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: *А.А. Почечура*

Группа: *8О-406Б*

Преподаватели: К.Г. Крашенинников,

А.Ю. Морозов

Москва, 2023

**Условие**

**Цель работы.** Научиться использовать GPU для обработки изображений. Использование текстурной памяти и двухмерной сетки потоков.

**Вариант 5. Выделение контуров. Метод Робертса.**

**Формат изображений.** Изображение является бинарным файлом, со следующей структурой:



В первых восьми байтах записывается размер изображения, далее построчно

все значения пикселей, где

● r -- красная составляющая цвета пикселя

● g -- зеленая составляющая цвета пикселя

● b -- синяя составляющая цвета пикселя

● a -- значение альфа-канала пикселя

Пример картинки размером 2 на 2, синего цвета, в шестнадцатеричной записи:

02000000 02000000 0000FF00 0000FF00 0000FF00 0000FF00.

Студентам предлагается самостоятельно написать конвертер на любом языке программирования для работы с вышеописанным форматом.

В данной лабораторной работе используются только цветовые составляющие изображения (r g b), альфа-канал не учитывается. При расчетах значений допускается ошибка в ±1. Ограничение: w < и h < . Во всех вариантах, кроме 2-го и 4-го, в пограничном случае, необходимо “расширять” изображение за его границы, при этом значения соответствующих пикселей дублируют граничные. То есть, для любых индексов i и j, координаты пикселя [ip, jp] будут определятся следующим образом:

ip := max(min(i, h), 1)

jp := max(min(j, w), 1)

**Входные данные.** На первой строке задается путь к исходному изображению, на второй, путь к конечному изображению. w\*h ≤

**Программное и аппаратное обеспечение**

**Графический процессор:**

Compute capability: 7.5

Name: Tesla T4

Total Global Memory: 15835398144

Shared memory per block: 49152

Registers per block: 65536

Warp size: 32

Max threads per block: (1024, 1024, 64)

Max block: (2147483647, 65535, 65535)

Total constant memory: 65536

Multiprocessors count: 40

**Процессор:**

vendor\_id : GenuineIntel

cpu family: 6

model: 85

model name: Intel(R) Xeon(R) CPU @ 2.00GHz

stepping: 3

microcode : 0xffffffff

cpu MHz: 2000.184

cache size : 39424 KB

physical id: 0

siblings: 2

core id: 0

cpu cores : 1

apicid: 0

initial apicid: 0

fpu: yes

fpu\_exception: yes

cpuid level: 13

wp : yes

bogomips : 4000.36

clflush size: 64

cache\_alignment : 64

address sizes: 46 bits physical, 48 bits virtual

**Оперативная память и жёсткий диск:**

Memory:

description: System memory

physical id: 0

size: 13GiB

Mem**:** 12982.6 MiB

Storage**:** 55 GiB

**Программное обеспечение:**

*Google Colab*

Операционная система: Ubuntu 22.04.2 LTS

Оболочка: python3

inxi: 3.3.13

**Метод решения**

В данном варианте производится выделение контуров объектов на картинке. Для этого нужно пройтись по всем пикселям и получить яркость каждого из них. Далее нужно вычислить градиент для каждого пикселя, который задаётся методом Робертса. Нужно найти разность яркостей между элементами, находящимися на диагоналях окошка. Само окошко, соответственно используемому методу, имеет размер 2х2. Так как для вычисления нового значения для конкретного пикселю требуется использовать значения других пикселей, находящихся в том числе в следующей строке, была применена двумерная сетка потоков. После вычисления градиента обновляем параметры пикселя (вычисленное значение заносится на все цветовые составляющие) и получаем обработанное изображение.

**Описание программы**

В начале требуется преобразовать данное на входе изображение в бинарный файл. Затем передадим изображение в виде бинарного файла на графический процессор. В ядре *kernel* проходимся по всем пикселям изображения с помощью двумерной сетки потоков размера << 16, 16 >, < 32, 32 >> и считаем значения яркости у текущего пикселя, пикселя справа, снизу и справа снизу относительно текущего. Значения яркости получаем с помощью функции *get\_bright*, которая имеет тип *\_\_device\_\_.* Для вычисления градиента потребуется получить разности между значениями яркостей пикселей, находящимися по диагонали окошка (размер окошка: 2х2). Разность между яркостями вычислим с помощью функции *minus* (тип *\_\_device\_\_*). Финальное значение градиента вычисляется в функции *calc\_grad* (тип *\_\_device\_\_*). На выходе задаём пикселю значения всех цветовых составляющих равными вычисленному градиенту. Затем копируем данные с графического процессора на центральный и получаем итоговое изображение.

**Результаты**

Продемонстрируем скорость работы программы в зависимости от количества потоков:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Количество блоков и потоков в блоке | Размер входных данных | n = 15 | n = 1000 | n = 1000.00 | n = 1000.00.00 |
| << 1, 4 >,  < 1, 4 >> | | 1.9950237745e-02 мс | 3.7682921904e-02 мс | 1.6912475104e+00 мс | 2.28581204482e+02 мс |
| << 2, 8 >,  < 2, 8 >> | | 5.0476849316e-02 мс | 5.1857312070-02 мс | 1.9457129751e-01 мс | 2.79412830412e+01 мс |
| << 8, 8 >,  < 16, 16 >> | | 5.4417659430e-02 мс | 4.1924781920-02 мс | 2.7192581806e-02 мс | 3.47124957021e+00 мс |
| << 8, 16 >,  < 8, 16 >> | | 5.1295762040e-02 мс | 2.1579108743e-02 мс | 2.5812947533e-02 мс | 2.38571750314e+00 мс |
| << 8, 8 >,  < 32, 32 >> | | 4.1209410248e-02 мс | 2.2823182057e-02 мс | 2.8839412107e-02 мс | 1.58317249122e-01 мс |
| << 16, 16 >,  < 32, 32 >> | | 3.9124712301e-02 мс | 2.7209479148e-02 мс | 3.0759102491e-02 мс | 1.48127592127e-01 мс |
| << 32, 32 >,  < 32, 32 >> | | 4.9712491231e-02 мс | 2.7190475249e-02 мс | 3.1132757129e-02 мс | 1.28754759145e-01 мс |

Сравним работу алгоритма на ГПУ и ЦПУ:

|  |  |  |  |
| --- | --- | --- | --- |
| Размер входных данных | Вид процессора | ГПУ  (<< 16, 16 >,  < 32, 32 >>) | ЦПУ |
| n = 15 | | 3.9124712301e-02 мс | 2.3000000000e-03 мс |
| n = 1000 | | 2.7209479148e-02 мс | 4.6000000000e-03 мс |
| n = 1000.00 | | 3.0759102491e-02 мс | 4.5900000000e-01 мс |
| n = 1000.00.00 | | 1.48127592127e-01 мс | 2.1528000000e+00 мс |

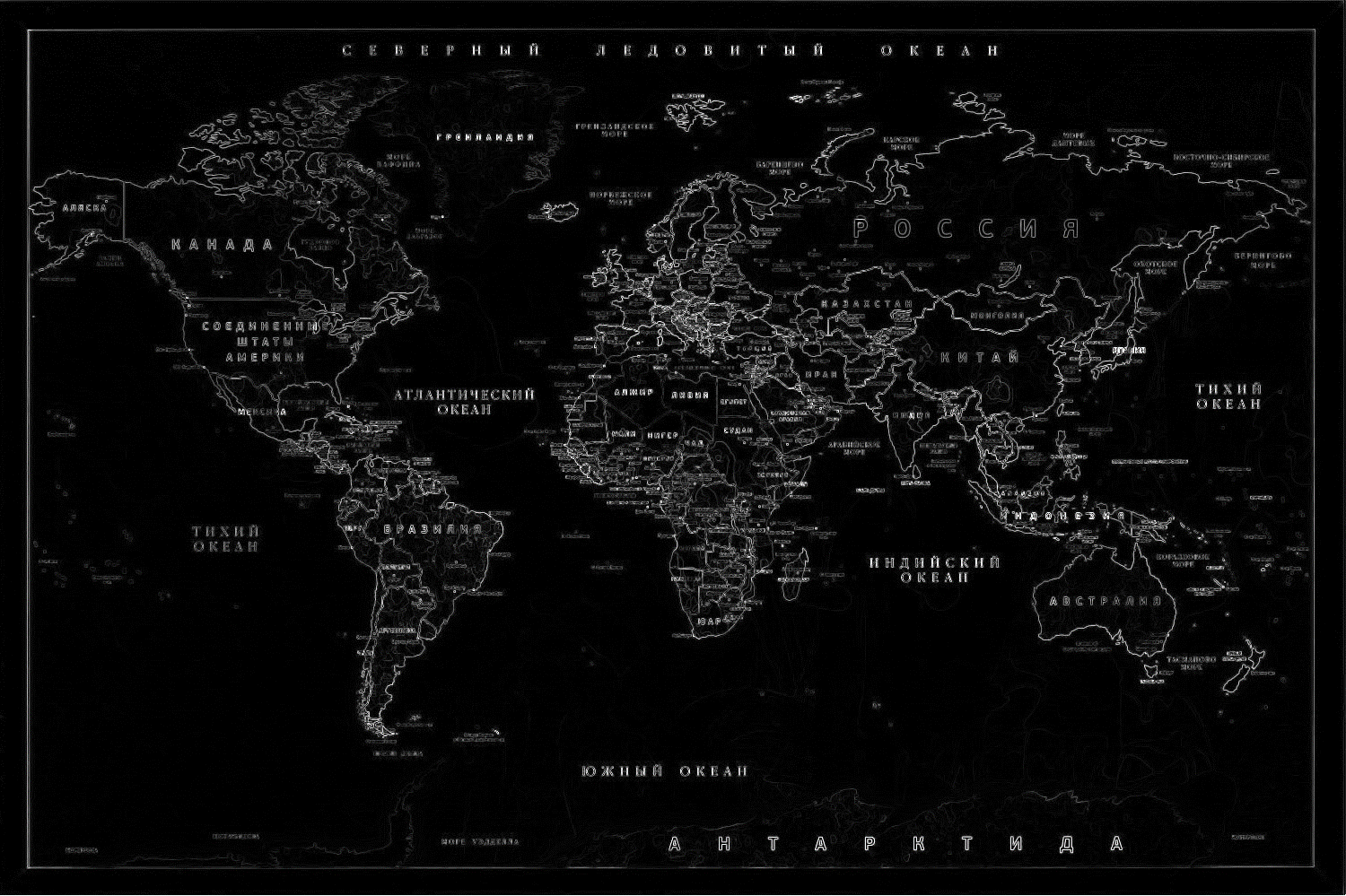
**Демонстрация результата**

Пример 1.

Входное изображение:



Выходное изображение:



Пример 2.

Входное изображение:



Выходное изображение:



**Выводы**

В данной лабораторной работе я узнал, по какому принципу работают фильтры, применяемые для обработки изображений. Делать лабораторную работу было очень интересно, особенно было увлекательно наблюдать за тем, какие получаются выходные изображения в процессе написания алгоритма. Также новым для меня в данной работе было использование двумерной сетки потоков и работа с графической памятью.