CAMERA

2020년도 1학기 앱 프로그래밍 13주차 수업

목 차

1~3교시

2020년도 1학기 앱 프로그래밍 – TAKE PICTURE & CAMERA FUNCTIONS

1. FULL SCREEN MODE

```
private void hideSystemUI() {
    View decorView = getWindow().getDecorView();
    decorView.setSystemUiVisibility(
            View.SYSTEM_UI_FLAG_IMMERSIVE
                    | View.SYSTEM_UI_FLAG_LAYOUT_STABLE
                    | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
                    | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
                    | View.SYSTEM_UI_FLAG_HIDE_NAVIGATION
                    | View.SYSTEM_UI_FLAG_FULLSCREEN);
}
```

2. TAKE PICTURE

- 사진 촬영(캡처)
  - 사전 작업
    - 미리보기 클래스 구성 → SurfaceView & SufaceHolder.Callback
    - 사진 영상 표시를 위한 뷰 레이아웃 구성
      - FramLayout → FreeView
    - 위 두 작업이 완료되면 사진 이미지 캡처 가능
    - 사진 캡처 버튼에 리스너 등록
  - 사진 촬영
    - 사진 촬영 리스너에 Camera.takePicture() 메서드 사용
    - JPEG 형식의 이미지 데이터 수신을 위해 Camera.PictureCallback 인터페이스 구현 필요
      - 데이터 수신 후 파일로 작성

2. TAKE PICTURE (계속)

- 캡처 버튼 결합과 리스너 등록
  - takePicture() 메서드를 호출하여 이미지 캡처 트리거 수행
  - 트리거 :“특정한 작동을 시작하기 위한 계기를 부여하는 입력”

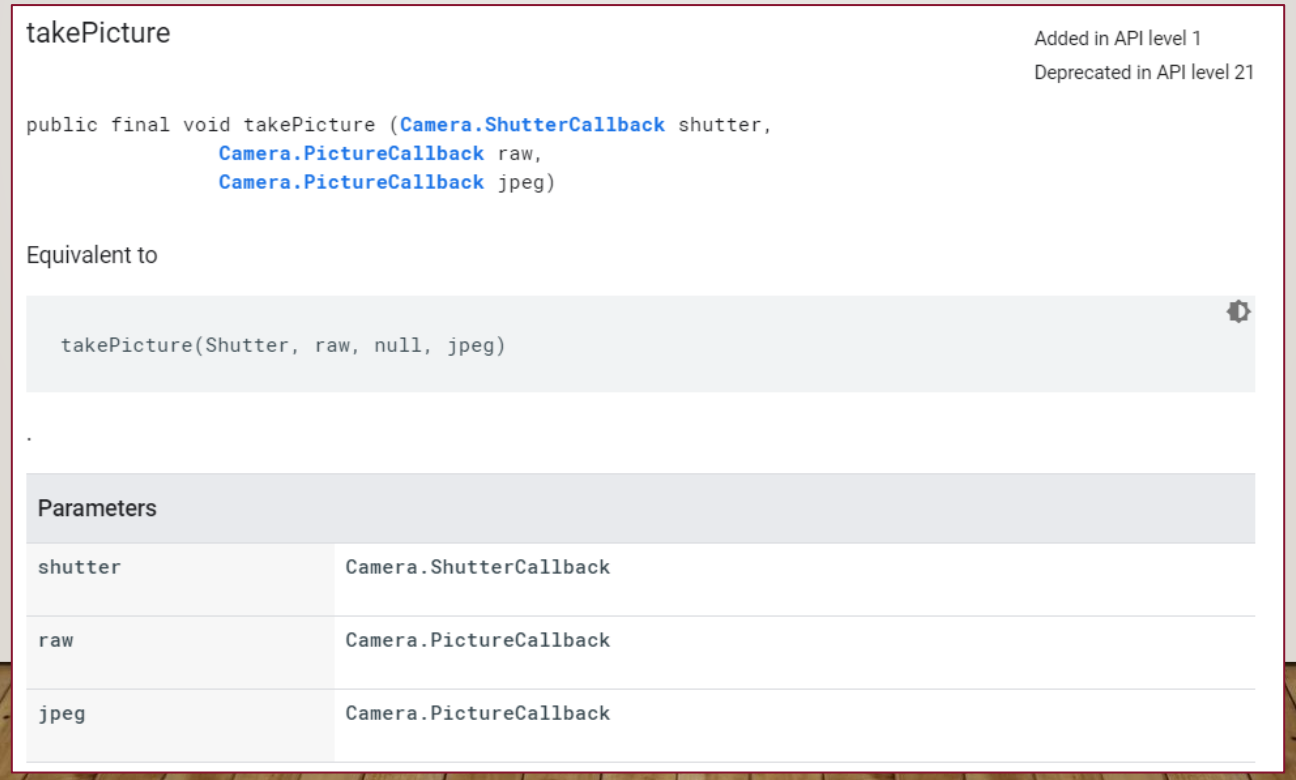
```
btn_capture = findViewById(R.id.btn_capture);
btn_capture.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // 카메라로 부터 이미지 얻기!
        mCamera.takePicture( shutter: null, raw: null, mPicture);
    }
});
```

- Camera 해제
  - Camera 객체 사용 완료 후 객체 해제 필요

```
@Override
protected void onPause() {
    super.onPause();
    mCamera.release();
}
```

2. TAKE PICTURE (계속)

- takePicture()
  - shutter : 이미지 캡처 순간을 위한 콜백 또는 null
  - raw : raw(압축 없는 원본) 이미지 데이터 콜백 또는 null
  - jpeg : JPEG 이미지 데이터 콜백 또는 null



takePicture

Added in API level 1

Deprecated in API level 21

```
public final void takePicture (Camera.ShutterCallback shutter,
                Camera.PictureCallback raw,
                Camera.PictureCallback jpeg)
```

Equivalent to

```
takePicture(Shutter, raw, null, jpeg)
```

.

Parameters	
shutter	Camera.ShutterCallback
raw	Camera.PictureCallback
jpeg	Camera.PictureCallback

3. PICTURE CALLBACK

- 카메라에서 수신한 이미지 저장을 위한 인터페이스 구현

```
// 콜백 객체
// 콜백 인터페이스 구현 후 이미지 데이터를 수신하고 파일로 작성한다.
private Camera.PictureCallback mPicture = new Camera.PictureCallback(){
    @Override
    public void onPictureTaken(byte[] data, Camera camera) {
        File pictureFile = getOutputMediaFile(MEDIA_TYPE_IMAGE);

        if(pictureFile == null){
            Log.d( tag: "MyCameraApp", msg: "미디어 파일 생성 실패, 저장소 권한을 확인하세요!");
            return;
        }

        try {
            FileOutputStream fos = new FileOutputStream(pictureFile);
            fos.write(data);
            fos.flush();
            fos.close();
            mCamera.startPreview();
        } catch (FileNotFoundException e) {
            Log.d( tag: "MyCameraApp", msg: "해당 파일이 없습니다!");
            e.printStackTrace();
        } catch (IOException e) {
            Log.d( tag: "CAMERA", msg: "파일 접근 오류!");
            e.printStackTrace();
        }
    }
};
```

4. SAVE A MEDIA FILE

- 미디어 파일 저장

```
// 이미지 또는 비디오 저장 파일 만들기 메서드
private static File getOutputMediaFile(int type){
    // 안전을 위해 SDCard가 Environment.getExternalStorageState ()를 사용하여 마운트되었는지 확인한다.
    File mediaStorageDir = new File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES), child: "");
    // 이 위치는 생성 된 이미지를 응용 프로그램간에 공유하고 앱을 제거한 후에도 유지하려는 경우에 가장 적합합니다.

    // 저장 디렉토리가 없으면 저장소 디렉토리 생성
    if (! mediaStorageDir.exists()){
        if (! mediaStorageDir.mkdirs()){
            Log.d( tag: "MyCameraApp", msg: "디렉토리 생성 실패!");
            return null;
        }
    }

    // 미디어 파일 이름 생성
    String timeStamp = new SimpleDateFormat( pattern: "yyyyMMdd_HHmmss").format(new Date());
    File mediaFile;
    if (type == MEDIA_TYPE_IMAGE){
        mediaFile = new File( pathname: mediaStorageDir.getPath() + File.separator + "IMG_"+ timeStamp + ".jpg");
    } else if(type == MEDIA_TYPE_VIDEO) {
        mediaFile = new File( pathname: mediaStorageDir.getPath() + File.separator + "VID_"+ timeStamp + ".mp4");
    } else {
        return null;
    }

    return mediaFile;
}
```

4. SAVE THE MEDIA FILE (계속)

- 미디어 파일 기본 저장 위치 확인

```
File mediaStorageDir = new File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES), child: "");
```

> ★ **참고:** Environment.getExternalStoragePublicDirectory()는 Android 2.2(API 레벨 8) 이상에서 이용할 수 있습니다. 이전 버전의 Android를 사용하는 기기를 대상으로 하는 경우, 대신 Environment.getExternalStorageDirectory()를 사용하세요. 자세한 내용은 공유된 파일 저장을 참조하세요.

- 안드로이드 10 이상의 저장소 접근 업데이트
  - sdcard에 접근하는 방법의 변경 → ContentsProvider 를 통한 접근만 허용
  - 임시로 직접 접근을 원하는 경우 매니패스트 파일에 속성 변경 [11 이상은 무시됨]
    - *[android:requestLegacyExternalStorage="true"]*
  - 관련 정보 → https://developer.android.com/preview/privacy/storage

4. SAVE THE MEDIA FILE (계속)

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.camarasample">

    <uses-permission android:name="android.permission.CAMERA"/> <!-- 카메라 기능 설치시 허가권 확인 -->
    <uses-feature android:name="android.hardware.camera"/> <!-- 카메라 기능 실행 시 접근 권한 -->
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/> <!-- SD 저장 권한 확인 -->
    <uses-permission android:name="android.permission.RECORD_AUDIO"/> <!-- 오디오 녹음 권한 -->
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/> <!-- 위치 태그를 위한 권한 -->
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<!--
    android:requestLegacyExternalStorage="true"
    안드로이드 정책 변경 - 임시 방편 그러나 사용에 문제 없음
-->
    <application
        android:requestLegacyExternalStorage="true"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="CamaraSample"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity"
            android:screenOrientation="landscape" />
        <activity android:name=".CheckPermissionActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

5. CAMERA FUNCTIONS

- 카메라 기능 → 앱에서 카메라 기능 제어 제공
  - 사진 형식, 플래시 모드, 초점 설정 등
  - Camera.Parameters 객체를 통한 접근 및 설정

- 보편적 카메라 기능
  - 우측 표에 표시
  - 미지원 기기의 기능 가용성 확인 필요

표 1. 보편적인 카메라 기능(각 기능이 도입된 Android API 레벨을 기준으로 분류).

기능	API 레벨	설명
얼굴 인식	14	사진에서 사람의 얼굴을 식별하여 초점, 측정 및 화이트밸런스 조정에 사용
측정 영역	14	이미지에서 화이트밸런스를 계산할 영역을 하나 이상 지정
초점 영역	14	이미지 내에서 초점을 맞추는 데 사용할 영역을 하나 이상 설정
White Balance Lock	14	자동 화이트밸런스 조정 중지 또는 시작
Exposure Lock	14	자동 노출 조정 중지 또는 시작
Video Snapshot	14	동영상을 촬영하는 동시에 사진 촬영(프레임 그랩)
타임랩스 동영상	11	지연이 설정된 프레임을 녹화하여 타임랩스 동영상 기록
Multiple Cameras	9	한 기기에서 두 대 이상의 카메라 지원, 전면 및 후면 카메라 포함
Focus Distance	9	카메라와 초점 내에 있는 물체 사이의 거리 보고
Zoom	8	이미지 배율 설정
Exposure Compensation	8	빛 노출 수준 증가 또는 감소
GPS Data	5	이미지에 지리적 위치 데이터 포함 또는 생략
White Balance	5	화이트밸런스 모드 설정(캡처된 이미지의 색상 값에 영향을 미침)
Focus Mode	5	카메라가 피사체에 초점을 맞추는 방식 설정(예: 자동, 고정, 매크로 또는 무한대)
Scene Mode	5	특정 사진 촬영 상황에 대한 사전 설정 모드 적용(예: 야간, 해변, 촛불을 켠 장면)
JPEG Quality	5	JPEG 이미지의 압축 수준 설정(이미지 출력 파일의 화질 및 크기 증가 또는 감소)
Flash Mode	5	플래시 켜기, 끄기 또는 자동 설정 사용
Color Effects	5	캡처된 이미지에 색상 효과 적용(예: 흑백, 세피아 톤 또는 네거티브)
Anti-Banding	5	JPEG 압축에 의한 색 그라데이션의 줄무늬 효과 완화
Picture Format	1	사진의 파일 형식 지정
Picture Size	1	저장된 사진의 픽셀 크기 지정

5. CAMERA FUNCTIONS (계속)

- 기능 가용성 확인
  - 카메라 기능이 지원되지 않는 기기에 대한 대응 요구
  - 특정 기능을 지원하는 경우라도 지원 수준 또는 옵션의 차이 존재
    - 카메라 기능을 갖는 앱 제작 시 지원 기능의 수준 결정이 요구됨
    - 기능 미지원 기기에 대한 대응 코드 작성 필요

  - 코드 작업
    - Camera 클래스의 Parameters 객체 인스턴스를 얻어 관련 메서드로 설정
      - 메서드 → Camera.getParameters()
    - 파라미터 : 카메라 기능(메서드) 및 상수를 정의한 객체
      - 메서드 → 파라미터에 정의된 기능 관련 확인 메서드
      - 상수 → 카메라 기능을 정의한 문자열 상수
      - https://developer.android.com/reference/android/hardware/Camera.Parameters

5. CAMERA FUNCTIONS (계속)

- 카메라 자동 초점 기능의 확인 예
  - 메서드
    - Camera.Parameters.getSupportedFocusModes() → String 타입의 List 반환
    - List.contains() → 괄호 사이의 문자열 포함 확인
  - 상수
    - Camera.Parameters 객체에 정의된 문자열 상수
      - Camera.Parameters.FOCUS_MODE_AUTO

```
private boolean checkAutoFocusFunction(){
    boolean isOk = false;

    Camera.Parameters params = mCamera.getParameters();
    List<String> focusModes = params.getSupportedFocusModes();

    if(focusModes.contains(Camera.Parameters.FOCUS_MODE_AUTO))
        isOk = true;

    return isOk;
}
```

5. CAMERA FUNCTIONS (계속)

- 카메라 자동 초점 기능 사용 설정
  - 일반적으로 카메라의 기능 설정은 다음의 과정을 통해 적용이 가능함
    1. Camera 객체 인스턴스 얻기
    2. getParameters() 메서드로 파라미터 얻기
    3. 파라미터 객체에 기능 설정 적용
    4. Camera 객체에 파라미터 적용

```
private void setAutoFocusFunction(){
    // 카메라 파라미터 얻기
    Camera.Parameters params = mCamera.getParameters();
    // 파라미터에 포커스 모드 설정
    params.setFocusMode(Camera.Parameters.FOCUS_MODE_AUTO);
    // 카메라 파라미터 설정
    mCamera.setParameters(params);
}
```

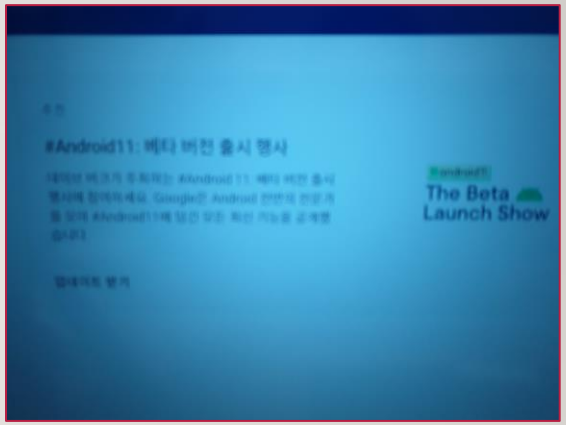
  - 위 과정을 통해 대부분의 기능 변경에 적용 가능
    - 파라미터에 대한 변경 사항은 카메라 미리보기에서 즉시 적용되어 표시됨
    - 그러나 자동 초점의 경우 추가 코드 작업이 요구됨

5. CAMERA FUNCTIONS (계속)

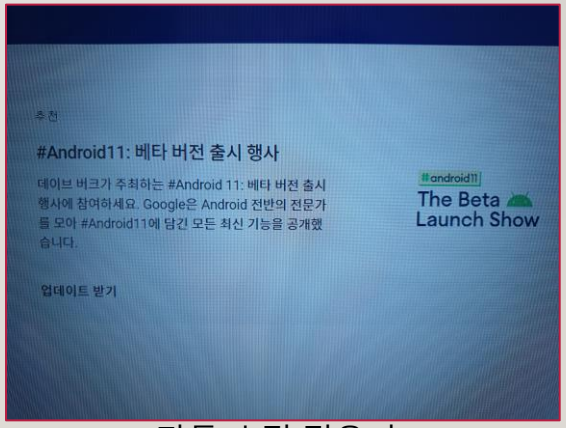
- 카메라 자동 초점의 추가 코드
  - 앞선 작업을 수행해도 자동 초점 기능 동작 안함

- 자동 초점 적용 방법
  - 촬영 시점 적용 방법
    - 사진 촬영 시 초점이 맞추어 지고 사진 촬영

  - 미리보기 적용 방법
    - 미리보기에 자동 초점을 설정하고 사진 촬영



[자동 초점 적용 전]



[자동 초점 적용 후]

5. CAMERA FUNCTIONS (계속)

- 촬영 시점 적용 방법
  - 촬영 버튼 리스너에 autofocus() 메서드를 통한 설정
    - 매개변수로 Camera.AutoFocusCallback 객체 전달
    - onAutoFocus() 메서드에서 Camera.takePicture() 메서드 호출
      - 초점 설정 성공시 매개변수 success 에 boolean 타입의 결과 확인후 촬영

```
btn_capture = findViewById(R.id.btn_capture);
btn_capture.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mCamera.autoFocus(new Camera.AutoFocusCallback() {
            @Override
            public void onAutoFocus(boolean success, Camera camera) {
                if(success) {
                    // 카메라로 부터 이미지 얻기!
                    mCamera.takePicture( shutter: null,  raw: null, mPicture);
                }
            }
        });
    }
});
```

5. CAMERA FUNCTIONS (계속)

- 미리보기 적용 방법
  - 미리보기는 동영상
    - 사진 촬영의 정지 영상이 아닌 비디오 형태의 동영상
  - Camera.Parameters 의 포커스 모드 변경
    - FOCUS_MODE_CONTINUOUS_VIDEO로 설정
    - 미리 보기 시작 전 카메라 인스턴스에 설정 적용 필요
    - FOCUS_MODE_CONTINUOUS_PICTURE 의 경우 정지 영상을 위한 파라미터
      - VIDEO 보다 초점 변경 속도측면에서 보다 적극적
      - 그러나VIDEO가 조금 더 부드럽게 초점 변경 수행

```
private void setAutoFocusPreview(){
    Camera.Parameters parameters = mCamera.getParameters();
    if (parameters.getSupportedFocusModes().contains(Camera.Parameters.FOCUS_MODE_CONTINUOUS_VIDEO)) {
        parameters.setFocusMode(Camera.Parameters.FOCUS_MODE_CONTINUOUS_VIDEO);
    }
    mCamera.setParameters(parameters);
}
```

6. 앱 프로그래밍 학습을 위한 실습 (과제)

- 실습 및 예습 문제
  - 카메라 자동 초점 기능이 적용된 사진 촬영 앱 구현
  - 카메라 후레쉬 기능 확인하여 켜는 기능 구현 → 자동 모드 아님
    - 안드로이드 개발자 사이트에서 카메라 파라미터를 확인하여 스스로 구현한다.
    - 레이아웃에 플레쉬를 켜고 끄는 스위치 위젯을 추가하여 구현한다. → 켬 / 끔

- 제출 방법
  - http://ctl.gtec.ac.kr 의 과제 제출란에 제출
  - Java 파일과 xml 파일을 제출한다.
  - 압축파일은 자신의 "반_학번_이름.zip" 로 한다.
  - 제출 일자와 시간을 엄수하여 제출하세요!