Android Camera API2

罗流毅

xluoly@gmail.com

2018-7-26

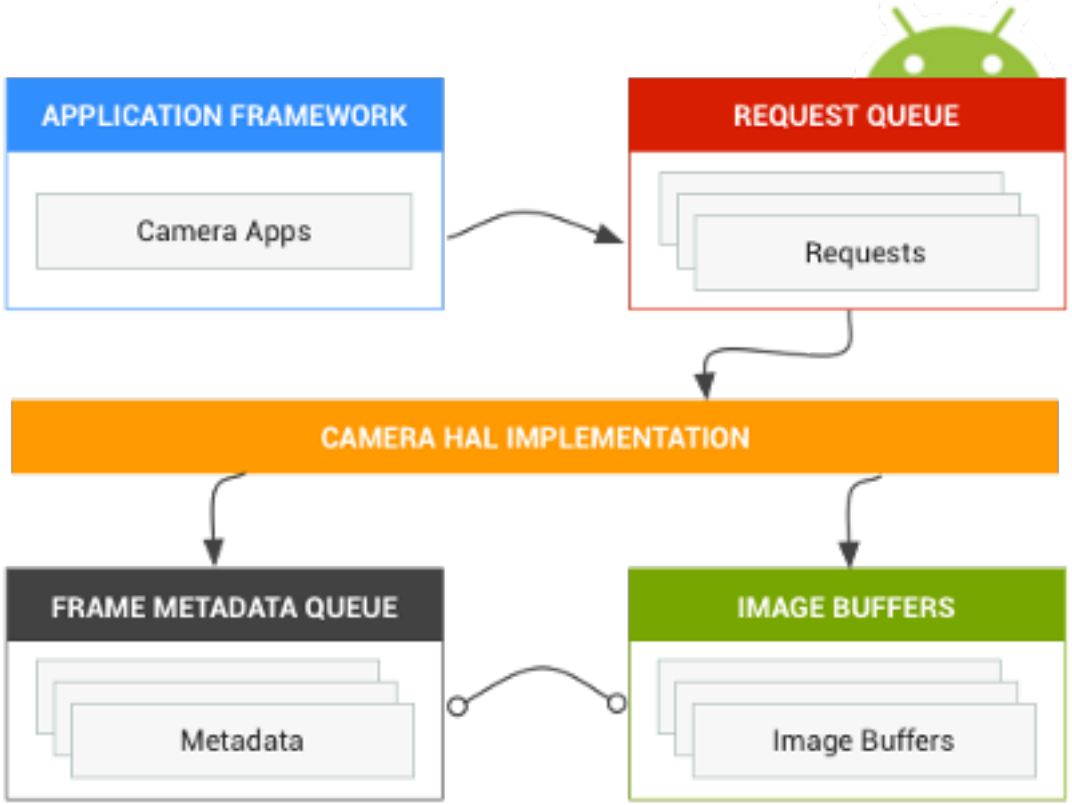
Agenda

Architecture of Camera1

android.hardware.Camera

三个工作模式:

1. Preview
2. Capture
3. Video recording

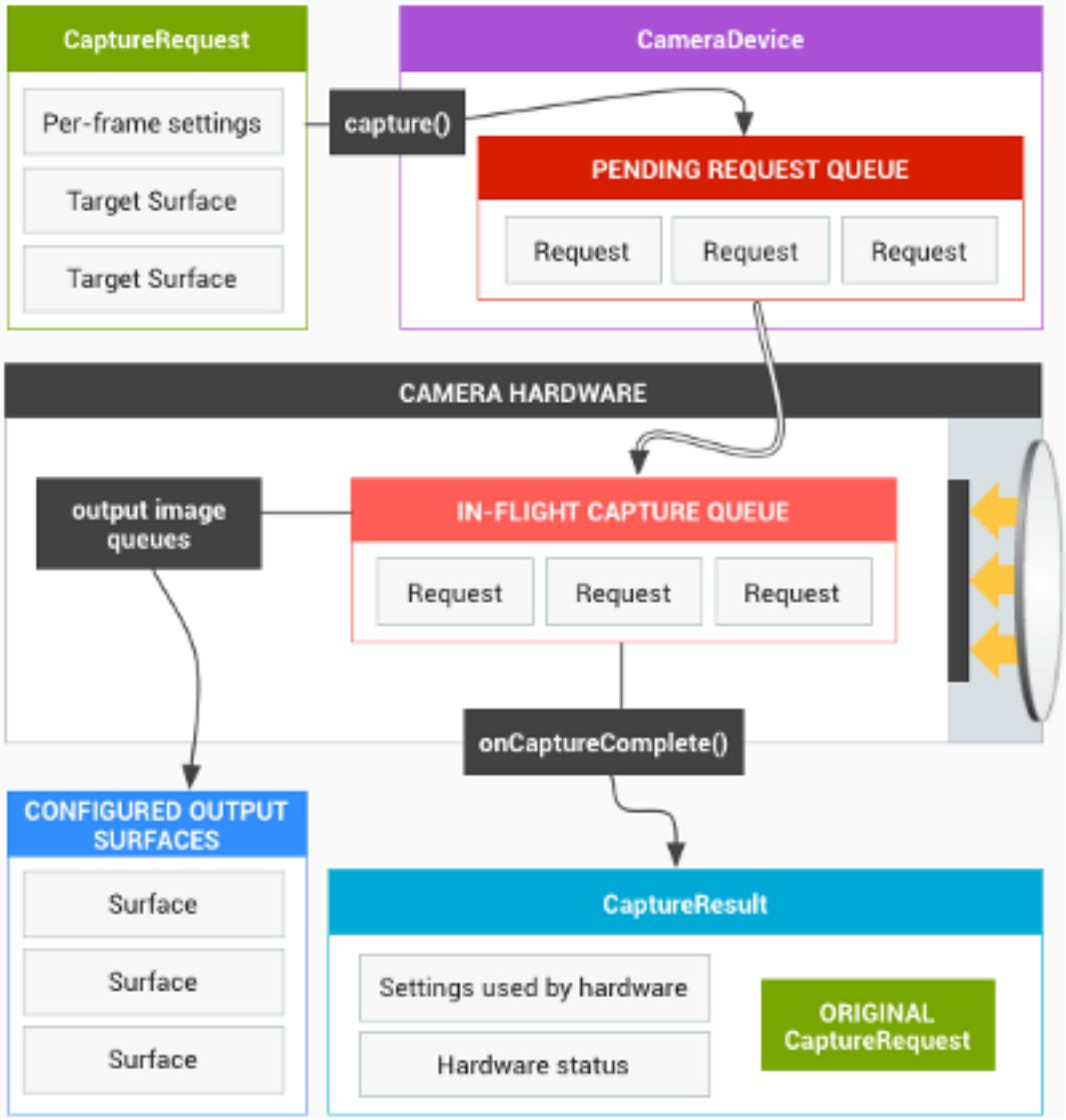


Limitations of Camera1

- 难以增加新的功能，如：快速拍照、零延迟拍照等
- 无法实现针对每帧的控制
- 无法实现 RAW 格式拍照

Architecture of Camera2

android.hardware.camera2



Features of Camera2

- 允许用户更好的控制聚焦、曝光等
- 可以对每个视频帧进行独立控制
- 可以保存 Sensor RAW data
- 更灵活的图像后期处理

Camera1 vs Camera2

Camera1

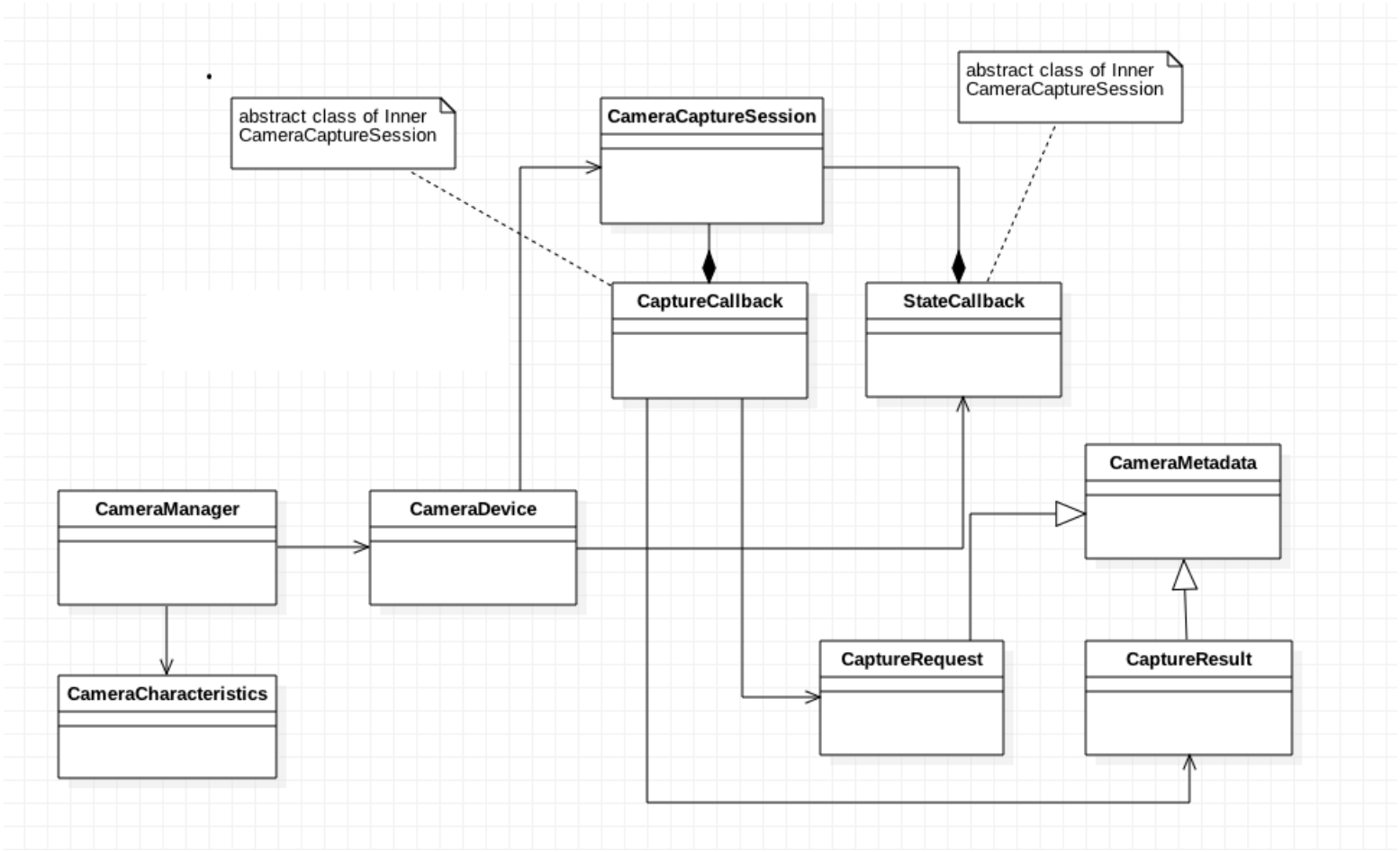
vs

Camera2

Camera2 主要的几个类

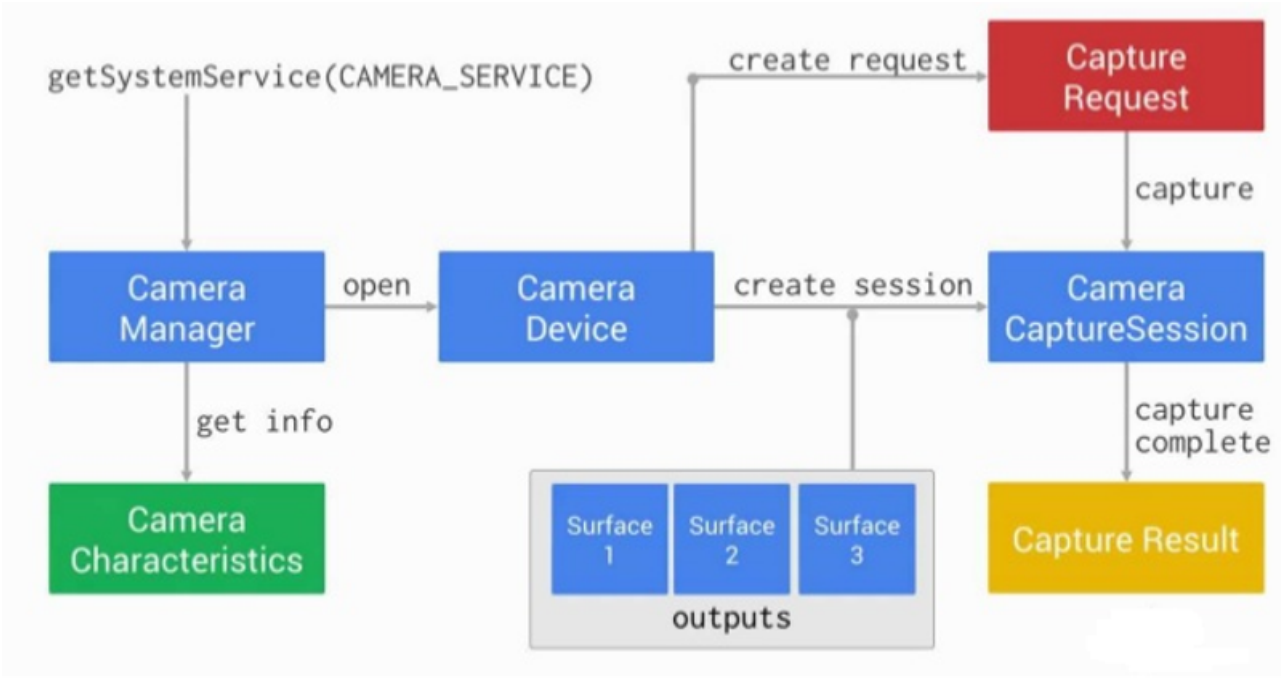
- CameraManager：最顶层的管理类，提供检测系统摄像头、打开摄像头等操作
- CameraCharacteristics：用于描述特定摄像头所支持的各种特性，通过 CameraManager 来获取
- CameraDevice：代表系统摄像头设备
- CameraCaptureSession：摄像头建立会话的类，预览、拍照和录影都要先通过它建立 Session 来实现，数据通过内部类 StateCallback 和 CaptureCallback 返回
- CameraRequest 和 CameraRequest.Builder：对摄像头的设定和控制，以及拍照、预览和录像等都是通过发送请求实现

Camera2 主要的几个类

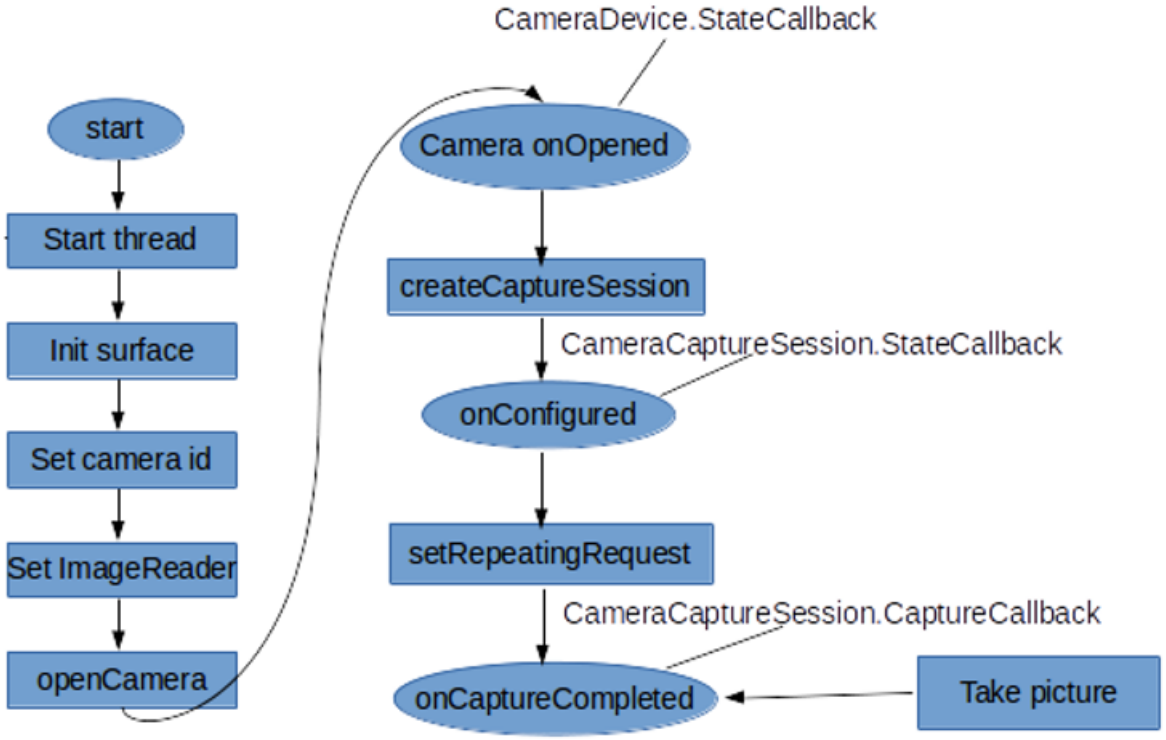


Agenda

框图


getSystemService(CAMERA_SERVICE)
create request
Capture Request
capture
Camera Manager
open
Camera Device
create session
Camera CaptureSession
get info
capture complete
Camera Characteristics
Surface 1
Surface 2
Surface 3
outputs
Capture Result

拍照的流程


CameraDevice.StateCallback
start
Start thread
Init surface
Set camera id
Set ImageReader
openCamera
Camera onOpened
createCaptureSession
CameraCaptureSession.StateCallback
onConfigured
setRepeatingRequest
CameraCaptureSession.CaptureCallback
onCaptureCompleted
Take picture

获取 CameraManager

```
mCameraManager = (CameraManager)
                 activity.getSystemService(Context.CAMERA_SERVICE);
```

查询摄像头

```
private String getCameraId(CAMERA camera) {
    int lensFacing = (camera == CAMERA.EXT) ?
            CameraCharacteristics.LENS_FACING_FRONT :
            CameraCharacteristics.LENS_FACING_BACK;
    try {
        for (String cameraId : mCameraManager.getCameraIdList()) {
            CameraCharacteristics characteristics
                    = mCameraManager.getCameraCharacteristics(cameraId);

        if (characteristics.get(CameraCharacteristics.LENS_FACING) == lensFacing
                return cameraId;
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return "";
}
```

打开摄像头设备

```
String cameraId = getCameraId(camera);
mCameraManager.openCamera(cameraId, mDeviceStateCallback, mBackgroundHandler);
```

建立 Session

```
SurfaceTexture texture = mTextureView.getSurfaceTexture();
texture.setDefaultBufferSize(mPreviewSize.getWidth(), mPreviewSize.getHeight())
Surface surface = new Surface(texture);
mPreviewRequestBuilder = mCameraDevice.createCaptureRequest(
                            CameraDevice.TEMPLATE_PREVIEW);
mPreviewRequestBuilder.addTarget(surface);
mCameraDevice.createCaptureSession(Arrays.asList(surface, mImageReader.getSurfa
        new CameraCaptureSession.StateCallback() {
            @Override
           public void onConfigured(@NonNull CameraCaptureSession session) {
                mPreviewSession = session;
                updatePreview();
            }
            @Override
        public void onConfigureFailed(@NonNull CameraCaptureSession session) {
            }
        }, mBackgroundHandler);
```

请求预览

```
private void updatePreview() {
    ...
    mPreviewRequestBuilder.set(CaptureRequest.CONTROL_AF_MODE,
            CaptureRequest.CONTROL_AF_MODE_CONTINUOUS_PICTURE);
    setAutoFlash(mPreviewRequestBuilder);
    mPreviewRequest = mPreviewRequestBuilder.build();
    mCaptureSession.setRepeatingRequest(mPreviewRequest,
            mCaptureCallback, mBackgroundHandler);
    ...
}
```

Take Picture

```
final CaptureRequest.Builder captureBuilder =
     mCameraDevice.createCaptureRequest(CameraDevice.TEMPLATE_STILL_CAPTURE);
captureBuilder.addTarget(mImageReader.getSurface());

captureBuilder.set(CaptureRequest.CONTROL_AF_MODE,
        CaptureRequest.CONTROL_AF_MODE_CONTINUOUS_PICTURE);
setAutoFlash(captureBuilder);

int rotation = activity.getWindowManager().getDefaultDisplay().getRotation();
captureBuilder.set(CaptureRequest.JPEG_ORIENTATION, getOrientation(rotation));

CameraCaptureSession.CaptureCallback CaptureCallback
        = new CameraCaptureSession.CaptureCallback() {

    @Override
    public void onCaptureCompleted(@NonNull CameraCaptureSession session,
                                   @NonNull CaptureRequest request,
                                   @NonNull TotalCaptureResult result) {
    }
};

mCaptureSession.stopRepeating();
```

保存 Image

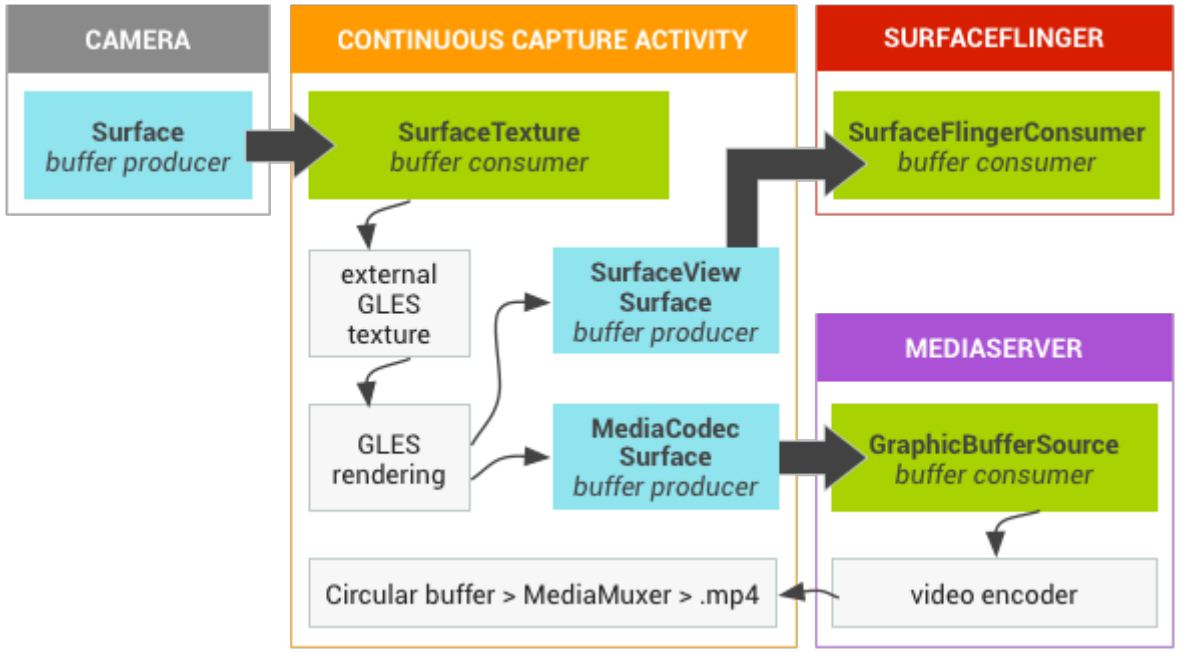
```
private final ImageReader.OnImageAvailableListener mOnImageAvailableListener
        = new ImageReader.OnImageAvailableListener() {

    @Override
    public void onImageAvailable(ImageReader reader) {
     mBackgroundHandler.post(new ImageSaver(reader.acquireNextImage(), mFile));
    }

};
```

Agenda

流程

- 打开 Camera 设备，与拍照的过程一样
- 设置参数，建立 MediaRecorder
- 从获取 MediaRecorder 的 input surface，建立 Capture Session
- Session 发送 repeating request 获取视频
- start MediaRecorder

系统框图


CAMERA
Surface
buffer producer
CONTINUOUS CAPTURE ACTIVITY
SurfaceTexture
buffer consumer
external
GLES
texture
GLES
rendering
SurfaceView
Surface
buffer producer
MediaCodec
Surface
buffer producer
Circular buffer > MediaMuxer > .mp4
SURFACEFLINGER
SurfaceFlingerConsumer
buffer consumer
MEDIASERVER
GraphicBufferSource
buffer consumer
video encoder

设置 MediaRecorder

```
mMediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
mMediaRecorder.setVideoSource(MediaRecorder.VideoSource.SURFACE);
mMediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
mMediaRecorder.setOutputFile(mNextVideoAbsolutePath);
mMediaRecorder.setVideoEncodingBitRate(10000000);
mMediaRecorder.setVideoFrameRate(30);
mMediaRecorder.setVideoSize(mVideoSize.getWidth(), mVideoSize.getHeight());
mMediaRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.H264);
mMediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AAC);
int rotation = activity.getWindowManager().getDefaultDisplay().getRotation();
switch (mSensorOrientation) {
    case SENSOR_ORIENTATION_DEFAULT_DEGREES:
      mMediaRecorder.setOrientationHint(DEFAULT_ORIENTATIONS.get(rotation));
        break;
    case SENSOR_ORIENTATION_INVERSE_DEGREES:
      mMediaRecorder.setOrientationHint(INVERSE_ORIENTATIONS.get(rotation));
        break;
}
mMediaRecorder.prepare();
```

建立 Session

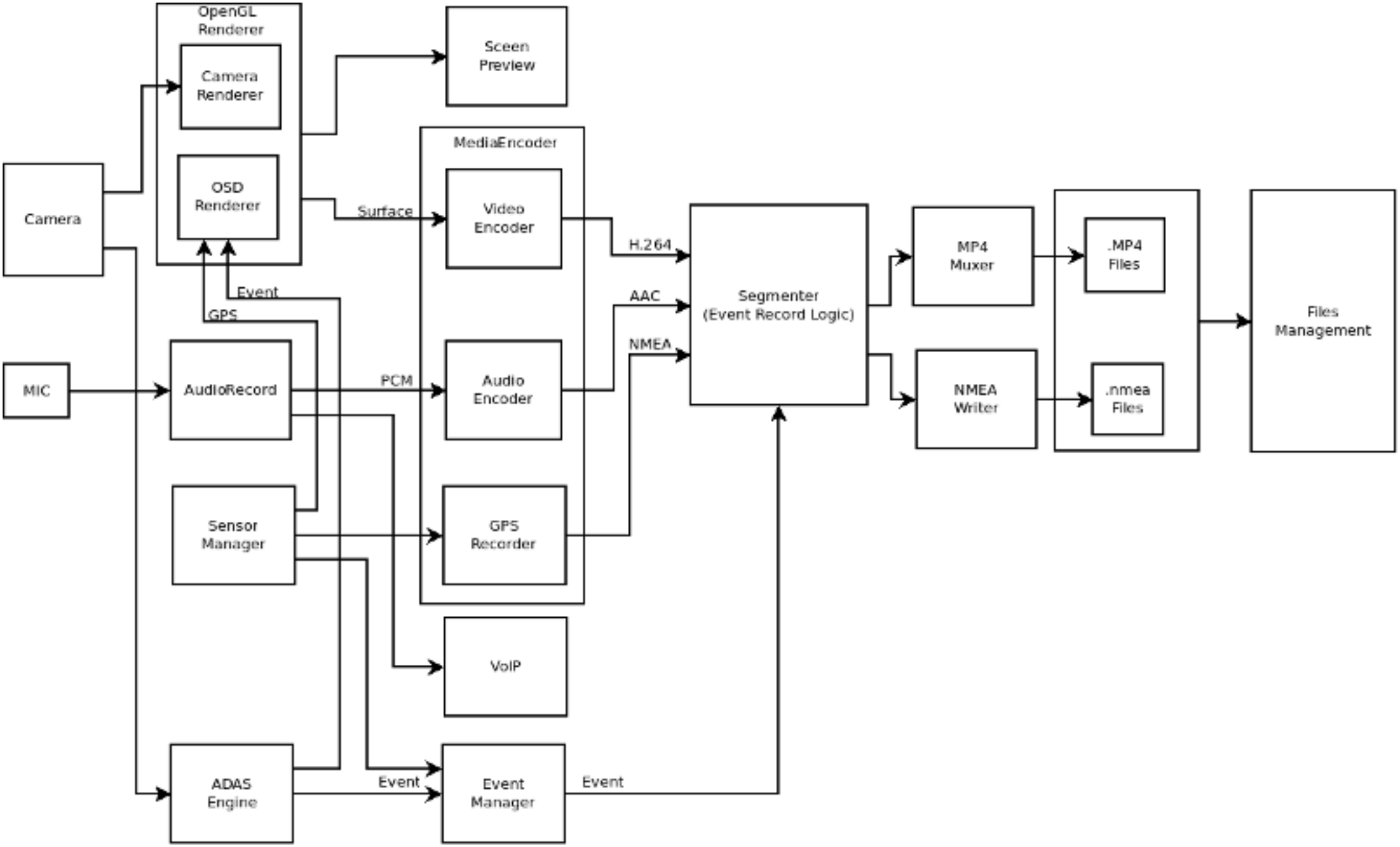
```
SurfaceTexture texture = mTextureView.getSurfaceTexture();
texture.setDefaultBufferSize(mPreviewSize.getWidth(), mPreviewSize.getHeight())
mPreviewBuilder = mCameraDevice.createCaptureRequest(CameraDevice.TEMPLATE_RECOR
List<Surface> surfaces = new ArrayList<>();
Surface previewSurface = new Surface(texture);
surfaces.add(previewSurface);
mPreviewBuilder.addTarget(previewSurface);
Surface recorderSurface = mMediaRecorder.getSurface();
surfaces.add(recorderSurface);
mPreviewBuilder.addTarget(recorderSurface);
mCameraDevice.createCaptureSession(surfaces, new CameraCaptureSession.StateCallb
    @Override
   public void onConfigured(@NonNull CameraCaptureSession cameraCaptureSession) {
        mPreviewSession = cameraCaptureSession;
        updatePreview();
        getActivity().runOnUiThread(new Runnable() {
            @Override
            public void run() {
                mMediaRecorder.start();
            }
        });
    }, mBackgroundHandler);
```

Agenda

流程

- 实例化 MediaCodec 作为 H.264 encoder，获取 input surface
- 通过 OpenGL 创建一个 surface 用于接收 camera 的图像输出
- 创建 capture session，传入 OpenGL surface
- 发送 repeating request 获取连续的视频流
- OpenGL 将 camera 输出的图像 texture 渲染到 MediaCodec input surface
- MediaCodec 对 input surface 进行编码，输出 H.264 数据流

系统框图


OpenGL
Renderer
Camera
Renderer
OSD
Renderer
Sceen
Preview
MediaEncoder
Camera
Surface
Video
Encoder
H.264
Segmenter
(Event Record Logic)
MP4
Muxer
.MP4
Files
Files
Management
Event
GPS
AAC
NMEA
MIC
AudioRecord
PCM
Audio
Encoder
NMEA
Writer
.nmea
Files
Sensor
Manager
GPS
Recorder
VoIP
ADAS
Engine
Event
Event
Manager
Event

创建 OpenGL surface

```
int texture = GLDrawer2D.initTex();
mInputSurface = new SurfaceTexture(texture);
mInputSurface.setDefaultBufferSize(1920, 1080);
mInputSurface.setOnFrameAvailableListener(EGLRenderer.this);
```

建立 Capture Session

```
Surface surface = new Surface(surfaceTexture);
mCaptureBuilder = mCameraDevice.createCaptureRequest(CameraDevice.TEMPLATE_RECOR
mCaptureBuilder.addTarget(surface);
mCameraDevice.createCaptureSession(Collections.singletonList(surface),
        new CameraCaptureSession.StateCallback() {
    @Override
   public void onConfigured(@NonNull CameraCaptureSession cameraCaptureSession) {
        mCaptureSession = cameraCaptureSession;
        updatePreview();
    }
    @Override
   public void onConfigureFailed(@NonNull CameraCaptureSession cameraCaptureSessi
    }
}, mBackgroundHandler);
```

渲染图像

```
@Override //OnFrameAvailableListener
public void onFrameAvailable(SurfaceTexture surfaceTexture) {
    mRenderHandler.sendEmptyMessage(MSG_UPDATE_FRAME);
}

private void drawFrame() {
    mInputSurface.updateTexImage();
    mInputSurface.getTransformMatrix(mTmpMatrix);
    mTextureController.setMatrix(mTmpMatrix);
    mEncoderSurface.makeCurrent();
    GLES20.glViewport(0, 0, 1920, 1080);
    mTextureController.draw();
    mEncoderSurface.setPresentationTime(mInputSurface.getTimestamp());
    if (mGroupOsd != null) {
        mGroupOsd.draw();
    }
    if (mFrameListener != null) {
        mFrameListener.frameAvailableSoon();
    }
    mEncoderSurface.swapBuffers();
}
```

与 MediaRecorder 录影的差异

- MediaRecorder：将 MediaRecorder input surface 传给 Camera，图像数据直接输出到 MediaRecorder surface
- MediaCodec：需要借助 OpenGL 渲染，必须将 camera 图像数据输出到 OpenGL 创建的一个中间 SurfaceTexture，再用 OpenGL 将 Texture 渲染到 MediaCodec input surface

Camera2 与 Camera1 的使用差异

	Camera1	Camera2
读取参数	Camera.getParameters()	CameraManager.getCameraCharacteristics(cameraId) characteristics.get(KEY)
设置参数	params.setPreviewSize(width,height) Camera.setParameters(params)	CaptureRequest.Builder.set(Key key, T value)
打开设备	Camera.open(CameraID)	CameraManager.openCamera(CameraId,CameraDevice.StateCallback,Handler)

启动预览	Camera.startPreview()	CaptureReqBuilder = camera.createCaptureRequest(CameraDevice.TEMPLATE_PREVIEW);
		CaptureReqBuilder.addTarget(Surface);
		session = Camera.createCaptureSession(...);
		session.setRepeatingRequest(CaptureBuilder.build(), ...);

图像方向	Camera.setDisplayOrientation(degrees)	没有直接设置预览方向的方法，通过设置预览View的转换矩阵实现
		matrix.postRotate(90 * (rotation - 2), centerX, centerY);
		mTextureView.setTransform(matrix);

原始数据	Camera.PreviewCallback中onPreviewFrame(byte[],Camera)	没有直接的Callback，通过ImageReader间接获得 注册ImageReader input surface给camera输出，然后从Image中读取数据

参考资料

- googlesamples/android-Camera2Basic
- googlesamples/android-Camera2Video

The End

Thank you!