Lab – Graphics, Animation, Touch, and Multi-Media

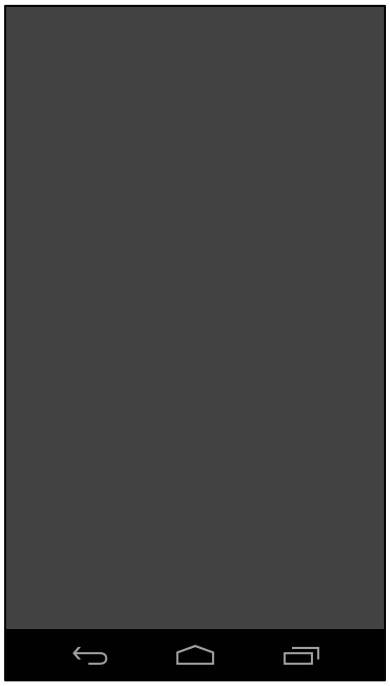
Graphics, Animation, Touch, and Multimedia

Objectives:

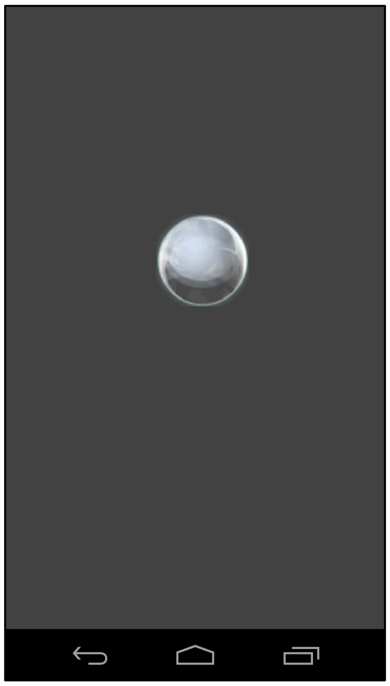
This week's lab is aimed at getting a better understanding of Graphics, Animation, Touch, and Multimedia. Upon completion of this lab you should understand how to display and animate images within your application, have the app respond to touch input, and have it play simple sound effects.

Exercise:

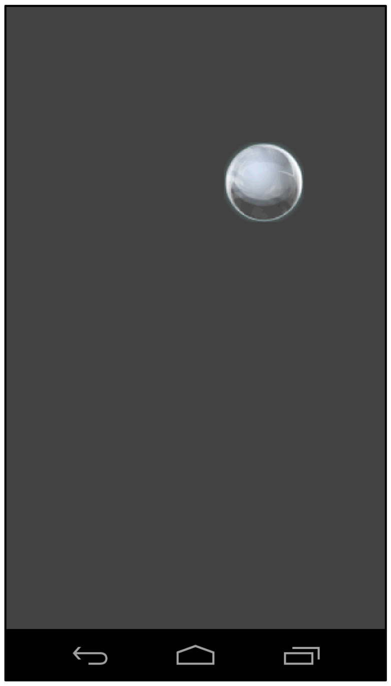
In this part, you will create an application that displays, animates and manipulates Bubbles. The application's UI will have a main display area that is initially empty as shown below.



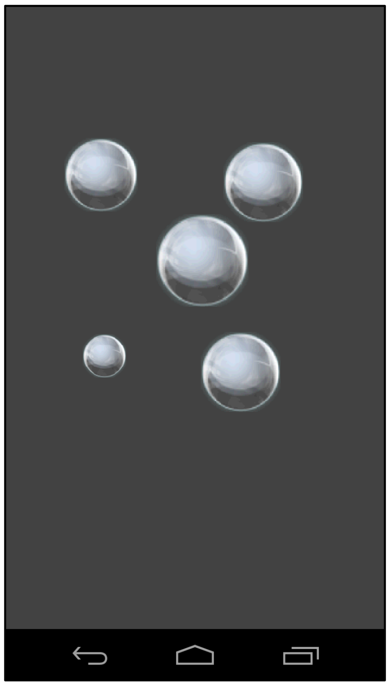
As shown below, when the user touches this empty screen, one new bubble should appear on the display. The bubble should then begin to move around the screen. The Bubble's size, direction and speed should be randomized within limits explained in source code skeleton.



Assuming the Bubble’s direction is up and to the right, the Bubble shown above might be in the following location after a couple seconds.



As the user continues to touch empty spaces on the screen, more bubbles should be added.



If the user presses the screen at a location already occupied by a bubble, then the bubble should "pop." That is, it should be removed from the screen and a bubble popping sound should be played. The sound file is in the skeleton code in the /res/raw/ bubble_pop.wav file.

In addition, if the user executes a "fling" gesture starting at a location already occupied on the screen, then the application should change the bubbles current direction and speed to match the direction and speed of the fling gesture.

Tips:

Each time your app adds a new Bubble, it should create a new BubbleView. The BubbleView class handles drawing and moving the Bubble, and also initiates removing the bubble from the main View and playing the popping sound.

New BubbleViews must be added to the app's main view, called mFrame, otherwise they won't be visible.

You also need to keep track the Bubbles as they move off the screen. Once a BubbleView moves completely off screen, its movement calculations should stop and it should be removed from the app's main View.

When a BubbleView is created, its size, movement direction and speed, and rotation speed are randomized, within bounds described in the skeleton code. We have added some special modes to facilitate testing, so we don't expressly test for this behavior in your app.

When a BubbleView changes position, you must notify the system that it has changed, otherwise it will not be redrawn.

Implementation Notes:

1. Download the application skeleton files and import them into your IDE.

2. Look for comments containing the string "TODO" and follow the associated instructions.

Testing:

The test cases for this Lab are in the GraphicsLabTest project, which is located in GraphicsLab/SourceFiles/TestCases/GraphicsLabTest.zip file. You can run the test cases either all at once, by right-clicking the project folder and then selecting Run As > Android JUnit Test, or one at a time, by right-clicking on an individual test case class (e.g., BubbleActivityFling.java) and then continuing as before. The test classes are Robotium test cases. You will eventually have to run each test case, one at a time, capture log output, and submit the output to Coursera.

These test cases are designed to drive your app through a set of steps, passing the test case is not a guarantee that your submission will be accepted. The BubbleActivityFling test case emits exactly 3 Log messages. The BubbleActivityFloatOffScreen test case emits exactly 2 Log messages. The BubbleActivityMultiple test case emits exactly 2 Log messages. The BubbleActivityPop test case emits exactly 3 Log messages.

Warnings:

1. These test cases have been tested on a Galaxy Nexus AVD emulator with API level 18. To limit configuration problems, you should test your app against a similar AVD.

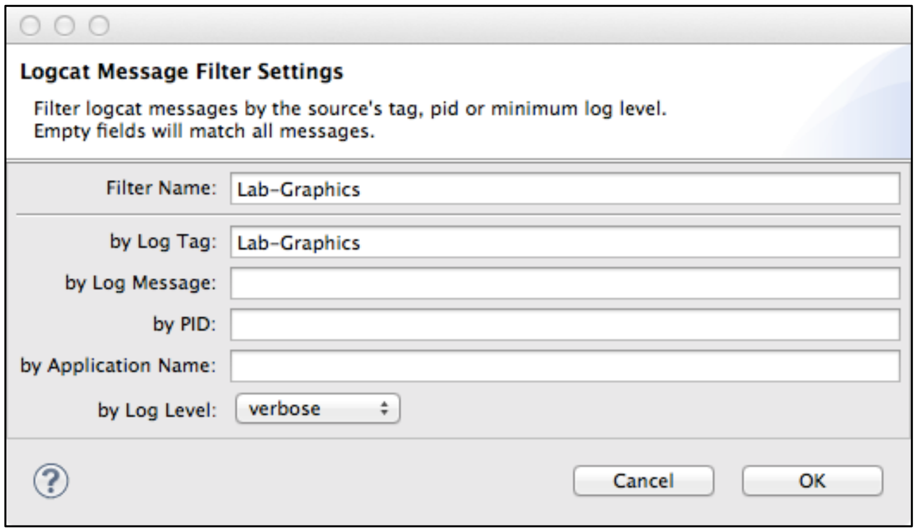
2. These test cases assume a display screen of at least 550x550 pixels.

3. On some emulators "fling" gestures will sometime report erroneous fling velocities or fail to recognize the fling. Keep an eye out for this erroneous behavior when you're running the BubbleActivityFling test case. The test case should 1) create a new BubbleView, 2) recognize a fling gesture, and 3) remove the BubbleView from mFrame when the BubbleView leaves the screen.

Once you've passed all the test cases, submit your log information to the Coursera system for grading.

Tip: Saving a LogCat filter

1. In the LogCat View, press the green "+" sign to "add a new LogCat filter."
2. A dialog box will pop up. Type in the information shown in the screenshot below
3. A saved filter called, "Lab-Graphics" will appear in the LogCat View



Logcat Message Filter Settings

Filter logcat messages by the source's tag, pid or minimum log level.
Empty fields will match all messages.

Filter Name:	Lab-Graphics
by Log Tag:	Lab-Graphics
by Log Message:	
by PID:	
by Application Name:	
by Log Level:	verbose

Tip: Running your test cases and capturing your LogCat output for submission.

1. For each test case, clear the LogCat console by clicking on the "Clear Log" button (the button with the red x in the upper-right of the LogCat View).
2. Then right-click on an individual test case file in the Project Explorer. Run As > Android JUnit Test.
3. When the test case finishes, if it's not already selected, click on the "Lab-Graphics" filter in the left-hand pane of the LogCat View.
4. Select everything within the LogCat View (Ctrl-A or Cmd-A) and press the "Export Selected Items To Text File" button (floppy disk icon) at the top-right of the LogCat View.
5. Submit this file to the Coursera system.