Android App Programming Directed Study ~ DrawingFun

Jenny Huang

December 8, 2014

Contents

1 first checkin 10/27/2014 **1**
1.1 Goal . . . 1
1.2 Course Introduction . . . 1
1.3 Project Introduction . . . 1
1.4 References . . . 1

2 Checkin for 11/3/2014 **3**
2.1 Buttons I have worked on . . . 3
2.1.1 Color$_{Picker}$: . . . 3
2.1.2 Undo/Redo: . . . 3
2.2 Functionalities and References . . . 3
2.2.1 Color$_{Picker}$: . . . 3
2.2.2 Undo/Redo Buttons: . . . 3
2.3 Snapshot . . . 5
2.4 Todo . . . 6
2.4.1 Drawing shapes with Finger for primitives . . . 6
2.4.2 Load image file button . . . 6
2.4.3 Erase Rectangle . . . 6
2.4.4 Undo/Redo . . . 6

3 Checkin for 11/10/2014 **6**
3.1 Buttons I have worked on . . . 6
3.1.1 shapeBtn for primitives . . . 6
3.2 Functionalities and References . . . 6
3.2.1 shapeBtn for primitives: Drawing shapes with Finger for primitives . . . 6
3.3 Snapshot . . . 8
3.4 Special Situation . . . 9
3.5 Todo . . . 9

4 Checkin for 11/17/2014 **9**
4.1 Buttons I have worked on . . . 9
4.2 Functionalities and References . . . 9
4.2.1 openBtn for loading an image file as an ImageView . . . 9
4.2.2 Undo/Redo . . . 9
4.2.3 References: all about Android . . . 10
4.3 Snapshot . . . 12
4.4 Todo . . . 13

5 Checkin for 12/01/2014 **13**
5.1 Buttons I have worked on . . . 13
5.2 Functionalities and References . . . 13
5.2.1 ImageView to Bitmap . . . 13
5.2.2 start newBtn . . . 13
5.2.3 Undo/Redo . . . 14
5.3 Snapshot . . . 16
5.4 Todo . . . 17

6 Checkin for 12/08/2014 **17**
6.1 Buttons I have worked on . . . 17
6.2 Functionalities and References . . . 17
6.2.1 Undo/Redo . . . 17
6.2.2 Erase button . . . 17
6.2.3 FloodFill . . . 17
6.2.4 Other Issues . . . 17
6.2.5 report . . . 18
6.3 Snapshot . . . 20
6.4 Todo . . . 21

7 Course Review **21**
7.1 Course Goal and General Review . . . 21
7.2 Course Benefits . . . 21

1 first checkin 10/27/2014

1.1 Goal

- According to the instructor's requirements that we are going to implement an simple window's Paint like Android app for later on integrating Unicon's 2D graphics to Android app.

1.2 Course Introduction

- We have only two students, the other one is an udnergraduate exchange students with solid Java programming background and relatively slightly week problem-solving skills. For the first more than half semester, we used Sudoku as the starting point and tried several different topics to get our hands wet.

1.3 Project Introduction

- It's after middle term already, the way we were currently trying on to make it work may just work perfectly for the other classmate, but for me, I feel like it takes forever for me to be able to make any significant progress. So about half a month ago, I was motivated and thought instead of surfacing around and having fun learning by trial and error, maybe I should start from an simple GUI app as a starting point and try my best to expend/extend the APP functionality from there. And also we would be able to work to our final project slightly earlier.

- This GUI will be my very second GUI interface that I have ever created for my Computer Science major, (this first one was an Python Tkinter GUI one week short project for plotting graphics with data abstracted from backend database during an internship;). And I guess it may still be slightly difficult for me to start write Android App code of my own line by line, so I simply searched internet, and trying an tutorial to make a working starting point Android Paint GUI. I integrated the codes from the reference link all together, fixed minor compile errors, and it worked!

- This "Copied" GUI will serve as the starting point, and my functionality updates start from here, and I will update my progress for this project later on by week according to the instructor's requirements and suggestions.

1.4 References

- http://code.tutsplus.com/tutorials/android-sdk-create-a-drawing-app-interface-creation--mo


98%
12:38
Drawing App

2 Checkin for 11/3/2014

2.1 Buttons I have worked on

2.1.1 Color_{Picker}:

2.1.2 Undo/Redo:

2.2 Functionalities and References

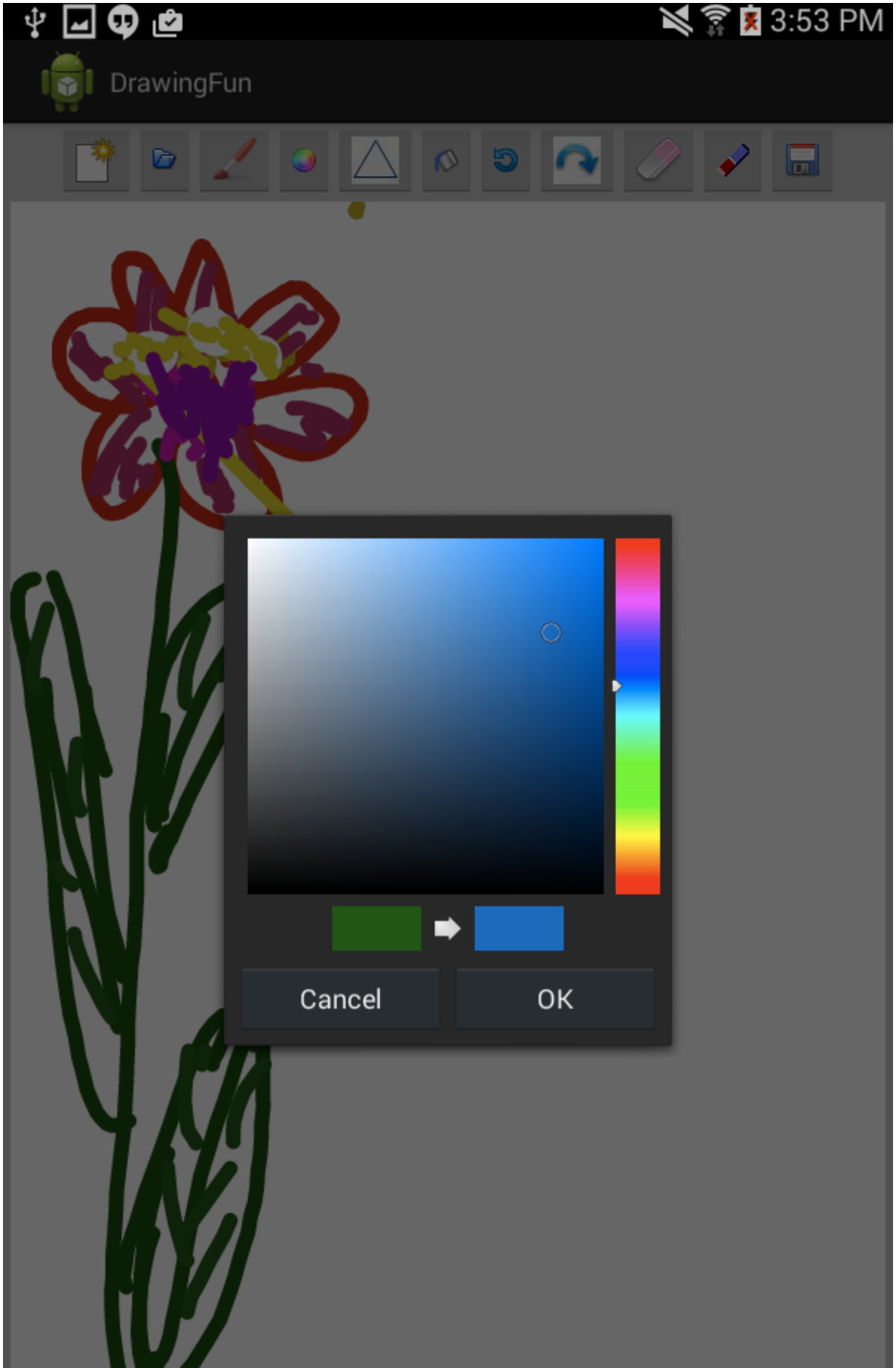
2.2.1 Color_{Picker}:

- Motivated by the Picasso Android app, seeing their multiple color choices, our starting point **12** fixed colors were too limited.

2.2.2 Undo/Redo Buttons:

- Also motivated by the Picasso app, intended to work on **Undo** button, and ended up found **Redo** button could be very convenient as well.
- needs to update these Undo/Redo methods later on, this is just the starting point most basic implementation for this button set.

2.3 Snapshot



2.4 Todo

2.4.1 Drawing shapes with Finger for primitives

refer to the reference below:

- http://gmariotti.blogspot.com/2014/01/drawing-shapes-with-fingers.html
- This button will be first priority to finish

2.4.2 Load image file button

2.4.3 Erase Rectangle

2.4.4 Undo/Redo

3 Checkin for 11/10/2014

3.1 Buttons I have worked on

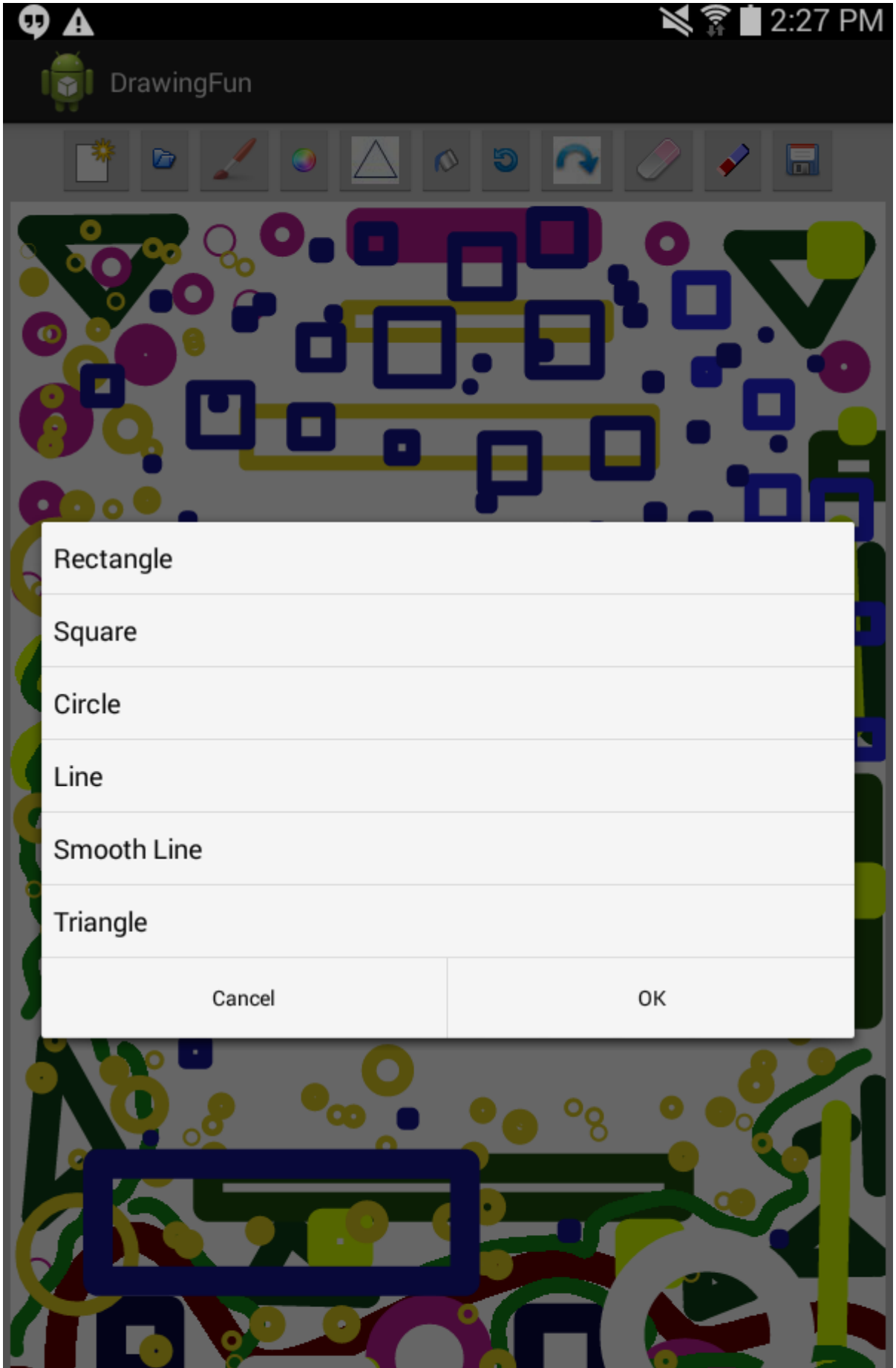
3.1.1 shapeBtn for primitives

3.2 Functionalities and References

3.2.1 shapeBtn for primitives: Drawing shapes with Finger for primitives

- refer to the reference below for some basic shapes: line, smooth line, circle, triangle, Rectangle, square
- http://gmariotti.blogspot.com/2014/01/drawing-shapes-with-fingers.html
- **ListView** in **Alert Dialog** is searched from online without direct reference.
- Since the erase was using draw smooth line. This button works also means that I could erase a "**Rectangle**" shape, or "**Circle**" shape.
- I have other course priority for the passed week, so I just have enough time to finish this course's priority, but I will try to work harder in order to finish all the functionalities for this course.
- It's not a good looking ListView, but yet it's a fully functional button.
- This button right now is fully functional, but to finish this project first, I have not spent any quality time to expand any primitives yet, rather than the existing six ones from the reference listed below.

3.3 Snapshot


2:27 PM
DrawingFun
Rectangle
Square
Circle
Line
Smooth Line
Triangle
Cancel
OK

3.4 Special Situation

- There were too many students piled/lined up in front of Dr. Jeffery's door, and he didn't break the line by stating that it's our direct study time. So the other classmate and I just stepped away from his office, and we didn't really meet during last week.
- The other classmate and I have talked, and we happened to have worked on the same shapeBtn, I applied ListView in a dialog box with all six drawing shapes applied, and he created a (ListView? not sure) with a clickable button as one element with four shapes applied. And he agreeed my ListView looked way prettier than his buttons did.
- But I am willing to and more than happy to think that he could have worked on something else important for him that I actually didn't have time to work on during the passed week.

3.5 Todo

- Load image file button
- Erase Rectangle
- Undo/Redo
- Fill paint

4 Checkin for 11/17/2014

4.1 Buttons I have worked on

- openBtn for loading an image file as an ImageView
- Undo/Redo

4.2 Functionalities and References

4.2.1 openBtn for loading an image file as an ImageView

- The method I applied is memory saving for AsyncTask, which is better than load images directly, which could potentially block UI for couple of seconds;
- Loaded an image from online, but would like to try load internal images from device later on, like a drawing which I saved earlier onto my internal device;
- Potentially apply layer oncepts to produce multiple layer drawing, needs suggestions to organize my idea how to implement this feature.
- **Question**: Right now, my image is an ImageView in my layout, what ideas that I could use to set/change/-transfer my ImageView to be my draw view background?
- References:

  http://www.learn2crack.com/2014/06/android-load-image-from-internet.html

  http://stackoverflow.com/questions/5776851/load-image-from-url

  https://github.com/koush/UrlImageViewHelper

4.2.2 Undo/Redo

- After implemented subclass SuperActivity class which extends Activity on week checkin for 11/10/2014 for my ListView implementation, subclass of Path() was very difficult for me to think about implement before, but after my trial on ListView, super/sub class in Java all made sense to me now. It's a piece of cake, and I know I can wrap whatever material I need in order to paint nice and neat.
- Implemented by developing a subclass myPath to wrap the super Path(), drawPaint color, and drawPaint strokesize together as an object.

- Based on previous progress that I can undo/redo only with all the drawCanvas with the same paint color, now my updo/redo paths could be colorful and with various strokesizes.

- References:

  Path() library:

  http://grepcode.com/file/repository.grepcode.com/java/ext/com.google.android/android/2.3.1_r1/android/graphics/Path.java

  Bitmap cacheing:

  http://stackoverflow.com/questions/3406910/efficient-2d-drawing-in-android/3408641#3408641

- **Questions**:

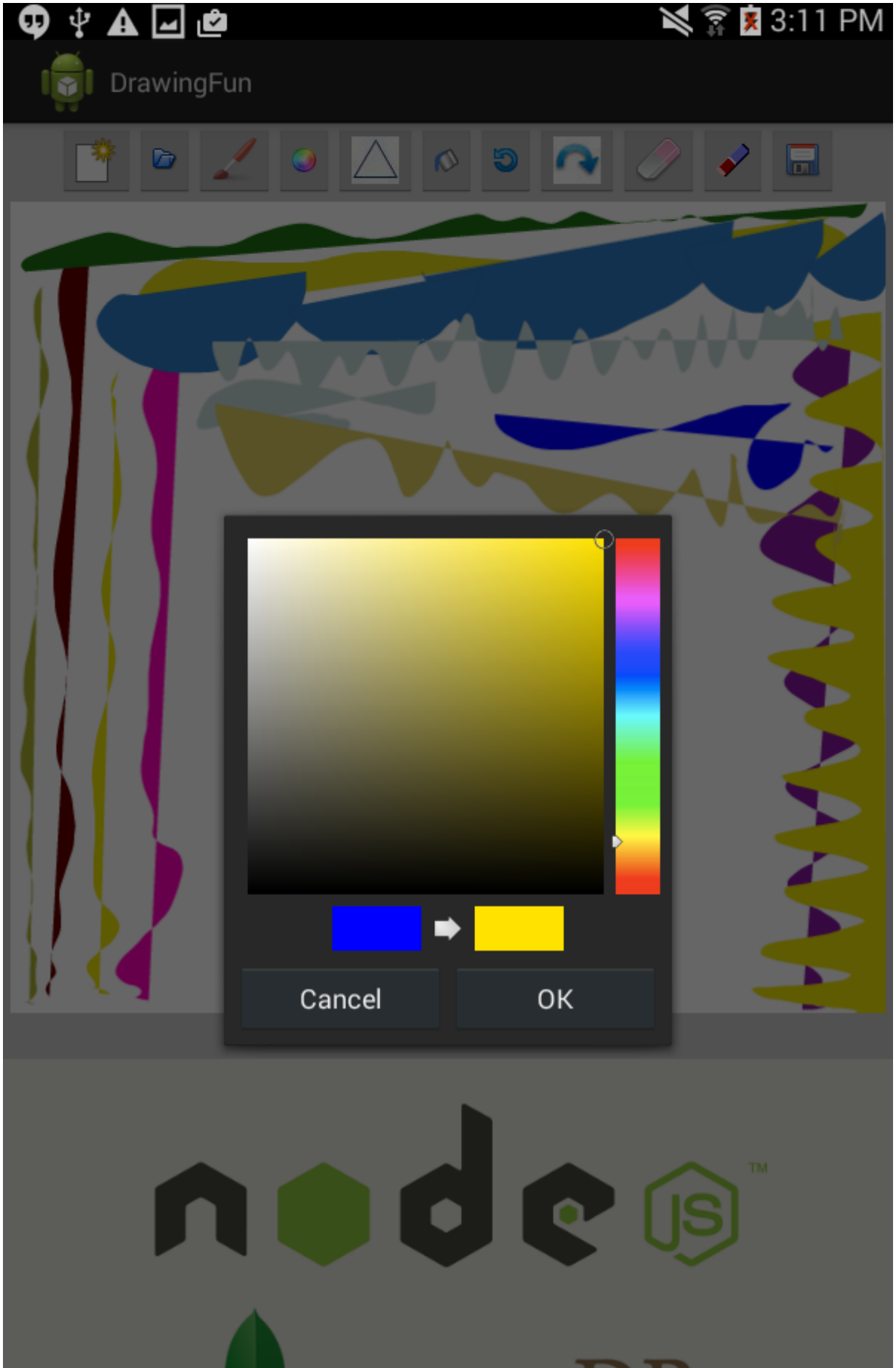
  1. Undo/Redo for simple path seem to behavor fairly ok, but instead of lineTo wired line, how do I implement smooth line? How could I differentiate different strokesizes more clear with lines I have so far?
  2. One little detail though, I dras after touch up, my paint color change delayed, how do I implement **real time**?

     Answered: drawPath.reset() produced all the trouble.
  3. About previous ListView six different shapes, with Undo/Redo properly functionaing, I realize I just lost my siz shapes again cause I need to rewrite/implement methods in order for them to be able to Undo/Redo ~? (My subclass works perfectly for this propose, just that I lost my internal link to primitives, which means I probably should rewrite my primitives draw methods according to undo/redo prerequirements. I don't think it will be difficult, but I don't have enough time for this for the pass week, and I need to organize my ideas about these implementation clear.)
  4. I prioritize undo/redo to be more important than any other buttons cause I know they would give me great practise together with primitives implementation methods rewrite. So I have not touch "Erase Rectangle" button and "paint fill" button yet. According to these idea, I would prioritize Rectangle rewrite with the highest priority, so that later on I can follow up with erase Rectangle (which means draw Rectangle first, fill with background color, and undo could remove this erase step). Correct me if I am wrong.

4.2.3 References: all about Android

- https://github.com/kesenhoo/android-training-course-in-chinese

4.3 Snapshot



4.4 Todo

Only two button left untouched, could do the following or anything I am interested to implement.

- Erase Rectangle
- Fill paint

May try to **save** into Galaxy… as Dr. Jeffery mentioned it last time when we meet during class;
Potential interests: may implement depends on how I spend thanksgiving ~

- touch ImageView Activities: zoomin, zoomout, rotate, fading, etc
- SurfaceView rotate images through new thread
- canvas save() and restore()
- OpenGL spinning circle
- widely used draw methods
- Easy draw operations

5 Checkin for 12/01/2014

5.1 Buttons I have worked on

- ImageView to Bitmap
- start newBtn
- Undo/Redo

5.2 Functionalities and References

5.2.1 ImageView to Bitmap

- Worked on Bitmap so that I can load a picture as my drawView background;
- This could be considered to be a trial, and could try to add user options to different background pictures later on;

5.2.2 start newBtn

- Realized that my newBtn lost its functionality during last checkin because of different mechanisms, and I fixed it after having implemented undo/redo for paths;
- The wired drawing path shapes (like the dramatic curves in previous Snapshot) got corrected as well by writing to Bitmap;
- References: mutuable immutable bitmaps
  http://stackoverflow.com/questions/13119582/android-immutable-bitmap-crash-error
- But I still failed to start new because some minor error about implementation. I uninstalled the app and restart, the error was still there;
- I was so focused on the mview thing that I completely lost focus on the true reason. Once I asked and the instuctor helped explain that invalidate() simply calls onDraw() function, I could immediately realize that I forgot to clean my undo/redo arraylist paths and undonePaths!
- It was the invalidate() function confused and prevented me from relaxing on the mview, and I was stubborn there for about one hour this afternoon. Realizing that I felt so sorry for myself for the one hour being so stupid! And right now I am on my way following the good habit reading Qt creator documents systematically before googling the correct answer only when I try to solve my technical difficulties, which is good.
- While it still worths a minutes to rewind and rethink about what happened during that one hour, how I trusted myself so much and suspected on low probability corner cases situations, rather then double check and confirm that all the steps/processes I had made were correct and reliable. I wished I spend the hour with a scientific attitude the latter.

5.2.3 Undo/Redo

- If I really don't want to separate/pack my ListView items into objects, will it be possible for me to use command pattern instead, and how difficult could command pattern to be comparatively spearking?
- References: List:
  - http://stackoverflow.com/questions/11114625/android-canvas-redo-and-undo-operation

  Command Pattern:
  - http://www.28im.com/android/a141932.html
  - http://www.javaworld.com/article/2077569/core-java/java-tip-68--learn-how-to-implement
    html
  - http://www.28im.com/android/a141932.html
  - http://blog.csdn.net/lovingprince/article/details/1532869
  - http://www.2cto.com/kf/201409/333267.html
  - http://www.2cto.com/kf/201406/309574.html
  - http://blog.csdn.net/rhljiayou/article/details/7212620
- Answers:
  - We didn't really talk about command patterns at all this afternoon, but rether to solve both the other classmate's and my technical difficulties, and also discussions about the questions we raised, for example, my interested ones including multiple layers Potentials when using Bitmap and removing any layers afterwards, and autosave nsapshots if we save bitmap every 20 minutes, and Potential values we could apply with those save displays in paths & undonePaths during each 20 minoutes interval.
  - I began to realize that I COULD have my own little brain-turning/intuitive ideas when I began to understand things, like I spent hours today just to understand Bitmap~

5.3 Snapshot

5.4 Todo

- finish the undone functions and wrap up project and do basic demo on coming Monday;
- short about one page summary, could at most to be 2 pages;

6 Checkin for 12/08/2014

6.1 Buttons I have worked on

- Undo/Redo
- Erase Button
- FloodFill

6.2 Functionalities and References

6.2.1 Undo/Redo

- **Cleaned** my contamination or origial bitmap in DrawView **clear()** method by replacing "new2Bitmap = originalBitmap; " with "new2Bitmap = bridgeBitmap.copy(Bitmap.Config.$ARGB_{8888}$, true);"
- I used bitmap only for the propose of adding the Yellow Rose which I liked it too much and wanted to keep it as a corner backgroung; But for the rest of drawings, they are all drawn on canvas instead of bitmap;
- I could draw all the contents in bitmap, but my **Straight Line** looked really wired on bitmap when I draw in progress, but otherwise I don't have any clear idea how to remember the start and end points and draw a straight line during onDraw. I need some idea here, if I continued to use bitmap instead of canvas;

6.2.2 Erase button

- Becuase I liked the Yellow Rose too much, I had to conpensate and rewrite the erase function to draw shapes using background color, because the old method doesn't work any more when I used bitmap; The function itself was not difficult at all though.

6.2.3 FloodFill

- Implemented on Bitmap instead of canvas
- I used bitmap only for the propose of adding the background image and do the FloodFill on the background image. But for the rest of drawings, they are all drawn on canvas instead of bitmap;
- Applied the following method, but it was way too slow, and look urgyly
- http://stackoverflow.com/questions/12669740/android-using-flood-fill-algorithm-getting-out
- http://stackoverflow.com/questions/8070401/android-flood-fill-algorithm
- http://www.codeproject.com/Articles/364413/Queue-Linear-Flood-Fill-A-Fast-Flood-Fill-Algor
- http://stackoverflow.com/questions/8723590/fill-the-complete-canvas-but-keep-the-bound-fil 12777805#12777805
- http://blog.csdn.net/jia20003/article/details/8908464

6.2.4 Other Issues

- setBrushSize
  - Issue: the setBrushSize option always changed the drawView color back to initial default color, which is Color.BLUE, and which is not convenient;
  - In **MainActivity**, when $draw_{btn}$ I setBrushSize, I have to do **drawView.setColor(mColor);**, otherwise, it always set drawView's paintColor to be default Color.BLUE;

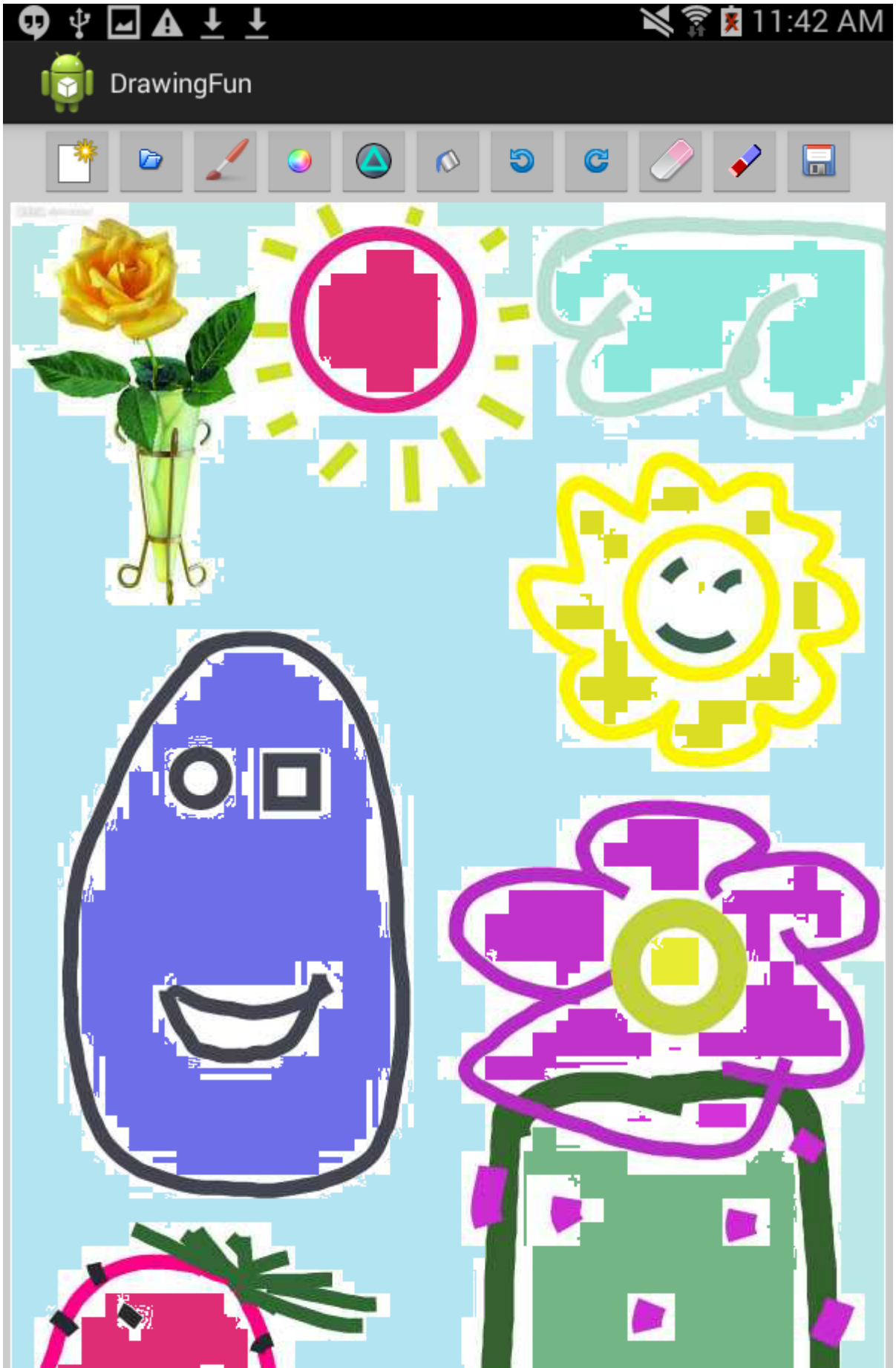
  - But I don't think the above implementation is logical. I don't think click on draw$_{btn}$ need to do anything about color when I suppose to set the brushSize; But I have difficulties to understand the process and find a better "logical" solution for it.

- onSizeChange
  - I think I have bug on **onSizeChange** function, because whenever I changed my device from vertically to horizontally, all the contents on my canvas just went away.
  - I can think the reason is because I didn't really draw anything on the bitmap yet, and that's the reason whenever I changed from vertical to horizontal, I have only my fresh loaded background bitmap;
  - But, if I want to draw on bitmap, I will have to reimplement my undo/redo differently, the current ArrayList method won't work any more.
  - If I reimplement on bitmap, what will be the good idea to implement it?
  - I would be happy if the instructor helps Introduce a little bit more 0about bitmap, undo/redo on it, and its utilities.

6.2.5 report

Course report is in **report.org** file, and main sections are also copied into the followed section for the reader's convenience.

6.3 Snapshot



6.4 Todo

7 Course Review

7.1 Course Goal and General Review

- Taking this course, I wanted to help myself stay on schedule and learn some cutting-edge knowledge as a starting point.
- I never had any "new" knowledge like "Android" learned before, this is the first time, and I enjoy it;
- I enjoyed two modules the most: the color popup dialog and undo/redo functionalities. And in the middle, ListView helped a little bit as well;
  - The color picker was not my original work, but for me at that time, it was very complicated and it forced me to understand all the Android framework for an App to function, the manifest, layout, value etc;
  - To implement a fully functional ListView together with the rest functionalities, I figured out my own way of creating a bridge SuperActivity class, which in term of Java-programming, created a start point of confidence that I can implement my ideas (any idears) in Java as far as I **Think** it through, clear. **It is always the ideas that matter, instead of any implementation.**
  - For undo/redo interface/implementation, I had thought to skip around by implementing Command Patterns, but now I am glad that Dr. Jeffery insisted us to apply interface/implementation. And I had been frustrated yet more than happy take my own effort to try, step by step, implement and see eventually it is working~! And during this process, I felt I began to be exposed to Java OOD, Android canvas, bitmap, drawing primitives, and I understood the theory behind them now, even only the parts that I implemented.

7.2 Course Benefits

- The latter half semester of implementing DrawFun Paint project helped me realize that I can perform great in concentrated topics, which helps me focus.
- It has been a challanging and interesting learning experience during this Android App Programming, and it successfully reached the target which I expected from this course. I learned the basic necessary knowledge to build my Android App and Java Programming background, and I practised and cultivated the necessary and usefull skills to think logically, solve problems and debug my codes.
- The course built knowledge, practised skills, as well as built confidence in programming and problem-solving, and help cultivated my **I CAN DO** attitude towards projects.
- After taking this course, I have a sufficient starting point to self-learn and practise Android App Programming. And now I am ready to prefer Java over c++ as my primary and first choice programming language, and I will try to conduct more practise on Java programming so I can be proficient on it in not far future.