SENG440

Week 9

Sensors

Ben Adams (benjamin.adams@canterbury.ac.nz)

Introduction to Sensors in Android

- Most devices have built in sensors to measure and monitor
  - motion
  - orientation (aka position of device)
  - environmental conditions
- Sensors deliver raw data to applications

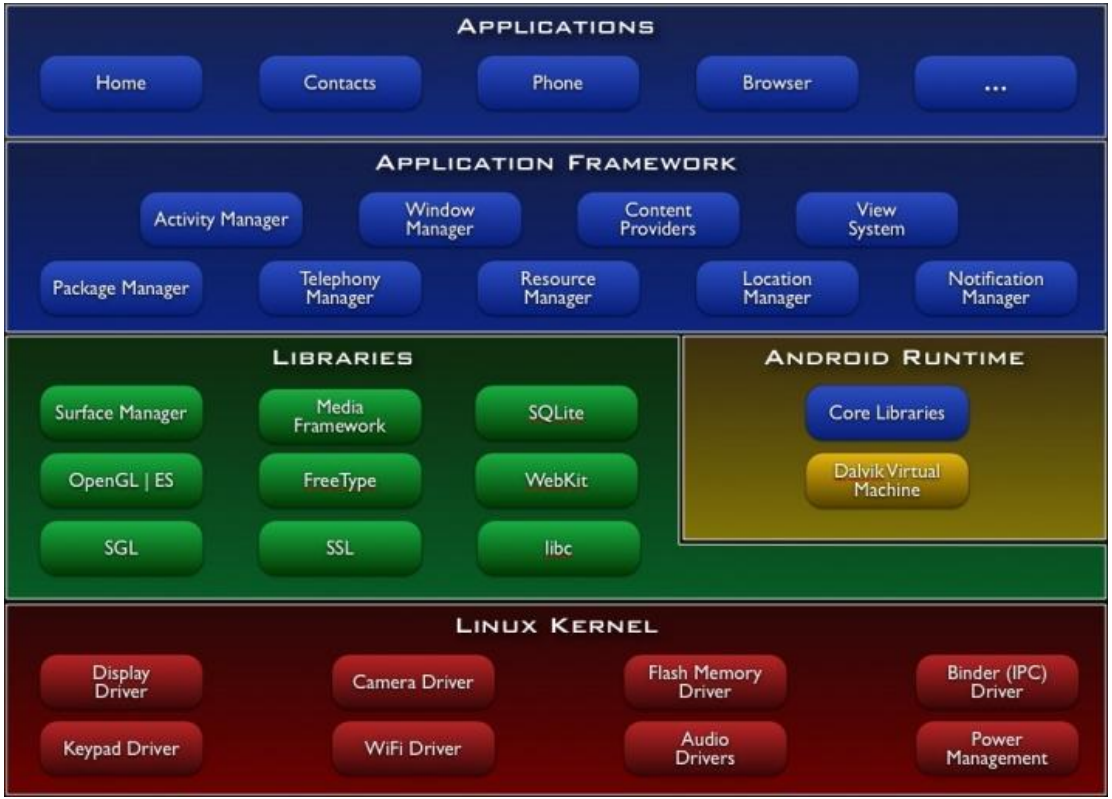
Sensor Framework

- Determine which sensors are available on a device.
- Determine an individual sensor's capabilities, such as its range, manufacturer, power requirements, and resolution.
- Acquire raw sensor data and define the minimum rate at which you acquire sensor data.
- Register and unregister sensor event listeners that monitor sensor changes.

Sensor Framework classes

- `SensorManager`
  - conduit between your classes and Sensors
- `Sensor`
  - abstract representations of Sensors on device
- `SensorEventListener`
  - register with SensorManager to listen for events from a Sensor
- `SensorEvent`
  - data sent to listener

Android Software Stack



Sensor Manager

Sensor Drivers

Two categories of sensors

- **Hardware sensors**
  - Built into the device
- **Software sensors**
  - Takes data from hardware sensors and manipulates it
  - From our perspective acts like a hardware sensor
  - Also known as synthetic or virtual sensors

Types of Sensors

- TYPE_ACCELEROMETER
  - Hardware
  - Acceleration in m/s^2
  - x, y, z axis
  - Includes gravity

Types of Sensors

- TYPE_AMBIENT_TEMPERATURE
  - Hardware
  - ”Room" temperature in degrees Celsius
  - no such sensor on many phones
- TYPE_GRAVITY
  - Software or hardware
  - Just gravity
  - If phone at rest it is same as TYPE_ACCELEROMETER

Types of Sensors

- TYPE_GYROSCOPE
  - Hardware
  - Measure device's rate of rotation in radians / second around 3 axis
- TYPE_LIGHT
  - Hardware
  - Light level in lx,
  - lux is SI measure illuminance in luminous flux per unit area

Types of Sensors

- TYPE_LINEAR_ACCELERATION
  - Software or hardware
  - Measure acceleration force applied to device in three axes, excluding the force of gravity
- TYPE_MAGNETIC_FIELD
  - Hardware
  - Ambient geomagnetic field in all three axes
  - uT micro Teslas

Types of Sensors

- TYPE_ORIENTATION [deprecated]
  - Software
  - Measure of degrees of rotation a device makes around all three axes
  - Use `SensorManager.getOrientation`
- TYPE_PRESSURE
  - Hardware
  - Ambient air pressure in hPa or mbar
  - Force per unit area
  - 1 Pascal = 1 Newton per square meter
  - hecto Pascals (100 Pascals)
  - milli bar - 1 mbar = 1 hecto Pascal

Types of Sensors

- TYPE_PROXIMITY
  - Hardware
  - Proximity of an object in cm relative to the view screen of a device
  - Usually binary (see range, resolution)
  - Typically used to determine if handset is being held to person's ear during a call
- TYPE_RELATIVE_HUMIDITY
  - Ambient humidity in percent (0 to 100)

Types of Sensors

- TYPE_ROTATION_VECTOR (ABSOLUTE)
  - Hardware or software
  - Orientation of device, three elements of the device's rotation vector
- TYPE_ROTATION_VECTOR
  - Orientation sensor
  - Replacement for TYPE_ORIENTATION
  - Combination of angle of rotation and access
  - Uses geomagnetic field in calculations
  - Compare to TYPE_GAME_ROTATION_VECTOR

Sensor Capabilities

- Various methods in Sensor class to get capabilities of Sensor:
  - minDelay (in microseconds)
  - power consumption in mA (microAmps)
  - maxRange
  - resolution

Sensor tools

Sensor Box for Android

HK SMARTER MOBI TECHNOLOGY CO.,LIMITED **Tools**

Everyone

Contains Ads

This app is available for your device

phyphox®
physical phone experiments

Using Sensors - Basics

- Obtain the `SensorManager` object
- Create a `SensorEventListener` for `SensorEvents`
  - Logic that responds to sensor event
  - Varying amounts of data from sensor depending on type of sensor
- Register the sensor listener with a `Sensor` via the `SensorManager`
- Unregister when done
  - A good thing to do in the `onPause` method

Using Sensors

- Using a `SensorManager`, register a `SensorEventListener` for the given sensor at the given sampling frequency:

```
registerListener(SensorEventListener listener, Sensor sensor, int samplingPeriodUs)
```

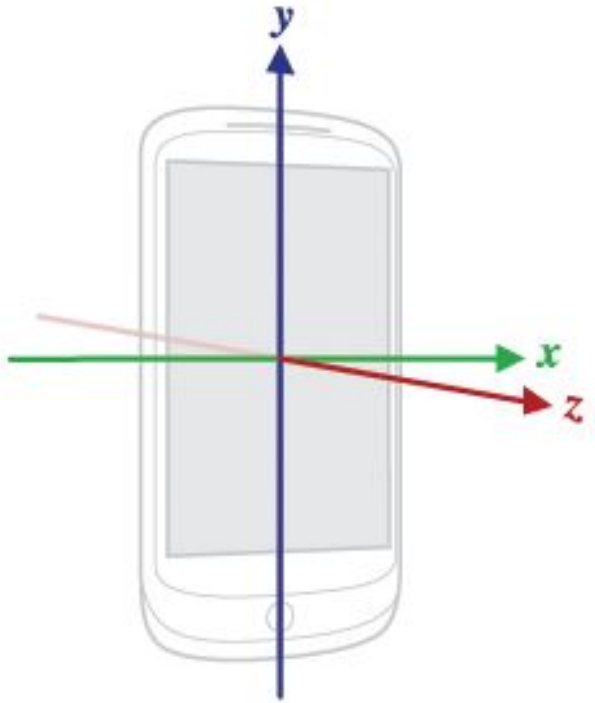
- The sampling frequency is just a hint:
  - `SENSOR_DELAY_NORMAL`,
  - `SENSOR_DELAY_UI`,
  - `SENSOR_DELAY_GAME`,
  - `SENSOR_DELAY_FASTEST`, or
  - time in microseconds (millionths of a second)

SensorEventListener

- Interface with two methods that you need to implement:
  - `onAccuracyChanged(Sensor sensor, int accuracy)`
  - `onSensorChanged(SensorEvent event)`
    - Sensor values have changed
    - This is the key method to override
    - Don't do significant computations in this method
- Don't hold onto the event
  - Part of pool of objects and the values may be altered soon

Sensor Coordinate System

- For most motion sensors:
- +x to the right
- +y up
- +z out of front face
- relative to device
- based on natural orientation of device
  - tablet -> landscape



Sensor Best Practices

- Unregister sensor listeners
  - When done with Sensor or activity using sensor paused (`onPause` method)

```
sensorManager.unregisterListener(sensorListener)
```

  - Otherwise data still sent and battery resources continue to be used

Sensors Best Practices

- Verify sensor available before using it
- Use `getSensorList` method and type
- Ensure list is not empty before trying to register a listener with a sensor

- Avoid deprecated sensors and methods
  - `TYPE_ORIENTATION` and `TYPE_TEMPERATURE` are deprecated as of Ice Cream Sandwich / Android 4.0

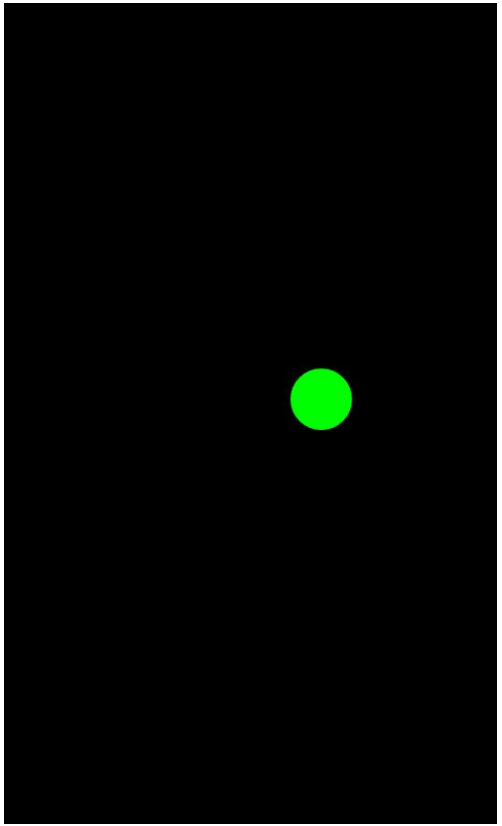
Sensors Best Practices

- Don't block the `onSensorChanged()` method
  - Recall the resolution on sensors
  - 50 updates a second for `onSensorChange` method not uncommon
  - When registering listener update is only a hint and may be ignored
  - If necessary save event and do work in another thread or async task

Sensor Sample - Moving Ball

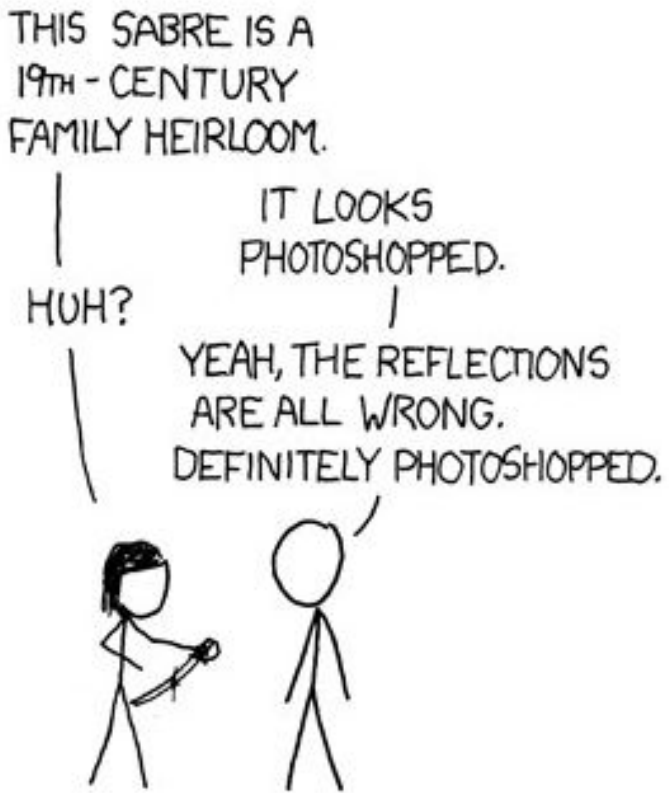
- Gross Simplification
- **velocity set equal to acceleration**

```
public void onSensorChanged(SensorEvent event) {
    //set ball speed based on phone tilt (ignore Z axis)
    // speed set equal to acceleration
    mBallVelocity.x = -event.values[0];
    mBallVelocity.y = event.values[1];
```



Reality is Unrealistic

- "When exposed to an exaggeration or fabrication about certain real-life occurrences or facts, some people will perceive the fictional account as being more true than any factual account."



Sensor Sample - Moving Ball

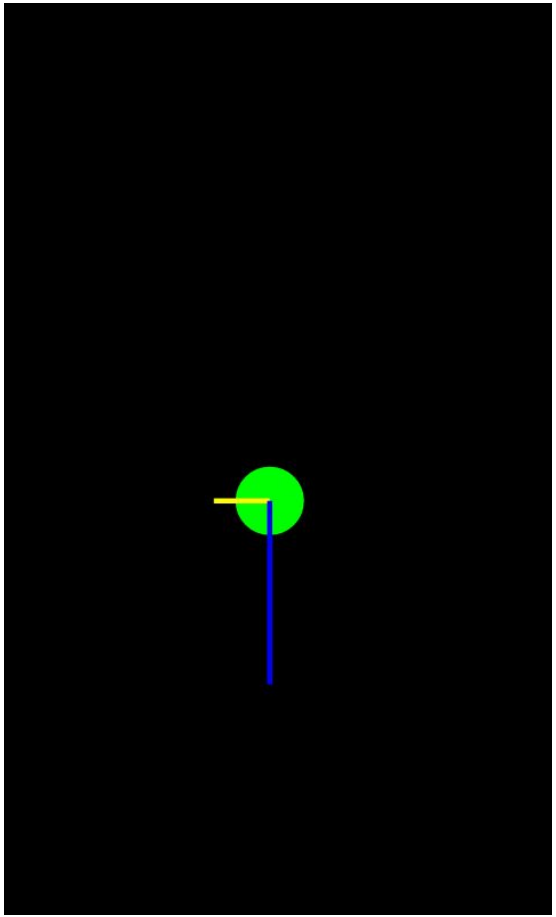
- Alternate Implementation

```
// try more realistic movement
float xA = -event.values[0];
float yA = event.values[1];
float aveXA = (xA + mPrevXAcc) / 2;
float aveYA = (yA + mPrevYAcc) / 2;
long currentTime = System.currentTimeMillis();
long elapsedTime = currentTime - mPrevTime;
mBallVelocity.x += aveXA * elapsedTime / 1000 / ACC_FUDGE_FACTOR; /
mBallVelocity.y += aveYA * elapsedTime / 1000 / ACC_FUDGE_FACTOR;

mPrevXAcc = xA;
mPrevYAcc = yA;
mPrevTime = currentTime;
```

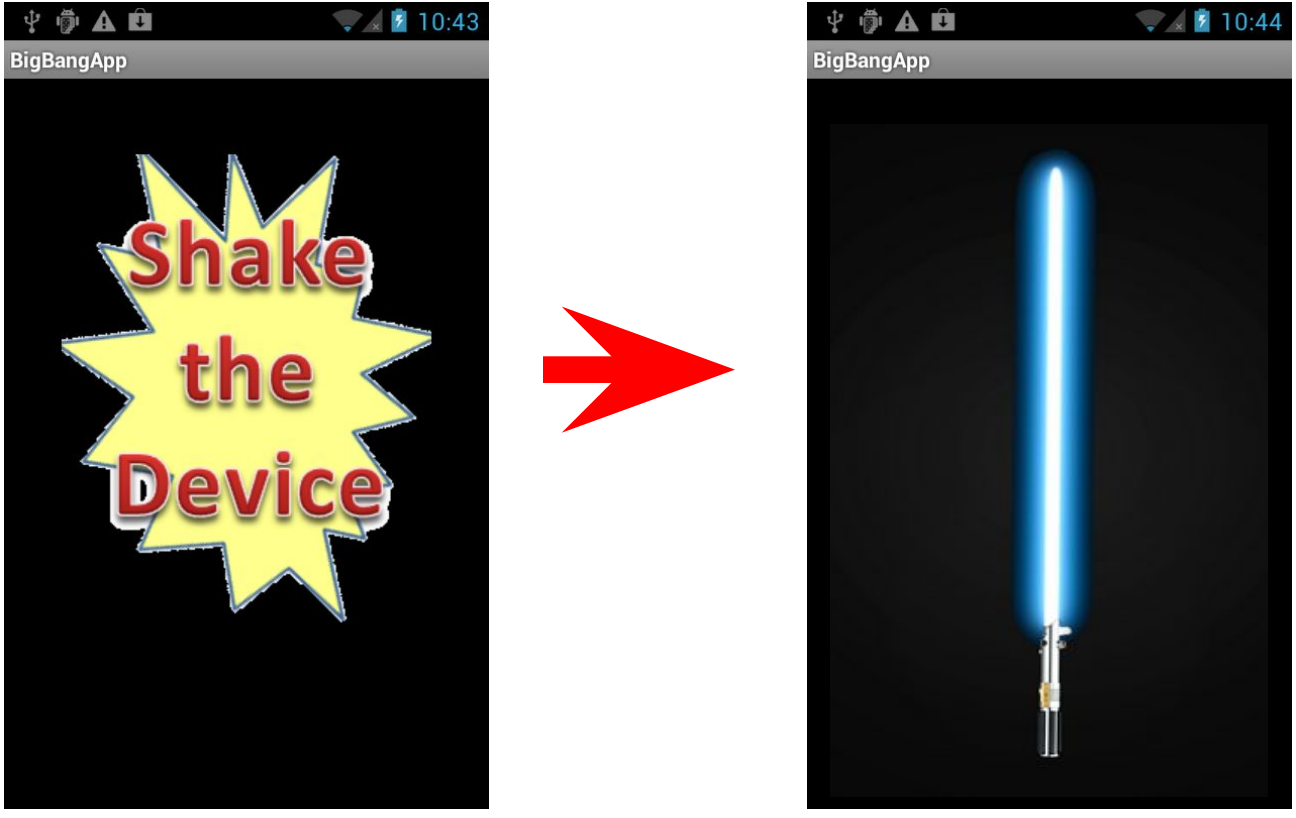
Sensor Sample

- Draw lines for x and y velocities

```
//called by invalidate()
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    mPaint.setStrokeWidth(1);
    mPaint.setColor(0xFF00FF00);
    canvas.drawCircle(mX, mY, mR, mPaint);

    mPaint.setStrokeWidth(3);
    mPaint.setColor(0xFFFFFF00);
    canvas.drawLine(mX, mY,
            mX + vX * 15, mY, mPaint);
    mPaint.setColor(0xFF0000FF);
    canvas.drawLine(mX, mY,
            mX, mY + vY * 15, mPaint);
```



Sound Effect App



Responding to Events

```
private class LinAccListener implements SensorEventListener {
    public void onSensorChanged(SensorEvent event) {
        if(event.sensor.getType() == Sensor.TYPE_LINEAR_ACCELERATION) {
            float x = event.values[0];
            float y = event.values[1];
            float z = event.values[2];
            float acc = (float)Math.sqrt( x * x + y * y);
            // Log.d("BBT", "" + acc);
            if(acc > 31) {
                Log.d("BBT", "" + acc);

                if(soundPlayer != null && !soundPlayer.isPlaying()) {
                    soundPlayer.start();
                    picture.setImageResource(R.drawable.crack);
                }
            }
        }
    }
```

Changing Images

- Use of an Image View
- Initial Image set in `onCreate`
- New image set in `onSensorChange`
- Register listener with `MediaPlayer`
- On completion reset image

```
@Override
public void onCompletion(MediaPlayer mp) {
    picture.setImageResource(R.drawable.shake);
}
```