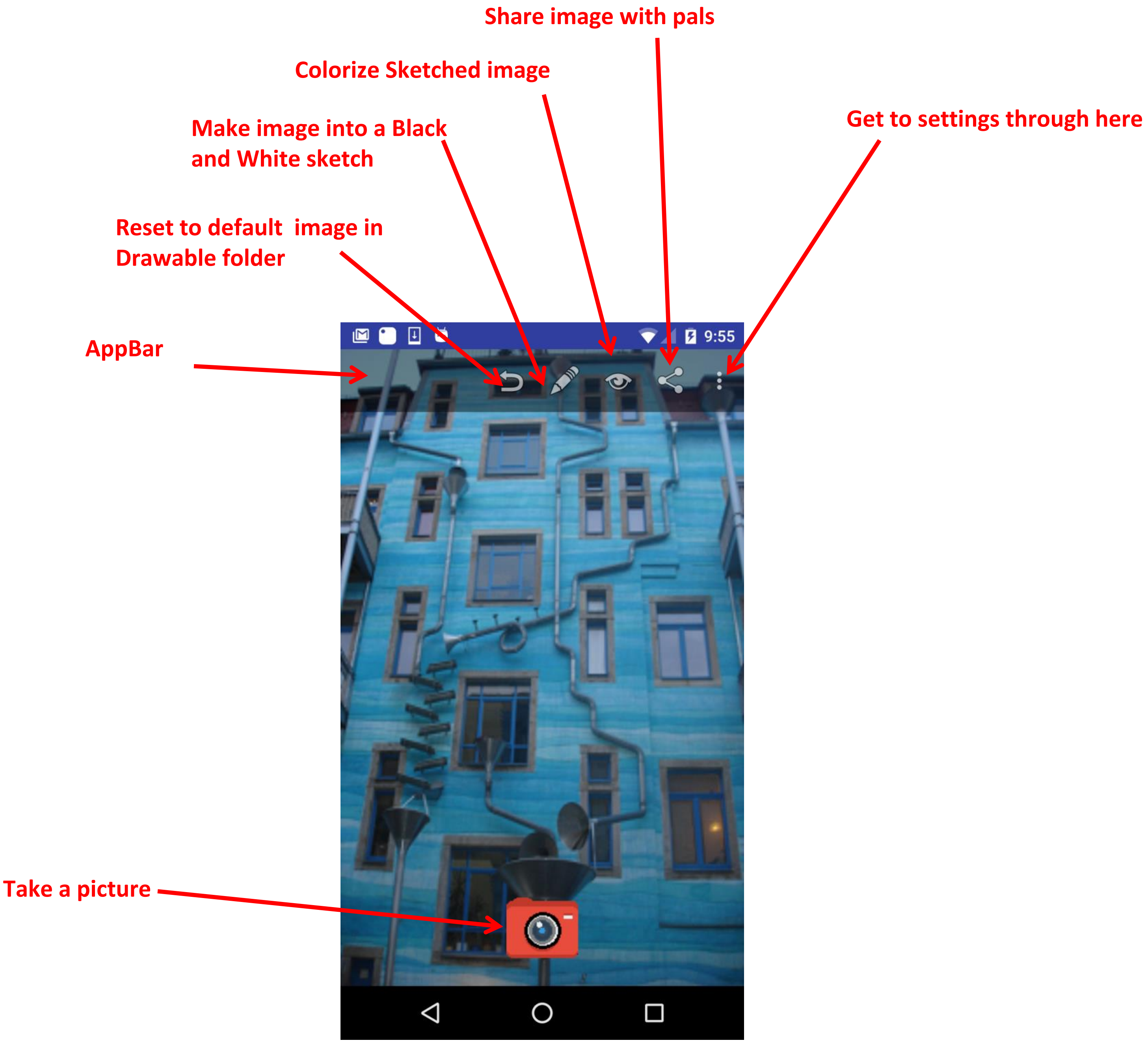
CPEN 475

Project 2: **Implicit intents-bitmap manipulation-libraries**

Overall Description

Please develop an application that can take a picture using the device camera, turn it into a Black and White sketch, colorize it, and share this image with others. On startup your GUI will look similar to the following. (Note that the image goes from edge to edge)



Share image with pals
Colorize Sketched image
Make image into a Black and White sketch
Get to settings through here
Reset to default image in Drawable folder
AppBar
9:55
Take a picture

What I have provided

- A starter project with;
    - A module library (bitmap_utilities) contains
        - Bitmap_Helpers – A wrapper class which simplifies and exposes image manipulation routines. Prefer these to other classes in this library
        - Other imaging processing static functions
    - A module Project2_Solution_Color which is where you will do most of your work.
    - This is also the module that you run the project from

Things I will look for;

1. Correctly and completely implementing project flow
2. Using the module library project I gave you as a library, please do not copy the bits you need into your project.
3. Correctly implemented AppBar
4. Getting the UI correct, this means
    - Correctly generated and displayed Black and White sketch
    - Correctly generated and displayed Color sketch
    - Correctly share a picture, Your app must append a subject, some default text and the image generated above automatically to the message. This means the user should not have to hunt for it in the messaging application.
5. Correctly implemented settings
6. When you install the app it will show a default image that you have in res\drawable
7. If you have taken a picture with the app or processed it using the sketch or colorize button then when you start the app the next time this should be th image it shows you
8. When you hit the reset button the app will again show the default image (see 5)

Grading:

10% colorize
10% sketchify
20% share
10% implicit camera intent set up with URI for file
10% onActivityResult handles both requestCode and resultCode as well as URI correctly
10% correctly implemented, error free gradle files (make sure your
10% permissions
10% settings and preference change listener
10% error checks

Project Hints

In general use the classes I gave you especially Camera_Helpers and BitMap_Helpers.

Set your camera on the lowest resolution possible to minimize waiting.

Layout

I used an imageview for the background image

Default Images

Place in the res/drawable folder. Mine is a PNG file. This image will be part of your final APK. The default image can be set in XML using;

```
android:src=
android:scaleType="fitXY"
```

You will eventually need to figure out what your screen dimensions are. The following code will give you this;

```
// Fetch screen height and width,
DisplayMetrics metrics = this.getResources().getDisplayMetrics();
screenheight = metrics.heightPixels;
screenwidth = metrics.widthPixels;
```

Permissions

You are using the camera and the file system which will probably violate Android OS security checks. Make sure you set proper permissions in your manifest. Special problems arise if you are targeting a device running Android 6.0. See the following;

http://www.captechconsulting.com/blogs/runtime-permissions-best-practices-and-how-to-gracefully-handle-permission-removal

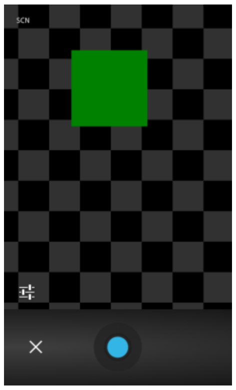
Locking Orientation

Usually the only time you want to lock orientation is when you are writing a game, where the phone needs to stay in landscape while you plink away at targets. For this app however, I am only going to test in portrait mode. In the real world, phones with slider keyboards would have a problem with this as keyboards are used in landscape mode. If your app is locked in portrait then it will appear sideways on the screen.   Use the following code in your manifest for the activity to lock your app into portrait mode if you wish.

```
android:screenOrientation="portrait"
```

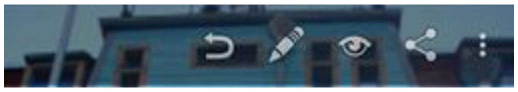
Emulator Shortcomings

Here is where the emulator is awful. Most are incapable of using the webcam on your development machine. It will however 'emulate' a camera, which looks like this.



Horrendous isn't it. It will be easier to develop this project on a physical device.

AppBar (also known as toolbar)


You have to set up an AppBar like this for this project. See lecture notes and the following links;

http://developer.android.com/training/appbar/index.html
http://www.sitepoint.com/material-design-android-design-support-library/
http://stackoverflow.com/questions/31231609/creating-a-button-in-android-toolbar
http://stackoverflow.com/questions/26505632/how-to-make-toolbar-transparent

Reset

Delete any images that may be in the devices external storage (use Camera_Helpers). Then reset the image showing with something similar to;

```
myImage.setImageResource(R.drawable.gutters);
myImage.setScaleType(ImageView.ScaleType.FIT_CENTER);
myImage.setScaleType(ImageView.ScaleType.FIT_XY);
```

Picture Help

Before invoking the camera app to take a picture, you may pass an extra EXTRA_OUTPUT to control where this image will be written. If the EXTRA_OUTPUT is not present, then a small sized image is returned as a Bitmap object in the data field of onActivityResult. This is useful for applications that only need a small image. If the EXTRA_OUTPUT is present, then the full-sized image will be written to the Uri value of EXTRA_OUTPUT.

You will need an intent to start the camera app, here is mine;

```
Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
```

You will need to specify a storage location for the camera app to put the picture
See Environment.*getExternalStoragePublicDirectory then you will build a Uri from this location. You then add the Uri to the previous intent with the putExtra syntax, here is mine;*

```
intent.putExtra(MediaStore.EXTRA_OUTPUT, outputFileUri);
```

Finaly, you will tell the system to start an app that is capable of taking a picture, here is how I did it;

```
            startActivityForResult(intent, TAKE_PICTURE);
```

startActivityForResult will invoke your devices picture taking application. The 'ForResult' portion means your activity wants the result of the cameras efforts. This comes your way via

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
```

The first parameter should be *TAKE_PICTURE* the second will be either *RESULT_OK* or RESULT_CANCELED. Please handle appropriately.

Scaling the image

Please read the following article for help on this;
http://developer.android.com/training/displaying-bitmaps/load-bitmap.html

You must not waste resources. This default image in the file should be scaled to fit your intended screen resolution. In fact most images you capture using the camera are going to have too high a resolution for your devices screen.
For example, the camera on the Galaxy Nexus takes photos up to 2592x1936 pixels (5 megapixels). If the bitmap configuration used is **ARGB_8888** (the default from the Android 2.3 onward) then loading this image into memory takes about 19MB of memory (2592*1936*4 bytes). But don't you lose resolution? No. The screen of a Galaxy Nexus had a density of 720×1280 (0.9 megapixels) anything higher is wasted. So scale your photos.

I've provided a static helper class that does all of this for you. See:

Camera_Helpers.loadAndScaleImage

Make the image Sketchy

I've provided a library of bitmap image processing functions. Some I found on the web, some I made.
This has been difficult for students in the past so I have created a wrapper class to do the hard work. See BitMap_Helpers. thresholdBmp

See the following link for Adobes description of the process
http://www.createblog.com/paintshop-pro-tutorials/14018-sketch-effect/

Colorize the image

To create a colorized image you must do several things:

1. Create a sketchy B/W image as above
2. Create a colorized image. I've create a helper function to ease this burden. See BitMap_Helpers.colorBmp
3. Add the images from 1 and 2 above. I've created another function to handle this as well. See BitMap_Helpers.merge

Invoking a communications app to send picture

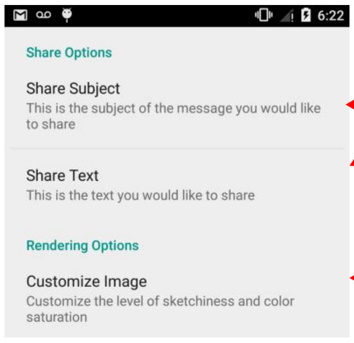
See http://developer.android.com/training/sharing/send.html#send-text-content for an overview of the process. Lots of ways to do this. For me I got the path to the directory where my picture was stored, created a File from that path and my filename like so;
File file = **new** File(path, *PROCESSED_FILE*);
Created a Uri from this file. I then used the Uri to verify that the file existed, (if not the camera failed to save the picture).
I then created an Action_Send implicit intent like so;

```
intent.setAction(Intent.ACTION_SEND);
```

attached a bunch of extra information to it via putExtra (Subject, Text of message, File Uri etc)
After that I started the activity. I also included a chooser, which is a disambiguation dialog that allows the user to choose an app to handle the request. You will need to research this process as well.

Settings

Clicking the 3 dots brings up the settings preference Activity:



6:22

Share Options

Share Subject

This is the subject of the message you would like to share

Share Text

This is the text you would like to share

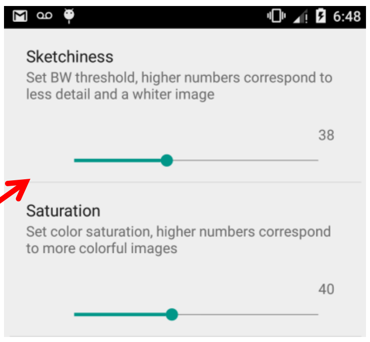
Rendering Options

Customize Image

Customize the level of sketchiness and color saturation

Calls up an EditText that allows you to customize your share info

More complex: I did this Using the seekbarpreference class included in project Or just 2 edittext boxes that only take numbers



6:48

Sketchiness

Set BW threshold, higher numbers correspond to less detail and a whiter image

38

Saturation

Set color saturation, higher numbers correspond to more colorful images

40

Customizing the image.

Select the 'Customize Image' preference screen above to decide the percentage of sketchiness and saturation. Two options:

1. You can use 2 EditTextPreference's that only take numbers (not shown)
2. You can use the SeekBarPreference class from
3. http://robobunny.com/wp/?p=190
   If you do it this way I'll add 5% to your project grade.