УДК 004.946:004.272

А.А. ЯРОВИЙ, Л.В. КРИЛИК, О.О. КУЛИК, Д.Г. ПАСІЧНИК, О.В. НЕБОЖАНОВ

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПОБУДОВИ 3D СЦЕН ДЛЯ ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ З ПІДВИЩЕНОЮ ШВИДКОДІЄЮ

Вінницький національний технічний університет
95, Хмельницьке шосе, Вінниця, 21021, Україна
Тел.: +380 (432) 439002, e-mail: a.yarovyy@vntu.edu.ua

Анотація. В проведених дослідженнях здійснено аналіз особливостей організації інформаційної технології побудови 3D-сцен. Реалізовано програмний засіб призначений для побудови 3D-сцен на основі як CPU, так і GPU-орієнтованої програмно-апаратної платформи. За рахунок застосування ядер GPU для оброблення елементів 3D-сцен досягнуто підвищення швидкодії процесу їх оброблення.
Ключові слова: гетерогенні обчислення, віртуальна реальність, GPGPU, 3D-сцени.
Аннотация. В проведенных исследованиях осуществлен анализ особенностей организации информационной технологии построения 3D-сцен. Реализован программный комплекс, предназначенный для построения 3D-сцен на основе как CPU, так и GPU-ориентированной программно-аппаратной платформы. За счет использования ядер GPU для обработки элементов 3D-сцен достигнуто повышение быстродействия процесса их обработки.
Ключевые слова: гетерогенные вычисления, виртуальная реальность, GPGPU, 3D-сцены.
Abstract. The analysis of organization features of information technology for the 3D-scene construction is carried out in researches. The software package for both CPU and GPU-oriented software and hardware platform was developed. By using GPU cores for particles processing of 3D-scene the performance increase of processing operation was achieved.
Key words: heterogeneous computing, virtual reality, GPGPU, 3D-scenes.
DOI: 10.31649/1681-7893-2018-36-2-28-34

ВСТУП

Останнім часом значного розвитку набули технології побудови віртуальної реальності (ВР), тобто створення технічними засобами такого світу (об'єктів та суб'єктів), який передається людині через її органи відчуттів та сприймається нею як справжній. Для створення переконливого комплексу відчуттів ВР імітує як впливи, так і реакції на них в реальному часі [1].

Наразі ВР активно використовується в різноманітних сферах життя: в індустрії розваг (відеоігри, кіно і серіали), при продажі нерухомості, при проектуванні складних процесів, в галузях освіти, медицини, військової промисловості тощо. Яскравим прикладом є застосування ВР для навчання фахівців таких професій, де набуття практичних навичок може бути пов'язане з ризиком для здоров'я або життя (пілоти, лікарі, сапери тощо). З активним поширенням технологій ВР відповідно активно збільшуються й інвестиції в їх розвиток [2].

Апаратно ВР реалізується за допомогою набору пристроїв, які більш повно імітують взаємодію з віртуальної середовищем (шоломи для голови (HMD), аудіопристрої, пристрої керування тощо). Ці пристрої для своєї роботи потребують спеціалізованого програмного комплексу, який створюється здебільшого на основі спеціальних програмних засобів розробки – рушіїв (Unreal Engine, Crytek, Unity тощо).

Проте чим складнішою є 3D-сцена, яку потрібно реалізувати (наприклад, за рахунок використання складних шейдерів, "запіканні" світла, одночасної обробки великої кількості активних анімацій в реальному часі), тим більшими стають вимоги, які рушій потребує в контексті обчислювальних потужностей системи. Це в свою чергу зумовлює актуальність застосування технологій, спрямованих на збільшення обчислювальної продуктивності системи, у тому числі за рахунок оптимізації взаємодії між CPU та GPU-технологіями в межах парадигми гетерогенних обчислень [3, 4].

МЕТА І ЗАДАЧІ ДОСЛІДЖЕННЯ

Метою дослідження є підвищення швидкодії оброблення 3D-сцен для віртуальної реальності у гетерогенних системах.

Відповідно до мети підлягають вирішенню такі задачі:

1. Аналіз особливостей процесу побудови 3D-сцен.
2. Аналіз програмно-апаратних засобів побудови 3D-сцен.

3. Розробка та реалізація інформаційної технології побудови 3D-сцен для віртуальної реальності на основі CPU та GPU-орієнтованої програмно-апаратної платформи.

ПРОЦЕС ПОБУДОВИ 3D-СЦЕН НА ОСНОВІ РУШІЯ UNREAL ENGINE

Першим та одним з найбільш важливих етапів процесу побудови 3D-сцени є побудова ландшафту, яка передбачає генерацію та редагування ландшафту під поставлену задачу. Рушій Unreal Engine має вбудовані інструменти для редагування місцевості. Інструмент “Ландшафт” дозволяє створити масивні світи, засновані на рельєфі. Це є корисним, коли необхідно створити невеликий ландшафт з мінімальною кількістю нерівностей. Але, за потреби у створенні природного ландшафту великого розміру зі значною кількістю нерівностей, доцільним може виявитись застосування зовнішніх інструментів, таких як World-Machine, результати роботи якого можуть бути завантажені до Unreal Engine.

World-Machine є надзвичайно потужним інструментом, який дозволяє створити будь який необхідний ландшафт природи (рис. 1). Він має багато різних алгоритмів генерації шуму на вибір; підтримує чотири різні види природних ефектів: термічне вивітрювання, берегова ерозія, регулярна ерозія і снігопад. Кожен з них має великий діапазон налаштувань (наприклад, зміна того, як далеко переміщується осад під час ерозії), що є значною перевагою в порівнянні з більшістю інших зовнішніх інструментів, які часто надають більш обмежений інструментарій [5]. В наявності також є власні інструменти створення доріг, річок тощо.

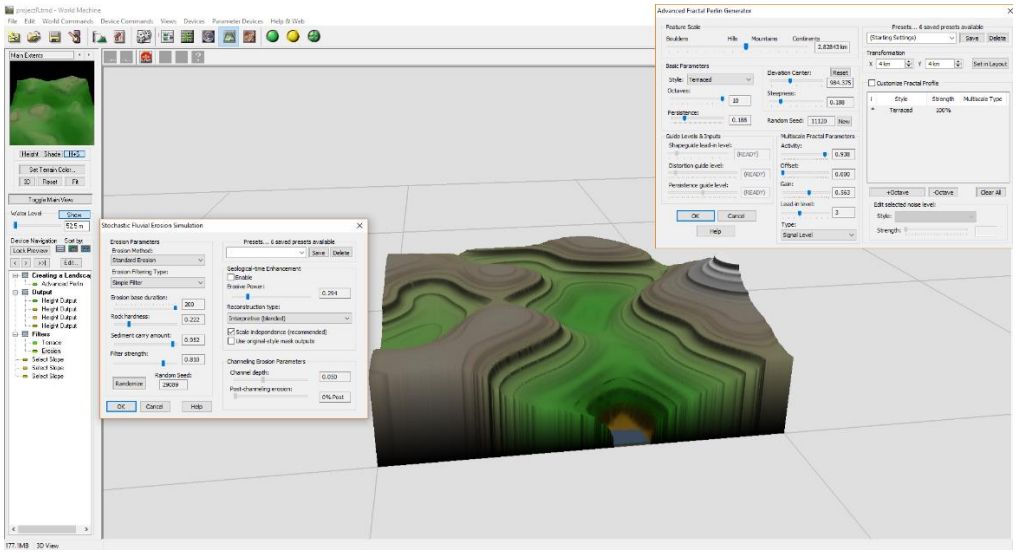

Рис. 1 – Створення ландшафту в World-Machine

Наступним етапом є знаходження та редагування під необхідну задачу усіх потрібних текстур. Кожній висоті в Unreal Engine буде присвоєно свій матеріал (трава, гравій, сніг). Коли світло зі сцени потрапляє на поверхню, матеріал використовується для розрахунку того, як цей світ взаємодіє з цією поверхнею. Для того, щоб ландшафт складався не з однієї, а з декількох текстур, необхідно створити багатошаровий матеріал, оскільки на один цілий об’єкт може бути використано лише один матеріал. Такий матеріал складається з декількох текстур, які можуть бути накладені одна на одну.

Основу матеріалу складає текстура або колір, які напряму або через відповідні функції підключені до каналу Base Colour. Інші канали є лише доповненням до основи. Головним з додаткових каналів є Normal Map. Текстури фактично є основою всього візуального ряду в 3D-середовищі. Від якості текстур залежить наскільки якісним та реалістичним буде отриманий результат. Текстури можна створити шляхом отримання необхідних фотографій та їх подальшому редагування в графічному редакторі, наприклад Photoshop. При побудові 3D-середовища використовувались текстури великої роздільної здатності, оскільки на даний момент це є найбільш ефективним для досягнення реалістичного зображення [6].

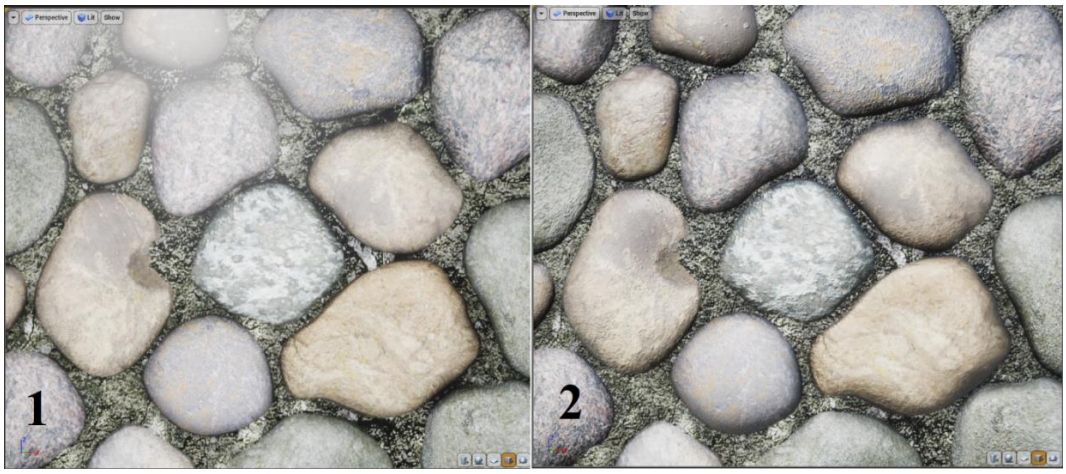

Рис. 2 – Приклад роботи детального текстурування

Детальне текстурування (Normal Map) використовується для створення ілюзії більш детальної інформації в текстурі шляхом введення повторюваної комбінації дифузійних та нормальних карт, які перевищують початкові значення Diffuse і Normal для об'єкта (рис. 2). Одним з матеріалів для створеного ландшафту у наведеному прикладі (рис. 2) є вологий камінь. При побудові його, окрім текстури каміння та детальної текстури необхідно було реалізувати функцію, яка створює з кольорової текстури, чорно білу, потім стає світлішою і в кінці накладає чорно-білу текстуру на кольорову.

Третім етапом є побудова необхідних матеріалів (шейдерів) на основі розроблених текстур, карт нормалей, та ефектів в редакторі матеріалів. Матеріали(шейдери) будуються в редакторі рушія Unreal Engine. Вони містять різні текстури та матеріальні вирази, які створюють мережу вузлів. Кінцевим результатом є матеріал, який можливо застосувати на ландшафті та об’єктах.

Потім виконується знаходження та розробка усіх необхідних Meshes для 3D середовища, а також їх подальше розташування в 3D середовищі. StaticMesh, чи предмет, являє собою елемент геометрії, який складається з множини багатокутників, які можуть бути кешованими у відеопам'яті. Це значно підвищує їх ефективність, що дозволяє їм бути набагато складнішими, ніж інші типи геометрії. Оскільки вони кешуються у відеопам'яті, StaticMeshes можна перекласти, обернути та масштабувати, але вони не можуть будь-яким чином анімувати вершини. StaticMeshes є основною одиницею, що використовується для створення геометрії світу для віртуальних середовищ, створених в Unreal Engine.

Після цього етапу проводиться накладання шейдерів на ландшафт та Meshes. В спрощеному вигляді, текстурування 3D-середовища в Unreal Engine є процесом використання готових матеріалів шляхом накладання їх на ландшафт та готові об’єкти. При текстуруванні матеріалу вперше, його шейдери компілюються. Потім, цей процес повторюється лише після зміни матеріалу. Тривалість компіляції безпосередньо залежить від складності матеріалу.

Наступним етапом є побудова графічних та фізичних ефектів в 3D-середовищі та їх подальше розташування, а також додавання та запікання світла. Unreal Engine підтримує використання різних, попередньо-розроблених сценаріїв освітлення в середовищі. Це дає змогу на одному рівні зберігати та відображати декілька налаштувань освітлення, що надає динамічному освітленню гнучкості. Можливість обирати між декількома сценаріями попереднього освітлення має особливе значення для проектів віртуальної реальності, які потребують високоякісної обробки з дотриманням високих показників продуктивності.

Під "запіканням світла" розуміють процес попередньої генерації та накладання тіней. Об'єкти, які були розташовані до цього процесу, стануть статичними і більше не будуть активно прораховуватися. Це зменшує навантаження при розрахунку 3D-сцени. Однак слід пам’ятати, що процес запікання потрібно проводити до розташування активних об’єктів, які потребують динамічних тіней.

Важливим є додавання персонажу з виглядом від першого лиця, оскільки саме він є аватаром користувача у віртуальному середовищі та дозволяє йому орієнтуватися в 3D-сцені.

Перш ніж проект Unreal Engine може бути розповсюджений серед користувачів, він повинен бути належним чином "запакований". Це гарантує, що весь код та вміст оновлено та реалізовано в належному форматі для роботи на потрібній цільовій програмно-апаратній платформі.

ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС ДЛЯ ПОБУДОВИ 3D-СЦЕН

Для відтворення віртуальної реальності в 3D-сцені на основі рушія Unreal Engine виявилися потрібними та були використані наступні програмно-апаратні засоби:

- персональний комп'ютер з графічною картою від NVIDIA, що підтримує програмний компонент Geforce Experience та його технологія GameStream;
- смартфон на операційній системі андроїд з необхідними сенсорами та кабелем MicroUSB для підключення до персонального комп'ютера (ПК);
- програмні компоненти Trinus VR, Moonlight, Tridef 3D;
- 3D середовище.

Першим етапом є встановлення на смартфоні Trinus VR та Google Card Board. Програмний компонент Trinus VR надає користувачеві смартфону гарнітуру віртуальної реальності без необхідності придбання додаткових апаратних засобів. Trinus VR використовує дисплей та давачі смартфону, щоб перетворити його у шолом віртуальної реальності для персонального комп'ютера.

Компонент Trinus VR необхідний для передачі даних гіроскопу (переміщення смартфона у просторі) на комп'ютер. Програмний компонент складеться з двох частин. Першу потрібно встановити на смартфон, а іншу на комп'ютер. Наступним кроком є підключення смартфону до ПК для подальшої передачі даних. Trinus VR підтримує два можливих варіанта підключення: через Wi-Fi або через USB-кабель.

Geforce Experience – утиліта розроблена компанією NVIDIA для програмної підтримки персональних комп'ютерів на базі її відеокарт. Основна ціль використання утиліти GeForce Experience є технологія NVIDIA GameStream. Саме вона надає змогу транслювати зображення з персонального комп'ютера на інший гаджет. В розглянутому випадку це смартфон на операційній системі Android, який складає основу шолома для віртуальної реальності на базі Google Cardboard.

Використання GeForce Experience є ефективним способом передачі зображення з персонального комп'ютера на шолом Google Cardboard, оскільки даний програмний компонент отримує відеопотік не з монітору, а з відеоадаптера NVIDIA. Відеопотік з самого відеоадаптера є набагато якіснішим, тому зображення у шоломі VR, що використовує такий метод, набагато чіткіше. До того ж технологія GameStream ніяк не навантажує центральний процесор чи смартфон, а лише малу кількість ядер відеоадаптера.

Однак, технологія GameStream офіційно підтримується лише на мобільних пристроях лінійки Shield від компанії NVIDIA. Тому для вирішення цього недоліку доцільним виявилось використання програмного забезпечення для смартфона Moonlight.

Moonlight – це реалізація протоколу NVIDIA GameStream з відкритим кодом. Розробники цього програмного продукту реалізували протокол, який використовує NVIDIA Shield та виконує набір корисних функцій. Moonlight дозволяє в режимі потоку передавати відеопотік 3D середовища з сумісного GameStream ПК на будь-який підтримуваний пристрій та дистанційно відтворювати його. Для користувача, це означає, що для передачі відеопотоку з персонального комп'ютера утилітою GameStream не обов'язково мати NVIDIA Shield. Також Moonlight має великий спектр налаштувань, таких як вибір роздільної здатності разом з частотою кадрів та вибір бітрейту відеопотоку. Крім того, система динамічного розширення, дозволяє стабілізувати кількість кадрів в секунду за рахунок зменшення якості зображення в необхідний момент.

Останнім етапом є налаштування програмного компоненту Tridef 3D. Програмний компонент Tridef 3D – спеціальний 3D-драйвер з певною конфігурацією. За допомогою цього драйвера, на екран смартфона буде виводитись зображення в режимі роздвоєння зображення, з функцією створення ефекту 3D-глибини в 3D-середовищі.

Після цього потрібно лише додати створене 3D-середовище та запустити його. Як тільки середовище завантажиться, воно буде реалізовано уже з необхідним ефектом роздвоєння зображення. Також, завдяки системі трекінгу можливим стає й керування камерою.

ПОБУДОВА ЕФЕКТУ ГРОЗИ У 3D-СЕРЕДОВИЩІ ІЗ ЗАСТОСУВАННЯМ ВІДЕОАДАПТЕРА

При побудові 3D-середовища було поставлено задачу відтворити складний візуальний ефект грози шляхом використання системи частинок. Ефект грози складався з частинок крапель дощу, частинок бризок від крапель на поверхні, частинок опалого листя з дерев, частинок блискавок, та частинок легкого туману.

Вищенаведені частинки було реалізовано двома способами: на основі CPU-ядер та на основі GPU-ядер. При реалізації на основі CPU-ядер частинки обробляються і обчислюються цілком за допомогою центрального процесора. При використанні GPU-ядер частинки спочатку з'являються на центральному процесорі, а потім обробляються і обчислюються цілком за допомогою графічного відеоадаптера. Перевага використання GPU полягає в тому, що за рахунок великої кількості ядер у сучасних відеоадаптерах, багато тисяч частинок можуть оброблятися одночасно, що дозволяє використовувати набагато більш щільні і деталізовані системи ефектів.

В обох випадках частинки загалом поводяться аналогічно, але все ж таки мають деякі ключові відмінності. Деякі функції, доступні на CPU (наприклад, світловипромінювання, контроль параметрів матеріалу і модулі Attraction), не підтримуються на GPU. Однак, обчислення частинок на GPU-ядрах є набагато ефективнішим методом. Візуальний ефект, що побудований на CPU-ядрах, може нормально відтворювати лише тисячі частинок. При обробці більшої кількості частинок, частина з них може не візуалізовуватись через обмеження у потужності CPU. Візуальний ефект, який побудований на GPU-ядрах, дозволяє створювати десятки і сотні тисяч частинок без значного впливу на продуктивність, що дуже важливо при відтворенні масштабних ефектів, таких як снігопад, дощ, іскри тощо. Нижче наведено приклад оброблення набору з 50000 частинок дощу на GPU-ядрах (рис. 3). При обробленні даного прикладу на CPU-ядрах, відображався переважно чорний екран із невеликими досить рідкими вкрапленнями частинок дощу, що свідчить про неякісне відображення ефекту.

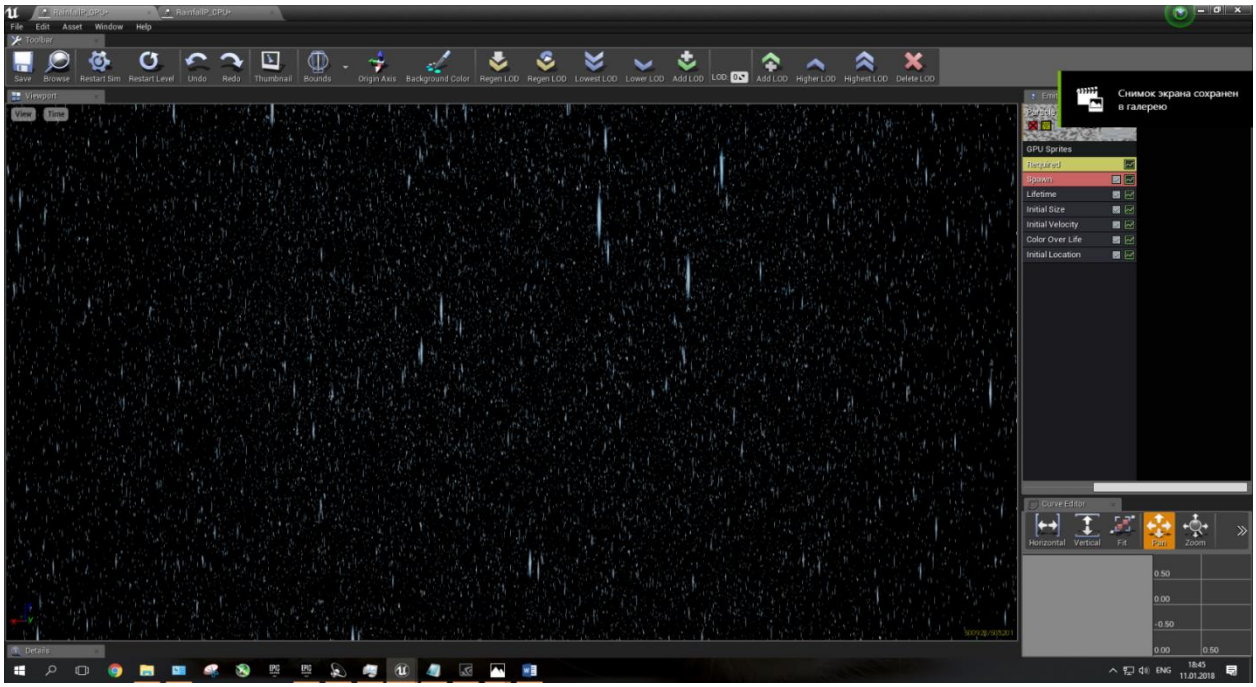

Рис. 3 – Приклад оброблення на GPU набору з 50 000 частинок дощу

Незважаючи на те, що реалізовані на GPU-ядрах частинки не підтримують всі функції, доступні для традиційних частинок центрального процесора, але вони забезпечують більшу ефективність, а також кілька унікальних функцій. Частинкам на основі GPU-ядер також можна призначати початкові атрибути, такі як розмір і швидкість, використовуючи методи, доступні традиційним частинкам на основі CPU-ядер. Рух частинок визначається законами ньютонівської динаміки. На кожному часовому кроці положення і швидкість частинки інтегруються вперед в залежності від її поточного стану, швидкості, постійного прискорення, сили опору тощо.

Найцікавішою особливістю реалізації на GPU-ядрах, окрім її ефективності, є наявність векторного поля. Векторне поле являє собою однорідну сітку векторів, яка впливає на рух частинок. Векторні поля динамічні і можуть бути переміщені в будь-який час. Поле також може бути вміщено в каскад (локальне векторне поле). Коли частинка входить в межі векторного поля, то поле буде впливати на її рух, і відповідно, коли частинка покидає межі поля, його вплив зникає.

Експериментальні дослідження проводились у спеціально побудованій 3D сцені у трьох режимах: без використання візуальних погодних ефектів, з використанням візуальних погодних ефектів побудованих на основі CPU-ядер та з використанням візуальних погодних ефектів побудованих на основі GPU-ядер. В ході експериментальних досліджень вимірювалась частота кадрів (fps) обробки побудованої 3D сцени, тобто частота, з якою на екрані з'являються послідовні зображення (кадри).

Кожен експеримент здійснювався протягом 60 секунд. Частота кадрів вимірювалась за допомогою програмного засобу Fraps. Параметрами частоти кадрів були кількість кадрів за весь час експерименту, мінімальна та максимальна частота кадрів (fps), середня частота кадрів (fps) за весь час експерименту (табл. 1, рис. 4). Експериментальні дослідження здійснювались на ПК такої конфігурації: операційна система – Windows 10 version 1703; центральний процесор – Intel Core I5 4690K (4 ядерна архітектура, 4 потоки, частота 3.5 GHz); відеоадаптер – Nvidia Geforce GTX980 (2048 ядер архітектури

Maxwell, 4Gb відеопам'яті, частота відеопроцесора 1152 Hz, частота відеопамяті 7010 Hz); оперативна пам'ять – 8Gb, 1866 Hz.

Таблиця 1 – Результати тестування

	Час експери-менту, мс	Сумарна кількість усіх кадрів	Мінімальна частота, кадрів/сек.	Максимальна частота, кадрів/сек.	Середня частота, кадрів/сек.
Без частинок	60000	4002	60	68	66.700
З частинками на основі GPU	60000	3869	60	67	64.483
З частинками на основі CPU	60000	3494	51	66	58.233

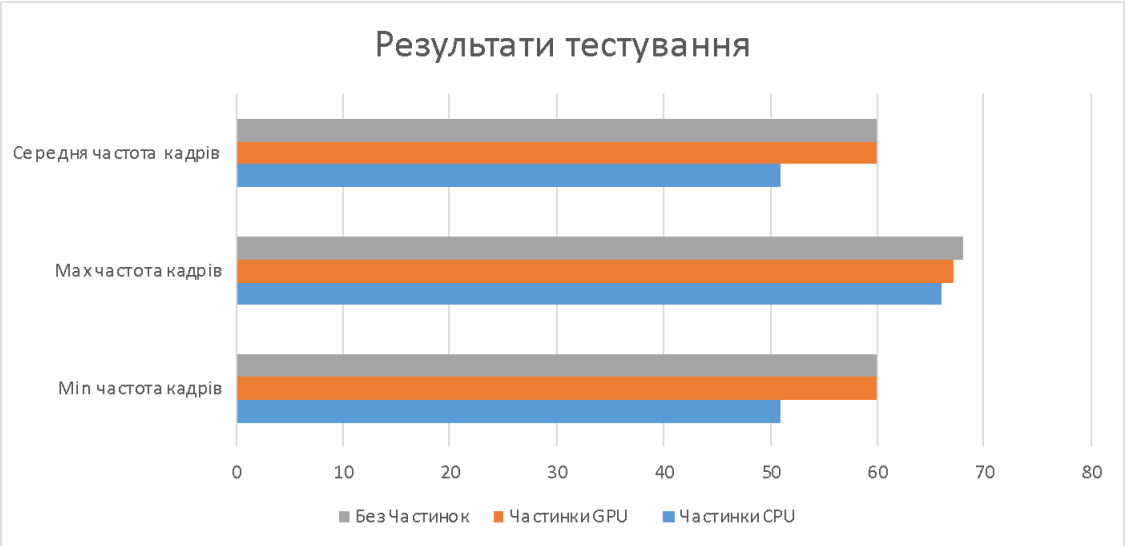

Рис. 4 – Результати тестування

ВИСНОВКИ

Отримані результати досліджень підтверджують доцільність використання GPU-орієнтованої програмно-апаратної платформи в процесі побудови та обробки 3D-сцен. Збільшення обчислювальної продуктивності, яке досягаються за рахунок великої кількості ядер GPU, дозволяє обробляти більш складні об'єкти та ефекти зі збереженням сталої частоти кадрів, що особливо важливо для ВР-додатків. Окрім того, перспективними є подальші дослідження у напрямку поглибленого застосування технологій гетерогенних обчислень, таких як Multi-GPU Programming, для вирішення вищенаведених задач.

СПИСОК ЛИТЕРАТУРИ

1. Що таке віртуальна реальність? [Електронний ресурс]. Режим доступу - https://futurum.today/shcho-take-virtualna-realnist
2. Небожанов О.В. (2017). Аналіз програмних засобів створення 3D сцен для віртуальної реальності. Молодь в технічних науках: дослідження, проблеми, перспективи. III Міжнародна науково-практична інтернет-конференція.
3. Яровий А. А., Кулик О. О., Кокряцька Н. І. (2015) Паралельно-ієрархічне перетворення плямоподібних зображень на основі Multi-GPU систем. Інформаційні технології та комп'ютерна інженерія, 34, 3, 72-80.
4. Пасічник Д.Г., Яровий А.А. (2016, березень) Аналіз підходів до підвищення якості растрових зображень на основі технології GPGPU. XLV Науково-технічна конференція Вінницького національного технічного університету, Вінниця, ВНТУ.
5. WorldMachine 2 [Електронний ресурс]. Режим доступу – http://www.mrbluesummers.com/324/blog/world-machine-2-review-2
6. Textures [Електронний ресурс]. Режим доступу – https://docs.unrealengine.com/latest/INT/Engine/Content/Types/Textures/

REFERENCES

1. 1. What is a virtual reality? [Electronic resource]. Access mode - https://futurum.today/shcho-take-virtualna-realnist
2. 2. Nebozhanov O.V. (2017). Analysis of software for creating 3D scenes for virtual reality. Youth in technical sciences: research, problems, perspectives. III International Scientific and Practical Internet Conference.
3. 3. Yarovoy AA, Kulik O., Kokryatskaya NI (2015) Parallel-hierarchical transformation of spot-based images based on Multi-GPU systems. Information Technology and Computer Engineering, 34, 3, 72-80.
4. 4. Pasichnyk DG, Yarovoy AA (2016, March) Analysis of approaches to improving the quality of raster images based on GPGPU technology. XLV Scientific and Technical Conference of Vinnytsia National Technical University, Vinnitsa, VNTU.
5. 5. WorldMachine 2 [Electronic resource]. Access Mode - http://www.mrbluesummers.com/324/blog/world-machine-2-review-2
6. 6. Textures [Electronic resource]. Access Mode - https://docs.unrealengine.com/latest/INT/Engine/Content/Types/Textures/

Надійшла до редакції 28.11.2018 р.

АНДРІЙ АНАТОЛІЙОВИЧ ЯРОВИЙ – д.т.н., професор, завідувач кафедри комп'ютерних наук, Вінницький національний технічний університет, Вінниця, Україна.

ЛЮДМИЛА ВІКТОРІВНА КРИЛИК – к.т.н., доцент, доцент кафедри комп'ютерних наук, Вінницький національний технічний університет, Вінниця, Україна.

ОЛЕКСАНДР ОЛЕКСАНДРОВИЧ КУЛИК – аспірант кафедри комп'ютерних наук, Вінницький національний технічний університет, Вінниця, Україна.

ДМИТРО ГЕННАДІЙОВИЧ ПАСІЧНИК – аспірант кафедри комп'ютерних наук, Вінницький національний технічний університет, Вінниця, Україна.

ОЛЕКСАНДР ВОЛОДИМИРОВИЧ НЕБОЖАНОВ – магістр кафедри комп'ютерних наук, Вінницький національний технічний університет, Вінниця, Україна.