Android

Fragments, Sensors

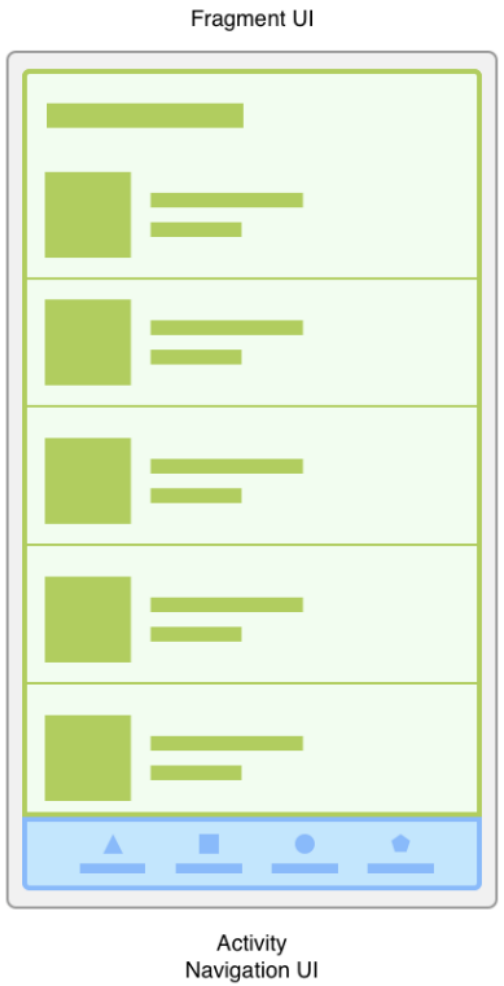
Contents

PDM
sayHello();

- Fragments
  - Mini-UI part
  - Lifecycle
  - How to?
- Sensors
  - Hardware or software
  - How to?

Fragment

- Portion of the User Interface
- Modularity and reusability
- Exists whitin an Activity
- Allow dynamic UI changes



Fragment

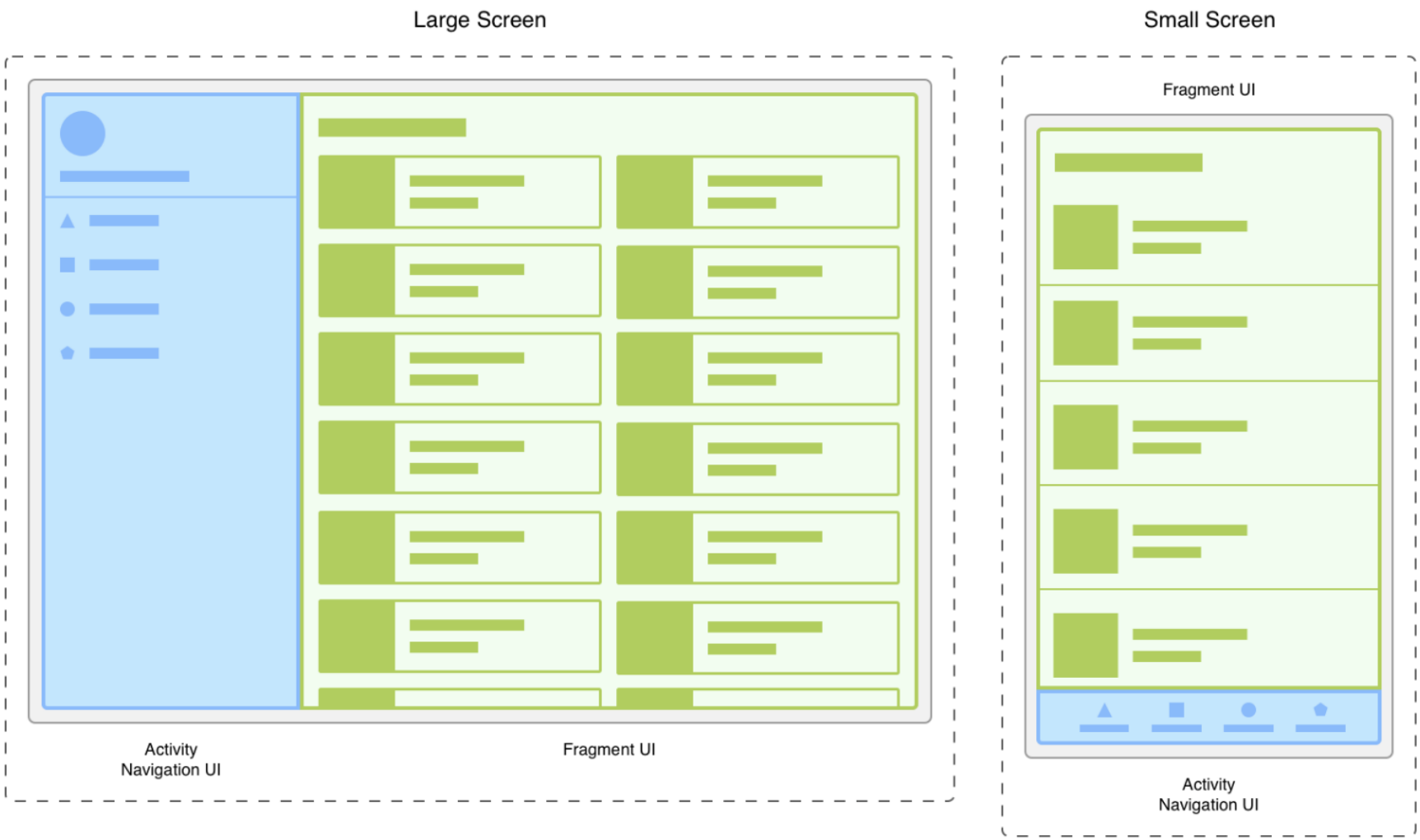
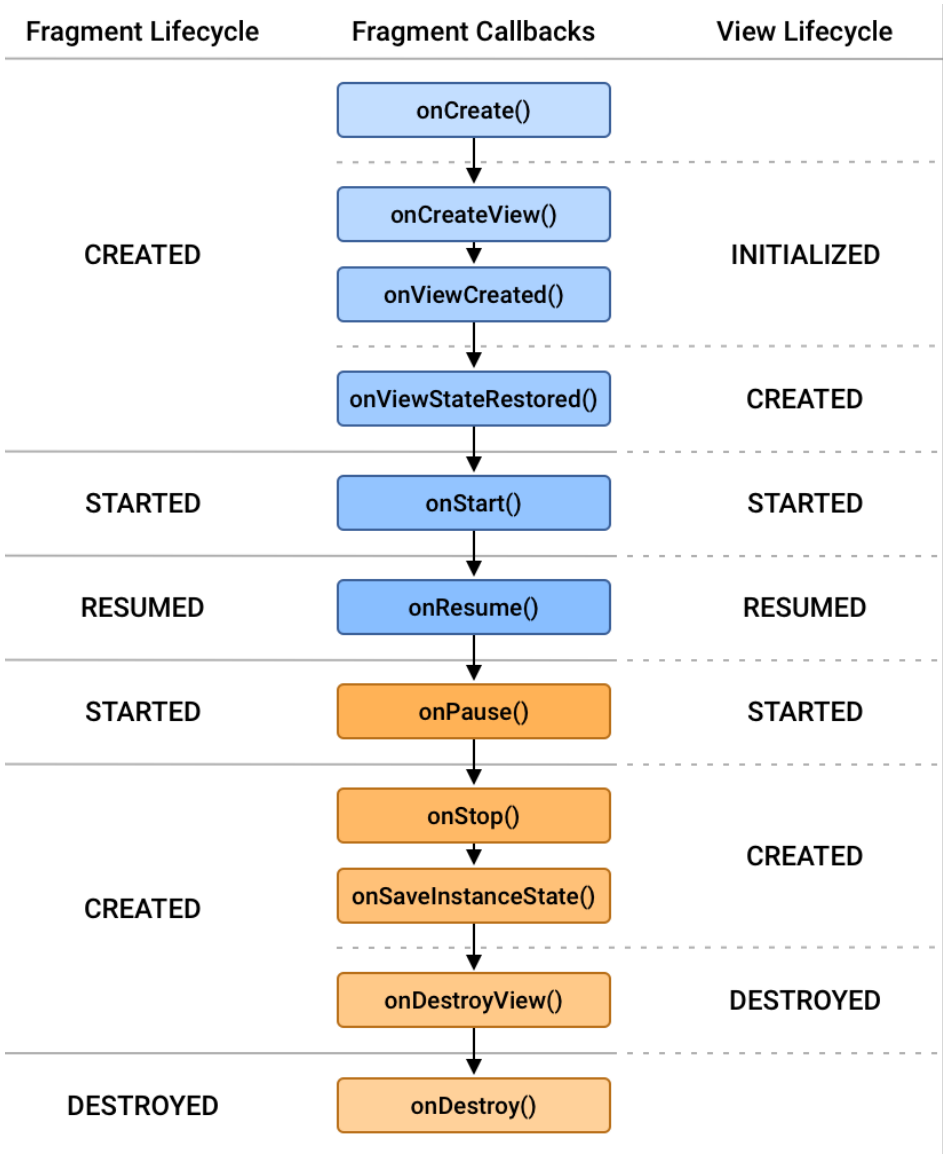


Figure 1. *Two versions of the same screen on different screen sizes. On the left, a large screen contains a navigation drawer that is controlled by the activity and a grid list that is controlled by the fragment. On the right, a small screen contains a bottom navigation bar that is controlled by the activity and a linear list that is controlled by the fragment.*

Fragment lifecycle



How to use a fragment

Step 1. Create a Fragment class

```
class ExampleFragment : Fragment(R.layout.example_fragment)
```

- Extend the Fragment class
- Add the layout in the constructor of the parent class
- Implement the needed lifecycle methods

How to use a fragment

Step 2. Add the Fragment to an activity

Statically

```
<!-- res/layout/example_activity.xml -->
<androidx.fragment.app.FragmentContainerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:name="com.example.ExampleFragment" />
```

How to use a fragment

Step 2. Add the Fragment to an activity

Programatically

```
<!-- res/layout/example_activity.xml -->
<androidx.fragment.app.FragmentContainerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

```
class ExampleActivity : AppCompatActivity(R.layout.example_activity) {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        if (savedInstanceState == null) {
            supportFragmentManager.commit {
                setReorderingAllowed(true)
                add<ExampleFragment>(R.id.fragment_container_view)
            }
        }
    }
}
```

Sensors

- Can interact with them through the Android framework
- Hardware or software

Sensor	Type	Description	Common Uses
`TYPE_ACCELEROMETER`	Hardware	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), including the force of gravity.	Motion detection (shake, tilt, etc.).
`TYPE_AMBIENT_TEMPERATURE`	Hardware	Measures the ambient room temperature in degrees Celsius (°C). See note below.	Monitoring air temperatures.
`TYPE_GRAVITY`	Software or Hardware	Measures the force of gravity in m/s^2 that is applied to a device on all three physical axes (x, y, z).	Motion detection (shake, tilt, etc.).
`TYPE_GYROSCOPE`	Hardware	Measures a device's rate of rotation in rad/s around each of the three physical axes (x, y, and z).	Rotation detection (spin, turn, etc.).
`TYPE_LIGHT`	Hardware	Measures the ambient light level (illumination) in lx.	Controlling screen brightness.
`TYPE_LINEAR_ACCELERATION`	Software or Hardware	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), excluding the force of gravity.	Monitoring acceleration along a single axis.
`TYPE_MAGNETIC_FIELD`	Hardware	Measures the ambient geomagnetic field for all three physical axes (x, y, z) in µT.	Creating a compass.

Sensor framework

- Sensor Manager
  - Acces existing sensors
  - Register and unregister sensor events listeners
  - Calibrate sensors
- Sensor
  - Instance of a specific sensor
  - Acces methods on a sensor (get its capabilities)
- SensorEvent
  - Used to provide informations about a sensor event
  - Is sent to users through listeners
  - Containts
    - the raw sensor data,
    - the type of sensor that generated the event
    - the accuracy of the data
    - the timestamp for the event
- SensorEventListener
  - Interface that allows the creation of callbacks
  - Is used when registering for events with the SensorManager
  - Is called when sensor values change

How to: interact with a sensor

Step 1: Get the SensorManager

```
private lateinit var sensorManager: SensorManager
...
sensorManager = getSystemService(Context.SENSOR_SERVICE) as SensorManager
```

Step 2: Get your sensor

```
private var accelerometer: Sensor? = null

// Get the accelerometer sensor
accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
```

Note: you should also check if your sensor exists :)

How to: interact with a sensor

Step 3: Implement the callbacks of the listener

```
// Define the SensorEventListener as a separate variable
private val accelerometerListener = object : SensorEventListener {
    override fun onSensorChanged(event: SensorEvent?) {
        if (event?.sensor?.type == Sensor.TYPE_ACCELEROMETER) {
            val x = event.values[0]
            val y = event.values[1]
            val z = event.values[2]

            val acceleration = Math.sqrt((x * x + y * y + z * z).toDouble()).toFloat()
            val currentTime = System.currentTimeMillis()

            if (acceleration > 12) { // Adjust threshold for shake detection
                if (currentTime - lastShakeTime > 1000) { // Avoid multiple triggers
                    lastShakeTime = currentTime
                    Toast.makeText(this@MainActivity, "Device shaken!", Toast.LENGTH_SHORT).show()
                }
            }
        }
    }

    override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {
        // Handle changes in sensor accuracy if needed
    }
}
```

How to: interact with a sensor

Step 4: Register the listener to receive sensor events

```
// Register the listener when the activity is resumed
accelerometer?.let {
        sensorManager.registerListener(accelerometerListener, it, SensorManager.SENSOR_DELAY_NORMAL)
    }
```

Step 5: Unregister the listener in the appropriate lifecycle callback

```
// Unregister the listener when the activity is paused
sensorManager.unregisterListener(accelerometerListener)
```

Full example


```
class SensorActivity : Activity(), SensorEventListener {
    private lateinit var sensorManager: SensorManager
    private var mLight: Sensor? = null

    public override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main)

        sensorManager = getSystemService(Context.SENSOR_SERVICE) as SensorManager
        mLight = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT)
    }

    override fun onAccuracyChanged(sensor: Sensor, accuracy: Int) {
        // Do something here if sensor accuracy changes.
    }

    override fun onSensorChanged(event: SensorEvent) {
        // The light sensor returns a single value.
        // Many sensors return 3 values, one for each axis.
        val lux = event.values[0]
        // Do something with this sensor value.
    }

    override fun onResume() {
        super.onResume()
        mLight?.also { light ->
            sensorManager.registerListener(this, light, SensorManager.SENSOR_DELAY_NORMAL)
        }
    }

    override fun onPause() {
        super.onPause()
        sensorManager.unregisterListener(this)
    }
}
```

Questions

PDM
sayHello();