LUKU 18

JOHDATUS ANDROID-OHJELMOINTIIN

Seuraavilla sivuilla esitellään joukko yksinkertaisia Android-ohjelmia.

2012-10-05 Tiedosto luotu.
2013-03-11 Viimeisin muutos.

MovingBall-sovellus: Button-olioita ja menuja käyttävä ohjelma

Seuraavilla sivuilla esitellään MovingBall-sovellus, joka koostuu sekä **.java**- että **.xml**-tiedostoista. Yleensä Android-sovellukset tulisi rakentaa siten että käyttöliittymään liittyvät asiat kuvataan XML-koodilla ja toiminnot rakennetaan Java-koodilla.

Tässä esiteltävä sovellus koostuu seuraavista tiedostoista, jotka esitellään myöhemmillä sivuilla:

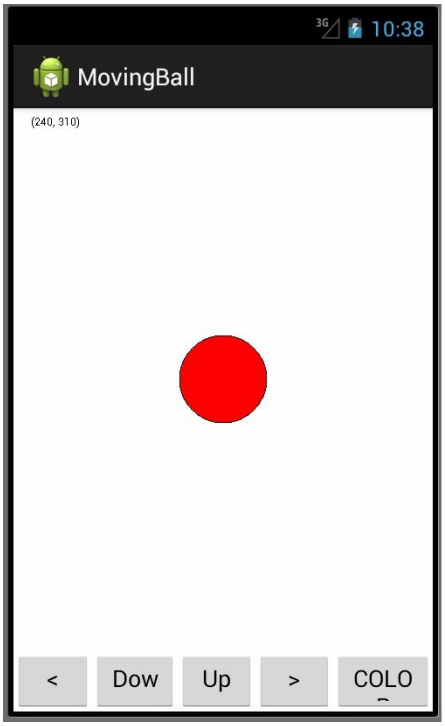
```
res/layout/main.xml
res/menu/color_selection_menu.xml
res/values/strings.xml
src/moving/ball/MovingBallActivity.java
src/moving/ball/MovingBallView.java
```

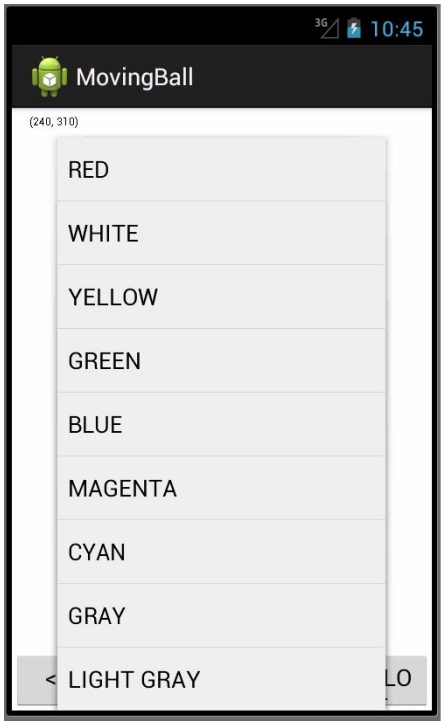
Yllä annettujen tiedostonimien eteen on kirjoitettu niiden kansioiden nimet joissa näiden tiedostojen tulee sijaita Android-projektin kansiohierarkiassa. **.java**-tiedostot tulee sijoittaa kansioihin jotka kuvastavat niiden `package`-määrittelyä. Edellä luetellut **.java**-tiedostot on sijoitettu mainittuihin kansioihin koska niiden alussa on määrittely

```
package moving.ball ;
```

Android-sovelluksiin kuuluu lisäksi olennaisena tiedosto **AndroidManifest.xml**, jonka ohjelmankehitysympäristö (esim. Eclipse) generoi automaattisesti projektille annettujen tietojen perusteella. Tämän projektin **AndroidManifest.xml** voi näyttää esim. seuraavalta:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="moving.ball"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MovingBallActivity" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```





Ylhäällä vasemmalla on ohjelman aloitusnäkymä. Käyttöliittymä on määritelty **main.xml**-tiedostossa. Käyttöliittymässä olevilla napeilla voidaan näytöllä olevaa palloa liikutella ja menujen avulla voidaan pallon väriä muuttaa.

Ylhäällä oikealla on Options-menu joka saadaan näkyviin Android-laitteen menunäppäimellä. (Tämmöistä näppäintä ei välttämättä tulevaisuuden Android-laitteissa ole.)

Vasemmalla on Context-menu joka saadaan näkyviin painamalla pitkään käyttöliittymän COLOR-nappia.

Molempien menujen sisältö on määritelty tiedostossa **color_selection_menu.xml**.

MovingBall-sovellusta suoritetaan tässä emulaattorissa.

Tämä XML-kuvaus määrittelee sovelluksen layoutin, eli sen miten eri oliot sijoittuvat näytölle. Sovelluksen pää-leyout on vertikaalinen eli pystysuora **`LinearLayout`**. Tuon pystysuoran lineaarisen layoutin sisällä on vielä vaakasuora **`LinearLayout`**, jonka sisällä on sovelluksessä käytettävät **`Button`**it eli painonappioliot.

Tässä viitataan XML-koodista Java-koodiin. **`MovingBallView`** on Java-luokka joka sijaitsee paketissa **`moving.ball`**. Tässä siis määritellään että sovelluksen näkymässä on **`MovingBallView`**-olio. Kun olio on näin määritelty, sitä ei tarvitse luoda Java-koodissa vaan ohjelman suoritusympäristö luo tämän olion automaattisesti. Tähän olioon päästään käsiksi Java-koodissa käyttämällä nimeä **`moving_ball_view`**.

```
<!--  main.xml for the MovingBall application. Copyright (c) Kari Laitinen
-->

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >

  <moving.ball.MovingBallView
      android:id="@+id/moving_ball_view"
      android:layout_width="fill_parent"
      android:layout_height="0dp"
      android:layout_weight="0.9"
  />


  <LinearLayout
      android:orientation="horizontal"
      android:layout_width="fill_parent"
      android:layout_height="0dp"
      android:layout_weight="0.1">
      <Button
          android:id="@+id/left_button"
          android:text=" &lt; "
          android:onClick="left_button_clicked"
          android:gravity="center_horizontal"
          android:layout_width="wrap_content"
          android:layout_height="fill_parent"
          android:layout_weight="1"/>
      <Button
          android:id="@+id/down_button"
          android:text="Down"
          android:onClick="down_button_clicked"
          android:gravity="center_horizontal"
          android:layout_width="wrap_content"
          android:layout_height="fill_parent"
          android:layout_weight="1"/>
```

Näillä lukuarvoilla saadaan aikaiseksi että painonapit vievät n. 10 % tilan näytön alareunasta. Kun **`layout_weight`** on määriteltynä tähän tapaan, tulee **`layout_height`** asettaa arvoon "0dp"

main.xml - 1: MovingBall-sovelluksen layout-tiedoston alkuosa.

```
        <Button
            android:id="@+id/up_button"
            android:text="Up"
            android:onClick="up_button_clicked"
            android:gravity="center_horizontal"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:layout_weight="1"/>
        <Button
            android:id="@+id/right_button"
            android:text=" &gt; "
            android:onClick="right_button_clicked"
            android:gravity="center_horizontal"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:layout_weight="1"/>
        <Button
            android:id="@+id/color_button"
            android:text="COLOR"
            android:gravity="center_horizontal"
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:layout_weight="1"/>
    </LinearLayout>

</LinearLayout>
```

Google suosittelee että Android-sovellusten käyttöliittymät määritellään XML-kuvauksilla kuten tässä on tehty. Käyttöliittymät voidaan toki tehdä myös Java-koodilla.

Sovelluksissa olevat merkkijonot eli stringit pitäisi kirjoittaa **strings.xml**-tiedostoon. Tätä periaatetta ei tässä sovelluksessa ole täysin noudatettu koska painonapeissa olevat tekstit on kuvattu tässä **main.xml**-tiedostossa. **strings.xml**-tiedoston käytöstä saadaan se hyöty että jos halutaan tehdä sovelluksiin useita kielivaihtoehtoja, voidaan kullekin kielelle tehdä oma XML-tiedosto johon kyseisen kielen stringit kirjoitetaan. Esim. jos halutaan sovellukseen vaihtoehtoisesti suomenkieliset tekstit, tehdään niitä varten oma **strings.xml** joka sijoitetaan projektiin seuraavasti: **res/values-fi/strings.xml**.

Painonappeja määriteltäessä jokaiselle `Button`-oliolle määritellään mm. `id` jolla siihen päästään tarvittaessa käsiksi Java-koodissa. Tässä määritellään että kun oikealle palloa siirtävää nappia painetaan, niin silloin kutsutaan Java-koodissa olevaa metodia nimeltä `right_button_clicked`. Tuo metodi tulee sijaita nimenomaan `Activity`-pohjaisessa luokassa.

XML-kielessä on muutamia 'varattuja' merkkejä. Kun näitä merkkejä halutaan käyttää esim. painonapin tekstissä, ne täytyy kirjoittaa merkkien & ja ; avulla seuraavasti:

```
&lt;    <  less than
&gt;    >  greater than
&amp;   &  ampersand
&apos;  '  apostrophe
&quot;  "  quotation mark
```

Eli jos painonappiin halutaan merkki `>`, sen paikalle täytyy kirjoittaa `>`

main.xml - 2. MovingBall-sovelluksen layout-tiedoston loppuosa.

XML-tiedoston alkuun voidaan laittaa tällainen rivi joka kuvaa käytetyn XML-version sekä merkkikoodauksen jota tiedostossa on käytetty. XML-tiedostot näyttävät Androidissa toimivan myös ilman tätä riviä.

```
<?xml version="1.0" encoding="utf-8"?>

<!-- color_selection_menu.xml  Copyright (c) Kari Laitinen

   2012-02-14 File created.
   2012-02-14 Last modification.

   This file specifies the menu with which the color
   can be changed in the MovingBall application.

   Inside Java code this menu is referred to with the name
   R.menu.color_selection_menu, i.e., the name of this file
   is used. Individual menu items are referred to with names
   such as red_color_item, which are given below.

-->

<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/red_color_item"
          android:title="@string/red"/>
    <item android:id="@+id/white_color_item"
          android:title="@string/white"/>
    <item android:id="@+id/yellow_color_item"
          android:title="@string/yellow"/>
    <item android:id="@+id/green_color_item"
          android:title="@string/green"/>
    <item android:id="@+id/blue_color_item"
          android:title="@string/blue"/>
    <item android:id="@+id/magenta_color_item"
          android:title="@string/magenta"/>
    <item android:id="@+id/cyan_color_item"
          android:title="@string/cyan"/>
    <item android:id="@+id/gray_color_item"
          android:title="@string/gray"/>
    <item android:id="@+id/light_gray_color_item"
          android:title="@string/light_gray"/>
</menu>
```

Tässä määritellään että tähän menuitemiin voidaan viitata Java-koodissa kirjoittamalla `R.id.cyan_color_item` ja menuitemin varsinainen teksti löytyy **strings.xml**-tiedostosta nimellä `cyan`.

color_selection_menu.xml. MovingBall-sovelluksen menun määrittely.

Tässä kaksi ensimmäistä stringiä ovat ohjelmankehitysympäristön automaattisesti generoimia. Loput ovat sovelluksen menussa käytettyjä stringejä. Esimerkiksi stringin jonka nimi on **`red`** sisältö on **`RED`**. Nimellä **`red`** voidaan tähän merkkijonoon viitata toisessa XML-tiedostossa tai Java-koodissa.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="hello">Hello World, MovingBallAndroid!</string>
   <string name="app_name">MovingBall</string>
   <string name="red">RED</string>
   <string name="white">WHITE</string>
   <string name="yellow">YELLOW</string>
   <string name="green">GREEN</string>
   <string name="blue">BLUE</string>
   <string name="magenta">MAGENTA</string>
   <string name="cyan">CYAN</string>
   <string name="gray">GRAY</string>
   <string name="light_gray">LIGHT GRAY</string>
</resources>
```

strings.xml. MovingBall-sovelluksen menussa käytetyt merkkijonot.

Tämän sovelluksen Java-osuus koostuu kahdesta erillisestä luokasta **`MovingBallActivity`** ja **`MovingBallView`**. Tämä Activity-pohjainen luokka on sovelluksen pääluokka. Luokan **`onCreate()`**-metodia kutsutaan automaattisesti silloin kun aktiviteetti luokaan, esim. silloin kun sovellus käynnistetään.

Yleensä **`onCreate()`**-metodin alussa määritellään mitä sovelluksen näytöllä on alussa. Tässä tapauksessa näyttöön pannaan näkymä johon viittaa **`R.layout.main`**. Käytännössä tämä tarkoittaa että **main.xml** määrittelee mitä näytöllä näytetään. Systeemi luo automaattisesti **`MovingBallView`**-olion ja **`Button`**-oliot jotka on määritelty **main.xml**-tiedostossa. **`R.layout.main`** on ohjelmointiympäristön automaattisesti generoima olioviittaaja joka on määritelty generoidussa tiedostossa **R.java**.

```
//  MovingBallActivity.java

package moving.ball ;

import android.app.Activity;
import android.app.AlertDialog;
import android.util.AttributeSet;
import android.os.Bundle;
import android.graphics.*;
import android.view.*;
import android.content.* ;
import android.widget.* ;   // Class Button, etc.
import android.widget.AdapterView.AdapterContextMenuInfo ;

public class MovingBallActivity extends Activity
{
   MovingBallView moving_ball_view;

   public void onCreate( Bundle savedInstanceState )
   {
      super.onCreate( savedInstanceState ) ;

      // The following call specifies that main.xml is the file
      // that specifies the content.

      setContentView( R.layout.main ) ;

      moving_ball_view = (MovingBallView) findViewById( R.id.moving_ball_view );

      // With the following call we get a reference to the Button
      // object that has been created based on the definition in main.xml.

      Button color_button = (Button) findViewById( R.id.color_button ) ;

      // with the following line we specify that the 'floating'
      // context menu must be shown when the 'color' button is long-pressed.

      registerForContextMenu( color_button ) ;
   }
```

MovingBallActivity.java - 1: Activityn onCreate()-metodi.

```
public void left_button_clicked( View view )
{
   moving_ball_view.move_ball_left() ;
}

public void down_button_clicked( View view )
{
   moving_ball_view.move_ball_down() ;
}

public void up_button_clicked( View view )
{
   moving_ball_view.move_ball_up() ;
}

public void right_button_clicked( View view )
{
   moving_ball_view.move_ball_right() ;
}

// The following method is called when the Options
// menu needs to be created. The definitions in
// res/menu/color_selection_menu.xml are used to
// create the menu.

public boolean onCreateOptionsMenu( Menu menu )
{
    super.onCreateOptionsMenu( menu ) ;
    MenuInflater inflater = getMenuInflater() ;
    inflater.inflate( R.menu.color_selection_menu, menu ) ;

    return true;
}

// In this program we have two menus which are specified with
// the same XML file.
// The following method is called when the 'floating'
// context menu needs to be created.

public void onCreateContextMenu( ContextMenu menu, View v,
                                 ContextMenu.ContextMenuInfo menuInfo )
{
   super.onCreateContextMenu( menu, v, menuInfo ) ;
   MenuInflater inflater = getMenuInflater() ;
   inflater.inflate( R.menu.color_selection_menu, menu ) ;
}
```

Jotain näistä neljästä metodista kutsutaan silloin kun käyttöliittymän pallonsiirtonappeja painetaan. Näiden metodien nimet on määritelty **main.xml**-tiedostossa samalla kun `Button`-oliot on siellä määritelty. Tässä Java-ohjelmassa emme siis ollenkaan tarvitse viittausia noihin pallonsiirtelyyn käytettäviin `Button`-olioihin.

Ohjelman ajosysteemi kutsuu näitä metodeita automaattisesti silloin kun täytyy luoda joko context-menu tai options-menu. Molempien menujen sisältö on määritelty tiedostossa **color_selection_menu.xml**, johon näissä metodeissa viitataan kirjoittamalla `R.menu.color_selection_menu`.

Näillä metodeilla ainoastaan luodaan menut. Menuvalintoihin reagoidaan jäljempänä olevilla metodeilla.

MovingBallActivity.java - 2: Varsinaisen midlettiluokan loppuosa.

Tätä metodia kutsutaan jäljempänä olevista kahdesta metodista jotka reagoivat menuvalintoihin.

Käytetyt id:t kuten `R.id.red_color_item` on systeemi luonut tiedostossa **color_selections_menu.xml** olevien määritysten perusteella. Luokassa `Color` määriteltyihin vakioväreihin voidaan viitata esim. kirjoittamalla `Color.RED`. Nämä vakiovärit ovat `int`-tyyppisiä arvoja jotka määrittelevät värin RGB-arvon.

```
//  The following method is used to process selections from
//  both menus, the options menu and the context menu.

protected boolean process_menu_selection( int menu_item_id )
{
   switch ( menu_item_id )
   {
      case R.id.red_color_item:
         moving_ball_view.set_ball_color( Color.RED ) ;
         return true;
      case R.id.white_color_item:
         moving_ball_view.set_ball_color( Color.WHITE ) ;
         return true;
      case R.id.yellow_color_item:
         moving_ball_view.set_ball_color( Color.YELLOW ) ;
         return true;
      case R.id.green_color_item:
         moving_ball_view.set_ball_color( Color.GREEN ) ;
         return true;
      case R.id.blue_color_item:
         moving_ball_view.set_ball_color( Color.BLUE ) ;
         return true;
      case R.id.magenta_color_item:
         moving_ball_view.set_ball_color( Color.MAGENTA ) ;
         return true;
      case R.id.cyan_color_item:
         moving_ball_view.set_ball_color( Color.CYAN ) ;
         return true;
      case R.id.gray_color_item:
         moving_ball_view.set_ball_color( Color.GRAY ) ;
         return true;
      case R.id.light_gray_color_item:
         moving_ball_view.set_ball_color( Color.LTGRAY ) ;
         return true;
      default:
         return false ;
   }
}
```

MovingBallActivity.java - 3: Menuvalintoja prosessoiva metodi.

Systeemi eli ohjelman ajoympäristö kutsuu alla olevia metodeita automaattisesti silloin kun jommasta kummasta menusta tehdään valinta. Menuvallinnat prosessoidaan edellä esitellyn metodin avulla.

Ao. metodit kutsuvat yläluokan vastaavia metodeita jos menuvalintoja ei jostain syysta saada prosessoiduksi.

```
   // The following method is called when a selection in
   // the Options menu has been made.

   public boolean onOptionsItemSelected( MenuItem menu_item )
   {
      if ( process_menu_selection( menu_item.getItemId() ) )
      {
         return true ;
      }
      else
      {
         return super.onOptionsItemSelected( menu_item ) ;
      }
   }

   // The following method is called when a selection in
   // the context menu has been made.

   public boolean onContextItemSelected( MenuItem menu_item )
   {
      if ( process_menu_selection( menu_item.getItemId() ) )
      {
         return true ;
      }
      else
      {
         return super.onContextItemSelected( menu_item ) ;
      }
   }
}
```

Näille metodeille tulee parametrina viittaus siihen **`MenuItem`**-olioon joka menusta tuli valituksi. **`getItemId()`** puolestaan kaivaa **`MenuItem`**-olion sisältä sen **`int`**-tyyppisen id:n.

MovingBallActivity.java - 4. Activity-pohjaisen luokan loppuosa.

Android-käyttöjärjestelmässä **View** on luokka joka toimii yläluokkana useille käyttöliittymäkomponenteille. **View**-luokasta johdetaan aliluokka kun halutaan tehdä käyttöliittymään olio johon halutaan itse tehdä piirtämisoperaatioita.

Tässä sovelluksessa **MovingBallView**-luokan olio on määritelty **main.xml**-tiedostossa käyttöliittymässä näkyväksi olioksi, ja tämä olio luodaan automaattisesti kun sovellus käynnistyy. Tämmöisiä XML-tiedostossa määriteltyjä olioita varten täytyy **View**-pohjaisiin luokkiin kirjoittaa tämänkaltainen konstruktori.

```
//  MovingBallView.java by Kari Laitinen

package moving.ball ;

import android.graphics.* ;
import android.view.* ;
import android.content.Context ;
import android.util.AttributeSet ;

public class MovingBallView extends View
{
   int ball_center_point_x = 100 ;
   int ball_center_point_y = 100 ;

   int ball_color  =  Color.RED ;

   public MovingBallView( Context context )
   {
      super( context ) ;
   }

   //  The following constructor is needed when MovingBallView object is
   //  specified in an XML file, and is thus created automatically.

   public MovingBallView( Context context, AttributeSet attributes )
   {
      super( context, attributes ) ;
   }

   public void onSizeChanged( int current_width_of_this_view,
                              int current_height_of_this_view,
                              int old_width_of_this_view,
                              int old_height_of_this_view )
   {
      ball_center_point_x  = current_width_of_this_view / 2 ;
      ball_center_point_y  = current_height_of_this_view  / 2 ;
   }
```

Käyttöjärjestelmä kutsuu automaattisesti **onSizeChanged**()-metodia silloin kun **View**-pohjaisen olion koko näytöllä muuttuu. Tämän perusteella olio voi ’säätää’ itseänsä tarpeen mukaan. Tässä sovelluksessa pallo pistetään keskelle näyttöaluetta jos sen koko muuttuu.

MovingBallView.java - 1: View-pohjaisen luokan alku.

Metodilla nimeltä **`invalidate()`** ilmoitetaan käyttöjärjestelmälle että tämän **View**in alue näytöllä ei ole enää 'validi' eli se pitää 'korjata'. Käyttöjärjestelmä kutsuu sitten **`onDraw()`**-metodia joka piirtää alueen sellaiseksi kuin sen pitää olla.

Kaikkia tällä sivulla olevia metodeita kutsutaan **`MovingBall-Activity`**-luokasta.

```
public void move_ball_left()
{
   ball_center_point_x -= 3 ;
   invalidate() ;
}

public void move_ball_down()
{
   ball_center_point_y += 3 ;
   invalidate() ;
}

public void move_ball_up()
{
   ball_center_point_y -= 3 ;
   invalidate() ;
}

public void move_ball_right()
{
   ball_center_point_x += 3 ;
   invalidate() ;
}

public void set_ball_color( int new_color )
{
   ball_color = new_color ;
   invalidate() ;
}
```

Värit ilmoitetaan Androidissa yleensä **`int`**-arvoina joiden sisältö on muotoa 0xAARRGGBB, eli 32-bittinen arvo jakaantuu neljäksi tavuksi joista ensimmäinen ilmoittaa ns. Alpha-arvon eli värin peittävyyden, ja loput tavut kuvaavat punaisen (R), vihreän (G) ja sinisen (B) määrän. **`Color`**-luokassa on määritelty valmiita vakioarvoja väreille kuten esim. **`Color.RED`**.

Androidissa ei käytetä samanlaista Color-luokkaa kuin standardi-Javassa.

MovingBallView - 2: Pallon liikutteluun ja värinvaihtoon tarkoitetut metodit.

Metodi nimeltä **`onDraw()`** on **`View`**-pohjaisissa luokissa se metodi jota käyttöjärjestelmä kutsuu silloin kun näytön sisältö on päivitettävä. Sille tulee parametrina **`Canvas`**-olio ja **`Canvas`**-luokka tarjoaa varsinaiset piirtometodit sovelluksen käyttöön.

Piirto-operaatioissa voidaan määritellä erilaisia maaleja eli **`Paint`**-olioita piirtämistä varten. Maaleille voidaan määritellä esimerkiksi väri ja omainaisuus että se täyttää piirrettävän kuvion.

Tässä kutsutaan **`drawCircle()`**-metodia jolle annetaan parametrina ympyrän keskipiste ja sen säde. Kun maalina käytetään täyttävää maalia, ympyrä tulee täytetyksi käytetyllä maalilla.

```
   protected void onDraw( Canvas canvas )
   {
      Paint background_paint = new Paint() ;
      background_paint.setColor( Color.WHITE ) ;

      canvas.drawPaint( background_paint ) ; // This fills the entire canvas.

      Paint filling_paint = new Paint() ;
      filling_paint.setStyle( Paint.Style.FILL ) ;
      filling_paint.setColor( ball_color ) ;

      canvas.drawCircle( ball_center_point_x,
                         ball_center_point_y, 50, filling_paint ) ;

      Paint outline_paint = new Paint() ;
      outline_paint.setStyle( Paint.Style.STROKE ) ;
      // Default color for a Paint is black.

      canvas.drawCircle( ball_center_point_x,
                         ball_center_point_y, 50, outline_paint ) ;

      canvas.drawText( "("   +  ball_center_point_x  +
                       ", "  +  ball_center_point_y  + ")",
                       20, 20, outline_paint ) ;

   }
}
```

MovingBallView.java - 3. Luokan lopussa oleva onDraw()-metodi.

GesturesDemo – Kosketusnäytön eleisiin reagointi

Tämä ohjelma demonstroi kuinka saadaan selville kosketusnäytöllä tehtävät eleet (gestures). Näitä eleitä ovat esimerkiksi eripituiset näytön painallukset, skrollaus ja flingaus. Tämän ohjelman molemmat luokat ovat tässä samassa tiedostossa.

Eleisiin regoimiseksi luokan täytyy toteuttaa **`GestureDetector.OnGestureListener`**-rajapinta, joka käsittää metodit joita kutsutaan silloin kun eri eleitä havaitaan tehdyn.

```
//  GesturesDemoActivity.java  Copyright (c) Kari Laitinen

package gestures.demo ;

import android.app.Activity ;
import android.os.Bundle ;
import android.graphics.* ;  // Classes Canvas, Color, Paint, RectF, etc.
import android.view.View ;
import android.view.MotionEvent ;
import android.view.GestureDetector ;
import android.content.Context ;
import java.util.ArrayList ;


final class GesturesDemoView extends View
                             implements GestureDetector.OnGestureListener
{
   int  view_width, view_height ;

   GestureDetector gesture_detector ;

   ArrayList<String> text_lines_to_screen = new ArrayList<String>() ;

   public GesturesDemoView( Context context )
   {
      super( context ) ;

      gesture_detector = new GestureDetector( context, this ) ;

      text_lines_to_screen.add( "With this application, you can" ) ;
      text_lines_to_screen.add( "explore what gesture events" ) ;
      text_lines_to_screen.add( "take palce when you play" ) ;
      text_lines_to_screen.add( "with the touch screen." ) ;
      text_lines_to_screen.add( "" ) ;
   }
```

Mainitun rajapinnan lisäksi eleisiin reagoimiseksi tulee käyttää **`GestureDetector`**-oliota, jolle syötetään kaikki kosketusnäytön tapahtumat. **`GestureDetector`**-olio osaa sitten nimensä mukaisesti ottaa selville muodostuuko näytön tapahtumista jokin määrätty ele.

GesturesDemoActivity.java - 1: GesturesDemoView-luokan alkuosa.

```
public boolean onDown( MotionEvent motion_event )
{
   text_lines_to_screen.add( "onDown()" ) ;

   invalidate() ;

   return true ;
}

public boolean onFling( MotionEvent first_down_motion,
                        MotionEvent last_move_motion,
                        float velocity_x, float velocity_y )
{
   text_lines_to_screen.add( "onFling()" ) ;

   invalidate() ;

   return true ;
}

public void onLongPress( MotionEvent first_down_motion )
{
   text_lines_to_screen.add( "onLongPress()" ) ;

   invalidate() ;
}

public boolean onScroll( MotionEvent first_down_motion,
                         MotionEvent last_move_motion,
                         float distance_x, float distance_y )
{
   text_lines_to_screen.add( "onScroll()" ) ;

   invalidate() ;

   return true ;
}
```

Tämän sivun metodit ja seuraavan sivun kaksi ensimmäistä metodia toteuttavat **`GestureDetector.OnGestureListener`**-rajapinnan. Näille kaikille metodeille tulee parametrina **`MotionEvent`**-olio tai kaksi oliota joiden perusteella saadaan selville mm. eleisiin liittyvät koskeusnäytön koordinaatit. **`MotionEvent`**-olioille on käytössä mm. metodit **`getX()`** ja **`getY()`** jotka palauttavat **`float`**-arvoina eleeseen liittyvän pisteen koordinaatit.

Metodeita **`onScroll()`** ja **`onFling()`** kutsutaan silloin kun kosketusnäyttöä pyyhkäistään. **`onScroll()`**-tapahtumia syntyy silloin kun näyttöä pyyhkäistään 'sievästi' tasaisella vauhdilla. **`onScroll()`**-tapahtumien perään syntyy **`onFling()`**-tapahtuma silloin kun pyyhkäisy on hiukan kiihtyvä ja sormi nostetaan kosketusnäytöltä.

Metodille **`onFling()`** tulee parametrina kaksi **`MotionEvent`**-oliota joista ensimmäinen kertoo pisteen jossa sormi pantiin näytölle ja toinen kertoo pisteen jossa sormi nostettiin näytöltä. Näiden olioiden avulla voidaan saada selville mm. mihin suuntaan 'flingaus' tapahtui näytöllä.

GesturesDemoActivity.java - 2: GestureDetector.OnGestureListener-rajapinnan metodeita.

Rajapinnassa on erilaisia painalluksiin liittyviä metodeita. **onDown()**-tapahtuma syntyy kun sormi koskettaa näyttöä. **onShowPress()**-tapahtuma tarkoitaa 'keskipitkää' näytön painallusta. Jos sormea pidetään näytöllä hiukan pidempään syntyy **onLongPress()**. Jos sormi otetaan näytöltä ennenkuin syntyy **onLongPress()**, generoituu **onSingleTapUp()**-tapahtuma.

Tässä ohjelmassa havaitaan kaikki kosketusnäytön eleet ja niistä generoituvat tapahtumat, mutta niille ei tehdä mitään sen kummempaa kuin että jokainen tapahtuma rekisteröidään **ArrayList**-taulukkoon tekstirivinä. Kun näkymä ('tämä' olio) invalidoidaan, **ArrayList**-pohjainen tapahtumalista tulostuu reaaliaikaisesti näytölle.

```
public void onShowPress( MotionEvent down_motion )
{
   text_lines_to_screen.add( "onShowPress()" ) ;

   invalidate() ;
}

public boolean onSingleTapUp( MotionEvent up_motion )
{
   text_lines_to_screen.add( "onSingleTapUp()" ) ;

   invalidate() ;
   return true ;
}


public void onSizeChanged( int current_width_of_this_view,
                           int current_height_of_this_view,
                           int old_width_of_this_view,
                           int old_height_of_this_view )
{
   view_width  = current_width_of_this_view ;
   view_height = current_height_of_this_view ;

}

public boolean onTouchEvent ( MotionEvent motion_event )
{
   gesture_detector.onTouchEvent( motion_event ) ;

   return true ;
}
```

Metodi **onTouchEvent()** on **View**-luokasta peritty ja tässä luokassa uudelleenkirjoitettu metodi joka käsittelee kaikki kosketusnäyttöön liittyvät tapahtumat. Jotta eleiden tunnistus saadaan toimimaan, kaikki **MotionEvent**-tapahtumat täytyy tähän tapaan siirtää **GestureDetector**ille käsiteltäväksi. **GestureDetector**in toiminnan seurauksena generoituu sitten kutsuja edellä oleviin eleisiin reagoiviin metodeihin.

GesturesDemoActivity.java - 3: Lisää GestureDetector.OnGestureListener-metodeita yms.

Tässä tuhotaan **ArrayList**-taulukon alusta niin paljon tekstirivejä että taulukkoon jää vain 10 viimeisintä riviä. Näin näytölle tulostetaan vain 10 viimeisimmän tapahtuman tieto.

```
    protected void onDraw( Canvas canvas )
    {
       Paint background_paint = new Paint() ;
       background_paint.setColor( Color.WHITE ) ;
       Paint outline_paint = new Paint() ;
       outline_paint.setStyle( Paint.Style.STROKE ) ;

       canvas.drawPaint( background_paint ) ; // This fills the entire canvas.

       while ( text_lines_to_screen.size() > 10 )
       {
          text_lines_to_screen.remove( 0 ) ;
       }

       for ( int line_index  =  0 ;
                 line_index  <  text_lines_to_screen.size() ;
                 line_index  ++ )
       {
          int text_color = Color.BLACK ;

          String text_line_to_screen = text_lines_to_screen.get( line_index ) ;

          if ( text_line_to_screen.equals( "onFling()" ) )
          {
             text_color = Color.BLUE ;
          }
          else if ( text_line_to_screen.equals( "onLongPress()" ) )
          {
             text_color = Color.RED ;
          }
          else if ( text_line_to_screen.equals( "onScroll()" ) )
          {
             text_color = Color.GREEN ;
          }
          else if ( text_line_to_screen.equals( "onShowPress()" ) )
          {
             text_color = Color.MAGENTA ;
          }
          else if ( text_line_to_screen.equals( "onSingleTapUp()" ) )
          {
             text_color = Color.CYAN ;
          }

          outline_paint.setColor( text_color ) ;
          canvas.drawText( text_line_to_screen,
                           20,  20 + 20 * line_index, outline_paint ) ;
       }
    }
}
```

GesturesDemoActivity.java - 4: onDraw()-metodi GesturesDemoView-luokan lopussa.

```
public class GesturesDemoActivity extends Activity
{
   /** Called when the activity is first created. */

   public void onCreate( Bundle savedInstanceState )
   {
      super.onCreate( savedInstanceState ) ;

      setContentView( new GesturesDemoView( this ) ) ;
   }
}
```

Android-sovelluksen **`Activity`**-pohjainen luokka voi olla lyhyt siinä tapauksessa kun XML-koodissa ei ole tehty ollenkaan käyttöliittymään liittyviä määrityksiä.

Kun tämä ohjelma käynnistyy, luodaan luokan **`GesturesDemoView`** olio ja pannaan se näytön sisällöksi. Tälle ohjelmalle riittää XML-layout-koodiksi ohjelmointiympäristön (esim. Eclipse) automaattisesti generoima koodinpätkä.

GesturesDemoActivity.java - 5. Lopussa oleva lyhyt GesturesDemoActivity-luokka.

Ohjelma tulostaa näytölle 10 viimeisintä kosketusnäytön eleisiin liittyvää tapahtumaa. Näitä tapahtumia voi syntyä melkoinen joukko jo yhdellä näytön pyyhkäisyllä.

Tämän ohjelman toiminnasta saa parhaan käsityksen kun käyttää sitä aidossa Android-laitteessa.

GesturesDemo-ohjelma toimii tässä emulaattorissa.