CAPÍTULO 3.
Gráficos en Android

3.1. Elementos gráficos

3.1.1. Canvas

```
public class EjemploGraficos extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new EjemploView(this));
    }

    public class EjemploView extends View {
        public EjemploView (Context context) {
            super(context);
        }
        @Override
        protected void onDraw(Canvas canvas) {
            //Dibujar aquí
        }
    }
}
```

3.1.2. Paint

```
Paint pincel = new Paint();
pincel.setColor(Color.BLUE);
pincel.setStrokeWidth(8);
pincel.setStyle(Style.STROKE);
canvas.drawCircle(150, 150, 100, pincel);
```

3.1.2.1. Definición de colores

```
int color;
color = Color.BLUE;                    //Azul opaco
color = Color.argb(127, 0, 255, 0);    //Verde transparente
color = 0x7F00FF00;                    //Verde transparente
color = getResources().getColor(R.color.color_Circulo);
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="color_circulo">#7fffff00</color>
</resources>
```

3.1.3. Path

```
Path trazo = new Path();
trazo.addCircle(150, 150, 100, Direction.CCW);

canvas.drawColor(Color.WHITE);
Paint pincel = new Paint();
pincel.setColor(Color.BLUE);
pincel.setStrokeWidth(8);
pincel.setStyle(Style.STROKE);
canvas.drawPath(trazo, pincel);
pincel.setStrokeWidth(1);
pincel.setStyle(Style.FILL);
pincel.setTextSize(20);
pincel.setTypeface(Typeface.SANS_SERIF);
canvas.drawTextOnPath("Desarrollo de aplicaciones para
            móviles con Android", trazo, 10, 40, pincel);
```

```
Path  trazo = new Path();
trazo.moveTo(50, 100);
trazo.cubicTo(60,70, 150,90, 200,110);
trazo.lineTo(300,200);
```

3.1.4. Drawable

3.1.4.1. BitmapDrawable

Declara la variable `miImagen` en la clase `EjemploView`:

```
private Drawable miImagen;
```

Escribe las siguientes tres líneas dentro del constructor de esta clase:

```
Resources res = context.getResources();
miImagen = res.getDrawable(R.drawable.icon);
miImagen.setBounds(new Rect(30,30,200,200));
```

Escribe la siguiente línea en el método `onDraw`:

```
miImagen.draw(canvas);
```

```
ImageView i = new ImageView(this);
i.setImageResource(R.drawable.mi_imagen);
```

```
<ImageView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:src="@drawable/mi_imagen"/>
```

3.1.4.2. **GradienDrawable**

```
<shape xmlns:android=
      "http://schemas.android.com/apk/res/android">
   <gradient
      android:startColor="#FFFFFF"
      android:endColor="#0000FF"
      android:angle="270" />
</shape>
```

Ahora introduce la siguiente línea en el constructor de una vista, para conseguir que este *drawable* sea utilizado como fondo.

```
setBackgroundResource(R.drawable.degradado);
```

El fondo de una vista también puede ser indicado en la definición de su *Layout* en XML. No tienes más que introducir el siguiente parámetro.

```
android:background="@drawable/degradado"
```

3.1.4.3. **TransitionDrawable**

```
<transition  xmlns:android=
      "http://schemas.android.com/apk/res/android">
  <item android:drawable="@drawable/asteroide1"/>
  <item android:drawable="@drawable/asteroide3"/>
</transition>
```

```
@Override
public void onCreate(Bundle savedInstanceState) {
   ImageView image = new ImageView(this);
   setContentView(image);
   TransitionDrawable transition = (TransitionDrawable)
            getResources().getDrawable(R.drawable.transicion);
   image.setImageDrawable(transition);
   transition.startTransition(2000);
}
```

3.1.4.4. ShapeDrawable

En la aplicación `EjemploGraficos` declara la variable `miImagen` en la clase `EjemploView` de la siguiente forma:

```
    private ShapeDrawable miImagen;
```

Escribe las siguientes tres líneas dentro del constructor de esta clase:

```
            miImagen  = new ShapeDrawable(new OvalShape());
            miImagen.getPaint().setColor(0xff0000ff);
            miImagen.setBounds(10, 10, 310, 60);
```

3.1.4.5. AnimationDrawable

```
<animation-list xmlns:android=
          "http://schemas.android.com/apk/res/android"
    android:oneshot="false">
       <item android:drawable="@drawable/misil1"
             android:duration="200" />
       <item android:drawable="@drawable/misil2"
             android:duration="200" />
       <item android:drawable="@drawable/misil3"
             android:duration="200" />
</animation-list>
```

```
AnimationDrawable animacion;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    animacion = (AnimationDrawable) getResources()
                      .getDrawable(R.anim.animacion);
    ImageView image = new ImageView(this);
    image.setBackgroundColor(Color.WHITE);
    image.setImageDrawable(animacion);
    setContentView(image);
}

public boolean onTouchEvent(MotionEvent event) {
    if (event.getAction() == MotionEvent.ACTION_DOWN){
       animacion.start();
```

```
            return true;
        }
        return super.onTouchEvent(event);
    }
```

3.2. **Creando la actividad principal de Asteroides**

Añade la siguiente clase al proyecto *Asteroides*.

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

Necesitaremos un *Layout* para la pantalla del juego. Crea el fichero `res/layout/juego.xml` con el siguiente contenido:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <org.example.asteroides.VistaJuego
      android:id="@+id/VistaJuego"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:focusable="true"
      android:background="@drawable/fondo" />
</LinearLayout>
```

3.2.1. La clase Gráfico

```
package org.example.asteroides;

class Grafico {

   private Drawable drawable;    //Imagen que dibujaremos
   private double posX, posY;    //Posición
   private double incX, incY;    //Velocidad desplazamiento
   private int angulo, rotacion;//Ángulo y velocidad rotación
   private int ancho, alto;      //Dimensiones de la imagen
   private int radioColision;    //Para determinar colisión
    //Donde dibujamos el gráfico (usada en view.ivalidate)
   private View view;
    // Para determinar el espacio a borrar (view.ivalidate)
   public static final int MAX_VELOCIDAD = 20;

   public Grafico(View view, Drawable drawable){
      this.view = view;
      this.drawable = drawable;
      ancho = drawable.getIntrinsicWidth();
      alto = drawable.getIntrinsicHeight();
      radioColision = (alto+ancho)/4;
   }

   public void dibujaGrafico(Canvas canvas){
      canvas.save();
      int x=(int) (posX+ancho/2);
      int y=(int) (posY+alto/2);
      canvas.rotate((float) angulo,(float) x,(float) y);
      drawable.setBounds((int)posX,          (int)posY,
                         (int)posX+ancho,    (int)posY+alto);
      drawable.draw(canvas);
      canvas.restore();
      int rInval =(int) distanciaE(0,0,ancho,alto)/2+MAX_VELOCIDAD;
      view.invalidate(x-rInval, y-rInval, x+rInval, y+rInval);
   };

   public void incrementaPos(){
      posX+=incX;
      // Si salimos de la pantalla, corregimos posición
      if(posX<-ancho/2){
         posX=view.getWidth()-ancho/2;}
      if(posX>view.getWidth()-ancho/2){
         posX=-ancho/2;}
```

```
        posY+=incY;
        // Si salimos de la pantalla, corregimos posición
        if(posY<-alto/2){
            posY=view.getHeight()-alto/2;}
        if(posY>view.getHeight()-alto/2){
            posY=-alto/2;}
        angulo += rotacion; //Actualizamos ángulo
    }

    public double distancia(Grafico g) {
        return distanciaE(posX,posY,g.posX,g.posY);
    }

    public boolean verificaColision(Grafico g) {
        return (distancia(g) < (radioColision+g.radioColision));
    }

    public static double distanciaE(double x, double y,
                                    double x2, double y2) {
        return Math.sqrt((x-x2)*(x-x2) + (y-y2)*(y-y2));
    }
}
```

```
package org.example.asteroides;

public class VistaJuego extends View {
    // //// ASTEROIDES //////
    private Vector<Grafico> Asteroides; // Vector con los Asteroides
    private int numAsteroides = 5; // Número inicial de asteroides
    private int numFragmentos = 3; // Fragmentos en que se divide

    public VistaJuego(Context context, AttributeSet attrs) {
        super(context, attrs);
        Drawable drawableNave, drawableAsteroide, drawableMisil;
        drawableAsteroide =
            context.getResources().getDrawable(R.drawable.asteroide1);
        Asteroides = new Vector<Grafico>();
        for (int i = 0; i < numAsteroides; i++) {
            Grafico asteroide = new Grafico(this, drawableAsteroide);
            asteroide.setIncY(Math.random() * 4 - 2);
            asteroide.setIncX(Math.random() * 4 - 2);
            asteroide.setAngulo((int) (Math.random() * 360));
            asteroide.setRotacion((int) (Math.random() * 8 - 4));
            Asteroides.add(asteroide);
```

```
        }
    }

    @Override
    protected void onSizeChanged(int ancho, int alto,
                                 int ancho_anter, int alto_anter) {
        super.onSizeChanged(ancho, alto, ancho_anter, alto_anter);
        // Una vez que conocemos nuestro ancho y alto.
        for (Grafico asteroide: Asteroides) {
            asteroide.setPosX(Math.random()*
                              (ancho-asteroide.getAncho()));
            asteroide.setPosY(Math.random()*
                              (alto-asteroide.getAlto()));
        }
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        for (Grafico asteroide: Asteroides) {
            asteroide.dibujaGrafico(canvas);
        }
    }
}
```

Declara las siguientes variables al comienzo de la clase `VistaJuego`:

```
    // //// NAVE //////
    // Gráfico de la nave
    private Grafico nave;
    private int giroNave; // Incremento de dirección
    private float aceleracionNave; // aumento de velocidad
    // Incremento estándar de giro y aceleración
    private static final int PASO_GIRO_NAVE = 5;
    private static final float PASO_ACELERACION_NAVE = 0.5f;
```

```
do {
        asteroide.setPosX(Math.random()*(w-asteroide.getAncho()));
        asteroide.setPosY(Math.random()*(h-asteroide.getAlto()));
} while(asteroide.distancia(nave) < (w+h)/5);
```

3.3. Representación de gráficos vectoriales en Asteroides

```
Path pathAsteroide = new Path();
pathAsteroide.moveTo((float)0.3,(float)0.0);
pathAsteroide.lineTo((float)0.6, (float)0.0);
pathAsteroide.lineTo((float)0.6, (float)0.3);
pathAsteroide.lineTo((float)0.8, (float)0.2);
pathAsteroide.lineTo((float)1.0, (float)0.4);
pathAsteroide.lineTo((float)0.8, (float)0.6);
pathAsteroide.lineTo((float)0.9, (float)0.9);
pathAsteroide.lineTo((float)0.8, (float)1.0);
pathAsteroide.lineTo((float)0.4, (float)1.0);
pathAsteroide.lineTo((float)0.0, (float)0.6);
pathAsteroide.lineTo((float)0.0, (float)0.2);
pathAsteroide.lineTo((float)0.3, (float)0.0);
ShapeDrawable dAsteroide = new ShapeDrawable(
                    new PathShape(pathAsteroide ,1, 1) );
dAsteroide.getPaint().setColor(Color.WHITE);
dAsteroide.getPaint().setStyle(Style.STROKE);
dAsteroide.setIntrinsicWidth(50);
dAsteroide.setIntrinsicHeight(50);
drawableAsteroide = dAsteroide;
```

3.4. Introduciendo el movimiento en Asteroides

```
protected void actualizaFisica() {
   long ahora = System.currentTimeMillis();
   // No hagas nada si el período de proceso no se ha cumplido.
   if (ultimoProceso + PERIODO_PROCESO > ahora) {
      return;
   }
   // Para una ejecución en tiempo real calculamos retardo
   double retardo = (ahora - ultimoProceso) / PERIODO_PROCESO;
   // Actualizamos posición nave
   nave.setAngulo((int) (nave.getAngulo() + giroNave * retardo));
   double nIncX = nave.getIncX() + aceleracionNave *
            Math.cos(Math.toRadians(nave.getAngulo())) * retardo;
   double nIncY = nave.getIncY() + aceleracionNave *
            Math.sin(Math.toRadians(nave.getAngulo())) * retardo;
   if (Grafico.distanciaE(0,0,nIncX,nIncY)<=Grafico.getMaxVelocidad()){
      nave.setIncX(nIncX);
      nave.setIncY(nIncY);
```

```
    }
    nave.incrementaPos();
    for (Grafico asteroide : Asteroides) {
       asteroide.incrementaPos();
    ultimoProceso = ahora;
}
```

Ahora necesitamos que esta función sea llamada continuamente. Para ello utilizaremos un *Thread* que definiremos de la siguiente forma:

```
class ThreadJuego extends Thread {
   @Override
   public void run() {
      while (true) {
         actualizaFisica();
      }
   }
}
```

Introducir esta línea al final del método `onSizeChanged()`:

```
thread.start();
```

También necesitas declarar las siguientes variables al principio de la clase:

```
// //// THREAD Y TIEMPO //////
// Thread encargado de procesar el juego
private ThreadJuego thread new ThreadJuego();
// Cada cuanto queremos procesar cambios (ms)
private static int PERIODO_PROCESO = 50;
// Cuando se realizó el último proceso
private long ultimoProceso = 0;
```