Комитет по образованию Правительства Санкт-Петербурга
САНКТ-ПЕТЕРБРУГСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по практической работе
МДК 01.02 «Разработка мобильных приложений»
Разработка многопоточного приложения

Выполнил
студент группы 493:
Лукьянов Ф-И. Ш.
Преподаватель: Фомин А.В.

Санкт-Петербург 2022

Интерфейс приложения

Приложение состоит из одной формы: на ней находятся все элементы интерфейса для задания параметров размытия, запуска и остановки потоков.

Основная форма

На рисунке 1 показан макет внешнего вида основной формы.

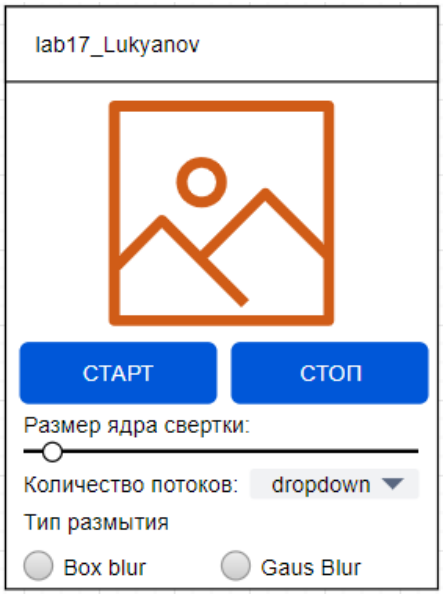

Рисунок 1 – Макет основной формы

На рисунке 2 показан внешний вид основной формы в приложении.

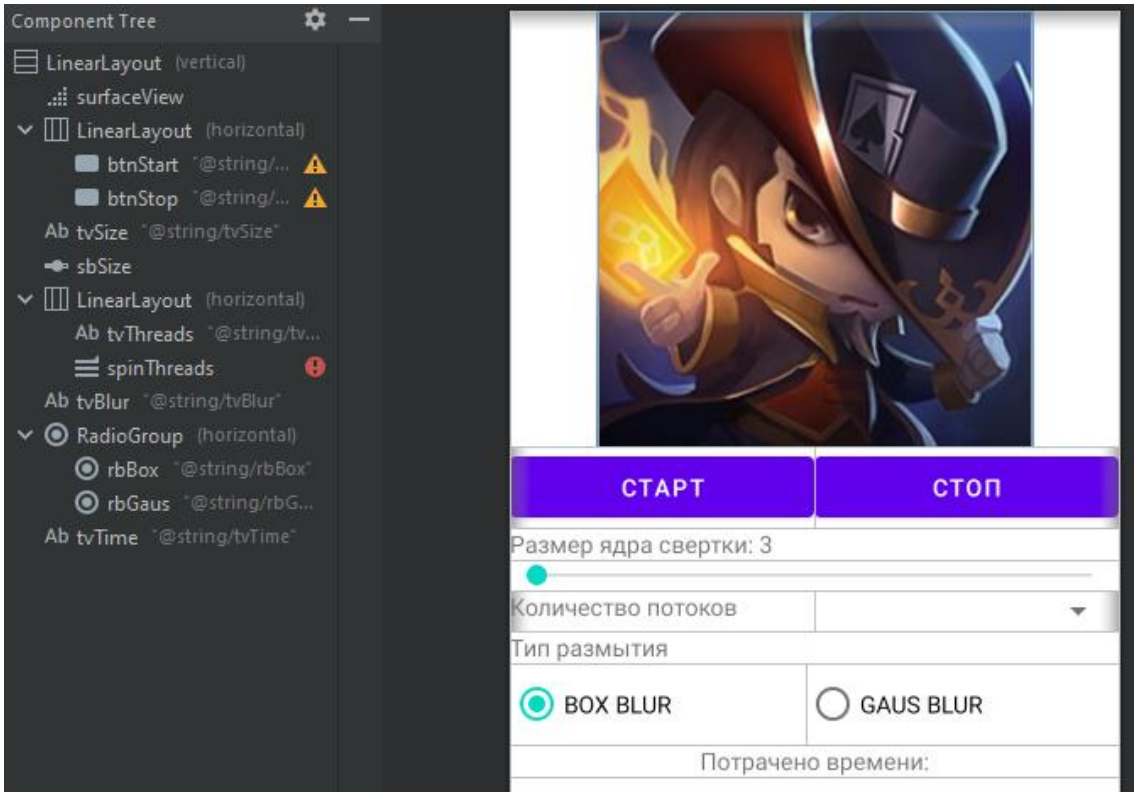

Рисунок 2 – Основная форма в приложении

Имена для компонентов в приложении брались из файла строковых ресурсов strings.xml (рис. 3).

```
<resources>
    <string name="app_name">lab17_Standart_Lukyanov</string>
    <string name="btnStart">СТАРТ</string>
    <string name="btnStop">СТОП</string>
    <string name="tvSize">Размер ядра свертки: 3</string>
    <string name="tvThreads">Количество потоков</string>
    <string name="tvBlur">Тип размытия</string>
    <string name="tvTime">Потрачено времени:</string>
    <string name="rbBox">BOX BLUR</string>
    <string name="rbGaus">GAUS BLUR</string>
</resources>
```

Рисунок 3 – Файл строковых ресурсов strings.xml

На рисунке 4 показана иконка приложения.

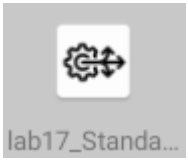

Рисунок 4 – Иконка приложения

На рисунке 5 показана картинка разрешением 256x256 пикселей в ресурсах приложения.

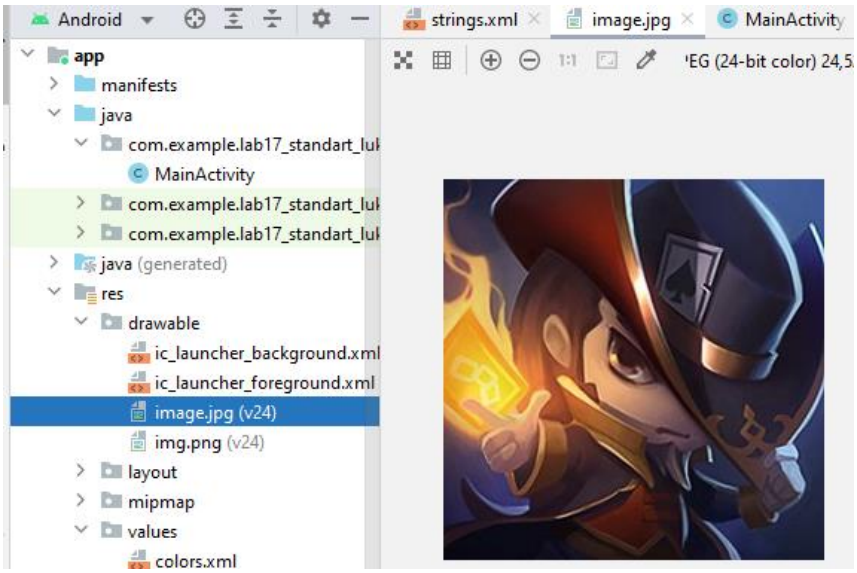

Рисунок 5 – Картинка в ресурсах

Демонстрация работы приложения

При запуске приложения отображается картинка из ресурсов и заполняется список для выбора количества потоков:

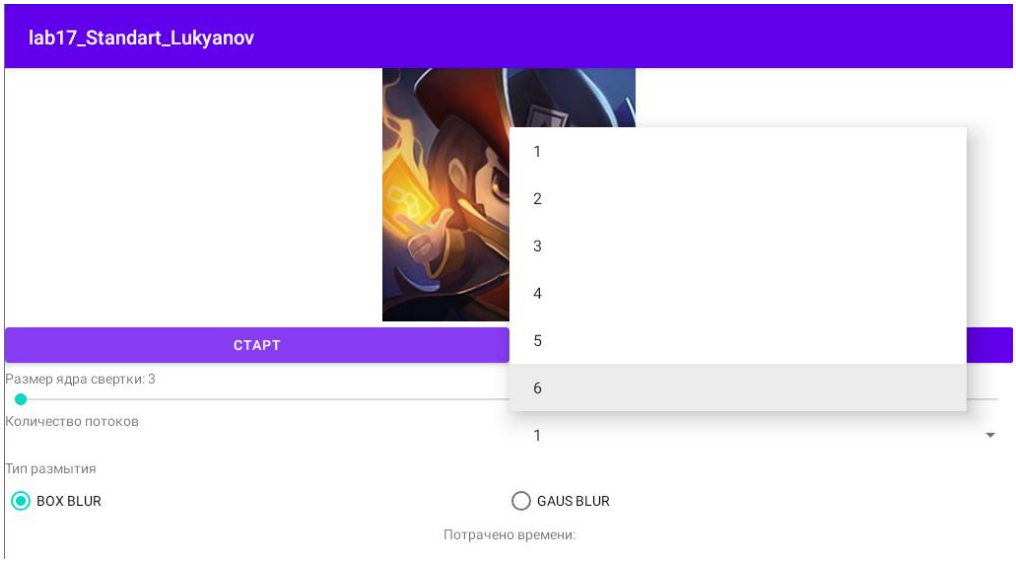

Рисунок 6 – Запуск приложения

При перемещении по слайдеру изменяется значение размера ядра свертки:

Рисунок 7 – Изменение значения размера ядра свертки

Изменяется количество задействованных потоков в соответствии с выбором из списка:

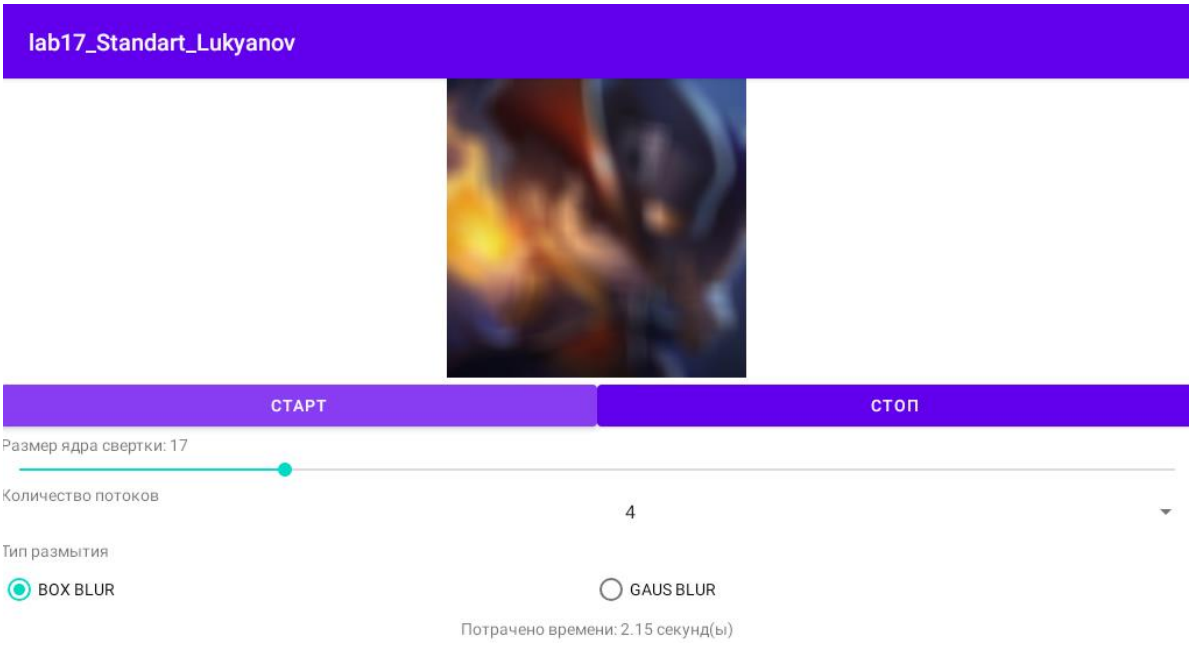

Рисунок 8 – Выбор количества потоков в приложении

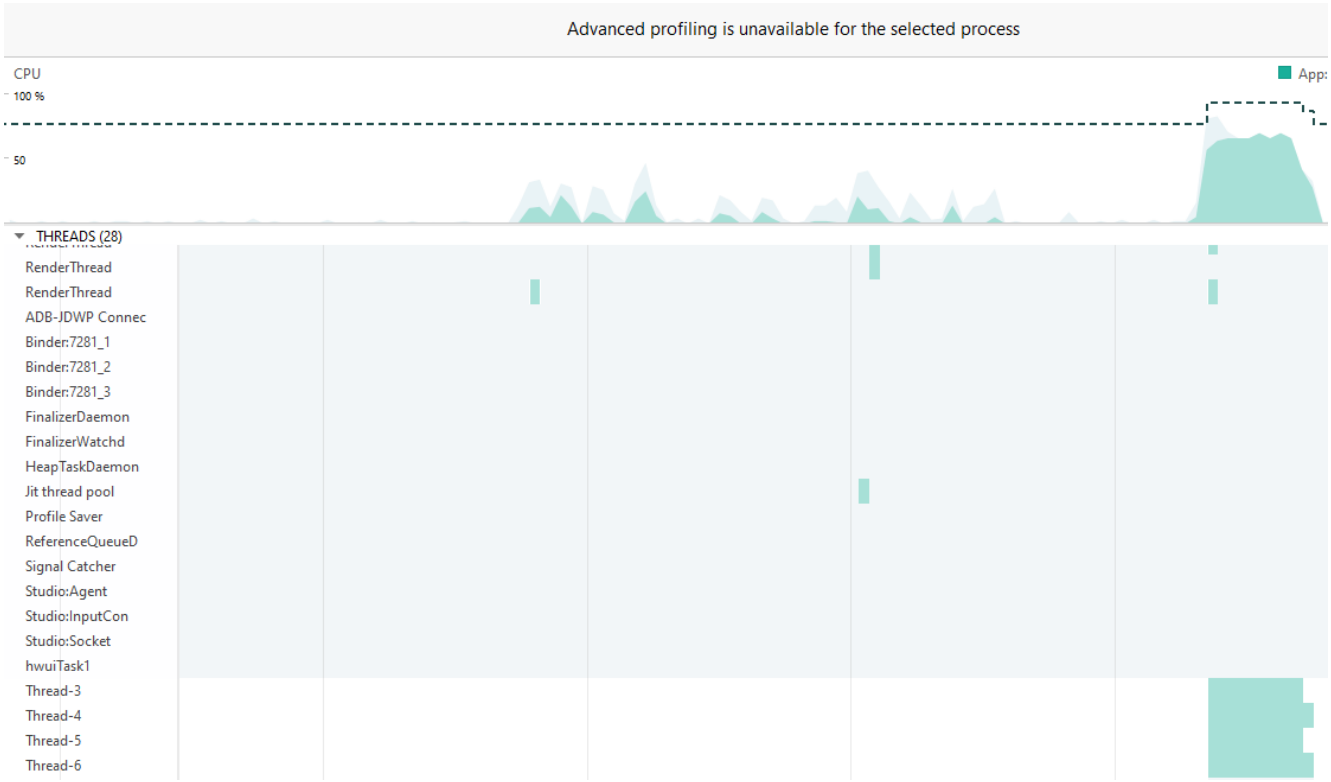

Рисунок 9 – Потоки в profiler

Также отображается затраченное на обработку время (рис. 8).

Если нажать «СТОП» во время обработки, то потоки остановится, и мы получим не полностью обработанное изображение:

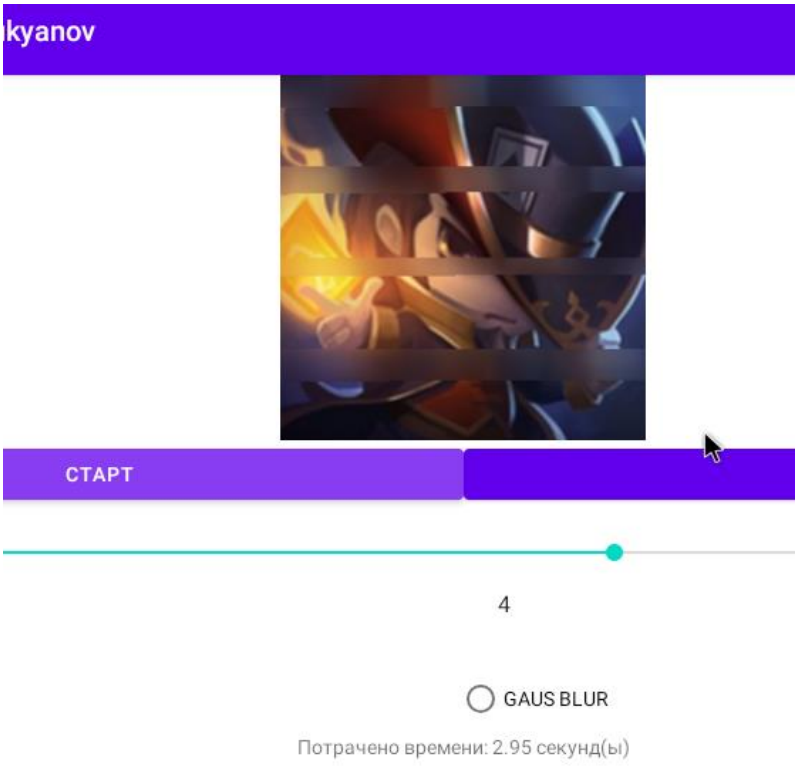

Рисунок 10 – Не полностью обработанное изображения

Также предусмотрен выбор типа размытия: «Box» (рис. 11) или «Gaussian» (рис.12).

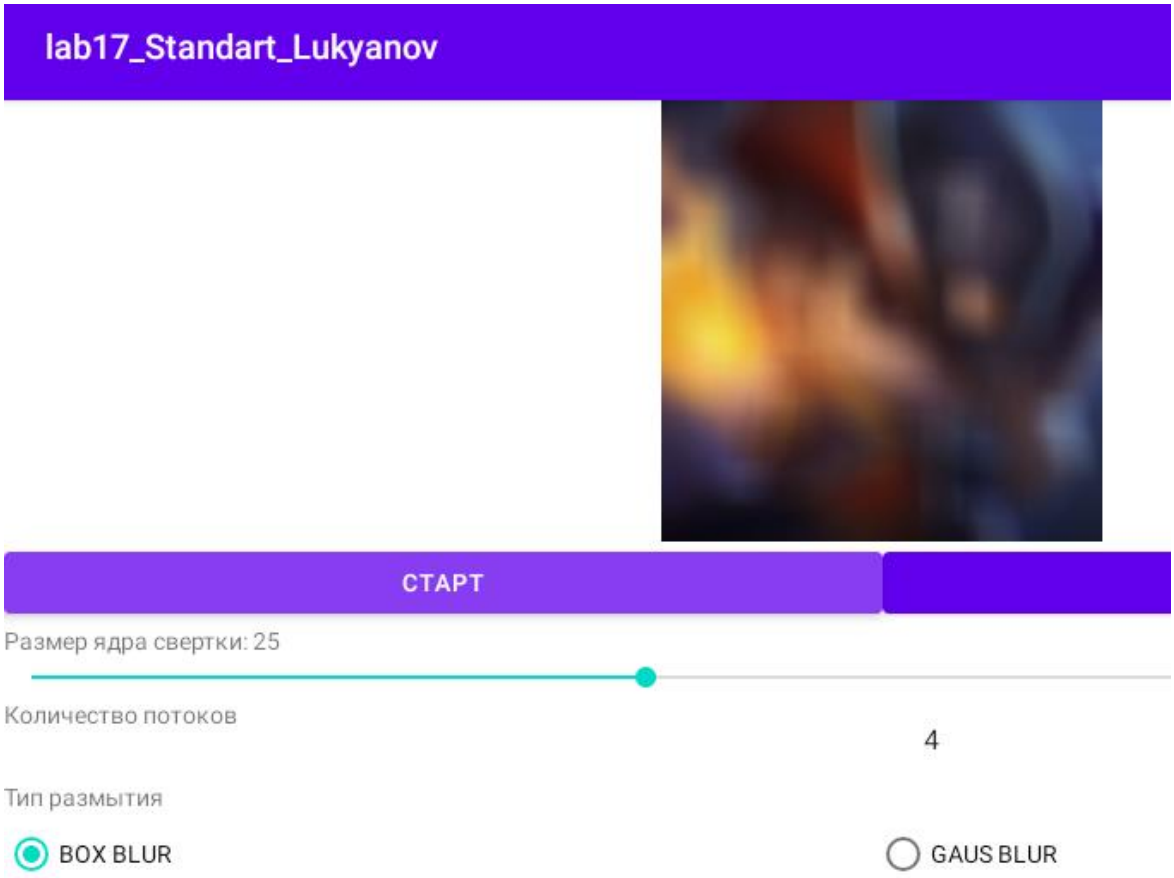

Рисунок 11 – Размытие «Box»

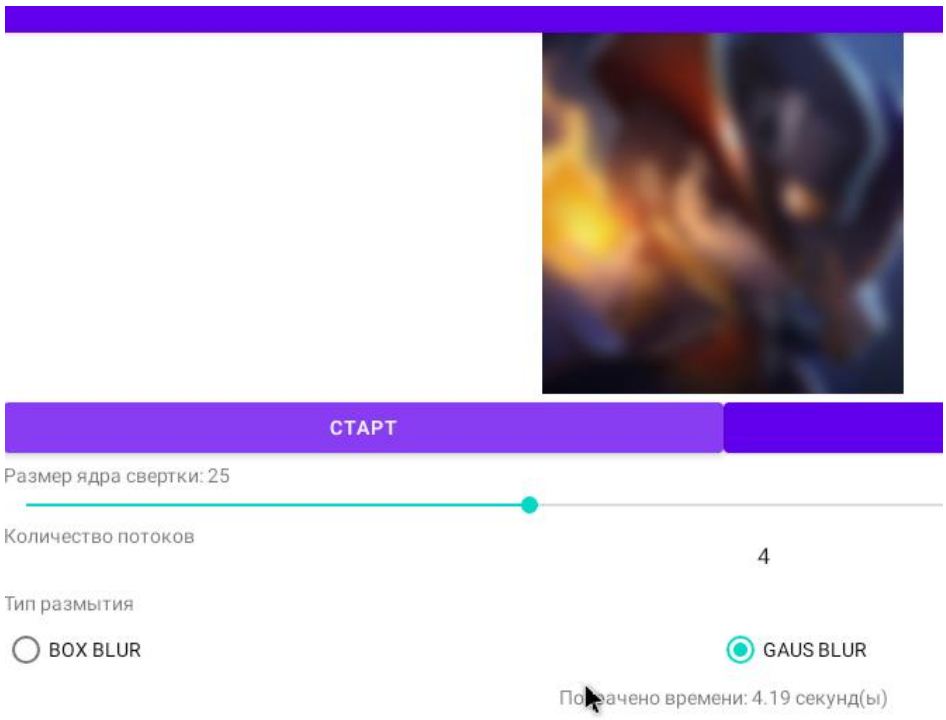

Рисунок 12 – Размытие «Gaussian»

Также можно задать размер ядра свертки: от 3 (рис. 13) до 64 (рис. 14).

Рисунок 13 – Размер ядра свертки 3

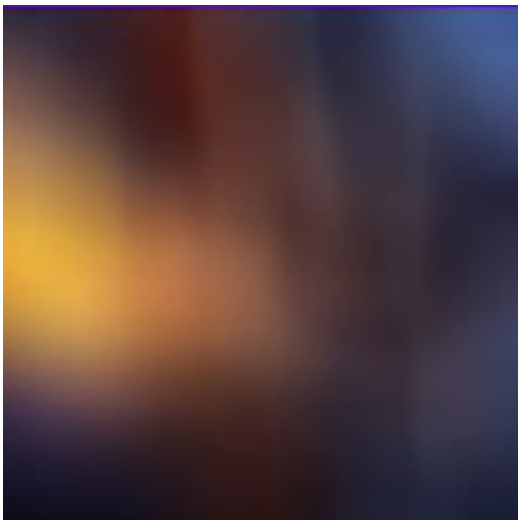

Рисунок 14 – Размер ядра свертки 64

Код приложения

Класс основной формы «MainActivity»:

```
package com.example.lab17_standart_lukyanov;

import androidx.annotation.RequiresApi;
import androidx.appcompat.app.AppCompatActivity;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.graphics.drawable.BitmapDrawable;
import android.graphics.drawable.Drawable;
import android.os.Build;
import android.os.Bundle;
import android.util.Log;
import android.view.SurfaceView;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.RadioButton;
import android.widget.SeekBar;
import android.widget.Spinner;
import android.widget.TextView;

import java.sql.Date;
import java.sql.Time;
import java.time.Duration;
import java.time.Instant;
import java.time.LocalDateTime;
import java.util.Calendar;
import java.util.stream.DoubleStream;

public class MainActivity extends AppCompatActivity {

    SurfaceView surfaceView;
    TextView tvSize;
    TextView tvTime;
    Spinner spinThreads;
    SeekBar sbSize;
    RadioButton rbBox;
    RadioButton rbGaus;

    ArrayAdapter<Integer> adp;

    Integer coreSize = 3, threadCount, w, h, c, px, py, red, green, blue;
    Boolean blur = false;

    Bitmap bmp;
    Bitmap res;

    Drawable draw;

    double[][] matrix;
    double sumMatrix;

    //final double sigma = 0.84089642;
    final double sigma = 5.5;

    Thread[] t;
```

```
    Instant start;
    Instant end;
    Duration timeElapsed;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        bmp = BitmapFactory.decodeResource(getResources(), R.drawable.image);
        w = bmp.getWidth();
        h = bmp.getHeight();
        res = Bitmap.createBitmap(w, h, Bitmap.Config.ARGB_8888);

        surfaceView = findViewById(R.id.surfaceView);

        tvSize = findViewById(R.id.tvSize);
        tvTime = findViewById(R.id.tvTime);
        spinThreads = findViewById(R.id.spinThreads);
        sbSize = findViewById(R.id.sbSize);
        rbBox = findViewById(R.id.rbBox);
        rbGaus = findViewById(R.id.rbGaus);

        adp = new ArrayAdapter<Integer>(this,
android.R.layout.simple_list_item_1);
        adp.add(1);
        adp.add(2);
        adp.add(3);
        adp.add(4);
        adp.add(5);
        adp.add(6);
        spinThreads.setAdapter(adp);

        sbSize.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
                coreSize = sbSize.getProgress() + 3;
                tvSize.setText("Размер ядра свертки: " + coreSize);
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });
    }

    public class Worker implements Runnable
    {
        public int y0;
        public int y1;

        public int w;
        public int h;

        public int coreSize;

        public Bitmap bmp;
        public Bitmap res;
```

```
public double[][] matrix;
public double sumMatrix;
double sumRed, sumGreen, sumBlue;

public boolean blur;

@Override
public void run()
{
    for (int y = y0; y < y1; y++)
    {
        for (int x = 0; x < w; x++)
        {
            int red = 0;
            int green = 0;
            int blue = 0;

            sumRed = 0;
            sumGreen = 0;
            sumBlue = 0;

            for (int v = 0; v < coreSize; v++)
            {
                for (int u = 0; u < coreSize; u++)
                {
                    int px = u + x - coreSize / 2;
                    int py = v + y - coreSize / 2;

                    if (px < 0) px = 0;
                    if (py < 0) py = 0;
                    if (px >= w) px = w - 1;
                    if (py >= h) py = h - 1;

                    if (px < 0)
                        Log.e("px value: " + px, "py value: " + py);
                    int c = bmp.getPixel(px, py);

                    sumRed += (double)Color.red(c) * matrix[u][v];
                    sumGreen += (double)Color.green(c) * matrix[u][v];
                    sumBlue += (double)Color.blue(c) * matrix[u][v];
                }
            }

            if (!blur)
            {
                sumRed/=sumMatrix;
                sumGreen/=sumMatrix;
                sumBlue/=sumMatrix;
            }

            red = (int)sumRed;
            green = (int)sumGreen;
            blue = (int)sumBlue;

            if (blur)
            {
                red /= coreSize * coreSize;
                green /= coreSize * coreSize;
                blue /= coreSize * coreSize;
            }
            res.setPixel(x, y, Color.rgb(red, green, blue));
            if (Thread.currentThread().isInterrupted()) return;
        }
```

```
        }
    }
}

@RequiresApi(api = Build.VERSION_CODES.O)
public void onStart(View view)
{
    start = Instant.now();
    threadCount = (Integer) spinThreads.getSelectedItem();
    if (rbBox.isChecked()) blur = true;
    else blur = false;

    matrix = fillMatrix(blur);
    sumMatrix = sumMatrix(matrix);

    t = new Thread[threadCount];
    Worker[] r = new Worker[threadCount];

    int s = h / threadCount;

    for (int i = 0; i < threadCount; i++)
    {
        r[i] = new Worker();
        r[i].bmp = bmp;
        r[i].res = res;
        r[i].w = w;
        r[i].h = h;
        r[i].coreSize = coreSize;
        r[i].matrix = matrix;
        r[i].blur = blur;
        r[i].sumMatrix = sumMatrix;
        r[i].y0 = s * i;
        r[i].y1 = r[i].y0 + s;
        t[i] = new Thread(r[i]);
        t[i].start();
    }

    Runnable run = new Runnable() {
        @Override
        public void run() {
            for (int i = 0; i < threadCount; i++) {
                try {
                    t[i].join();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
            draw = new BitmapDrawable(getResources(), res);
            runOnUiThread(() ->
            {
                surfaceView.setForeground(draw);
                setTime();
            });
        }
    };
    Thread tr = new Thread(run);
    tr.start();
}

double sumMatrix(double[][] matrix)
{
    double sum = 0;
    for (int i = 0; i < matrix.length; i++)
    {
```

```
            for (int j = 0; j < matrix.length; j++)
            {
                sum += matrix[i][j];
            }
        }
        return sum;
    }

    double[][] fillMatrix(boolean box)
    {
        double[][] matrix = new double[coreSize][coreSize];
        double e;
        double g1;
        double x, y;
        int median = coreSize / 2;
        for (int i = 0; i < coreSize; i++)
        {
            for (int j = 0; j < coreSize; j++)
            {
                if (box) matrix[i][j] = 1;
                else
                {
                    g1 = 1.0D / (2 * Math.PI * (sigma*sigma));
                    x = Math.pow((i-median), 2);
                    y = Math.pow((j-median), 2);
                    e = Math.pow(Math.E, ((x + y) / (2 * (sigma*sigma))) / -1.0D);
                    matrix[i][j] = g1 * e;
                }
            }
        }
        return matrix;
    }

    @RequiresApi(api = Build.VERSION_CODES.O)
    public void onStop(View v)
    {
        for (int i = 0; i < threadCount; i++)
        {
            t[i].interrupt();
        }
        draw = new BitmapDrawable(getResources(), res);
        surfaceView.setForeground(draw);
        setTime();
    }

    @RequiresApi(api = Build.VERSION_CODES.O)
    void setTime()
    {
        end = Instant.now();
        timeElapsed = Duration.between(start, end);
        float sec;
        sec = (int)timeElapsed.toMillis()/1000.0f;
        tvTime.setText("Потрачено времени: " + String.format("%.2f", sec) + "
секунд(ы)");
    }
}
```