Multimedia y ciclo de vida de una aplicación (II)

Multimedia

Mª Alejandra Tomás Fernández

Multimedia y Streaming en Android (I)

- Android **provee clases de soporte para codificar y decodificar** una gran variedad de formatos multimedia comunes.

- Esas clases nos **van a permitir**:
  - **Reproducir y grabar audio**
  - **Reproducir y grabar video**
  - **Capturar y visualizar imágenes fijas**
  - **Imprimir imágenes y documentos varios.**

- Las **principales clases** son:
  - **MediaPlayer**
  - **MediaRecorder**
  - **AudioManager**

Multimedia y Streaming en Android (II)

- 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 varios posibles protocolos:**
    - ***RTSP (con sus protocolos RDP y SDP)***
    - ***HTTP/HTTPS progressive streaming (HTTPS desde versión 3.0)***
    - ***HTTP/HTTPS live streaming (desde versión 3.0)***

Clases en Android más usadas en Multimedia (I)

- **AUDIO**:
  - →**MediaPlayer**: Reproducción de audio/vídeo desde ficheros o *streams*.
  - →**MediaRecorder**: Permite grabar audio y vídeo.
  - →**MediaController**: Visualiza controles estándar para MediaPlayer (pausa, stop...).
  - →**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 (***AudioTrack***)
  - →**JetPlayer**: Reproduce audio y video interactivo creado con JetCreator. **Se usa en videojuegos y son sonidos MIDI.**

Clases en Android más usadas en Multimedia (II)

- **RingtoneManager y Ringtone**: Gestionan sonidos que llegan cuando nos llaman, cuando llega una notificación y cuando se apaga una alarma.

Clases en Android más usadas en Multimedia (III)

- **VIDEO**:
  - →**MediaPlayer**: Reproducción de audio/vídeo desde ficheros o *streams*.
  - →**MediaRecorder**: Permite grabar audio y vídeo.
  - →**MediaController**: Visualiza controles estándar para MediaPlayer (pausa, stop...).
  - →**JetPlayer**: Reproduce audio y video interactivo creado con JetCreator. **Se usa en videojuegos y son sonidos MIDI.**
  - →**VideoView**: Vista que permite la reproducción de vídeo.
  - →**Camera**: Cómo utilizar la cámara para tomar fotos y video.

Clases en Android más usadas en Multimedia (IV)

- **IMÁGENES FIJAS**:
  - →**Camera**: Cómo utilizar la cámara para tomar fotos y video.
  - →**FaceDetector**: Identifica la cara de personas desde imágenes

Clases en Android más usadas en Multimedia (V)

- **IMPRIMIR IMÁGENES Y DOCUMENTOS**:

→PrintHelper
→PrintManager
→WebView
→PrintJob
→PrintDocumentAdapter

Formatos de AUDIO, VIDEO e IMAGEN soportados

- Android soporta varios formatos de multimedia.
- Los tenéis en el siguiente enlace:

FORMATOS MULTIMEDIA SOPORTADOS

Audio Focus

- Cuando se diseñan aplicaciones que incorporan reproducción de sonido, es **MUY IMPORTANTE**, que **pidan "el foco de audio (AUDIO FOCUS)".**
- En Android podría darse el caso de **varias aplicaciones sonando a la vez.**
- Para evitar que eso suceda, existe el concepto de **AUDIO FOCUS**.
- De esta forma, cuando se diseña una aplicación con audio, lo primero a hacer siempre es:
  - **Pedir el foco de audio (Audio Focus).**
  - Si nos lo conceden, **registramos el escuchador que controla los botones hardware multimedia del dispositivo** y lanzamos el play.

Permisos en el Manifest

- Para usar multimedia se necesitan los siguientes **permisos**:

  → **Permiso de Internet**: se necesitará sólo si se manejan archivos desde stream en la red:

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

  → **Wake Lock Permission:** es aconsejable para evitar que la pantalla no se oscurezca (se vaya apagando) o el procesador se vaya a dormir, usar los métodos MediaPlayer.setScreenOnWhilePlaying() or MediaPlayer.setWakeMode() .

  Se puede mantener despierta **sólo la CPU** (**wakelock parcial**) o **CPU y pantalla** (**wakelock total**).

  Estos métodos requieren el permiso:

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

AUDIO

Mª Alejandra Tomás Fernández

Identificar qué Audio Stream usar

- **Este es el primer paso que hay que hacer para crear una app que incorpore multimedia.**
- Android mantiene un "**Audio Stream o canal de audio**" **separado** para:
  - → **Reproducir música**
  - → **Reproducir alarmas**
  - → **Notificaciones**
  - → **Tono de las llamadas entrantes**
  - → **Sonidos del sistema**
  - → **Volumen de la llamada (durante).**
  - → **Y tonos DTMF (doble tono multifrecuencia):** son las señales que enviamos cuando pulsamos las teclas del teléfono. Por ejemplo cuando llamamos a nuestra compañia para informar de avería y nos piden teclear nuestro nº, ellos recogen esas señales.
- Tener los **canales separados**, **permite a los usuario, controlar el volumen de cada canal independientemente**.
- Muchos de esos **canales son restringidos a eventos del sistema**.
- A menos que nuestra app sea una nueva alarma de reloj o similar, usaremos casi siempre el canal de audio de música, que se identifica como AudioManager.STREAM_MUSIC

Clases usadas en AUDIO: AudioManager (I)

- Gestiona capacidades básicas de audio como:
  - Manipular el volumen del dispositivo
  - Reproducir efectos de sonido del sistema
  - Cambiar el modo de timbre del dispositivo
  - etc

- Las aplicaciones adquieren una instancia de AudioManager llamando al método:

  Context.getSystemService (Context.AUDIO_SERVICE)

- Una vez hecho esto, la aplicación puede:
  - cargar y luego reproducir efectos de sonido.
  - Ajustar el volumen del dispositivo
  - Controlar el hardware del dispositivo (*periféricos como cascos con cable o bluetooth,…)*

    Por ejemplo: poner en silencio el micrófono, encender los auriculares Bluetooth.

Clases usadas en AUDIO: AudioManager (II)

- **Principales métodos de AudioManager:**
  - **getStreamVolume (int streamType):** obtiene el volumen definido por el usuario para el tipo de canal indicado como parámetro.
  - **getStreamMaxVolume (int streamType):** obtiene el volumen máximo que se puede definir para el tipo de canal indicado como parámetro.
  - **isMusicActive ():** indica si se está reproduciendo música
  - **getRingerMode():** devuelve el modo de sonidos del dispositivo; puede tomar las constantes *RINGER_MODE_NORMAL, RINGER_MODE_SILENT, RINGER_MODE_VIBRATE.*
  - *requestAudioFocus (*
    AudioManager.OnAudioFocusChangeListener i,
    int streamType,
    int durationHint): envía una petición para obtener el foco del canal de audio. El parámetro "**durationHint**" indica cómo queremos ganar el foco: temporalmente (por ej. para notificaciones,…), permanentemente,…
  - *abandonAudioFocus (AudioManager.OnAudioFocusChangeListener i,)*

Clases usadas en AUDIO: AudioManager (II)

- Existen **métodos adicionales** que permiten *conocer si el audio se reproduce a través de un dispositivo Bluetooth*, *ajustar el volumen de un tipo de audio,…*

- Para **establecer el volumen del audio que vamos a reproducir en esa actividad y controlarlo** con los **BOTONES DEL DISPOSITIVO,** debemos usar el **método**

  setVolumeControlStream (int streamType)

  en el onCreate() de la actividad. A partir de esta llamada, el usuario podrá usar los botones del dispositivo para subir y bajar el volumen.

Clases usadas en AUDIO: SoundPool (I)

- Representa una colección de muestras o pistas de audio o streams que se pueden cargar en memoria desde un recurso (*dentro de la APK*) o desde el sistema de archivos.
- Permite reproducir sonidos de forma rápida y simultánea.
- Puede mezclar juntas y reproducir simultáneamente múltiples muestras.
- Recomendable para reproducir pequeños archivos de audio que no deben exceder de 1 MB de tamaño.
- Así, se suele usar en los juegos para reproducir los efectos de sonido: *disparos, explosiones,…*
- SoundPool utiliza el servicio de la clase MediaPlayer para decodificar el audio en un formato crudo (PCM de16 bits), lo que después permite reproducirlo rápidamente por el hardware sin tener que decodificarlas cada vez.
  - De ahí su velocidad, no los decodifica cada vez que los queramos reproducir, sólo cuando los carga.

Clases usadas en AUDIO: SoundPool (II)

- Es **posible repetir los sonidos en un bucle** tantas veces como se desee, **indicando un valor de repetición al reproducirlo**.
- También se puede **mantener reproduciéndose en un bucle infinito** con **el valor -1.**
  - En este último caso, **será necesario detenerlo con el stop().**
- También **podemos establecer la velocidad de reproducción** del sonido, cuyo **rango** puede estar **entre 0.5 y 2.0.**
  - Una **velocidad de reproducción de 1.0** indica que el sonido se reproduce a su **frecuencia original**.
  - Una **velocidad de reproducción de 2.0** indica que el sonido se reproduce **al doble de su frecuencia original.**
  - Una **velocidad de reproducción de 0.5** indica que el sonido se reproducirá lentamente, a **la mitad de su frecuencia original**.

Clases usadas en AUDIO: SoundPool (III)

- **Cuando se crea un objeto SoundPool, hay que establecer mediante un parámetro el nº máximo de sonidos que se pueden reproducir simultáneamente.**
  - Este parámetro **no tiene por qué coincidir con el nº de sonidos cargados**
- Cuando se **reproduce un sonido con su método play(), hay que indicar su prioridad:**
  - Este parámetro **permite que el sistema detenga el flujo con prioridad más baja, cuando el nº de reproducciones simultáneas supere el máximo** indicado en el **constructor**.
  - **Si todos tienen la misma prioridad**, se parará el **más antiguo**.
  - **Si el nuevo flujo es de menor prioridad, éste no se reproducirá**.
- SoundPool realiza la carga de los archivos multimedia de forma **asíncrona,** es decir, el **SO lanzará el sonido con** el **Listener OnLoadCompleteListener cuando se haya completado la carga de cada uno de los archivos.**

Clases usadas en AUDIO: SoundPool (IV)

¿Cómo se usa SoundPool?

1. **Se crea un objeto SoundPool:**

```
SoundPool soundPool =new SoundPool (5, AudioManager.STREAM_MUSIC, 0)
```

- Máx nº repeticiones simultáneas
- Tipo de canal de audio
- Calidad de reproducción. No se usa. Poner 0.

A partir de la API 21 ese constructor fue "deprecated". Aconsejan usar el **SoundPool.Builder**. El equivalente a lo anterior sería:

Clases usadas en AUDIO: SoundPool (V)

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {

    AudioAttributes audioAttrib = new AudioAttributes.Builder()
            .setUsage(AudioAttributes.USAGE_GAME)
            .setContentType(AudioAttributes.CONTENT_TYPE_SONIFICATION)
            .build();
    SoundPool mSound = new SoundPool.Builder()
                    .setAudioAttributes(audioAttrib)
                    .setMaxStreams(6)
                    .build();
}
else {

    mSound = new SoundPool(6, AudioManager.STREAM_MUSIC, 0);
}
```

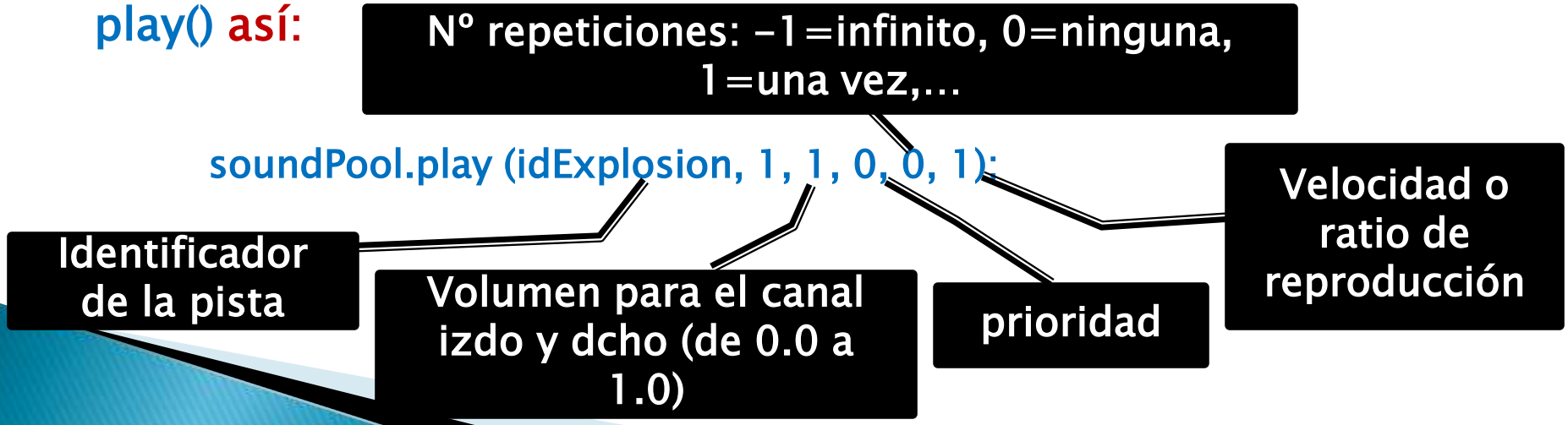
Clases usadas en AUDIO: SoundPool (VI)

2. **Se cargan las pistas o streams mediante el método load().**

Hay varios constructores. **Por ejemplo:**

```
idDisparo = soundPool.load (context, R.raw.disparo, 0);
idExplosion= soundPool.load (context, R.raw.explosion, 0);
```

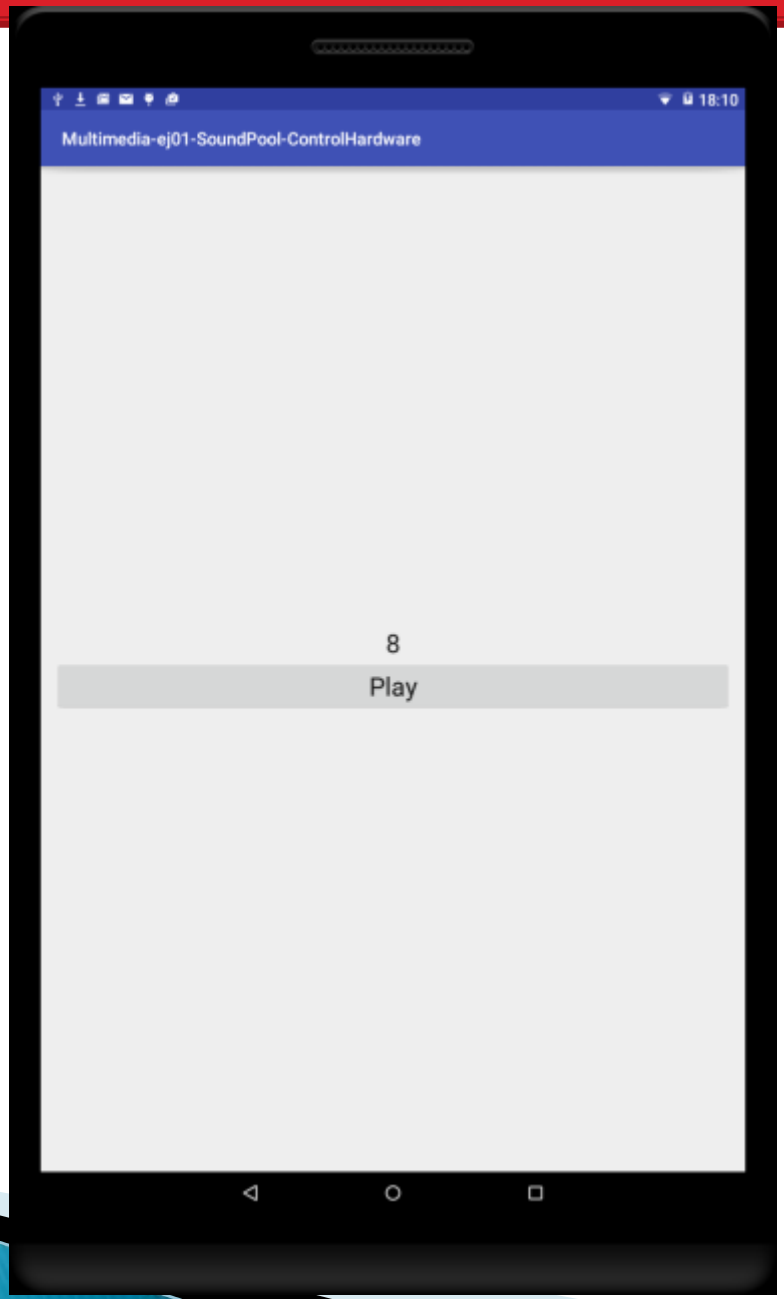
3. **Cuando queramos que suene el sonido, llamaremos al método play() así:**



EJEMPLO DE USO DE AudioManager y SoundPool

- Presenta un **TextView** que muestra el **volumen actual**.
- Presenta un **botón PLAY** que cuando **se pulsa reproduce un sonido llamado "audio.mp3" repetido 3 veces** seguidas y con el volumen actual.
- Si se pulsan los **botones físicos de subir y/o bajar el volumen del dispositivo móvil (tablet o teléfono)**, el sonido se **adecúa al nuevo volumen, así como el TextView que mostrará el volumen actualizado**.

EJEMPLO DE USO DE AudioManager y SoundPool



Mª Alejandra Tomás Fernández

EJEMPLO DE USO DE AudioManager y SoundPool

AndroidManifest.xml

```
<manifest ...>

    <application ...>
        <activity android:name=".MainActivity">
            ...
        </activity>
        <receiver android:name=".RemoteControlReceiver">
            <intent-filter>
                <action android:name="android.media.VOLUME_CHANGED_ACTION" />

            </intent-filter>
        </receiver>
    </application>

</manifest>
```

Registrando el Broadcast Receiver

EJEMPLO DE USO DE AudioManager y SoundPool

RemoteControlReceiver.java

Método que se recibe de un BroadCast Receiver

```
/**CLASE PARA EL BROADCAST RECEIVER   */
public class RemoteControlReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        if ("android.media.VOLUME_CHANGED_ACTION".equals(intent.getAction())) {
            int volActual=
                    (Integer)intent.getExtras()
                            .get("android.media.EXTRA_VOLUME_STREAM_VALUE");
            int volAnterior=
                    (Integer)intent.getExtras()
                            .get("android.media.EXTRA_PREV_VOLUME_STREAM_VALUE");

            if (volActual>volAnterior) {
                // subir volumen
                Toast.makeText(context, "SUBIENDO VOLUMEN...",
                                                    Toast.LENGTH_LONG).show();
                MainActivity.tvVolumen.setText(String.valueOf(volActual));

            }
```

EJEMPLO DE USO DE AudioManager y SoundPool

RemoteControlReceiver.java

```
else if (volActual<volAnterior){
                // bajar volumen
                Toast.makeText(context, "BAJANDO VOLUMEN...",
                                                        Toast.LENGTH_LONG).show();
                MainActivity.tvVolumen.setText(String.valueOf(volActual));
            }
        }
    }
}
```

EJEMPLO DE USO DE AudioManager y SoundPool

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    protected static AudioManager am;
    protected static TextView tvVolumen;
    private Button btPlay;
    private SoundPool mSound;
    private int idSonido;
    private boolean canPlaySound=false; //me dice si tengo el auto focus
    private int volumenActual,volumenMaximo;

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

        //obtengo el AudioManager
        am = (AudioManager) getSystemService(Context.AUDIO_SERVICE);

        //aviso que voy a controlar el volumen de la música
        setVolumeControlStream(AudioManager.STREAM_MUSIC);
```

EJEMPLO DE USO DE AudioManager y SoundPool

MainActivity.java

```
//obtengo referencias objetos
tvVolumen = (TextView) findViewById(R.id.tvVolumen);
btPlay = (Button) findViewById(R.id.btPlay);

//averiguo volumen actual y máximo
volumenActual = am.getStreamVolume(AudioManager.STREAM_MUSIC);
volumenMaximo=am.getStreamMaxVolume(AudioManager.STREAM_MUSIC);

//muestro el volumen actual
tvVolumen.setText(String.valueOf(volumenActual));

// Desactivo el botón Play para que el usuario no pueda pulsar en él
// antes de que los sonidos estén cargados.
btPlay.setEnabled(false);

//creo la instancia del SoundPool
crearInstanciaSoundPool();

//cargo los diferentes sonidos. En este caso sólo hay uno
idSonido=mSound.load(this,R.raw.audio,1);
```

EJEMPLO DE USO DE AudioManager y SoundPool

MainActivity.java

```
// Asigno el listener OnLoadCompleteListener al SoundPool
//para saber cuándo están cargados los sonidos
  mSound.setOnLoadCompleteListener(new SoundPool.OnLoadCompleteListener()
   {
      @Override
      public void onLoadComplete(SoundPool soundPool, int sampleId,
                                 int status) {
         // Si el sonido ha sido cargado ya
         if (status==0) {
            //activo el botón play
            btPlay.setEnabled(true);
         } else {
            Log.i("MIAPLI", "imposible cargar el sonido");
            finish();
         }
      }
   });
```

EJEMPLO DE USO DE AudioManager y SoundPool

MainActivity.java

```
//reproduzco el sonido usando un SoundPool al hacer click
    btPlay.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(canPlaySound){
                //puedo reproducir sonido
                am.playSoundEffect(AudioManager.FX_KEY_CLICK, volumenActual);
                mSound.play(idSonido,
                        (float)volumenActual/volumenMaximo,
                        (float)volumenActual/volumenMaximo,
                        1,3,1);//prioridad 1, repito 3 veces y velocidad 1
            }
        }
        });
```

EJEMPLO DE USO DE AudioManager y SoundPool

MainActivity.java

```
    //PIDO EL AUDIO FOCUS O FOCO DEL CANAL DE AUDIO
    int result = am.requestAudioFocus(afChangeListener,
                     // Use the music stream.
                      AudioManager.STREAM_MUSIC,
                    // Request transient focus (temporal).
                    AudioManager.AUDIOFOCUS_GAIN_TRANSIENT);

    // Start playback.
    if (result == AudioManager.AUDIOFOCUS_REQUEST_GRANTED) {
        //registro el oyente de cambios en los  botones multimedia de volumen
        am.registerMediaButtonEventReceiver (new
                        ComponentName ( getPackageName(),
                                                RemoteControlReceiver.class.getName()));
        //puedo reproducir sonidos
        canPlaySound=true;
    }
}
```

EJEMPLO DE USO DE AudioManager y SoundPool

MainActivity.java

```
/** Método que crea una instancia de SoundPool adaptada a la versión de SO que
 * tengamos
 *
 */
private void crearInstanciaSoundPool() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {

        AudioAttributes audioAttrib = new AudioAttributes.Builder()
                .setUsage(AudioAttributes.USAGE_GAME)
                .setContentType(AudioAttributes.CONTENT_TYPE_SONIFICATION)
                .build();
        mSound = new SoundPool.Builder()
                .setAudioAttributes(audioAttrib)
                .setMaxStreams(6)
                .build();
    } else {

        mSound = new SoundPool(6, AudioManager.STREAM_MUSIC, 0);
    }
}
```

EJEMPLO DE USO DE AudioManager y SoundPool

MainActivity.java

```
// Preparo para reproducir los efectos de sonido
    @Override
    protected void onResume() {
        super.onResume();
        am.loadSoundEffects();
        if (mSound !=null) {
                    //cargo los sonidos
                    idSonido=mSound.load(this,R.raw.audio,1);
          }else{
                    crearInstanciaSoundPool();
                    //cargo los sonidos
                   idSonido=mSound.load(this,R.raw.audio,1);
        }
    }
```

EJEMPLO DE USO DE AudioManager y SoundPool

MainActivity.java

```
//libero recursos
    @Override
    protected void onPause() {
        super.onPause();
        if (mSound !=null) {
                    //descargo de memoria el sonido
                    mSound.unload(idSonido);
                    //libero el SoundPool
                    mSound.release();
                  mSound = null;
        }
        //descargo los efectos de sonido de las teclas de memoria
        am.unloadSoundEffects();


    }
```

EJEMPLO DE USO DE AudioManager y SoundPool

MainActivity.java

```
// Oyente que escucha cambios en la adquisición del Audio Focus
    OnAudioFocusChangeListener afChangeListener =
            new OnAudioFocusChangeListener() {
                public void onAudioFocusChange(int focusChange) {
                    if (focusChange == AudioManager.AUDIOFOCUS_LOSS_TRANSIENT) {
                        // Pause playback
                        canPlaySound=false;
                    } else if (focusChange == AudioManager.AUDIOFOCUS_GAIN) {
                        // Resume playback
                        canPlaySound=true;
                    } else if (focusChange == AudioManager.AUDIOFOCUS_LOSS) {
                        //desregistro el oyente de cambios en botones multimedia
                        am.unregisterMediaButtonEventReceiver(new
ComponentName(getPackageName(), RemoteControlReceiver.class.getName()));
                        //desregistro el oyente de cambios en el audio focus
                        am.abandonAudioFocus(this);
                        // Stop playback
                        canPlaySound=false;
                    }
                }
            };
}
```