****

**МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (МАДИ)** Кафедра “Высшая математика”  
  
  
**Отчет по лабораторной работе №3**По дисциплине: Компьютерная графика

Оставить на изображении только пиксели *красного* цвета.

Выполнил:

Студент группы 3бПМ

Дубенская Д.В.Проверил:

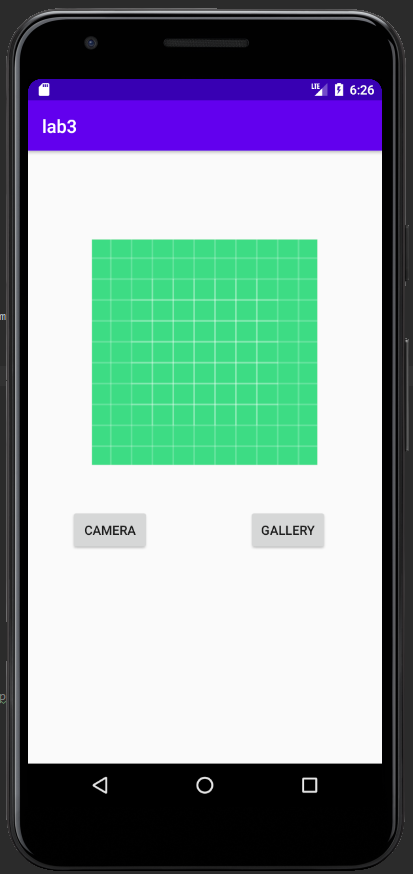
Мосева М.С.

г. Москва, 2020

**Описание.**

Работа с камерой, захват фотографии и её обработка, а именно удаление всех пикселей, которые не являются *красными*.

**Интерфейс приложения.**



**Пример работы приложения.**

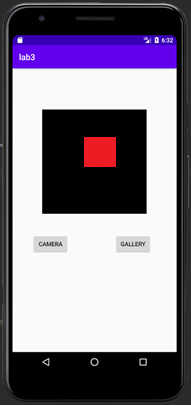
По нажатию на кнопку «Camera» открывается стандартный интерфейс приложения «Камера» на смартфоне.

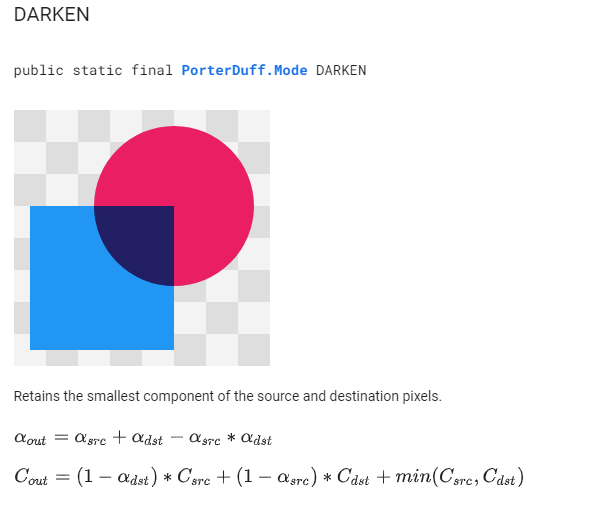
Далее происходит захват фотографии.



Далее нажимаем на центральную кнопку, чтобы обработать данное изображение, а именно - создать цветовую маску с помощью функции ctxColor и наложить маску на исходное изображение.

paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.*DARKEN*));





Как мы видим, фотография успешно обработана.

**Вывод.**

В данной лабораторной работе реализована работа с фотографиями, выделение маски фотографии по цвету и удаление всего с изображения, кроме заданного цвета.

**Код программы:**

activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <ImageView  
 android:id="@+id/displayImageView"  
 android:layout\_width="250dp"  
 android:layout\_height="250dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.229"  
 app:srcCompat="@drawable/ic\_launcher\_background" />  
  
 <Button  
 android:id="@+id/cameraBtn"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="48dp"  
 android:text="Camera"  
 app:layout\_constraintEnd\_toStartOf="@+id/galleryBtn"  
 app:layout\_constraintHorizontal\_bias="0.301"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/displayImageView" />  
  
 <Button  
 android:id="@+id/galleryBtn"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="48dp"  
 android:layout\_marginEnd="60dp"  
 android:text="Gallery"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/displayImageView" />  
</androidx.constraintlayout.widget.ConstraintLayout>

MainActivity.java

package com.example.lab3;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.app.ActivityCompat;  
import androidx.core.content.ContextCompat;  
  
import org.opencv.android.BaseLoaderCallback;  
import org.opencv.android.LoaderCallbackInterface;  
import org.opencv.android.OpenCVLoader;  
import org.opencv.android.Utils;  
import android.Manifest;  
import android.content.Intent;  
import android.content.pm.PackageManager;  
import android.graphics.Bitmap;  
import android.graphics.Canvas;  
import android.graphics.Color;  
import android.graphics.ImageDecoder;  
import android.graphics.Matrix;  
import android.graphics.Paint;  
import android.graphics.PorterDuff;  
import android.graphics.PorterDuffXfermode;  
import android.graphics.drawable.BitmapDrawable;  
import android.os.Bundle;  
import android.provider.MediaStore;  
import android.util.Log;  
import android.view.View;  
import android.widget.Button;  
import android.widget.ImageView;  
import android.widget.Toast;  
  
import org.opencv.core.Core;  
import org.opencv.core.CvType;  
import org.opencv.core.Mat;  
import org.opencv.core.Scalar;  
import org.opencv.imgproc.Imgproc;  
import org.opencv.core.\*;  
import org.opencv.imgcodecs.Imgcodecs;  
  
  
  
public class MainActivity extends AppCompatActivity {  
 ImageView selectedImage;  
 Button cameraBtn, galleryBtn;  
 static final int *REQUEST\_IMAGE\_CAPTURE* = 1;  
 Mat m;  
 private static final String *TAG* = "OCVSample::Activity";  
 static{ System.*loadLibrary*("opencv\_java3"); }  
  
 public MainActivity()  
 {  
 if (!OpenCVLoader.*initDebug*())  
 {  
 System.*out*.println("GG");  
 }  
  
 }  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 OpenCVLoader.*initDebug*();  
  
 selectedImage =findViewById(R.id.*displayImageView*);  
 cameraBtn =findViewById(R.id.*cameraBtn*);  
 galleryBtn =findViewById(R.id.*galleryBtn*);  
  
 cameraBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 askCameraPermissions();  
 }  
 });  
  
 galleryBtn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Toast.*makeText*(MainActivity.this, "Gallery bnt is clicked", Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 }  
  
 private void askCameraPermissions() {  
 if(ContextCompat.*checkSelfPermission*(this, Manifest.permission.*CAMERA*) != PackageManager.*PERMISSION\_GRANTED*){  
 ActivityCompat.*requestPermissions*(this, new String[] {Manifest.permission.*CAMERA*}, 101);  
 }else {  
 openCamera();  
 }  
 }  
  
  
 public void onRequestPermissionResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults){  
 if(requestCode == 101 ){  
 if(grantResults.length > 0 && grantResults[0] == PackageManager.*PERMISSION\_GRANTED*){  
 openCamera();  
 }else{  
 Toast.*makeText*(this,"Camera Permission is required", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 }  
  
 private void openCamera() {  
  
 Intent takePictureIntent = new Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*);  
 if (takePictureIntent.resolveActivity(getPackageManager()) != null) {  
 startActivityForResult(takePictureIntent, *REQUEST\_IMAGE\_CAPTURE*);  
 }  
 }  
 public static Bitmap overlayBitmapToCenter(Bitmap bitmap1, Bitmap bitmap2) {  
 int bitmap1Width = bitmap1.getWidth();  
 int bitmap1Height = bitmap1.getHeight();  
 int bitmap2Width = bitmap2.getWidth();  
 int bitmap2Height = bitmap2.getHeight();  
  
 float marginLeft = (float) (bitmap1Width \* 0.5 - bitmap2Width \* 0.5);  
 float marginTop = (float) (bitmap1Height \* 0.5 - bitmap2Height \* 0.5);  
  
  
 Bitmap overlayBitmap = Bitmap.*createBitmap*(bitmap1Width, bitmap1Height, bitmap1.getConfig());  
  
 Canvas canvas = new Canvas(overlayBitmap);  
 canvas.drawBitmap(bitmap1, new Matrix(), null);  
 canvas.drawBitmap(bitmap2, marginLeft, marginTop, null);  
  
 return overlayBitmap;  
 }  
 @Override  
 protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
 System.*out*.println("onActivityResult");  
 if (requestCode == *REQUEST\_IMAGE\_CAPTURE* && resultCode == *RESULT\_OK*) {  
 Bundle extras = data.getExtras();  
 Bitmap image = (Bitmap) extras.get("data");  
 int width = image.getWidth();  
 int height = image.getHeight();  
 Bitmap overlayBitmap = Bitmap.*createBitmap*(width, height, image.getConfig());  
  
 selectedImage.setImageBitmap(image);  
  
  
   
  
 Mat roi = new Mat();  
 Mat dst = new Mat();  
 Utils.*bitmapToMat*(image, roi);  
 Mat roiTmp = roi.clone();  
 Mat mask=roi.clone();  
  
 Mat mask1=roi.clone();  
 Mat mask2=roi.clone();  
  
 Imgproc.*cvtColor*(roi, roiTmp, Imgproc.*COLOR\_RGB2HSV*);  
  
 Scalar lower1 = new Scalar(0, 150, 100);  
 Scalar upper1 = new Scalar(20, 255, 255);  
 Scalar lower2 = new Scalar(140, 100, 100);  
 Scalar upper2 = new Scalar(179, 255,255);  
  
 Core.*inRange*(roiTmp,lower1,upper1,mask1);  
 Core.*inRange*(roiTmp,lower2,upper2,mask2);  
 Core.*addWeighted*( mask1, 1.0, mask2, 1.0, 0.0, dst);  
 Utils.*matToBitmap*(dst,overlayBitmap);  
  
 Bitmap result = Bitmap.*createBitmap*(overlayBitmap.getWidth(), overlayBitmap.getHeight(), Bitmap.Config.*ARGB\_8888*);  
 Canvas tempCanvas = new Canvas(result);  
 Paint paint = new Paint(Paint.*ANTI\_ALIAS\_FLAG*);  
 paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.*DARKEN*));  
 tempCanvas.drawBitmap(image, 0, 0, null);  
 tempCanvas.drawBitmap(overlayBitmap, 0, 0, paint);  
  
  
 selectedImage.setImageBitmap(result);  
 }  
 }  
 private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {  
 @Override  
 public void onManagerConnected(int status) {  
 switch (status) {  
 case LoaderCallbackInterface.*SUCCESS*:  
 {  
 Log.*i*(*TAG*, "OpenCV loaded successfully");  
 m=new Mat();  
 } break;  
 default:  
 {  
 super.onManagerConnected(status);  
 } break;  
 }  
 }  
 };  
  
}