# Grabación de audio/vídeo y gráficos avanzados en Android - Ejercicios

## Índice

# 1. Síntesis de voz con Text to Speech

En este primer ejercicio vamos a utilizar el motor *Text to Speech* para crear una aplicación que lea el texto contenido en un `EditText` de la actividad principal. Para ello el primer paso será descargar de las plantillas la aplicación *SintesisVoz*. La aplicación contiene una única actividad. La idea es que al pulsar el botón *Leer* se lea el texto en el cuadro de edición. Existen dos botones de radio para escoger la pronunciación (inglés o español).

Deberemos seguir los siguientes pasos:

- Inserta el código necesario en el método `initTextToSpeech` para que se lance un `Intent` implícito para comprobar si el motor *Text to Speech* está instalado en el sistema:

```
Intent intent = new Intent(Engine.ACTION_CHECK_TTS_DATA);
startActivityForResult(intent, TTS_DATA_CHECK);
```

- En el manejador `onActivityResult` incorporamos el código necesario para inicializar el motor *Text to Speech* en el caso en el que esté instalado, o para instalarlo en el caso en el que no lo estuviera.

```
if (requestCode == TTS_DATA_CHECK) {
        if (resultCode == Engine.CHECK_VOICE_DATA_PASS) {
                tts = new TextToSpeech(this, new OnInitListener() {
                    public void onInit(int status) {
                            if (status == TextToSpeech.SUCCESS) {
                                    ttsIsInit = true;
                                    Locale loc = new Locale("es","","");
                                    if (tts.isLanguageAvailable(loc)
                                                                    >=
TextToSpeech.LANG_AVAILABLE)
                                            tts.setLanguage(loc);
                                    tts.setPitch(0.8f);
                                    tts.setSpeechRate(1.1f);
                            }
                    }
            });
    } else {
            Intent installVoice = new
Intent(Engine.ACTION_INSTALL_TTS_DATA);
            startActivity(installVoice);
    }
}
```

**Nota:**

En el código anterior `tts` es un objeto de la clase `TextToSpeech` que ya está definido en la plantilla. La variable booleana `ttsIsInit` tendrá valor `true` en el caso en el que el motor de síntesis de voz se haya inicializado correctamente. La utilizaremos más adelante para comprobar si se puede leer o no un texto. Mediante el objeto `loc` inicializamos el idioma a español, ya que es el botón de radio seleccionado por defecto al iniciar la actividad.

- Añade el código necesario en el método `onDestroy` para liberar los recursos

asociados a la instancia de *Text to Speech* cuando la actividad vaya a ser destruida:

```
if (tts != null) {
        tts.stop();
    tts.shutdown();
}
```

- El manejador del click del botón *Leer* simplemente llama al método `speak`, que será el encargado de utilizar el objeto `TextToSpeech` para leer el texto en la vista `EditText`. Introduce el código necesario para hacer esto; no olvides de comprobar si el motor *Text to Speech* está inicializado por medio de la variable booleana `ttsIsInit`.
- Por último añade el código necesario a los manejadores del click de los botones de radio para que se cambie el idioma a español o inglés según corresponda. Observa cómo se usa la clase `Locale` en `onActivityResult` para hacer exactamente lo mismo.

## 2. Gráficos 3D

En las plantillas de la sesión tenemos una aplicación `Graficos` en la que podemos ver un ejemplo completo de cómo utilizar `SurfaceView` tanto para gráficos 2D con el `Canvas` como para gráficos 3D con OpenGL, y también de cómo utilizar `GLSurfaceView`.

*a)* Si ejecutamos la aplicación veremos un triángulo rotando alrededor del eje Y. Observar el código fuente, y modificarlo para que el triángulo rote alrededor del eje X, en lugar de Y.

*b)* También podemos ver que hemos creado, además de la clase `Triangulo3D`, la clase `Cubo3D`. Modificar el código para que en lugar de mostrar el triángulo se muestre el cubo.

## 3. Grabación de vídeo con MediaRecorder (*)

En este ejercicio optativo utilizaremos la aplicación *Video* que se te proporciona en las plantillas para crear una aplicación que permita guardar vídeo, mostrándolo en pantalla mientras éste se graba. La interfaz de la actividad principal tiene dos botones, *Grabar* y *Parar*, y una vista `SurfaceView` sobre la que se previsualizará el vídeo siendo grabado.

Debes seguir los siguientes pasos:

- Añade los permisos necesarios en el *Manifest* de la aplicación para poder grabar audio y vídeo y para poder guardar el resultado en la tarjeta SD (recuerda que el siguiente código debe aparecer antes del elemento `application`):

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

- Añade un atributo a la clase `VideoActivity`:

```
MediaRecorder mediaRecorder;
```

- Inicializa el objeto `MediaRecorder` en el método `onCreate`:

```
mediaRecorder = new MediaRecorder();
```

- Para poder previsualizar el vídeo en el `SurfaceView` hemos de obtener su *holder*. Como esta operación es asíncrona, debemos añadir los manejadores adecuados, de tal forma que sólo se pueda reproducir la previsualización cuando todo esté listo. El primer paso consiste en hacer que la clase `VideoActivity` implemente la interfaz `SurfaceHolder.Callback`. Para implementar esta interfaz deberás añadir los siguiente métodos a la clase:

```
public void surfaceCreated(SurfaceHolder holder) {
        // TODO: asociar la superficie al MediaRecorder
}

public void surfaceDestroyed(SurfaceHolder holder) {
        // TODO: liberar los recursos
}
```

- Añadimos en `onCreate` el código necesario para obtener el *holder* de la superficie y asociarle como manejador la propia clase `VideoActivity`:

```
m_holder = superficie.getHolder();
m_holder.addCallback(this);
m_holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
```

- Gracias al método `surfaceCreated` podremos asociar el objeto `MediaRecorder` al *holder* del `SurfaceView`. Dentro de esta misma función le daremos al atributo booleano `preparado` el valor true, lo cual nos permitirá saber que ya podemos iniciar la reproducción:

```
mediaRecorder.setPreviewDisplay(holder.getSurface());
preparado = true;
```

- En el método `surfaceDestroyed` simplemente invocaremos el método `release` del objeto `MediaRecorder`, para liberar los recursos del objeto al finalizar la actividad.
- Se ha añadido un método `configurar` a la clase `VideoActivity` que se utilizará para indicar la fuente de audio y vídeo, el nombre del fichero donde guardaremos el vídeo grabado, y algunos parámetros más. En esa función debes añadir el siguiente código. Fíjate cómo se ha incluido una llamada a `prepare` al final:

```
if (mediaRecorder != null) {
        try {

                // Inicializando el objeto MediaRecorder
mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
mediaRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);

mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);

mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);
mediaRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.DEFAULT);
```

```
                mediaRecorder.setOutputFile("/mnt/sdcard/DCIM/video.3gp");

                mediaRecorder.prepare();

        } catch (IllegalArgumentException e) {
                Log.d("MEDIA_PLAYER", e.getMessage());
        } catch (IllegalStateException e) {
                Log.d("MEDIA_PLAYER", e.getMessage());
        } catch (IOException e) {
                Log.d("MEDIA_PLAYER", e.getMessage());
        }
}
```

- Sólo queda introducir el código necesario para iniciar y detener la reproducción. En el manejador del botón *Grabar* invocaremos al método `start` del objeto `MediaRecorder`, sin olvidar realizar una llamada previa al método `configurar`.
- En el manejador del botón *Parar* invocamos en primer lugar el método `stop` y en segundo lugar el método `reset` del objeto `MediaRecorder`. Con esto podríamos volver a utilizar este objeto llamando a `configurar` y a `start`.

**Aviso:**

A la hora de redactar estos ejercicios existía un bug que impedía volver a utilizar un objeto `MediaRecorder` tras haber usado `reset`. Puede que sea necesario que tras hacer un `reset` debas invocar el método `release` y crear una nueva instancia del objeto `MediaRecorder` con el operador `new`.