LẬP TRÌNH DI ĐỘNG

Bài 13: Sensors + Animations

Nhắc lại bài trước

- Các dịch vụ đa phương tiện (phần II):
  - 2 cách chơi video trong android:
    - VideoView kết hợp với MediaController
    - MediaPlayer kết hợp với SurfaceView
  - Cách thứ 2 cho phép lập trình viên can thiệp nhiều hơn vào việc chơi video
  - Dịch vụ text-to-speech: cho phép chuyển đổi từ văn bản thành lời phát âm tương ứng
  - Dịch vụ camera: cho phép chụp ảnh hoặc quay video, sử dụng intent có sẵn của hệ thống hoặc thông qua lập trình class Camera

Nhắc lại bài trước

- Android cung cấp nhiều dịch vụ vị trí phong phú
- Dịch vụ định vị toàn cầu (GPS - global positioning service): sử dụng tín hiệu vệ tinh (+ tính hiệu mạng) để tính toán ra vị trí hiện tại trên bản đồ
- Dịch vụ thông tin địa lý (GL - geocoding location): cho phép tìm kiếm địa chỉ trong CSDL bản đồ dựa trên vị trí kinh độ + vĩ độ hoặc dựa trên tên gợi nhớ
- Dịch vụ bản đồ: dịch vụ của Google, cần phải đăng ký mới sử dụng được, cho phép nhúng bản đồ của Google vào ứng dụng và tương tác

Nội dung

1. Cảm biến (sensor)
    1. Sensor và SensorManager
    2. Các loại sensor thông dụng
    3. Các bước làm việc với sensor
    4. Ví dụ: “la bàn” đơn giản
    5. Kinh nghiệm làm việc với sensor
2. Hiệu ứng đồ họa (animation)
    1. Giới thiệu về animation
    2. Tạo animation bằng XML và Code
    3. Drawable animation

Phần 1.1

Sensor và SensorManager

Sensor và SensorManager

- Sensor: chip cảm ứng nằm trong thiết bị, cung cấp dữ liệu mà nó đo đạc được cho hệ điều hành
- Trong android, các sensor được quản lý chung bởi SensorManager, một dịch vụ hệ thống

```
SensorManager sm = (SensorManager)
  getSystemService(SENSOR_SERVICE);
```

- Thông qua SensorManager lập trình viên có thể:
  - Lấy danh sách các sensor có trong hệ thống hiện tại
  - Lấy đối tượng để làm việc trực tiếp với từng sensor
  - Đăng kí listener để xử lý sự kiện do các sensor báo về

Sensor và SensorManager

- Muốn lấy một sensor cụ thể, sử dụng phương thức `getDefaultSensor(TYPE)`, tham số TYPE sẽ quy định kiểu đối tượng Sensor muốn lấy ra
- Các loại sensor hiện được Android OS hỗ trợ (hằng số khai báo trong class Sensor):
  - TYPE_ACCELEROMETER: cảm biến gia tốc
  - TYPE_AMBIENT_TEMPERATURE: cảm biến nhiệt độ môi trường
  - TYPE_GRAVITY: cảm biến trọng lực
  - TYPE_GYROSCOPE: cảm biến con quay hồi chuyển
  - TYPE_LIGHT: cảm biến ánh sáng

Sensor và SensorManager

- Các loại sensor hiện được Android OS hỗ trợ (tiếp):
  - TYPE_LINEAR_ACCELERATION: cảm biến gia tốc tuyến tính
  - TYPE_MAGNETIC_FIELD: cảm biến từ tính
  - TYPE_PRESSURE: cảm biến áp suất
  - TYPE_PROXIMITY: cảm biến khoảng cách gần
  - TYPE_RELATIVE_HUMIDITY: cảm biến độ ẩm
  - TYPE_ROTATION_VECTOR: cảm biến xoay
  - TYPE_GAME_ROTATION_VECTOR: cảm biến xoay 2D
  - TYPE_SIGNIFICANT_MOTION: cảm biến chuyển động

Sensor và SensorManager

```
public class SensorActivity extends Activity
                                    implements SensorEventListener {
    final SensorManager sm;
    final Sensor light;

    public SensorActivity() {
        sm = (SensorManager) getSystemService(SENSOR_SERVICE);
        light = sm.getDefaultSensor(Sensor.TYPE_LIGHT);
    }
    protected void onResume() {
        super.onResume();
        sm.registerListener(this, light,
                            SensorManager.SENSOR_DELAY_NORMAL);
    }
```

Sensor và SensorManager

```
    protected void onPause() {
        super.onPause();
        sm.unregisterListener(this);
    }

    // khi độ chính xác của sensor thay đổi
    //     UNRELIABLE <-> LOW <-> MEDIUM <-> HIGH
    public void onAccuracyChanged(Sensor s, int accuracy) {
    }

    // khi thông số sensor cập nhật
    public void onSensorChanged(SensorEvent event) {
    }
}
```

Phần 1.2

Các loại sensor thông dụng

Các loại sensor

- Cảm biến trong Android chia làm 3 nhóm
  - Cảm biến chuyển động
  - Cảm biến vị trí
  - Cảm biến môi trường
- Mỗi loại sensor có những đặc điểm vật lý khác nhau, muốn hiểu chính xác các chi tiết các sensor cần đọc tài liệu hướng dẫn (cần có kiến thức nhất định về vật lý và xử lý số liệu)
- Một số sensor là loại virtual (ảo), tức là kết quả được tính toán hoặc nội suy từ nguồn khác

Các loại sensor

- Android SDK không có các class định sẵn cho từng loại sensor mà chỉ có TYPE của sensor, dữ liệu do sensor trả về là float (trường hợp cảm biến 1 đầu ra – chẳng hạn đo ánh sáng) hoặc float[] (trường hợp cảm biến nhiều đầu ra)
- TYPE_AMBIENT_TEMPERATURE: cảm biến nhiệt độ, đơn vị đo là ^{0}C
- TYPE_LIGHT: cảm biến ánh sáng, đơn vị đo là lx
- TYPE_PRESSURE: cảm biến áp suất không khí, đơn vị đo là mbar

Các loại sensor

- TYPE_PROXIMITY: cảm biến khoảng cách đến đối tượng, đơn vị đo là cm
- TYPE_RELATIVE_HUMIDITY: cảm biến độ ẩm, đơn vị là %
- TYPE_ACCELEROMETER / TYPE_GRAVITY: cảm biến gia tốc / hấp dẫn trong 3D(x,y,z), đơn vị m/s^2
- TYPE_GYROSCOPE: cảm biến tốc độ góc quay trong 3D, đơn vị rad/s
- TYPE_MAGNETIC_FIELD: cảm biến lực từ trong 3D, đơn vị là µT

Phần 1.3

Các bước làm việc với sensor

Các bước làm việc với sensor

1. Lấy SensorManager từ các dịch vụ hệ thống, thông qua `getSystemService(Context.SENSOR_SERVICE)`
2. Từ SensorManager lấy đối tượng Sensor điều khiển cảm biến cần sử dụng

```
sensor = sensorService.getDefaultSensor(TYPE);
```

3. Cài đặt các bộ nghe (listener) phù hợp để xử lý các số liệu do cảm biến trả về
4. Tắt sensor trong những tình huống không cần thiết để tránh thiết bị tiêu tốn năng lượng
5. Xử lý lỗi hoặc thay đổi độ nhạy của thiết bị

Phần 1.4

Ví dụ: “la bàn” đơn giản

Ví dụ: "la bàn" đơn giản

- "la bàn": thiết bị định hướng 2D
- Dùng cảm biến góc xoay (TYPE_ORIENTATION)
- Viết một custom view (MyCompassView) để hiển thị la bàn đơn giản
- Cập nhật dữ liệu mỗi khi nhận được phản hồi từ cảm biến
- Hủy việc theo dõi cảm biến khi ứng dụng bị tạm ngưng (để tiết kiệm năng lượng)

Ví dụ: La bàn đơn giản

```
public class MainActivity extends Activity {
    private static SensorManager sensorService;
    private MyCompassView compass;
    private Sensor sensor = null;

    // kết thúc app: hủy đăng ký cảm biến (giúp hệ điều hành có thể
    // tắt cảm biến nếu cần)
    protected void onDestroy() {
        super.onDestroy();
        if (sensor != null)
            sensorService.unregisterListener(mySensorEventListener);
    }
```

Ví dụ: La bàn đơn giản

```
// khởi tạo app: tạo giao diện + đăng ký lấy dữ liệu từ cảm biến
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    compass = new MyCompassView(this);
    setContentView(compass);
    sensorService = (SensorManager)
        getSystemService(Context.SENSOR_SERVICE);
    sensor = sensorService.getDefaultSensor(Sensor.TYPE_ORIENTATION);
    if (sensor != null)
        sensorService.registerListener(mySensorEventListener, sensor,
            SensorManager.SENSOR_DELAY_NORMAL);
    else
        finish();
}
```

Ví dụ: La bàn đơn giản

```
    // xử lý dữ liệu cảm biến gửi về: cập nhật hình ảnh la bàn
    private SensorEventListener mySensorEventListener =
        new SensorEventListener() {
            @Override
            public void onAccuracyChanged(Sensor s, int accuracy) {
            }
            @Override
            public void onSensorChanged(SensorEvent event) {
                compass.updateData(event.values[0]);
            }
        };
}
```

Ví dụ: La bàn đơn giản

```
public class MyCompassView extends View {
    private Paint area;
    private float arc = 0;
    // constructor
    public MyCompassView(Context context) {
        super(context);
        init();
    }
    // vẽ lại la bàn với góc mới
    public void updateData(float position) {
        arc = position;
        invalidate();
    }
```

Ví dụ: La bàn đơn giản

```
// vẽ lại hình ảnh la bàn ứng với số liệu mới
protected void onDraw(Canvas c) {
    int xPoint = getMeasuredWidth() / 2;
    int yPoint = getMeasuredHeight() / 2;
    float radius = (float) (Math.max(xPoint, yPoint) * 0.6);
    c.drawCircle(xPoint, yPoint, radius, area);
    float x = (float) (xPoint +
       radius * Math.sin((double) (-arc) / 180 * Math.PI));
    float y = (float) (yPoint –
       radius * Math.cos((double) (-arc) / 180 * Math.PI));
    c.drawLine(xPoint, yPoint, x, y, area);
    c.drawText(String.valueOf(arc), xPoint, yPoint, area);
}
```

Ví dụ: La bàn đơn giản

```
    // hàm vẽ lại hình ảnh của view
    private void init() {
        area = new Paint();
        area.setAntiAlias(true);
        area.setColor(Color.RED);
        area.setStrokeWidth(3);
        area.setStyle(Paint.Style.STROKE);
        area.setTextSize(30);
    }
}
```

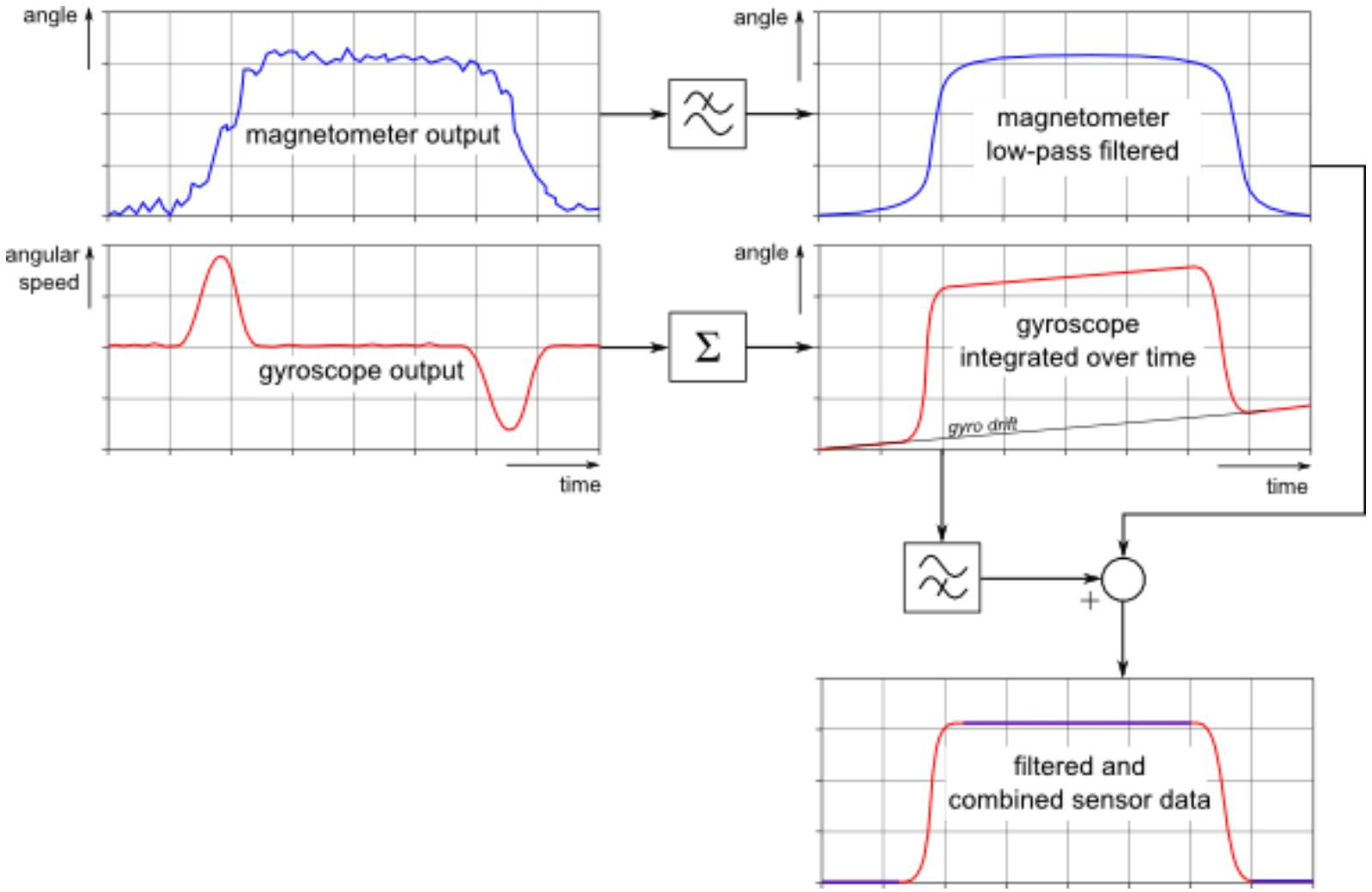
Phần 1.5

Kinh nghiệm làm việc với sensor

Kinh nghiệm làm việc với sensor

- Nhất thiết phải giải phóng sensor khi không cần thiết, nếu không ứng dụng sẽ rất hao pin
- Hệ thống không tự động tắt sensor kể cả khi tắt màn hình
- Chú ý khi làm việc với các thông số 3D: các chiều có thể bị hoán đổi vị trí khi người sử dụng đặt thiết bị theo chiều âm (ví dụ: máy bị lật úp)
- Nên kiểm thử trên thiết bị thật và hiệu chỉnh độ nhạy dần dần
- Kết hợp nhiều sensor để thiết bị "nhạy cảm" hơn

Kinh nghiệm làm việc với sensor



Phần 2.1

Giới thiệu về animation

Animations – nguyên tắc

- Animation: tập hợp các API cho phép biến đổi các view trong một thời gian ngắn (từ android 3.0)
- Tạo animation theo nhiều cách:
  - Định nghĩa trong XML (folder "res/anim/") và nạp bởi câu lệnh `AnimationUtils.loadAnimation`
  - Tạo code: tạo các biến đối tượng phù hợp
- Các animation có thể ghép với nhau thành để thực hiện nhiều hiệu ứng (gọi là `AnimationSet`)
- (advanced) Tự tạo animation: kế thừa `Animation` và viết lại `applyTransformation`

Animations – nguyên tắc

- Các animation được cung cấp bởi Android
  - AlphaAnimation: Thay đổi độ trong suốt của đối tượng
    - Fade In / Fade Out
    - Blink
  - RotateAnimation: Xoay đối tượng
  - ScaleAnimation: Thay đổi kích thước
    - Zoom In / Zoom Out
    - Slide Up / Slide Down
    - Bounce
  - TranslateAnimation: Dịch chuyển đối tượng
- Bằng cách điều chỉnh các tham số ta có thể tạo các animation khác nhau

Phần 2.2

Tạo animation bằng XML và Code

Animations – XML vs Code

- Ví dụ về XML: res/animator/fadein.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android" >
<alpha android:duration="1000"
     android:interpolator="@android:anim/accelerate_interpolator"
     android:fromAlpha="0.0"
     android:toAlpha="1.0" />
</set>

// nạp animation từ XML để sử dụng
Animation ani = AnimationUtils.loadAnimation(this, R.animator.fadein);
```

- Ví dụ tạo animation bằng code:

```
Animation ani = new AlphaAnimation(1, 0);
ani.setInterpolator(new AccelerateInterpolator());
ani.setDuration(1000);
```

- Sử dụng animation:

```
myView.startAnimation(ani);
```

Animations – XML vs Code

Một số thuộc tính quan trọng

- android:duration: thời gian chạy hiệu ứng (ms)
- android:startOffset: thời điểm bắt đầu chạy (ms)
- android:fillAfter: có giữ lại trạng thái sau hiệu ứng không (true/false)
- android:repeatCount: số lần lặp lại hiệu ứng
- android:repeatMode: chế độ lặp (RESTART | REVERSE)
- android:interpolator: kiểu diễn biến của hiệu ứng
  - *"@android:anim/accelerate_interpolator"*
  - "*@android:anim/ accelerate_decelerate_interpolator*"
  - …

Animations – XML vs Code

Có thể xử lý sự kiện khi hiệu ứng bắt đầu, kết thúc hoặc lặp lại bằng AnimationListener

```
rotAni.setAnimationListener(new AnimationListener() {
    // kết thúc hiệu ứng
    public void onAnimationEnd(Animation arg0) {
        layerImage.setVisibility(View.INVISIBLE);
        layerButtons.setVisibility(View.VISIBLE);
    }
    // chuẩn bị lặp lại hiệu ứng
    public void onAnimationRepeat(Animation arg0) { }
    // bắt đầu hiệu ứng
    public void onAnimationStart(Animation arg0) { }
});
```

Animations – Interpolator

- Interpolator: trình điều khiển quá trình thực hiện hiệu ứng
- Một số Interpolator thông dụng:
  - AccelerateDecelerateInterpolator: chậm-nhanh-chậm
  - AccelerateInterpolator: nhanh dần
  - DecelerateInterpolator: chậm dần
  - CycleInterpolator: lặp lại animation theo chu kì
  - LinearInterpolator: tuyến tính, không đổi
- Có thể tự viết interpolator riêng nếu muốn (phải tìm hiểu thêm về Interpolator.Result)

Animations – set of animations

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
  android:interpolator="@android:anim/accelerate_interpolator"
  android:shareInterpolator="true" android:duration="1000" android:fillAfter="false" >

  <alpha android:fromAlpha="1.0" android:toAlpha="0.0" />

  <rotate android:fromDegrees="0" android:toDegrees="360"
    android:pivotX="50%" android:pivotY="50%" />

  <scale android:fillAfter="true"
    android:fromXScale="1.0" android:fromYScale="1.0"
    android:toXScale="0.0" android:toYScale="0.0"
    android:pivotX="50%" android:pivotY="50%" />

</set>
```

Animations – example

```
public class MainActivity extends Activity {
    Animation x;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        x = AnimationUtils.loadAnimation(this, R.animator.abc);
    }

    public void btnAni(View v) {
        View t = findViewById(R.id.editText1);
        t.startAnimation(x);
    }
}
```

Phần 2.3

Drawable animation

Drawable Animation

- 3 kiểu animations trong Android:
  - Property animation: gốc của mọi animation
  - View animation: animation trên một view và sử dụng một image
  - Drawable animation: animation trên một view và sử dụng dãy image
- Drawable Animation: hoạt hình bằng dãy các ảnh

```
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
android:oneshot="true">
  <item android:drawable="@drawable/rocket_thrust1" android:duration="200" />
  <item android:drawable="@drawable/rocket_thrust2" android:duration="200" />
  <item android:drawable="@drawable/rocket_thrust3" android:duration="200" />
</animation-list>
```

Drawable Animation: ví dụ

```
AnimationDrawable rocketAnimation;

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    ImageView rocketImage = (ImageView) findViewById(R.id.rocket_image);
    rocketImage.setBackgroundResource(R.drawable.rocket_thrust);
    rocketAnimation = (AnimationDrawable) rocketImage.getBackground();
}

public boolean onTouchEvent(MotionEvent event) {
    if (event.getAction() != MotionEvent.ACTION_DOWN) return super.onTouchEvent(event);
    rocketAnimation.start();
    return true;
}
```