Acceso a cámara

De todas las alternativas existentes para la toma de fotografías desde Android veremos la más secilla. Dicha alternativa consiste en hacer uso de un `Intent` implícito cuyo parámetro sea la constante `ACTION_IMAGE_CAPTURE` definida en la clase `MediaStore` (de la que hablaremos más adelante). Al hacer uso de la siguiente línea de código

```
startActivityForResult(new Intent(MediaStore.ACTION_IMAGE_CAPTURE),
TAKE_PICTURE);
```

se ejecutará la aplicación nativa para la toma de fotografías (o cualquier otra actividad instalada que pueda hacerse cargo de dicha tarea), la cual también le permitirá al usuario modificar las opciones pertinentes. Como vemos estamos ante otro ejemplo de reutilización de componentes dentro del sistema Android que nos va a permitir como desarrolladores ahorrar bastante tiempo a la hora de crear nuestras propias aplicaciones. Debemos recordar que al hacer uso del método `startActivityForResult` la actividad actual quedará a la espera de que la fotografía sea tomada; una vez hecho esto la actividad volverá a estar en ejecución, pasándose el control al método `onActivityResult`.

En versiones anteriores del SDK de Android la emulación de la cámara no estaba soportada. Hoy en día es posible simular la cámara del dispositivo virtual por medio de una webcam, así que ya no es necesario utilizar un dispositivo real para poder probar estos ejemplos.

Se devolverá la imagen de tipo `Bitmap` en el parámetro extra de nombre `data` devuelto por el Intent en el método `onActivityResult()`.

En el siguiente ejemplo tenemos el esqueleto de una actividad en la que se utiliza un `Intent` para tomar una fotografía. Deberemos llamar al método getPicture() . En onActivityResult() se determina examinando el valor del campo extra data devuelto por el Intent. Por último, una vez tomada la

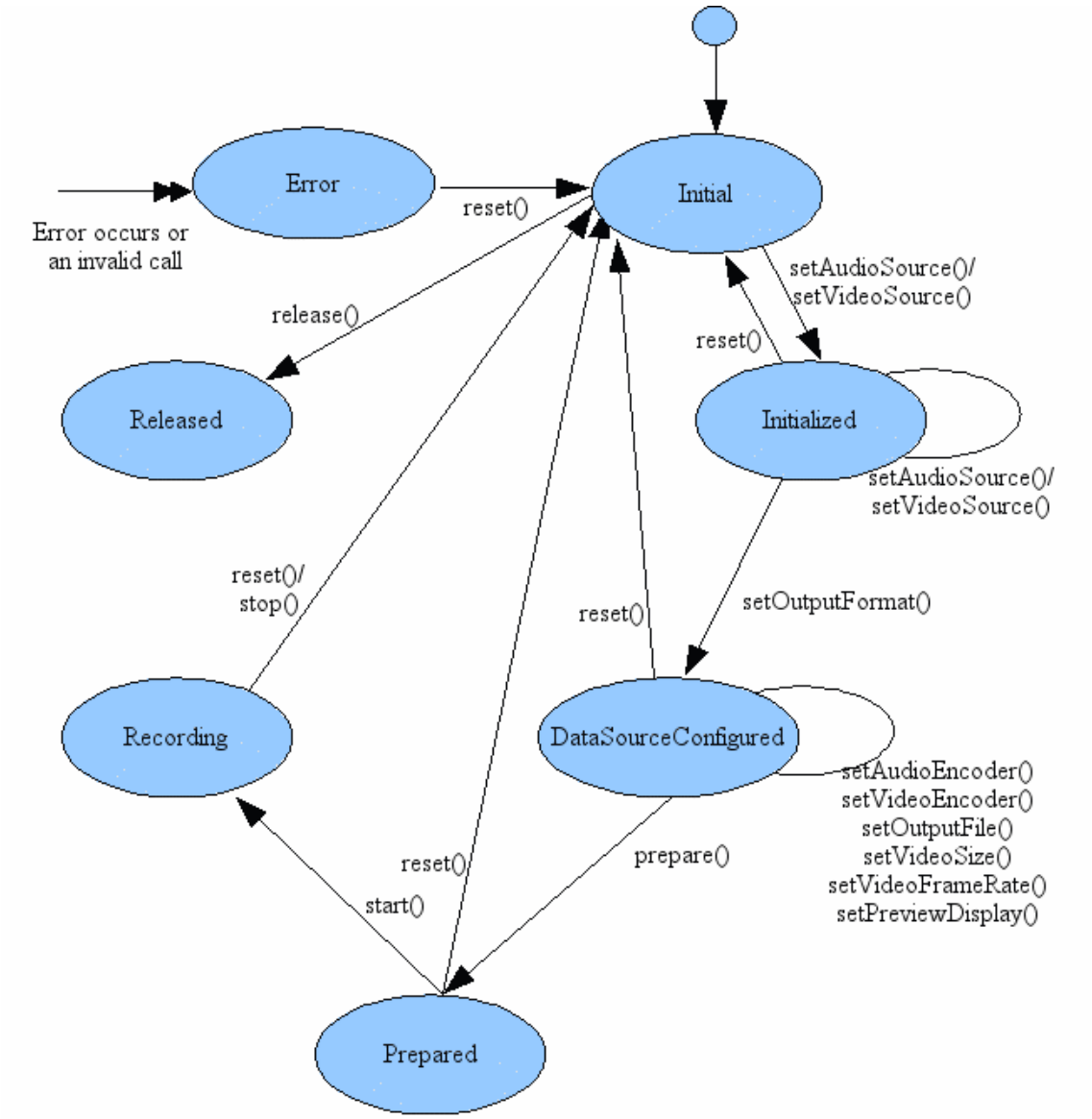
fotografía, se puede almacenar en el Media Store (hablamos de esto un poco más adelante) o procesarla dentro de nuestra aplicación antes de descartarla.

```
// La siguiente constante se usará para identificar el Intent lanzado
// para tomar una fotografía, de tal forma que podamos saber
// en onActivityResult que fue esa la actividad que acaba de terminar
private static int TAKE_PICTURE = 1;

private void getPicture() {
    // Preparamos y lanzamos el intent implícito para la captura
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    startActivityForResult(intent, TAKE_PICTURE);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    if (requestCode == TAKE_PICTURE) {
        // Comprobamos si el Intent ha devuelto la imagen
        if (data != null) {
            if (data.hasExtra("data")) {
                Bitmap thumbnail = data.getParcelableExtra("data");
}
        }
    }
}
```

Grabación de video: MediaRecorder



MediaRecorder state diagram

La clase `MediaRecorder` es similar a la clase `MediaPlayer.` También se basa en una máquina de estados, aunque más simple, que van pasando a medida que llamamos a los métodos asociados. Esto quiere decir que el orden en el cual se inicializa y se realizan operaciones con los objetos de este tipo es importante. En resumen, los pasos para utilizar un objeto `MediaRecorder` serían los siguientes:

1. Crear un nuevo objeto `MediaRecorder`.
2. Asignar la fuente a partir de la cual se grabará el contenido.
3. Definir el formato de salida.
4. Especificar las características del video: códec, framerate y resolución de salida.
5. Seleccionar un fichero de salida.
6. Prepararse para la grabación.
7. Realizar la grabación.
8. Terminar la grabación.
9. Liberar los recursos asociados.

Los métodos `setAudioSource` y `setVideoSource` permiten especificar la fuente de datos por medio de constantes estáticas definidias en `MediaRecorder.AudioSource`, y `MediaRecorder.VideoSource`, respectivamente. El siguiente paso consiste en especificar el formato de salida por medio del método `setOutputFormat` que recibirá como parámetro una constante entre las definidas en `MediaRecorder.VideoEncoder`, respectivamente. Seguidamente podremos configurar también el framerate o la resolución de salida si se desea. Finalmente indicamos la localización donde se guardará el contenido grabado

por medio del método `setOutPutFile` y preparamos el inicio de la grabación llamando al método prepare.

El siguiente código muestra cómo configurar un objeto `MediaRecorder` para capturar audio y video del micrófono y la cámara usando un códec estándar y grabando el resultado en la tarjeta SD:

```
mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
mediaRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);
mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
mediaRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.MPEG_4_SP);
mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);
```

Para comenzar la grabación, una vez inicializados todos los parámetros, utilizaremos el método `start`:

```
mediaRecorder.start();
```

Cuando se desee finalizar la grabación se deberá hacer uso en primer lugar del método stop y a continuación invocar el método `reset`. Una vez seguidos estos pasos será posible volver a utilizar el objeto llamando a de nuevo a `setAudioSource` y `setVideoSource`. Por último, tenemos que llamar a reléase para liberar los recursos asociados al objeto `MediaRecorder` (ya no podrá volver a ser usado, se tendrá que crear de nuevo):

```
mediaRecorder.stop();
mediaRecorder.reset();
mediaRecorder.release();
```

Texto a Voz (TextToSpeech)

Android permite convertir texto en voz. No solo convertirlo sino además hablarlo con diferentes acentos. Para ello Android nos brinda la clase TextToSpeech, para instanciar un objeto de la clase necesitas el contexto de la aplicación, y especificar la instancia que implementará el listener **initListener,** cuyo método onInit tenemos que implementar.

```
TextToSpeech  ttobj=new TextToSpeech(getApplicationContext(), new
TextToSpeech.OnInitListener() {

   @Override

   public void onInit(int status) {

   }

});
```

En este método será dónde debemos añadir las configuraciones necesarias tales como el lenguaje:

```
ttobj.setLanguage(Locale.UK);
```

Una vez ya hemos configurado la instancia, podemos usarlo:

```
ttobj.speak(toSpeak, TextToSpeech.QUEUE_FLUSH, null);
```

Para más información sobre la clase accede a:

https://developer.android.com/reference/android/speech/tts/TextToSpeech.html