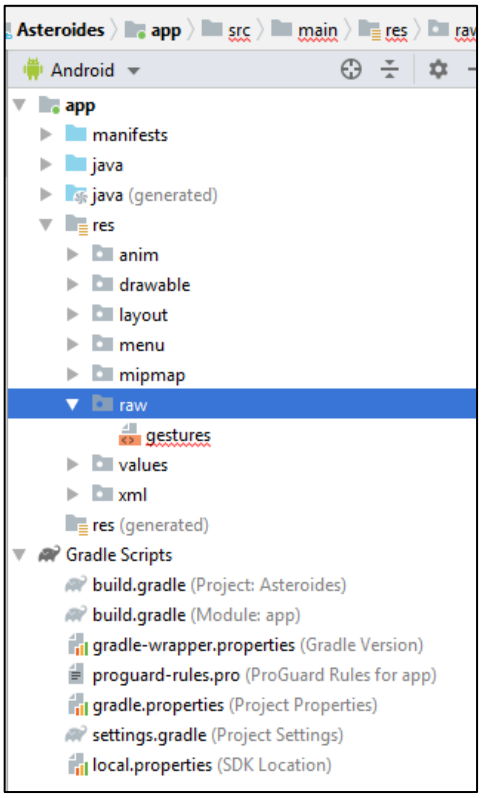
PRACTICA 20: ASTEROIDES IX

Parte V: Añadiendo gestures a Asteroides

Paso 1. Crea la carpeta res/raw y copia el fichero gestures, que contiene la librería creada anteriormente.



Paso 2. Modifica el *Layoutmain.xml* para que disponga de un GestureOverlayView. Añade al principio del res/layout/main.xml el siguiente código. Cierra la etiqueta al final del fichero.

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"
    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="Asteroides"
        android:gravity="center"
        android:textSize="25sp"
        android:layout_marginBottom="20dp"/>
    <Button
        android:id="@+id/button01"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="lanzarJuego"
        android:text="@string/Arrancar" />
        <!--style="@style/EstiloBoton"-->
    <Button
        android:id="@+id/button02"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="lanzarPreferencias"
        android:text="Configurar" />
        <!--style="@style/EstiloBoton.Alternativo"-->
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.gesture.GestureOverlayView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gestures"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gestureStrokeType="multiple"
    android:fadeOffset="800">
    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Asteroides"
        android:gravity="center"
        android:textSize="25sp"
        android:layout_marginBottom="20dp"/>
    <Button
        android:id="@+id/button01"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="lanzarJuego"
        android:text="@string/Arrancar" />
        <!--style="@style/EstiloBoton"-->
    <Button
        android:id="@+id/button02"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="lanzarPreferencias"
        android:text="Configurar" />
        <!--style="@style/EstiloBoton.Alternativo"-->
    • • • •
    <Button
        android:id="@+id/button04"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="lanzarPuntuaciones"
        android:text="Puntuaciones" />
</android.gesture.GestureOverlayView>
```

Paso 3. Cuando el usuario esté utilizando este Layout ha de poder introducir alguna de las cuatro gestures de la librería de forma que se ejecute la acción correspondiente.

En la clase MainActivity.java añade en la definición:

```
public class MainActivity extends AppCompatActivity implements GestureOverlayView.OnGesturePerformedListener {

    private GestureLibrary libreria;

    public static AlmacenPuntuaciones almacen= new AlmacenPuntuacionesLista();
    private Button bAcercaDe;
    private Button bSalir;
    private Button bPlay;
    private Button bConfigurar;
    private Animation animacion;

    @Override
    public void onGesturePerformed(GestureOverlayView gestureOverlayView, Gesture gesture) {

    }
```

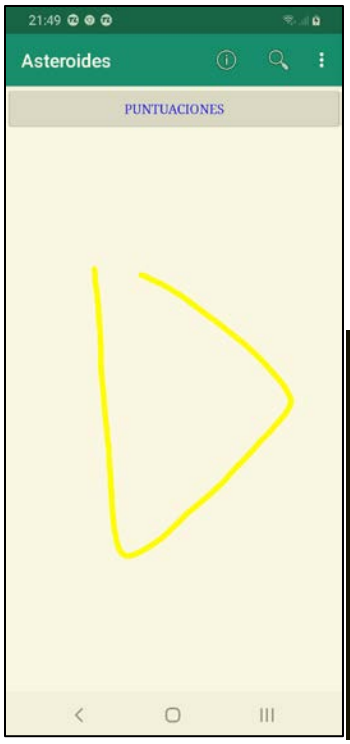
Paso 4. Añade al final del método onCreate:

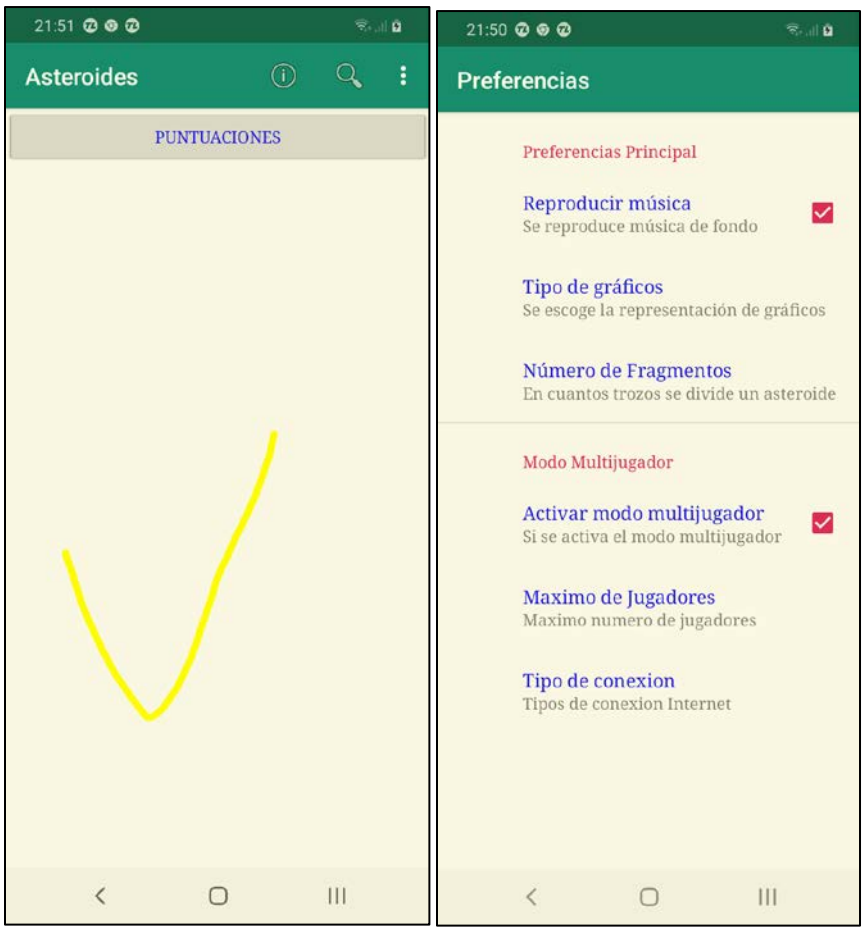
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

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

    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);

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

    libreria= GestureLibraries.fromRawResource( context: this, R.raw.gestures);
    if(!libreria.load()) {
        finish();
    }
    GestureOverlayView gesturesView = (GestureOverlayView) findViewById(R.id.gestures);
    gesturesView.addOnGesturePerformedListener(this);
}
```

Paso 5. Añade el siguiente método:

```
@Override
    public void onGesturePerformed(GestureOverlayView ov, Gesture gesture) {
     ArrayList<Prediction> predictions=libreria.recognize(gesture);
     if(predictions.size()>0){
         String comando = predictions.get(0).name;
         if(comando.equals("play")){
             lanzarJuego( view: null);
         } else if(comando.equals("configurar")){
             lanzarPreferencias( view: null);
         } else if(comando.equals("acerca_de")){
             lanzarAcercaDe( view: null);
         } else if(comando.equals("cancelar")){
             finish();
         }
     }
}
```

Paso 7. Ejecuta la aplicación y prueba el acceso a los diferentes apartados de la aplicación mediantes gestures.





Parte VI: Manejo de la nave con el sensor de orientación

Paso 1. En primer lugar, implementa la interfaz SensorEventListener.

public class VistaJuego **extends** View **implements** SensorEventListener {

```
public class VistaJuego extends View implements SensorEventListener {

    @Override
    public void onSensorChanged(SensorEvent sensorEvent) {

    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int i) {

    }
```

Paso 2. En el constructor registra el sensor e indica que nuestro objeto recogerá la llamada *callback*:

```
    SensorManager mSensorManager = (SensorManager) context.getSystemService(Context.SENSOR_SERVICE);
    List<Sensor> listSensors = mSensorManager.getSensorList(Sensor.TYPE_ORIENTATION);
    if (!listSensors.isEmpty()) {
        Sensor orientationSensor = listSensors.get(0);
        mSensorManager.registerListener( listener: this, orientationSensor, SensorManager.SENSOR_DELAY_GAME);
    }
}
```

Paso 3. Añade los siguientes dos métodos que implementan la interfaz SensorEventListener:

```
public class VistaJuego extends View implements SensorEventListener {

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy){}

    private boolean hayValorInicial = false;
    private float valorInicial;


    @Override
    public void onSensorChanged(SensorEvent event) {
        float valor = event.values[1];
        if (!hayValorInicial){
            valorInicial = valor;
            hayValorInicial = true;
        }
        giroNave=(int) (valor-valorInicial)/3 ;
    }
```

Paso 4. Prueba la aplicación. Has de tener cuidado de que el terminal este en una posición cómoda al entrar en la actividad Juego, dado que el movimiento de la nave se obtiene con la diferencia de la posición del terminal con respecto a la posición inicial.

Parte VII: Manejo de la nave con sensor de aceleración

Modifica el ejemplo anterior para utilizar el sensor de aceleración en lugar del de orientación. Gracias a la fuerza de gravedad que la Tierra ejerce sobre el terminal podremos saber si este está horizontal. En caso de que la nave este horizontal (o casi) no ha de girar, pero cuando el terminal se incline, la nave a de girar proporcionalmente a esta inclinación. Utiliza los programas anteriores para descubrir que eje (x, y o z) es el que te interesa y el rango de valores que proporciona.

Parte VIII: Aceleración de la nave con sensores

¿Te animarías a controlar la aceleración de la nave con los sensores? Ten cuidado de que no acelere con mucha facilidad, este juego resulta muy difícil cuando la nave está en movimiento. Puede ser una buena idea que permitas también decelerar la nave.

Parte IX: Configuración de tipo de entrada en preferencias

Todos los controles de la nave (teclado, pantalla táctil y sensores) están activados simultáneamente. El teclado y la pantalla táctil no interfieren cuando el usuario no quiere utilizarlos. Sin embargo, la activación de los sensores sí que molestará a los usuarios que no quieran utilizar este método de entrada.

Paso 1. Crea nuevas entradas en la configuración para activar o desactivar cada tipo de entrada (o al menos la de los sensores).

Paso 2. Modifica el código anterior para que se desactiven las entradas que el usuario no haya seleccionado.

Parte X: Introduciendo un misil en Asteroides

Paso 1. En primer lugar añade las siguientes variables a la clase VistaJuego:

```
// //// MISIL //////
private Grafico misil;
private static int PASO_VELOCIDAD_MISIL = 12;
private boolean misilActivo = false;
private int tiempoMisil;
```

Paso 2. Para trabajar con gráficos vectoriales, puedes crear en el constructor la variable drawableMisil de la siguiente forma:

```
    //MISIL
    ShapeDrawable dMisil = new ShapeDrawable(new RectShape());
    dMisil.getPaint().setColor(Color.WHITE);
    dMisil.getPaint().setStyle(Paint.Style.STROKE);
    dMisil.setIntrinsicWidth(15);
    dMisil.setIntrinsicHeight(3);
    drawableMisil = dMisil;

    setBackgroundColor(Color.BLACK);

} else{
```

Paso 3. Crea la variable drawableMisil para el caso de que se deseen gráficos en *bitmap*, utilizando el fichero misil1.png.

```
    //MISIL
    ShapeDrawable dMisil = new ShapeDrawable(new RectShape());
    dMisil.getPaint().setColor(Color.WHITE);
    dMisil.getPaint().setStyle(Paint.Style.STROKE);
    dMisil.setIntrinsicWidth(15);
    dMisil.setIntrinsicHeight(3);
    drawableMisil = dMisil;

    setBackgroundColor(Color.BLACK);

} else{
    drawableAsteroide = ContextCompat.getDrawable(context, R.drawable.asteroide1);
    drawableNave =  ContextCompat.getDrawable(context, R.drawable.nave);
    drawableMisil =  ContextCompat.getDrawable(context, R.drawable.misil1);
}
```

Paso 4. Inicializa el objeto misil, de forma similar a como se ha hecho en nave.

```
nave = new Grafico( view: this, drawableNave);
Asteroides = new Vector<Grafico>();
misil = new Grafico( view: this, drawableMisil);
```

Paso 5. En el método onDraw() dibuja misil, solo si lo indica la variable misilActivo.

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

    if (misilActivo) misil.dibujaGrafico(canvas);

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

Paso 6. Quita los comentarios de las llamadas a ActivaMisil().

```
@Override
public boolean onTouchEvent (MotionEvent event) {
    super.onTouchEvent(event);
    float x = event.getX();
    float y = event.getY();
    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            disparo=true;
            break;
        case MotionEvent.ACTION_MOVE:
            float dx = Math.abs(x - mX);
            float dy = Math.abs(y - mY);
            if (dy<6 && dx>6){          //Desplazamiento horizontal produce giro
                giroNave = Math.round((x - mX) / 2);
                disparo = false;
            } else if (dx<6 && dy>6){  //Desplazamiento vertical produce aceleracion
                aceleracionNave = Math.abs(Math.round((mY - y) / 25));
                disparo = false;
            }
            break;
        case MotionEvent.ACTION_UP:    //Pulsacion sin movimiento, provoca un disparo
            giroNave = 0;
            aceleracionNave = 0;
            if (disparo){
                ActivaMisil();
            }
            break;
    }
    mX=x; mY=y;
    return true;
}
```

```
@Override
public boolean onKeyDown(int codigoTecla, KeyEvent evento) {
    super.onKeyDown(codigoTecla, evento);
    // Suponemos que vamos a procesar la pulsación
    boolean procesada = true;
    switch (codigoTecla) {
        case KeyEvent.KEYCODE_DPAD_UP:
            aceleracionNave = +PASO_ACELERACION_NAVE;
            break;
        case KeyEvent.KEYCODE_DPAD_LEFT:
            giroNave = -PASO_GIRO_NAVE;
            break;
        case KeyEvent.KEYCODE_DPAD_RIGHT:
            giroNave = +PASO_GIRO_NAVE;
            break;
        case KeyEvent.KEYCODE_DPAD_CENTER:
        case KeyEvent.KEYCODE_ENTER:
            ActivaMisil();
            break;
        default:  // Si estamos aquí, no hay pulsación que nos interese
            procesada = false;
            break;
    }
    return procesada;
}
```

Paso 7. En el método actualizaFisica() añade las siguientes líneas:

```
    // Actualizamos posición de misil
    if (misilActivo) {
        misil.incrementaPos(retardo);
        tiempoMisil-=retardo;
        if (tiempoMisil < 0) {
            misilActivo = false;
        } else {
            for (int i = 0; i < Asteroides.size(); i++)
                if (misil.verificaColision(Asteroides.elementAt(i))) {
                    destruyeAsteroide(i);
                    break;
                }
        }
    }
}
```

Paso 8. Añade los siguientes dos métodos:

```
private void destruyeAsteroide(int i) {
    Asteroides.remove(i);
    misilActivo = false;
}

private void ActivaMisil() {
    misil.setPosX(nave.getPosX());
    misil.setPosY(nave.getPosY());
    misil.setAngulo(nave.getAngulo());
    misil.setIncX(Math.cos(Math.toRadians(misil.getAngulo())) * PASO_VELOCIDAD_MISIL);
    misil.setIncY(Math.sin(Math.toRadians(misil.getAngulo())) * PASO_VELOCIDAD_MISIL);
    tiempoMisil = (int) Math.min(this.getWidth() / Math.abs( misil.getIncX()),
            this.getHeight() / Math.abs(misil.getIncY())) - 2;
    misilActivo = true;
}
```

Este último método requiere alguna explicación. Cuando se quiere activar un nuevo misil este ha de partir del centro de la nave. Como las coordenadas X, Y de Grafico, corresponden a la esquina superior izquierda hay que hacer algunos ajustes. El ángulo del misil ha de ser el mismo que actualmente tiene la nave. El módulo de la velocidad de la nave nos la indica la constante PASO_VELOCIDAD_MISIL. Para descomponerla en sus componentes X e Y utilizamos el coseno y el seno. Dada la naturaleza del espacio del juego (lo que sale por un lado aparece por el otro) si disparáramos un misil este podría acabar chocando contra la nave. Para solucionarlo vamos a dar un tiempo de vida al misil para impedir que pueda llegar de nuevo a la nave (tiempoMisil). Para obtener este tiempo nos quedamos con el mínimo entre el ancho dividido la velocidad en X y el alto dividido entre la velocidad en Y. Luego le restamos una constante. Terminamos activando el misil.

Paso 9. Verifica que todo funciona correctamente.

NOTA: *Este código es posible que de algún problema de acceso concurrente a los datos desde dos threads diferentes. Se resolverá en el siguiente capítulo.*

Parte XI: Disparando varios misiles a la vez

Tal y como se ha planteado el código solo es posible lanzar un misil cada vez. Si disparamos un segundo misil el primero desaparece. ¿Podrías modificar el código para que se pudieran lanzar tantos misiles cómo quisieras? Si no tienes muy claro por donde empezar, a continuación se plantean los pasos para una posible solución:

Paso 1. Elimina la variable misil, y en su lugar crea una vector de gráficos:

prívate Vector <Grafico>Misiles;

```
// //// MISIL //////
//private Grafico misil;
private Vector <Grafico> Misiles;
private static int PASO_VELOCIDAD_MISIL = 12;
private boolean misilActivo = false;
private int tiempoMisil;
```

Paso 2. Elimina la variable misilActivo. Cuando el array de misiles está vacío querrá decir que no hay ningún misil activo.

```
// //// MISIL //////
//private Grafico misil;
private Vector <Grafico> Misiles;
private Vector <Boolean> misilesActivos;
private static int PASO_VELOCIDAD_MISIL = 12;
//private boolean misilActivo = false;
private int tiempoMisil;
```

Paso 3. Elimina la variable tiempoMisil, y en su lugar crea una vector de enteros:

prívate Vector <Integer>tiempoMisiles;

```
// //// MISIL //////
//private Grafico misil;
private Vector <Grafico> Misiles;
private Vector <Boolean> misilesActivos;
private Vector <Integer> tiempoMisiles;
private static int PASO_VELOCIDAD_MISIL = 12;
private int numMisiles = 10;
//private boolean misilActivo = false;
//private int tiempoMisil;
```

Los elementos de los arrays Misiles y tiempoMisiles han de estar emparejados. Es decir, al misil en posición *x* de Misiles le quedará un tiempo que se almacenará en la posición *x* de tiempoMisiles.

***Nota sobre Java**: Observa como la variable tiempoMisil antes era de tipo int, pero ahora tiempoMisiles es en vector de Integer, no de int. Estos dos tipos de datos representan un número entero, pero el primero es una clase y el segundo un tipo simple. Hemos tenido que realizar este cambio dado que la clase Vector solo admite elementos que sean clases. Para más información consultar Apendice A: Referencia Java, Envolventes (wrappers).*

Paso 4. En los métodos onDraw() y actualizaFisica() tendrás que añadir un bucle para recorrer todos los misiles.

En el Constructor de VistaJuego, creamos los vectores Misiles, misilesActivo y tiempoMisiles

```
//////////////2. CONSTRUCTOR/////////////////////
//misil = new Grafico(this, drawableMisil);
Misiles = new Vector<Grafico>();
tiempoMisiles = new Vector<Integer>();
misilesActivos = new Vector<Boolean>();
for (int i=0; i<numMisiles; i++){
    Misiles.add(new Grafico( view: this, drawableMisil));
    misilesActivos.add(false);
    tiempoMisiles.add(0);
}
////////////////////////////////////////////////////
```

En onDraw:

```
//////////3. ONDRAW/////////////////
/*@Override
protected synchronized void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    nave.dibujaGrafico(canvas);
    if (misilActivo) misil.dibujaGrafico(canvas);
    for (Grafico asteroide: Asteroides) {
        asteroide.dibujaGrafico(canvas);
    }
}*/

@Override
protected synchronized void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    nave.dibujaGrafico(canvas);
    for (int p=0;p<Misiles.size();p++){
        if (misilesActivos.elementAt(p))
            Misiles.elementAt(p).dibujaGrafico(canvas);
    }
    for (Grafico asteroide: Asteroides) {
        asteroide.dibujaGrafico(canvas);
    }
}
```

En actualizarFisica:

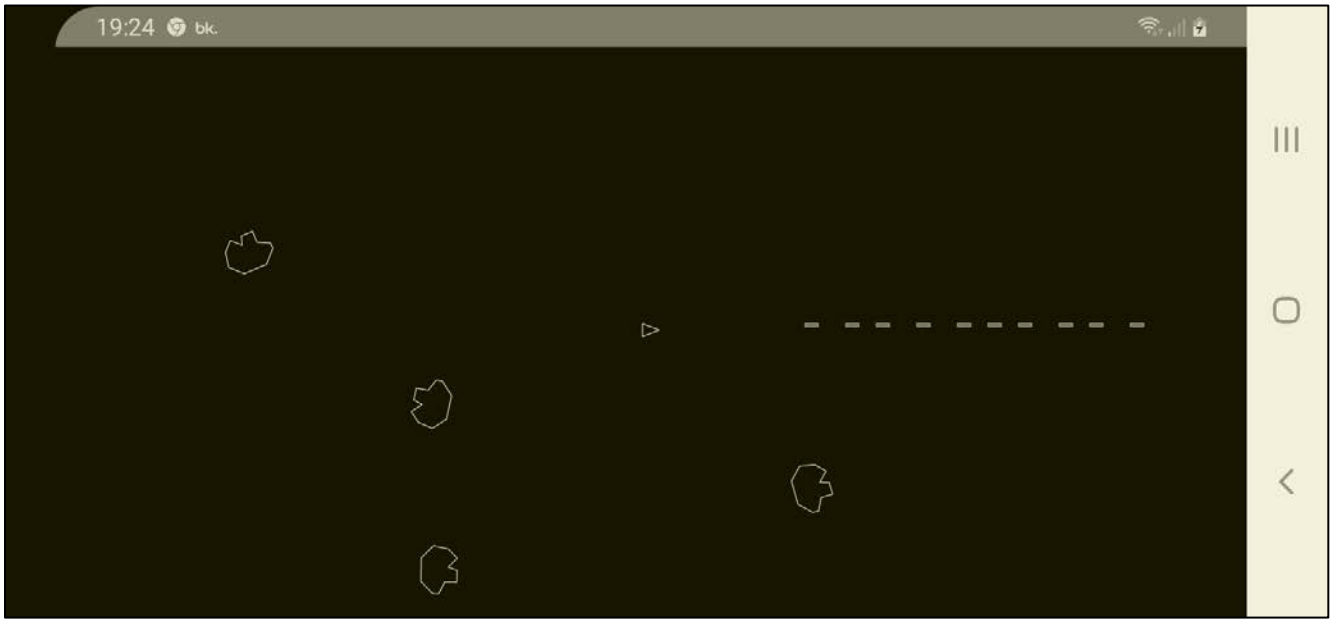
```
////// 4. ACTUALIZA POSICION MISIL//////////
/*if (misilActivo) {
    misil.incrementaPos(retardo);
    tiempoMisil-=retardo;
    if (tiempoMisil < 0) {
        //misilActivo = false;
    } else {
        for (int i = 0; i < Asteroides.size(); i++)
            if (misil.verificaColision(Asteroides.elementAt(i))) {
                destruyeAsteroide(i);
                break;
            }
    }
}*/

for (int p=0; p<Misiles.size();p++){
    if (misilesActivos.elementAt(p)){
        Misiles.elementAt(p).incrementaPos(retardo);
        tiempoMisiles.set(p, tiempoMisiles.get(p)-(int)retardo);
        if (tiempoMisiles.elementAt(p)<0){
            misilesActivos.set(p, false);
        }else{
            for (int i = 0; i < Asteroides.size(); i++) {
                if (Misiles.elementAt(p).verificaColision(Asteroides.elementAt(i))) {
                    Asteroides.remove(i);
                    misilesActivos.set(p, false);
                    break;
                }
            }
        }
    }
}
```

En ActivaMisil

```
/////////5. ACTIVA MISIL////////////////
/*private void ActivaMisil() {
    misil.setPosX(nave.getPosX());
    misil.setPosY(nave.getPosY());
    misil.setAngulo(nave.getAngulo());
    misil.setIncX(Math.cos(Math.toRadians(misil.getAngulo())) * PASO_VELOCIDAD_MISIL);
    misil.setIncY(Math.sin(Math.toRadians(misil.getAngulo())) * PASO_VELOCIDAD_MISIL);
    tiempoMisil = (int) Math.min(this.getWidth() / Math.abs( misil.getIncX()),
            this.getHeight() / Math.abs(misil.getIncY())) - 2;
    misilActivo = true;
}*/
private void ActivaMisil() {
    for(int p=0;p<Misiles.size();p++) {
        if (!misilesActivos.elementAt(p)){
            Misiles.elementAt(p).setPosX(nave.getPosX());
            Misiles.elementAt(p).setPosY(nave.getPosY());
            Misiles.elementAt(p).setAngulo(nave.getAngulo());
            Misiles.elementAt(p).setIncX(Math.cos(Math.toRadians(Misiles.elementAt(p).getAngulo())) * PASO_VELOCIDAD_MISIL);
            Misiles.elementAt(p).setIncY(Math.sin(Math.toRadians(Misiles.elementAt(p).getAngulo())) * PASO_VELOCIDAD_MISIL);
            tiempoMisiles.set(p,(int) Math.min(this.getWidth() / Math.abs(Misiles.elementAt(p).getIncX()),
                this.getHeight() / Math.abs(Misiles.elementAt(p).getIncY())) - 2);
            misilesActivos.set(p,true);
            break;
        }
    }
}
```

Paso 5. Observa la siguiente línea. Permite decrementar el elemento m de tiempoMisil

tiempoMisiles.set(m, tiempoMisiles.get(m)-1);



NOTA: Para volver a tener la interfaz anterior sin gestures, tal y como muestra la imagen siguiente:

Debemos hacer los siguientes cambios en MainActivity.java en el método onCreate y en activity_main.xml:

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

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

    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);

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

    libreria= GestureLibraries.fromRawResource( context: this, R.raw.gestures);
    if(!libreria.load()) {
        finish();
    }
    //GestureOverlayView gesturesView = (GestureOverlayView) findViewById(R.id.gestures);
    //gesturesView.addOnGesturePerformedListener(this);
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--<android.gesture.GestureOverlayView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gestures"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gestureStrokeType="multiple"
    android:fadeOffset="800">-->
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:padding="@dimen/margen_botones"
    tools:context=".MainActivity" >
    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Asteroides"
        android:gravity="center"
        android:textSize="25sp"
        android:layout_marginBottom="20dp"/>
    <Button
        android:id="@+id/button01"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="lanzarJuego"
        android:text="Jugar" />
        <!--style="@style/EstiloBoton"-->
```