Utilizando multimedia en Android

La integración de contenido multimedia en nuestras aplicaciones resulta muy sencilla gracias a la gran variedad de facilidades que nos proporciona el API.

Concretamente podemos reproducir audio y vídeo desde orígenes distintos:

- Desde un fichero almacenado en el dispositivo.
- Desde un recurso que está incrustado en el paquete de la aplicación (fichero *.apk*).
- Desde un *stream* que es leído desde una conexión de red. En este punto admite dos posibles protocolos (*http://* y *rstp://*)

También resulta sencilla la grabación de audio y vídeo, siempre que el *hardware* del dispositivo lo permita.

En la siguiente lista se muestran las clases de Android que nos permitirán acceder a los servicios Multimedia:

- ✓ **MediaPayer**: Reproducción de audio/vídeo desde ficheros o *streams*.
- ✓ **MediaControler**: Visualiza controles estándar para `MediaPlayer` (pausa, stop...).
- ✓ **VideoView**: Vista que permite la reproducción de vídeo.
- ✓ **MediaRecorder**: Permite grabar audio y vídeo.
- ✓ **AsyncPlayer**: Reproduce lista de audios desde un *thread* secundario.
- ✓ **AudioManager**: Gestiona varias propiedades del sistema (volumen, tonos…).
- ✓ **AudioTrack**: Reproduce un búfer de audio PCM directamente por *hardware*.
- ✓ **SoundPool**: Maneja y reproduce una colección de recursos de audio.
- ✓ **JetPlayer**: Reprocuce audio y video interactivo creado con `JetCreator`.
- ✓ **Camera:** Cómo utilizar la cámara para tomar fotos y video.
- ✓ **FaceDetector**: Identifica la cara de la gente en un bitmap.

La plataforma Android soporta una gran variedad de formatos, muchos de los cuales pueden ser tanto decodificados como codificados. A continuación, mostramos una tabla con los formatos multimedia soportados. No obstante algunos modelos de móviles pueden soportar formatos adicionales que no se incluyen en la tabla, como por ejemplo DivX.

Cada desarrollador es libre de usar los formatos incluidos en el núcleo del sistema o aquellos que solo se incluyen en algunos dispositivos.

Tipo	Formato	Codifica	Decodifica	Detalles	fichero soportado
Audio	AAC LC/LTP	X	X	Mono/estéreo con cualquier combinación estándar de frecuencia > 160 Kbps y ratios de muestreo de 8 a 48kHz	3GPP (.3gp) MPEG-4(.mp4) No soporta raw AAC (.aac) MPEG-TS (.ts)
	HE-AACv1	a partir 4.1	X		
	HE-AACv2		X		
	AAC ELD	a partir 4.1	a partir 4.1	Mono/estéreo, 16-8kHz	
	AMR-NB	X	X	4.75 a 12.2 Kbps muestreada a @ 8kHz	3GPP (.3gp)
	AMR-WB	X	X	9 ratios de 6.60 Kbps a 23.85 Kbps a @ 16kHz	3GPP (.3gp)
	MP3		X	Mono/estéreo de 8 a 320 Kbps, bit rate constante (CBR) o variable (VBR)	MP3 (.mp3)
	MIDI		X	MIDI tipo 0 y 1. DLS v1 y v2. XMF y XMF móvil. Soporte para tonos de llamada RTTTL / RTX, OTA y iMelody.	Tipo 0 y 1 (.mid, .xmf, .mxmf). RTTTL / RTX (.rtttl, .rtx), OTA (.ota) iMelody (.imy)
	Ogg Vorbis		X		Ogg (.ogg) Matroska (.mkva partir 4.0)
	FLAC		a partir 3.1	mono/estereo (no multicanal)	FLAC (.flac)
	PCM/WAVE	a partir 4.1	X	8 y 16 bits PCM lineal (frecuencias limitadas por el hardware)	WAVE (.wav)
Imagen	JPEG	X	X	Base + progresivo	JPEG (.jpg)
	GIF		X		GIF (.gif)
	PNG	X	X		PNG (.png)
	BMP		X		BMP (.bmp)
	WEBP	a partir 4.0	a partir 4.0		WebP (.webp)
Video	H.263	X	X		3GPP (.3gp) MPEG-4 (.mp4)
	H.264 AVC	a partir 3.0	X	Baseline Profile (BP)	3GPP (.3gp) MPEG-4 (.mp4)
	MPEG-4 SP		X		3GPP (.3gp)
	VP8	a partir 4.3	a partir 2.3.3	Streaming a partir 4.0	WebM (.webm) Matroska (.mkv)

Formatos multimedia soportados en Android.

La vista `VideoView`

La forma más fácil de incluir un vídeo en una aplicación es incluir una vista de tipo `VideoView`. Veamos cómo hacerlo en el siguiente ejemplo:

Ejemplo: Reproducir un video con ViedoView

1. Crea una nueva aplicación.

2.- Damos permiso de acceso a internet en el Manifest

```
<uses-permission android:name="android.permission.INTERNET" />
```

3. Añadimos un `VideoView` a nuestro Layout

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:weightSum="1">
    <VideoView
        android:id="@+id/surface_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

4. Lo programamos

```
public class MainActivity extends AppCompatActivity {

    private VideoView mVideoView;

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

        mVideoView = (VideoView)findViewById(R.id.surface_view);

        String URLstring ="http://www.sample-videos.com/" +
            "video/mp4/720/big_buck_bunny_720p_1mb.mp4";

        mVideoView.setVideoURI(Uri.parse(URLstring));

         //si queremos reproducir un fichero local
        //mVideoView.setVideoPath("/mnt/sdcard/video.mp4");

        mVideoView.start();
        mVideoView.requestFocus();

    }
}
```

Si queremos reproducir un fichero local, en el parámetro del método `setVideoPath()` del objeto `VideoView` indicamos un fichero local.

Ejecutamos la aplicación y observamos el resultado.

5.- Si queremos añadir controles:

Añadimos antes del start (antes de `mVideoView.start();`)

```
mVideoView.setMediaController(new MediaController(this));
```

E incluimos el import correspondiente

```
import android.widget.MediaController;
```

Nos quedaría asi:

```
package com.example.amaia.ud03ej01multimedia;

import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.MediaController;
import android.widget.VideoView;

public class MainActivity extends AppCompatActivity {

    private VideoView mVideoView;

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

        mVideoView = (VideoView)findViewById(R.id.surface_view);
        //Si queremos utilizar un streaming
        String URLstring ="http://www.sample-videos.com/" +
                "video/mp4/720/big_buck_bunny_720p_1mb.mp4";
        mVideoView.setVideoURI(Uri.parse(URLstring));

        //si queremos reproducir un fichero local
        //mVideoView.setVideoPath("/mnt/sdcard/video.mp4");

        //Para añadir controles
        mVideoView.setMediaController(new MediaController(this));

        mVideoView.start();
        mVideoView.requestFocus();

    }
}
```

La clase `MediaPlayer`

La reproducción multimedia en Android se lleva a cabo principalmente por medio de la clase `MediaPlayer`. Veremos las características más importantes de esta clase y cómo podemos sacarle provecho.

Un objeto `MediaPlayer` va a poder pasar por gran variedad de estados: inicializados sus recursos (*initialized*), preparando la reproducción (*preparing*), preparado para reproducir (*prepared*), reproduciendo (*started*), en pausa (*paused*), parado (*stopped*), reproducción completada (*playback completed*), finalizado (*end*) y con error (*error*). Resulta importante conocer en qué estado se encuentra dado que muchos de los métodos solo pueden ser llamados desde ciertos estados. Por

ejemplo, no podemos ponerlo en reproducción (método `start()`) si no está en estado preparado. O no podremos ponerlo en pausa (`pause()`), si está parado. Si llamamos a un método no admitido para un determinado estado se producirá un error de ejecución.

La siguiente tabla permite conocer los métodos que podemos invocar desde cada uno de los estados y cuál es el nuevo estado al que iremos tras invocarlo. Existen dos tipos de métodos, los que no están subrayados representan métodos llamados de forma síncrona desde nuestro código, mientras que los que están subrayados representan métodos llamados de forma asíncrona por el sistema.

		Estado entrada							
		Idle	**Initialized**	**Preparing**	**Prepared**	**Started**	**Paused**	**Stopped**	**Playback Completed**
Estado salida	**Initialized**	setDataSource							
	Preparing		prepareAsync					prepareAsync	
	Prepared		prepare	onPrepared	seekTo			prepare	
	Started				start	seekTo start	start		start
	Paused					pause	seekTo pause		
	Stopped				stop	stop	stop	stop	stop
	Playback Completed					onCompletion			seekTo
	End	Release	release	release	release	release	release	release	release
	Error	onError	onError	onError	onError	onError	onError	onError	onError

Transiciones entre estados de la clase `MediaPlayer`

Reproducción de audio con `MediaPlayer`

Si el audio o vídeo lo vamos a reproducir siempre en nuestra aplicación resulta interesante incluirlo en el paquete `.apk` y tratarlo como un recurso.

1. Crea una nueva carpeta con nombre `raw` dentro de la carpeta `res`.
2. Arrastra a su interior el fichero de audio. Por ejemplo *audio.mp3*.
3. Añade las siguientes líneas de código:

```
MediaPlayer mp = MediaPlayer.create(this, R.raw.audio);
mp.start();
```

Si deseamos parar la reproducción tendremos que utilizar el método `stop()` . Si a continuación queremos volver a reproducirlo utilizaremos el método `prepare()` y a continuación `start()`. También podemos usar `pause()` y `start()` para ponerlo en pausa y reanudarlo.

Si en lugar de un recurso preferimos reproducirlo desde un fichero utilizaremos el siguiente código. Observa como en este caso es necesario llamar al método `prepare()`. En el caso anterior no ha sido necesario dado que esta llamada se hace desde `create()`.

```
MediaPlayer mp = new MediaPlayer();
mp.setDataSource(RUTA_DEL_FICHERO);
mp.prepare();
mp.start();
```

Un reproductor *audio* con botones

Al ejercicio anterior le añadimos botones:

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

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Iniciar Audio"
        android:id="@+id/button"
        android:onClick="iniciar"
        android:nestedScrollingEnabled="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Pausar"
        android:id="@+id/button2"
        android:onClick="pausar" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Detener"
        android:id="@+id/button3"
        android:onClick="detener" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Continuar"
        android:id="@+id/button4"
        android:onClick="continuar" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="No reproducir en forma circular"
        android:id="@+id/button5"
        android:onClick="circular" />
</LinearLayout>
```

```
package com.example.amaia.ud03ej02multimedia;

import android.media.MediaPlayer;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
```

```
public class MainActivity extends AppCompatActivity {

    private Button b1;
    private MediaPlayer mp;
    private int posicion = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        b1 = (Button) findViewById(R.id.button5);
    }

    public void destruir() {
        if (mp != null)
            mp.release();
    }

    public void iniciar(View v) {
        destruir();
        mp = MediaPlayer.create(this, R.raw.audio);
        mp.start();
        String op = b1.getText().toString();
        if (op.equals("No reproducir en forma circular"))
            mp.setLooping(false);
        else
            mp.setLooping(true);
    }

    public void pausar(View v) {
        if (mp != null && mp.isPlaying()) {
            posicion = mp.getCurrentPosition();
            mp.pause();
        }
    }

    public void continuar(View v) {
        if (mp != null && mp.isPlaying() == false) {
            mp.seekTo(posicion);
            mp.start();
        }
    }

    public void detener(View v) {
        if (mp != null) {
            mp.stop();
            posicion = 0;
        }
    }

    public void circular(View v) {
        detener(null);
        String op = b1.getText().toString();
        if (op.equals("No reproducir en forma circular"))
            b1.setText("reproducir en forma circular");
        else
            b1.setText("No reproducir en forma circular");
    }
}
```

Crear un reproductor multimedia paso a paso

Profundizando en el objeto MediaPlayer, por medio de un ejemplo, donde trataremos de realizar un reproductor de videos personalizado.

1. En la carpeta `res/drawable`, arrastra cuatro ficheros de iconos, para el play, pause, stop y log, por ejemplo:

 play.png,  stop.png,  pause.png y  log.png.

2. Definimos el fichero con el `layout` de la aplicación.

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

    <LinearLayout
        android:id="@+id/ButonsLayout"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:orientation="horizontal"
        android:layout_alignParentTop="true">

        <ImageButton android:id="@+id/play"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:src="@drawable/play"/>

        <ImageButton android:id="@+id/pause"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:src="@drawable/pause"/>

        <ImageButton android:id="@+id/stop"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:src="@drawable/stop"/>

        <ImageButton android:id="@+id/logButton"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:src="@drawable/log"/>

        <EditText    android:id="@+id/path"
            android:layout_height="match_parent"
            android:layout_width="match_parent"
            android:text="/data/video.3gp"/>
    </LinearLayout>

    <VideoView android:id="@+id/surfaceView"
        android:layout_height="match_parent"
        android:layout_width="match_parent"
        android:layout_below="@+id/ButonsLayout"/>

    <ScrollView android:id="@+id/ScrollView"
        android:layout_height="100px"
        android:layout_width="match_parent"
        android:layout_alignParentBottom="true">
```

```
        <TextView    android:id="@+id/Log"
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:text="Log:"/>
    </ScrollView>
</RelativeLayout>
```

3. Creamos el código de la actividad.

```
package com.example.amaia.ej03_multimedia;

import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnBufferingUpdateListener;
import android.media.MediaPlayer.OnCompletionListener;
import android.media.MediaPlayer.OnPreparedListener;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity implements
      OnBufferingUpdateListener, OnCompletionListener,
        OnPreparedListener, SurfaceHolder.Callback {

    private MediaPlayer mediaPlayer;
    private SurfaceView surfaceView;
    private SurfaceHolder surfaceHolder;
    private EditText editText;
    private ImageButton bPlay, bPause, bStop, bLog;
    private TextView logTextView;
    private boolean pause;
    private String path;
    private int savePos = 0;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);
        surfaceView = (SurfaceView) findViewById(R.id.surfaceView);
        surfaceHolder = surfaceView.getHolder();
        surfaceHolder.addCallback(this);
        surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
        editText = (EditText) findViewById(R.id.path);
        editText.setText(
                "http://personales.gan.upv.es/~jtomas/video.3gp");
        logTextView = (TextView) findViewById(R.id.Log);

        bPlay = (ImageButton) findViewById(R.id.play);
        bPlay.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
                Log.i("aaa","Boton play pulsado");
                if (mediaPlayer != null) {
                    if (pause) {
                        mediaPlayer.start();
                    } else {
                        playVideo();
```

```
                }
            }
        }
    });

    bPause = (ImageButton) findViewById(R.id.pause);
    bPause.setOnClickListener(new View.OnClickListener() {
        public void onClick(View view) {
            Log.i("aaa","Boton PAUSE pulsado");
            if (mediaPlayer != null) {
                pause = true;
                mediaPlayer.pause();
            }
        }
    });

    bStop = (ImageButton) findViewById(R.id.stop);
    bStop.setOnClickListener(new View.OnClickListener() {
        public void onClick(View view) {
            Log.i("aaa","Boton STOP pulsado");
            if (mediaPlayer != null) {
                pause = false;
                mediaPlayer.stop();
            }
        }
    });

    bLog = (ImageButton) findViewById(R.id.logButton);
    bLog.setOnClickListener(new View.OnClickListener() {
        public void onClick(View view) {
            Log.i("aaa","Boton LOG pulsado");
            if (logTextView.getVisibility()==TextView.VISIBLE) {
                logTextView.setVisibility(TextView.INVISIBLE);
            } else {
                logTextView.setVisibility(TextView.VISIBLE);
            }
        }
    });

    log("");
}


private void playVideo() {
    try {
        pause = false;
        path = editText.getText().toString();
        mediaPlayer = new MediaPlayer();
        mediaPlayer.setDataSource(path);
        mediaPlayer.setDisplay(surfaceHolder);
        mediaPlayer.prepare();
        // mMediaPlayer.prepareAsync(); Para streaming
        mediaPlayer.setOnBufferingUpdateListener(this);
        mediaPlayer.setOnCompletionListener(this);
        mediaPlayer.setOnPreparedListener(this);
        mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
        mediaPlayer.seekTo(savePos);
    } catch (Exception e) {
        log("ERROR: " + e.getMessage());
    }
}
```

```
@Override
public void onBufferingUpdate(MediaPlayer arg0, int percent) {
    log("onBufferingUpdate percent:" + percent);
}

@Override
public void onCompletion(MediaPlayer mp) {
    log("onCompletion called");
}

@Override
public void onPrepared(MediaPlayer mediaplayer) {
    log("onPrepared called");
    int mVideoWidth = mediaPlayer.getVideoWidth();
    int mVideoHeight = mediaPlayer.getVideoHeight();
    if (mVideoWidth != 0 && mVideoHeight != 0) {
        surfaceHolder.setFixedSize(mVideoWidth, mVideoHeight);
        mediaPlayer.start();
    }
}

@Override
public void surfaceCreated(SurfaceHolder holder) {
    log("surfaceCreated called");
    playVideo();
}

public void surfaceChanged(SurfaceHolder surfaceholder,
                           int i, int j, int k) {
    log("surfaceChanged called");
}
public void surfaceDestroyed(SurfaceHolder surfaceholder) {
    log("surfaceDestroyed called");
}

@Override
protected void onDestroy() {
    super.onDestroy();
    if (mediaPlayer != null) {
        mediaPlayer.release();
        mediaPlayer = null;
    }
}

@Override
public void onPause() {
    super.onPause();
    if (mediaPlayer != null & !pause) {
        mediaPlayer.pause();
    }
}

@Override
public void onResume() {
    super.onResume();
    if (mediaPlayer != null & !pause) {
        mediaPlayer.start();
    }
}

@Override
protected void onSaveInstanceState(Bundle guardarEstado) {
    super.onSaveInstanceState(guardarEstado);
```

```
            if (mediaPlayer != null) {
                int pos = mediaPlayer.getCurrentPosition();
                guardarEstado.putString("ruta", path);
                guardarEstado.putInt("posicion", pos);
            }
        }

        @Override
        protected void onRestoreInstanceState(Bundle recEstado) {
            super.onRestoreInstanceState(recEstado);
            if (recEstado != null) {
                path = recEstado.getString("ruta");
                savePos = recEstado.getInt("posicion");
            }
        }
        private void log(String s) {
            logTextView.append(s + "\n");
        }

    }
```

4. Añadimos en el *Manifest*, el permiso para acceder a internet.

```
<uses-permission
    android:name="android.permission.INTERNET">
</uses-permission>
```

Como se puede ver la aplicación extiende la clase **`AppCompatActivity`**. Además implementamos cuatro interfaces que corresponden a varios escuchadores de eventos. Luego continúa la declaración de variables. Las primeras corresponden a diferentes elementos de la aplicación y su significado resulta obvio. La variable **`pause`** nos indica si el usuario ha pulsado el botón correspondiente, la variable **`path`** nos indica dónde está el vídeo en reproducción y la variable **`savePos`** almacena la posición de reproducción.

El código continúa con la definición del método **`playVideo().`** Este método se encarga de obtener la ruta de reproducción, crear un nuevo objeto **`MediaPlayer`,** luego se le asigna la ruta y la superficie de visualización, a continuación se prepara la reproducción del vídeo. En caso de querer reproducir un *stream* desde la red, esta función puede tardar bastante tiempo, en tal caso es recomendable utilizar en su lugar el método **`prepareAsync()`** que permite continuar con la ejecución del programa, aunque sin esperar a que el vídeo esté preparado. Las siguientes tres líneas asignan a nuestro objeto varios escuchadores de eventos que serán descritos más adelante. Tras preparar el tipo de audio, se sitúa la posición de reproducción a los milisegundos indicados en la variable **`savePos.`** Si se trata de una nueva reproducción, esta variable será cero.

Los métodos **`onBufferingUpdate`** y **`onCompletion`** implementan los interfaces **`OnBufferingUpdateListener`** y **`OnCompletionListener.`** El primero irá mostrando el porcentaje de obtención de búfer de reproducción, mientras que el segundo será invocado cuando el vídeo en reproducción llegue al final.

El método **`onPrepared`** implementa la interfaz **`OnPreperedListener.`** Es invocado una vez que el vídeo ya está preparado para su reproducción. En este momento podemos conocer el alto y el ancho del vídeo y ponerlo en reproducción.

Los métodos **`surfaceCreated`** y **`SurfaceChanged`** implementan la interfaz **`SurfaceHolder.Callback.`** Se invocarán cuando nuestra superficie de visualización se cree,

cambie o se destruya. Los métodos que siguen corresponden a acciones del ciclo de vida de una actividad:

El método **onDestroy** se invoca cuando la actividad va a ser destruida. Dado que un objeto de la clase **MediaPlayer** consume muchos recursos, resulta interesante liberarlos lo antes posible.

Los métodos **onPause** y **onResume** se invocan cuando la actividad pasa a un segundo plano y cuando vuelve a primer plano. Dado que queremos que el vídeo deje de reproducirse y continúe reproduciéndose en cada uno de estos casos, se invocan a los métodos **pause()** y **start()**, respectivamente. No hay que confundir esta acción con la variable **pause** que lo que indica es que el usuario ha pulsado el botón correspondiente.

Cuando este sistema necesita memoria, puede decidir eliminar nuestra actividad. Antes de hacerlo llamará al método **onSaveInstanceState** para darnos la oportunidad de guardar información sensible. Si más adelante el usuario vuelve a la aplicación, esta se volverá a cargar, invocándose el método **onRestoreInstanceState**, donde podremos restaurar el estado perdido. En nuestro caso la información a guardar son las variables **path** y **savePos,** que representan el vídeo y la posición que estamos reproduciendo.

Ocurre el mismo proceso cuando el usuario cambia la posición del teléfono. Es decir, cuando el teléfono se voltea las actividades son destruidas y vueltas a crear, por lo que también se invocan estos métodos.

El último método **log** es utilizado por varios escuchadores de eventos para mostrar información sobre lo que está pasando. Esta información puede visualizarse o no, utilizando el botón correspondiente.

La clase `SoundPool`

Hemos aprendido a utilizar la clase `MediaPlayer` para reproducir audio y video. En un primer ejemplo vamos a aprender a utilizarla para introducir efectos de audio. Como veremos esta clase no resulta adecuada para este uso. A continuación se introducirá la clase **SoundPool** y se mostrará mediante un ejemplo como esta sí que resulta eficiente para **juegos.**

La clase **SoundPool** maneja y reproduce de forma rápida recursos de audio en las aplicaciones.

Un **SoundPool** es una colección de pistas de audio que se pueden cargar en la memoria desde un recurso (dentro de la APK) o desde el sistema de archivos. `SoundPool` utiliza el servicio de la clase `MediaPlayer` para descodificar el audio en un formato crudo (PCM de16 bits), lo que después permite reproducirlo rápidamente por el hardware sin tener que decodificarlas cada vez.

Los sonidos pueden repetirse en un bucle una cantidad establecida de veces, definiendo el valor al reproducirlo, o dejarse reproduciendo en un bucle infinito con -1. En este caso, será necesario detenerlo con el método **stop()**.

La velocidad de reproducción también se puede cambiar. El rango de velocidad de reproducción va de 0.5 a 2.0. Una tasa de reproducción de 1.0 hace que el sonido se reproduzca en su frecuencia original. Una tasa de reproducción de 2.0 hace que el sonido se reproduzca al doble de su frecuencia original, y una tasa de reproducción de 0.5 hace que se reproduzca a la mitad de su frecuencia original.

Cuando se crea un `SoundPool` hay que establecer en un parámetro del constructor el máximo de pistas que se pueden reproducir simultáneamente. Este parámetro no tiene por qué coincidir con el número de pistas cargadas. Cuando se pone una pista en reproducción (método **`play()`**) hay que indicar una prioridad. Esta prioridad se utiliza para decidir que se hará cuando el número de reproducciones activas exceda el valor máximo establecido en el constructor. En este caso, se detendrá el flujo con la prioridad más baja, y en caso de que haya varios, se detendrá el más antiguo. En caso de que el nuevo flujo sea el de menor prioridad, este no se reproducirá.

Una lisita de todos los métodos de esta clase la puedes encontrar en el siguiente link:

https://developer.android.com/reference/android/media/SoundPool.html

En el siguiente ejemplo veremos cómo se reproduce un sonido con `MediaPlayer`, con Naranja con `SoundPool` y de rojo como se construye el `SoundPool` a partir de la versión 21 ya que está obsoleto el constructor.

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.SoundPool;

import android.view.View;

public class MainActivity extends AppCompatActivity {

    MediaPlayer mpDisparo, mpExplosion;

    SoundPool soundPool;
    int idDisparo, idExplosion;

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

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

        soundPool = new SoundPool(5, AudioManager.STREAM_MUSIC, 0);
        idDisparo = soundPool.load(this, R.raw.disparo, 0);
        idExplosion = soundPool.load(this, R.raw.explosion, 0);


/*        if((android.os.Build.VERSION.SDK_INT) == 21){
            SoundPool.Builder idExplosion = new SoundPool.Builder();
            idExplosion.setMaxStreams(5);
            soundPool = idExplosion.build();
        }
        else{
            soundPool = new SoundPool(5, AudioManager.STREAM MUSIC, 0);
        }
        idExplosion = soundPool.load(this, R.raw.explosion, 0);
*/
    }

    public void disparo(View view) {
        mpDisparo.start();
    }
```

```
    public void explosion(View view) {
        mpExplosion.start();
    }

    public void disparo2(View view) {
        soundPool.play(idDisparo, 1, 1, 1, 0, 1);
    }

    public void explosion2(View view) {
        soundPool.play(idExplosion, 1, 1, 0, 0, 1);
    }

}
```

Grabación de audio: La clase `MediaRecorder`

Las APIs de Android disponen de facilidades para capturar audio y video, permitiendo su codificación en diferentes formatos. La clase `MediaRecorder` nos permitirá de forma sencilla integrar esta funcionalidad en la aplicación.

La mayoría de dispositivos disponen de micrófono para capturar audio, sin embargo esta facilidad no ha sido integrada en el emulador. Por lo tanto, has de probar los ejemplos en un dispositivo real.

La clase **`MediaRecorder`** dispone de una serie de métodos que podrás utilizar para configurar la grabación:

- **`setAudioSource(int audio_source)`** – Dispositivo que se utilizará como fuente del sonido. Normalmente **`MediaRecorder.AudioSource.MIC`**. También se pueden utilizar otras constantes como **`DEFULT`**, **`CAMCORDER`**, **`VOICE_CALL`**, **`VOICE_COMUNICATION`**, **...**
- **`setOutputFile (String fichero)`** – Nombre del fichero de salida.
    - **`setOutputFormat(int output_forma)`** – Formato del fichero de salida. Se pueden utilizar constantes de la clase `MediaRecorder.OutputFormat`: `DEFAULT`, `AMR_NB`, `AMR_WB`, `RAW_AMR (ARM)`, `MPEG_4(MP4)` y `THREE_GPP (3GPP)`.
    - **`setAudioEncoder(int audio_encoder)`** – Codificación del audio. Cuatro posibles constantes de la clase `MediaRecorder.AudioEncoder`: `AAC`, `AMR_NB`, `AMR_WB` y `DEFAULT`.
    - **`setAudioChannels(int numeroCanales)`** – Especificamos el número de canales 1: mono y 2: estéreo.
    - **`setAudioEncodingBitRate (int bitRate)`** (desde nivel de API 8) – Especificamos los bits por segundo (bps) a utilizar en la codificación.
    - **`setAudioSamplingRate (int samplingRate)`** (desde nivel de API 8) – Especificamos el número de muestras por segundo a utilizar en la codificación.
    - **`setMaxDuration (int max_duration_ms)`** (desde nivel de API 3) – Indicamos una duración máxima para la grabación. Tras este tiempo se detendrá.

- **`setMaxFileSize (long max_filesize_bytes)`** (desde nivel de API 3) – Indicamos un tamaño máximo para el fichero. Al alcanzar el tamaño se detendrá.
- **`prepare()`** – Prepara la grabación para la captura de datos. Antes de llamarlo hay que configurar la grabación y después ya podemos invocar al método `start()`.
- **`start()`** – Comienza la captura
- **`stop()`** – Finaliza la captura
- **`reset()`** – Reinicia el objeto como si lo acabáramos de crear. Hay que volver a configurarlo.
- **`release()`** – Libera todos los recursos utilizados de forma inmediata. Si no llamas al método se liberarán cuando el objeto sea destruido.

La clase `MediaRecorder` también dispone de métodos que podrás utilizar para configurar la grabación de **video**. Más información en:

http://developer.android.com/reference/android/media/MediaRecorder.html

La siguiente tabla muestra los diferentes métodos que pueden ser ejecutados en cada estado y el estado que alcanzaremos tras la llamada:

		Estado entrada					
		Initial	**Initialized**	**DataSource Configured**	**Prepared**	**Recording**	**Error**
Estado salida	**Initial**		reset	reset	reset	reset stop	reset
	Initialized	setAudioSource setVideoSource	setAudioSource setVideoSource				
	DataSource Configured		setOutputFormat	setAudioEncoder setVideoEncoder setOutputFile setVideoSize setVideoFrameRate setPreviewDisplay			
	Prepared			prepare			
	Recording				start		
	Released	releas					
	Error	llamada incorrecta	llamada incorrecta	llamada incorrecta	llamada incorrecta	llamada incorrecta	

Transiciones entre estados de la clase MediaRecorder

Ejemplo Grabador de sonidos:

1.- Añadimos los permisos al Manifest

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

¡¡¡IMPORTANTE!!!: Además hay que añadir y/o controlar los permisos desde el API 23 en los ajustes

2.- Creamos el Layout:

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

```
        android:orientation="horizontal">

        <Button
            android:id="@+id/bGrabar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Grabar"
            android:onClick="grabar"/>

        <Button
            android:id="@+id/bDetener"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Detener Grabacion"
            android:onClick="detenerGrabacion"/>

        <Button
            android:id="@+id/bReproducir"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Reproducir"
            android:onClick="reproducir"/>
    </LinearLayout>
```

3.- La programación quedaría así:

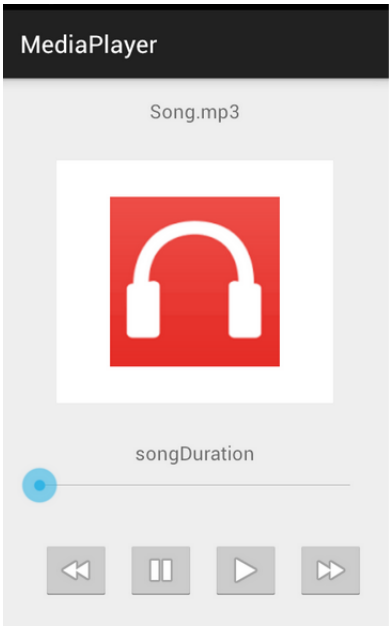
```
    package com.example.amaia.ej05_multimedia;

    import android.media.MediaPlayer;
    import android.media.MediaRecorder;
    import android.os.Environment;
    import android.support.v7.app.AppCompatActivity;
    import android.os.Bundle;
    import android.util.Log;
    import android.view.View;
    import android.widget.Button;

    import java.io.IOException;

    public class MainActivity extends AppCompatActivity {

        private Button bGrabar, bDetener, bReproducir;

        private static final String LOG_TAG = "Grabadora";
        private MediaRecorder mediaRecorder;
        private MediaPlayer mediaPlayer;

        private static String fichero = Environment.
                getExternalStorageDirectory().getAbsolutePath()+"/audio.3gp";


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

            bGrabar = (Button) findViewById(R.id.bGrabar);
            bDetener = (Button) findViewById(R.id.bDetener);
            bReproducir = (Button) findViewById(R.id.bReproducir);
            bDetener.setEnabled(false);
            bReproducir.setEnabled(false);
        }
```

```
    public void grabar(View view) {
        bDetener.setEnabled(false);
        bReproducir.setEnabled(false);

        mediaRecorder = new MediaRecorder();
        mediaRecorder.setOutputFile(fichero);
        mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        mediaRecorder.setOutputFormat(MediaRecorder.
                OutputFormat.THREE_GPP);
        mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.
                AMR_NB);

        try {
            mediaRecorder.prepare();

        } catch (IOException e) {
            Log.e(LOG_TAG, "Fallo en grabación");
        }
        mediaRecorder.start();
    }

    public void detenerGrabacion(View view) {
        bGrabar.setEnabled(true);
        bDetener.setEnabled(false);
        bReproducir.setEnabled(true);

        mediaRecorder.stop();
        mediaRecorder.release();
    }

    public void reproducir(View view) {

        mediaPlayer = new MediaPlayer();

        try {
            mediaPlayer.setDataSource(fichero);
            mediaPlayer.prepare();
            mediaPlayer.start();
        } catch (IOException e) {
            Log.e(LOG_TAG, "Fallo en reproducción");
        }

    }

}
```

Comenzamos declarando dos objetos de las clases **`MediaRecorder y MediaPlayer`** que serán usados para la grabación y reproducción respectivamente. También se declaran dos **`String`:** La constante **`LOG_TAG`** será utilizada como etiqueta para identificar nuestra aplicación en el **fichero** de Log. La variable fichero identifica en nombre del fichero donde se guardará la grabación. Este fichero se almacenará en una carpeta especialmente creada para nuestra aplicación. En el método **`onCreate()`** simplemente habilitamos el botón de grabar.

Actividad Multimedia

Realizar el siguiente reproductor de audio:



Tienes el código en siguiente enlace:

http://mobilemerit.com/how-to-use-media-player-in-android-studio/