Sensors

Marco Ronchetti
Università degli Studi di Trento

Sensor categories

Motion sensors

- measure acceleration forces and rotational forces along three axes. This category includes accelerometers, gravity sensors, gyroscopes.

Environmental sensors

- measure various environmental parameters, such as ambient air temperature and pressure, illumination, and humidity. This category includes barometers, photometers, and thermometers.

Position sensors

- measure the physical position of a device. This category includes orientation sensors and magnetometers.

Basic code for managing sensors

```
public class SensorActivity extends Activity, implements SensorEventListener {
   private final SensorManager sm;
   private final Sensor sAcc;
   public SensorActivity() {
      sm= (SensorManager)getSystemService(SENSOR_SERVICE);
      sAcc= sm.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
   }
   protected void onPause() {
      super.onPause();
      sm.unregisterListener(this);
   }
   protected void onResume() {
      super.onResume();
      sm.registerListener(this, sAcc, SensorManager.SENSOR_DELAY_NORMAL);
   }
   public void onAccuracyChanged(Sensor sensor, int accuracy) { }
   public void onSensorChanged(SensorEvent event) { }
}
```

SensorManager

SensorManager sm=Context.getSystemService(SENSOR_SERVICE);

List<Sensor> getSensorList(int type)

- get the list of available sensors of a certain type. Sensor

Sensor getDefaultSensor(int type)

- Use this method to get the default sensor for a given type

void registerListener(SensorEventListener listener, Sensor sensor, int rate)

- Registers a SensorEventListener for the given sensor.

void unregisterListener(SensorEventListener listener, Sensor sensor)

- Unregisters a listener for the sensors with which it is registered.

void unregisterListener(SensorEventListener listener)

- Unregisters a listener for all sensors.

- Some methods for transforming data (Vector to matrix representation etc.)

Sensor types

int constants of the Sensor class describing sensor types:

TYPE_ACCELEROMETER

TYPE_ALL A constant describing all sensor types.

TYPE_AMBIENT_TEMPERATURE

TYPE_GRAVITY

TYPE_GYROSCOPE

TYPE_LIGHT

TYPE_LINEAR_ACCELERATION

TYPE_MAGNETIC_FIELD

TYPE_PRESSURE

TYPE_PROXIMITY

TYPE_RELATIVE_HUMIDITY

TYPE_ROTATION_VECTOR

TYPE_STEP_COUNTER

TYPE_HEART_BEAT

Accelerometer

*"Sensor's values are in **meters/second^2** units. A sensor measures the acceleration applied to the device. For this reason, when the device is sitting on a table (and obviously not accelerating), the accelerometer reads a magnitude of **g** = 9.81 m/s^2. Similarly, when the device is in free-fall and therefore dangerously accelerating towards to ground at 9.81 m/s^2, its accelerometer reads a magnitude of 0 m/s^2."* (Android Developers – sensors)

Orientation sensor

"A compass is a navigational instrument for determining ***direction*** *relative to the Earth's* ***magnetic poles****. It consists of a magnetized pointer (usually marked on the North end) free to align itself with* ***Earth's magnetic field****."* (Compass EN Wiki)

In Android's terminology it is called **Orientation sensor**.

Gyroscope

*"A gyroscope is an instrument consisting of a rapidly **spinning wheel** so mounted as to use the tendency of such a wheel to maintain a fixed position in space, and to resist any force which tries to change it. The way it will move if a twisting force is applied depends on the extent and orientation of the force and the way the gyroscope is mounted. A free vertically spinning gyroscope remains vertical as the carrying vehicle tilts, so providing an artificial horizon. A horizontal gyroscope will maintain a certain bearing, and therefore indicate a vessel's heading as it turns. Modern gyroscopes (including those built-in in smartphones) no longer have a spinning wheel."* (Gyroscope Cambridge Encyclopedia)

*"All values are in **radians/second** and measure the rate of rotation around the **X**, **Y** and **Z** axis. The coordinate system is the same as is used for the acceleration sensor."* (Android Developers – sensors) Rotation is positive in the **counter-clockwise** direction.

Sensor class

float getMaximumRange()

- maximum range of the sensor in the sensor's unit.

int getMinDelay()

- minimum delay allowed between two events in microsecond or zero if this sensor only returns a value when the data it's measuring changes

String getName()

float getPower()

- the power in mA used by this sensor while in use

float getResolution()

- resolution of the sensor in the sensor's unit.

int getType()

String getVendor()

int getVersion()

```
SensorManager sm= (SensorManager)getSystemService(SENSOR_SERVICE);
List<Sensor> sensorList = sm.getSensorList(Sensor.TYPE_ALL);
StringBuilder sensorString = new StringBuilder("Sensors:\n");
for(int i=0; i<sensorList.size(); i++) {
    sensorString.append(sensorList.get(i).getName()).append(", \n");
}
```

HTC EVO 4G
BMA150 3-axis Accelerometer
AK8973 3-axis Magnetic field sensor
AK8973 Orientation sensor
CM3602 Proximity sensor
CM3602 Light sensor

Samsung Nexus-S
KR3DM 3-axis Accelerometer
AK8973 3-axis Magnetic field sensor
AK8973 Orientation sensor
GP2A Light sensor
GP2A Proximity sensor
K3G Gyroscope sensor
Gravity Sensor
Linear Acceleration Sensor
Rotation Vector Sensor

Interface SensorEventListener

abstract void onAccuracyChanged(Sensor sensor, int accuracy)

- Called when the accuracy of a sensor has changed.

abstract void onSensorChanged(SensorEvent event)

- Called when sensor values have changed.

Code examples

- http://www.vogella.com/articles/AndroidSensor/article.html accelerometer and compass examples

- http://developer.android.com/guide/topics/sensors/sensors_overview.html and following pages

Sensors limitation - 1

From Jim Steele

Using available sensors in the Android platform: current limitations and expected improvements

Comparing the sensors on these two phones demonstrates the **sensor fragmentation** now found in Android:

1) **Non-standard sensor availability**: The Nexus-S has a gyroscope (from ST Micro), but the EVO does not. In fact, most Android devices do not have a gyroscope. There is no standard availability of sensors across devices.

2) **Non-standard sensor capability**: The BMA150 is a Bosch Sensortec 10-bit accelerometer, and the KR3DM is a ST Micro 12-bit accelerometer (using a special part number). In fact, there is no standard capability requirement for sensors across devices to ensure consistent resolution, noise floor, or update rate.

Sensors limitation - 2

3) **Sensors not fully specified**: The AK8973 is an AKM magnetometer, which is only 8-bits. Analyzing this data stream shows it is low-pass filtered. This fact is not published on the phone or even the sensor datasheet. Many sensors have characteristics not specified such as bias changes, non-uniform gain, and skew (coupling between measurement axes). **Algorithms that use sensors without knowing these extra characteristics may produce incorrect information**.

4) **Broken virtual sensors**: The AKM sensor driver abstracts out an orientation virtual sensor which is derived from the combination of two sensors: the accelerometer and magnetometer. However, support for this virtual sensor was dropped early on, so the TYPE_ORIENTATION sensor is deprecated and the method **SensorManager.getOrientation()** should be used instead..

The sensor differences between just these two phones is substantial. So when a developer is faced with **writing apps utilizing sensors across as many devices as possible, it is a daunting task**.

Furthermore, **the Android platform is not optimized for real-time sensor data acquisition**.

What can you do with accelerometer and gyroscope?

http://www.starlino.com/imu_guide.html

A Guide To using IMU (Accelerometer and Gyroscope Devices) in Embedded Applications.

"This guide is intended to everyone interested in inertial MEMS (Micro-Electro-Mechanical Systems) sensors, in particular Accelerometers and Gyroscopes as well as combination IMU devices (Inertial Measurement Unit)."

- what does an accelerometer measure
- what does a gyroscope (aka gyro) measure
- how to convert analog-to-digital (ADC) readings that you get from these sensor to physical units (those would be g for accelerometer, deg/s for gyroscope)
- how to combine accelerometer and gyroscope readings in order to obtain accurate information about the inclination of your device relative to the ground plane

http://www.starlino.com/dcm_tutorial.html

DCM Tutorial – An Introduction to Orientation Kinematics

SensorSimulator

The Android Emulator (v25.2.2 and higher), launched with Android Studio 2.2 can simulate the following sensor types:

TYPE_ACCELEROMETER
TYPE_AMBIENT_TEMPERATURE
TYPE_GRAVITY
TYPE_GYROSCOPE
TYPE_LIGHT
TYPE_LINEAR_ACCELERATION
TYPE_MAGNETIC_FIELD
TYPE_ORIENTATION
TYPE_PRESSURE
TYPE_PROXIMITY
TYPE_RELATIVE_HUMIDITY
TYPE_ROTATION_VECTOR
TYPE_TEMPERATURE

As defined here: https://developer.android.com/guide/topics/sensors/sensors_overview.html

Sensori esterni

Posiziona il mouse sopra per ingrandire

Popolare OBD2 Interfaccia Diagnostica ELM327 Bluetooth Più Recente V2.1 Lavori Multi-Brand Auto A Benzina ELM 327 Supporta Multi-Protocolli

Guarda titolo originale in inglese

4.3 (4 voti) | 9 ordini

Prezzo: **US $3.17** / parte

Prezzo all'ingrosso:

Spedizione: US $1.00 a Italy tramite China Post Ordinary Small Packet Plus

Consegna: 44-59 giorni

Quantità: – 1 + parte (968 parti disponibile)

Prezzo totale: US $4.17

Acquista ora | Aggiungi al carrello

Aggiungi alla Lista dei Dedideri (31 Aggiunti)

Resi: Resi accettati se il prodotto non corrisponde alla descrizione, le spese di spedizione del reso sono a carico dell'acquirente; oppure è possibile conservare il prodotto e concordare il rimborso con il venditore
Visualizza dettagli

Garanzie del venditore: Consegna puntuale
60 giorni

Pagamento: VISA MasterCard Maestro PayPal WESTERN UNION Bank Transfer

Visualizzza di più

Sensori esterni

S9 Schermo a colori Blood Pressure Ossigeno Heart Rate Monitor Smartwatch sportivo per Android / iOS - NERO

5 (1 Recensioni dei clienti) | Fai una domanda

Prezzo: €20.07

Opzioni di magazzino: China

Quantità 1

Colore: Nero | Rosso | viola | Green | Blu | Arancione

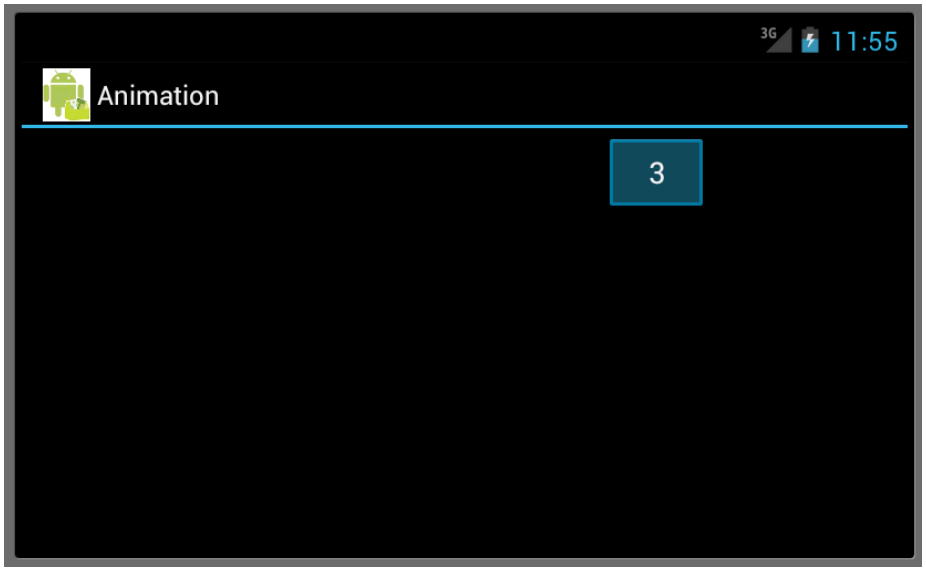
Spese di spedizione €0.82 a Italy Via Posta Tracciabile

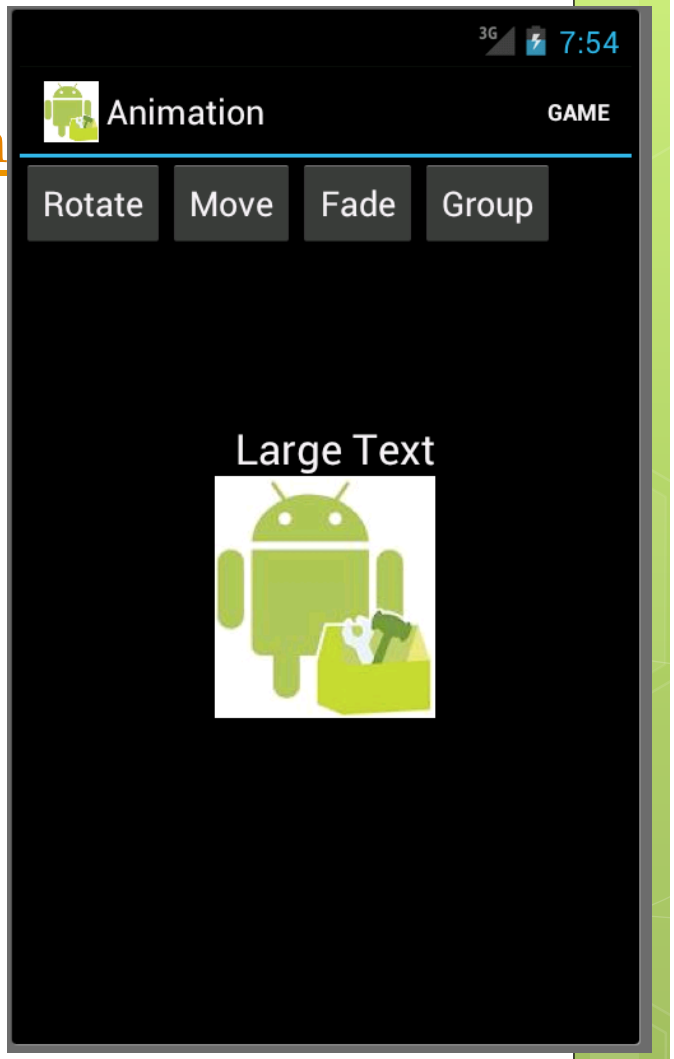
Basic Animation

Marco Ronchetti
Università degli Studi di Trento

An example

Modified from an example by Vogella





Download the source from:

http://latemar.science.unitn.it/segue_userFiles/2012Mobile/Animation.zip

3 ways to do animation

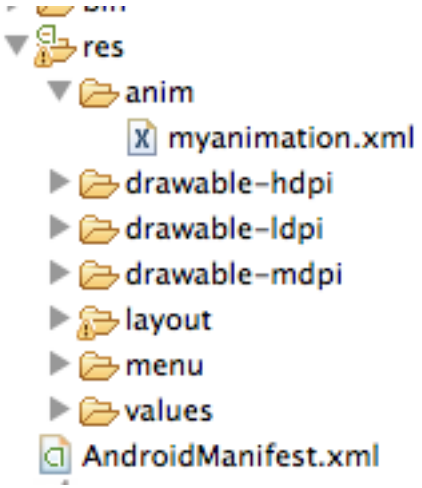
- Property Animation
  - lets you animate properties of any object, including ones that are not rendered to the screen. The system is extensible and lets you animate properties of custom types as well.
- View Animation
  - An older system, can only be used for Views. It is relatively easy to setup and offers enough capabilities to meet many application's needs.
- Drawable Animation
  - involves displaying Drawable resources one after another, like a roll of film.
  - useful if you want to animate things that are easier to represent with Drawable resources, such as a progression of bitmaps

See https://developer.android.com/training/animation/overview

Example 1 – loading animation from xml

```
ImageView aniView = (ImageView) findViewById(R.id.imageView1);
Animation animation1 = AnimationUtils.loadAnimation
        (this,R.anim.myanimation);
aniView.startAnimation(animation1);
```

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
   android:shareInterpolator="true">
   <rotate android:fromDegrees="0”
      android:toDegrees="360"
      android:duration="5000"
      android:pivotX="50%”
      android:pivotY="90%"
      android:startOffset="0">
   </rotate>
</set>
```



Example 2 –animating from code

```
ImageView aniView = (ImageView) findViewById(R.id.imageView1);
ObjectAnimator animation1 = ObjectAnimator.ofFloat(aniView,
                        "rotation", dest);
animation1.setDuration(2000);
animation1.start();

// Example of animation lifecycle trapping
animation1.setAnimationListener(new AnimationListener(){
   @Override
    public void onAnimationEnd(Animation animation) {…}
 @Override
    public void onAnimationRepeat (Animation animation) {…}
 @Override
    public void onAnimationStart (Animation animation) {…}
});
```

ObjectAnimator

static ObjectAnimator ofFloat(Object target, String propertyName, float... values)

- Constructs and returns an ObjectAnimator that animates between float values.

public static ObjectAnimator ofFloat(T target, Property<T, Float> property, float... values)

- Animate a given (float) property in object target

ofInt is similar

Fading demo

```
float dest = 1;
if (aniView.getAlpha() > 0) {
   dest = 0;
}
ObjectAnimator animation3 = ObjectAnimator.ofFloat(aniView,"alpha", dest);
animation3.setDuration(2000);
animation3.start();
```

TypeEvaluator

Interface that implements:

public abstract T evaluate (float fraction, T startValue, T endValue)

- This function should return a linear interpolation between the start and end values, given the fraction parameter.
- The calculation is expected to be simple parametric calculation: result = x0 + t * (x1 - x0), where x0 is startValue, x1 is endValue, and t is fraction.

ObjectAnimator

static ObjectAnimator ofObject(Object target, String propertyName, TypeEvaluator evaluator, Object... values)

- Constructs and returns an ObjectAnimator that animates between Object values.
- .

static <T, V> ObjectAnimator ofObject(T target, Property<T, V> property, TypeEvaluator<V> evaluator, V... values)

Constructs and returns an ObjectAnimator that animates a given property between Object values.

AnimatorSet: combining animations

Since Android 3.0

void playSequentially(Animator... items)
void playTogether(Animator... items)
void start()
void end()

AnimatorSet.Builder play(Animator anim)

- This method creates a Builder object, which is used to set up playing constraints.

AnimatorSet.Builder

AnimatorSet.Builder after(Animator anim)

- anim starts when player ends.

AnimatorSet.Builder after(long delay)

- Start player after specified delay.

AnimatorSet.Builder before(Animator anim)

- start player when anim ends.

AnimatorSet.Builder with(Animator anim)

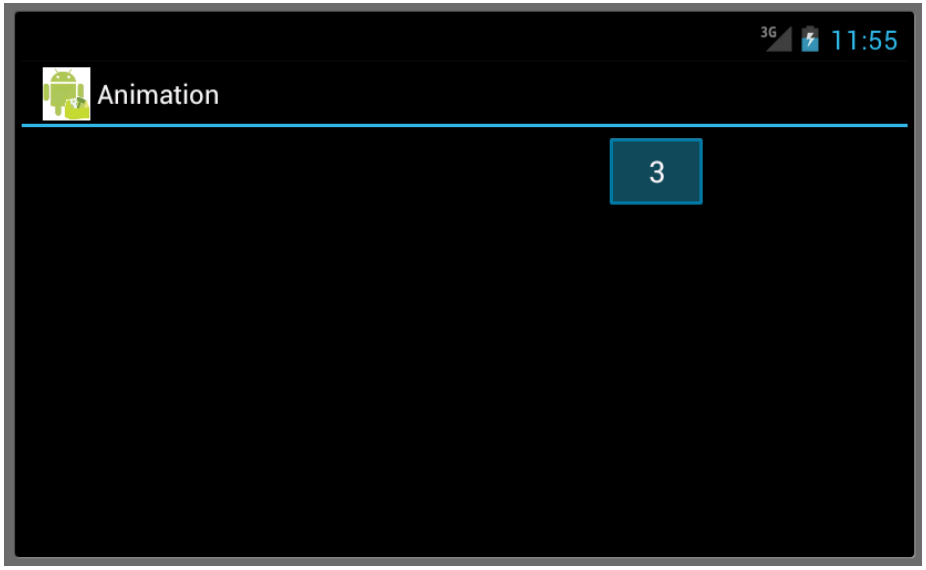
- Starts player and anim at the same time.

AnimatorSet: coreography

```
ObjectAnimator fadeOut = ObjectAnimator.ofFloat(aniView, "alpha", 0f);
fadeOut.setDuration(2000);

ObjectAnimator mover = ObjectAnimator.ofFloat(aniView,
        "translationX", -500f, 0f);
mover.setDuration(2000);

ObjectAnimator fadeIn = ObjectAnimator.ofFloat(aniView, "alpha",1f);
fadeIn.setDuration(2000);

AnimatorSet animatorSet = new AnimatorSet();
animatorSet.play(mover).with(fadeIn).after(fadeOut);
animatorSet.start();
```

Example

See the example in the zip file connected with this lecture:



See how the game behaves when rotating the device with and without **onSaveInstanceState** and **onRestoreInstanceState**

Physics based animation and layout changes animation

https://developer.android.com/training/animation/overview#physics-based