

# Lab – Graphics and Animation

---

## *Graphics and Animation*

### Objectives:

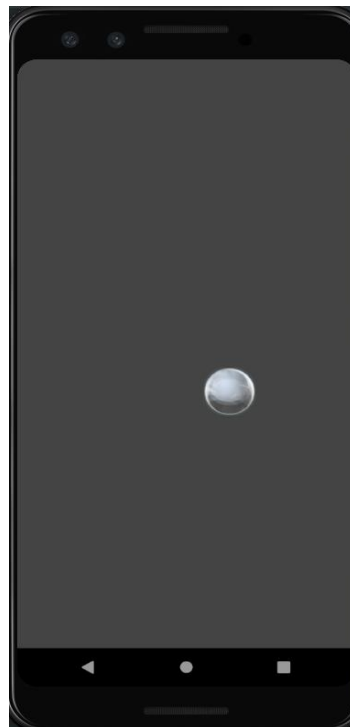
This week's lab is aimed at getting a better understanding of Graphics, Animation, and Multimedia. Upon completion of this lab you should understand how to display and animate images within your application and have it play simple sound effects.

### Exercise:

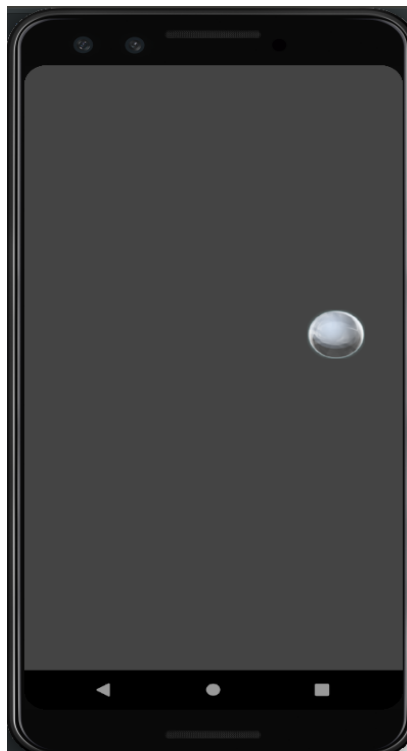
In this part, you will create an application that displays and animates Bubbles. The application's UI will have a main display area that is initially empty as shown below.



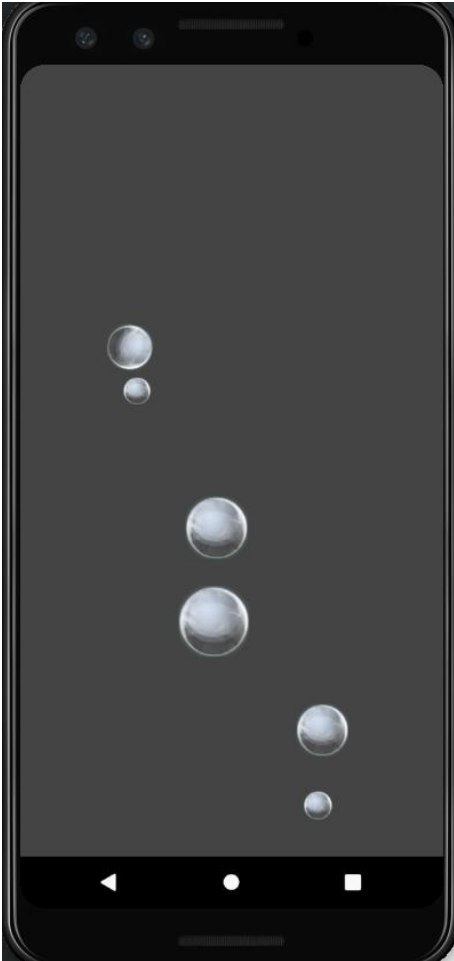
As shown below, when the user presses the back button, the options menu opens. The user then clicks on the “Add a Bubble” menu item and one new bubble should appear on the display. The bubble should then begin to move around the screen. The Bubble's size, direction and speed should be randomized within 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 press the “Add a Bubble” menu item, more bubbles should be added.



If the user instead presses the “Delete a Bubble” menu item, then the newest bubble that’s still visible 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.

[This screencast shows the app in operation.](#)

### 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 of 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. The app will show the menu if the user hits the back button. Click "Quit" in the menu to exit the application.

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

## Implementation Notes:

1. Clone 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 Lab9\_GraphicsLab project. You can run the test cases either all at once, by right clicking the test package and then selecting Run 'Tests in 'course.lab...', or one at a time, by right-clicking on an individual test case class (e.g., BubbleActivityPop.kt) and then continuing as before. The test classes are Robotium test cases.

## Warnings:

1. These test cases have been tested on a Pixel 3 AVD emulator with API level 29. 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.

## Submission

To submit your work, you will need to commit your solution to your repo on GitLab by running the following command: `git push origin master`.

Note: if you have not already pushed this branch to your repo on GitLab you will need to make a slight modification for this first time and run this instead: `git push -u origin master`. This sets up tracking between your local branch and a branch with the same name on your repo in GitLab.