EE447 Lab1 Draw Line
Report

付昊源 517021910753

April 7, 2020

1 Lab Overview

This lab includes setting up Android studio and Java environment, as well as understanding Android studio file structure and implementing a naive 'Hello world' app on a virtual android machine. Besides, we also have four choices to further modify or accomplish new functions. The four choices are DrawLineSample, EX03_02, TinyDialer and GroupMessage. Here I take DrawLineSample as my choice and modify some parts of the source code as well as implementing a new function. This DrawLine app is tested on my cellphone.

2 Lab Procedure

2.1 Android Studio and Java

According to the tutorial taught on the class, I first installed Java JDK and configured the environment variables, which will be used by installing android studio by default. In next step, I downloaded android studio installation program and installed it. It's deserved to notice that the path had better include no Chinese characters to avoid troubles. After android studio installation, I also install an android virtual machine for the ease to debug and test. Actually, the most believable way to test is on real mobilephone, but it's tedious to do so, and virtual machine is a good step. After installation, the last step is to add Android SDK which is downloaded before into compile environment. And from now, I can start a demo project to test.

2.2 Android Studio File Structure

After creating a new project in android studio and running that target module, I find it has already realized the 'Hello, world!' text in a blank background. The left side of the IDE is file structure in android. I find that this file structure is different from real file structure. Maybe useful files have been re-permutated for ease to check and find. What we need to focus on is three directories: manifests, java and res. The general function of them have been discussed during class, so I only add some of my own understandings here. The widgets like textviews and buttons can be modified both in .xml files and java codes. Besides, android studio is a highly convenient tool, in which many keywords can be predicted into

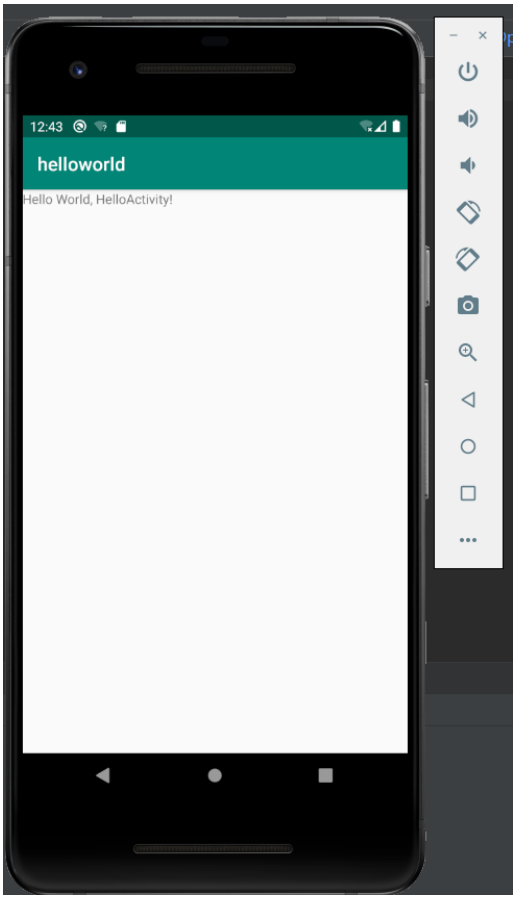

图 1: Hello World app in android virtual machine

a group of options with only part of letters typed in. And many override functions can be printed into the edit area. All these features make android studio a general development IDE and give developers more power and efficiency.

2.3 Hello World Module

The first task in this lab is to implement an app with blank background and text 'Hello world, HelloActivity!' at the top left side of parent window (cellphone screen). Actually, the only thing we need to change is in file res/layout/activity_main.xml. I modified

```
android:text="Hello World!"
```

into

```
android:text="Hello world, HelloActivity!"
```

and comment some of the constraints so that the text can be placed to the top left corner instead of original middle of the screen. Here, android studio also offers a preview window to let user know exactly what the layout will be after modification. After modifying codes, I made the project and run this module into my android virtual machine. The screen shot is shown at Figure 1.

2.4 DrawLineSample

Next task in this lab is to successfully make one of the four examples mentioned above, as well as adding some changes to codes to implement new functions. Here I choose DrawLineSample.

The original app can only draw black lines at only a part of the screen. Besides, there is no way to clean these lines except for exiting this app and opening it again. However, it has provided usage to a few powerful libraries and APIs. To get better performance, I do the following modifications:

1. Change the size of bitmap, so that it can fit any size of screen.
2. Add more colors to the drawing lines and implement a gradient color when user is writing a line.
3. Add a CLEAR button to clear all lines on the current canvas.

Next I will explain what I did for each modifications in detail. The appearance of my app is shown at Figure 2(a).

2.4.1 Resizing Bitmap

The initial bitmap is set to a fix size 480×960, which means it cannot fit any type of screen. One effective way to improve this is to get the parent width and height, and set the size of bitmap dynamically. After searching APIs, I find class *DisplayMatrics* can be used to get parent width and height pixel by

```
package com.android.DrawLineSample;
import android.util.DisplayMetrics;

DisplayMetrics dm2 = getResources().getDisplayMetrics();
int width = dm2.widthPixels;
int height = dm2.heightPixels;
bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
```

Thus, app can get the width and height pixels from system and set the size of bitmap dynamically, which is more user-friendly.

2.4.2 Gradient Color lines

In the source code, I find that there are 3 kinds of events when user is touching the screen: putting finger on the screen, moving along the screen and leaving from the screen. When user is putting the finger on the screen, app records the position x and y, for future use. Each time when app detects that user's finger is moving along the screen, it draw a line on the bitmap according to last x, y position and current x, y position. And when user's finger is put up from the screen, nothing happens and a line is finished.

To produce a gradient color line, I set a counter. Each time app detects user's movement, the counter increases by 1. And the counter is actually indexed to a RGB color space which is set by myself. Thus, the line can produce the effect of gradient color. Actually, the most difficult work in this function is to find a beautiful RGB color space, and make sure the counter is reset when it hits the margin of color space. I tested letters 'SJTU' in my mobile phone, which is shown at figure 2(b).

(a) Overview of my GUI design

(b) 'SJTU' written in colorful lines

图 2: DrawLine app tested on real mobile phone

2.4.3 CLEAR button

First of all, I have to admit that this button is a fake button. It doesn't belong to widget in android studio, but a color-filled rectangular with strings that I drew manually. The reason why I didn't use the widget button is that, in onCreat() function, the program didn't read layout from main.xml, but from TextView class. Thus, I cannot add a button onto canvas in TextView.

Each time user clicks the region surrounded by the rectangular button, the program knows that user wants to clear the canvas. It will execute clear() function to let all lines on the canvas into transparent color, which means they have been cleared. Since this pdf report cannot include gif files, TA can try this function on either real mobilephone or virtual machine using my project files.

3 Conlusion and Future Work

In conclusion, this lab helps me build environment of Android Studio and Java, and gives me the first look of android programming with some useful packages and APIs. After modifying some codes in the example projects, I understand the work flow of android studio more deeply, and basically achieve my initial thoughts and design of the draw line app, which is rather exciting.

For future works, I hope to find a proper way to add a real button to the bitmap or canvas so that it can use those button-based methods and work flow instead of drawing a fake button as I implemented in this lab.

Besides, I always use the *Notability* app in Ipad, which provides two kinds of eraser. One is partial eraser, which clears the region of the eraser passes by. This is rather easy, since it is equivalent to draw transparent-color lines (may be thicker) on the bitmap. But what's more complex is the second kind of eraser: whole eraser. It clears a whole line when the eraser touches any part of the line. I am really interested in implementing such function in future work.