

ВИКОРИСТАННЯ CAMERA ДЛЯ ОТРИМАННЯ ФОТО

Детальніше: <http://developer.android.com/training/camera/photobasics.html>

Підхід Android до делегування дій іншим аплікаціям — виклик Intent для опису того, що слід виконати. Процес включає три складові: сам Intent, виклик зовнішнього Activity та деякий код для виконання додаткових дій.

Додайте до manifest.xml дозвіл на запис на зовнішній носій:

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

Наступний метод генерує стійке до колізій ім'я файла. Для подальшого використання шлях до файла зберігається у полі класу mCurrentPhotoPath:

```
String mCurrentPhotoPath;

private File createImageFile() throws IOException {
    // Create an image file name
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HH:mm:ss").format(new Date());
    String imageFileName = "JPEG_" + timeStamp + "_";
    File storageDir = Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_PICTURES);
    File image = File.createTempFile(
        imageFileName, /* prefix */
        ".jpg",       /* suffix */
        storageDir    /* directory */
    );
    // Save a file: path for use with ACTION_VIEW intents
    mCurrentPhotoPath = "file:" + image.getAbsolutePath();
    return image;
}
```

Даний метод створює файл для фото і може бути використаний для створення та виклику Intent:

```
static final int REQUEST_TAKE_PHOTO = 1;

private void dispatchTakePictureIntent() {
    Intent takePictureIntent =
        new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    // Ensure that there's a camera activity to handle the intent
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        // Create the File where the photo should go
        File photoFile = null;
        try {
```

```

        photoFile = createImageFile();
    } catch (IOException ex) {
        // Error occurred while creating the File
    }
    // Continue only if the File was successfully created
    if (photoFile != null) {
        takePictureIntent.putExtra(
            MediaStore.EXTRA_OUTPUT,
            Uri.fromFile(photoFile));
        startActivityForResult(
            takePictureIntent,
            REQUEST_TAKE_PHOTO);
    }
}
}

```

Якщо використати `MediaStore.EXTRA_OUTPUT`, отримане зображення буде збережено за вказаною адресою і жодних даних не буде передано до `onActivityResult`. Отримати доступ до зображення можна за попередньо збереженою адресою (у даному прикладі — у полі `mCurrentPhotoPath`).

Після того, як фото створено, відобразимо його у `ImageView`:

```

<ImageView
    android:id="@+id/image"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

```

```

@Override
protected void onActivityResult(
    int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_TAKE_PHOTO && resultCode == RESULT_OK) {
        Bitmap bMap =
            BitmapFactory.decodeFile(mCurrentPhotoPath.replace("file:", ""));
        mImageView.setImageBitmap(bMap);
    }
}

```

Зверніть увагу на те, що у `BitmapFactory.decodeFile()` видалено підрядок `"file:"`, попередньо доданий у `createImageFile()`. Для наведеного прикладу `"file:"` можна взагалі не включати до `createImageFile()`. Тут залишено без змін код, запропонований Google, з метою можливого використання у подальшому.

Ініціалізуйте в `OnCreate()` `ImageView` `mImageView` та викличте `dispatchTakePictureIntent()`.

Повний код `Activity` (змінить пакет на власний):

```
package com.example.student.cameraintent;
```

```
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Environment;
import android.provider.MediaStore;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ImageView;
import java.io.File;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
```

```
public class MainActivity extends AppCompatActivity {
    static final int REQUEST_TAKE_PHOTO = 1;
    ImageView mImageView;
    String mCurrentPhotoPath;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mImageView = (ImageView) findViewById(R.id.image);
        dispatchTakePictureIntent();
    }

    private void dispatchTakePictureIntent() {
        Intent takePictureIntent =
            new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        // Ensure that there's a camera activity to handle the intent
        if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
            // Create the File where the photo should go
            File photoFile = null;
            try {
                photoFile = createImageFile();
            } catch (IOException ex) {
                // Error occurred while creating the File
            }
            // Continue only if the File was successfully created
            if (photoFile != null) {
                takePictureIntent.putExtra(
                    MediaStore.EXTRA_OUTPUT,
                    Uri.fromFile(photoFile));
                startActivityForResult(
                    takePictureIntent,
                    REQUEST_TAKE_PHOTO);
            }
        }
    }
}
```

```

@Override
protected void onActivityResult(
    int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_TAKE_PHOTO && resultCode == RESULT_OK) {
        Bitmap bMap =
            BitmapFactory.decodeFile(mCurrentPhotoPath.replace("file:", ""));
        mImageView.setImageBitmap(bMap);
    }
}

private File createImageFile() throws IOException {
    // Create an image file name
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HH:mm:ss").format(new
Date());
    String imageFileName = "JPEG_" + timeStamp + "_";
    File storageDir = Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_PICTURES);
    File image = File.createTempFile(
        imageFileName, /* prefix */
        ".jpg",        /* suffix */
        storageDir      /* directory */
    );
    // Save a file: path for use with ACTION_VIEW intents
    mCurrentPhotoPath = "file:" + image.getAbsolutePath();
    return image;
}
}

```

Layout (змінить пакет на власний):

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.student.cameraintent.MainActivity">
    <ImageView
        android:id="@+id/image"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>

```

Візьміть до уваги зміни у отриманні дозволів, запроваджені у Android 6.0. Ознайомитися із отриманнями дозволів у режимі реального часу можна за посиланнями:

<http://developer.android.com/training/permissions/requesting.html>,
<http://developer.android.com/guide/topics/security/permissions.html#normal-dangerous>,
<http://developer.android.com/guide/topics/security/normal-permissions.html>.

У випадку використання Android 6+ слід виконати рекомендації, викладені за вказаними посиланнями. У гіршому випадкові — увімкнути дозволи уручну (Settings -> Apps -> Ваш додаток -> Permissions, рис. 1).

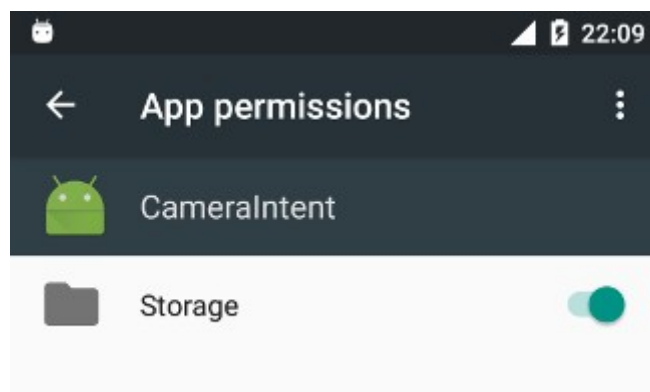


Рисунок 1 — Дозвіл на доступ до зовнішнього сховища

Залежно від моделі телефона дана програма може не коректно відображати отримане фото у ImageView. Розглянемо результати, отримані за допомогою Sony Ericsson Experia Neo V (рис. 2, Android 4.0.4) та Nexus 5 (рис. 3, Android 6.0.1).

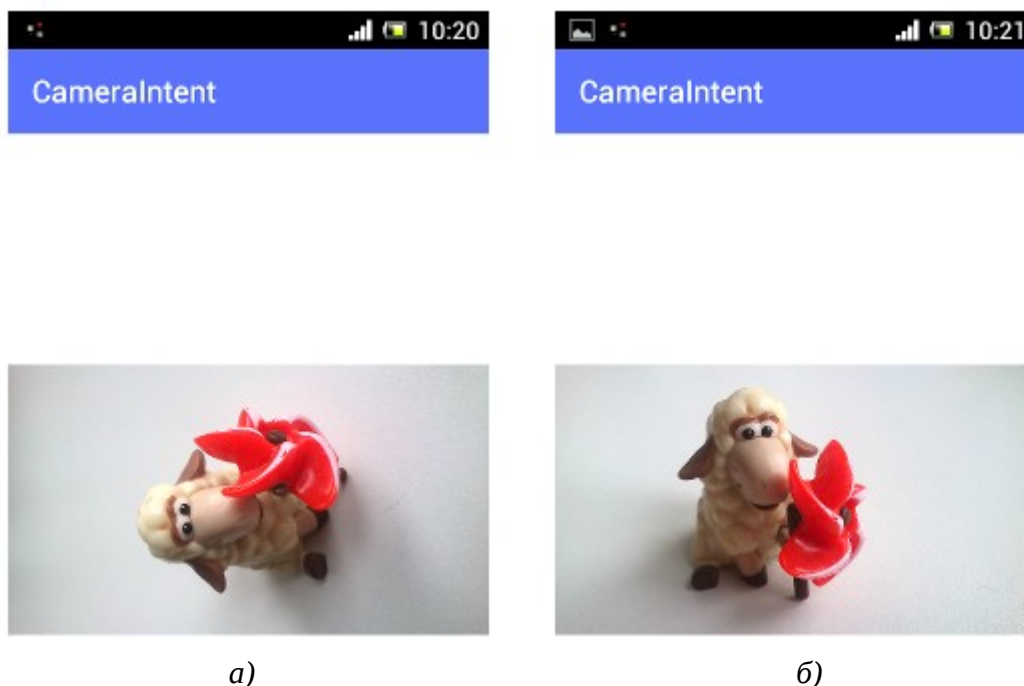


Рисунок 2 — Поворот зображення при виведенні у ImageView (Sony Ericsson Experia Neo V, Android 4.0.4): а — портретна орієнтація при фотографуванні; б — альбомна орієнтація при фотографуванні

Результат, зображений на рис. 2.а, отримано при портретній орієнтації телефона у момент фотографування, 2.б — альбомній. Як видно, фото із портретною орієнтацією повернуто при виведенні у ImageView на 90 градусів проти годинникової стрілки.

Сенсор камери є орієнтованим відносно корпусу апарата і жорстко зафіксованим. ImageView відображає Bitmap, а Bitmap не робить автоматичного розвороту зображення із використанням даних про орієнтацію сенсора у момент фотографування. Як рядки зображення було записано (зліва-направо і згори-униз) — так ImageView його і відтворить.

Існують 2 варіанти роботи із орієнтованими зображеннями. Перший полягає у збереженні порядку розташуванні рядків, отриманому при фотографуванні, і додаванні інформації про поворот у EXIF. Другий: порядок рядків у зображенні міняють одразу при отриманні зображення, не зберігаючи цієї інформації у EXIF.

На рис. 2 продемонстровано перший варіант. Розглянемо, як виконує даний код Nexus 5:

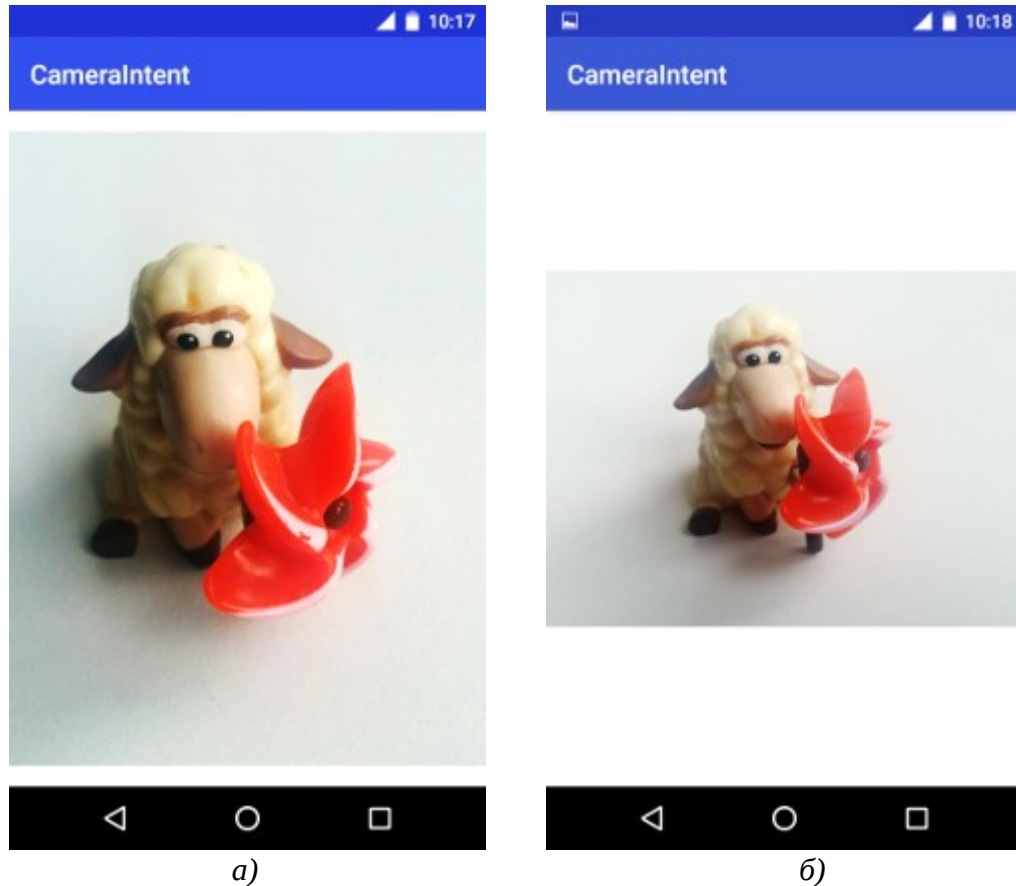


Рисунок 3 — Відсутність повороту зображення при виведенні у ImageView (Nexus 5, Android 6.0.1): а — портретна орієнтація при фотографуванні; б — альбомна орієнтація при фотографуванні

Як видно, Nexus 5 під керуванням Android 6.0.1 відображає зображення коректно, змінюючи порядок рядків одразу при збереженні у файл.

Таким чином, задача полягає в отриманні даних про орієнтацію камери і поверненні зображення при формуванні Bitmap для виведення у ImageView.

Скористаємося Gimp і переглянемо властивості фото з рис. 2.а (рис. 4). Зверніть увагу на властивість Orientation. У даному випадкові її значення “Right-Top”. Це означає, що верхній рядок даного зображення був при фотографуванні правим стовпчиком (right), а лівий стовпчик — верхнім рядком (top). Отже, зображення насправді розвернуто на 90 градусів проти годинникової стрілки. Потрібно при виведенні повернути його назад.

Скористаємося класом ExifInterface. Конструктор класу:

```
public ExifInterface (String filename)
```

Детальніше: <http://developer.android.com/reference/android/media/ExifInterface.html>



Рисунок 4 — Інформація про орієнтацію зображення (EXIF) з рис. 2.а

Для визначення повороту використаємо метод `getAttributeInt`:

```
public int getAttributeInt (String tag, int defaultValue)
```

Метод повертає ціле число, котре відповідає вказаному тегові `tag`. Якщо тег відсутній в описові JPEG-файла або його значення не може бути визнане як ціле число, метод повертає `defaultValue`.

Оскільки необхідно встановити орієнтацію зображення, скористаємося тегом

```
public static final String TAG_ORIENTATION
```

Можливі значення результату:

[ORIENTATION_ROTATE_90](#),

[ORIENTATION_ROTATE_180](#)

[ORIENTATION_ROTATE_270](#).

Встановимо значення `defaultValue`: [ORIENTATION_NORMAL](#).

Залежно від результату `getAttributeInt()` слід повернути зображення на 90, 180 чи 270 градусів.

Бітмеп зображення повернемо за допомогою класу `Matrix`. Додамо наступні методи до `Activity`:

```

private Bitmap rotateBitmap(Bitmap bMap, int degrees) {
    Matrix matrix = new Matrix();
    matrix.postRotate(degrees);
    Bitmap rotatedMatrix = Bitmap.createBitmap(
        bMap,
        0,
        0,
        bMap.getWidth(),
        bMap.getHeight(),
        matrix,
        true);
    bMap.recycle();
    return rotatedMatrix;
}

```

```

private Bitmap getBitmapRotatedIfRequired(String pathToImageToBeRotated) throws
IOException {
    Bitmap bMap =
        BitmapFactory.decodeFile(pathToImageToBeRotated.replace("file:", ""));
    ExifInterface ei = new ExifInterface(pathToImageToBeRotated.replace("file:", ""));
    int orientation = ei.getAttributeInt(ExifInterface.TAG_ORIENTATION,
    ExifInterface.ORIENTATION_NORMAL);
    switch (orientation) {
        case ExifInterface.ORIENTATION_ROTATE_90:
            return rotateBitmap(bMap, 90);
        case ExifInterface.ORIENTATION_ROTATE_180:
            return rotateBitmap(bMap, 180);
        case ExifInterface.ORIENTATION_ROTATE_270:
            return rotateBitmap(bMap, 270);
        default:
            return bMap;
    }
}

```

Модифікуємо onActivityResult():

```

@Override
protected void onActivityResult(
    int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_TAKE_PHOTO && resultCode == RESULT_OK) {
        Bitmap bMap = null;
        try {
            bMap = getBitmapRotatedIfRequired(mCurrentPhotoPath);
        }
    }
}

```



```
        catch(IOException e){  
        }  
        mImageView.setImageBitmap(bMap);  
    }  
}
```

Важливо: робота із Bitmap у випадкові файлів великих розмірів вимагає обережного використання пам'яті. Рекомендовано ознайомитися зі статтею “[Displaying Bitmaps Efficiently](#)”.