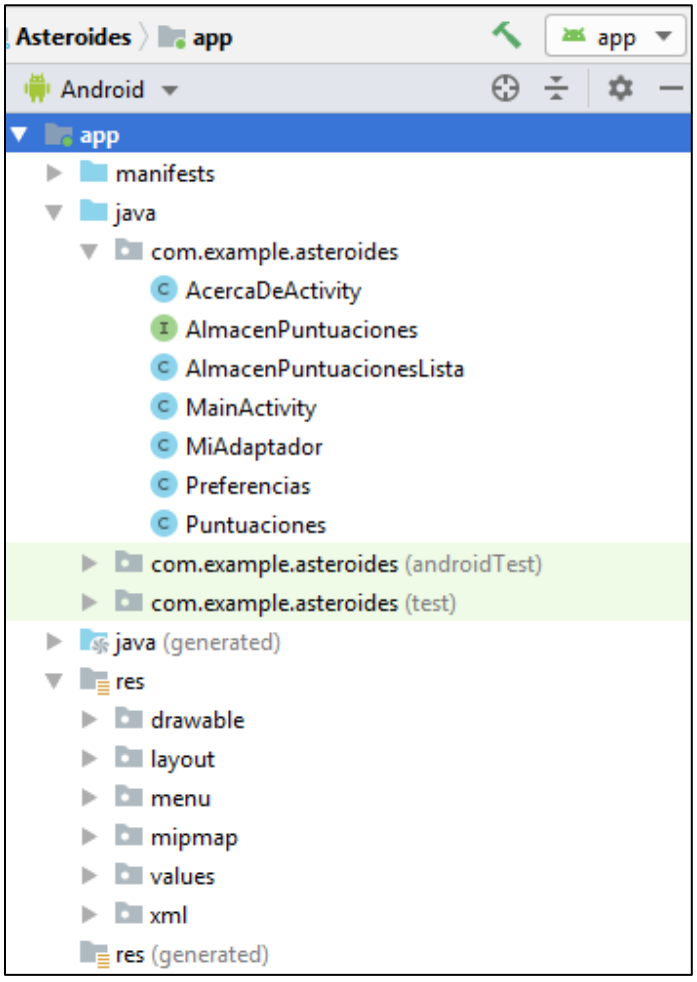
PRACTICA 17: ASTEROIDES VI

Parte I: *Definir un GradienDrawable*

Paso 1. Abre el proyecto Asteroides.



Paso 2. Crea el siguiente fichero *res/drawable/degradado.xml*:

degradado.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android">
    <gradient
        android:startColor="#FFFFFF"
        android:endColor="#0000FF"
        android:angle="270" />
</shape>
```

Paso 3. Podrías introducir la siguiente línea en el constructor de una vista, para conseguir que este *drawable* sea utilizado como fondo

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

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

    @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);

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

Paso 4. Resulta más conveniente definir el fondo de una vista en su *Layout* en XML en lugar de hacerlo por código. Comenta la línea introducida en el punto anterior e introduce el siguiente atributo en el *Layout main.xml*.

android:background="@drawable/degradado"

activity_main.xml

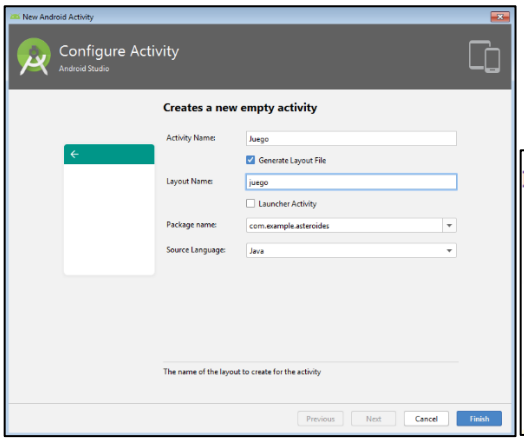
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:style="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical"
    android:layout_width="match_parent"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

Parte II: *Creando la actividad principal en Asteroides*

Lo primero que necesitamos es crear una actividad que controle la pantalla del juego propiamente dicho. A esta actividad le llamaremos Juego.

Paso 1. Abre el proyecto Asteroides.

Paso 2. Lo primero que necesitamos es crear una actividad que controle la pantalla del juego propiamente dicho. Crea la clase Juego con el siguiente código.




```
public class Juego extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.juego);
    }
}
```

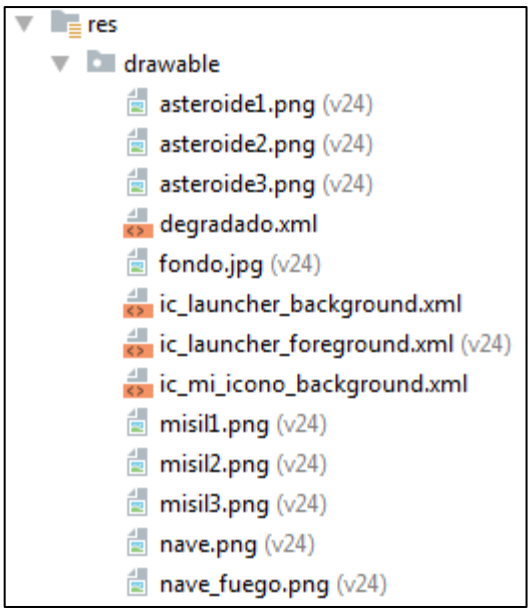
Paso 3. Necesitaremos un *Layout* para la pantalla del juego. Crea el fichero res/layout/juego.xml con el siguiente contenido:

juego.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <com.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>
```

Como puedes observar, cuando diseñamos un *Layout* en XML, no estamos limitados a escoger los elementos que tenemos en la paleta de vistas; vamos a utilizar vistas creadas por nosotros. En este ejemplo utilizaremos la vista com.example.asteroides.VistaJuego que será descrita más adelante. Va a ser esta vista la que lleve el peso de la aplicación

Paso 4. Observa que se ha indicado el parámetro android:background asociado al recurso @drawable/fondo. Por lo tanto, tendremos que poner el recurso *fondo.jpg* en la carpeta correspondiente. Copia también los gráficos correspondientes a los asteroides, la nave y el misil a la carpeta res/drawable. Estos gráficos serán utilizados en los siguientes apartados. Los puedes encontrar en www.androidcurso.com.



Paso 5. De momento no podremos ejecutar la aplicación hasta haber definido la clase VistaJuego.

Parte III: *La clase Grafico*

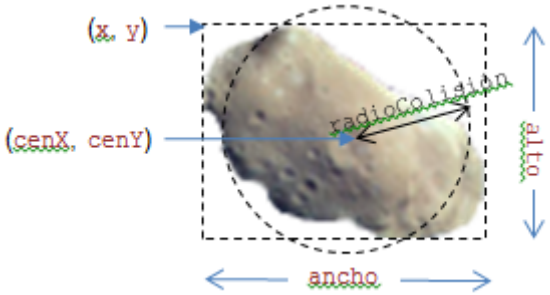
Paso 1. Crea una nueva clase Grafico en el proyecto *Asteroides* y copia el siguiente código

Grafico.java

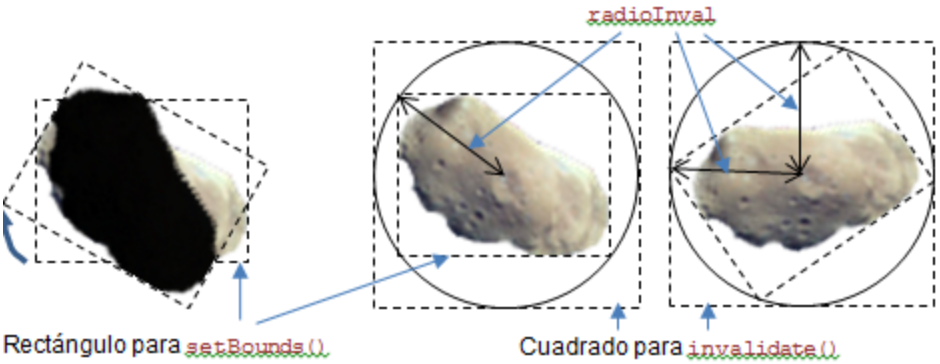
```
public class Grafico {
    private Drawable drawable;    //Imagen que dibujaremos
    private double posX, posY;    //Posición
    private double incX, incY;    //Velocidad desplazamiento
    private int angulo, rotacion;//Ángulo y velocidad rotación
    private int ancho, alto;      //Dimensiones 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,
                right: (int)posX+ancho, bottom: (int)posY+alto);
        drawable.draw(canvas);
        canvas.restore();
        int rInval = (int) Math.hypot(ancho,alto)/2 + MAX_VELOCIDAD;
        view.invalidate( l: x-rInval, t: y-rInval, r: x+rInval, b: y+rInval);
    }
    public void incrementaPos(double factor){
        posX+=incX * factor;
        // 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 * factor;
        if(posY<-alto/2) {posY=view.getHeight()-alto/2;}
        if(posY>view.getHeight()-alto/2) {posY=-alto/2;}
        angulo += rotacion * factor; //Actualizamos ángulo
    }
    public double distancia(Grafico g) {
        return Math.hypot(posX-g.posX, posY-g.posY);
    }
    public boolean verificaColision(Grafico g) {
        return(distancia(g) < (radioColision+g.radioColision));
    }
}
```

Cada objeto de la clase Grafico se caracteriza por situarse en unas coordenadas centrales (cenX, cenY). Por lo tanto su esquina superior derecha estará en las coordenadas (x, y) = (cenX - ancho/2, cenY - alto/2). Aunque un graficos pueden ser alargados (si alto diferente a ancho), a efecto de las colisiones, vamos a considerar que son círculos. El radio de este círculo se podría calcular como ancho/2 ó alto/2, según tomáramos el radio mayor o el radio menor. Lo que hacemos es tomar una media de estos dos posibles radios: (ancho/2+alto/2)/2= (ancho+alto)/4. El método verificaColision()

comprueba si hemos colisionado con otro gráfico. Para ello se comprueba si la distancia al otro Grafico es menor a la suma de los dos radios de colisión.

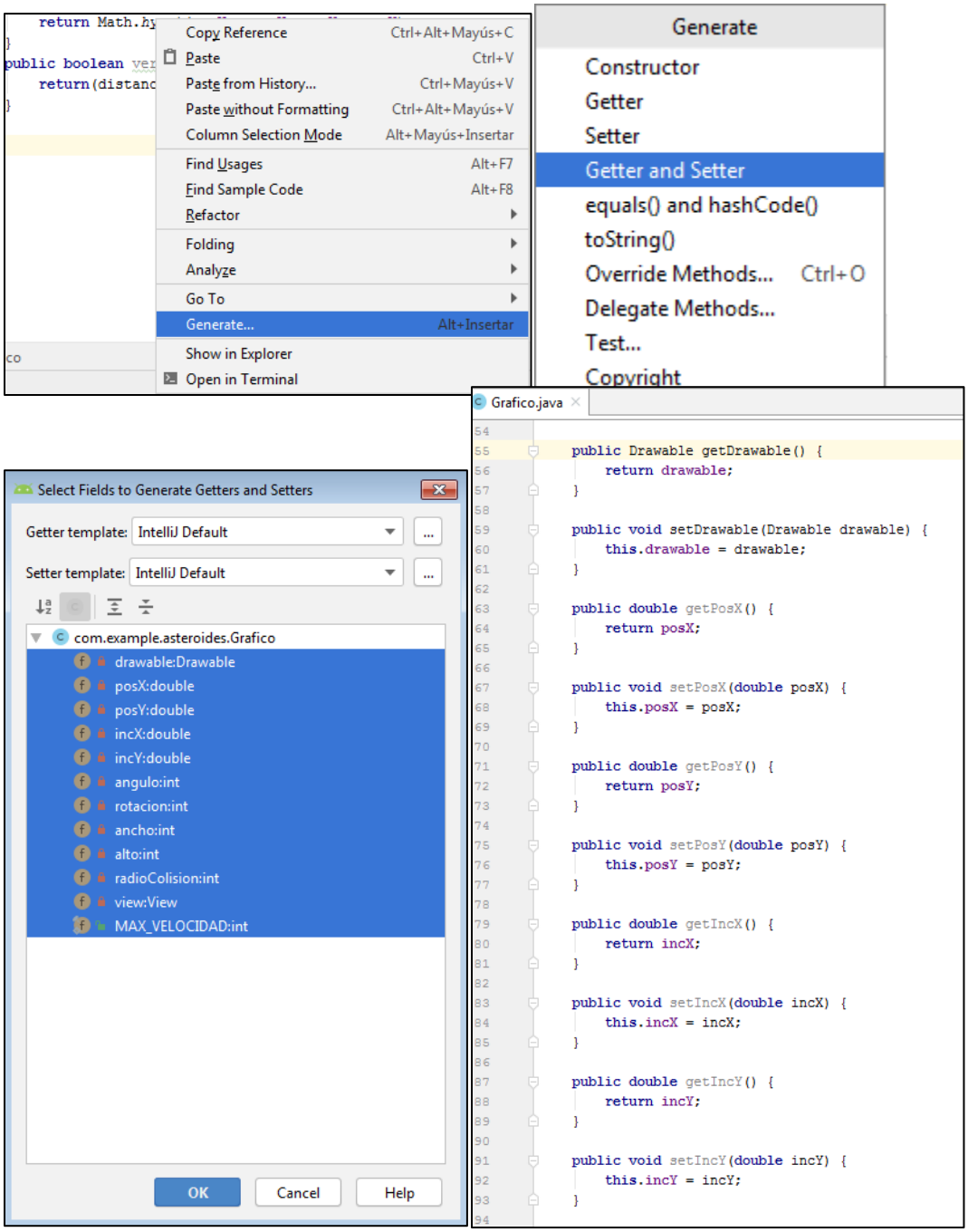


El método dibujaGrafico() se encarga de dibujar el Drawable del Grafico en un Canvas. Comienza indicando los límites donde se situará el Drawable, utilizando setBounds(). Luego, guarda la matriz de transformación del Canvas. A continuación, aplica una transformación de rotación según lo indicado en la variable angulo utilizando como eje de rotación (cenX, cenY). Se dibuja el Drawable en el Canvas y se recupera la matriz de transformación, para que la rotación introducida no se aplique en futuras operaciones con este Canvas.
Para finalizar hacemos dos llamadas al método invalidate() de la vista donde estamos dibujando el Grafico. Con este método informamos a la vista que tiene que ser redibujada. Para mejorar la eficiencia indicaremos solo el rectángulo que hemos modificado. De esta forma, la vista no tendrá que redibujarse en su totalidad. En un primer momento podríamos pensar que el rectángulo de invalidación coincide con el utilizado en setBounds(). Pero, hay que recordar que hemos realizado una rotación sobre el Drawable y, como puede verse en la ilustración de la izquierda, posiblemente el Drawable se salga de este rectángulo. Para resolver este problema vamos a aumentar el área de invalidación a un cuadrado con la mitad de su lado igual a la mitad de la diagonal de gráfico. Este valor es precalculado en la variable radioInval.



Hay que tener en cuenta que hemos desplazado el Drawable desde una posición anterior. Por lo tanto, también es necesario indicar a la vista que redibuje el área donde estaba antes el Grafico. Con este fin vamos a utilizar las variables xAnterior y yAnterior.
Otro método interesante es incrementaPos() que es utilizado para modificar la posición y ángulo del Grafico según la velocidad de translación (incX, incY) y la velocidad de rotación (rotacion). Este método tiene el parámetro, factor, que permite ajustar esta velocidad. Con un valor igual a 1, tendremos una velocidad normal; si vale 2, el grafico se mueve al doble de velocidad. En el juego original de Asteroides, si un gráfico salía por un lado de la pantalla, aparecía de nuevo por el lado opuesto. Este comportamiento es implementado en las últimas líneas del método

Paso 2. Al principio de la clase hemos definido varios campos con el modificador private. Vamos a necesitar acceder a estos campos desde fuera de la clase, por lo que resulta necesario declarar los métodos *get* y *set* correspondientes. Vamos a realizar esta tarea de forma automática utilizando la herramienta de Android Studio. Sitúa el cursor al final de la clase (justo antes de la última llave) y pulsa con el botón derecho. Selecciona en el menú desplegable Generate + *Getter and Setter…* En la ventana de diálogo marca todos los campos y pulsa *OK*. Android Studio hará el trabajo por nosotros.



Nota sobre Java: *En el tutorial Java Esencial / Encapsulamiento y visibilidad puedes aprender más sobre los métodos get y set. Lo encontrarás en la Web www.androidcurso.com.*

Parte IV: *La clase VistaJuego*

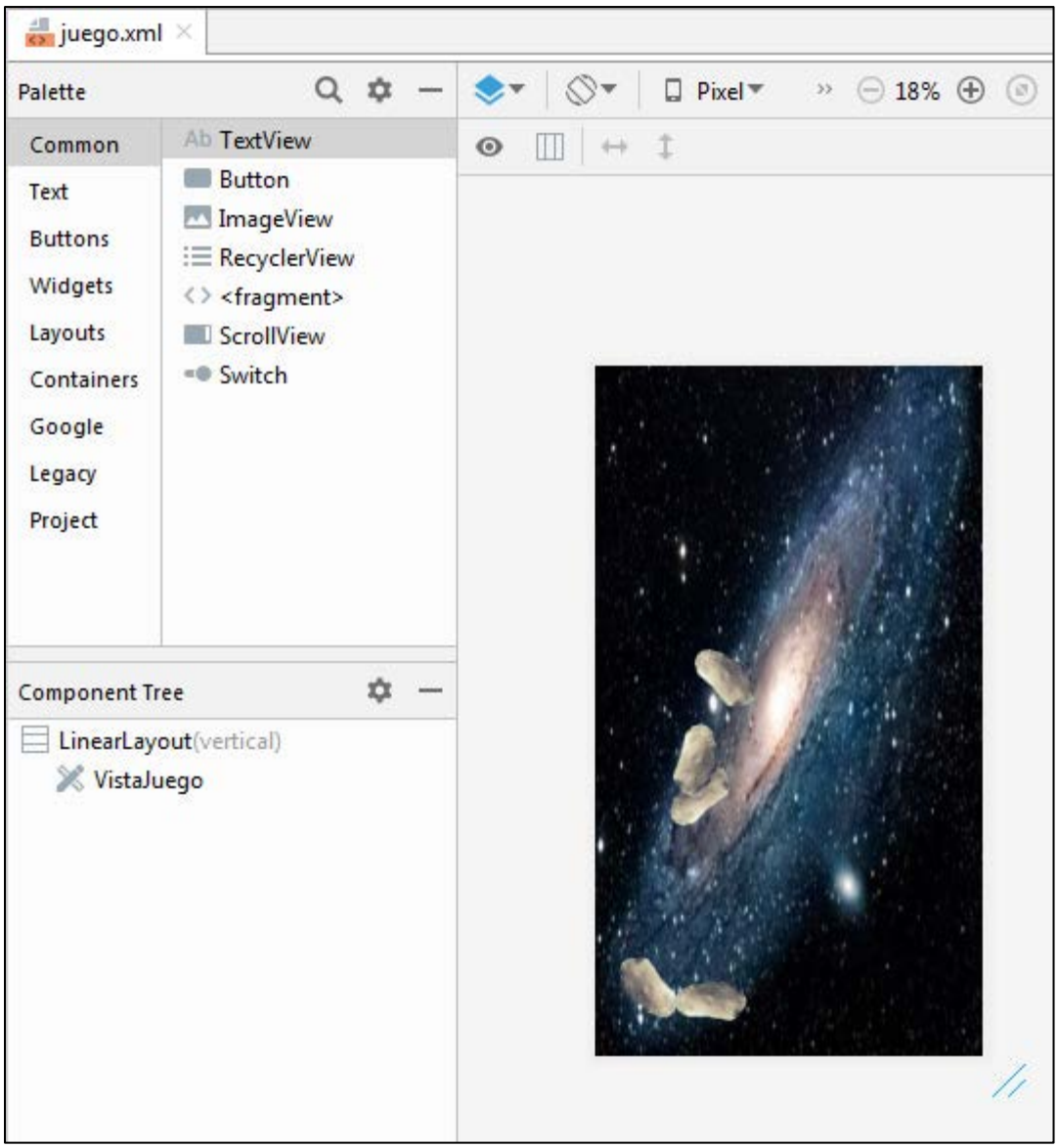
Paso 1. Crea una nueva clase VistaJuego en el proyecto *Asteroides* y copia el siguiente código:

VistaJuego.java

```
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 = ContextCompat.getDrawable(context, R.drawable.asteroide1);
        Asteroides = new Vector<Grafico>();
        for (int i = 0; i < numAsteroides; i++) {
            Grafico asteroide = new Grafico( view: 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);
        }
    }
}
```

Como ves se han declarado tres métodos. En el constructor creamos los asteroides e inicializamos su velocidad, ángulo y rotación. Sin embargo, resulta imposible iniciar su posición, dado que no conocemos el alto y ancho de la pantalla. Esta información será conocida cuando se llame a onSizeChanged(). Observa cómo en esta función los asteroides están situados de forma aleatoria. El último método, onDraw(), es el más importante de la clase View, dado que es el responsable de dibujar la vista.

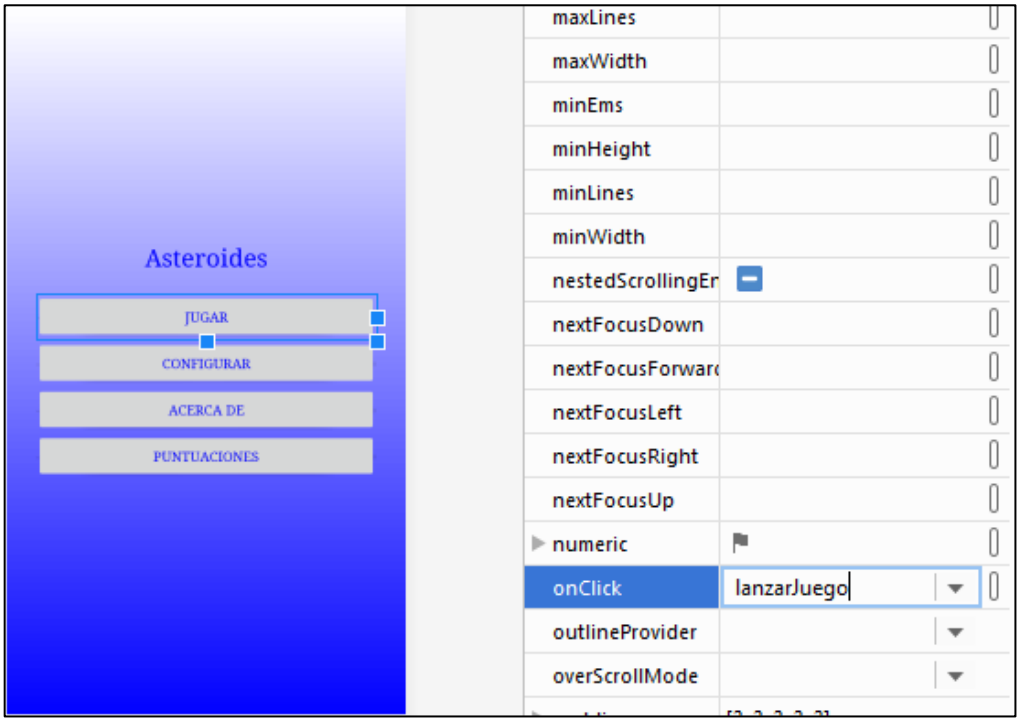
Paso 2. Hemos creado una vista personalizada. No tendría demasiado sentido, pero podrá ser utilizada en cualquier otra aplicación que desarrolles. Visualiza el *Layout* juego.xml y observa como la nueva vista se integra perfectamente en el entorno de desarrollo.



Paso 3. Registra la actividad Juego en *AndroidManifest.xml*.

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name=".AcercaDeActivity"
    android:label="Acerca de ..."
    android:theme="@style/Theme.AppCompat.Light.Dialog" />
<activity
    android:name=".Preferencias"
    android:label="Preferencias" />
<activity android:name=".Puntuaciones" />
<activity android:name=".Juego"></activity>
```

Paso 4. Asocia al atributo onClick del botón Jugar del Layout main el método lanzarJuego. Crea el método lanzarJuego dentro de la clase Asteroides, de forma similar al método lanzarAcercaDe, pero esta vez arrancando la actividad Juego.




```
public void lanzarJuego(View view){
    Intent i = new Intent( packageContext: this, Juego.class);
    startActivity(i);
}
```

Paso 5. Ejecuta la aplicación. Has de ver cinco asteroides repartidos al azar por la pantalla