Susana Bardón Restrepo
Rocio Cruz Santana
Christian Fernández Alegre

Documentación Multimedia

Contenido

1.- Requisitos:

Para la tablet Nexus 7, tendremos que descargar previamente las siguientes aplicaciones:

- Grabador de sonidos.
- Cámara.
- Reproductor de música.

2.- Menú Principal

Partimos que lo primero que se visualizará en nuestro proyecto es un ListView, que como todos sabemos no es más que una lista de elementos desplazables, que en nuestro caso, contendrá los puntos que hemos expuestos anteriormente.

Para la realización de un listView tendremos que seguir estos pasos:

- Creamos un nuevo proyecto
- Eliminar el TextView de “hola mundo” del layout y en la opción Composite arrastrar la opción de listView.
- Crear un nuevo layout en clic derecho new /Other/Android xml File y se pone el nombre del nuevo layout (nunca puede ser con mayúsculas)
- En este layout pondremos el diseño de nuestro listView en nuestro caso sería ImageView (que se encuentra en (Imágenes&Media) y un textView (Form Witgest)

Esto es para la parte visual o gráfica. Para poder ver las opciones que deseamos e interactuar con ellas, necesitamos el código para el que seguiremos los siguientes procedimientos:

- Ir a src
- Crear una nueva clase. Damos click derecho new /Class y se le pone el nombre deseado en nuestro caso Opción esta clase se realizará para crear las opciones que tendrá nuestro listView su código sería el siguiente:

```
3  public class Opcion {
4      private String nombre;
5      private int numero;
6
7      // contructor implicito de java
8      public Opcion() {
9
10     }
11
12     public Opcion(String n, int nu) {
13         if (n != null) {
14             this.nombre = n;
15         }
16         this.numero = nu;
17     }
18
19     public String getNombre() {
20         return nombre;
21     }
22
23     public void setNombre(String nombre) {
24         this.nombre = nombre;
25     }
26
27     public int getNumero() {
28         return numero;
29     }
30
31     public void setNumero(int numero) {
32         this.numero = numero;
33     }
34 }
```

Otro paso importante para inflar el listView sería la creación de la clase MyAdapterOpciones extends ArrayAdapter<Opcion> esta clase la creamos igual que la anterior su código es el siguiente:

```
public class MyAdapterProyect extends ArrayAdapter<Opcion> {

    private Activity context;
    private Opcion[] datos;
    private int[] imagenes = { R.drawable.music2, R.drawable.record_move,
            R.drawable.image, R.drawable.record_sound, R.drawable.video,
            R.drawable.camera3 };

    public MyAdapterProyect(Activity context, int textViewResouseId, Opcion[] n) {
        super(context, textViewResouseId, n);
        this.context = context;
        this.datos = n;
    }

    static class ViewHolder {
        protected TextView nombre;
        protected ImageView ima;
    }

public View getView(int position, View convertView, ViewGroup parent) {

     View iten = convertView;

     ViewHolder holder;// holder es titular

     if (convertView == null) {
         // intance element inflater

         LayoutInflater inflater = context.getLayoutInflater();
         iten = inflater.inflate(R.layout.segundo, null);

         holder = new ViewHolder();

         holder.nombre = (TextView) iten.findViewById(R.id.textView1);
         holder.ima = (ImageView) iten.findViewById(R.id.imageView1);

         iten.setTag(holder);

     } else {
         holder = (ViewHolder) iten.getTag();
     }

     holder.nombre.setText(datos[position].getNombre());
     holder.ima.setImageResource(imagenes[datos[position].getNumero()]);

     return (iten);
}
```

En la class Multimedia extends Activity que por definir de alguna forma sería la madre de todas las demás clases, donde se le pasará el nombre y el número de la opción y se definirán todos los elementos a usar por nuestra aplicación. Esta clase no la creamos, ella sola se crea cuando hacemos el proyecto, nosotros sólo la implementamos, su código sería:

```
public class Multimedia extends Activity implements TextToSpeech.OnInitListener {

    private ListView list;
    private TextToSpeech voz;
    private Opcion[] em;
    private Intent intent;
    boolean primeravez = true;


    /*
     * opciones que mostrara nuestro listView
     */
    public String[] nombres = { "Reproducir Música", "Reproducir Vídeo",
            " Galería de Imágenes", "Grabar Sonido", "Grabar Vídeo", "Cámara" };
    public int nun = 0;

    // constante que identifica el tipo de dialogo a mostrar:

    private static final int DIALOGO_ALERTA = 2;

@Override
public void onCreate(Bundle savedInstanceState) {

     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_multimedia);

     list = (ListView) findViewById(R.id.listView1);

     em = cargarNews();
     MyAdapterProyect adapter = new MyAdapterProyect(this, R.layout.segundo,
             em);
     list.setAdapter(adapter);
     /**
      * sistema de lectura de frases (salida de texto a voz), que pasa un
      * texto escrito, a un formato de audio
      */
     voz = new TextToSpeech(this, this);

     // para mandarlo para las otra main

     list.setOnItemClickListener(new OnItemClickListener() {

         public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                 long arg3) {
             Opcion obj = (Opcion) arg0.getItemAtPosition(arg2);
             nun = arg2;
             clases(nun);
         }

     });
}
```

```
/**
 * Dependiendo del el numero entero que se le pase sera la clase que se
 * ejecutara
 */
public void clases(int nun) {

    switch (nun) {
    case 0:
        intent = new Intent("com.example.listviewopcionesproyect.RepMusic");
        break;
    case 1:
        intent = new Intent(
                "com.example.listviewopcionesproyect.ActiviVideo");
        break;
    case 2:
        intent = new Intent(
                "com.example.listviewopcionesproyect.ActivyGaleria");
        break;
    case 3:
        intent = new Intent("com.example.listviewopcionesproyect.Grabador");
        break;
    case 4:
        intent = new Intent(
                "com.example.listviewopcionesproyect.GrabarVideo");
        break;
    case 5:
        intent = new Intent(
                "com.example.listviewopcionesproyect.ActivCamara");
        break;
    }
    startActivity(intent);
    onInit(nun);
}
```

```
/**
 * para cargar el listView
 */
private Opcion[] cargarNews() {
    Opcion[] n = new Opcion[nombres.length];
    for (int i = 0; i < n.length; i++) {
        n[i] = new Opcion(nombres[i], i);
    }

    return n;
}

@Override
// este es para inflar el menu1 que es el menu principal
public boolean onCreateOptionsMenu(Menu menu) {

    getMenuInflater().inflate(R.menu.menu1, menu);
    return true;
}
```

```
/**
 * segundo acerca de ... te manda a la documentación de android y te muestra
 * el nombre de los programadores , tercero cierra la aplicación
 *
 */
public boolean onOptionsItemSelected(MenuItem item) {
    Intent i;
    switch (item.getItemId()) {
    case R.id.item2:
        break;
    case R.id.item3:
        showDialog(DIALOGO_ALERTA);
        crearDialogoAlerta();

        break;

    case R.id.item4:
        crearDatos();
        break;

    case R.id.item5:
        i = new Intent("android.intent.action.VIEW",
                Uri.parse("http://developer.android.com"));
        startActivity(i);
        break;
    }
    return true;
}

/**
* Datos de los programadores del proyecto
* se mostrara en un toast personalizado
*
*/

   private void crearDatos() {
       Toast toast3 = new Toast(getApplicationContext());

       LayoutInflater inflater = getLayoutInflater();
       //aqui le pongo la segunda acti.xml y el liner layout
       View layout = inflater.inflate(R.layout.toast,
                       (ViewGroup) findViewById(R.id.linerLayout));



       toast3.setDuration(Toast.LENGTH_LONG);
       toast3.setView(layout);
       toast3.show();

   }
```

```
// este método es para crear la clase dialogo
protected Dialog onCreateDialog(int id) {
    Dialog dialogo = null;

    if(id==DIALOGO_ALERTA){
        dialogo = crearDialogoAlerta();
    }else{
        dialogo = null;
    }
    return dialogo;
}
```

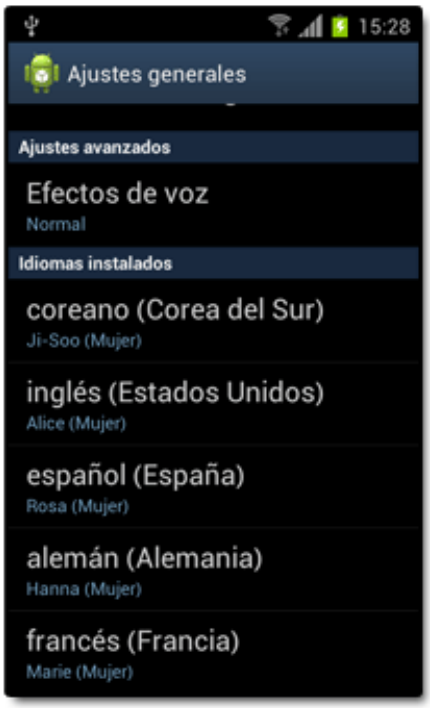
```
// aqui saltara la alerta para que el usuario sepa que a dado a finalizar y lo leera en voz alta

private Dialog crearDialogoAlerta() {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);

    builder.setTitle("Información");// TITULO DE LA VENTANA
    builder.setMessage("Alerta finalizara la aplicación.");// MENSAJE A
                                                           // SALIR
    voz.speak("Alerta finalizará la aplicación", TextToSpeech.QUEUE_FLUSH,
            null);
    builder.setPositiveButton("ok", new OnClickListener() {

        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
            finish();// cuando se le de al ok finalizara la aplicación
        }
    });
    return builder.create();

}
```

Bien ahora explicaremos que es todo este código anterior. Empezaremos por implements TextToSpeech.OnInitListener, esto no es más que la llamada o implementación para usar el lector que tiene Android, es decir:

En **Android** existe la denominada **síntesis de voz** o **TTS** (Text-to-Speech, en inglés), que como bien dicen sus siglas no es más que un sistema de lectura de frases (salida de texto a voz), que pasa un **texto escrito**, a un formato de **audio leído**, idóneo para usuarios con problemas de visión, conducción en carretera, lectura de ebooks o simplemente, cuando no nos apetece leer.

El sistema, generalmente, incorpora un **motor TTS** instalado por defecto, con el cual las aplicaciones (o el propio sistema) pueden leer textos en voz alta.

En **Ajustes** / **Idioma e introducción** / **Salida de texto a voz** se puede elegir y seleccionar el motor TTS que queremos definir como principal, junto a opciones de personalización.



```
/**
 * a este método se le pasa un entero que sera de la posición del list donde
 * se halla dado el click y esto le lo que se encuentre en el array
 * en esa posición */

public void onInit(int nun) {

    if (primeravez) {
        voz.speak("bienvenido al proyecto multimedia",
                TextToSpeech.QUEUE_FLUSH, null);
        primeravez = false;
    } else {
        voz.speak(nombres[nun], TextToSpeech.QUEUE_FLUSH, null);
    }

}
```

En este método en específico, lo que hacemos es pasarle el texto a leer y le ponemos que utilice .QUEUE_FLUSH que no es más, que todas las entradas en la cola de reproducción y que realizará esta acción tantas veces como se llame a este método.

Por otro lado el Intent es una descripción abstracta de una operación a realizar. Se puede utilizar con startActivity para lanzar una actividad, que es lo que realizaremos nosotros al darle click encima de una de las opciones, pues dependiendo si es una u otra nos remitirá a esa actividad, (es decir damos sobre la cámara y nos remitirá a la actividad de la cámara se puede considerar como la unión entre actividades), para lograr esto anterior tenemos que realizar algunas acciones en AndroidManifest.xml
Hay dos formas de realizarlo:

La primera que es la de ir a AndroidManifest.xml y poner el siguiente código:

-Nombre de la clase a donde se quiere ir.

```
<activity
    android:name=".ActivCamara"
```

-Nombre que queremos que se vea cuando se habrá la actividad.

```
android:label="Cámara" >
```

-Esta ruta es la más importante, es como la dirección, sería el nombre del proyecto y el nombre de la clase de destino esta ruta tiene que está escrita igual tanto en Multimedia como aquí:

```
<intent-filter>
    <action android:name="com.example.listviewopcionesproyect.ActivCamara" />
```

-Y default para que no sea la primera en ejecutarse.

```
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
```

El AndroidManifest.xml de nuestra aplicación se vería así después de haber utilizado esta forma :

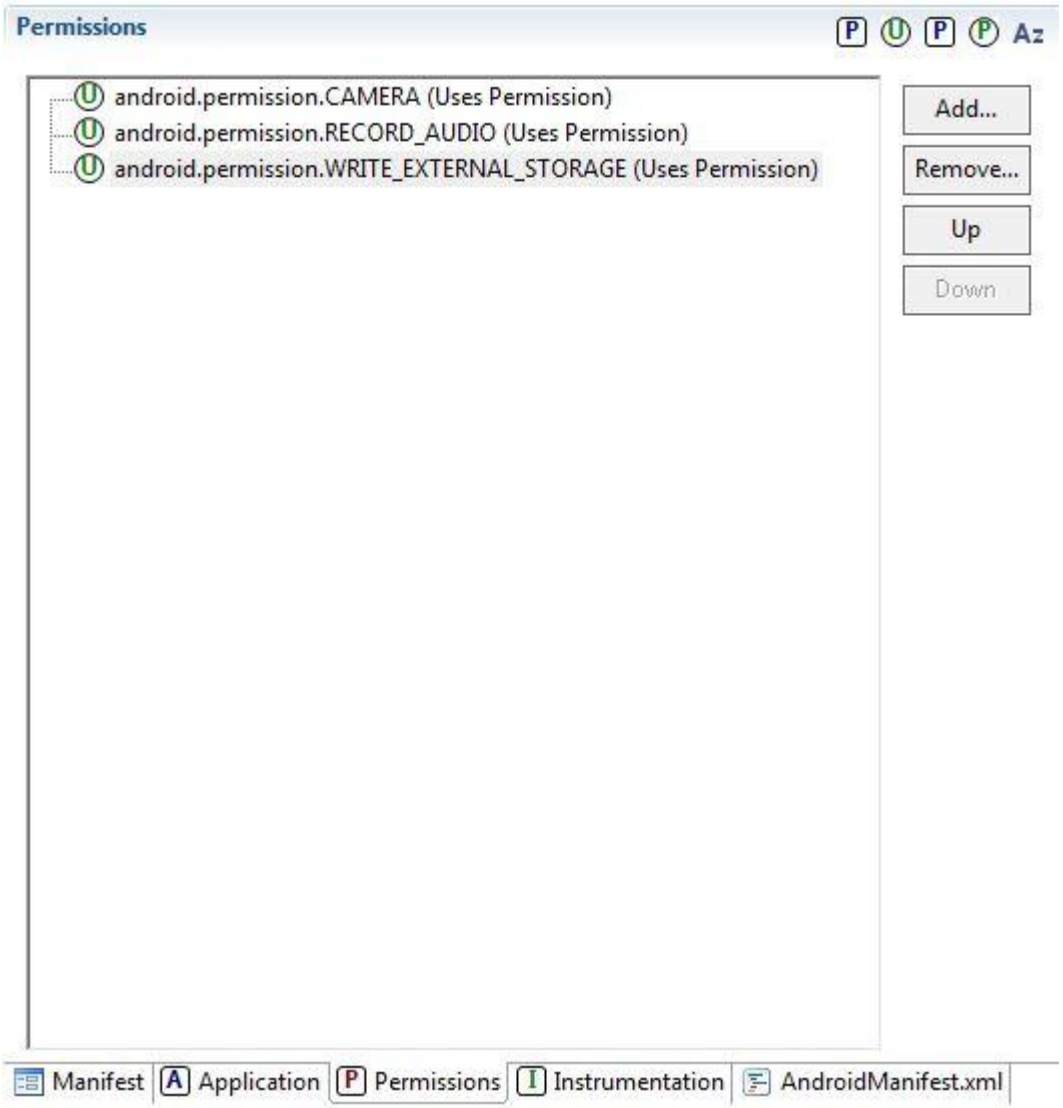
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.iessanandres.proyectomultimedia"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".Multimedia"
            android:label="@string/title_activity_multimedia" android:configChanges="mcc|mnc|locale|
            touchscreen|keyboard|keyboardHidden|navigation|screenLayout|uiMode|screenSize|smallestScreenSize|fontScale">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>


        <activity
            android:name=".ActivCamara"
            android:label="Cámara" >
            <intent-filter>
                <action android:name="com.example.listviewopcionesproyect.ActivCamara" />

                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
```

```
<activity
    android:name=".RepMusic"
    android:label="Reproductor Música" >
    <intent-filter>
        <action android:name="com.example.listviewopcionesproyect.RepMusic" />

        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>

 <activity
    android:name=".ActivyGaleria"
    android:label="Visor" >
    <intent-filter>
        <action android:name= "com.example.listviewopcionesproyect.ActivyGaleria" />

        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>

  <activity
    android:name=".ActiviVideo"
    android:label="Video" >
    <intent-filter>
        <action android:name="com.example.listviewopcionesproyect.ActiviVideo" />

        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>

         <activity
          android:name=".Grabador"
          android:label="Grabador de Voz" >
          <intent-filter>
              <action android:name="com.example.listviewopcionesproyect.Grabador" />

              <category android:name="android.intent.category.DEFAULT" />
          </intent-filter>
      </activity>


          <activity
          android:name=".GrabarVideo"
          android:label="Grabador de Video" >
          <intent-filter>
              <action android:name="com.example.listviewopcionesproyect.GrabarVideo" />

              <category android:name="android.intent.category.DEFAULT" />
          </intent-filter>
      </activity>



  </application>

</manifest>
```

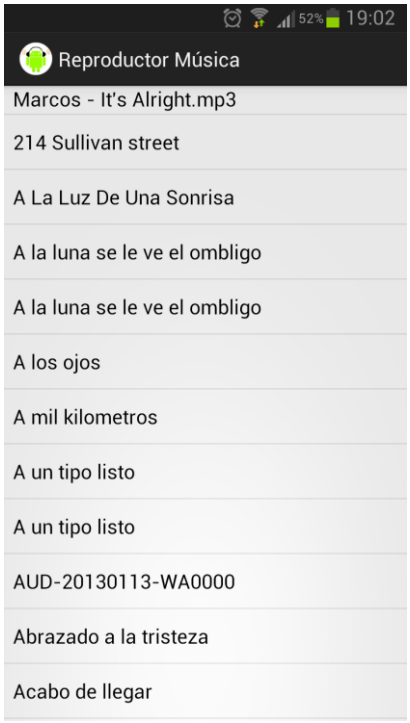
La segunda forma sería la siguiente :

<activity android:name="SecondActivity"> </activity>

Los permisos que tendrá nuestro proyecto serán los siguientes:



3.- Lista Reproducción Música



En esta opción queríamos que apareciera la lista de todos los títulos de sonidos, internos del dispositivo. Para ello la actividad tendrá que extender del ListActivity.

Tendremos un SimpleCursorAdapter, este objeto es un cursor que asigna columnas para TextView o ImageView, definidos en un archivo XML. Puede especificarse las columnas que se desea mostrar, y el archivo XML que define la aparición de estos puntos de vista.

Tendremos también una Uri, que será la ruta de acceso a los sonidos internos del dispositivo.

En el método OnCreate(), cogeremos los títulos de sonidos, y número total de ellos, y los meteremos en un cursor, junto con la ruta de acceso, y de aquí se mostrarán las filas. Ver código a continuación:

```
public class RepMusic extends ListActivity {

    // Filas del la lista
    SimpleCursorAdapter fila;
    // Ruta de las diferentes sonidos internos del dispositivo
    final Uri rutaAudio = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;

    @SuppressWarnings("deprecation")
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // coger los títulos de los sonidos
        String[] titulos = { MediaStore.MediaColumns.TITLE };
        // número total de sonidos que tiene el dispositivo
        int[] to = { android.R.id.text1 };

        // cursor que guarda, la ruta del sonido, un intent, una selección, y
        // los títulos.
        @SuppressWarnings("deprecation")
        Cursor cursor = managedQuery(rutaAudio, null, null, null,
                MediaStore.Audio.Media.TITLE);

        // y la fila se carga en esta ventana, con lo que contenga el cursor,
        // los títulos y el número de sonidos q tenga el dispositivo
        fila = new SimpleCursorAdapter(this,
                android.R.layout.simple_list_item_1, cursor, titulos, to);

        // se carga la fila del adaptador
        setListAdapter(fila);
    }
```

El siguiente método es para la selección de un sonido. Nosotros utilizamos un cursor, éste nos proporcionaría de forma aleatoria la lectura-escritura de acceso al conjunto de resultados devuelto por una consulta de base de datos, pero en nuestro caso, guardamos el número de fila seleccionado por el usuario, y en otra variable de tipo Uri, guardamos la ruta de acceso al sonido y el número de audio escogido.

Con esta última variable y utilizando el ACTION_VIEW, abriríamos el reproductor de audio de Android, reproduciendo así el sonido escogido.

```
// Selección del sonido de la lista
@Override
protected void onListItemClick(ListView l, View v, int position, long id) {

    // Se crea un cursor que coge la fila
    Cursor cursor = fila.getCursor();

    // Y se coge la posición del audio marcado
    cursor.moveToPosition(position);

    // Se guarda el numero de la fila escogida
    String num = cursor.getString(cursor
            .getColumnIndex(MediaStore.Audio.Media._ID));

    // Se guarda la ruta del sonido, con la ruta escogida y el numero
    // seleccionado
    Uri playableUri = Uri.withAppendedPath(rutaAudio, num);

    // Se muestra un mensaje con la ruta del sonido escogido
    Toast.makeText(this, "Play: " + playableUri.toString(),
            Toast.LENGTH_SHORT).show();

    // Se abre en una ventana nueva, para reproducir el sonido
    Intent intent = new Intent();
    intent.setAction(Intent.ACTION_VIEW);
    intent.setData(playableUri);
    startActivity(intent);

}
```

Conclusión:

Con el reproductor de música hemos utilizado cursores, para recoger la ruta interna del sonido, y poder utilizarlo con posterioridad.

La forma más sencilla es utilizar el intent ACTION_VIEW, para lanzar la aplicación interna del dispositivo.

Para más información:
http://developer.android.com/reference/android/provider/MediaStore.Audio.html

4.- Lista Reproducción Videos

En este proyecto de lista de reproducción de Videos, sólo habría que cambiar, la ruta de acceso del MediaStore, hacia Videos, con:

MEDIASTORE.VIDEO.MEDIA.EXTERNAL_CONTENT_URI;

El resto del código es igual que el anterior.

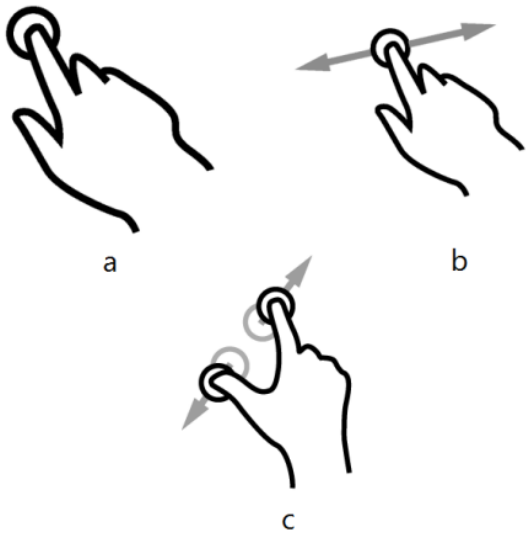
```
public class ActiviVideo extends ListActivity {

    SimpleCursorAdapter adapter;
    final Uri mediaSrc = MediaStore.Video.Media.EXTERNAL_CONTENT_URI;

    @SuppressWarnings("deprecation")
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // coger los titulos de los videos
        String[] titulos = { MediaStore.MediaColumns.TITLE };
        int[] to = { android.R.id.text1 };

        Cursor cursor = managedQuery(mediaSrc, null, null, null,
                MediaStore.Audio.Media.TITLE);

        adapter = new SimpleCursorAdapter(this,
                android.R.layout.simple_list_item_1, cursor, titulos, to);
        setListAdapter(adapter);
    }
```

Para más información:
http://developer.android.com/reference/android/provider/MediaStore.Video.html

5.- Galería de Imágenes

La Galería de imágenes consta de una clase implementada por el evento OnTouchListener, que no es más que la utilización táctil, para modificar o desplazar la imagen.

Multi-touch es simplemente una extensión de la interfaz de usuario común con pantalla táctil, es el uso de dos o más dedos en lugar de uno. Toque, arrastre, desplazamiento, son algunos de los gestos de los dedos que siempre se han apoyado en Android.



Tres gestos táctiles comunes:

a) un grifo (toque)
b) arrastrar
c) zoom pizca. (zoom)

Con el zoom pizca, colocamos los dos dedos en la pantalla y apretamos juntos para hacer que la imagen con la que estamos trabajando se haga más pequeña, o separarlos para hacerla más grande.

Antes de Android 2.0 se tuvo que utilizar un control de zoom torpe con los iconos que ha presionado para acercar y alejar (por ejemplo, los setBuiltInZoomControls()). Pero gracias a su nuevo soporte multi-touch, ahora se puede pellizcar para hacer zoom en Android. Siempre y cuando la aplicación lo admite, por supuesto.

Advertencia:
Multi-touch, tal como se aplica en los actuales teléfonos con Android es muy ligero. De hecho, es tan erróneo que roza lo inservible. La API publica periódicamente los puntos de datos no válidos o imposible, sobre todo durante la transición de un dedo a dos dedos en la pantalla y viceversa.

Para la interfaz gráfica pondrá el siguiente código:

```
<?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:background="@drawable/vi"
    android:orientation="vertical" >

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_weight="0.06"

        android:scaleType="matrix" />

</LinearLayout>
```

Con el fin de mover y acercar la imagen que vamos a usar una característica de la clase ImageView, la llamada transformación de matriz. Utilizando una matriz que puede representar cualquier tipo de traducción, rotación o sesgo que queramos hacer en la imagen.
Esta clase ya entiende lo dicho con anterioridad solamente indicándole android:scaleType = "matrix" :atributo que indica que vamos a utilizar una matriz para controlar la posición y la escala de la imagen.
En android manifest no se pone nada, ya que por algunos foros cuando buscábamos información decían que había que agregar permisos, no es necesario hacer nada de eso, de hecho no sirve para nada, lo que sí es obligatorio es el método dumpEvent(), que lo explicaremos más abajo.

Empezaremos declarando las variables a utilizar, como se muestra a continuación:

```
public class ActivyGaleria extends Activity implements OnTouchListener {

    public int code = 1;
    private Intent intent;
    private ImageView iv;

    private static final String TAG = "Touch";

    /**
     * Estas matrices se utilizan para mover y ampliar imagen tines que
     * importar, la grafica sino no funciona, tenemos que declarar dos matrices
     * como variables (una para el valor de la corriente y otro para el valor
     * inicial antes de la transformación)
     */
    Matrix matrix = new Matrix();
    Matrix savedMatrix = new Matrix();

    // Se puede estar en uno de estos 3 estados cuando estamos utilizando la
    // matrix
    static final int NINGUNO = 0;
    static final int DRAG = 1;
    static final int ZOOM = 2;
    int mode = 0;

    // representa las coordenadas y puntos flotantes(inicial, y de en medio) que
    // definen
    // si la imagen está en un plano bidimensional o no
    PointF start = new PointF();
    PointF mid = new PointF();
    float oldDist = 1f;
```

El método onCreate, se relaciona el control del layout, con el código, y se le añade el evento setOnTouchListener, que lo que permite es modificar, desplazar o rotar en su propio eje la imagen. Y se llama al evento galería, que llamará mediante un intent (ACTION_PICK) a la galería interna del dispositivo, y se cogerán las imágenes que se encuentran en la memoria interna del sistema.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.visor);

    iv = (ImageView) findViewById(R.id.imageView1);
    // aqui le asignas a la imagen evento multitouch
    iv.setOnTouchListener(this);
    galeria();
}

/**
 * para ir a la galería necesitamos un intent, ya que utilizaremos la
 * galeria del dispositivo y un código distintos que asignamos en nuestro
 * caso un 1 siempre tienen que ser mayor que 0
 */
private void galeria() {
    intent = new Intent(Intent.ACTION_PICK,
            android.provider.MediaStore.Images.Media.INTERNAL_CONTENT_URI);
    // mandarlo la galeria de imágenes
    startActivityForResult(intent, code);

}
```

```
/**
 * Actividades que devuelven resultados: la actividad padre sobrescribirá
 * onActivityResult(int requestCode, int resultCode, Intent data)
 *
 * recibimos el URI de la imágen y construimos un Bitmap a partir de un
 * stream de bytes por eso utilizamos en InputStream que seria para la
 * apertura del canal de comunicación el BufferedInputStream nuevo,
 * proporcionando la posibilidad de leer bytes. El paso de una fuente null
 * crea un BufferedInputStream cerrado. Todas las operaciones de lectura en
 * tal corriente se producirá una IOException. En caso de que ocurra la
 * exception llamara el método salir que nos devolvera a la main principal
 */

protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    Uri selectedImage = null;
    InputStream is;

    selectedImage = data.getData();

    if (resultCode != RESULT_CANCELED) {
        if (selectedImage != null) {
            try {
                is = getContentResolver().openInputStream(selectedImage);
                BufferedInputStream bis = new BufferedInputStream(is);
                Bitmap bitmap = BitmapFactory.decodeStream(bis);

                iv.setImageBitmap(bitmap);

            } catch (FileNotFoundException e) {
                Toast.makeText(this, "tienes que elegir una imagen",Toast.LENGTH_SHORT).show();
            }


        } else {
            finish();
        }
    } else {
        Toast.makeText(this, "galeria cancelada", Toast.LENGTH_SHORT).show();
    }
}
```

Ahora vamos a utilizar el onTouch() para transformar la imagen ,este es un método boolean nos devuelve si ha sido cargado o no , en caso positivo es que hubo un movimiento y responderá ante él. También se podrá ver la utilización de la clase Log que en el api su utiliza para la salida de registros, generalmente se utilizan los métodos Log.v () ,Log.d (), Log.i () ,Log.w () y Log.e ().Esta clase nunca puede ser compilada en una aplicación ,excepto durante el desarrollo de la misma.
Consejo: declarar un TAG como constante y lo pudimos observar al inicio de nuestra clase para después llamarlo como se podrá ver en el código que exponemos a continuación .

```
@Override
public boolean onTouch(View v, MotionEvent event) {
    iv = (ImageView) v;

    // Descarga evento táctil para iniciar
    dumpEvent(event);
    // Procesar eventos de toque aquí
    /**
     * Cuando se inicia el gesto recordamos el valor actual de la matriz de
     * transformación y la posición inicial del puntero. Cada vez que el
     * dedo se mueve, se inicia la matriz de transformación a lo largo de su
     * valor original .Un gesto de arrastre empieza cuando el primer dedo se
     * presiona a la pantalla (ACTION_DOWN) y termina cuando se retira (o
     * ACTION_UP ACTION_POINTER_UP).
     */
    switch (event.getAction() & MotionEvent.ACTION_MASK) {
    case MotionEvent.ACTION_DOWN:
        savedMatrix.set(matrix);
        start.set(event.getX(), event.getY());
        Log.d(TAG, "mode=DRAG");
        mode = DRAG;
        break;
    case MotionEvent.ACTION_UP:
    case MotionEvent.ACTION_POINTER_UP:
        mode = NINGUNO;
        Log.d(TAG, "mode=NINGUNO");
        break;


    case MotionEvent.ACTION_POINTER_DOWN:
        oldDist = spacing(event);
        Log.d(TAG, "oldDist=" + oldDist);
        if (oldDist > 10f) {
            savedMatrix.set(matrix);
            midPoint(mid, event);
            mode = ZOOM;
```

```
            Log.d(TAG, "mode=ZOOM");
        }
        break;

    case MotionEvent.ACTION_MOVE:
        if (mode == DRAG) {
            matrix.set(savedMatrix);
            matrix.postTranslate(event.getX() - start.x, event.getY()
            - start.y);
        } else if (mode == ZOOM) {
            float newDist = spacing(event);
            Log.d(TAG, "newDist=" + newDist);
            if (newDist > 10f) {
                matrix.set(savedMatrix);
                float scale = newDist / oldDist;
                matrix.postScale(scale, scale, mid.x, mid.y);
            }
        }
        break;

    }

    //
    /**
     * Realizar la transformación // La variable de matriz se calcula dentro
     * de la sentencia switch cuando // ponemos en práctica los gestos. es
     * decir la imagen cambian cuando la tocamos
     */
    iv.setImageMatrix(matrix);

    return true;// indica evento estuvo a cargo
}
```

```
//metodo que calcula el espacio inicial al final
private float spacing(MotionEvent event) {
    float x = event.getX(0) - event.getX(1);
    float y = event.getY(0) - event.getY(1);
    return FloatMath.sqrt(x * x + y * y);

}


//metodo que calcula del inicio al medio
private void midPoint(PointF point, MotionEvent event) {
    float x = event.getX(0) + event.getX(1);
    float y = event.getY(0) + event.getY(1);
    point.set(x / 2, y / 2);
}
```

El método dumpEvent() en nuestra actividad, no se puede ejecutar en el emulador (en realidad se puede, pero el emulador no es compatible con multi-touch, de modo que los resultados no serán muy interesantes). Y lo que hace es decir donde ha pulsado el

usuario, si ha hecho un arrastre por la pantalla (esto dirá que el usuario ha hecho ACTION_MOVE en ciertas coordenadas) y así sucesivamente con el resto de movimientos .

```
//este se pone para saber que movimiento se hace
// ** Muestra un evento en la vista Logcat, para la depuración aqui es de
// donde se toma del movil o table la posición donde se a pulsado* /

private void dumpEvent(MotionEvent event) {
    String[] nombre = { "ABAJO", "UP", "movimiento", "CANCELAR", "fuera",
            "POINTER_DOWN", "POINTER_UP", "7?", "8?", "9?" };

    StringBuilder sb = new StringBuilder();
    int action = event.getAction();
    int actionCode = action & MotionEvent.ACTION_MASK;
    sb.append("event ACTION_").append(nombre[actionCode]);


    if (actionCode == MotionEvent.ACTION_POINTER_DOWN
            || actionCode == MotionEvent.ACTION_POINTER_UP) {
        sb.append("(pid ").append(
                action >> MotionEvent.ACTION_POINTER_ID_SHIFT);
        sb.append(")");
    }
    sb.append("[");
    for (int i = 0; i < event.getPointerCount(); i++) {
        sb.append("#").append(i);
        sb.append("(pid ").append(event.getPointerId(i));
        sb.append(")=").append((int) event.getX(i));
        sb.append(",").append((int) event.getY(i));
        if (i + 1 < event.getPointerCount())
            sb.append(";");
    }
    sb.append("]");
    Log.d(TAG, sb.toString());
}
```

El paquete android.gesture proporciona una forma de crear, reconocer, cargar y guardar los gestos. Desafortunadamente, no es muy útil en la práctica. Entre otros problemas, es que no viene con un conjunto estándar de una función de gestos (como tocar y arrastrar) y no es compatible con gestos Multi-Touch como pellizcar zoom.

Conclusión:

Uso y manejo de Multi-touch.

Utilización de intents para para realizar acciones específicas con el acceso a la galería de fotos y la visualización de una de las imágenes escogida .

Para más información:

http://developer.android.com/reference/android/view/View.OnTouchListener.html

6.- Grabación de audio

Para esta aplicación la forma más sencilla de capturar audio en Android es mediante el grabador que provee el propio sistema operativo.

```
public class Grabador extends Activity implements TextToSpeech.OnInitListener {


    private Button grabar;
    private Button repro;
    int peticion = 2;
    Uri url1;
    Intent intent = null;
    private TextToSpeech voz;

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

        grabar = (Button) findViewById(R.id.boton1);

        repro = (Button) findViewById(R.id.boton2);
        voz = new TextToSpeech(this, this);
    }
```

Invocamos la aplicación de grabación (tiene como ventana que la interfaz le es familiar al usuario, ya que muchas aplicaciones utilizan esta característica) y luego recuperamos el audio grabado.

Cuando se presiona el botón de grabar el audio mediante un Intent activamos la aplicación de grabación propia de Android. Seguidamente llamamos al método startActivityForResult para poder recuperar la grabación después de finalizada a través del método onActivityResult:

```
/**
 *
 * mediante el Intent activamos la aplicación de grabación propia de
 * Android. Seguidamente llamamos al método startActivityForResult para
 * poder recuperar la grabación luego de finalizada a través del método
 * onActivityResult: lo de la peticion es que hay que pasarle un valor
 * positivo mayor que 0
 */
public void grabar(View v) {
    intent = new Intent(MediaStore.Audio.Media.RECORD_SOUND_ACTION);
    startActivityForResult(intent, peticion);

}
```

Debemos pasar al método startActivityForResult además de la referencia del Intent una variable con un valor mayor que 0 (luego este valor retornará al método onActivityResult)

Finalizada la grabación se ejecuta el método onActivityResult, donde almacenamos en la variable url1 la referencia al archivo de audio creado si por algún motivo damos a reproducir ante de grabar te saldrá un mensaje que dice que no hay grabación :

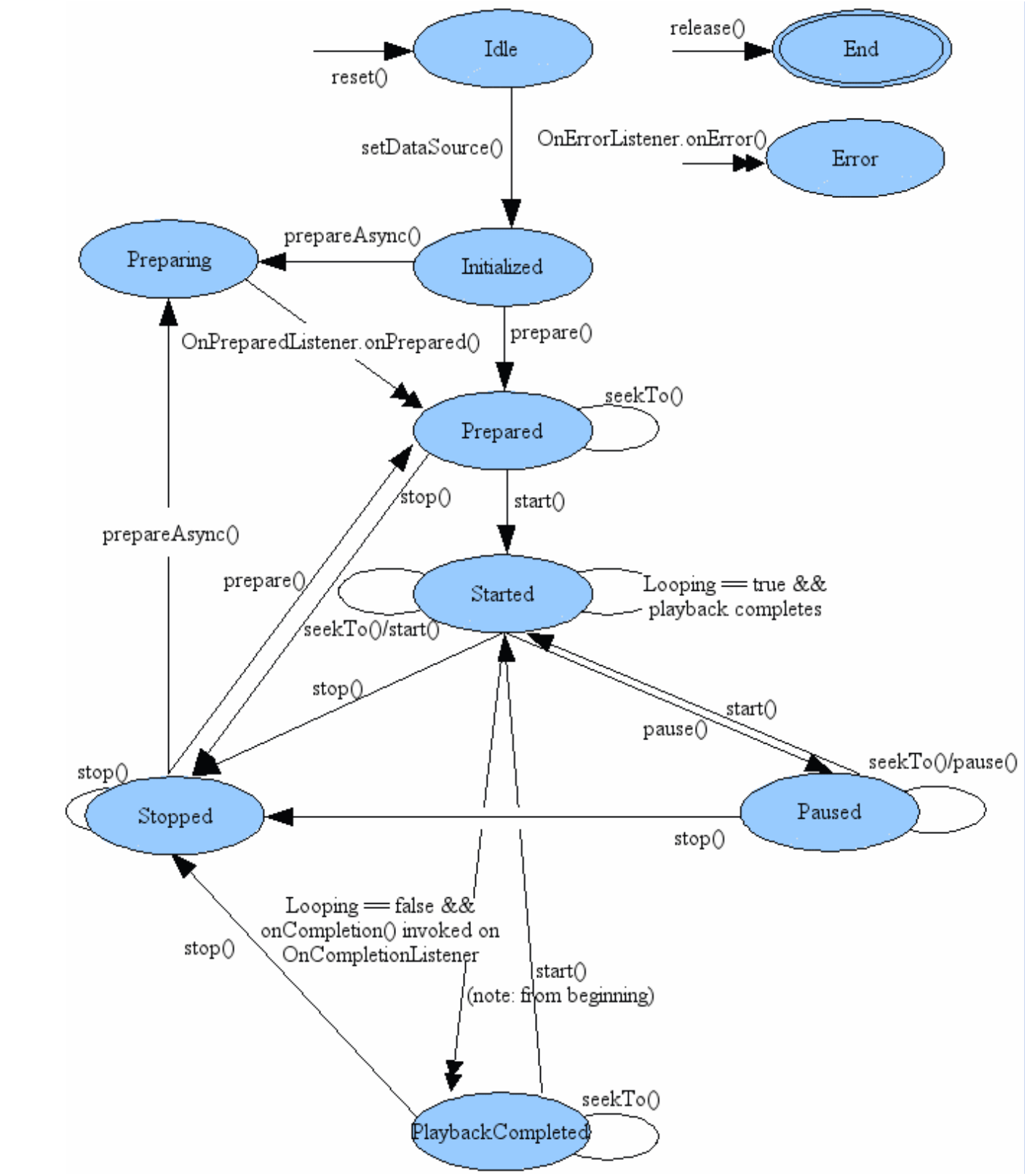
```
/**
 * esto es en la ruta a guardar la el audio y depues reproducirlo mediante la
 * uri que se le dara al Mediaplayer
 */
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == RESULT_OK && requestCode == peticion) {
        url1 = data.getData();
    }
}
```

Por último para ejecutar el contenido de la grabación utilizamos la clase MediaPlayer que se puede utilizar para controlar la reproducción de archivos de audio / vídeo y corrientes:

```
/**
 * para poder reproducir el audio utilizamos la clase mediaplayer y hacemos
 * que funcione mediante el start
 */
public void repoducir(View v) {
    if (url1 == null) {
        voz.speak("no hay grabación", TextToSpeech.QUEUE_FLUSH, null);
        Toast.makeText(this, "no hay grabación", Toast.LENGTH_SHORT).show();
    } else {
        MediaPlayer mediaPlayer = MediaPlayer.create(this, url1);
        mediaPlayer.start();
    }
}
```

El control de reproducción de archivos de audio / video se gestiona como una máquina de estados. El siguiente diagrama muestra el ciclo de vida y los estados de un objeto MediaPlayer impulsado por las operaciones de control de reproducción compatibles. Los óvalos representan los estados de un objeto MediaPlayer pueden residir pulg. Los arcos representan las operaciones de control de reproducción que impulsan la transición del estado del objeto.

Hay dos tipos de arcos. Los arcos con una punta de flecha representan solo llamadas síncronas a métodos, mientras que aquellos con una punta de flecha doble representar llamadas asincrónicas a métodos.



A partir de este diagrama de estado, se puede ver que un objeto MediaPlayer tiene los siguientes estados:

- Cuando un objeto MediaPlayer se crea (nuevo() o reset()), y se llama, pasa al estado de inactividad. Si se llama después del método liberación() pasará al estado terminado. Entre estos dos estados el de inactividad, y terminado es el ciclo de vida del MediaPlayer.
    - Hay una diferencia entre un objeto MediaPlayer creado y otro después del reset(), que pasaría a inactividad. Cuando está en este estado y se llama a alguno de estos métodos: flotar(), pausa(), start(), seekTo(),prepare(), o prepareAsync(), daría la excepción OnErrorListener.onError().
    - Se recomienda que un objeto MediaPlayer cuando no se esté utilizando se llame al método release(), para que los recursos que se estén utilizando (aceleración del hardware) puedan ser liberados. Si no podría fallar el objeto MediaPlayer y llegar al estado de terminar, que no se puede volver a utilizar.
    - Los objetos MediaPlayer creados con el método nuevo() se encuentran en el estado Ilde, mientras que los creados con los sobrecargados están en estado preparado().

- En general, una operación de reproducción puede fallar, por diversas razones: el formato de audio/video es incompatible, mal interpretado, resolución muy alta, tiempo de espera muy alto,… Por lo tanto los informes de errores son una parte muy importante. Por ello el reproductor interno proporciona la excepción OnErrorListener.onError().
    - Es importante que cuando se entra en una excepción el objeto MediaPlayer entra en el estado de error, incluso si no está añadida dicha excepción en la aplicación.
    - Para volver a utilizar el objeto MediaPlayer cuando está en el estado de error, puede llamarse al método reset() para restaurarlo.
    - La excepción IllegalStateException se produce para evitar errores de programación por métodos como prepare(), prepareAsync().

- Con la llamada de los métodos SetDataSource(FileDescriptor), SetDataSource(String), SetDataSource(Contexto, Uri) o SetDataSource (FileDescriptor, largo, largo) se pasa el objeto MediaPlayer de un estado de inactividad a inicializado.
    - La excepción IllegalStateExcepcion se produce si SetDataSource() se llama a cualquier otro estado que no sea el inicializado.
    - También las excepciones IllegalArgumentException y IOException pueden lanzarse a través del método SetDataSouerce.

- Un objeto MediaPlayer debe estar en el estado preparado antes de iniciar la reproducción.
    - Hay dos maneras síncrona y asíncrona para llegar al estado preparado.

- Prepare() (síncrono) transfiere el objeto al estado preparado una vez devuelto.
- PrepareAsync() (asíncrona) primero transfiere el objeto del estado preparado y después lo devuelve mientras sigue trabajando.

  - El estado preparado es transitorio, y soporta que llamemos a cualquier método mientras el objeto MediaPlayer está en ese estado.
  - La excepción IllegalStateException se produce con los métodos prepare(), prepareAsync().
  - Mientras que en el estado preparado, se encuentra el método screenOnWhilePlaying en él podemos ajustar el volumen.

- Para iniciar la reproducción se llama a start(), si te devuelve el estado preparado, podremos llamar a isPlaying() para comprobar si está iniciado.
  - Si en la preparación del estado se llama al método OnBufferingUpdateListener.onBufferingUpdate(), te permitirá llevar un seguimiento del estado de almacenamiento del buffer.
  - Al llamar a start() no tiene ningún efecto sobre el objeto MediaPlayer, ya que está en la preparación del estado.

- La reproducción se puede detener a través del método pausa(), en la posición actual. Pero debemos tener cuidado porque el cambio del estado start() a pausa() es de forma asíncrona. Puede tardar algunos segundos de retraso en actualizarse con el método isPlaying().
  - Podemos llamar al método start() cuando el objeto MediaPlayer está en pausa y queremos reanudar en la misma posición.
  - Para pausar el objeto MediaPlayer se hace con el método pausa().

- Si queremos detener el objeto MediaPlayer iniciado anteriormente, podemos hacerlo con el método stop()
  - Una vez que el objeto MediaPlayer está en el estado parado, la reproducción no se puede reanudar hasta llamar a los métodos prepare() o prepareAsync(), y pasar a un nuevo estado.
  - El método stop() no tiene ningún efecto si el objeto MediaPlayer se encuentra parado.

- La posición de reproducción se puede ajustar con una llamada a **seekTo (int)** .
  - Cuando llamamos de modo asíncrona al método seekTo(int), la operación puede llevar algún tiempo de retraso, sobre todo si es de audio/video.
  - Tenga en cuenta que **seekTo (int)** también se puede llamar en los otros estados, como Preparado , En pausa y PlaybackCompleted estado.

- Además, la posición actual de reproducción actual se puede recuperar con una llamada a **getCurrentPosition ()** , que es útil para aplicaciones tales como un reproductor de música que necesita para realizar un seguimiento del progreso de la reproducción.

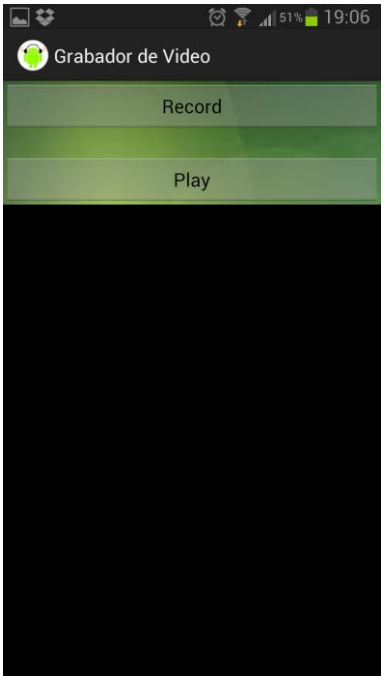
- Cuando la reproducción alcanza el final, la reproducción finaliza.
  - Si el modo de bucle se estaba preparando para *verdadero* con **setLooping (booleano)** , el objeto MediaPlayer permanecerá en la Introducción del estado.
  - Si el modo del bucle se establece en false, te pide la complementación con OnCompletion.onCompletion().
  - Cuando estamos en el estado PlayBackCompleted, llamando al método start(), se puede reiniciar la reproducción ya sea audio o video.

Conclusión:

Manejo y configuración de Media Player: una vez grabado un sonido, el Media Player nos sirvió para reproducirlo.

Para más información:
http://developer.android.com/reference/android/media/MediaPlayer.html

7.- Grabar Vídeo



Para esta aplicación, hemos propuesto otra forma diferente para grabar el video. Se podría hacer de la misma forma que el Grabar Sonido, cambiando solamente los intent de : RECORD_SOUND_ACTION por ACTION_VIDEO_CAPTURE, pero como decíamos, vamos a cambiar un poco la forma.

Solo necesitaremos poner en el layout 2 botones, uno para grabar y otro para reproducir. Además de un VideoView, para ver la reproducción del video. Y una variable para saber si se ha guardado bien el video, esta como las anteriores tiene que ser un entero mayor que 0.

```
public class GrabarVideo extends Activity implements TextToSpeech.OnInitListener  {

    /*
     * sin permisos, de utilizacion de ningun tipo 1- en el diseño ponemos dos
     * botones, para grabar, y reproducir el video 2- definimos los objetos a
     * utilizar
     */

    private Button bGrabar;
    private Button bReprod;
    private VideoView video;
    private TextToSpeech voz;

    // codigo de android para saber si se ha guardado bien el video
    final static int REQUEST_VIDEO_CAPTURED = 1;
    //guardar la ruta del video donde se guardará
    Uri rutaVideo = null;
```

Más abajo explicamos una forma comentada, por si alguien no le gusta, poner eventos clicks en los botones del layout. Nosotros tenemos relacionados los clicks del layout.

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.grabarvideo);

    bGrabar = (Button) findViewById(R.id.button1);
    bReprod = (Button) findViewById(R.id.button2);
    video = (VideoView) findViewById(R.id.videoView1);
    voz= new TextToSpeech(this,this);

    // //sin clicks en los botones del diseño
    // bGrabar.setOnClickListener(new OnClickListener() {
    //
    // @Override
    // public void onClick(View arg0) {     //
    // //llamar a la aplicacion interna del video propio del dispositivo
    // Intent intent = new Intent(
    // android.provider.MediaStore.ACTION_VIDEO_CAPTURE);
    // startActivityForResult(intent, REQUEST_VIDEO_CAPTURED);
    // }
    // });
    //
    // bReprod.setOnClickListener(new Button.OnClickListener() {        //
    // @Override
    // public void onClick(View arg0) {     //
    // si pulsamos reproducir si n haber grabado nada, nos mostrará un mensaje de
    // no hay video, y en caso contrario, mostraria la reproduccion de mismo, con su ruta
    // if (rutaVideo == null) {
    // Toast.makeText(MainActivity.this, "No hay Video",
    // Toast.LENGTH_SHORT).show();
    // } else {
    // Toast.makeText(MainActivity.this, "Playback: " + rutaVideo.getPath(), Toast.LENGTH_SHORT).s
    // video.setVideoURI(rutaVideo);
    // video.start();
    // }
    // }
```

Para grabar el video, solamente llamaremos a la aplicación de android de grabar video, con un intent de tipo ACTION_VIDEO_CAPTURE:

```
/*
 * este metodo es para aceptar el guardado del video o no. si se guarda el
 * video despues de realizado, se mostrará la ruta del video en caso de
 * cancelar el guardado, se mostrará otro mensaje
 */
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

    if (resultCode == RESULT_OK) {
        if (requestCode == REQUEST_VIDEO_CAPTURED) {
            rutaVideo = data.getData();
            Toast.makeText(this, rutaVideo.getPath(), Toast.LENGTH_SHORT).show();
        }
    } else if (resultCode == RESULT_CANCELED) {
        rutaVideo = null;
        Toast.makeText(this, "Ha cancelado el guardado", Toast.LENGTH_SHORT).show();
    }
}


      // llamar a la aplicacion interna del dispositivo, para grabar
      public void clickGrabar(View v) {
          Intent intent = new Intent(
                  android.provider.MediaStore.ACTION_VIDEO_CAPTURE);
          startActivityForResult(intent, REQUEST_VIDEO_CAPTURED);

      }


 //si pulsamos reproducir sin haber grabado nada, nos mostrará un
 // mensaje de no hay video, y en caso contrario, mostraria la reproduccion de
 // mismo, y el mensaje con su ruta
 public void clickReproducir(View v) {

     if (rutaVideo == null) {
         voz.speak("No hay Video", TextToSpeech.QUEUE_FLUSH, null);

         Toast.makeText(GrabarVideo.this, "No hay Video",
                 Toast.LENGTH_SHORT).show();
     } else {
         Toast.makeText(GrabarVideo.this,
                 "Reproduciendo: " + rutaVideo.getPath(), Toast.LENGTH_SHORT).show();
         video.setVideoURI(rutaVideo);
         video.start();
     }
 }
```

Conclusiones:

Utilización de la cámara de vídeo: hicimos uso de la cámara de grabación del dispositivo y consolidamos los conocimientos del apartado superior de la utilización del MediaPlayer.

Para más información:
http://developer.android.com/reference/android/media/MediaPlayer.html

8.- Android Cámara

La mayoría de los dispositivos Android tienen una cámara. Algunos dispositivos tienen una en la parte delantera y otra en la parte trasera.

La utilización de la cámara en el dispositivo Android se puede hacer a través de la integración de la aplicación de la cámara existente. En este caso se inicia la aplicación de la cámara existente a través de un INTENT, para poder obtener los datos una vez que el usuario vuelve a nuestra aplicación.

También se puede integrar directamente la cámara a la aplicación a través de la cámara de API.

La clase Camera se utiliza para configurar los ajustes de captura de imágenes, inicio / parada de vista previa, fotos instantáneas, y recuperar los marcos. Esta clase gestiona el hardware de la cámara real.

Para acceder a la cámara del dispositivo, debe declarar la CÁMARA . Por ejemplo, si utiliza la cámara y la función de enfoque automático, el Manifiesto debería incluir lo siguiente:

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />
```

El Código de nuestra aplicación sería :

```
public class ActivCamara extends Activity implements
        TextToSpeech.OnInitListener {

    private ImageView iv;
    private String name = "";// para la ruta
    public Intent intent;
    public Uri output = null;
    public final int code = 1;
    private TextToSpeech voz;
    private int num = 0;

    /**
     *
     * SimpleDateFormat clase concreta para dar formato a fechas y analizar de
     * una manera sensible a la localidad. Formateo convierte una fecha en una
     * cadena esto lo utilizaremos para dar nombre a la foto que queremos
     * guardar.
     */
    String fe = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());

/**
 * Environment-Proporciona acceso a las variables de entorno.
 * setExternalStorageDirectory()-Obtiene el directorio Android de
 * almacenamiento externo Este directorio puede ser pensado como medios de
 * almacenamiento compartido. Se trata de un sistema de archivos que puede
 * contener una cantidad relativamente grande de datos y que se comparte
 * entre todas las aplicaciones. Normalmente, se ve como tarjeta SD, pero
 * también puede ser implementado como almacenamiento integrado en el
 * dispositivo .
 *
 * En otras palabras, en nuestro programa la utilizamos para
 * ponerle la ruta donde guardaremos la imágen.
 *
 * no se utiliza ningún layout porque las imagenes se podra ver con los
 * medios del dispositivo movil.
 *
 */
```

El método onCreate: se inicializa con una llamada a getExternalStorageDirectory() que guardará un archivo en la memoria interna del dispositivo, que tendrá por nombre la fecha, la hora, minutos y segundos tomados del sistema y la extensión .jpg, relacionamos el objeto ImageView, con la imagen, y se inicializa el objeto TextToSpeech, explicado con anterioridad.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.camara);
    // para el nombre de la imágen a guardar
    name = Environment.getExternalStorageDirectory() + "/test" + fe
            + ".jpg";

    iv = (ImageView) findViewById(R.id.imageView1);

    // voz inicialización del objeto
    voz = new TextToSpeech(this, this);

    camara();

}
```

La forma más sencilla de tomar fotografías es utilizar un intent con ACTION_IMAGE_CAPTURE, acción que pertenece al Media Store

```
/**
 * aqui tendremos declarado e inicializados la uri donde se encontrara la
 * imagen y el intent para ejecutar la cámara del android
 */
private void camara() {
    // aqui se pasara a la camara directamente

    intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

    output = Uri.fromFile(new File(name));// guardar como tipo de
                                          // archivo

    intent.putExtra(MediaStore.EXTRA_OUTPUT, output);

    startActivityForResult(intent, code);

}
```

La función onActivityResult() es para indicar que queremos hacer con la imagen recibida (ya sea de la cámara o de la galería) una vez ha sido seleccionada. Es necesario revisar si la imagen viene de la cámara TAKE_PICTURE.

Buscamos y creamos el bitmap para el ImageView:
ImageView iv = (ImageView)findViewById(R.id.imgView);
iv.setImageBitmap(BitmapFactory.decodeFile(name));

Si quisiéramos incluir esa imagen en nuestra galería, utilizamos un MediaScannerConnectionClient

```
/**
 * ete método es para ver y guardar en la galería la imagen tomada por la
 * cámara
 */
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

    /**
     * Para guardar la imagen en la galería, utilizamos una conexión a un MediaScanner
     */

    if (resultCode == RESULT_CANCELED) {
        num = 1;
        onInit(num);
        // si se le da a cancel se saldra a la actvidad principal
        finish();

    } else {
        new MediaScannerConnectionClient() {
            private MediaScannerConnection msc = null;
            {
                msc = new MediaScannerConnection(getApplicationContext(),
                        this);
                msc.connect();
            }

            // notifica la llamada de conexion con MediaScanner
            public void onMediaScannerConnected() {
                msc.scanFile(name, null);
            }
```

```
            /**
             * notifica al usuario cuando el escáner terminado de escanear
             * un archivo. Parámetros a pasarle ruta de acceso al archivo
             * que se ha escaneado. el URI para el archivo si la operación
             * tuvo éxito y el escaneo de archivos se añaden a la base de
             * datos de medios de comunicación y después desconecta
             */

            public void onScanCompleted(String path, Uri uri) {
                msc.disconnect();
            }
        };
        num = 2;
        onInit(num);
        iv.setImageBitmap(BitmapFactory.decodeFile(name));

    }

}
```

Conclusión:

Utilización de intents para el uso de la cámara: se utilizaron intents para realizar acciones específicas, la existencia de intents para acciones comunes predefinidas nos facilitan muchas tareas, en este caso no fue necesario acezar directamente el hardware de la cámara si no pudimos tomar una fotografía de una manera más sencilla.

Para más información:
http://developer.android.com/reference/android/hardware/Camera.html