CS 2063 Mobile Application Development

“FinalTouch” - Final Project Deliverable

Apr. 2/19

Avery Briggs 3471065

Allan Boutilier: 3563756

**2.3.1 Product Perspective Explain in general once the Mobile App put to work. What it will do and how it will interact with other third party software components (if applicable).**

**2.3.2 Mobile App Features Explain how your mobile app will make use of the Android API package the group choose in the project phase-1**

**2.3.3 Mobile App functionality List all the main functions of your Mobile App.**

Our group has developed a simple photo-editing app that will enable the user to adjust the settings of their pictures using simple touch gestures. They will have the ability to apply high-quality image processing filters at the touch of their fingers. We have made use of the android.gesture package for our project, which will allow the user to interact with their phone in more ways than simple taps or buttons for all functionality. We will allow the user to take and edit their pictures from inside the application, quickly and efficiently. A goal we had in mind when designing this app, is to help the user discover new ways of using their phone. A lot of apps traditionally overuse the same UI elements, and we wanted ours to stand out by having the user be more interactive with our app. Our app is called FinalTouch, which is a play on words of the fact that when people often edit their photos, it is called touching them up, and that our app makes use of touch gestures.

Our application is divided into four main activities, first being the initial menu screen. This is a simple screen designed to guide the user to the other portions of the application, featuring only menu buttons that start intents. Second is the TakePicActivity, which will allow the user to take a picture from within the app using a simple image capture intent provided from the MediaStore package. The user will have the ability to save the captured image to their phone’s library which allows other applications to use the captured image. This is done through another intent that will ask for external write permission and if granted, will create an image file on the user’s device. The Third screen is a simple reference center where the user can learn the basic functionality of the app, like which filter is mapped to which gesture. There are sample images for the user to visually compare the quality of the image filter. They will also have a textView providing them with real time feedback on the interpreted gesture and the filter that was applied. The last screen in the application is FilterActivity. This is the main component of our app, and the main screen we think users will be spending the most in. It provides the user the ability to select an image from their personal gallery and insert it into an imageView for processing. Once the image is set, the user may apply any number of filters to the image (provided that the image is still processable) and view their changes in real time. Each filter addition will stack on the current ImageView, until the user manually resets the image. They then have the option of saving their edited picture back to their gallery without effecting the original image.

FinalTouch makes use of several different packages and libraries courtesy of android and [Zomato@github.com](mailto:Zomato@github.com). Zomato provided us with the necessary photo editing software that would allow us to create this app. This library proved to be very simple to implement and very powerful, it can produce very high-quality images, and quality only improves with the camera hardware from each phone. We also made use of the ViewCard package allowing us to clean up the UI with more colours and more descriptive buttons. The app is built using existing framework provided from the gesture package, we simply overwrote the desired methods with our own functionality. This was done through the use of an inner subclass within our activity that would handle our touchEvents. We decided to map most of our filters to the onFling methods because a fling seemed to be the most natural gesture that was unused. We wanted the most unique filter, the Viginette filter to have a different gesture since it is unlike the other filters, so we chose the double tap for this filter option. Initially we had designed a button that would reset the current image, deleting all the filtering progress done by the user. We wanted this to be a totally unique action, because using it by accident would be very frustrating, that is why we opted to use the onLongPress method for this option. It is a gesture that would be difficult to perform by accident and is totally different than the other types of gestures available. We also made use of some built-in intents that allow us to capture images from our app, as well as reading and writing to external storage. Our application can run locally on your device without the use of the Internet or a network connection, and therefore does not use any third-party software when running.

FinalTouch makes clever use of the android.gesture package, because it provides the user with more intuitive ways of interacting with their device. Allowing the user to use simple motion gestures, instead of creating a mapping a button to each of the desired actions, reduced the overall space that was required for the UI on each of our displayed screens within the app. We were able to exchange space that would be for a button, with an increased view of the image being edited. This improved functionality was only possible through use of the gesture package. The Implementation involved overriding the onTouchEvent method that is called when an android screen is tapped. Then creating an inner sub-class that would override the GestureDetector library, an instance of this class will be used in conjunction with the onTouchEvent method. We implemented two interfaces with this class, one to allow us to use onDoubleTapEvents, and one to easily override gestures, OnGestureListener. This allowed us to individually override each type of touchevent (doubleTap, singleTap, longPress, etc…). Some gestures are easier to work with than others, for example, when a user simply swipes across the screen, often there will be several (sometimes 5 or more) distinct gestures will be captured. That’s because the touch listener is very sensitive, and every movement is considered a new gesture. Scrolling seems to be the most difficult since it is near impossible for a human to easily swipe their screen without triggering a scroll. Because of this limitation we opted to not override the onScroll touchevent since there would be several triggered upon every touch and the results would be inconsistent and vary depending on the individual touch. We also faced other limitations concerning what would be considered a natural movement for a user to make on their phone. For example, our app makes use of four directional onFling events, up, down, left, and right. We had initially planned on including more directionality, but soon came to realize that implementing an eight-way directional onFling handler would be too sensitive and out-of-the-ordinary for a user to mimic. For this reason, we chose to stick with the four-way handler. This proves to be the main reason that custom gestures do not always catch on. If an action seems difficult or odd to the user, they won’t use that feature.

The main functions of our application include:

MainActivity:

Used as an application home screen where the user will be directed on start-up.

Provides menu options to navigate between activities.

TakePicActivity:

Access device camera and allow the user to capture an image.

Access device external storage and ask for write permission to store the image.

Provides menu options to navigate between activities.

Library Activity

Allows the user to apply a filter by using a touch gesture to some sample images.

Provides menu options to navigate between activities.

FilterActivity

Allows the user to apply a filter by using a touch gesture to a selected image.

Allow the user to reset their photo if they want to restart the editing process.

Access device external storage and ask for write permission to store the image.

Access device external storage and ask for read permission to fetch an image.

Rotate the image to the correct orientation (if necessary) before displaying.

Provides menu options to navigate between activities.

**2.3.4 Mobile App code submit the Mobile App’s source codes to D2L.**

Can be found in the attached zip file.