Второй курс, осенний семестр 2016/17
Конспект семинаров по программированию под Android

Собрано 24 октября 2016 г. в 02:49

Содержание

Глава #1
Звук
17 октября

1.1. Микрофон

1.1.1. Разрешения

Если мы хотим записывать что-то, используя микрофон, то нужно дописать строку в `AndroidManifest.xml`:

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

При необходимости сохранения результатов на устройстве надо добавить туда же:

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

1.1.2. Запись

За это отвечает класс `android.media.MediaRecorder`. Создаем его экземпляр (пустой конструктор) и начинаем настраивать:

MediaRecorder.setAudioSource — источник аудиопотока. В нашем случае это `MediaRecorder.AudioSource.MIC`

MediaRecorder.setOutputFormat — формат выходного файла. Рекомендуется использовать `MediaRecorder.OutputFormat.AMR_{WB,NB}`. WB означает Wide Band, NB — Narrow Band.

MediaRecorder.setOutputFile — выходной файл либо в виде пути к файлу, либо в виде файлового дескриптора

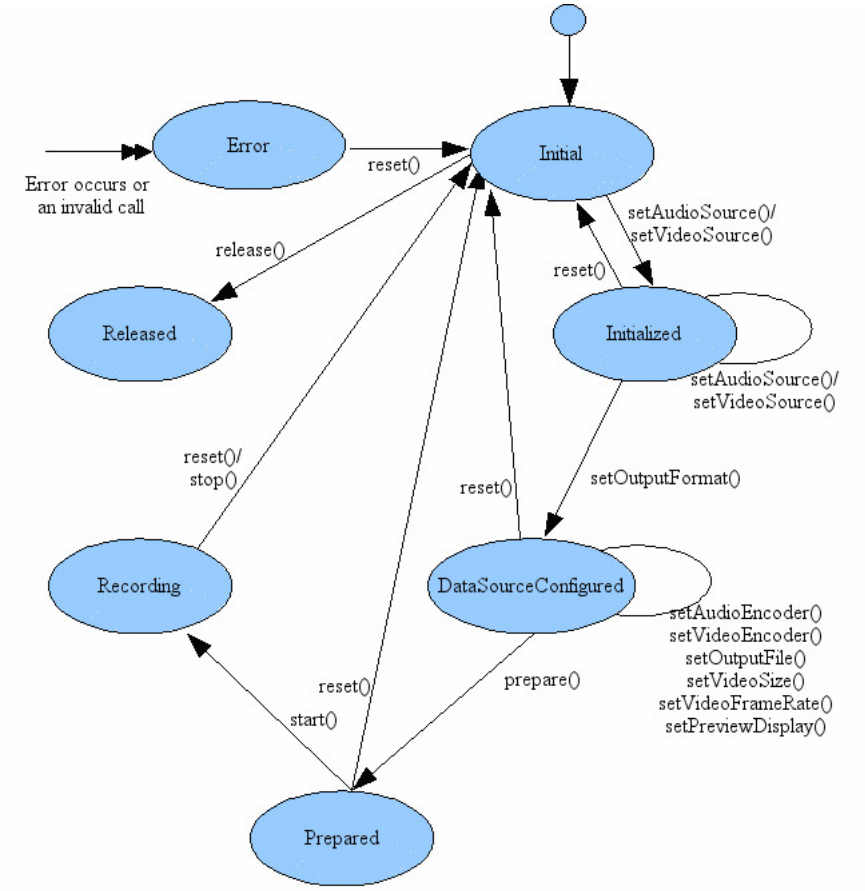
MediaRecorder.setAudioEncoder — то, как аудипоток будет закодирован в выходном файле. Если метод не будет вызван совсем, то файл в итоге окажется пустым. Возможные варианты

MediaRecorder.prepare() — подготовка к записи. Нужно вызвать, когда мы закончили с настройкой, чтобы при записи все параметры, заданные предыдущими методами, были учтены

Для начала записи вызывается метод `MediaRecorder.start()`, для её завершения — `MediaRecorder.stop()`.

1.1.3. Замечания

1. `MediaRecorder` устроен как конечный автомат, а его методы как-то переводят экземпляр класса из одних состояний в другие, поэтому важно понимать, когда и какие методы можно вызывать. Вот диаграмма состояний:



MediaRecorder state diagram

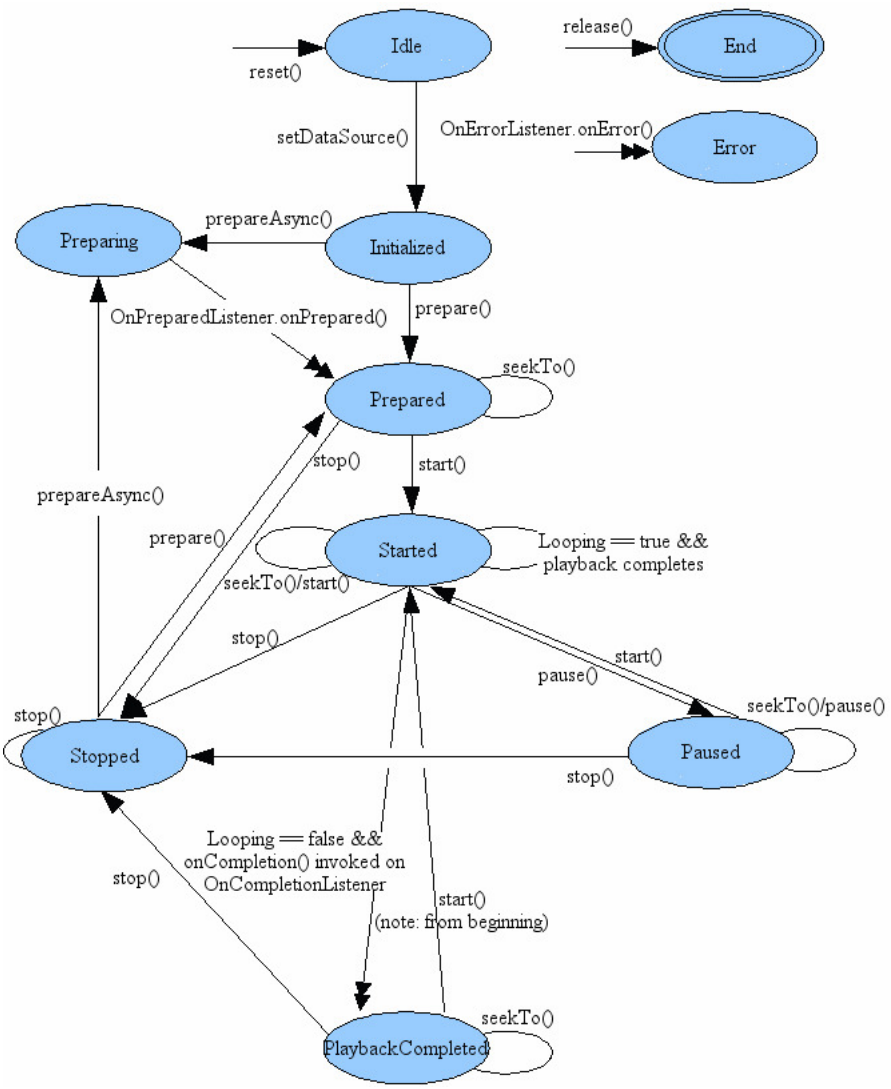
2. Также считается хорошим тоном вызывать `MediaRecorder.release()`, когда он уже не нужен. Этот метод освобождает ресурсы, занятые нашим экземпляром `MediaRecorder` в текущий момент. В противном случае у нас могут возникнуть проблемы с производительностью или вообще исключение, например, если устройство не поддерживает одновременную работу нескольких экземпляров одного кодека.

1.2. Проигрывание

В отличие от записи, дополнительных разрешений нам не нужно (если мы только не хотим, скажем, проигрывать аудиопоток из Интернета). Поэтому сразу перейдем к следующему этапу. Для воспроизведения можно использовать `android.media.MediaPlayer`.
Работа с ним в целом аналогична таковой с `android.media.MediaRecorder`: мы все так же создаем экземпляр, задаем ему источник с помощью `MediaPlayer.setDataSource` (лучше почитать по ссылке и выбрать наиболее адекватный способ — там много специализаций), подготавливаем его, используя `MediaPlayer.prepare()`, и воспроизводим при помощи `MediaPlayer.start()`.

1.2.1. Замечания

1. Структура аналогична — это все тот же автомат и поэтому нужно следить, из каких состояний и что мы вызываем:



2. `MediaPlayer.prepare()` лучше никогда не пользоваться в основном потоке, особенно, если аудио подгружается из Сети. Почему: метод загружает все ресурсы, необходимые для работы, что может создать неплохие такие тормоза, если не запускать его асинхронно (можно написать как-то по-своему это обрабатывать в отдельных потоках и использовать именно `MediaPlayer.prepare()`, а можно воспользоваться встроенным `MediaPlayer.prepareAsync()`)

3. Воспроизведение можно поставить на паузу: для этого есть `MediaPlayer.pause()`

4. Можно перематывать при помощи `MediaPlayer.seekTo(msec)` на `msec` миллисекунд от начала

5. Опять же, не забываем про `MediaPlayer.release`!

1.3. Пример

Чтобы не копировать стену текста, я просто оставлю это здесь: https://developer.android.com/guide/topics/media/audio-capture.html#example
По ссылке приведен пример записи и воспроизведения звука.

Глава #2
Камера
17 октября

2.1. AndroidManifest.xml

Разрешения:

Камера — нужно, если мы не используем какое-то внешнее приложение с уже имеющимся разрешением

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

Память — нужно, если мы хотим записывать изображенияя/видео на внешние носители (SD карта)

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

Звук — если хотим снимать со звуком

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

Geo tagging — если хотим добавлять к изображениям место съемки

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

Также следует учесть, что не любая камера умеет в, скажем, автофокус, поэтому необходимость наличия подобных способностей (features) нужно тоже заявлять.
Список поддерживаемых возможностей устройств.

Наличие камеры :

```
1 <uses-feature android:name="android.hardware.camera" />
```

Пример объявления функции, которая не является жизненно необходимой :

```
1 <uses-feature android:name="android.hardware.camera" android:required="false
    " />
```

Также необходимо для geo tagging'а в API $\geqslant$ 21 :

```
1 <uses-feature android:name="android.hardware.location.gps" />
```

Далее будут описан примерный алгоритм действий для разных версий API.

2.2. android.hardware.Camera

С появлением API 21 класс `Camera` был объявлен устаревшим (deprecated), так что сейчас его стоит использовать, разве что, для обеспечения обратной совместимости с API < 21.

Последовательность действий:

1. Получить рабочий экземпляр `Camera`. Используем `Camera.open()` — получим первую попавшуюся (в большинстве случаев это задняя камера) или ошибку, если камеры в устройстве нет. Также можно указать числовой параметр, если мы знаем `cameraId` той камеры, которую хотим использовать
2. Получим её параметры. `Camera.getParameters()`
3. Возможно, мы хотим их поменять. Тогда меняем полученный объект и вызываем `Camera.setParameters` с ним в качестве аргумента
4. Настроим ориентацию: `Camera.setDisplayOrientation`. Аргумент — угол поворота по часовой стрелке, кратный 90 градусам
5. Инициализируем `SurfaceHolder` (где отображается то, что видит камера) и задаем его камере `Camera.setSurfaceHolder`. В большинстве случаев мы будем пользоваться SurfaceView, так как реализовывать интерфейс `SurfaceHolder` самому довольно муторно
6. `Camera.startPreview` — для запуска Preview
7. `Camera.takePicture(Camera.ShutterCallback, Camera.PictureCallback, Camera.PictureCallback, Camera.PictureCallback)`, где аргументы — аргументы типа `Callback` — имеют соответствующие методы, которые вызываются, когда происходит то или иное событие. Подробнее
8. После съемки preview останавливается. Если хочется ещё поснимать, нужно вернуться к шагу 6.
9. Можно остановить его и самому, используя `Camera.stopPreview()`
10. Важно никогда не забывать делать `Camera.release()` по уже озвученным причинам. Только тут мы дополнительно еще и не будем давать никому пользоваться камерой.

2.3. android.hardware.camera2

Что поменялось:

1. Теперь для работы с камерами есть `CameraManager`. Чтобы получить его экземпляр, нужно:

```
1 CameraManager manager = (CameraManager) getSystemService(Context.
    CAMERA_SERVICE);
```

2. Камеры в нем доступны по строкам-идентификаторам. Их характеристики можно получить, вызвав `CameraManager.getCameraCharacteristics(ID)`
3. Для записи нужно создавать `CaptureSession` с использованием соответствующего метода
4. Затем создаем `CaptureRequest`, определяющий необходимые параметры
5. Столько раз, сколько нужно выполняем `capture`
6. В итоге получается `TotalCaptureResult` с информацией о камере в момент записи. Также в указанные нами `SurfaceHolder`ы отправятся снятые данные

Ссылка на описание

Глава #3
Google features
17 октября

Есть код. *Repo*

На *Link* много справок про API, они должны помочь разобраться в происходящем.

3.1. Что потребуется?

Google предоставляет API к своим сервисам, в том числе и для Android (а не просто для Java).

Фрагмент - нечто, размещённое внутри Activity, тоже имеет свой жизненный цикл и т.д.

Для работы надо установить библиотеку Google Play Services

В Andoid Studio SdkManager -> Tools можно найти, как установить.

В *Link* есть API Manager, там можно подключить (Enable API) требуемый API сервис. Вы получите ключ для работы. Разумеется, нужно залогиниться и создать/добавить проект.

Credentials - Вам предложать туда сходить. Там создаются ключи. Копируете ключ (Alza...)

В manifest.xml внутри тега applications создаётся тег meta-data, там задаём поля android:name и android:key - имя API и полученный ключ *Example*.

3.2. Google Maps

Здесь, как упоминалось на семинаре, надо запрашивать соответствующие разрешения, если у Вас относительно новая версия Android.

Code Ниже комментарии по поводу этого кода.

GoogleApiClient - класс для работы, создаём экземпляр через builder (в onCreate()). Говорим, какие callback'и будут обрабатывать события в разных ситуациях: смогли/не смогли подключиться и т.п.

У этого класса есть, разумеется, методы connect()/disconnect()/isConnected().

LocationServises.API - с его помощью можно получать местоположение.

Создадим mapFragment.

getMapAsync - метод для загрузки карты.

Метод onMapReady вызывается тогда, когда карта будет подгружена и готова, чтоб с ней работали.

Для добавления на карту маркера используем markerOptions() - аналог Builder'а.

Методом moveCamera можно задать координаты того окна, которое Вы перед собой увидите, если Вас, конечно, не устраивает, что будет открыта карта мира, а маркер будет проставлен не очень удобно.

У других гугловских штук методы аналогичны тем, что у карт. Вообще, конечно, лучше читать на соответствующей страничке, как работать с конкретным сервисом.

3.3. Датчики (Sensors)

Presentation

Repo