Відвідувати об’єкти в віртуальному світі можливо внаслідок використання спеціальних (Hello Mars, Jaunt VR, Ocean Rift, End Space VR) програмних застосунків, які працюють під мобільні пристрої та з використання мобільних веб-браузерів. Варто зазначити, віртуальні тури, як допоміжна частина туристичної активності, використовується різними економічними суб’єктами для підтримки та розвитку своєї основної діяльності. Також, формування в перспективі віртуальної індустрії, віртуальні тури стануть кінцевим продуктом споживання користувачів (наприклад, в Діснейленді, Орландо (США) атракціон, де людина потрапляє в симуляцію польоту над природними ландшафтами з використанням відеоряду та різних спецефектів, від ароматів та звукового супроводу до ілюзії вітру, швидкості й напрямку, які регулюються в залежності від картинки на екрані).

Таким чином, 3D-тури надають економію часу; детальний перегляд всіх наявних об’єктів; оригінальність та привабливість в рекламуванні способом; цілодобову доступність (сферичні панорами, які розміщені на сайтах, доступні в будь-який час доби для перегляду); різноманітність використання одних і тих самих турів, як в глобальній мережі, так і для демонстрування матеріалів.

Список використаних джерел

1. Программы для создания виртуальных туров. URL: http://compress.ru/article. aspx?id=15669. (дата звернення: 02.11.2019).

2. 3D-туры: что это такое, и в чём их преимущества. URL: http://3d-bel.ru/about-3d-tours. (дата звернення: 02.11.2019).

3. Романова М. М. Инновации в индустрии туризма. URL: https://tourlib.net/statti_tourism /romanova2.htm. (дата звернення: 02.11.2019).

ОГЛЯД NVIDIA CUDA (COMPUTE UNIFIED DEVICE ARCHITECTURE), ЙОГО ОСОБЛИВОСТІ ТА ЗАСТОСУВАННЯ

Сеньків Арсен Ігорович
магістрант спеціальності Середня освіта (Інформатика)
Тернопільський національний педагогічний університет імені Володимира Гнатюка
senkiv_ai@fizmat.tnpu.edu.ua

Струк Оксана Олегівна
кандидат фізико-математичних наук, доцент кафедри інформатики та методики її навчання
Тернопільський національний педагогічний університет імені Володимира Гнатюка
oksana.struk@gmail.com

CUDA (*Compute Unified Device Architecture*) – це платформа паралельних обчислень розроблена компанією NVIDIA, що дозволяє значно збільшити кількість обрахунків використовуючи потужність GPU (графічного процесора) [1].

Застосування

NVIDIA PhysX – завдяки CUDA більшість обчислень, пов’язаних з фізикою, виконуються відеокартою.

Обробка відеопотку в режимі реального часу – завдяки CUDA з’явилась можливість реалізації алгоритмів одночасного слідкування за багатьма об’єктами у відеопотоці.

CUDA-версія пакету стиснення текстур NVIDIA Texture Tools 2 – друга версія цього пакету підтримує алгоритми стиску BC4 та BC5, реалізованих в технології DirectX 11. Завдяки застосуванню CUDA швидкість стиску і розпакування текстур збільшилась в 12 разів.

Конвертація відео – завдяки CUDA конвертація відбувається в 3–4 рази швише.

Фільтрація аудіо – реалізація FIR-фільтра на CUDA пришвидшила його роботу в сотні разів.

Переваги CUDA над традиційних шляхами GPGPU обчислень.

Інтерфейс програмування додатків CUDA базується на мові програмування C з розширеннями, що спрощує процес вивчення і впровадження архітектури CUDA.

CUDA забезпечує доступ до розділеної між потоками пам'яті обсягом 16 Кб на мультипроцесор, яку можна використати для організування кешу з широкою смугою пропускання, відносно текстурних вибірок.

Ефективніше передавання даних між системою і пам'яттю відеокарти.

Не вимагає надмірних графічних API.

Лінійна адресація в пам'яті, і *gather* і *scatter*, можливість запису в довільну адресу.

Апаратна підтримка цілочисельних і бітових операцій.

Основні обмеження CUDA

Не підтримує рекурсію для виконуваних функцій.

Мінімальна ширина блоку в 32 потоки.

Закрита архітектура CUDA.

CUDA і мова C

Технологія CUDA вводить ряд додаткових розширень для мови C, що необхідні для написання коду для GPU:

Специфікатори функцій – вказують як і звідки будуть виконуватись функції.

Специфікатори змінних – вказують тип використовуваної пам'яті GPU.

Специфікатори запуску ядра GPU.

Вбудовані змінні для ідентифікації ниток, блоків та інших параметрів при використанні коду в ядрі GPU.

Додаткові типи змінних.

Всього в CUDA є три специфікатори функцій:

__host__ виконується в CPU, викликається з CPU.

__global__ – виконується в GPU, викликається з CPU.

__device__ виконується в GPU, викликається з GPU.

Специфікатори запуску ядра використовують для опису кількості блоків, ниток і пам'яті, які потрібно виділити для розрахунків в GPU. Синтаксис запуску ядра має такий вигляд:

myKernelFunc<<<gridSize, blockSize, sharedMemSize, cudaStream>>>(float param1, float* param2)*,

gridSize – розмірність сітки блоків (dim3), виділеної для розрахунків.

blockSize – розмір блоку (dim3), виділеного для розрахунків.

sharedMemSize – розмір додаткової пам'яті, що виділиться при запуску ядра.

cudaStream – змінна *cudaStream_t*, яка задає потік, в якому відбудеться виклик.

myKernelFunc – функція ядра (специфікатор *__global__*).

Деякі змінні при виклику ядра можна опустити, а саме *sharedMemSize* і *cudaStream*.

Вбудовані змінні:

gridDim – розмір сітки, тип *dim3*. Дозволяє дізнатись розмір сітки, виділеної при запуску ядра.

blockDim – розмір блоку, тип *dim3*. Дозволяє дізнатись розмір блоку, виділеного при запуску ядра.

blockInx – індекс поточного блоку в обрахунках GPU, тип *uint3*.

threadInx – індекс поточної нитки в обрахунках GPU, тип *uint3*.

warpSize – розмір warp-а, має тип *int*.

CUDA host API

CUDA host API – це сполучна ланка між CPU та GPU. Її можна розділити на API низького рівня під назвою CUDA driver API, що надає доступ до драйвера користувацького режиму CUDA та API високого рівня – CUDA runtime API.

CUDA runtime API включає в себе такі функції [2]:

Device Management – включає функції для загального керування GPU (отримання інформації про можливості GPU, перемикання між GPU при роботі в SLI-режимі).

Version Management – включає функції керування версіями інтерфейсу.

Thread Management – керування нитками.

Stream Management – керування потоками.

Event Management – функція створення і керування подіями.

Execution Control – функції запуску і виконання ядра CUDA.

Memory Management – функції керування пам'яттю GPU.

Texture Reference Manager – робота з об'єктами текстур через CUDA.

Graph Management – функції створення і керування графами.

OpenGL Interoperability – функції взаємодії з OpenGL API.

Direct3D 9 Interoperability – функції взаємодії з Direct3D 9 API.

Direct3D 10 Interoperability – функції взаємодії з Direct3D 10 API.

Direct3D 11 Interoperability – функції взаємодії з Direct3D 11 API.

Error Handling – функції обробки помилок.

CUDA driver API має подібні до CUDA runtime API функції окрім *Thread Management* та включає в себе [3]:

Initialization – включає функції ініціалізації інтерфейсу низького рівня CUDA.

Context Management – включає функції управління контекстом інтерфейсу низького рівня CUDA.

Virtual Memory Management – функції керування віртуальною пам'яттю GPU.

Таким чином, обчислення з використанням графічних адаптерів показують максимальну ефективність в завданнях, які не потребують інтенсивного звернення до пам'яті. Але, якщо завдання потребує великої кількості пам'яті (кілька гігабайт), то, на даному етапі розвитку технології CUDA її не доцільно вирішувати за допомогою GPU.

Список використаних джерел

1. NVIDIA Corporation. NVIDIA CUDA GPUs NVIDIA Corporation URL: https://developer.nvidia.com/cuda-gpus. (дата звернення 15.03.2020).
2. NVIDIA Corporation. CUDA runtime API / NVIDIA Corporation. 2019. URL: https://docs.nvidia.com/cuda/cuda-runtime-api. (дата звернення 18.03.2020)..
3. NVIDIA Corporation. CUDA driver API / NVIDIA Corporation. 2019. URL: https://docs.nvidia.com/cuda/cuda-driver-api. (дата звернення 15.03.2020).

ВИКОРИСТАННЯ ПЛАТФОРМИ ARDUINO У НАВЧАЛЬНІЙ ДІЯЛЬНОСТІ УЧНІВ

Стефанюк Ярослав Олегович
магістрант спеціальності Середня освіта (Фізика)
Тернопільський національний педагогічний університет імені Володимира Гнатюка
м. Тернопіль, Україна
poterjashka42@gmail.com
Федчишин Ольга Михайлівна
кандидат педагогічних наук, викладач кафедри фізики та методики її навчання
Тернопільський національний педагогічний університет імені Володимира Гнатюка
м. Тернопіль, Україна
olga.fedchishin.77@gmail.com

Використання Arduino у навчально-виховному процесі розкриває нові можливості для учнів, а саме реалізацію проектно-дослідницької діяльності в освітньому процесі, розвиток творчих здібностей учнів, індивідуалізацію навчальної діяльності. Загалом, учні добре реагують на навчальні дисципліни, які включають в себе програмування роботів. Це не підтверджено дослідженнями, але ресурси для батьків і учителів, що стосуються робототехніки, з кожним днем набувають все більшої популярності.

Платформа Arduino – це сімейство мікроконтролерів для легкого створювання автоматики та робототехніки. Початкова ціль Arduino – це навчання. Учням набагато цікавіше вчитися, якщо вони можуть застосувати нові знання та вміння на практиці й побачити результати своїх старань. Це набагато цікавіше, аніж слухати «суху» теорію на уроках.

Програма для Arduino називається «скетч», яка створюється в програмному середовищі Arduino IDE та програмується власною мовою Arduino wiring яка є спрощеною C++ та наслідує від неї синтаксис. За допомогою Arduino проектів можливе ефективне використання мобільних телефонів під час навчального процесу. Оскільки учні можуть керувати створеними пристроями за допомогою своїх смартфонів. Для прикладу взято стандартний скетч Arduino IDE «blink», який вмикає та вимикає апаратний світлодіод на певний час. За основу взято