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.
Вариант 6. Выделение контуров. Метод Превитта.

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

Москва, 2021

Условие

Цель работы.

Научиться использовать GPU для обработки изображений.
Использование текстурной памяти.

Вариант 6. Выделение контуров. Метод Превитта.

Входные данные. На первой строке задается путь к исходному изображению, на второй, путь к конечному изображению. w*h ≤ 10^8

Программное и аппаратное обеспечение

Nvidia GeForce GTX 660

Compute capability: 3.0
Графическая память: 2048MB
Регистров на блок: 65536
Нитей на блок: 1024
Мультипроцессоров: 5
Всего ядер: 960

Intel(R) Core(TM) i5-3570 CPU @ 3.40GHz

Тактовая частота: 3.4 GHz
Кэш-память: 6 MB

Оперативная память

Объём: 8 GB

Жёсткий диск

Объём: 2 TB

Программное обеспечение

OS: Windows 10
IDE: Visual Studio 2019
CUDA: v10.2 nvcc

Метод решения

Переписываем ядро в примере с лекции в соответствии с алгоритмом:
Берём все окружающие текущий пиксель пиксели, по ним вычисляем вектора Gx и Gy. По этим векторам считаем яркость в соответствии со спрятанной на лекции формулой. Наконец, находим корень разности квадратов двух яркостей, это и будет цвет текущего пикселя после обработки.

Описание программы

1. Макрос CSC

Проверяет, с каким статусом завершаются CUDA-операции, и в, случае ошибки, выводит на стандартный поток ошибок debug-информацию.

2. Ядро.
   Основная функция, работающая на устройстве. Принимает указатель на выходной вектор и размерности картинки. В цикле достаёт из текстурной памяти по текстурной ссылке требуемые для вычисления пиксели, совершает требуемые вычисления.

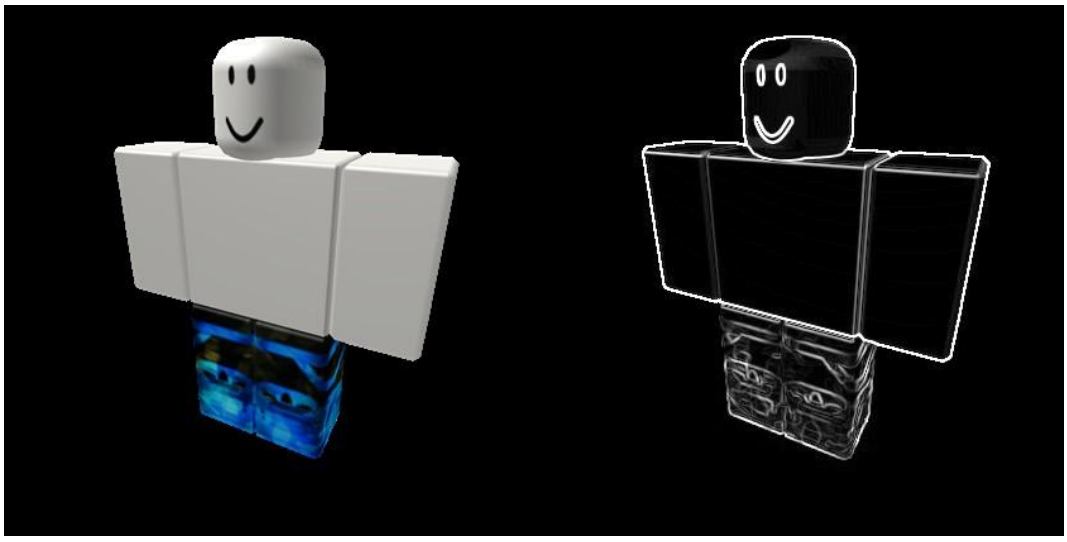
3. Main
   Считывает данные из файла, закидывает их на GPU, подготавливает текстурную ссылку, связывает её с данными, выгружает обработанный массив в файл.

Результаты

Размер теста	<<<1, 32>>>	<<<32, 32>>>	dim3(32,32),dim3(32,32)	CPU
128 на 128	89.78	24.63	3.57	0.0084
420 на 420	914.2	78.74	6.8	0.086
1024 на 1024	5018.73	184.79	23.6	0.492

Демонстрация работы

Применение фильтра к картинке 420 на 420 пикселей



Выводы

Проделав работу, я изучил принципы устройства текстурной памяти GPU, осознал, почему это быстро, удобно и полезно. Реализованный мною алгоритм применяется для обработки изображений. Его можно использовать, например, для выделения дорог на фотографиях (они будут выглядеть как длинные непрерывные линии). Реализация алгоритма не слишком трудная, в основном благодаря тому, что основа программного кода была выдана нам в качестве примера на лекции. Благодаря этому можно было сосредоточиться на самом алгоритме. Самой большой трудностью оказалось вспомнить, что за две недели до начала выполнения мною задания на лекции было сказано, что для определения яркости пикселя следует использовать определённую формулу. Из сравнения скорости работы алгоритма видно, что затрачиваемое время растёт в меньшей степени при большем числе блоков и нитей.