

# Laboratory – Sensors

---

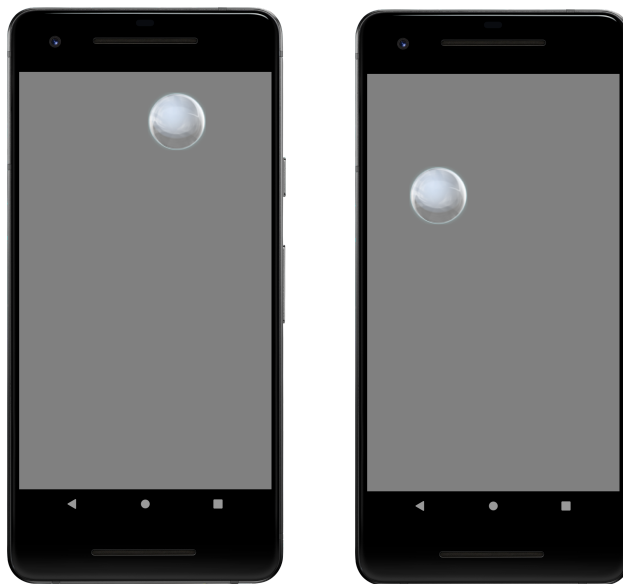
*Learn about capturing and using sensor readings as input to your apps*

## Objectives

Familiarize yourself with capturing accelerometer input and then using that input to guide the animation of a View on the display. Create an application that displays the image of a bubble (or ball) that moves in a given direction on the display. When the user tilts the device, the bubble image will move in a direction and velocity proportional to the device's movement. If the bubble's trajectory leads it to the edge of the display, the image's motion stops in that direction. Once you've completed this Lab you should understand how to access the accelerometer, and how to receive accelerometer input and use it within your app.

## LabSensors

This lab involves an app called LabSensors. When it runs, the app displays a user interface like that shown below. The trajectory of the bubble will be determined by the orientation of the device with respect to the force of gravity. Over time, the bubble image may drift across the display until it reaches the edge of the display. In this case, the bubble's movement in that direction will be blocked.



This application includes several components. It includes a ViewModel class called “BubblePositionViewModel,” which manages sensor input, simulates the movement of the bubble in accordance with the sensor input, and publishes the current position of the bubble as LiveData. The app also include an Activity called “BubbleActivity.” BubbleActivity is responsible for displaying the bubble at its current position. It also provides the BubblePositionViewModel with information about

the size of the display and a reference to the SensorManager, and tells the BubblePositionViewModel when to start its simulation. BubblePositionViewModel observes the lifecycle of the BubbleActivity, so that it can pause the simulation when the BubbleActivity is not in the foreground.

See the screencast, LabSensors.mp4, that's included in the Lab directory.

## Testing

There is one test case with several evaluation points. Each evaluation occurs at a step labelled “evaluation point”

The first test case operates as follows:

1. Start the AVD, click on the Extended Controls > Virtual Sensors > Device Pose tab and set the Z-Rot to 0.0, X-Rot to -90.0, and Y-Rot to 0.0
2. Start the app in portrait mode.
3. Check that the UI shows the bubble (evaluation point 1).
4. Check that the bubble is not moving (evaluation point 2).
5. Go back to the Extended Controls and change the X-Rot to approach -10. Check that the UI shows the bubble moving straight down with increasing velocity (evaluation point 3)
6. Go back to the Extended Controls and change the Y-Rot to approach 60.0. Check that the UI shows the bubble moving right with increasing velocity (evaluation point 4)
7. Go back to the Extended Controls and change any orientation values so that the bubble ends up in the top, right corner of the display. Check that the bubble appears in the top, right corner (evaluation point 5)

After completing your solution, you will record a screencast while performing the manual test. Afterward, you will submit your code and the screencast via git. You can record a screencast using services available in the Logcat console. See:  
<https://developer.android.com/studio/debug/am-video>

## Submission

When you are ready just commit your solution to your repo on GitLab by running the following commands:

```
% git add path/to/changed/files
% git commit -m "completed Lab11_SensorsLab"
% git push origin main
```

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 main

This sets up tracking between your local branch and a branch with the same name on your repo in GitLab.

## Some Implementation Notes

We are providing template code and layout resources for this application. Only modify the areas marked with the word TODO.

We have done our testing on an emulator using a Pixel 5 AVD with API level 31. To limit configuration problems, you should test your app against a similar AVD.