Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

Lập trình Android

Bài 27: *Gestures và Touch Events*

Phòng LT & Mạng

http://csc.edu.vn/lap-trinh-va-csdl

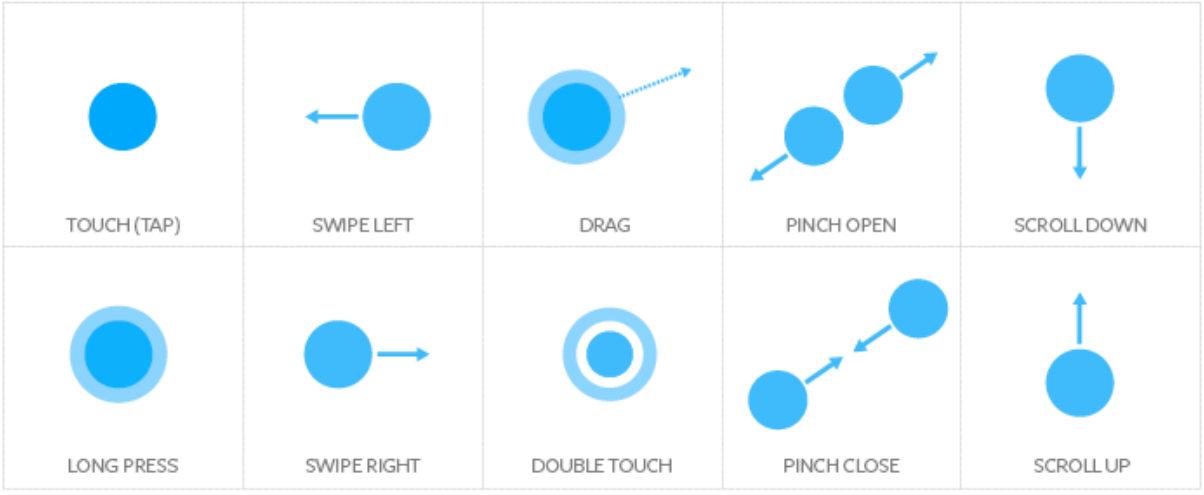
2019

Nội dung

1. Gestures và Touch Events
2. Gesture Detector
3. Double Tap Gesture
4. Swipe Gesture
5. Pinch Gesture

Gestures và Touch Events

- Nhận dạng các cử chỉ (**gestures**) và xử lý các sự kiện chạm (**touch events**) là một phần quan trọng trong lập trình xử lý các tương tác của người dùng. Các gestures thường gặp gồm:
  - **Touch (tap)**: cử chỉ 1 chạm màn hình bằng một ngón tay, tương tự click trên môi trường desktop
  - **Long press (long tap)**: cử chỉ dùng một ngón tay nhấn và giữ màn hình (touch & hold)
  - **Double tap**: cử chỉ dùng một ngón tay nhấn 2 lần liên tiếp lên màn hình
  - **Swipe (scroll)**: cử chỉ vuốt màn hình bằng một ngón tay, theo các hướng: từ trái sang phải (right), từ phải sang trái (left), từ trên xuống dưới (down), từ dưới lên trên (up)
  - **Pinch**: cử chỉ dùng 2 ngón tay thu vào (close) hoặc mở ra (open)
  - **Drag**: cử chỉ giữ và kéo di chuyển một view



Gestures và Touch Events

- Trung tâm của tất cả gestures là interface **OnTouchListener** và phương thức **onTouch**, nơi có thể truy cập vào dữ liệu **MotionEvent**. Mỗi view đều có **onTouchListener**:

```
myView.setOnTouchListener(new View.OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        return true;
    }
});
```

- Các gestures được minh hoạ trong bài học gồm:
  - **Swipe**
  - **Double tap**
  - **Pinch**

Gestures và Touch Events (2)

- Mỗi **onTouch** event có thể truy cập đến dữ liệu MotionEvent, mô tả **action code** và **axis values**
- **Action code** cho biết trạng thái thay đổi xảy ra như con trỏ di chuyển lên hay xuống
  - **getAction()** – trả về một hằng số, ví dụ: MotionEvent.ACTION_DOWN
- **Axis values** cho biết vị trí và các thuộc tính của sự di chuyển
  - **getX()** – trả về hoành độ của touch event
  - **getY()** – trả về tung độ của touch event
- **Gesture Detectors**: Bên trong **onTouch** event, có thể sử dụng **GestureDetector** để xác định gesture gì thông qua chuỗi các sự kiện chuyển động

Double Tap

- Sử dụng class **OnDoubleTapListener** được cung cấp tại: https://gist.github.com/nesquena/b2f023bb04190b2653c7
- Áp dụng vào ứng dụng:

```
myView.setOnTouchListener(new OnDoubleTapListener(this) {
    @Override
    public void onDoubleTap(MotionEvent e) {
        Toast.makeText(MainActivity.this, "Double Tap", Toast.LENGTH_SHORT).show();
    }
});
```

Class OnDoubleTapListener

```
public class OnDoubleTapListener implements View.OnTouchListener {
    private GestureDetector gestureDetector;

    public OnDoubleTapListener(Context c) {
        gestureDetector = new GestureDetector(c, new GestureListener());
    }

    public boolean onTouch(final View view, final MotionEvent motionEvent) {
        return gestureDetector.onTouchEvent(motionEvent);
    }

    private final class GestureListener extends GestureDetector.SimpleOnGestureListener {

        @Override
        public boolean onDown(MotionEvent e) {
            return true;
        }

        @Override
        public boolean onDoubleTap(MotionEvent e) {
            OnDoubleTapListener.this.onDoubleTap(e);
            return super.onDoubleTap(e);
        }
    }

    public void onDoubleTap(MotionEvent e) {
        // To be overridden when implementing listener
    }
}
```

Swipe Gesture

- Sử dụng class **OnSwipeListener** được cung cấp tại:
  https://gist.github.com/nesquena/ed58f34791da00da9751 và áp dụng vào ứng dụng:

```
myView.setOnTouchListener(new OnSwipeTouchListener(this) {
    @Override
    public void onSwipeDown() {
        Toast.makeText(MainActivity.this, "Down", Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onSwipeLeft() {
        Toast.makeText(MainActivity.this, "Left", Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onSwipeUp() {
        Toast.makeText(MainActivity.this, "Up", Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onSwipeRight() {
        Toast.makeText(MainActivity.this, "Right", Toast.LENGTH_SHORT).show();
    }
});
```

Class OnSwipeListener

```
public class OnSwipeTouchListener implements View.OnTouchListener {

    private GestureDetector gestureDetector;

    public OnSwipeTouchListener(Context c) {
        gestureDetector = new GestureDetector(c, new GestureListener());
    }

    public boolean onTouch(final View view, final MotionEvent motionEvent) {
        return gestureDetector.onTouchEvent(motionEvent);
    }

    public void onSwipeRight() {
    }

    public void onSwipeLeft() {
    }

    public void onSwipeUp() {
    }

    public void onSwipeDown() {
    }
}
```

OnSwipeListener (2)

```
private final class GestureListener extends GestureDetector.SimpleOnGestureListener {
    private static final int SWIPE_THRESHOLD = 100;
    private static final int SWIPE_VELOCITY_THRESHOLD = 100;

    @Override
    public boolean onDown(MotionEvent e) {
        return true;
    }

    // Determines the fling velocity and then fires the appropriate swipe event accordingly
    @Override
    public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {
        // …
    }
}
```

OnSwipeListener (3)

```
@Override
public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {
    boolean result = false;
    try {
        float diffY = e2.getY() - e1.getY();
        float diffX = e2.getX() - e1.getX();
        if (Math.abs(diffX) > Math.abs(diffY)) {
            if (Math.abs(diffX) > SWIPE_THRESHOLD && Math.abs(velocityX) > SWIPE_VELOCITY_THRESHOLD) {
                if (diffX > 0) {
                    onSwipeRight();
                } else {
                    onSwipeLeft();
                }
            }
        } else {
            if (Math.abs(diffY) > SWIPE_THRESHOLD && Math.abs(velocityY) > SWIPE_VELOCITY_THRESHOLD) {
                if (diffY > 0) {
                    onSwipeDown();
                } else {
                    onSwipeUp();
                }
            }
        }
    } catch (Exception exception) {
        exception.printStackTrace();
    }
    return result;
}
```

Pinch Gesture

- Xây dựng class **OnImageScaleListener** kế thừa từ **ScaleGestureDetector.SimpleOnScaleGestureListener**

```
public class OnImageScaleListener extends ScaleGestureDetector.SimpleOnScaleGestureListener {
    ImageView imageView; // hình ảnh xảy ra sự kiện pinch
    float factor; // tỉ lệ co giãn, mặc định là 1

    public OnImageScaleListener(ImageView iv) {
        super();
        imageView = iv;
        factor = 1.0f;
    }

    @Override
    public boolean onScale(ScaleGestureDetector detector) {
        float scaleFactor = detector.getScaleFactor() - 1;
        factor += scaleFactor; // xác định tỉ lệ mới
        imageView.setScaleX(factor); // thay đổi tỉ lệ của hình ảnh
        imageView.setScaleY(factor);
        return true;
    }
}
```

Pinch Gesture (2)

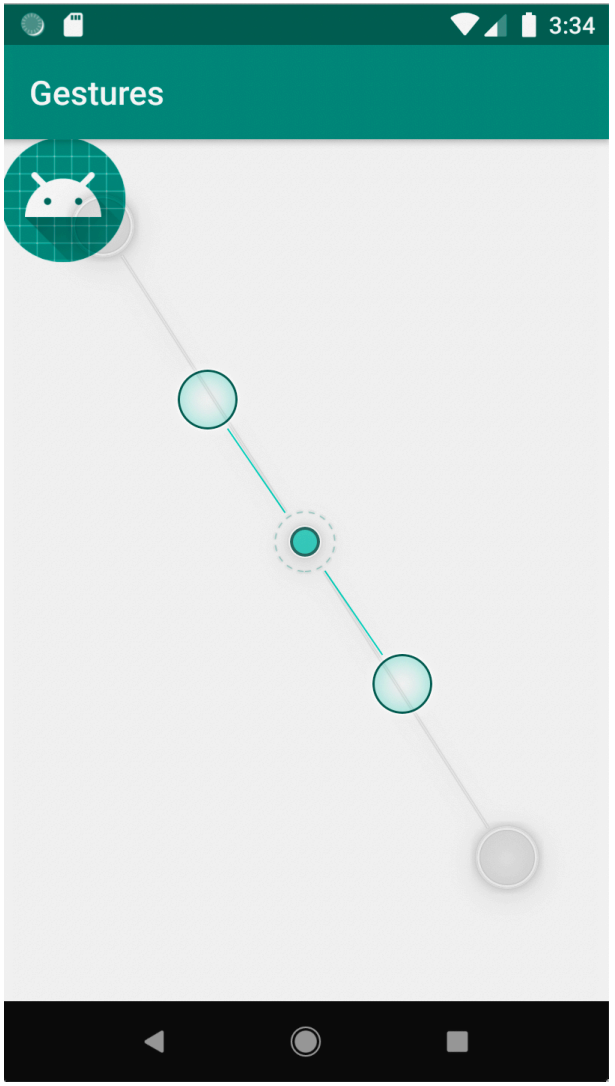
- Tạo đối tượng **ScaleGestureDetector** trong Activity/Fragment, đối tượng này sẽ xử lý sự kiện **onTouch** lên **ImageView**

```
public class MainActivity extends AppCompatActivity {
    ImageView imageView;
    ScaleGestureDetector scaleGestureDetector;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        myView = findViewById(R.id.myView);
        imageView = findViewById(R.id.imageView);

        scaleGestureDetector = new ScaleGestureDetector(this, new OnImageScaleListener(imageView));

        imageView.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View view, MotionEvent motionEvent) {
                return scaleGestureDetector.onTouchEvent(motionEvent);
            }
        });
    }
}
```

Pinch Gesture (3)

- Kết quả thử nghiệm:
  - Nhấn giữ phím **Ctrl** (Windows) hoặc **Cmd** (Mac)

  và kéo chuột để thực hiện cử chỉ **Pinch**



Q&A