МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №2
по курсу «Программирование графических процессоров»

Обработка изображений на GPU. Фильтры.

Выполнил: А.В. Куликов
Группа: 8О-408Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2020

Условие

Цель работы: Научиться использовать GPU для обработки изображений. Использование текстурной памяти.

Вариант 4. SSAA.

Необходимо реализовать избыточную выборку сглаживания. Исходное изображение представляет собой “экранный буфер”, на выходе должно быть сглаженное изображение, полученное уменьшением исходного.

Программное и аппаратное обеспечение

Видеокарта	GeForce GT 545
Compute capability	2.1
Графическая память	3004 Мб
Разделяемая память	48 Кб
Константная память	64 Кб
Количество регистров на блок	32768
Максимальное кол-во блоков	65535*65535*65535
Максимальное кол-во нитей в блоке	1024
Кол-во мультипроцессоров	3
Ядер CUDA	144

Процессор	Intel Core i7-3770
ОЗУ	16 Гб
ЖД	

Операционная система	Ubuntu 16.04.6 LTS
IDE	VS Code
Компилятор	nvcc V7.5.17

Метод решения

Вычисляем размеры окна как ширина/новая ширина, высота/новыя высота.

Проходим по изображению окном, для каждой позиции вычисляем среднее арифметическое для каждой из компонент. Это и будет новое значение соответствующего пикселя в выходном изображении.

Описание программы

В файле main.cu находится основная логика программы, включая ввод/вывод, работа с памятью, а также само ядро для вычисления на ГПУ.

В файле error.h находится функционал для обработки ошибок CUDA API и хендлеры для стандартных сигналов SIGSEGV, SIGABRT.

В файле benchmark.h находится функционал для измерения времени исполнения кода.

Поставленную задачу решает ядро SSAA. В нем вычисляются вертикальный индекс idy и горизонтальный индекс idx, на их основе каждой нити ставятся в соответствие клетки, на основе которых будут вычисляться новые пиксели изображения при помощи функции avg_kernel.

В функции avg_kernel просто вычисляются средние значения по всему окну для каждого канала и выдается выходной пиксель.

При этом исходное изображение содержится в текстурной памяти, что позволяет достичь большей скорости обращения к памяти за счет ее кеширования.

Результаты

Тестирование ядер с различными конфигурациями
Маленький файл, полученный из bmp-файла 800x600 (1,8 мб)

---	32 блок	64 блок	128 блок	256 блок	512 блок	1024 блок
1 грид	11.675872	4.636832	2.562176	1.194112	0.772928	0.545056
32 грид	1.001376	0.431584	0.289472	0.210464	0.232992	0.254816
128 грид	0.629728	0.308160	0.216384	0.187936	0.256000	0.299776
256 грид	0.569120	0.289728	0.224096	0.233888	0.290400	0.378720
512 грид	0.568064	0.295616	0.262496	0.240704	0.340576	0.484544
1024 грид	0.585280	0.306432	0.269152	0.284256	0.437120	0.738464

Лучший результат: 0.187936 мс при параметрах <128, 256>

Средний файл, полученный из bmp-файла 1920x1080 (7,9 мб)

---	32 блок	64 блок	128 блок	256 блок	512 блок	1024 блок
1 грид	50.388672	19.768192	10.757184	5.005984	3.114400	2.105184
32 грид	4.077952	1.765792	1.199040	0.791584	0.897600	0.896064
128 грид	2.610400	1.193792	0.894112	0.765952	0.874720	0.909152
256 грид	2.373568	1.123296	0.878368	0.767232	0.896896	1.000928
512 грид	2.361792	1.132704	0.892288	0.781760	0.971840	1.151136
1024 грид	2.326080	1.079392	0.901472	0.811680	1.078080	1.436960

Лучший результат: 0.765952 мс при параметрах <128, 256>

Большой файл, полученный из bmp-файла 6000x4000 (91,6 мб)

---	32 блок	64 блок	128 блок	256 блок	512 блок	1024 блок
1 грид	582.685364	226.815521	121.469597	57.807297	35.182911	23.967936
32 грид	48.603489	20.614336	13.647168	8.719456	10.036640	9.878048
128 грид	30.822975	13.808416	10.161088	8.535232	9.279968	9.271456
256 грид	27.240864	12.471488	9.649728	8.345152	9.275776	9.223712
512 грид	27.201504	12.438432	9.571776	8.281344	9.363648	9.314688
1024 грид	26.581087	12.157248	9.544672	8.316896	9.503872	9.672256

Лучший результат: 8.281344 мс при параметрах <512, 256>

Сравнение с CPU
Маленький файл: вектор из 100 чисел
Результат: 4.962685 мс

Средний файл: вектор из 10000 чисел
Результат: 20.675654 мс

Большой файл: вектор из 1000000 чисел
Результат: 151.049954 мс

Результаты работы программы
Исходное изображение

Результат

Выводы

Данный алгоритм является самостоятельным алгоритмом сглаживания. Это самое качественное, но, к сожалению, самое вычислительно сложное сглаживание. Он может быть использован (и даже использовался) во всевозможных компьютерных играх, возможно, в 3d-софте.

Особых проблем в решении задачи не возникало. В решении данной задачи программа с использованием GPU значительно превзошла реализацию, работающую только на CPU.