CSC 472 / 372
Mobile Application Development for Android

Prof. Xiaoping Jia
School of Computing, CDM
DePaul University
xjia@cdm.depaul.edu
@DePaulSWEng

Outline

- Sensors in Android devices
- Motion sensors
- Accelerometer
- Gyroscope

Sensors

- Convert a wide range of physical measurements into digital signals.
- Android devices may include several types of sensors
  - Motion sensors
    - Measure the acceleration, linear and rotational, of the device
  - Position sensors
    - Measure the position and orientation of the device, e.g., geo location, proximity
  - Environment sensors
    - Measure environmental parameters, e.g., ambient temperature and pressure, illumination, and humidity, etc.

Android Sensor Framework

- Supports various sensor related tasks
- Determine the availability and capability of sensors
- Acquire raw data from sensors at a given sampling rate
- Monitor sensor changes using sensor listeners

Sensor Types

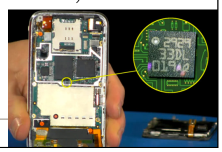
- Sensors in Android can be hardware-based or software-based
- *Hardware-based* sensors are physical components built into a device
  - You may acquire the raw data
  - e.g., accelerometer, gyroscope
- *Software-based sensors* derive data from one or more hardware-based sensors
  - a.k.a. *virtual sensors*, or *synthetic sensors*, e.g., gravity
  - Use the same API as the hardware-based sensors

Why Motion Sensors?

- Your device is aware of its precise motion within its environment
- Controls are no longer limited to the UI widgets on the screen
- Opens the door to a new world of
  - gaming possibilities
  - motion based gestures
  - medical devices
  - other creative uses

The Accelerometer

- A hardware-based sensor measures *accelerations* (in m/s^2) in each of the three physical axes.
  - Acceleration is the *rate of change* of velocity.
  - *Acceleration of a device = Earth Gravity + User Acceleration*
- Present in early Android devices
  - API available since Android 1.5 (API level 3)
- Low power consumption
- Responsive
- Noisy



DePaul University

Acceleration Data

Sir Isaac Newton
1642 – 1726

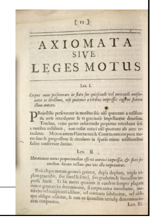
- The accelerometer measures the acceleration of the device (A_d)
  - The relation to the forces applied to the sensor (F_s)

    $A_d = - \sum F_s$ / mass
  - The forces include the force of *gravity*

Newton's *Second Law of Motion* (1687)

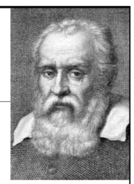
$$F = m \bullet a$$

1 N = 1 kg·m/s^2



DePaul University

Gravity



Galileo Galilei
1564 – 1642

- When the device is still, the accelerometer reading is

  $g \approx 9.81\text{m/s}^2$

  Average Earth gravitational acceleration at sea level
- Separate the gravity

  $A_d = - g - \sum F$ / mass

  User acceleration
- Two software-based sensors
  - *Gravity* – Earth gravitational acceleration
  - *Linear Acceleration* – Acceleration sans gravity

DePaul University

The Gyroscope

- A device for measuring and maintaining orientation
- A hardware-based sensor measures the rotation rate (in radian/s) around each of the three physical axes
  - Radian: standard unit of angular measure

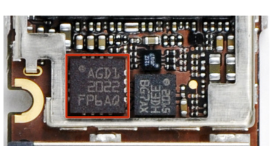
    1 rad = 180° / π ≈ 57.2958°

Gyroscope invented by Léon Foucault in 1852.

DePaul University 13

The Gyroscope

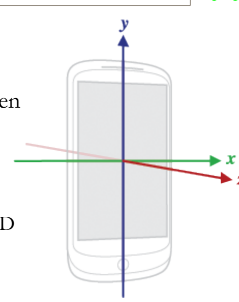
- MEMS gyroscope (micro-electromechanical system)
- API available since Android 2.3 (API level 9)
- Responsive
- Precise
- Inherent bias drift



DePaul University 14

Senor Coordinate System

- Standard 3-axis coordinate system
- Relative to the device's screen in the default orientation
- **Never changes when the device is rotated.**
  - The coordinate system for 2-D graphics rotates according to the device screen orientation.



DePaul University 15

Sensor Manager

- A system service that manages all sensors
- Useful methods:
  - Determine availability of sensors `getSensorList(type)`
  - Access an instance of a specific type of sensor `getDefaultSensor(type)`
  - Determine the capability and attribute of a sensor `getResolution() getMaximumRange()`
  - Minimum delay in sensing data, i.e., sampling rate `getMinDelay()`

DEPAUL UNIVERSITY 16

Monitoring Sensor Data

- Implement the callback methods in the *Sensor Event Listener* interface
  - `onSensorChanged()`
  - `onAccuracyChanged()`
- Register the listener and specify a sampling rate
  - Normal (≈5fps), UI (≈17fps), Game (≈50fps)
- Unregister the listener when unused or paused
  - The system will *not* disable sensors automatically when the screen turns off.

DEPAUL UNIVERSITY 17

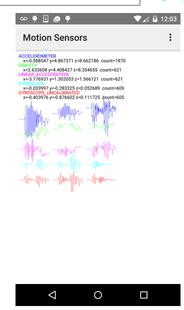
Sensor Data

- The current reading of the sensors are delivered in the `values` array of the *Sensor Event*
- For acceleration data
  - `values[0]` acceleration along the X axis
  - `Values[1]` acceleration along the Y axis
  - `Values[2]` acceleration along the Z axis
- For rotation data (gyroscope)
  - `values[0]` rate of rotation around the X axis
  - `Values[1]` rate of rotation around the Y axis
  - `Values[2]` rate of rotation around the Z axis

DEPAUL UNIVERSITY 18

The Motion Sensor App

- Demonstrate the use of motion sensors
- Acquire data from five sensors
  - Accelerometer
  - Gravity
  - Linear acceleration
  - Gyroscope
  - Gyroscope uncalibrated
- Display sensor data textually and graphically



DEPAUL UNIVERSITY 19

Screen Orientation

- For apps using motion sensors, we want to fix the screen orientation in the *Android Manifest*

```
<?xml version="1.0" encoding="utf-8"?>
<manifest … >
  <application … >
    <activity
      android:name=".MainActivity"
      android:label="@string/app_name"
      android:screenOrientation="portrait" >
      <intent-filter> … </intent-filter>
    </activity>
  </application>
</manifest>
```

DEPAUL UNIVERSITY 20

The Motion Sensors App – The Custom View

```
public class MyView extends View {
  private SensorManager sensorManager;
  private int[] sensorTypes = {
       Sensor.TYPE_ACCELEROMETER,
       Sensor.TYPE_GRAVITY,
       Sensor.TYPE_LINEAR_ACCELERATION,
       Sensor.TYPE_GYROSCOPE,
       Sensor.TYPE_GYROSCOPE_UNCALIBRATED
  };
  private Sensor[] sensors = new Sensor[5];
  private SensorEventListener[] listeners =
               new SensorEventListener[5];
  private float[][] sensorData = new float[5][3];
  int[] colors = { … };
  private String[] labels = { "ACCELEROMETER", "GRAVITY", … };
…
```

Arrays for: sensors, listeners, and current readings – for each sensor, 3 axes.

DEPAUL UNIVERSITY 22

The Custom View – The Constructors

```
public class MyView extends View {
  public MyView(Context context) {
    super(context);
    initSensors();
  }
  public MyView(Context context, AttributeSet attrs) {
    super(context, attrs);
    initSensors();
  }

  private float[][][] sensorDataHistory = new float[200][5][3];
  private int start = 0;
```

Position of the earliest historic data

Store historic sensor data in a circular array. 200 historic data points.

The Custom View – Initialize the Sensors

```
private void initSensors() {
  sensorManager = (SensorManager)
    getContext().getSystemService(Context.SENSOR_SERVICE);
  for (int s = 0; s < 5; s++) {
    final float[] data = sensorData[s];
    sensors[s] = sensorManager.getDefaultSensor(sensorTypes[s]);
    if (sensors[s] != null) {
      listeners[s] = new SensorEventListener() {
        @Override public void onSensorChanged(SensorEvent event) {
          for (int i = 0; i < 3; i++) data[i] = event.values[i];
          invalidate();
        }
        @Override public void onAccuracyChanged(Sensor sensor,
                                                int accuracy) { }
      };
    }
  }
}
```

The Custom View – Register & Unregister the Listeners

```
void resume() {
  for (int s = 0; s < 5; s++) {
    if (listeners[s] != null)
      sensorManager.registerListener(listeners[s], sensors[s],
                                     SensorManager.SENSOR_DELAY_NORMAL);
  }
}
void pause() {
  for (int s = 0; s < 5; s++) {
    if (listeners[s] != null)
      sensorManager.unregisterListener(listeners[s]);
  }
}
```


The Motion Sensors App – Displaying the Sensor Data, 1/3

```
@Override protected void onDraw(Canvas canvas) {
  canvas.drawColor(Color.WHITE);
  paint.setAntiAlias(true);
  paint.setColor(Color.BLACK);
  paint.setTypeface(Typeface.DEFAULT);
  paint.setTextSize(30);
  for (int s = 0; s < 5; s++) {
    String msg = String.format("x=%f y=%f z=%f",
        sensorData[s][0], sensorData[s][1], sensorData[s][2]);
    paint.setColor(colors[s]);
    canvas.drawText(labels[s], 20, 50 + 60 * s, paint);
    paint.setColor(Color.BLACK);
    canvas.drawText(msg, 50, 80 + 60 * s, paint);
  }
  …
```

Display the sensor data textually.

The Motion Sensors App – Displaying the Sensor Data, 2/3

```
@Override protected void onDraw(Canvas canvas) {
```

Display textual data.

```
  for (int s = 0; s < 5; s++) {
    for (int i = 0; i < 3; i++)
      sensorDataHistory[start][s][i] = sensorData[s][i];
  }
  start = ++start % 200;
```

Add the latest sensor data to historic data array. Drop the earliest data point.

Draw historical graph.

```
}
```

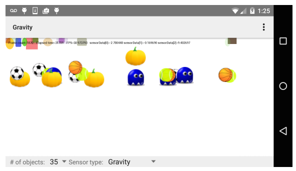

The Motion Sensors App – Displaying the Sensor Data, 3/3

```
@Override protected void onDraw(Canvas canvas) {
  …
  paint.setStyle(Paint.Style.STROKE);
  for (int s = 0; s < 5; s++) {
    int ybase = 450 + s * 100;
    paint.setColor(colors[s]);
    for (int axis = 0; axis < 3; axis++) {
      int xbase = 20 + axis * 220;
      Path path = new Path();
      path.moveTo(xbase, ybase - sensorDataHistory[start][s][axis]*8);
      for (int i = 1; i < 200; i++) {
        int h = (start + i) % 200;
        path.lineTo(xbase + i, ybase - sensorDataHistory[h][s][axis]*8);
      }
      canvas.drawPath(path, paint);
    }
  }
}
```

Draw historical graph.

The Gravity App

- Extension of the *Bouncing Objects* app
  - Drawing using Surface View
- Use motion sensors to control the movement of the objects.
  - Gravity
  - Linear acceleration (demo)
  - Gyroscope (demo)
- Most code is same as the *Bouncing Objects* app



The Gravity App – The Custom View

```
public class MyView extends SurfaceView
        implements SurfaceHolder.Callback {
  …
  private SensorManager sensorManager;
  private Sensor sensor;
  private SensorEventListener listener;
  private float[] sensorData = new float[3];

  private static final int MAX_V = 15;
  private static final float GRAVITY_F = 0.2f;

  Constructors and methods
}
```


The Custom View – Constructors

```
public MyView(Context context) {
  super(context);
  init();
}
public MyView(Context context, AttributeSet attrs) {
  super(context, attrs);
  init();
}
private void init() {
  holder = getHolder();
  holder.addCallback(this);
  initShapes(5);
  sensorManager = (SensorManager)
    getContext().getSystemService(Context.SENSOR_SERVICE);
  initSensor(Sensor.TYPE_GRAVITY);
}
```


The Custom View – Initialization of the Sensor

```
private void initSensor(int type) {
  sensor = sensorManager.getDefaultSensor(type);
  if (sensor != null) {
    listener = new SensorEventListener() {
      @Override public void onSensorChanged(SensorEvent event) {
        for (int i = 0; i < 3; i++)
          sensorData[i] = event.values[i];
      }
      @Override public void onAccuracyChanged(Sensor sensor,
                                             int accuracy) { }
    };
  }
}
```


The Custom View – Control the Movement of Objects

```
class MyShape {
  void move() {
    float gx = sensorData[1]; // landscape mode
    float gy = sensorData[0]; // landscape mode
    dx += gx * GRAVITY_F;
    dy += gy * GRAVITY_F;
    dx = Math.min(Math.max(dx, -MAX_V), MAX_V);
    dy = Math.min(Math.max(dy, -MAX_V), MAX_V);
    Rect bounds = drawable.getBounds();
    if (bounds.right >= width && dx > 0 ||
        bounds.left < 0 && dx < 0) dx = -dx;
    if (bounds.bottom >= height && dy > 0 ||
        bounds.top < 0 && dy < 0) dy = -dy;
    bounds.left += dx; bounds.right += dx;
    bounds.top += dy; bounds.bottom += dy;
  }
  …
}
```

Map the sensor coordinates to 2D landscape coordinates

Use acceleration to adjust velocity

The Custom View – Register & Unregister the Listener

```
public void startAnimation() {
  done = false;
  if (listener != null)
    sensorManager.registerListener(listener, sensor,
                          SensorManager.SENSOR_DELAY_GANE);
  if (surfaceAvailble) startRenderingThread();
}

public void stopAnimation() {
  done = true;
  if (listener != null)
    sensorManager.unregisterListener(listener);
}
```

The Sample Code

- The sample apps in this lecture are available in D2L
  - MotionSensors.zip
  - Gravity.zip
- Each zip archive contains the entire project folder
- Unzip the file and import to Android Studio

Next …

- Final project demo, next Tuesday

