того, відділений час на підтримку такого коду відводиться на витрату на реалізацію нових функцій. Наприклад, розгортання архітектури MVC може займати значний час, і в майбутньому воно може призвести до вигоди, але може і не виявитися корисним в разі заміни окремих компонентів програми.

Згідно з архітектурою MVC, потрібно створювати таку кількість класів і інтерфейсів для кожної окремої програмної сутності. За моїми оцінками, такий підхід може вповільнити розробку на ранніх етапах до 50%.

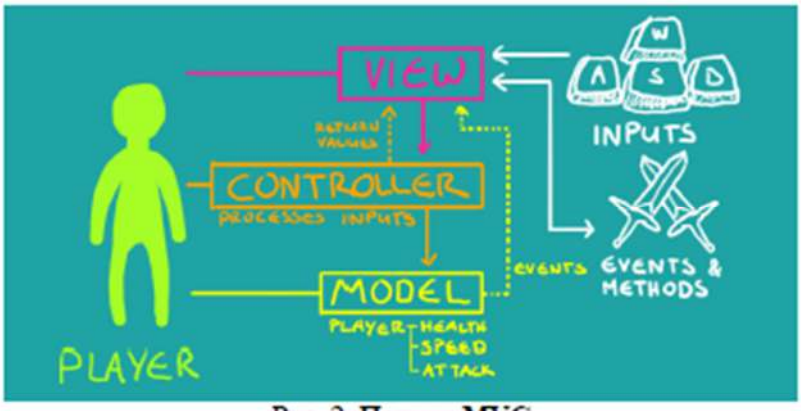


Рис. 2. Патерн MVC

3.2. Фокус на GPU

Графічна продуктивність прямо пов'язана з декількома факторами, такими як частота оновлення (фреймрейт), складність пікселів та геометрична складність (кількість вершин). Зменшення цих параметрів може покращити продуктивність. У цьому контексті, техніка Occlusion Culling може бути корисною, оскільки Unity приховує об'єкти, які не попадають в поле зору, що дозволяє ефективніше управляти ресурсами.

Особливо на мобільних пристроях, продуктивність часто залежить від швидкості заповнення (швидкість заповнення = кількість пікселів на екрані * складність шейдера * перерендеринг). Шейдери можуть бути головною причиною проблем, тому рекомендується використовувати мобільні шейдери, що постачаються з Unity або розробляти максимально прості шейдери. Якщо можливо, замініть піксельні шейдери на менш обчислювально витратні.

Ось контрольний список для хорошої практики оптимізації GPU:

- Кількість матеріалів має бути якомога нижчою. Це робить батчінг для Unity більш легким;
- Використовуйте атласи текстур (великі зображення, що містять менші) замість великої кількості окремих текстур. Це зробить завантаження швидшим;
- Використовуйте Renderer.sharedMaterial замість Renderer.material, якщо використовуєте атласи текстур та загальні матеріали;
- Рендер піксельного освітлення доріг.

3.3. Оптимізація шейдеру

Перевірка обмеження філрейту (fillrate) є досить простою: якщо зниження роздільної здатності призводить до покращення швидкості гри, то це може свідчити про обмеження філрейту.

Для зменшення складності шейдерів і поліпшення продуктивності рекомендується використовувати такі методи:

- Уникайте шейдерів із альфа-тестом, використовуйте альфа-змішані версії шейдерів.
- Альфа-тест може бути вимогливим для швидкості, тому використання альфа-змішання може допомогти покращити продуктивність.
- Використовуйте простий і оптимізований код шейдерів, такий, як у "Mobile" шейдерах, які постачаються з Unity.Шейдери з оптимізованим кодом дозволяють підтримувати графічну продуктивність на мобільних пристроях.
- Уникайте використання дорогих математичних функцій у коді шейдерів, таких як pow, exp, log, cos, sin, tan і інші.
- Замість цього варто спробувати використовувати заздалегідь обчислені текстури, що може поліпшити продуктивність.Для досягнення кращої продуктивності робіть кількість точних розрахунків (з типами float, half, fixed в Cg) якомога меншою.
- Зменшення використання точних розрахунків може позитивно позначитися на продуктивності шейдера.

3.4. Фокус на GPU

Часто виникає ситуація, коли обробка пікселів у грі обмежується процесором, і це залишає невикористаними потужності, особливо на багатоядерних процесорах. В таких випадках доцільно розглядати можливість перенесення певних функцій з графічного процесора (GPU) на центральний процесор (CPU), що підтримується Unity. Серед цих функцій можуть бути mesh skinning (скіннінг моделей), батчінг малих об'єктів та оновлення геометрії частинок.

Проте, потрібно бути особливо обережними при перенесенні функцій з GPU на CPU. Якщо ви не маєте обмежень за кількістю графічних викликів (draw calls), то використання батчингу може насправді знизити продуктивність. Він може зробити операцію відсічення об'єктів (culling) менш ефективною і зробити багато об'єктів залежними від освітлення, що може вплинути на швидкість гри та відображення.

Отже, при вирішенні, які функції переносити з GPU на CPU, слід ретельно враховувати контекст і обмеження конкретної гри для досягнення оптимальної продуктивності.

3.5. Оптимізація Фізики

Фізика може значно навантажити процесор, і це можна легко відстежити за допомогою профайлера редактора. Якщо фізика сильно навантажує процесор, ось кілька рекомендацій для оптимізації:

- Налаштуйте Time.fixedDeltaTime: В Project settings -> Time налаштуйте параметр Time.fixedDeltaTime якомога вище. Якщо ваша гра має повільний рух, то менше фіксованих оновлень може бути достатньо. На високому темпі гри може знадобитися більше частіших розрахунків, щоб уникнути проблем з колізіями.
- Physics.solverIterationCount: Параметр Physics.solverIterationCount в Physics settings також може бути використаний для підгонки продуктивності фізики.Обережно використовуйте об'єкти типу Cloth: Якщо можливо, обмежуйте використання об'єктів типу Cloth, так як вони можуть створювати значне навантаження на процесор.
- Використовуйте Rigidbodies обережно: Використовуйте Rigidbodies лише там, де вони дійсно необхідні. Надмірне використання Rigidbodies може призвести до зайвого обчислювального навантаження.
- Використовуйте примітивні колайдери: Для колізій рекомендується використовувати прості колайдери замість складних форм, якщо це можливо.
- Не рухайте статичні колайдери: Спробуйте уникати переміщення статичних колайдерів (колайдерів без Rigidbody), оскільки це може значно вплинути на продуктивність. Ви можете додати Rigidbody та встановити isKinematic в true, якщо необхідно зміщувати статичні колайдери.
- Використовуйте інструменти для профілювання: В Windows можна використовувати набір інструментів NVidia's AgPerfMon для більш докладного профілювання та отримання деталей щодо продуктивності.

Висновки

Оптимізація гри - це складний та невід'ємний процес у розробці ігор. Цей процес вимагає уважності та ретельного підходу, оскільки неконтрольовані зміни можуть призвести до негативних наслідків для продуктивності та якості гри.

Оптимізація - це завжди пошук балансу між двома протилежностями, кожна з яких працює лише у конкретних випадках. Десь підійде одне рішення, десь протилежне.

Список використаних джерел:

1. Hui, Ng & Chieng, Liu & Ting, Wen & Mohamed, Hasimah & Mohd Arshad, Muhammad. (2013), «Cross-platform mobile applications for android and iOS», 6th Joint IFIP Wireless and Mobile Networking Conference (WMNC), Pp. 1- doi: https://doi.org/10.1109/WMNC.2013.6548969.
2. П. Дейтел, Х. Дейтел, Э. Дейтел, М. Моргано (2013), «Android для программистов: создаём приложения», 560 с.
3. Sedykh I.A. (2020), "Computer games industry", 74 p
4. Buckland Mat. Programming Game AI by Example – Texas, Wordware Publishing. – 2004. – s. 25 – 43
5. Unity [Електронний ресурс] / режим доступу: https://unity.com/