PRACTICA 18: ASTEROIDES VII

Parte V: *Introduciendo la nave en VistaJuego*

Paso 1. Declara las siguientes variables al comienzo de la clase VistaJuego:

```
public class VistaJuego extends View {
    // //// NAVE //////
    private Grafico nave;// Gráfico de la 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;
```

Algunas de estas variables serán utilizadas en el siguiente capítulo.

Paso 2. En el constructor de la clase instancia la variable drawableNave de forma similar como se ha hecho en drawableAsteroide.

```
public VistaJuego(Context context, AttributeSet attrs) {
    super(context, attrs);
    Drawable drawableNave, drawableAsteroide, drawableMisil;
    drawableAsteroide = ContextCompat.getDrawable(context, R.drawable.asteroide1);

    drawableNave =  ContextCompat.getDrawable(context, R.drawable.nave);
```

Paso 3. Inicializa también en el constructor la variable nave de la siguiente forma:

nave = **new** Grafico(**this**, drawableNave);

```
public VistaJuego(Context context, AttributeSet attrs) {
    super(context, attrs);
    Drawable drawableNave, drawableAsteroide, drawableMisil;
    drawableAsteroide = ContextCompat.getDrawable(context, R.drawable.asteroide1);
    drawableNave =  ContextCompat.getDrawable(context, R.drawable.nave);

    nave = new Grafico( view: this, drawableNave);
```

Paso 4. En el método onSizeChanged() posiciona la nave justo en el centro de la vista.

```
@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.

    nave.setPosX(ancho/2-nave.getAncho()/2);
    nave.setPosY(alto/2-nave.getAncho()/2);

    for (Grafico asteroide: Asteroides) {
        asteroide.setPosX(Math.random()* (ancho-asteroide.getAncho()));
        asteroide.setPosY(Math.random()* (alto-asteroide.getAlto()));
    }
}
```

Paso 5. En el método onDraw() dibuja la nave en el Canvas.

```
@Override protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);

    nave.dibujaGrafico(canvas);

    for (Grafico asteroide: Asteroides) {
        asteroide.dibujaGrafico(canvas);
    }
}
```

Paso 6. Ejecuta la aplicación. La nave ha de aparecer centrada:

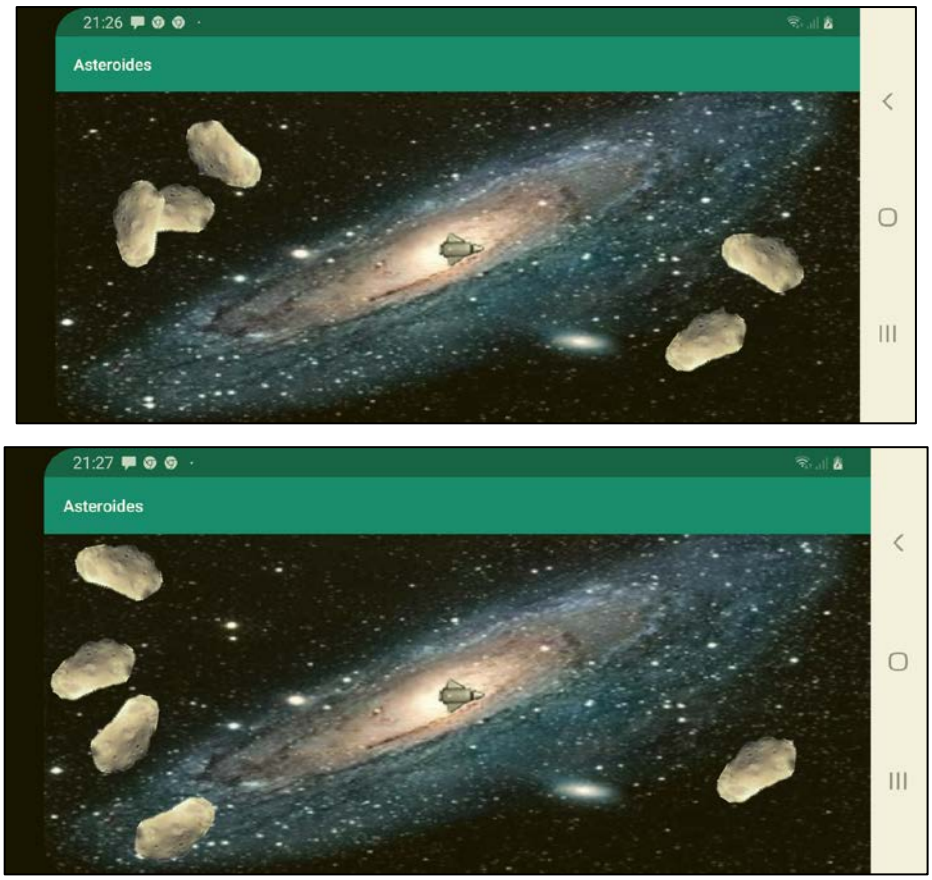
Paso 7. Si cuando situamos los asteroides, alguno coincide con la posición de la nave, el jugador no tendrá ninguna opción de sobrevivir. Sería más interesante asegurarnos de que al posicionar los asteroides estos se encuentran a una distancia adecuada a la nave, y en caso contrario tratar de obtener otra posición. Para conseguirlo puedes utilizar el siguiente código en sustitución de las dos líneas asteroide.setPosX(…) y asteroide.setPosY(…).

```
@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.

    nave.setPosX(ancho/2-nave.getAncho()/2);
    nave.setPosY(alto/2-nave.getAncho()/2);

    for (Grafico asteroide: Asteroides) {
        do {
            asteroide.setPosX(Math.random() * (ancho - asteroide.getAncho()));
            asteroide.setPosY(Math.random() * (alto - asteroide.getAlto()));
        } while (asteroide.distancia(nave)<(ancho+alto)/5);
    }
}
```

Parte VI: *Evitando que VistaJuego cambie su representación con el dispositivo en horizontal y en vertical*

Paso 1. Ejecuta la aplicación.

Paso 2. Cambia de orientación la pantalla del dispositivo. En el emulador se consigue pulsando la tecla *Ctrl-F11*.



Paso 3. Observa cómo cada vez, se reinicializa la vista, regenerando los asteroides. Esto nos impediría jugar de forma adecuada. Para solucionarlo edita *AndroidManifest.xml*. En la lengüeta Application selecciona la actividad Juego. En los parámetros de la derecha selecciona en *Screen orientation: landscape*.

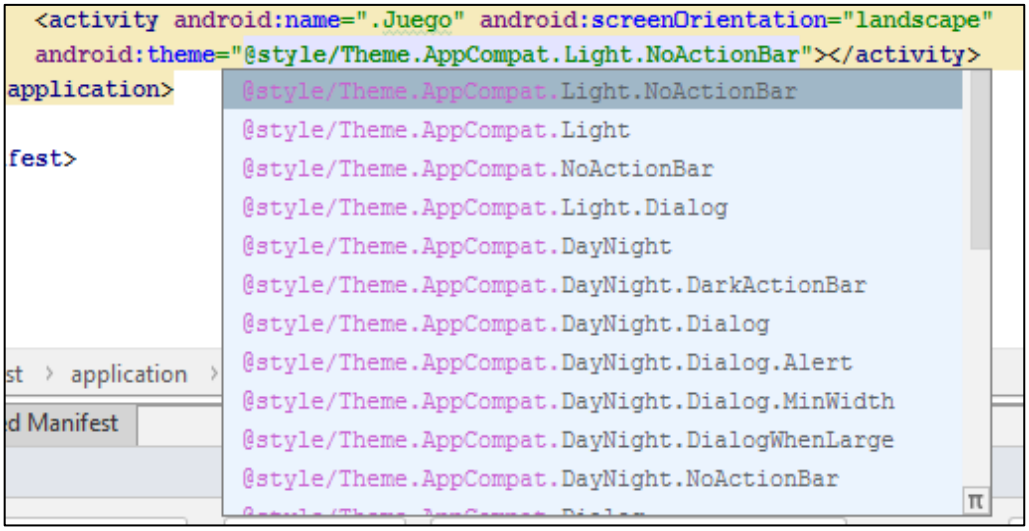
```
    <activity
        android:name=".Preferencias"
        android:label="Preferencias" />
    <activity android:name=".Puntuaciones" />
    <activity android:name=".Juego" android:screenOrientation="landscape"></activity>
</application>
```

Paso 4. Ejecuta de nuevo la aplicación. Observa como la actividad Juego será siempre representada en modo horizontal, de forma independiente a la posición del teléfono.

Paso 5. Abre de nuevo las propiedades de la actividad Juego. En *Theme* selecciona el valor @style/Theme.AppCompat.Light.NoActionBar. Este tema visualizará la vista ocupando toda la pantalla, sin la barra de notificaciones ni el nombre de la aplicación.

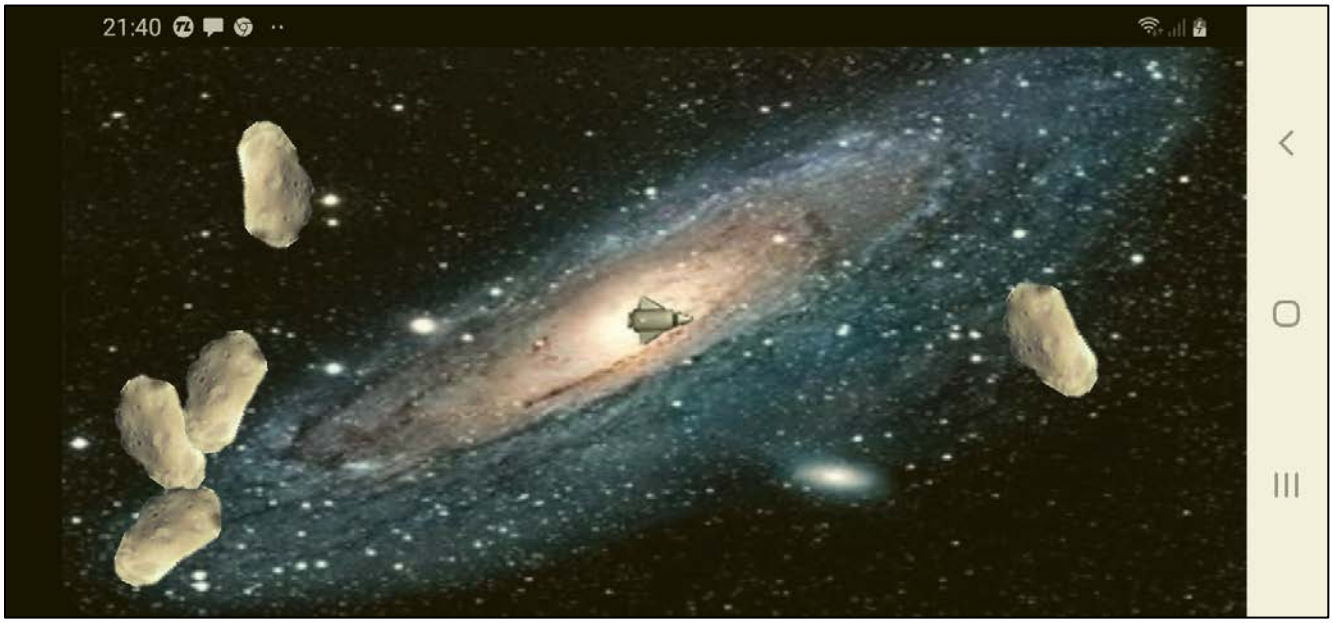
```
    <activity
        android:name=".Preferencias"
        android:label="Preferencias" />
    <activity android:name=".Puntuaciones" />
    <activity android:name=".Juego" android:screenOrientation="landscape"
    android:theme="@style/Theme.AppCompat.Light.NoActionBar"></activity>
</application>
```

Paso 6. Si en *Theme* pulsas el botón *Browse…* y seleccionas el botón circular *System Resources* puedes ver una lista de estilos definidos en el sistema.



***NOTA**. En algunas instalaciones esta lista puede que no te funcione.*

Paso 7. Ejecuta la aplicación en un terminal real y verifica el resultado.



***NOTA**: en un emulador si cambias la orientación (Crtl-F11) esta cambiará igualmente. Se trata de un error de simulación, al no soportar esta configuración.*

Parte VII: Representación vectorial de los Asteroides

Paso 1. Abre la clase VistaJuego.

Paso 2. En el constructor reemplaza la línea:

drawableAsteroide = context.getResources().getDrawable(R.drawable.*asteroide1*);

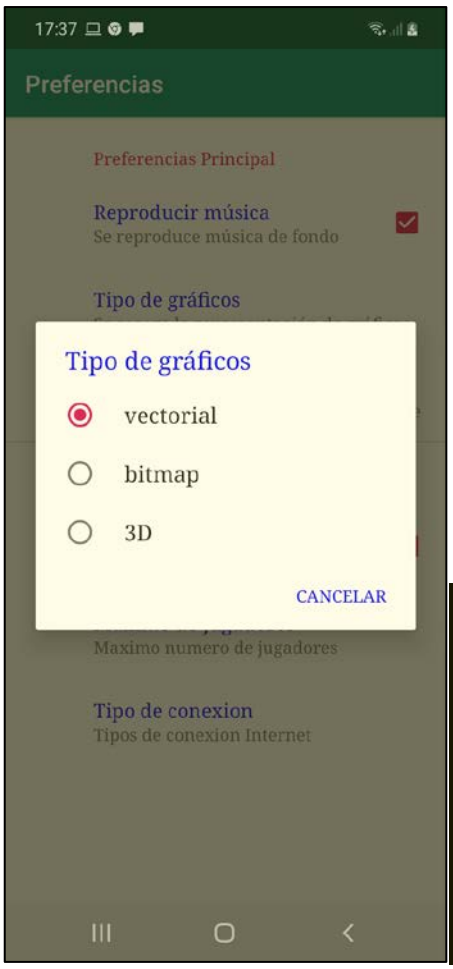
```
public VistaJuego(Context context, AttributeSet attrs) {
    super(context, attrs);
    Drawable drawableNave, drawableAsteroide, drawableMisil;

    drawableAsteroide = ContextCompat.getDrawable(context, R.drawable.asteroide1);

    drawableNave =  ContextCompat.getDrawable(context, R.drawable.nave);
```

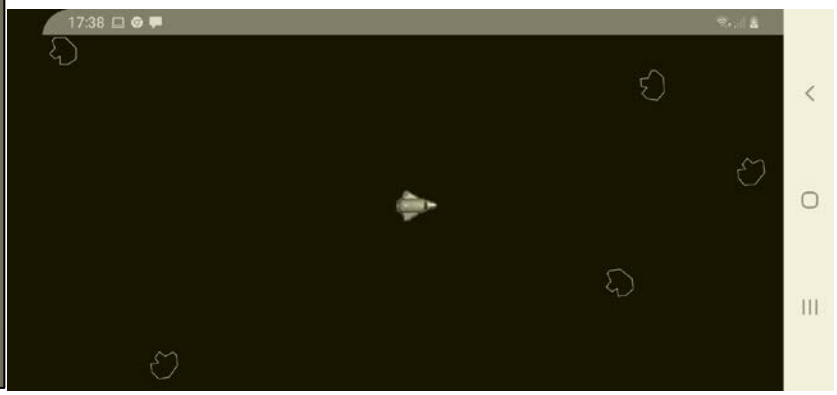
por el siguiente código:

```
public VistaJuego(Context context, AttributeSet attrs) {
    super(context, attrs);
    Drawable drawableNave, drawableAsteroide, drawableMisil;

    //drawableAsteroide = ContextCompat.getDrawable(context, R.drawable.asteroide1);
    SharedPreferences pref = PreferenceManager.getDefaultSharedPreferences(getContext());
    if (pref.getString( s: "graficos",  s1: "1").equals("0")) {
        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,  stdWidth: 1,  stdHeight: 1));
        dAsteroide.getPaint().setColor(Color.WHITE);
        dAsteroide.getPaint().setStyle(Paint.Style.STROKE);
        dAsteroide.setIntrinsicWidth(50);
        dAsteroide.setIntrinsicHeight(50);
        drawableAsteroide = dAsteroide;
        setBackgroundColor(Color.BLACK);
    } else {
        drawableAsteroide = ContextCompat.getDrawable(context, R.drawable.asteroide1);
    }
```

Lo primero que hace este código es consultar en las preferencias para ver si el usuario ha escogido gráficos vectoriales. En caso negativo se realizará la misma inicialización de drawableAsteroide que teníamos antes. En caso afirmativo comenzamos creando la variable pathAsteroide de la clase Path. En este objeto se introducen todas las órdenes de dibujo necesarias para dibujar un asteroide. Luego se crea la variable dAsteroide de la clase ShapeDrawable para crea un *drawable* a partir del *path*. Los últimos dos parámetros (...,1,1) significan el valor de escala aplicado al eje x y al eje y. Luego se indica el color y el estilo del pincel e indicamos el alto y ancho por defecto del *drawable*. Finalmente asignamos el objeto creado a drawableAsteroide.

Paso 3. Ejecuta la aplicación y selecciona el tipo de gráficos adecuado en las preferencias.





Parte VIII: Representación de la nave con VectorDrawable

A partir de la versión 5 de Android, es posible definir un gráfico vectorial en un recurso XML. Esta forma de trabajar resulta mucho más interesante que la propuesta en el ejercicio “Representación vectorial de los asteroides”. Dado que permite separar del código aspectos relacionados con el diseño de la aplicación. Por ejemplo, el diseñador gráfico de nuestra empresa podrá modificar directamente la forma de un asteroide, sin tener que acceder al código.

Paso 1. Añade a las preferencias un nuevo tipo de gráficos. Además de vectorial y bitmap, se ha de poder escoger la opción VectorDrawable.

arrays.xml

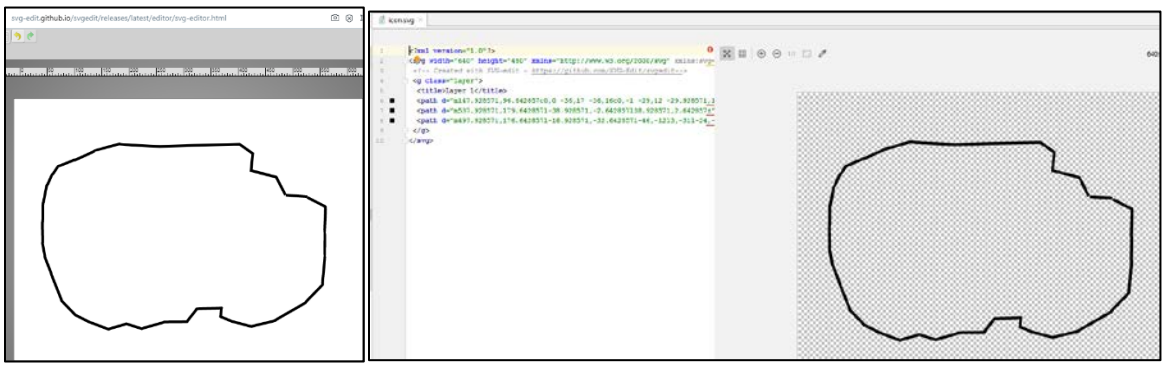
```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="tiposGraficos">
        <item>vectorial</item>
        <item>bitmap</item>
        <item>VectorDrawable</item>
        <item>3D</item>
    </string-array>
    <string-array name="tiposGraficosValores">
        <item>0</item>
        <item>1</item>
        <item>2</item>
    </string-array>

    <string-array name="tiposConexion">
        <item>Bluetooth</item>
        <item>Wi-Fi</item>
        <item>Internet</item>
    </string-array>

    <string-array name="tiposConexionValores">
        <item>0</item>
        <item>1</item>
        <item>2</item>
    </string-array>
</resources>
```

Paso 2. Crea tres recursos de tipo VectorDrawable que representen tres asteroides de tamaños distintos. Para esto tienes varias opciones.

a) Puedes copiar el XML de la estrella y modificar el path usando los puntos definidos en "Representación vectorial de los asteroides".

b) Puedes usar un editor vectorial como Adobe Photoshop o SVG-Edit (https://github.com/SVG-Edit/svgedit, pulsar en Try SVG-edit here.) y crear tus propios asteroides.

https://svgedit.netlify.app/editor/index.html



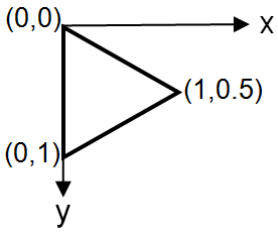
c) Puedes buscar en Internet algún fichero SVG con forma de asteroide.

Paso 3. Modifica el código para que al seleccionar como tipo de gráficos VectorDrawable, se utilicen estos recursos.

Parte IX: Representación vectorial de la nave

Como habrás comprobado en el ejercicio anterior la nave se representa siempre utilizando un fichero png. En esta práctica has de intentar que también pueda representarse vectorialmente.

Paso 1. Crea un nuevo objeto de la clase Path para representar la nave dentro de la sección if introducida en el ejercicio anterior. Como puedes ver en la ilustración siguiente ha de ser un simple triángulo. Como el ángulo de rotación inicial es cero, la nave ha de mirar a la derecha.




```
//NAVE VECTORIAL
Path pathNave = new Path();
pathNave.moveTo((float)0.0,(float)0.0);
pathNave.lineTo((float)1.0,(float)0.5);
pathNave.lineTo((float)0.0,(float)1.0);
pathNave.lineTo((float)0.0,(float)0.0);
```

Paso 2. Crea un nuevo ShapeDrawable a partir de Path anterior. Unas dimensiones adecuadas para que la nave pueden ser 20 de ancho y 15 de alto.

```
ShapeDrawable dNave = new ShapeDrawable(
        new PathShape(pathNave,  stdWidth: 1,  stdHeight: 1));
dNave.getPaint().setColor(Color.WHITE);
dNave.getPaint().setStyle(Paint.Style.STROKE);
dNave.setIntrinsicWidth(20);
dNave.setIntrinsicHeight(15);
drawableNave = dNave;
```

Paso 3. Inicializa la variable drawableNave de forma adecuada.

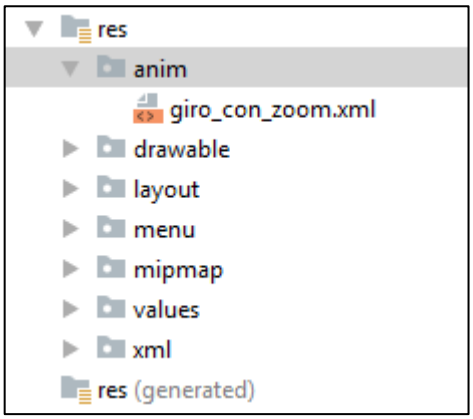
```
public VistaJuego(Context context, AttributeSet attrs) {
    super(context, attrs);
    Drawable drawableNave, drawableAsteroide;
    SharedPreferences pref = PreferenceManager.getDefaultSharedPreferences(getContext())
    if (pref.getString( s: "graficos",  s1: "1").equals("0")) {
        //ASTEROIDE VECTORIAL
        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,  stdWidth: 1,  stdHeight: 1));
        dAsteroide.getPaint().setColor(Color.WHITE);
        dAsteroide.getPaint().setStyle(Paint.Style.STROKE);
        dAsteroide.setIntrinsicWidth(50);
        dAsteroide.setIntrinsicHeight(50);
        drawableAsteroide = dAsteroide;
        //NAVE VECTORIAL
        Path pathNave = new Path();
        pathNave.moveTo((float)0.0,(float)0.0);
        pathNave.lineTo((float)1.0,(float)0.5);
        pathNave.lineTo((float)0.0,(float)1.0);
        pathNave.lineTo((float)0.0,(float)0.0);

        ShapeDrawable dNave = new ShapeDrawable(
                new PathShape(pathNave,  stdWidth: 1,  stdHeight: 1));
        dNave.getPaint().setColor(Color.WHITE);
        dNave.getPaint().setStyle(Paint.Style.STROKE);
        dNave.setIntrinsicWidth(20);
        dNave.setIntrinsicHeight(15);
        drawableNave = dNave;
        setBackgroundColor(Color.BLACK);

    } else{
        drawableAsteroide = ContextCompat.getDrawable(context, R.drawable.asteroide1);
        drawableNave =  ContextCompat.getDrawable(context, R.drawable.nave);
    }
```

Parte X: *Introduciendo animaciones en Asteroides*

En esta práctica has de conseguir que los diferentes elementos del Layout inicial de Asteroides vayan apareciendo uno tras otro con diferentes efectos.

Paso 1. Abre el proyecto Asteroides.

Paso 2. Crea una nueva animación con nombre *giro_con_zoom.xml*. Ha de durar dos segundos y de forma simultánea ha de hacer un zoom de escala 3 a 1 y un giro de dos vueltas (720º). El punto de anclaje de la rotación y el zoom ha de ser el centro de la vista.



giro_con_zoom.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <scale
        android:duration="2000"
        android:fromXScale="3.0"
        android:fromYScale="3.0"
        android:toXScale="1.0"
        android:toYScale="1.0" />
    <rotate
        android:startOffset="0"
        android:duration="2000"
        android:fromDegrees="0"
        android:toDegrees="720"
        android:pivotX="50%"
        android:pivotY="50%"/>
</set>
```

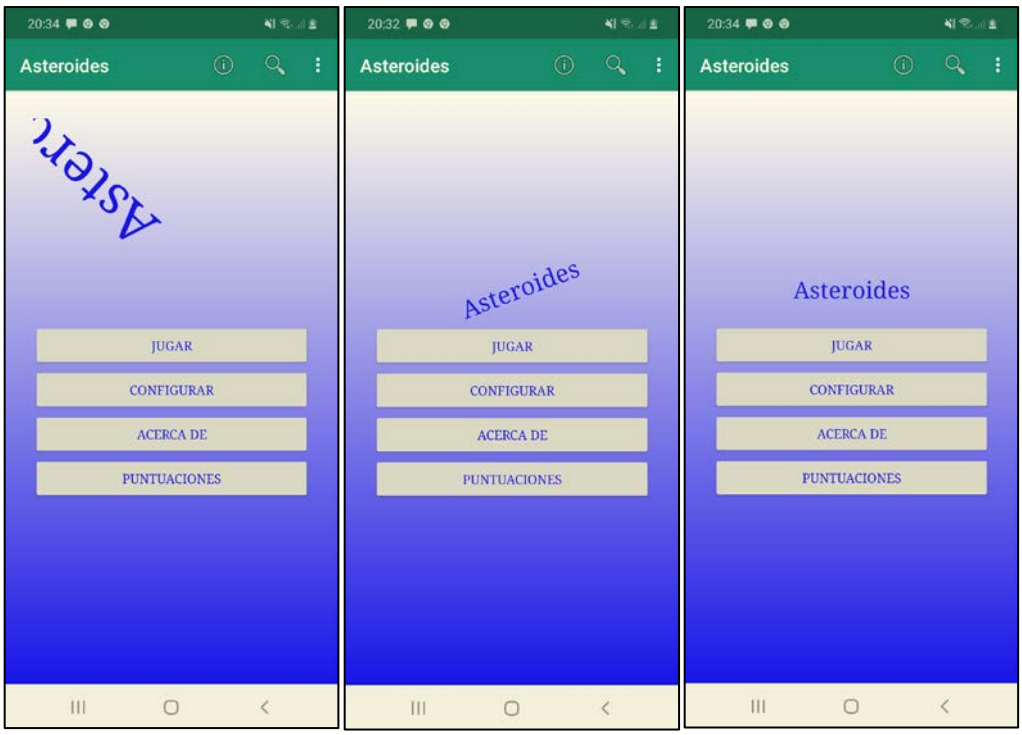
Paso 3. Selecciona el Layout activity_*main.xml* y pon un id al TextView correspondiente al título.

activity_main.xml

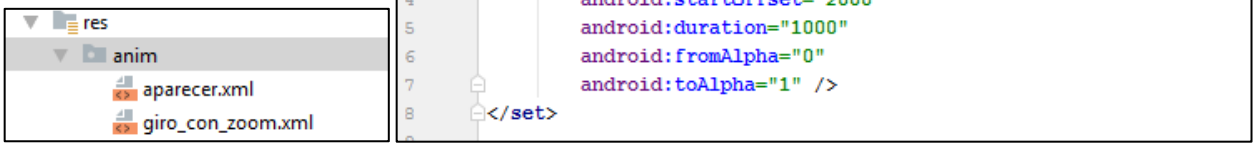
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:style="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:padding="30dp"
    tools:context=".MainActivity"
    android:background="@drawable/degradado">
    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/tituloAplicacion"
        android:gravity="center"
        android:textSize="25sp"
        android:layout_marginBottom="20dp"/>
```

Paso 4. En la actividad inicial de Asteroides, crea un objeto correspondiente a este TextView y aplícale la animación anterior.

```java
public class MainActivity extends AppCompatActivity {
    public static AlmacenPuntuaciones almacen= new AlmacenPuntuacionesLista();
    private Button bAcercaDe;
    private Button bSalir;
    private Animation animacion;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //setImageResource(R.drawable.img1)

        bAcercaDe = findViewById(R.id.button03);
        bAcercaDe.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
                lanzarAcercaDe( view: null);
            }
        });
        //bAcercaDe.setBackgroundResource(R.drawable.degradado);

        TextView texto = (TextView) findViewById(R.id.textView);
        animacion = AnimationUtils.loadAnimation( context: this, R.anim.giro_con_zoom);
        texto.startAnimation(animacion);
```



Paso 5. Crea una nueva animación con nombre *aparecer.xml*. Ha de comenzar a los dos segundos, durar un segundo y modificar el valor de *alpha* de 0 hasta 1.



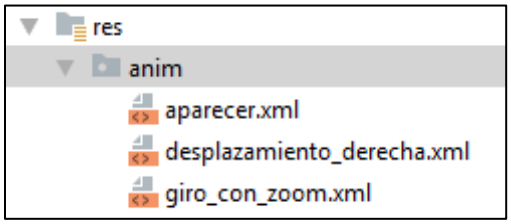
aparecer.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <alpha
        android:startOffset="2000"
        android:duration="1000"
        android:fromAlpha="0"
        android:toAlpha="1" />
</set>
```

Paso 6. Aplica esta animación al primer botón.

```java
public class MainActivity extends AppCompatActivity {
    public static AlmacenPuntuaciones almacen= new AlmacenPuntuacionesLista();
    private Button bAcercaDe;
    private Button bSalir;
    private Button bPlay;
    private Animation animacion;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //setImageResource(R.drawable.img1)

        bAcercaDe = findViewById(R.id.button03);
        bAcercaDe.setOnClickListener((view) -> { lanzarAcercaDe( view: null); });
        //bAcercaDe.setBackgroundResource(R.drawable.degradado);

        TextView texto = (TextView) findViewById(R.id.textView);
        animacion = AnimationUtils.loadAnimation( context: this, R.anim.giro_con_zoom);
        texto.startAnimation(animacion);

        bPlay = (Button) findViewById(R.id.button01);
        Animation animacion2 = AnimationUtils.loadAnimation( context: this, R.anim.aparecer);
        bPlay.startAnimation(animacion2);

        bSalir = findViewById(R.id.button04);
        /*bSalir.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
                finish();
            }
        });*/
```

Paso 7. Crea una nueva animación con nombre *desplazamiento_derecha.xml*. Ha de comenzar a los tres segundos, durar un segundo y modificar el valor de *desplazamiento x de* 400 hasta 0. Prueba también algún tipo de interpolación.



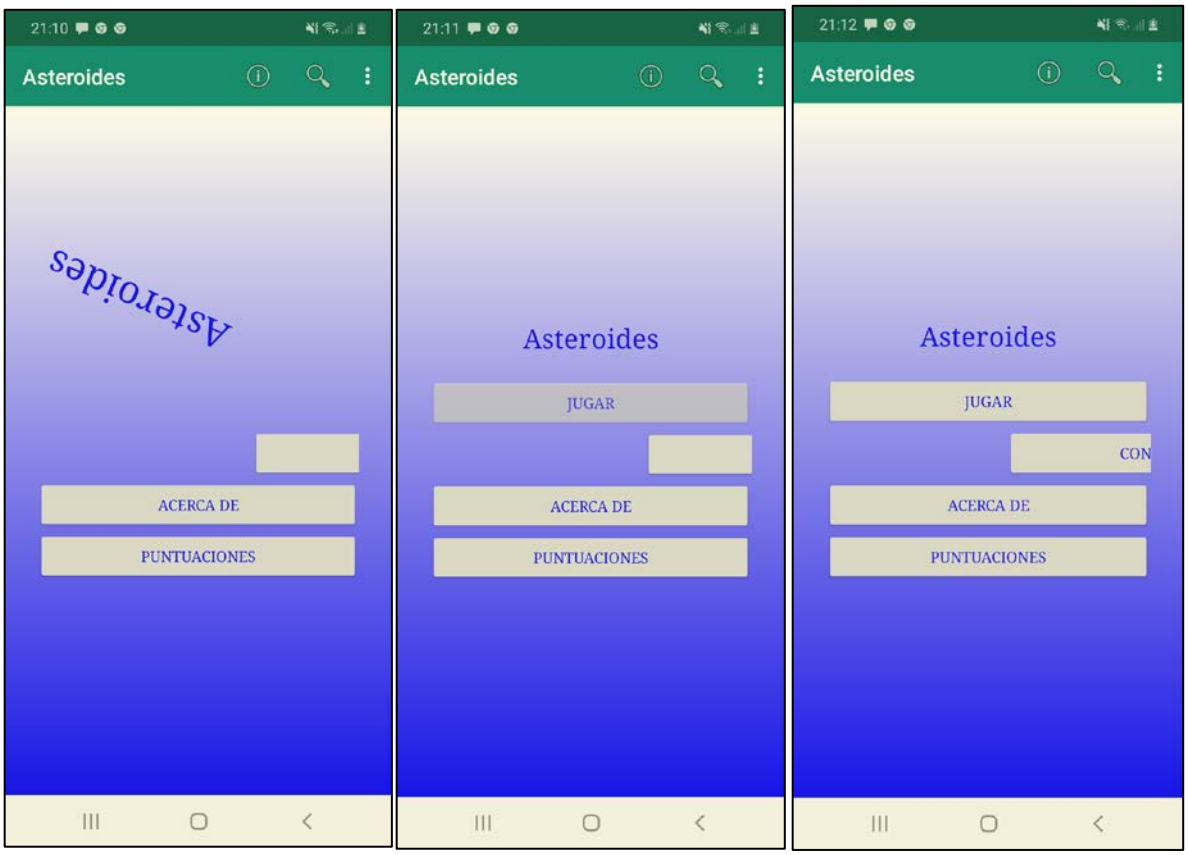
desplazamiento_derecha.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:startOffset="3000"
        android:duration="1000"
        android:fromXDelta="400"
        android:fromYDelta="0"
        android:toXDelta="0"
        android:toYDelta="0" />
</set>
```

Paso 8. Aplica esta animación al segundo botón.

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //setImageResource(R.drawable.img1)

    bAcercaDe = findViewById(R.id.button03);
    bAcercaDe.setOnClickListener((view) -> { lanzarAcercaDe( view: null); });
    //bAcercaDe.setBackgroundResource(R.drawable.degradado);

    TextView texto = (TextView) findViewById(R.id.textView);
    animacion = AnimationUtils.loadAnimation( context: this, R.anim.giro_con_zoom);
    texto.startAnimation(animacion);

    bPlay = (Button) findViewById(R.id.button01);
    Animation animacion2 = AnimationUtils.loadAnimation( context: this, R.anim.aparecer);
    bPlay.startAnimation(animacion2);

    bConfigurar = (Button) findViewById(R.id.button02);
    Animation animacion3 = AnimationUtils.loadAnimation( context: this, R.anim.desplazamiento_derecha);
    bConfigurar.startAnimation(animacion3);
```

Paso 9. Si dispones de tiempo crea dos nuevas animaciones a tu gusto y aplícalas al tercer y cuarto botón.



Paso 10. Aplica la animación *giro_con_zoom.xml* al botón “Acerca de” cuando sea pulsado. Observa como al lanzar la nueva actividad AcercaDe, la actividad principal continúa ejecutándose.

```java
public class MainActivity extends AppCompatActivity {
    public static AlmacenPuntuaciones almacen= new AlmacenPuntuacionesLista();
    private Button bAcercaDe;
    private Button bSalir;
    private Button bPlay;
    private Button bConfigurar;
    private Animation animacion;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //setImageResource(R.drawable.img1)

        bAcercaDe = findViewById(R.id.button03);
        bAcercaDe.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
                bAcercaDe.startAnimation(animacion);
                lanzarAcercaDe( view: null);
            }
        });
```