الفصل العاشر:

التجاوب مع حركة الفأرة وانزلاق الإصبع.

الكلمات المفتاحية:

حركة الفأرة Mouse Gestures، الانزلاق Swiping.

ملخص:

نعرض في هذا الفصل كيفية التجاوب مع الانزلاق (الفأرة أو الإصبع).

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

- معالجة اللمس.
- معالجة الانزلاق

المخطط:

التجاوب مع حركة الفأرة وانزلاق الاصبع

- 2 وحدة (Learning Objects)

1. حركة الفارة وانزلاق الإصبع Mouse Gestures and Swiping

الأهداف التعليمية:

- التجاوب مع حركة الفأرة وانزلاق الإصبع

الانزلاق

- ندعو الانزلاق Swipe انزلاق الإصبع فوق الشاشة الشاشة (أو تحريك الفأرة مع استمرار الضغط على زر الفأرة). ويُستخدم عادةً في تطبيقات الموبايل للقبول accept/الرفض reject، المسح delete، الإخفاء dismiss.
- من أهم حركات الانزلاق:
  - يسار: لا no، سلبي negative، مسح delete.
  - يمين: نعم yes، إيجابي positive، تأكيد approve.

معالجة الانزلاق

- لا يدعم الأندرويد كثيراً الانزلاق، فهو يسمح بكشف اللمس touch والحركة motion. إلا أن عتبة (threshold) ما يُمكن اعتباره انزلاق وكيفية التجاوب معه متروكة للمبرمج. يُمكن استخدام الإجرائية onTouch:

```
public class MyActivity extends Activity
        implements OnTouchListener {
    @Override
    public boolean onTouch(View view, MotionEvent event) {
        ...
    }
}
```

- يُمكنك كتابة مستمع للحركة (gesture listener) لكشف الانزلاق:

```
class GestureHelper extends SimpleOnGestureListener {
@Override
public boolean onFling(MotionEvent e1, MotionEvent e2,
            float velocityX, float velocityY) {
// did the mouse move far enough, fast enough?
        ...
    }
}
```

- عليك استعمال كاشف حركة (gesture detector) في المستمع:

```
public class Name extends Activity implements OnTouchListener {
    private GestureDetector gesture;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        gesture = new GestureDetector(this,
```

```
                new GestureHelper());

    }

    @Override

    public boolean onTouchEvent(View v, MotionEvent e) {

        return gesture.onTouchEvent(v, e);

    }

}
```

- سنقوم في المثال التطبيقي التالي بكتابة مكتبة برمجية لتسهيل التعامل مع الانزلاق تحوي إجرائيات كشف الانزلاق ويُمكنك استخدامها في تطبيقاتك:

```
// write whichever of the onSwipe___ methods you want

view.setOnTouchListener(new OnSwipeListener(this) {

public void onSwipeLeft(float distance) { ... }

public void onSwipeRight(float distance) { ... }

public void onSwipeUp(float distance) { ... }

public void onSwipeDown(float distance) { ... }

        });
```

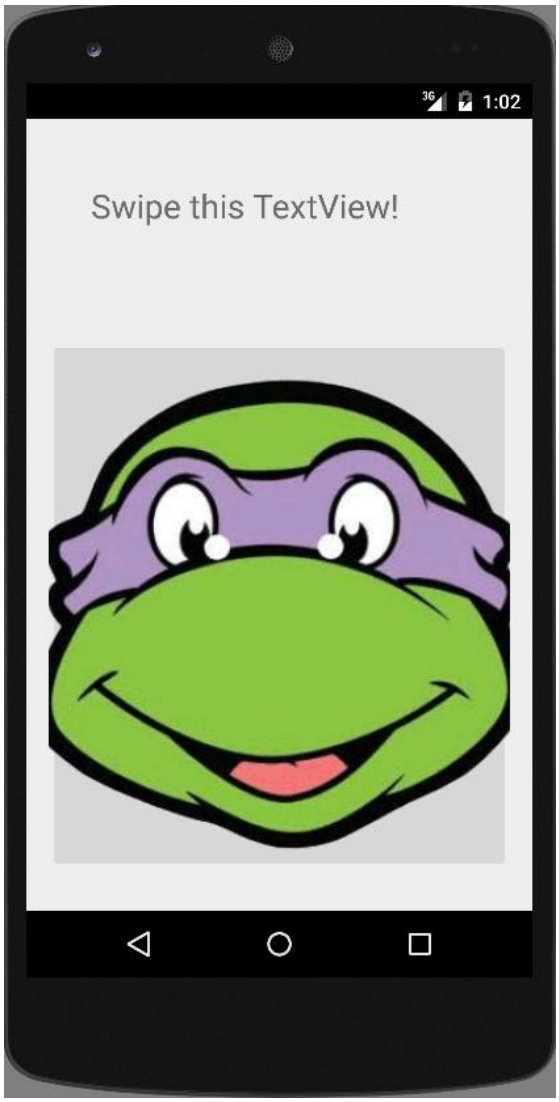
2. مثال تعليمي

الأهداف التعليمية:

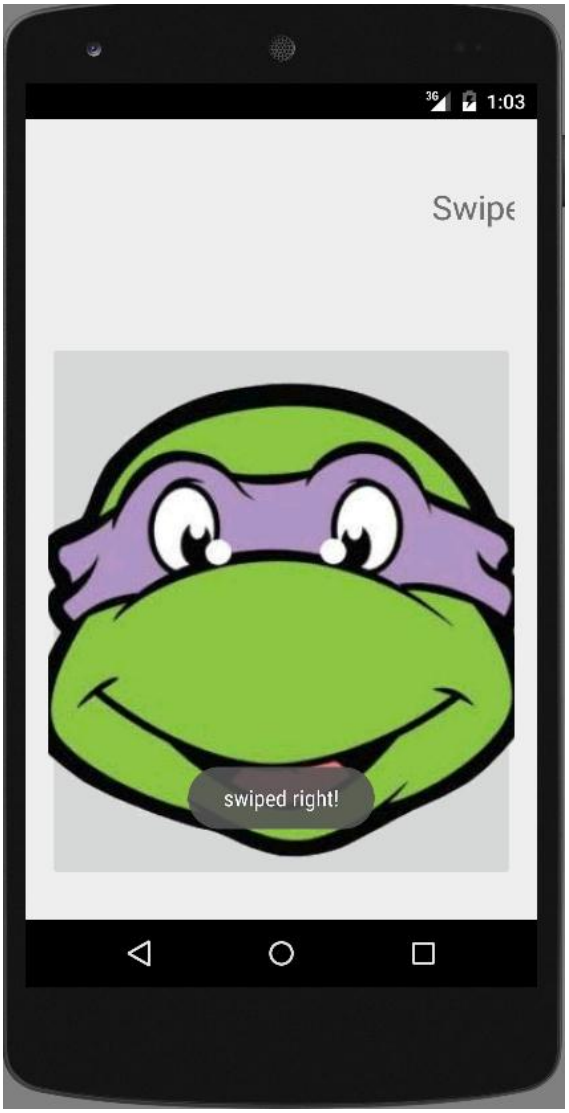
- مثال تعليمي

مثال تعليمي

- تُظهر واجهة التطبيق عارض نص (TextView) يُمكن زلقه بالإصبع (أو الفأرة) نحو اليمين أو نحو اليسار. وفي كلا الحالتين تظهر رسالة إعلام موافقة:



- بعد الانزلاق يميناً، تظهر رسالة إعلام موافقة:



- يكون ملف النشاط activity_swiping.xml:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center|top"
    android:paddingLeft="@dimen/activity_horizontal_margin"
```

```
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".SwipingActivity">
    <TextView
        android:id="@+id/textview1"
        android:text="Swipe this TextView!"
        android:textAlignment="center"
        android:textSize="24sp"
        android:padding="30dp"
        android:layout_marginBottom="50dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <ImageButton
        android:id="@+id/button1"
        android:src="@drawable/tmntdon"
        android:text="Swipe this button!"
        android:textSize="24sp"
        android:onClick="onClickButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

- يكون ملف الكود SwipingActivity.java:

```
package com.example.basel.swiping;
/*
```

```
 * This activity is a short demonstration of the provided OnSwipeListener library.
 * You can swipe a TextView left and right to trigger an event.
 */
import android.app.Activity;
import android.os.Bundle;
import android.view.*;
import android.widget.*;


public class SwipingActivity extends Activity {
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_swiping);


TextView textView = (TextView) findViewById(R.id.textview1);
textView.setOnTouchListener(
          new OnSwipeListener(this) {
                    {
                        setDragHorizontal(true);
                        setExitScreenOnSwipe(true);
                        setAnimationDelay(1000);
                    }


@Override
 public void onSwipeLeft(float distance) {
```

```
 Toast.makeText(SwipingActivity.this, "swiped left!",
Toast.LENGTH_SHORT).show();

                    }



@Override

public void onSwipeRight(float distance) {

                        Toast.makeText(SwipingActivity.this, "swiped
right!", Toast.LENGTH_SHORT).show();

                    }

                }

        );

    }



public void onClickButton1(View view) {

        Toast.makeText(this, "button 1 clicked",
Toast.LENGTH_SHORT).show();

    }

}
```

- يكون ملف المكتبة المساعدة للانزلاق OnSwipeListener.java:

```
package com.example.basel.swiping;

/*

   This class makes it easier to detect mouse/finger swipe
   motions on a view.

   An OnSwipeListener has several configuration 'set'
    methods that all return

   the listener itself so that they can be chained together.

   This code is based on an example found at the following URL:
```

```
   http://stackoverflow.com/questions/4139288/android-how-to-handle-right-to-left-swipe-gestures

*/


import android.content.*;

import android.content.res.*;

import android.view.*;


public class OnSwipeListener implements View.OnTouchListener {

    private boolean dragHorizontal = false;

    private boolean dragVertical = false;

    private boolean dragSnapBack = false;

    private boolean animated = true;

    private boolean exitScreenOnSwipe = false;

    private long animationDelay = 500;

    private float dragSnapThreshold = 50;

    private float swipeDistanceThreshold = 100;

    private float swipeVelocityThreshold = 100;

    private float dragPrevX;

    private float dragPrevY;

    private GestureDetector gestureDetector = null;

    private Impl swiper = null;

    private View draggedView = null;


    /**

     * Constructs a new listener for the
```

```
       given context (activity or fragment).

     */

public OnSwipeListener(Context context) {

gestureDetector = new GestureDetector(context, new GestureListener());

if (context instanceof Impl) {

             swiper = (Impl) context;

         }

    }



    /**

     * You can override this method if you want to
       subclass OnSwipeListener.

     * Called when the user swipes the view to the left.

     */

    public void onSwipeLeft(float distance) {

         if (swiper != null) {

             swiper.onSwipeLeft(distance);

         }

    }



    /**

     * You can override this method if you want to
       subclass OnSwipeListener.

     * Called when the user swipes the view to the right.

     */

    public void onSwipeRight(float distance) {

         if (swiper != null) {
```

```
            swiper.onSwipeRight(distance);

        }

    }



    /**

     * You can override this method if you
       want to subclass OnSwipeListener.

     * Called when the user swipes the view upward.

     */

    public void onSwipeUp(float distance) {

        if (swiper != null) {

            swiper.onSwipeUp(distance);

        }

    }



    /**

     * You can override this method if you
       want to subclass OnSwipeListener.

     * Called when the user swipes the view downward.

     */

    public void onSwipeDown(float distance) {

        if (swiper != null) {

            swiper.onSwipeDown(distance);

        }

    }



    /**
```

```
     * Internal method used to implement mouse touch events.
     * Not to be called directly by clients.
     */
@Override
public final boolean onTouch(View view, MotionEvent event) {
if (view != null) {
            draggedView = view;
}

boolean gesture = gestureDetector.onTouchEvent(event);

int action = event.getAction();
if (dragHorizontal || dragVertical) {
  if (action == MotionEvent.ACTION_DOWN ||
      action == MotionEvent.ACTION_POINTER_DOWN) {
                // initialViewX = view.getX();
                // initialViewY = view.getY();
   } else if (action == MotionEvent.ACTION_MOVE) {
                float dragCurrX = event.getRawX();
                float dragCurrY = event.getRawY();
                if (dragHorizontal) {
     view.setTranslationX(view.getTranslationX() + dragCurrX - dragPrevX);
                }
                if (dragVertical) {
     view.setTranslationY(view.getTranslationY() + dragCurrY - dragPrevY);
                }
```

```
    } else if (action == MotionEvent.ACTION_UP ||
               action == MotionEvent.ACTION_CANCEL ||
               action == MotionEvent.ACTION_POINTER_UP) {

               if (dragSnapBack) {

                   float dx = event.getRawX() - dragPrevX;

                   float dy = event.getRawY() - dragPrevY;



                   boolean shouldDoX = Math.abs(dx) <=
            dragSnapThreshold || dragSnapThreshold <= 0;

                   boolean shouldDoY = Math.abs(dy) <=
dragSnapThreshold || dragSnapThreshold <= 0;



               if (animated) {

                 ViewPropertyAnimator anim = view.animate();

                      if (shouldDoX) {

                          anim.translationX(0);

                      }

                      if (shouldDoY) {

                          anim.translationY(0);

                      }

                      anim.setDuration(animationDelay);

                      anim.start();

                  } else {

                      if (shouldDoX) {

                          view.setTranslationX(0);

                      }
```

```
                        if (shouldDoY) {
                            view.setTranslationY(0);
                        }
                    }
                }
            }
            dragPrevX = event.getRawX();
            dragPrevY = event.getRawY();
        }

        return gesture;
    }

    /**
     * Sets the number of pixels before the
         listener considers the user to have swiped.
     */
    public OnSwipeListener setDistanceThreshold(float px) {
        swipeDistanceThreshold = px;
        return this;
    }

    /**
     * Sets the rate of finger speed before the
        listener considers the user to have swiped.
     */
```

```
    public OnSwipeListener setVelocityThreshold(float px) {

        swipeVelocityThreshold = px;

        return this;

    }



    /**

     * Sets the number of pixels in which the
         view will "snap back" if dragged.

     */

    public OnSwipeListener setDragSnapThreshold(float px) {

        dragSnapThreshold = px;

        if (dragSnapThreshold > 0) {

            setDragSnapBack(true);

        }

        return this;

    }



    /**

     * Sets the number of milliseconds long
          that each drag/snap animation will take.

     */

public OnSwipeListener setAnimationDelay(long ms) {

        animationDelay = ms;

        setAnimated(animationDelay > 0);

        return this;

    }
```

```
    /**
     * Sets whether the view should slide itself off
           the screen once it has been swiped.
     */
public OnSwipeListener setExitScreenOnSwipe(boolean exit) {
        exitScreenOnSwipe = exit;
        return this;
    }

    /**
     * Sets whether the view should track the
            user's finger as it is dragged horizontally.
     */
public OnSwipeListener setDragHorizontal(boolean drag) {
        dragHorizontal = drag;
        return this;
    }

   /**
     * Sets whether the view should track the user's
                          finger as it is dragged vertically.
     */
public OnSwipeListener setDragVertical(boolean drag) {
        dragVertical = drag;
        return this;
    }
```

```
    /**
     * Sets whether the view should snap back into
                     position when the user stops dragging it.
     */
public OnSwipeListener setDragSnapBack(boolean snap) {
        dragSnapBack = snap;
        return this;
    }

    /**
     * Sets whether the view should animate itself
                   when it snaps back or slides off screen.
     */
public OnSwipeListener setAnimated(boolean anim) {
        animated = anim;
        return this;
    }

    /*
     * Internal class to implement finger gesture tracking.
     */
private class GestureListener extends
              GestureDetector.SimpleOnGestureListener {
        @Override
        public boolean onDown(MotionEvent e) {
            return true;
        }
```

```
@Override

public boolean onFling
      (MotionEvent e1, MotionEvent e2, float vx, float vy) {

            float dx = e2.getRawX() - e1.getRawX();

            float dy = e2.getRawY() - e1.getRawY();



            Configuration config =
draggedView.getContext().getApplicationContext().getResources().getC
onfiguration();

            int screenWidth = config.screenWidthDp;

            int screenHeight = config.screenHeightDp;



            if (Math.abs(dx) > Math.abs(dy)

                    && Math.abs(dx) > swipeDistanceThreshold

                    && Math.abs(vx) > swipeVelocityThreshold) {

            if (dx > 0) {

            onSwipeRight(dx);

            dragEdgeHelper(screenWidth * 2, true, 0, false);

           } else {

            onSwipeLeft(-dx);

            dragEdgeHelper(-screenWidth, true, 0, false);

                }

                return true;

            } else if (Math.abs(dy) > Math.abs(dx)

                 && Math.abs(dy) > swipeDistanceThreshold
```

```
                && Math.abs(vy) > swipeVelocityThreshold) {

            if (dy > 0) {

            onSwipeDown(dy);

            dragEdgeHelper
                      (0, false, screenHeight * 2, true);

              } else {

                  onSwipeUp(-dy);

                  dragEdgeHelper
                         (0, false, -screenHeight, true);

              }

              return true;

          }



          return false;

       }



private void dragEdgeHelper
      (float tx, boolean useTX, float ty, boolean useTY) {

          if (exitScreenOnSwipe && draggedView != null) {

              if (animated) {

                  ViewPropertyAnimator anim =
                                  draggedView.animate()

                          .setDuration(animationDelay);

                  if (useTX) {

                      anim.translationX(tx);

                  }
```

```
                    if (useTY) {
                        anim.translationY(ty);
                    }
                    anim.start();
                } else {
                    draggedView.setVisibility(View.INVISIBLE);
                }
            }
        }
    }

    /**
     * Interface that can be implemented by
           your activity/fragment to make it
     * possible to put the onSwipe___ methods
           directly in your activity class.
     */
    public static interface Impl {
        void onSwipeLeft(float distance);
        void onSwipeRight(float distance);
        void onSwipeUp(float distance);
        void onSwipeDown(float distance);
    }
}
```