МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №2
по курсу «Программирование графических процессоров»

Обработка изображений на GPU. Фильтры.

Выполнил: Семин А. В.
Группа: 8О-406Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2023

Условие

1. **Цель работы.** Научиться использовать GPU для обработки изображений. Использование текстурной памяти и двухмерной сетки потоков.
2. **Вариант 4. SSAA.**
   Необходимо реализовать избыточную выборку сглаживания. Исходное изображение представляет собой “экранный буфер”, на выходе должно быть сглаженное изображение, полученное уменьшением исходного.
   Входные данные. На первой строке задается путь к исходному изображению, на второй, путь к конечному изображению. На следующей строке, два числа wn и hn - размеры нового изображения, гарантируется, что размеры исходного изображения соответственно кратны им. $w*h \leq 4 * 10^8$.

Программное и аппаратное обеспечение

Графический процессор (GeForce GTX 1650 Ti)

1. Количество потоковых процессоров: 1024
2. Частота ядра: 1350 МГц
3. Частота в режиме Boost: 1485 МГц
4. Количество транзисторов: 6,600 млн
5. Тип памяти: DDR6
6. Видеопамять: 4096 МБ
7. Частота памяти: 12000 МГц

Процессор AMD Ryzen 7 4800H

1. ядра: 8
2. потоки: 16
3. частота: 2.9 ГГц
4. максимальная частота: 4.2 ГГц
5. кэш 1 уровня: 64 КБ (на ядро)
6. кэш 2 уровня 512 КБ (на ядро)
7. кэш 3 уровня: 8 МБ (общий)

16 ГБ ОЗУ и 512 ГБ SSD.
OS – Windows 11Домашняя, WSL, IDE – VS Code, Compiler - nvcc, g++.

Метод решения

Считываем из стандартного потока ввода название входного и выходного файлов, результирующие размеры w и h изображения после преобразования. Считываем из файла его размеры и бинарные данные. Затем создаем текстурный объект. SSAA сглаживание выполняется следующим образом внутри ядра: у изображения для каждого блока пикселей размера (w/wn) x (h/hn) считаем среднее значение по каждой оси x, y, z. Результат записываем в выходной массив. После работы ядра копируем полученные данные с GPU на CPU и записываем в выходной файл.

Описание программы

Программа состоит из одного файла, в котором находятся функция ядра и main. В главной функции выполняются описанные выше действия считывания, копирования и записи данных, вызов выполнения функции ядра, а также создание текстурного объекта, выделение и очистка памяти.

Сигнатура функции ядра:

__global__ void ssaa_smoothing(cudaTextureObject_t tex, uchar4 *dev, const ImageSize out_img, int w_diff, int h_diff)

Результаты

Время выведено в ***миллисекундах***.

Конфигурация	Размер тестового файла				
	$80 * 80$	$250 * 250$	$900 * 900$	$2200 * 2200$	$9000 * 9000$
CPU	24	248	4518	27266	586179
<<<(16, 16), (16,16) >>>	0.084288	0.084064	0.136000	0.313920	3.573856
<<<(32, 32), (32,32) >>>	0.083968	0.103200	0.161120	0.271584	2.991040
<<<(64, 64), (32,32) >>>	0.144096	0.144544	0.175232	0.370176	2.949792
<<<(256, 256), (32,32) >>>	1.025536	1.033984	1.082496	1.295008	4.343904
<<<(512, 512), (32,32) >>>	3.857312	3.740160	3.948448	4.056832	7.116896

Примеры работы

Исходная картинка (в приближении):

Результат обработки (в приближении):

Размер исходной картинки: *856 x 574* пикселей.
Размер обработанной картинки: *426 x 287* пикселей.
Как видно из результата, около прямой вместо обычной лесенки появилась небольшая окантовка из пикселей серого оттенка, что и является показателем совершенного сглаживания.

Выводы

Данный алгоритм широко применяется во многих областях, окружающих нас. Среди них такие, как видеоигры, компьютерная графика и визуализация, фото- и видеомонтаж.

В ходе выполнения лабораторной работы главной сложностью было разобраться в работе алгоритма и понять, как реализовать ядро на основе теоретических знаний о работе SSAA. Также, как видно из таблицы с результатами, этот алгоритм работает намного быстрее на GPU по сравнению с CPU.