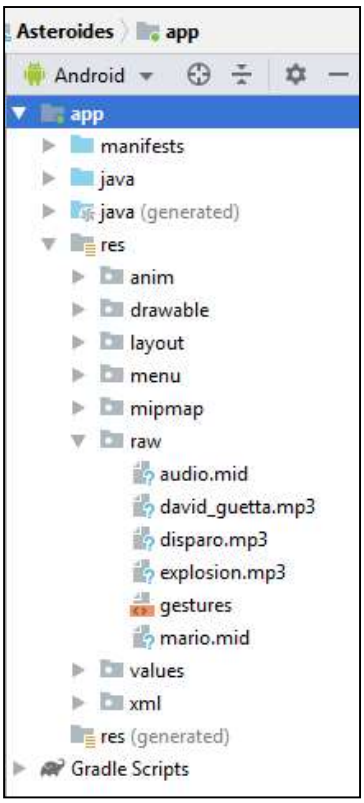
## PRACTICA 2: ASTEROIDES XI

### Parte VII: Introduciendo efectos de audio con MediaPlayer en Asteroides

**Paso 1.** Abre el proyecto Asteroides.

**Paso 2.** Copia los siguientes ficheros disparo.mp3 y explosion.mp3 a la carpeta *res/raw*:




**Paso3**. Abre la clase VistaJuego y añade las siguientes variables:

```
public class VistaJuego extends View implements SensorEventListener {

    MediaPlayer mpDisparo, mpExplosion;
```

**Paso 4.** En el constructor de la clase inicialízalas de la siguiente forma:

```
public VistaJuego(Context context, AttributeSet attrs) {
    super(context, attrs);

    mpDisparo= MediaPlayer.create(context,R.raw.disparo);
    mpExplosion= MediaPlayer.create(context,R.raw.explosion);
```

**Paso5**. Añade en el método ActivaMisil() de VistaJuego:

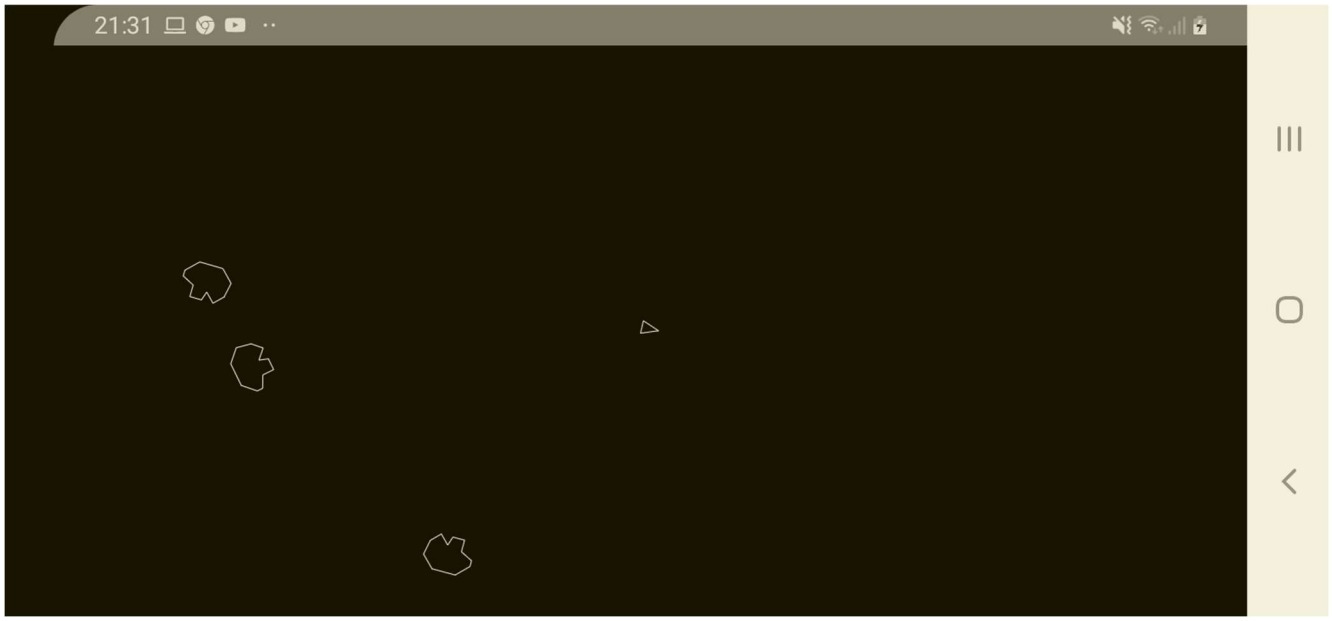
```
/////////5. ACTIVA MISIL////////////////
/*private void ActivaMisil() {
    misil.setPosX(nave.getPosX());
    misil.setPosY(nave.getPosY());
    misil.setAngulo(nave.getAngulo());
    misil.setIncX(Math.cos(Math.toRadians(misil.getAngulo())) * PASO_VELOCIDAD_MISIL);
    misil.setIncY(Math.sin(Math.toRadians(misil.getAngulo())) * 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);

            mpDisparo.start();

            break;
        }
    }
}
```

**Paso 6.** Añade en el método destruyeAsteroide() de VistaJuego:

```
////// 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);

                    mpExplosion.start();

                    break;
                }
            }
        }
    }
}
```

**Paso7**. Ejecuta el programa y verifica el resultado. ¿Qué pasa cuando disparamos o destruimos un asteroide? ¿El sonido se oye de forma inmediata?

No, se oye de forma desacompasada y a destiempo




## Parte VIII: Introduciendo efectos de audio con SoundPool

**Paso 1.** El proyecto Asteroides elimina todo el código introducido a partir del punto 3, del ejercicio anterior.

Eliminamos o comentamos todo el siguiente código

MediaPlayer *mpDisparo*, *mpExplosion*; //En la declaración de datos miembro de VistaJuego

*mpDisparo*= MediaPlayer.*create*(context,R.raw.*disparo*);//En el constructor de VistaJuego

*mpExplosion*= MediaPlayer.*create*(context, R.raw.*explosion*);//En el constructor de VistaJuego

mpDisparo.start();//En la rutina activaMisil

mpExplosion.start();

```
public class VistaJuego extends View implements SensorEventListener {

    //MediaPlayer mpDisparo, mpExplosion;
```

```
public VistaJuego(Context context, AttributeSet attrs) {
    super(context, attrs);

    //mpDisparo= MediaPlayer.create(context,R.raw.disparo);
    //mpExplosion= MediaPlayer.create(context,R.raw.explosion);
```

```
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);

                    //mpExplosion.start();

                    break;
                }
            }
        }
    }
}
```

```
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);

            //mpDisparo.start();

            break;
        }
    }
}
```

**Paso2**. Abre la clase VistaJuego y añade las siguientes variables:

```
public class VistaJuego extends View implements SensorEventListener {

    // //// MULTIMEDIA //////
    SoundPool soundPool;
    int idDisparo, idExplosion;
```

**Paso 3.** En el constructor de la clase inicialízalas de la siguiente forma:

```
public VistaJuego(Context context, AttributeSet attrs) {
    super(context, attrs);

    //soundPool = new SoundPool( 5, AudioManager.STREAM_MUSIC, 0);
    soundPool = new SoundPool.Builder().setMaxStreams(5) .build();
    idDisparo = soundPool.load(context, R.raw.disparo,  priority: 0);
    idExplosion = soundPool.load(context, R.raw.explosion,  priority: 0);
```

En el constructor de SoundPool hay que indicar tres parámetros. El primero corresponde al máximo de reproducciones simultáneas. El segundo es el tipo de stream de audio (normalmente STREAM_MUSIC). El tercero es la calidad de reproducción, aunque actualmente no se implementa.

Cada una de las pistas ha de ser cargada mediante el método load(). Existen muchas sobrecargas para este método. La versión utilizada en el ejemplo permite cargar las pistas desde los recursos. El último parámetro corresponde a la prioridad, aunque actualmente no tiene ninguna utilidad.

**Paso 4**. Añade en el método ActivaMisil() de VistaJuego:

```
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);

            //mpDisparo.start();
            soundPool.play(idDisparo,  leftVolume: 1,  rightVolume: 1,  priority: 1,  loop: 0,  rate: 1);

            break;
        }
    }
}
```

El método play() permite reproducir una pista. Hay que indicarle:

- **soundID**: El identificador de pista, devuelto por la función load();
- **leftVolume:** Volumen para el canal izquierdo (rango = 0.0 a 1.0).
- **rightVolume:** Volumen para el canal derecho (rango = 0.0 a 1.0).
- **priority:** Prioridad de stream (0 = prioridad más baja)
- **loop:** El número de repeticiones (-1= siempre, 0=solo una vez, 1=repetir una vez, … ).
- **rate:** Ratio de reproducción, con el que podremos modificar la velocidad o pitch (1.0 reproducción normal, rango: 0.5 a 2.0).

**Paso 5.** Añade en el método destruyeAsteroide() de VistaJuego:

```
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);

                    //mpExplosion.start();
                    soundPool.play(idExplosion,  leftVolume: 1,  rightVolume: 1,  priority: 0,  loop: 0,  rate: 1);

                    break;
                }
            }
        }
    }
}
```

Los parámetros utilizados para la explosión son similares, solo hemos introducido una prioridad menor. La consecuencia será que si ya hay un total de 5 (ver constructor) pistas reproduciéndose y se pide la

reproducción de un nuevo disparo. El sistema detendrá la reproducción de la explosión más antigua, por tener esta menos prioridad.

**Paso 6.** Ejecuta el programa y verifica el resultado. ¿Qué pasa cuando disparamos o destruimos un asteroide? ¿El sonido se oye de forma inmediata?

Si, se oye mejor pero se puede mejorar el resultado quizás incrementando el ratio de reproducción.

**Paso 7.** Modifica algunos de los parámetros del método play() y verifica el resultado.

Haremos soundPool.play(idDisparo, 1, 0, 1, 0, 2)

soundPool.play(idExplosion, 1, 0, 0, 0, 2);

Eliminamos un canal para intentar eliminar posible distorsión e incrementamos el ratio de reproducción.

Hay una mejora en el tiempo de respuesta y rapidez de ejecución de los sonidos