

Score Calculator

Goal

The goal for this homework assignment is to get you practice with Android Studio using Buttons, Text Views, and doing a little layout. You will be working individually, but you can ask others for help of course. You will be creating a calculator app that is specific to the scoring system used in the Golf Ball Delivery project. Here is a link to the full info on the [project](#) and here is the link to an [individual scoring sheet](#). To be honest the individual scoring sheet is all you need to look at, open that link now. Usually we do that math by hand on the sheet, but why not make an app to do that. :) Here is an example screenshot of one possible Score Calculator app UI:

Distance [ft]	Points
Near Ball	0
Far Ball	0
Robot Home	0

Look at the [individual scoring sheet](#) next to this screenshot and you'll hopefully be able to figure out how things work without much explanation ([here is a video](#) if you WANT more explanation or you can download the app by visiting <http://www.rose-hulman.edu/~fisherds/AndroidApps/ScoreCalculator.apk> from your Android device). **Your UI can be anything you want** so long as you meet the requirements listed below. The screenshot above is just one implementation. As for the "beauty" of your UI it must look **somewhat ok** on the device you use to demo your app. If your app looks TERRIBLE on any other size screen that is just fine. :)

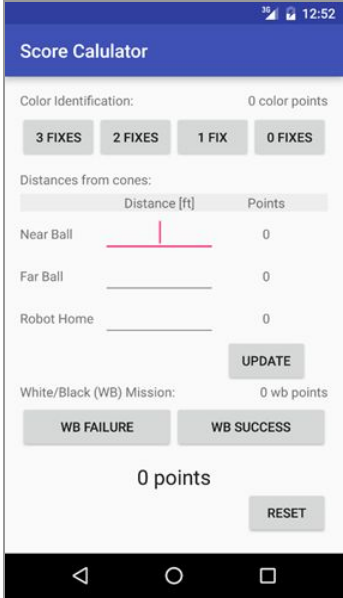
Functional Requirements

- Must be able to accurately implement the math needed for the [individual scoring sheet](#)
- Must not require a user to know point breakdowns. They need to be able to enter things like number of balls fixes / right, distances away (in ft), and whether the white/black ball mission was a success. Your app converts all of those things to points
- Must be easy to use and clearly show (in some way) the breakdown of how their score reached that total
- Must be able to reset the calculator for a new match
- Must look somewhat ok on the device used (layouts specific to that device are fine)

Your demo

You will demo your app to a TA or Dr. Fisher using either a real device or an emulator (real device is preferred). During your demo you will be asked to perform the following tests:

Perfect score

	<p>Initial screen before any buttons are pressed</p>
--	--

Score Calculator

Color Identification: 150 color points

3 FIXES 2 FIXES 1 FIX 0 FIXES

Distances from cones:

	Distance [ft]	Points
Near Ball		0
Far Ball		0
Robot Home		0

UPDATE

White/Black (WB) Mission: 0 wb points

WB FAILURE WB SUCCESS

150 points

RESET

Some mechanism to get a perfect ball color identification with no fixing needed (here I pressed 0 Fixes)

Score Calculator

Color Identification: 150 color points

3 FIXES 2 FIXES 1 FIX 0 FIXES

Distances from cones:

	Distance [ft]	Points
Near Ball	1	110
Far Ball	4	220
Robot Home	0	110

UPDATE

White/Black (WB) Mission: 0 wb points

WB FAILURE WB SUCCESS

590 points

RESET

Some mechanism to put all three distance values to 0 to 5 (perfect scores)

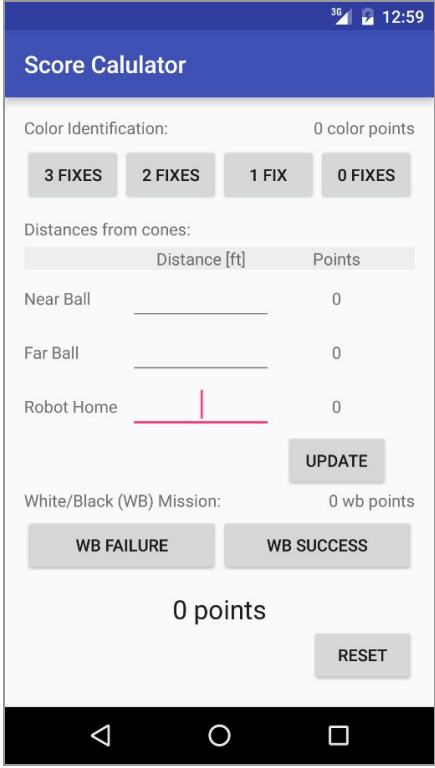
Notice: I added a button called Update that will update the distance points based on the distances [ft]. Personally I know how to add a listener for keys (it's called a TextWatcher) and that does work better than requiring a user to press Update. However I added the Update button, just to let you know that adding a button and only taking action on the button press is totally fine. This is your first app, no need to be perfect.

Also if you pick a totally different UI (perhaps a dropdown list of feet away ranges), then an Update button might not make sense at all anyway.

Score Calculator		12:57
Color Identification:		150 color points
<div>3 FIXES2 FIXES1 FIX0 FIXES</div>		
Distances from cones:		
	Distance [ft]	Points
Near Ball	1	110
Far Ball	4	220
Robot Home	0	110
		UPDATE
White/Black (WB) Mission:		60 wb points
<div>WB FAILUREWB SUCCESS</div>		
650 points		
		RESET

A mechanism to score the points for the white / black ball mission

That series of events should show the score as 650 (a perfect score). After doing the perfect score we'll test Reset

	<p>Clicked Rest. All things back to exactly the same as when the app opened.</p>
---	--

After that we'll just randomly click around your UI a bit to make sure everything works as expected.

Again feel free to make the UI whatever you find fun. Just thinking of ideas I'm sure radio buttons (RadioGroup) could be used for the color fixes UI, sliders (SeekBar) might work for the distances, a White/Black success switch (Switch), or maybe putting the Reset in the Action Bar. Try something out to make this app yours. OR make it exactly like my demo (which tries to use skills you already know most of the time). ;)

You will probably find the hardest part is making the layout look good. Don't go nuts with it if you don't want though. :) For most of you it's your first solo app. Making it work and making it useable is plenty. Lining things up is more for the OCD in me.

3G 1:07

Score Calculator

Color Identification: 75 color points

3 FIXES 2 FIXES 1 FIX 0 FIXES

Distances from cones:

	Distance [ft]	Points
Near Ball	10	100
Far Ball	20	160
Robot Home	30	50

UPDATE

White/Black (WB) Mission: 60 wb points

WB FAILURE WB SUCCESS

445 points

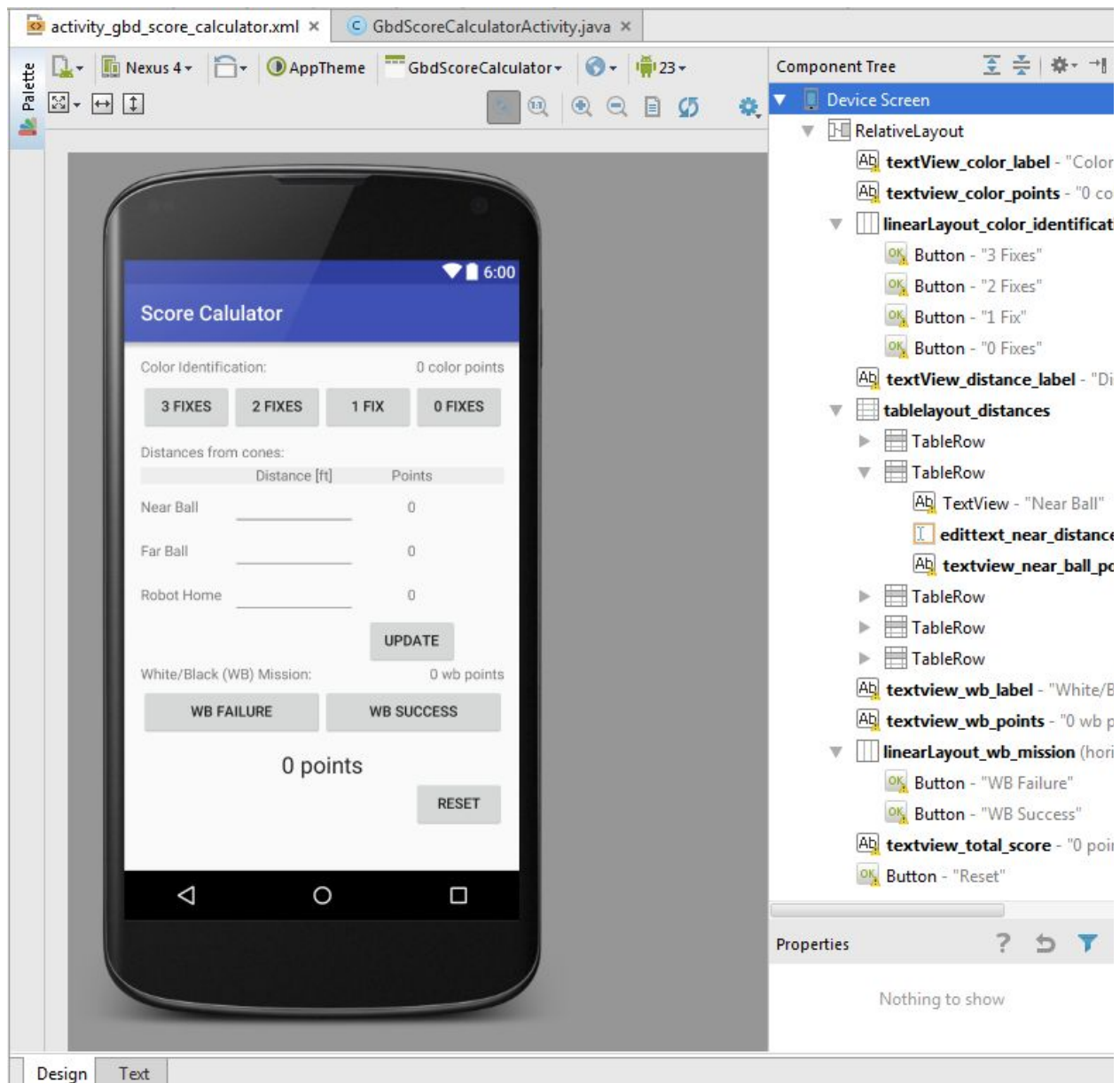
RESET

Hints and suggestions

This is COMPLETELY optional reading. Just trying to help if you are feeling overwhelmed with all the new stuff going on. If you are following **roughly** the UI shown here, these are some tips for getting going.

Step #1: Start with the layout

If you want to reproduce the screenshots you will have to learn a bit about layouts on your own. Learning is ok. You are allowed to learn things beyond what you have been taught to solve this problem. :) Here is the layout that I used:



I choose to use some Linear Layouts and a TableLayout. If you wanted to learn about them there are MANY resources on the web. LinearLayout is VERY easy to learn. For example: <http://developer.android.com/guide/topics/ui/layout/linear.html>

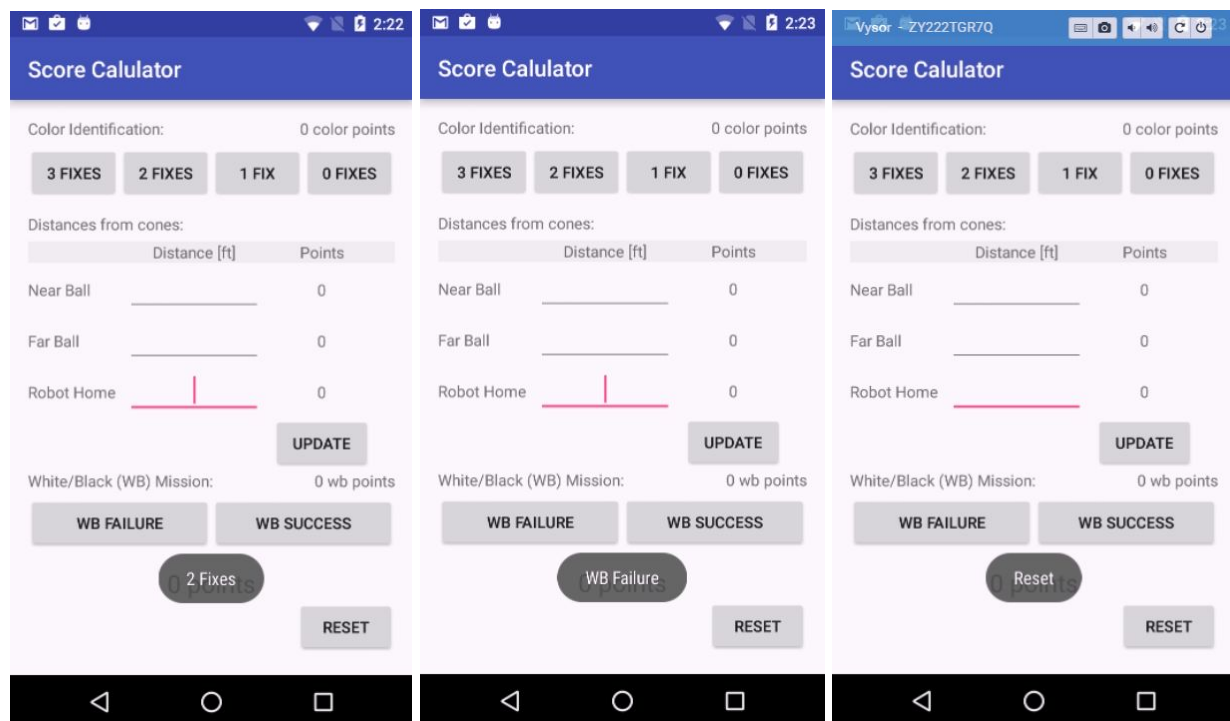
If you want to try a GridLayout instead of a TableLayout here is some info:

<https://web.archive.org/web/20140630024448/http://blog.stylingandroid.com/archives/669>

<http://android-developers.blogspot.com/2011/11/new-layout-widgets-space-and-gridlayout.html>

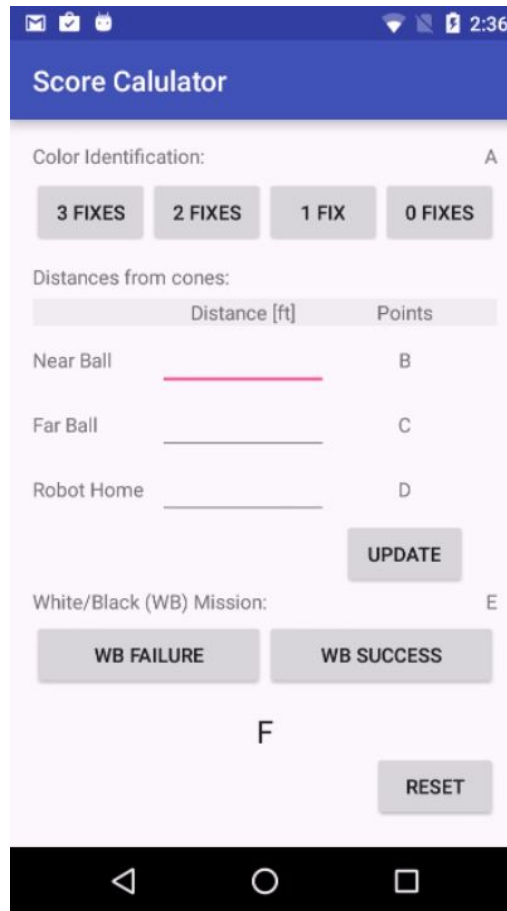
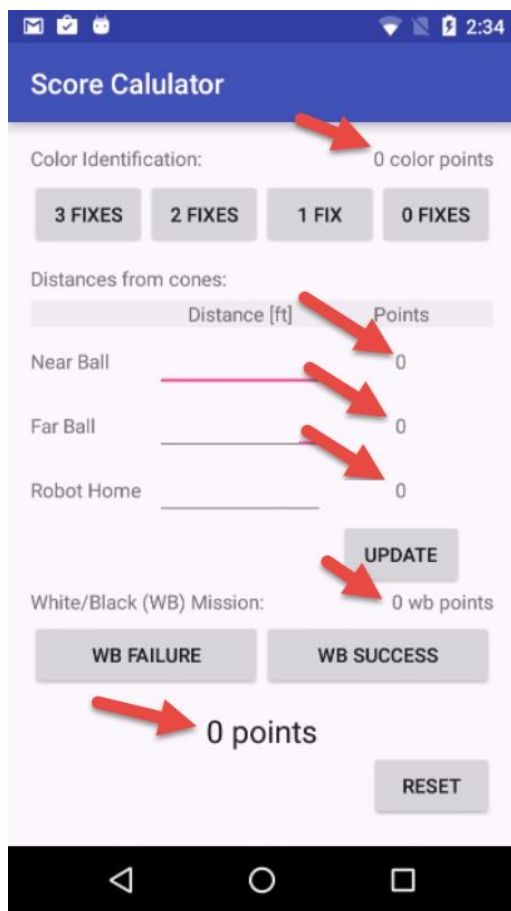
Step #2: Connect your buttons

After my layout looked ok on my device, I setup each button to an onClick listener and made each one display a Toast message. I tested each button individually before writing any real code.



Step #3: Connect your text views

There are a total of 6 TextViews that need to change. I made a connection to each one. Then I did a setText in the Java code just to make sure I could programmatically change them. For my quick test I just made them become the strings A, B, C, D, E, and F.



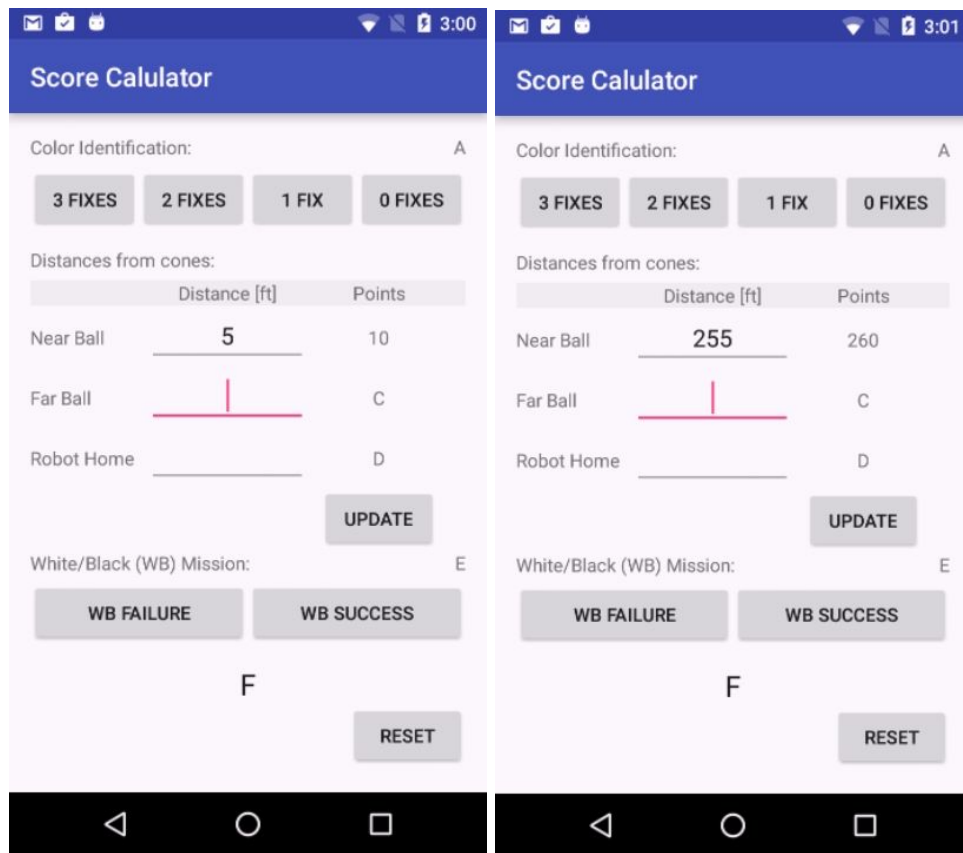
Your UI choices might be VERY different. Just sharing a development process that could be used.

Step #4: Converting the EditText values to numbers

Next I needed connections to my 3 EditText objects, I called mine:

- `private EditText mNearBallDistanceEditText;`
- `private EditText mFarBallDistanceEditText;`
- `private EditText mRobotHomeDistanceEditText;`

Then at some point in the code I'll need to get their values as an int (so that I can use < if statements). I started with a simple proof of concept to make sure I could get the value, convert it to an int, do math with it, then write it back to a TextView (as a String). My silly proof of concept test was to **take the near ball distance** and **add 5 to it**, then **display it in the points text view**.



Doing that tasks took either some knowledge of Java or knowing how to search the internet for help. If you know Java you might know that `Integer.parseInt()` is one option to convert a String to an int. So I wrote some code like this...

```
int nearBallDistance = Integer.parseInt(mNearBallDistanceEditText.getText().toString());
nearBallDistance += 5;
mNearBallPointsTextView.setText("" + nearBallDistance);
```

Notice how the string goes to an int then back to a String later. `"" + an int` is a cute little way to do that conversion to a String in Java. If you didn't know that you might search the internet for "android edittext convert to int" and the first hit would be a StackOverflow...

<http://stackoverflow.com/questions/15037465/converting-edittext-to-int-android>

Here is what you'd find there...



I'm very sleepy and tired right now but wouldn't this work?:

3

```
EditText et = (EditText)findViewById(R.id.editText1);
String sTextFromET = et.getText().toString();
int nIntFromET = new Integer(sTextFromET).intValue();
```

OR

```
try
{
    int nIntFromET = Integer.parseInt(sTextFromET);
}
catch (NumberFormatException e)
{
    // handle the exception
}
```

Their solution is even more robust than mine. It catches the exception if the EditText is not a valid number (an empty EditText is the main concern). I'll add that to my code and I'll set the value to some big distance away if an exception occurs (btw setting it to 0 would be unfortunate, as 0 ft away would get a perfect score).

```
int nearBallDistance;
try {
    nearBallDistance = Integer.parseInt(mNearBallDistanceEditText.getText().toString());
} catch (NumberFormatException e) {
    nearBallDistance = 100; // Some big number.
}
int farBallDistance;
try {
    farBallDistance = Integer.parseInt(mFarBallDistanceEditText.getText().toString());
} catch (NumberFormatException e) {
    farBallDistance = 100;
}
int robotHomeDistance;
try {
    robotHomeDistance = Integer.parseInt(mRobotHomeDistanceEditText.getText().toString());
} catch (NumberFormatException e) {
    robotHomeDistance = 100;
}
```

I promise you that StackOverflow is your friend. Use it. :) The try/catch block is a Java thing that I could talk more about. This code is nice because it'll handle empty text boxes smoothly when the distance isn't yet known.

Step #5: Write some code

Finally I implemented the actual code. I made a few variables to track the scores:

- `mColorPoints`
- `mNearBallPoints`
- `mFarBallPoints`
- `mRobotHomePoints`
- `mWbPoints`

And I made their values get set correctly at appropriate times. Finally I made all my buttons call an `updateView` function that I wrote to make sure the view matched the values. That function updates all 6 of my TextViews including the total score Text View.

Happy Coding!

This assignment is individual, but feel free to ask others for help while developing it.