الفصل التاسع:

الرسومات Graphics والإحياء Animations

الكلمات المفتاحية:

رسومات Graphics، النياسب Threads، الإحياء Animation، العرض المخصص Custom View، الفرشاة Paint، محارف الرسم Typeface، الصور النقطية Bitmap images.

ملخص:

نتعرض في هذا الفصل لكيفية رسم الأشكال وإلى آليات التحريك.

أهداف تعليمية:

يتعرف الطالب في هذا الفصل على:

- طرق الرسم.
- طرق التحريك.
- استخدام النياسب.
- معالجة أحداث الفأرة واللمس.
- معالجة أحداث لوحة المفاتيح.

المخطط:

الرسومات Graphics والإحياء Animations

- 3 وحدات (Learning Objects)

1. الرسومات Graphics

الأهداف التعليمية:

- الرسومات Graphics

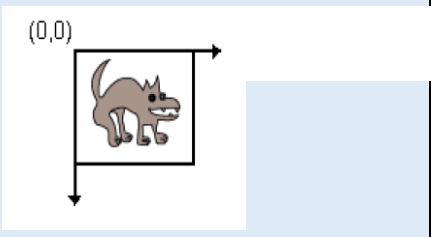
الرسومات Graphics

- كي تتمكن من رسم رسومات ثنائية البعد على الشاشة، عليك تعريف صف مشتق مخصص من صف العرض View.

قالب العرض المخصص Custom View Template

- يكون قالب العرض المخصص مشتق من الصف View ويحوي الإجرائية onDraw لكتابة الكود اللازم للرسم:

```
public class ClassName extends View {

    // required constructor

public ClassName(Context context,
AttributeSet attrs) {
        super(context, attrs);
    }
    // this method draws on the view
    @Override

    protected void onDraw(Canvas canvas) {


     super.onDraw(canvas);


     drawing code;

    }

}

// recall: y-axis increases downward!
```



استخدام العرض المخصص

- يُمكنك إدراج عرضك المخصص في ملف النشاط XML:

```
<!-- res/layout/activity_main.xml -->

<RelativeLayout ...

    tools:context=".MainActivity">

<packageName.ClassName

android:layout_width="match_parent"

android:layout_height="match_parent"

    ...

    />

</RelativeLayout>
```

طرق الرسم Canvas Object Methods

- يتم الرسم باستخدام إحدى طرق الرسم المُعرّفة في الصف Canvas.
- نُبين فيما يلي أهم طرق الرسم:

- ***c***.drawARGB(***alpha***, ***r***, ***g***, ***b***); – fill window with color (rgb=0–255)
- ***c***.drawArc(…); – draw a partial ellipse
- ***c***.drawBitmap(***bmp***, ***x***, ***y***, null); – draw an image
- ***c***.drawCircle(***centerX***, ***centerY***, ***r***, ***paint***); – draw a circle
- ***c***.drawLine(***x1***, ***y1***, ***x2***, ***y2***, ***paint***); – draw a line segment
- ***c***.drawOval(***x1***, ***y1***, ***x2***, ***y2***, ***paint***);
- ***c***.drawOval(new RectF(***x1***, ***y1***, ***x2***, ***y2***), ***paint***); – draw oval/circle
- ***c***.drawPoint(***x***, ***y***, ***paint***); – color a single pixel
- ***c***.drawRect(***x1***, ***y1***, ***x2***, ***y2***, ***paint***);
- ***c***.drawRect(new RectF(***x1***, ***y1***, ***x2***, ***y2***), ***paint***); – draw rectangle
- ***c***.drawRoundRect(***x1***, ***y1***, ***x2***, ***y2***, ***rx***, ***ry***, ***paint***);
- ***c***.drawRoundRect(new RectF(***x1***, ***y1***, ***x2***, ***y2***), ***rx***, ***ry***, ***paint***);
- ***c***.drawText("***str***", ***x***, ***y***, ***paint***); – draw a text string
- ***c***.getWidth(), ***c***.getHeight() – get dimensions of drawing area

فرشاة الرسم Paint

- تحتاج العديد من طرق الرسم إلى فرشاة Paint. تُبين التعليمات التالية إعداد الفرشاة:

```
Paint name = new Paint();

name.setARGB(alpha, red, green, blue);

// example

Paint purple = new Paint();

purple.setARGB(255, 255, 0, 255);

purple.setStyle(Style.FILL_AND_STROKE); // FILL, STROKE
```

- للفرشاة العديد من الطرق:

getTextBounds, measureText, setAlpha, setAntiAlias, setStrokeWidth, setStyle, setTextAlign, setTextSize, setTypeface

محارف الكتابة Typeface

- تُدعى محارف الكتابة في أندرويد Typeface. يتم إعداد المحارف باستخدام فرشاة كما تُبين التعليمات التالية:

```
// example: use a 40-point monospaced blue font

Paint p = new Paint();

p.setTypeface(

   Typeface.create(Typeface.MONOSPACE, Typeface.BOLD));

p.setTextSize(40);

p.setARGB(255, 0, 0, 255);
```

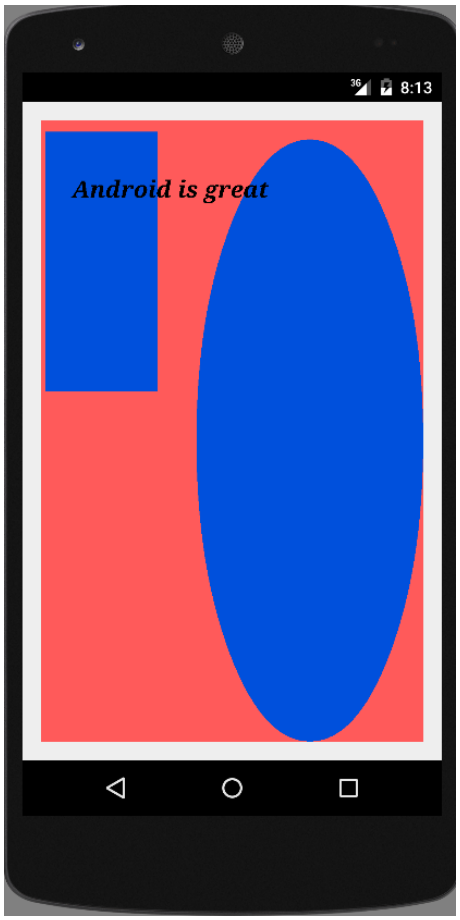
الصور النقطية Bitmap Images

يُمكن رسم صورة باستخدام الصف Bitmap أو قراءتها من ملف. كما تُبين التعليمات التالية:

```
// example: draw heart.png on screen at (0, 0)
Bitmap bmp = BitmapFactory.decodeResource(
        getResources(), R.drawable.heart);
        canvas.drawBitmap(bmp, 0, 0, null);
// you can also read a Bitmap from an input stream
URL url = new URL("http://example.com/myImage.jpg");
Bitmap bmp = BitmapFactory.decodeStream(url.openStream());
```

مثال تطبيقي

يُبين المثال التالي بعض تعليمات الرسم. حيث يُظهر التنفيذ:



يكون ملف النشاط XML:

```
<RelativeLayout

    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: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=".TargetsActivity">

<com.example.basel.targets.ExampleView

        android:layout_width="match_parent"

        android:layout_height="match_parent" />

</RelativeLayout>
```

ويكون ملف الكود:

```
package com.example.basel.targets;

/*
 * This class is a graphical view of a basic example of 2D graphics.
 */
import android.content.Context;

import android.graphics.*;

import android.util.AttributeSet;

import android.view.View;

public class ExampleView extends View {

public ExampleView(Context context, AttributeSet attrs) {

        super(context, attrs);

    }

/*

* This method draws some shapes and text on the view.

*/

@Override

protected void onDraw(Canvas canvas) {

        super.onDraw(canvas);



canvas.drawARGB(255, 255, 90, 90);



Paint aqua = new Paint();

aqua.setARGB(255, 0, 80, 220);
```

```
canvas.drawRect(new RectF(10, 30, 300, 700), aqua);


canvas.drawOval(new RectF(400, 50, getWidth(), getHeight()), aqua);



Paint font = new Paint();

font.setARGB(255, 0, 0, 0);

font.setTypeface(Typeface.create(Typeface.SERIF,
Typeface.BOLD_ITALIC));

font.setTextSize(60);



canvas.drawText("Android is great", 80, 200, font);

    }

}
```

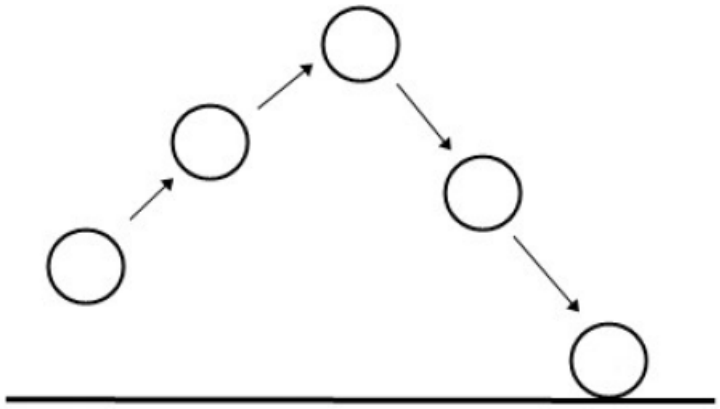
2. الإحياء Animation

الأهداف التعليمية:

- الإحياء Animation

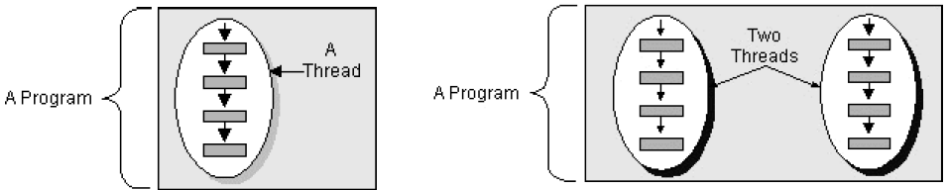
الإحياء Animations

- يتطلب إحياء العرض View إعادة الرسم في مجالات زمنية منتظمة يتم في كل منها تغيير أماكن الأغراض (تحريك).
- يتم الطلب من العرض بإعادة رسم نفسه باستخدام طريقته `invalidate`.
- لا يُمكن القيام بذلك باستخدام حلقة (loop) لأن ذلك سيؤدي إلى تجميد التطبيق.
- إذا أردت استخدام الإحياء (في الألعاب مثلاً)، فعليك استخدام نيسب thread لتعديل الرسومات وإعادة الرسم بشكل دوري.



النياسب Threads

- يكون النيسب thread عبارة عن تسلسل من تنفيذ بعض المهام الجزئية ضمن برنامج واحد. وهو طريقة لكتابة برامج تبدو وكأنها تقوم بمهام متعددة بشكل متزامن (simultaneously, concurrency).
- تتشارك النياسب لنفس المعالج البيانات مع بعضها البعض. فمثلاً، يُمكن لنيسب الوصول إلى متحولات نيسب آخر.



- يتم استخدام النيسب بتمرير غرض من النمط Runnable مع الإجرائية Run التي تحوي الكود المطلوب تنفيذه:

```
Thread thread = new Thread(new Runnable() {

public void run() {

// code to execute in thread goes here

        }

});

thread.start();
```

- من طرق النيسب الأخرى:

start, stop, sleep, isRunning, join

إعادة رسم العرض في نيسب

- لا يُمكن في أندرويد إنشاء نيسب واستدعاء الطريقة invalidate على العرض (View) من هذا النيسب. بل يجب إنشاء غرض معالج (Handler) للقيام لهذا الاستدعاء مما يتطلب وجود غرض ثاني Runnable.
- تُبين التعليمات التالية طريقة إسناد نيسب لإعادة الرسم:

```
// repaint the view a single time, in another thread

Thread thread = new Thread(new Runnable() {

public void run() {

        Handler h = new Handler(Looper.getMainLooper());

        handler.post(new Runnable() {

     public void run() {

            myView.invalidate();

          }

   });

  }

});

thread.start();
```

معالجة أحداث الفأرة واللمس Mouse and Touch Events

- للتجاوب مع إصبع المستخدم، قم بكتابة الطريقة onTouchEvent في صف العرض المخصص.

وحيث تكون الأفعال:

ACTION_DOWN, ACTION_UP, ACTION_MOVE, …

كما تُبين التعليمات التالية:

```
@Override
public boolean onTouchEvent(MotionEvent event) {
        float x = event.getX();
        float y = event.getY();
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
      // code to run when finger is pressed
        }
   return super.onTouchEvent(event);
 }
```

معالجة أحداث لوحة المفاتيح Keyboard Events

- للتجاوب مع ضغط المفاتيح من لوحة المفاتيح:
  - قم أولاً في باني (constructor) العرض المخصص بإعداد التطبيق ليستقبل التركيز (focus) في العرض:

```
requestFocus();

setFocusableInTouchMode(true);
```

  - قم بكتابة الإجرائية onKeyDown/Up في العرض المخصص. يكون لكل مفتاح كود معين مثل: KeyEvent.KEYCODE_ENTER

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_X) {
    // code to run when user presses the X key
        }
 return super.onKeyDown(keyCode, event);
 }
```

صف الجني Sprite class

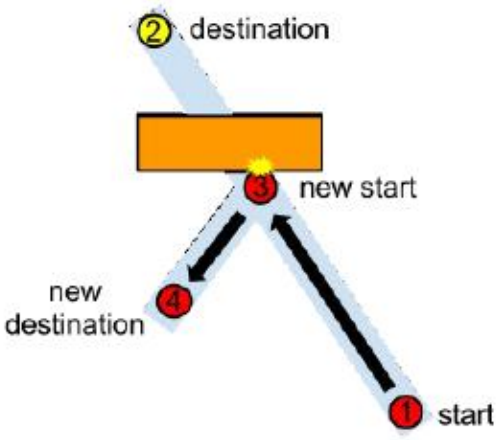
- يتم تعريف مثل هذا الصف (واحد أو أكثر) في الألعاب ويحوي عادةً بيانات مثل:

location, size, velocity, shape/image, points, …

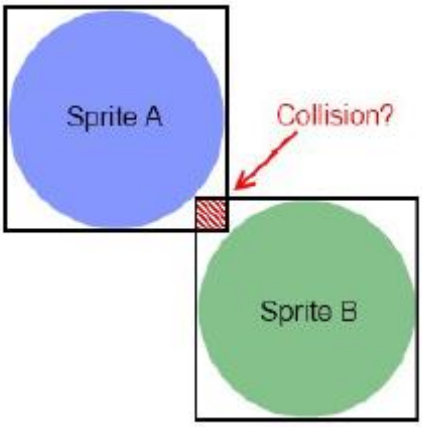
```
// an example sprite class

public class Sprite {

    RectF rect;

    float dx, dy;

    Paint paint;

    ...

}
```

كشف التصادم Collision Detection

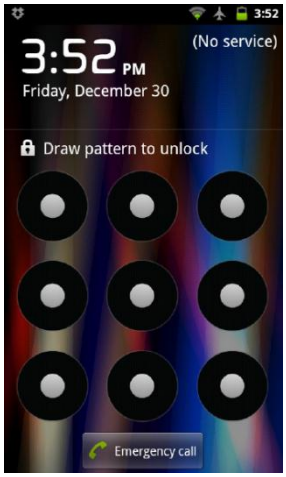
- نحتاج عادةً لكشف التصادم لمعرفة فيما إذا كان غرضين (من الصف Sprite) يلامسان بعضهما البعض.



- يكون مثلاً لصف المستطيل `RectF` طرق لكشف التماس مثلاً:
  - rect1.contains(x, y)
  - rect1.contains(rect2)
  - RectF.intersects(rect1, rect2)



- يكون كشف التماس لأشكال غير مستطيلة أصعب قليلاً.



استخدام القفل WakeLock

- لمنع الشاشة من التقطيع (Blanking):
  - أضف في الملف AndroidManifest.xml:

```
<uses-permission

android:name="android.permission.WAKE_LOCK" />
```

- ضع في الكود:

```
// create the lock (probably in onCreate)

PowerManager pwr = (PowerManager) getSystemService(POWER_SERVICE);

WakeLock lock = pwr.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK,"my
lock");

// turn on the lock (in onResume)

lock.acquire();

// turn off the lock (in onPause)

lock.release();
```

وضع كامل الشاشة Full Screen Mode

- لجعل التطبيق يملئ كامل الشاشة، ضع التعليمات التالية في الإجرائية onCreate:

```
requestWindowFeature(Window.FEATURE_NO_TITLE);

getWindow().setFlags(

    WindowManager.LayoutParams.FLAG_FULLSCREEN,

    WindowManager.LayoutParams.FLAG_FULLSCREEN);
```

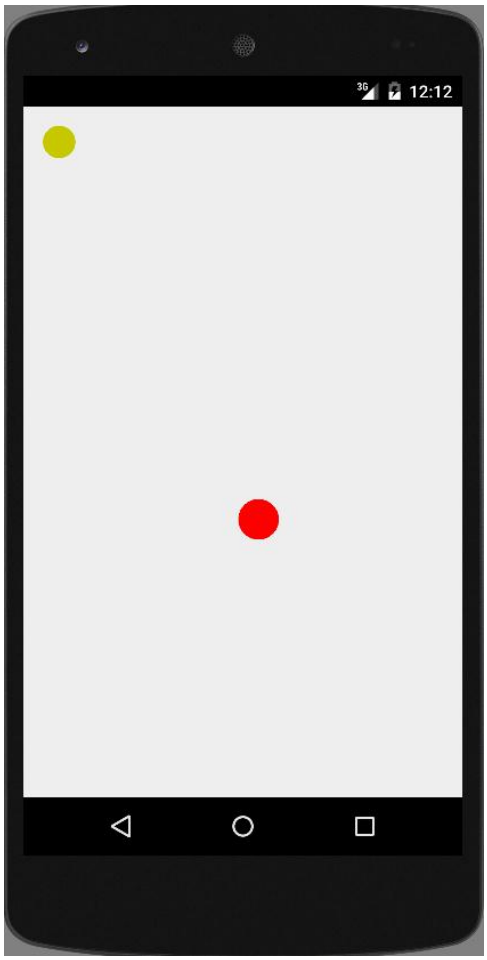
3. مثال تعليمي

الأهداف التعليمية:

- مثال تعليمي (الإحياء)

مثال تعليمي

- عند تنفيذ التطبيق تظهر طابة حمراء تتحرك في جميع الاتجاهات وعندما تصطدم بحافة النافذة ترتد منعكسة وهكذا. كما تتحرك الكرة الصفراء عند قيام المستخدم بلمس أحد حواف الشاشة نحو هذه الحافة.



- يكون ملف النشاط activity_bouncing_ball.xml:

```
<RelativeLayout
    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: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=".BouncingBallActivity">

<com.example.basel.bouncingball.BouncingBallView
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</RelativeLayout>
```

- يكون ملف الصف المخصص BouncingBallView.java:

```
package com.example.basel.bouncingball;

/*
 * This class is a graphical view of a simple animated app
 * with a ball that moves around and bounces off the edges of the screen,
 * as well as a yellow "pac-man" sprite that can move around in response
 * to the user touching the screen at its various edges.
 */

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.RectF;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.view.View;
```

```
public class BouncingBallView extends View {

    private static final float BALL_SIZE = 100;

    private static final float BALL_MAX_VELOCITY = 80;


    private Sprite ball;

    private Sprite pacman;

    private DrawingThread dthread;


    /*

    * This constructor sets up the initial state of the view and sprites.

    */

public BouncingBallView(Context context, AttributeSet attrs) {

        super(context, attrs);


// set up initial state of ball

ball = new Sprite();

ball.setLocation(200, 200);

ball.setSize(BALL_SIZE, BALL_SIZE);

ball.setVelocity(

                (float) ((Math.random() - .5) * 2 * BALL_MAX_VELOCITY),

              (float) ((Math.random() - .5) * 2 * BALL_MAX_VELOCITY));

ball.paint.setARGB(255, 255, 0, 0);


// set up initial state of pac-man

pacman = new Sprite();
```

```
pacman.setLocation(0, 0);

pacman.setSize(80, 80);

pacman.paint.setARGB(255, 200, 200, 0);


// start a drawing thread to animate screen at 50 frames/sec

        dthread = new DrawingThread(this, 50);

        dthread.start();

    }


/*

 * Called when the user touches the screen with their finger.

* Used to control Pac-Man's movement.

* If the user touches the edges of the screen, moves Pac-Man toward that edge.

*/

@Override

public boolean onTouchEvent(MotionEvent event) {

        float x = event.getX();

        float y = event.getY();

        int w = getWidth();

        int h = getHeight();


        if (x < w / 5) {

            pacman.dx = -10;                // left edge of screen

        } else if (x >= w * 4 / 5) {

            pacman.dx = 10;                 // right edge
```

```
        } else {
            pacman.dx = 0;                  // center
        }
        if (y < h / 5) {
            pacman.dy = -10;                // top edge
        } else if (y >= h * 4 / 5) {
            pacman.dy = 10;                 // bottom edge
        } else {
            pacman.dy = 0;                  // center
        }

        return super.onTouchEvent(event);
}

    /*
    * This method draws the bouncing ball and Pac-Man on the screen,
    * and also updates their positions for the next time the screen
    * is redrawn.
    */
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        canvas.drawOval(ball.rect, ball.paint);
        canvas.drawOval(pacman.rect, pacman.paint);
```

```
        updateSprites();
    }

// updates sprites' positions between frames of animation
private void updateSprites() {
        pacman.move();
        ball.move();

// handle ball bouncing off edges
if (ball.rect.left < 0 || ball.rect.right >= getWidth()) {
            ball.dx = -ball.dx;
        }
if (ball.rect.top < 0 || ball.rect.bottom >= getHeight()) {
            ball.dy = -ball.dy;
        }
    }
}
```

- يستخدم صف العرض المخصص صف "الجني" (الكرة) التالي Sprite.java:

```
package com.example.basel.bouncingball;

import android.graphics.Paint;
import android.graphics.RectF;

public  class Sprite {
    // state (fields)
```

```
    public RectF rect = new RectF();

    public float dx = 0;

    public float dy = 0;

    public Paint paint = new Paint();


/* Constructs a default empty sprite. */

public Sprite() {

        // empty

    }


  /* Constructs a sprite of the given location and size. */

  public Sprite(float x, float y, float width, float height) {

        setLocation(x, y);

        setSize(width, height);

    }


    /* Tells the sprite to move
         itself by its current velocity dx,dy. */

          public void move() {

              rect.offset(dx, dy);

          }


          /* Stops the sprite from moving by
      setting its velocity to 0,0. */

    public void stopMoving() {
```

```
        setVelocity(0, 0);
    }

    /* Sets the sprite's x,y location on screen to be the given
values. */
    public void setLocation(float x, float y) {
        rect.offsetTo(x, y);
    }

    /* Sets the sprite's size to be the given values. */
    public void setSize(float width, float height) {
        rect.right = rect.left + width;
        rect.bottom = rect.top + height;
    }

/* Sets the sprites dx,dy velocity to be the given values. */
    public void setVelocity(float dx, float dy) {
        this.dx = dx;
        this.dy = dy;
    }
}
```

- يستخدم صف العرض المخصص صف نيسب الرسم التالي DrawingThread.java:

```
package com.example.basel.bouncingball;


import android.os.Handler;
import android.os.Looper;
import android.view.View;


/*
 * This class is a helper to wrap up some of the icky code needed to
 * initiate an animation thread that repaints a view at regular intervals.
 */


public class DrawingThread {
    private View view = null;
    private int fps;
    private Thread thread = null;
    private Handler handler = null;
    private volatile boolean isRunning = false;


   /**
    * Constructs a new DrawingThread to update the given view
    * the given number of times per second.
    * Does NOT start the thread running; call start() to do so.
    */
public DrawingThread(View view, int fps) {
        if (view == null || fps <= 0) {
```

```
            throw new IllegalArgumentException();
        }
        this.view = view;
        this.fps = fps;
        this.handler = new Handler(Looper.getMainLooper());
    }

    /**
     * Returns true if the drawing thread is currently started and running.
     */
    public boolean isRunning() {
        return thread != null;
    }

/**
 * Starts the thread running so that it will repaint the view repeatedly.
*/
    public void start() {
        if (thread == null) {
            thread = new Thread(new MainRunner());
            thread.start();
        }
    }

    /**
     * Stops the thread so that it will not repaint the view any more.
```

```
    */
    public void stop() {
        if (thread != null) {
            isRunning = false;
            try {
                thread.join();
            } catch (InterruptedException ie) {
                // empty
            }
            thread = null;
        }
    }


    /*
     * Small runnable helper class that contains the thread's main loop
     * to repeatedly sleep-and-redraw the view.
     */
    private class MainRunner implements Runnable {
        public void run() {
            isRunning = true;
            while (isRunning) {
              // sleep for a short time between frames of animation
                try {
                    Thread.sleep(1000 / fps);
                } catch (InterruptedException ie) {
```

```
                    isRunning = false;
                }


                // post a message that will cause the view to redraw
                handler.post(new Updater());
            }
        }
    }


    /*
    * Small runnable helper class needed by Android to redraw a view.
    */
    private class Updater implements Runnable {
        public void run() {
            view.invalidate();
        }
    }
}
```

- ويكون الكود الأساسي bouncing_ball.java:

```
package com.example.basel.bouncingball;


/*
 * This activity is basically just an empty shell to hold the BouncingBallView.
 */
import android.app.Activity;
import android.os.Bundle;


public class bouncing_ball extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_bouncing_ball);
    }
}
```