HW3: *BluSwap my face!*

this assignment, you are going to get practice with activity lifecycle, camera and pictures, and face detection with MLkit. This assignment is an individual assignment, intended to be implemented in **Kotlin**.

Note: For this assignment, you will want to use an actual phone to properly test your App.

Description

This assignment should take the form of an Android App, with the following functionalities:

- **Option 1**: take 2 different pictures with the phone camera, and display them on the screen, or

- **Option 2:** select 2 different pictures from your phone gallery, and display then on the screen.

Note: you are not expected to implement both options. Feel free to choose one of the two offered options, and implement it.

- **Event 1:** an event (for example, **a button click**, but other intuitive and user-friendly events are acceptable to) should trigger the swapping the face area of the two pictures (*hint: an easy implementation is to exchange the pixel values of the face area*).

- **Event 2**: An event (for example, another button click) should trigger the **blurring** of the two taken/selected pictures. The expected result should be two new pictures where the two faces look unrecognizable, compared to the original picture. Upon rotation, the pictures present on the screen should be maintained (*hint: you have two options to persist the state here - ViewModel or savedInstanceState() and the save the picture uri*).

Note 1: It is acceptable that the swap/blur effect is lost upon rotation, but the original images from the camera or the gallery should be maintained. It is optional, but you could use a ViewModel to maintain the Bitmap on the screen with the current effects.

Note 2: having 2 different xml files, one for portrait, and one for landscape is optional.

- **Going back:** add a clear/back widget (button or other widget of your choice) to display the original pictures.

- **Further edits to the picture:** add the pictures in a **DrawView**, so that the user can also paint on top of the pictures.

Submission

Submit your assignment using Github and Canvas, under the Assignment HW3. The due date is shown in Canvas. You will want to push the following to your personal Github repo:

- Android Studio Project folder
- README.txt or README.md

The README file should contain your name and email address, along with the name of your app and a brief description of it. Specifically, we are interested in the description of the photo editing that your app is performing. Finally, please include any special instructions that the user might need to know in order to use your app properly (if there are any). For example:

Tamara Bonaci <tbonaci@uw.edu>
Sorting Hat - This app displays the sorting hat and a picture of the user. It returns the house where the user belongs.

Grading (0-100 points)

Your submission will be graded by building and running it and evaluating its functionality. Your code will not be graded on style, but we still encourage you to follow good overall coding style for your own good.

- 0-10 points: the app compiles, builds and loads without errors.
- 0-10 points: the app runs, and satisfies the minimum functionality (takes two photos, displays them, and has a button-based event)
- 0-20 points: functionalities to swap faces, and back/clear event to undo swap work correctly.
- 0-20 points: functionalities to blur faces and back/clear event to undo blur work correctly.
-
- 0-20 points: functionality to visualize pictures in a DrawView to paint on top of the views works correctly.
- 0-10 points: Images are retained on the screen upon phone rotation.
- 0-10 points: The User Interface is well organized, and well documented.

RESOURCES

Adding a DrawView to your app (example in Pictionary App on Canvas)

1. Add library in your build.grade(Module:App)

```
// AndroidDraw Library
implementation 'com.github.divyanshub024:AndroidDraw:v0.1'
Also, the  minSdkVersion  should be 16
```

2. Add in your build.gradle (Module:project)

```
allprojects {
    repositories {
        google()
        jcenter()
        maven { url 'https://jitpack.io' }
    }
```

3. Add the DrawView widget in your 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">

  <com.divyanshu.draw.widget.DrawView
      android:id="@+id/draw_view"
      android:layout_width="the size that you want"
      android:layout_height="the size that you want"
      android:background="can set a background color here "
      app:layout_constraintTop_toTopOf="parent" /> --- an example of a constrain
```

4. Refer to the drawView object from your activity

```
import com.divyanshu.draw.widget.DrawView

private var mydrawView: DrawView? = null

// Setup view instances IN YOUR onCreate()
mydrawView = findViewById(R.id.draw_view)
```

5. Add an event to detect a motion event (when the users ends drawing)

```
// Will run  after every stroke drew
mydrawView?.setOnTouchListener { _, event ->
  // As we have interrupted DrawView's touch event,
  // we first need to pass touch events through to the instance for the drawing
to show up
  mydrawView?.onTouchEvent(event)
  // Then if user finished a touch event, run an action
  if (event.action == MotionEvent.ACTION_UP) {

     //action goes here
  }

  true
}
```

6. Change the configuration and add a background Bitmap. For the assignment, this background bitmap should be the image taken with the camera, or selected from the gallery.

```
mydrawView?.setStrokeWidth(10.0f)
mydrawView?.setColor(Color.WHITE)

mydrawView?.background= BitmapDrawable(getResources(), imageBitmap);
```

Connect your App to Firebase and use the MLkit

For reference, refer to slides from Lecture 5, and the official guides (which are very helpful!) https://firebase.google.com/docs/android/setup/?gclid=Cj0KCQjw4dr0BRCxARIsAKUNjWTSjIIXib0utMK3CMrPcT2mKxj3klUsWm8dB7oWN8LdvNq-z051aa8aAho_EALw_wcB

To use the MLkit, you need to add the dependencies for the ML Kit Android libraries to your module (app-level) Gradle file (app/build.gradle)

```
apply plugin: 'com.google.gms.google-services'

dependencies {
  // ...
  implementation 'com.google.firebase:firebase-ml-vision:24.0.2'
  // If you want to detect face contours (landmark detection and classification
  // don't require this additional model):
  implementation 'com.google.firebase:firebase-ml-vision-face-model:20.0.0'}
```

Face detection concepts and Bitmaps

You do not need to detect the face shape with very high accuracy. It is good enough if you detect some landmarks and select a rectangle around the face area. For example, you can detect the eyes and mouth, and define a rectangle based on those landmarks. Remember that the landmark values x,y, correspond to particular pixel positions in your image (bitmap).

If you want to achieve a higher accuracy for the face detection, instead of using the landmarks detection, you can use the contour points.

https://firebase.google.com/docs/ml-kit/face-detection-concepts

https://firebase.google.com/docs/ml-kit/android/detect-faces

If you use the code provided in the example App "TakePicture" for the assignment, the image will be returned as a Bitmap. You can get and change the value of particular pixels with these functions:

```
setPixel(x: Int, y: Int, color: Int)
getPixel(x: Int, y: Int)
```

You can find details about all the functions of Bitmap here - **https://developer.android.com/reference/android/graphics/Bitmap#setPixel(int,%20int,%20int)**

Each pixel value (color) is an Integer that corresponds to the ARGB color to write into the bitmap.

You will also find useful function that creates a new bitmap based on a previous bitmap. For example, you can create a new bitmap which contains the pixels of the face area detected and use this to run the function of blur. Then, copy this blurred sub-image back to the original image.

Copy source Bitmap from a particular pixel location (xSource, ySource), the amount of pixels pixelsWidth in the x direction, and pixelsHeight in the y direction

```
newBitmap = Bitmap.createBitmap(sourceBitmap, xSource, ySource, pixelsWidth,
pixelsHeight)
```

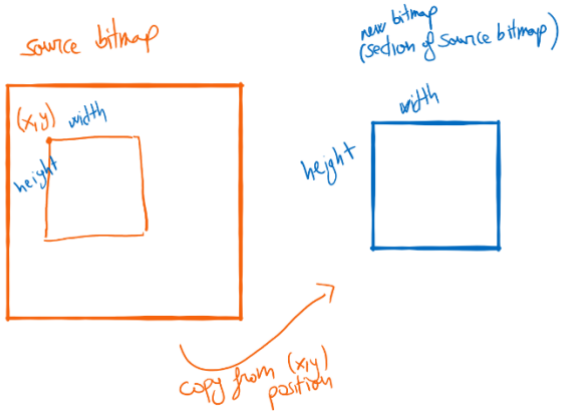

Figure 1: example of Bitmap.createBitmap

This function creates a scaled version of a bitmap (to reduce the bitmap size). By reducing the size of the bitmap, the face detector will run much faster.

```
val resizedBitmap = Bitmap.createScaledBitmap(
    imageBitmapSource,
```

```
    (imageBitmapSource.width / scaleFactor),
    (imageBitmapSource.height / scaleFactor),
    true)
```

Face blur

When we blur an image, we make the color transition from one side of an edge in the image to another smooth rather than sudden. The effect is to average out rapid changes in pixel intensity. The blur, or smoothing, of an image removes "outlier" pixels that may be noise in the image. Blurring is an example of applying a *low-pass filter* to an image. In computer vision, the term "low-pass filter" applies to removing noise from an image while leaving the majority of the image intact. A blur is a very common operation we need to perform before other tasks such as edge detection.

To blur the face, you can use the following Kotlin ode, or alternatively, a different algorithm or method of your choice. This function will blur the entire input bitmap. You could also modify it to just blur a section of the image.

```
fun bitmapBlur(sentBitmap: Bitmap, scale: Float, radius: Int): Bitmap? {
        var sentBitmap = sentBitmap

        val width = Math.round(sentBitmap.width * scale)
        val height = Math.round(sentBitmap.height * scale)
        sentBitmap = Bitmap.createScaledBitmap(sentBitmap, width, height, false)

        val bitmap = sentBitmap.copy(sentBitmap.config, true)

        if (radius < 1) {
            return null
        }

        val w = bitmap.width
        val h = bitmap.height

        val pix = IntArray(w * h)
        Log.e("pix", w.toString() + " " + h + " " + pix.size)
        bitmap.getPixels(pix, 0, w, 0, 0, w, h)

        val wm = w - 1
        val hm = h - 1
        val wh = w * h
        val div = radius + radius + 1

        val r = IntArray(wh)
        val g = IntArray(wh)
        val b = IntArray(wh)
        var rsum: Int
        var gsum: Int
        var bsum: Int
        var x: Int
        var y: Int
        var i: Int
        var p: Int
        var yp: Int
        var yi: Int
        var yw: Int
        val vmin = IntArray(Math.max(w, h))

        var divsum = div + 1 shr 1
```

```
        divsum *= divsum
        val dv = IntArray(256 * divsum)
        i = 0
        while (i < 256 * divsum) {
            dv[i] = i / divsum
            i++
        }

        yi = 0
        yw = yi

        val stack = Array(div) { IntArray(3) }
        var stackpointer: Int
        var stackstart: Int
        var sir: IntArray
        var rbs: Int
        val r1 = radius + 1
        var routsum: Int
        var goutsum: Int
        var boutsum: Int
        var rinsum: Int
        var ginsum: Int
        var binsum: Int

        y = 0
        while (y < h) {
            bsum = 0
            gsum = bsum
            rsum = gsum
            boutsum = rsum
            goutsum = boutsum
            routsum = goutsum
            binsum = routsum
            ginsum = binsum
            rinsum = ginsum
            i = -radius
            while (i <= radius) {
                p = pix[yi + Math.min(wm, Math.max(i, 0))]
                sir = stack[i + radius]
                sir[0] = p and 0xff0000 shr 16
                sir[1] = p and 0x00ff00 shr 8
                sir[2] = p and 0x0000ff
                rbs = r1 - Math.abs(i)
                rsum += sir[0] * rbs
                gsum += sir[1] * rbs
                bsum += sir[2] * rbs
                if (i > 0) {
                    rinsum += sir[0]
                    ginsum += sir[1]
                    binsum += sir[2]
                } else {
                    routsum += sir[0]
                    goutsum += sir[1]
                    boutsum += sir[2]
                }
                i++
            }
            stackpointer = radius

            x = 0
            while (x < w) {

                r[yi] = dv[rsum]
                g[yi] = dv[gsum]
                b[yi] = dv[bsum]

                rsum -= routsum
                gsum -= goutsum
                bsum -= boutsum
```

```
            stackstart = stackpointer - radius + div
            sir = stack[stackstart % div]

            routsum -= sir[0]
            goutsum -= sir[1]
            boutsum -= sir[2]

            if (y == 0) {
                vmin[x] = Math.min(x + radius + 1, wm)
            }
            p = pix[yw + vmin[x]]

            sir[0] = p and 0xff0000 shr 16
            sir[1] = p and 0x00ff00 shr 8
            sir[2] = p and 0x0000ff

            rinsum += sir[0]
            ginsum += sir[1]
            binsum += sir[2]

            rsum += rinsum
            gsum += ginsum
            bsum += binsum

            stackpointer = (stackpointer + 1) % div
            sir = stack[stackpointer % div]

            routsum += sir[0]
            goutsum += sir[1]
            boutsum += sir[2]

            rinsum -= sir[0]
            ginsum -= sir[1]
            binsum -= sir[2]

            yi++
            x++
        }
        yw += w
        y++
    }
    x = 0
    while (x < w) {
        bsum = 0
        gsum = bsum
        rsum = gsum
        boutsum = rsum
        goutsum = boutsum
        routsum = goutsum
        binsum = routsum
        ginsum = binsum
        rinsum = ginsum
        yp = -radius * w
        i = -radius
        while (i <= radius) {
            yi = Math.max(0, yp) + x

            sir = stack[i + radius]

            sir[0] = r[yi]
            sir[1] = g[yi]
            sir[2] = b[yi]

            rbs = r1 - Math.abs(i)

            rsum += r[yi] * rbs
            gsum += g[yi] * rbs
            bsum += b[yi] * rbs
```

```
            if (i > 0) {
                rinsum += sir[0]
                ginsum += sir[1]
                binsum += sir[2]
            } else {
                routsum += sir[0]
                goutsum += sir[1]
                boutsum += sir[2]
            }

            if (i < hm) {
                yp += w
            }
            i++
        }
        yi = x
        stackpointer = radius
        y = 0
        while (y < h) {
            // Preserve alpha channel: ( 0xff000000 & pix[yi] )
            pix[yi] = -0x1000000 and pix[yi] or (dv[rsum] shl 16) or (dv[gsum] shl 8) or dv[bsum]

            rsum -= routsum
            gsum -= goutsum
            bsum -= boutsum

            stackstart = stackpointer - radius + div
            sir = stack[stackstart % div]

            routsum -= sir[0]
            goutsum -= sir[1]
            boutsum -= sir[2]

            if (x == 0) {
                vmin[y] = Math.min(y + r1, hm) * w
            }
            p = x + vmin[y]

            sir[0] = r[p]
            sir[1] = g[p]
            sir[2] = b[p]

            rinsum += sir[0]
            ginsum += sir[1]
            binsum += sir[2]

            rsum += rinsum
            gsum += ginsum
            bsum += binsum

            stackpointer = (stackpointer + 1) % div
            sir = stack[stackpointer]

            routsum += sir[0]
            goutsum += sir[1]
            boutsum += sir[2]

            rinsum -= sir[0]
            ginsum -= sir[1]
            binsum -= sir[2]

            yi += w
            y++
        }
        x++
    }

    Log.e("pix", w.toString() + " " + h + " " + pix.size)
```

```
        bitmap.setPixels(pix, 0, w, 0, 0, w, h)

        return bitmap
    }
```