18-551 Lab 1

In this assignment, you will be creating a basic Android app, with three activities:

- A splash screen
- An image processing interface
- An audio processing interface

VirtualBox Notes (for lab computers)

The password for Ubuntu is `labuser123`

Just a reminder on how you can share folders/files between the Ubuntu guest OS in virtualbox and the host OS.

1. Go to `Devices -> Shared Folders` in the VirtualBox menu
2. Select `Machine Folders` and click on the add icon in the right panel of the window
3. Browse to the folder you want to share or create a new folder. Name it something you'll remember, e.g. "sharedFolder"
4. Now, you will have to mount the shared folder in your Ubuntu guest. Open a command window and create a directory where you want to mount the shared folder. For example, create a directory called 'HostSharedFolder' in you home directory, using the command `mkdir ~/HostSharedFolder`.
5. Now type `sudo mount -t vboxsf sharedFolder ~/HostSharedFolder`. Type the password `labuser123` when prompted
6. Now, whatever you put into your 'sharedFolder' in the host OS will appear in the folder 'HostSharedFolder' in the guest.

Splash Screen

This activity is the entry point (main activity) to your app. Design a splash screen with an image representing your group as well as the name and andrew IDs of your group members.

In order to complete this section, you will need to know the following

1. Creating a layout
2. Adding TextView widgets to your layout
3. Adding ImageView widgets

You may find the following links helpful

- TextView (developer.android.com)
- Graphical UI Editor (developer.android.com)
- Intents (developer.android.com)
- Activity (developer.android.com)

This section should be straightforward, because an activity is setup by default when you create a new project. Be sure to review the lecture notes about the use of the strings.xml file as this will be useful when creating TextView objects. Also don't be afraid to edit the layout XML files. The format is pretty easy to understand and you can often do things quicker or better than with the graphical editor.

Recall that the R.java file in gen/<packagename> contains automatically generated IDs for all resources defined in XML files. You'll have to use the constants defined in this file if you want to access object you defined in XML in code, e.g. if you want to dynamically change the position of a button, or something like that.

Whenever you create a new resource by adding a new XML file, make sure Eclipse realizes and regenerates a new R.java file; some combination of refreshing and cleaning/building usually works.

Menus

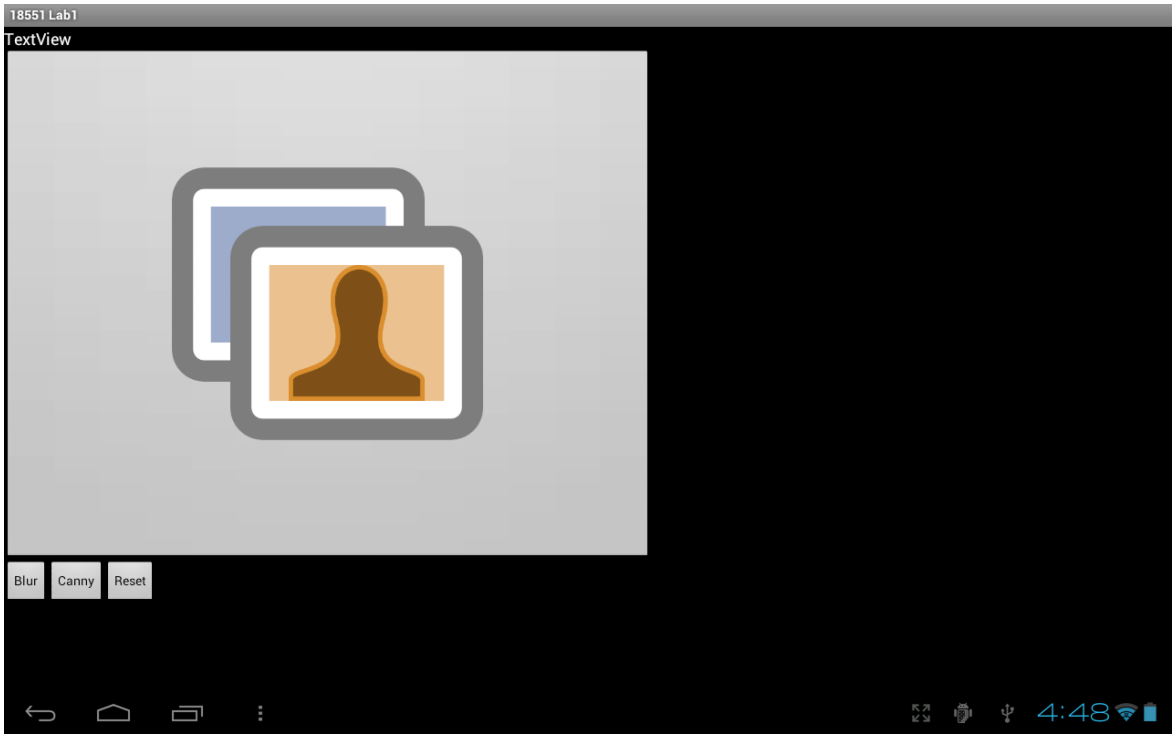
In order to transition between the image processing and audio processing activities, you will need to implement a menu with menu items:

- Image Processing
- Audio Processing

The Android developer documentation has a good explanation of how menus work. In particular, you'll probably want to create an Options Menu. The section "Creating an Options Menu" walks you through how to create a basic menu. At a high-level, you have to respond to menu creation (menus don't exist until the user presses the menu button) and then respond to selections.

Image Processing Interface

This activity, will provide an interface to the user, to load an existing image from the tablet (internal or external storage is fine) . Following this, the user is given an option to perform either image blur or canny edge detection. A sample interface screen is shown below (the final design and layout is left to you):



In order to complete this portion of the lab successfully, you will have to call the Open Computer Vision (OpenCV) functions (refer to Utils.h and the package org.opencv.android) in order to blur and in order to perform canny edge detection on an image. There are in-built functions available that perform blur and canny edge detection. The OpenCV library for android development has already been installed on the lab machines. You are expected to call the appropriate functions from your java code in order to achieve the desired results.

The image blur and canny options may be provided to the user using button widgets (you can use any other widgets for this purpose, depending on your interface tastes).

Hints:

1. Explore how image gallery is accessed through android.
2. The function startActivityForResult() will be useful.
3. You may want to determine how memory leaks can be identified using DDMS. Garbage collection, if required, can be done using the function System.gc().

Try playing around with an example Android OpenCV project (in `<OPENCV_ANDROID_DIR>/samples`). Check the troubleshooting section below if it doesn't work out of the box.

Troubleshooting OpenCV:

Can't Find OpenCV Functions (lots of red X's everywhere)

If Eclipse doesn't find the OpenCV library, right click on the project folder and go to Properties -> Android -> Libraries and add the OpenCV-2.4.3 library that is in your workspace. If there is already an OpenCV library (with a red X next to it), remove the old library. Then run Project -> Clean (make sure Eclipse rebuilds afterwards) and hopefully it will work!

If it still doesn't work, edit the location of the library in the actual `project.properties` file (located one level underneath your project folder in Eclipse), as Eclipse seems to overwrite its own settings.

Additional installation (non-code) bugs can be fixed using Android (NDK/OpenCV) Installation Notes and Googling your error

Utils.bitmapToMat() returns an empty (0 by 0) Mat

While OpenCV can handle images in many formats, the conversion functions in Utils only work with bitmaps in ARGB_8888 format. Depending on how you obtain your bitmap, it may or may not be in this format. If `Utils.bitmapToMat()` returns an empty matrix, then your bitmap is probably in RGB_565 format. If you search online, you'll find there are many possible ways to convert between bitmap formats. `Bitmap.copy()` is certainly the easiest, although it probably isn't a good permanent solution; copying image data too often is a huge performance and memory cost.

After processing, my image does not appear on the screen

If you process an image, convert it back to a bitmap, and set it as the source of an ImageView, but it still doesn't display, here are some possible problems. First, make sure your conversion in the other direction (from bitmap to Mat) is actually working, and follow the advice above if it isn't. There may also be an issue with the color format of your bitmap, but we haven't been able to figure out what setup details make this required.

Imgproc.Canny() causes an error

The Canny operator only works on grayscale images. If you convert your image to grayscale before calling the function, it should work. Be sure to convert back to color before displaying the result.

Audio Processing Interface

In this activity, you will design a user interface to load, process and playback audio files from the SD card. You will interface with the C code that you wrote in HW1 (using the NDK) to process this audio signal.

1. You will denoise the tone as you did in HW1 and play it back
2. You will modulate the tone using a sine wave and play it back.

Because of the moderate difficulty of playing back processed audio with Android, you are allowed to play the resulting .dat files on your computer (again using SoX to convert them back to .wav files)

(SoX is not installed on the VM images handed out in lab. If you have internet access in your VM, you can install SoX using `sudo apt-get install sox`*. If you don't have internet access, you can use an SD card reader on your laptop, which hopefully also has SoX on it).*

Troubleshooting Native Code

Linking Errors

There are many reasons why you might have linking errors (`UnsatisfiedLinkException` in Java); these are the problems we had. First, make sure your library/module does not have the same name as a system library. This probably won't happen, but `audio` is the name of a system library, for instance. Second, you may need to compile your library for the `armaebi-v7a` architecture, instead of the default `armaebi`. This is useful anyway, if your signal processing code uses floating point operations (it probably does). You can do this two ways: add the flag `APP_ABI := armaebi-v7a` in the `Application.mk` file, or run `ndk-build APP_ABI=armaebi-v7a` when you compile.