КОМИТЕТ ПО ОБРАЗОВАНИЮ ПРАВИТЕЛЬСТВА САНКТ-ПЕТЕРБУРГА

Санкт-Петербургское государственное
бюджетное профессиональное образование учреждение
«Колледж информационных технологий»

ОТЧЕТ
по практической работе
«Разработка многопоточного приложения»
по МДК01.03.

Работу выполнил студент 493 гр.:
Кашицын Артем Андреевич
Преподаватель:
Фомин Александр Валерьевич

Санкт-Петербург 2022

ОГЛАВЛЕНИЕ

ОПИСАНИЕ ИНТЕРФЕЙСА

Макет основного экрана приложения отображен на рисунке 1.

Рисунок 1 – Макет основного экрана

На основном экране расположены элементы под следующими номерами:

1 – SurfaceView

2 – Button

3 – TextView

4 – Slider

5 – Spinner

6 – RadioButton

SurfaceView отображает исходное изображение и поддерживает рисование в отдельном потоке, подходит для ресурсоемких операций и быстрых обновлений.

Button – это кнопка, по нажатию которой выполняется определенная функция. Функция «Запуск» предназначена для размытия изображения с выбранным типом размытия, размера матрицы, количества обрабатывающих потоков. Функция «Стоп» останавливает обрабатывающие потоки и выводят частичное изображение.

TextView используется как подпись для определения пользователем поля ввода данных, также отображает затраченное время на обработку.

Slider представляет собой горизонтальную полосу прокрутки, двигая указатель, пользователь выбирает размер матрицы от 3x3 до 64x64 с шагом 1.

Spinner – раскрывающийся список, в котором пользователь выбирает количество обрабатывающих изображение потоков от 1 до 8.

RadioButton – кнопка, которая принимает два значения (нажатое и не нажатое), при этом из нескольких RadioButton можно активировать только один переключатель. Служат для выбора типа размытия.

ДЕМОНСТРАЦИЯ ФУНКЦИЙ

Иконка приложения отображена на рисунке 2.

Рисунок 2 – Иконка приложения

После запуска приложения, пользователь видит исходное изображение. Затем настраивает размытие: размер матрицы, количество потоков для обработки, тип обработки.

После запуска обработки создается Bitmap на основе исходного изображения. В класс GlobalImage (который представляет собой данные для вывода обработанного изображения) помещается созданный Bitmap, ширина и высота изображения. Создается массив потоков, размер которого указывает пользователь.

Есть два класса BoxBlurHelper и Gaussian BlurHelper, которые наследуют класс Runnable (в который помещаются инструкции для отдельного потока). В экземпляры этих классов помещаются начальные и конечные координаты изображения для обработки, размер матрицы свертки. Алгоритмы заключаются в переборе пикселей от начальных до конечных координат, обработка матрицы пикселей определенного размера вокруг текущего пикселя, а затем сумма полученных обработанных элементов матрицы является делителем для значения цвета текущего пикселя, который помещается в итогове изображение. Для BoxBlur обработкой является деление на сумму матрицы, для Gaussian Blur – определенный коэффициент, вычисляемый по формуле.

Для каждого элемента массива потоков определяется свой выбранный экземпляр класса обработки. Соответственно, изображение делится на горизонтальные полосы, обработкой которых занимаются отдельные потоки.

От потоков программа ожидает выполнение обработки, если остается необработанный кусок изображения, который дорабатывает освободившийся поток.

Перед самым запуском всех потоков засекается время, и останавливается перед выводом изображения на пользовательскую форму.

Вид приложение после запуска на рисунке 3.

Рисунок 3 – Начальный вид приложение

Выберем матрицу 3x3, 1 поток и тип размытия Box Blur. Спустя 0.43 секунд изображение чуть-чуть размылось на рисунке 4.

Рисунок 4 – Обработка одним потоком с матрицей 3x3

Повысим коэффициент матрицы до размера 10x10. За 3.64 секунды получился результат на рисунке 5.

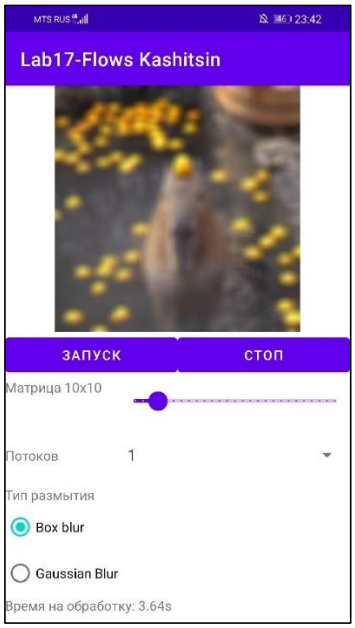

Рисунок 5 – Обработка одним потоком с матрицей 10x10

Теперь повысим количество потоков до 4, и получаем тот же результат быстрее на 0.17 секунды на рисунке 6.

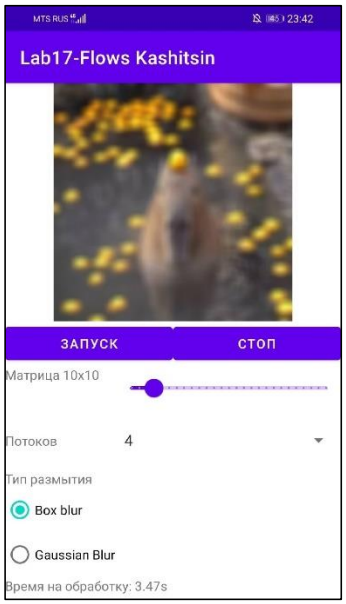

Рисунок 6 – Обработка четырьмя потоков с матрицей 10x10

Поменяем тип размытия на Gaussian Blur и увеличим количество потоков до 7. Результат показан на рисунке 7.

Рисунок 7 – Обработка Гауссова размытия семью потоками с матрицей 10x10

Средство Profiler в Android Studio позволяет проанализировать работу процессора, оперативной памяти и сети. Запустим обработку 32x32 матрицы с использованием 7 потоков, и в Profiler'е выбираем «CPU», чтобы посмотреть работу процессора. На рисунке 8 показан запуск и работа 7 потоков с названием thread.

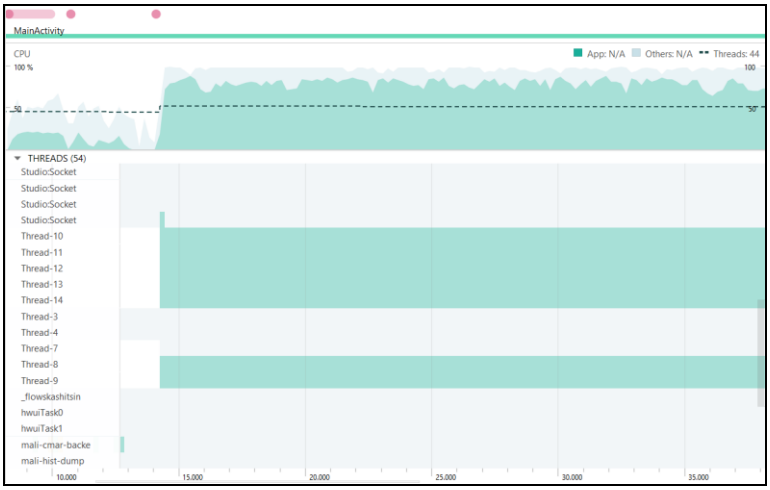

Рисунок 8 – Просмотр Profiler. Запуск потоков

Также в верху видим общую загруженность процессора, она заметна возросла после запуска 7 потоков. Всего активно 44 потоков – указано в правом углу. Ниже видим отметки времени.

Стоит отметить, что запуск потоков происходит в отдельно создаваемом потоке, иначе пользовательский интерфейс, за который отвечают другие (не

нагруженные вычислением созданного алгоритма) потоки, был бы не доступен, и пользователь не смог бы остановить работу потоков.

Остановка потоков происходит по кнопке «Стоп», посылая для всех потоков команду interrupt, который в создаваемом Runnable обрабатывается и завершает указанную потоку операцию. На рисунке 9 показана остановка потоков, а также затраченное время.

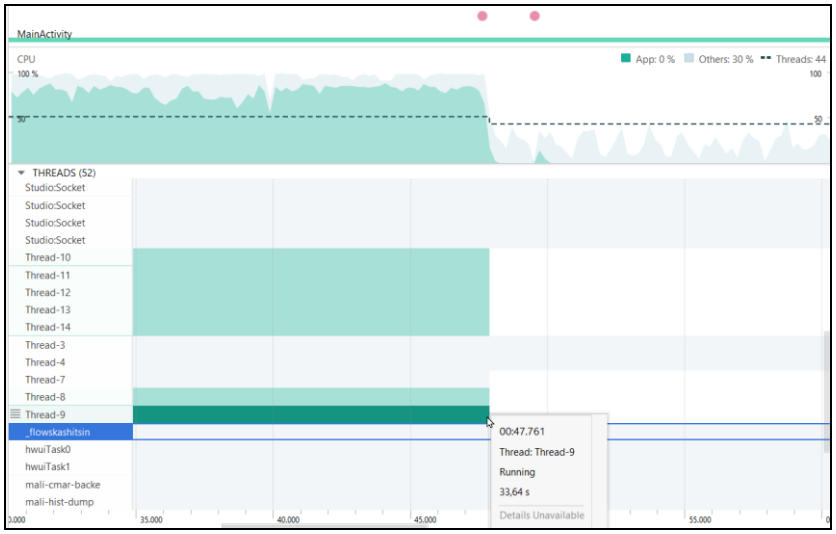

Рисунок 9 – Просмотр Profiler. Остановка потоков

Частично обработанное изображение показано на рисунке 10.

Рисунок 10 – Демонстрация результата остановки потоков.

ИТОГ РАБОТЫ

Ссылка на репозиторий с готовым проектом: https://github.com/aza1rat/LabFlowsBlurKashitsin