MINISTERUL EDUCAŢIEI REPUBLICII MOLDOVA

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Facultatea „Calculatoare, Informatică şi Microelectronică”

FILIERA ANGLOFONĂ

RAPORT

Lucrare de laborator nr. 4

la Programarea Aplicațiilor Mobile

A efectuat:

st. gr. FAF-151 Bîzdîga Stanislav

A verificat:

asist.univ. Sergiu Ciudin

Chișinău-2017

Laboratory Work 4

Subject: Custom Progress/Loading Bar

Objectives:

Develop an app on Android using Android Studio.

Creating a graphical control view of progress with its implementation in practice.

A GIF or Custom animation that will be implemented in the code will be chosen by the developer.

Tasks:

The raster graphics will be converted into vector elements.

The graphical control will be over the current activity that called it.

The control has implemented at least 3 events (START, PROGRESS [%], STOP)

Testing and depicting implementation will be performed within an ASYNC TASK with a maximum execution delay of 5-10s.

Introduction

This lab work is of advanced skill and importance since it is the first ever experience with animations in mobile application development.
It is supposed to form an understanding of concepts like frames, animation, its properties and more.

The amount of tasks seems on point, however each of them has their own quirks that might slow down the process of developing through the fact that there's no initial knowledge to begin with, yet their completion will be the way to go about learning the requirements.

Therefore, anything that's to be on the list of learning during this lab, will be of help to sharpen the skill of mobile development and the mastery of animation.

Short Theory

Animation in android is possible in many ways.

Tween Animation takes some parameters such as start value, end value, size, time duration, rotation angle e.t.c and perform the required animation on that object. It can be applied to any type of object. So in order to use this , android has provided us a class called Animation. [1]

In order to perform animation in android , we are going to call a static function loadAnimation() of the class AnimationUtils. We are going to receive the result in an instance of Animation Object. [1]

Its syntax is as follows:

```
Animation animation = AnimationUtils.loadAnimation(getApplicationContext(),

   R.anim.myanimation);
```

Note the second parameter. It is the name of the our animation xml file. You have to create a new folder called **anim** under res directory and make an xml file under anim folder. [1]

This animation class has many useful functions which are listed below:

Sr.No	Method & Description
1	**start()** This method starts the animation.
2	**setDuration(long duration)** This method sets the duration of an animation.
3	**getDuration()** This method gets the duration which is set by above method
4	**end()** This method ends the animation.
5	**cancel()** This method cancels the animation.

In order to apply this animation to an object , we will just call the startAnimation() method of the object.

Its syntax is:

```
ImageView image1 = (ImageView)findViewById(R.id.imageView1);

image.startAnimation(animation);
```

Task Implementation and result verification

Testing the progress bar:

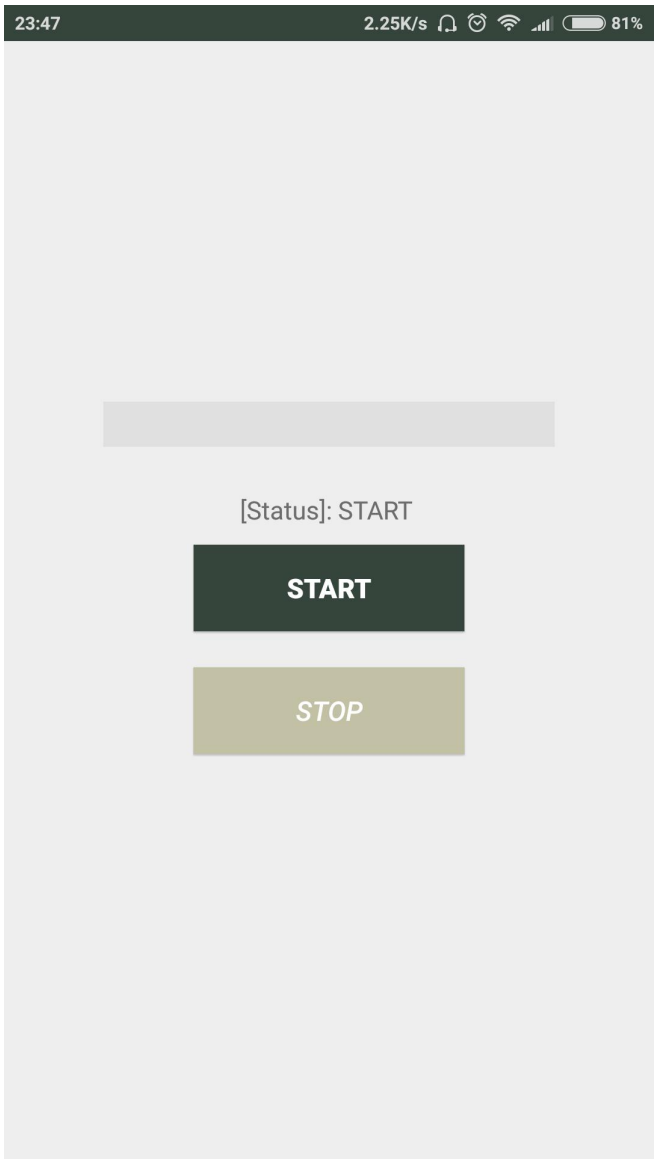

fig.1. App's initial screen

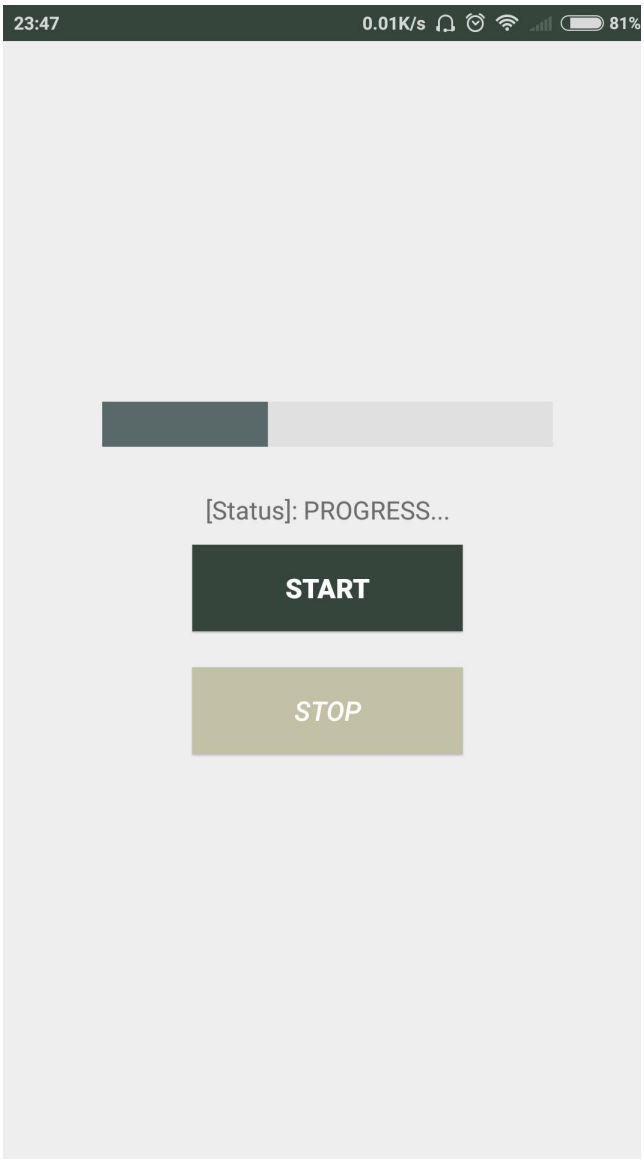

fig.2. Clicking start, fills the bar up

As visible from the previous screen shots, there's a few states the app passes through.

As mentioned in the task, these should be START, PROGRESS…, and STOP, by default, but here we have also DONE. They are noted in the status text view situated just below the loading bar.
They all can be seen in the *fig.1-4.*

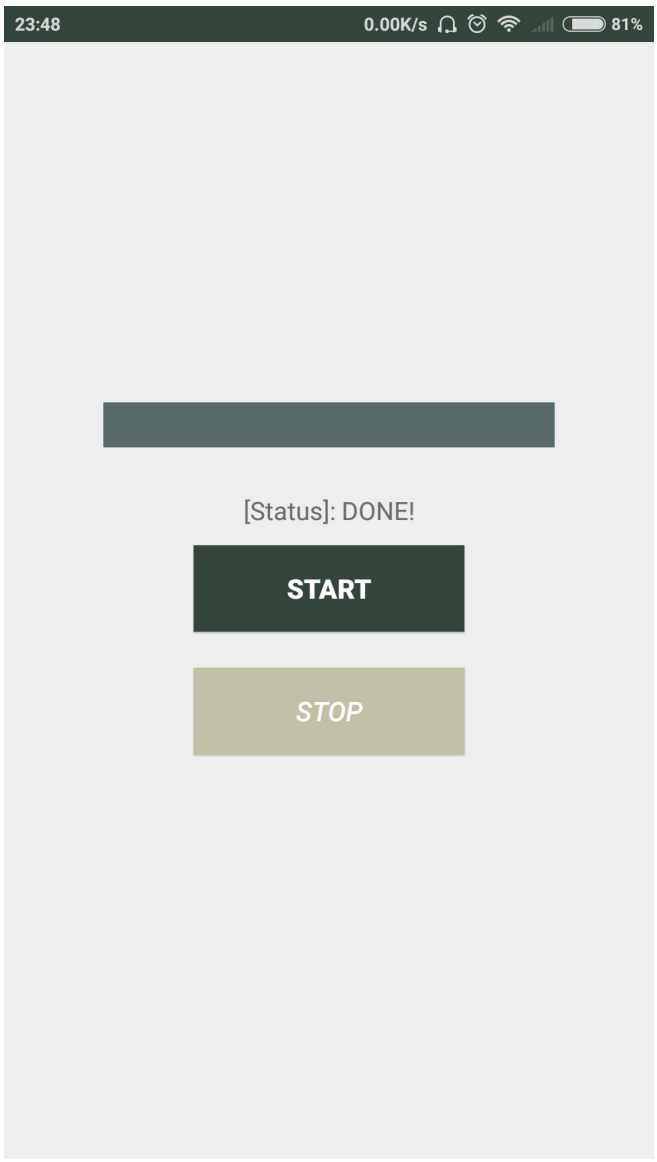

fig.3. Finishing the loading

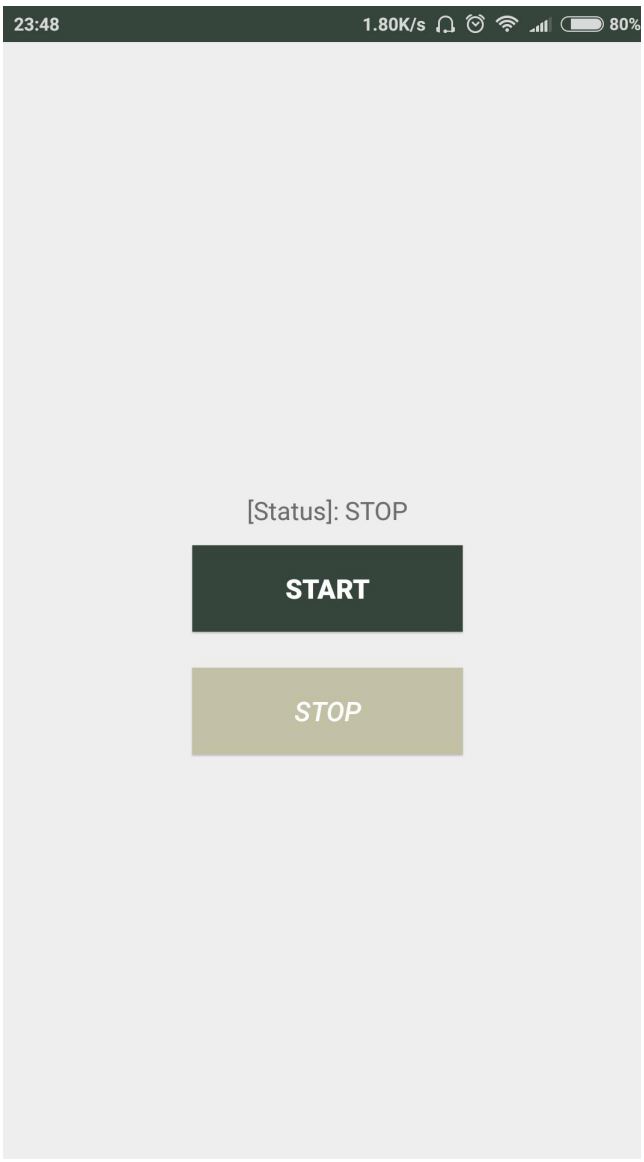

fig.4. Stopping the loading with the stop button

Due finishing the loading it is clearly observable that the state changes to DONE, and thus, no further effect is produced by clicking on the start button. However, anytime the user is able to click on the stop button, which will stop the progress (note that, stop does not mean pause), and thus, it leads to the progress nullification and restarting the progress.

Once the start button is clicked over again, after the stop button was enabled, the effect it produces is similar to the one in - *fig.2*.

Worth noting that the progress is being tracked by a handler (like async task) which cancels its runnable anytime the stop button is clicked. This way the progress will not be set to done when the expected time to load passes. In case the stop button was not touched though, it sets the progress done, just on due time.

Conclusion

During this laboratory work, there has been few challenges with the personally set tasks, and regardless the learning and problem solving skills it was futile.

The whole project took a while to complete, thankfully, the progress was linear. This is a nice fact because without progress, there's no regain of motivation in continuing, which is the requirement for any of this laboratory works to be completed. However, the bug with changing the value of the position of points in a path (i.e. the vector points of an image) within an animation didn't seem to ever work, even when trying to copy the same code from the official android site explaining the animations. It was a weird bug that didn't seem to work whatever attempt was taken to fix it. Wildly enough, the other animation types like translation, rotation or scaling were working.

Obviously, there was a bunch of studying involved for things like vector creation in xml and other animation concepts and methods to do the same thing.

This very laboratory work proved to be a great first step into the animation and beauty of app development on a mobile platform with style. Personally it is the area that was exciting for long ago!

Bibliography

- **[1] Android - Animations**
  https://www.tutorialspoint.com/android/android_animations.htm

- **Handwritings from Mobile Applications Programming course** of Lector:
  prof.univ. I.Antohi.
  Chișinău: UTM
  2017

Annex

Code source:

https://github.com/StasBizdiga/PAM