У нас есть фотик и котик - работаем с камерой

Практически все современные телефоны и планшеты снабжаются камерами, что позволяет фотографировать любимых котиков.

Камера

Вы можете программно запустить из своей программы системное приложение "Камера" (в этом случае вам не понадобятся дополнительные разрешения) через намерение.

```
Intent intent = new Intent();
intent.setAction(Intent.ACTION_CAMERA_BUTTON);
intent.putExtra(Intent.EXTRA_KEY_EVENT, new KeyEvent(KeyEvent.ACTION_DOWN,
                KeyEvent.KEYCODE_CAMERA));
sendOrderedBroadcast(intent, null);
```

А вообще у пользователя могут стоять разные приложения, способные фотографировать. Тогда у вас будет появляться диалоговое окно с выбором нужного приложения. Они все имеют в своём составе такую запись в манифесте (для общего развития):

```
<action android:name="android.media.action.IMAGE_CAPTURE"/>
<category android:name="android.intent.category.DEFAULT"/>
```

У Гугла есть своя программа Google Камера. Запустим её, зная имя пакета.

```
Intent intent = getIntent();

intent.setComponent(null);
intent.setPackage("com.google.android.GoogleCamera");
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
startActivity(intent);
```

При вызове метода **getIntent()** вместо **new Intent()** приложение запускалось сразу, иначе - выводилось диалоговое окно выбора программы из списка. Также нужно быть уверенным, что программа установлена, в примере нет кода проверки.

Делаем фотографии и сохраняем результат. Простой пример

Просто включить камеру не слишком интересно. Рассмотрим практичный пример, когда мы программно запустим приложение "Камера", а полученную фотографию сохраним в папке. Для начала сфокусируемся на основной задаче, а потом напишем более сложное приложение.

Используйте статическую константу **MediaStore.ACTION_IMAGE_CAPTURE** для создания намерения, которое потом нужно передать методу **startActivityForResult()**. Разместите на форме кнопку и **ImageView**, в который будем помещать полученный снимок. Этот код

```
private static final int CAMERA_REQUEST = 0;
private ImageView imageView;

// в методе onCreate()
imageView = findViewById(R.id.imageView);

// Щелчок кнопки
public void onClick(View v) {
    Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    startActivityForResult(cameraIntent, CAMERA_REQUEST);

    // Длинный вариант
    // Intent cameraIntent = new Intent();
    // cameraIntent.setAction(MediaStore.ACTION_IMAGE_CAPTURE);
    // startActivityForResult(cameraIntent, CAMERA_REQUEST);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == CAMERA_REQUEST && resultCode == RESULT_OK) {
                // Фотка сделана, извлекаем картинку
        Bitmap thumbnailBitmap = (Bitmap) data.getExtras().get("data");
        imageView.setImageBitmap(thumbnailBitmap);
    }
}
```

Данный код запустит приложение, работающее с камерой, позволяя пользователю поменять настройки изображения, что освобождает вас от необходимости создавать своё собственное приложение для этих нужд. Вполне возможно, что у вас будет несколько приложений, умеющих делать фотографии, тогда сначала появится окно выбора программы.

При тестировании примера на своём телефоне я обнаружил небольшую проблему - когда снимок передавался обратно на моё приложение, то оно находилось в альбомном режиме, а потом возвращалось в портретный режим. При этом полученный снимок терялся. Поэтому перед нажатием кнопки я поворачивал телефон в альбомный режим, чтобы пример работал корректно. Поэтому вам надо предусмотреть подобное поведение, например, запретить приложению реагировать на поворот и таким образом избежать перезапуска **Activity**. У некоторых телефонов такой проблемы нет.

По умолчанию фотография возвращается в виде объекта **Bitmap**, содержащего миниатюру. Этот объект находится в параметре **data**, передаваемом в метод **onActivityResult()**. Чтобы получить миниатюру в виде объекта **Bitmap**, нужно вызвать метод **getParcelableExtra()** из

Если вы укажете исходящий путь **URI** с помощью параметра **MediaStore.EXTRA_OUTPUT** в запущенном намерении, полноразмерное изображение, снятое камерой, сохранится в заданном месте. В таком случае в метод **onActivityResult()** не будет передана миниатюра, а итоговое намерение продемонстрирует значение **null**.

В следующем примере показано, как при создании снимка получать миниатюру или полноценное изображение, используя намерение. Изображение будет сохранено во внешнем хранилище под именем **test.jpg**.

```
// и если нет, то не знаю, кто его читал.

package ru.alexanderklimov.photocamera;

import android.content.Intent;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.provider.MediaStore;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.ImageView;

import java.io.File;

public class MainActivity extends AppCompatActivity {

    private static int TAKE_PICTURE_REQUEST = 1;
    private ImageView imageView;
    private Uri outputFileUri;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        imageView = findViewById(R.id.imageView);
    }

    public void onClick(View view) {
        //getThumbnailPicture();
        saveFullImage();
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == TAKE_PICTURE_REQUEST && resultCode == RESULT_OK) {
            // Проверяем, содержит ли результат маленькую картинку
            if (data != null) {
                if (data.hasExtra("data")) {
                    Bitmap thumbnailBitmap = data.getParcelableExtra("data");
                    // Какие-то действия с миниатюрой
                    imageView.setImageBitmap(thumbnailBitmap);
```

```
                // сохраненным по адресу outputFileUri
                imageView.setImageURI(outputFileUri);
            }
        }
    }

    private void getThumbnailPicture() {
        Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        startActivityForResult(intent, TAKE_PICTURE_REQUEST);
    }

    private void saveFullImage() {
        Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        File file = new File(Environment.getExternalStorageDirectory(),
                "test.jpg");
        outputFileUri = Uri.fromFile(file);
        intent.putExtra(MediaStore.EXTRA_OUTPUT, outputFileUri);
        startActivityForResult(intent, TAKE_PICTURE_REQUEST);
    }
}
```

Добавим разрешение на запись файла в хранилище.

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

В реальных приложениях для создания имён файлов используют текущую дату, чтобы обеспечить уникальность и не затереть существующую фотографию.

В Android 6.0 Marshmallow пример перестал работать из-за новых правил с разрешениями. Новый вариант представлен в соответствующей статье.

На этом неприятности не закончились. В Lollipop для **Uri** нужно использовать метод **setClipData()**, а в **Nougat** запретили и сам **URI**, заставляя переходить на **FileProvider**.

Делаем фотографии или видео и сохраняем результат. Улучшенный пример

> Второй пример основан на тренировочном примере из документации Google (https://developer.android.com/training/camera/photobasics?hl=ru). Там же вы можете скачать исходный пример. Здесь я только своими словами пытаюсь объяснить код приложения

которые скачали ваше приложение, должны сфотографировать кота, а затем информация о месте съёмки и другие параметры попадают к вам в центр для обработки. Фотографирование котов - это часть вашего приложения. Не обязательно придумывать велосипед и работать напрямую с функциями камеры. У вас уже есть системное приложение Камера, с помощью которого можно быстро сделать фотографию и получить результат обратно в приложение.

Сначала сделаем небольшие приготовления. Есть класс устройств, у которых нет камер, например, электронные ридеры. Чтобы пользователи этих устройств не скачивали зря ваше приложение, которое окажется для них бесполезным, пропишем в манифесте требование наличия камеры.

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

Оформим отдельную функцию для запуска намерения.

```
private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(takePictureIntent, TAKE_PICTURE_REQUEST);
    }
}
```

Метод **resolveActivity()** поможет проверить активности, способное сделать фотографию. Если подходящего приложения не найдётся, то мы можем сделать кнопку для съёмки недоступной.

Полученное с камеры изображение можно обработать в методе **onActivityResult()**.

Системное приложение **Камера** кодирует фото в возвращаемом намерении, которое поступает в метод **onActivityResult()** в виде небольшого **Bitmap** в ключе **data**. Следующий код получает изображение и выводит его в **ImageView**.

```
private void handleSmallCameraPhoto(Intent intent) {
    Bundle extras = intent.getExtras();
    mImageBitmap = (Bitmap) extras.get("data");
    mImageView.setImageBitmap(mImageBitmap);
}
```

Полученное миниатюрное изображение из "data" вполне годится для использования в качестве аватара в контактах. Если мы хотим получить полноразмерное изображение, то напишем дополнительный код для другой кнопки.

В Android 2.2 и выше (API 8) есть специальный метод, чтобы получить стандартный путь для фотографий:

```
storageDir = new File(
    Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_PICTURES
    ),
    getAlbumName()
);
```

Указанная папка будет хранить фотографии даже после удаления приложения. Методы **getExternalFilesDir()** и **getFilesDir()** работают с папками приложения, поэтому фотографии будут удалены во время деинсталяции. Иногда такой подход может пригодиться, если фотки специфичны для приложения и нет необходимости сохранять их в общей папке.

Необходимо позаботиться об уникальности имени файла, чтобы избежать конфликтов:

```
String mCurrentPhotoPath;

private File createImageFile() throws IOException {
    // Create an image file name
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
    String imageFileName = "JPEG_" + timeStamp + "_";
    File storageDir = getExternalFilesDir(Environment.DIRECTORY_PICTURES);
    File image = File.createTempFile(
        imageFileName,  /* prefix */
        ".jpg",         /* suffix */
        storageDir      /* directory */
    );

    // Save a file: path for use with ACTION_VIEW intents
    mCurrentPhotoPath = image.getAbsolutePath();
    return image;
}
```

В нашем примере имя файла формируется из даты, что позволяет не бояться создания дубликатов.

Если у нас есть место для сохранения изображения, то сообщите путь приложению камеры через намерение:

```
File file = createImageFile();
takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(file));
```

следования тексту приложение у меня крашилось. Поэтому я внес поправки и расскажу действительно правильный вариант

Итак, прописываем разрешение **WRITE_EXTERNAL_STORAGE**. Но не прописывайте разрешение **android.permission.CAMERA**! Также добавьте использование камеры. Внутри секции **application** добавьте блок **provider**.

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

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

<provider
    android:name="android.support.v4.content.FileProvider"
    android:authorities="com.example.android.provider"
    android:exported="false"
    android:grantUriPermissions="true">
    <meta-data
        android:name="android.support.FILE_PROVIDER_PATHS"
        android:resource="@xml/file_paths"></meta-data>
</provider>
```

Дальше код для активности, который содержит кнопку и **ImageView**.

```
// и если нет, то не знаю, что с ней делать.
package ru.alexanderklimov.photo;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.provider.MediaStore;
import android.support.v4.content.FileProvider;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.ImageView;
import android.widget.Toast;

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;
    private String mCurrentPhotoPath;
    private ImageView imageView;
    private Uri photoURI;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        imageView = findViewById(R.id.imageView);
    }

    public void onClick(View view) {
        dispatchTakePictureIntent();
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == REQUEST_TAKE_PHOTO && resultCode == RESULT_OK) {
            imageView.setImageURI(photoURI);
        }
    }
```

```
        String imageFileName = "JPEG_" + timeStamp + "_";
        File storageDir = getExternalFilesDir(Environment.DIRECTORY_PICTURES);
        File image = File.createTempFile(
                imageFileName,  /* prefix */
                ".jpg",         /* suffix */
                storageDir      /* directory */
        );

        // Save a file: path for use with ACTION_VIEW intents
        mCurrentPhotoPath = image.getAbsolutePath();
        return image;
    }

    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
                Toast.makeText(this, "Error!", Toast.LENGTH_SHORT).show();
            }
            // Continue only if the File was successfully created
            if (photoFile != null) {
                photoURI = FileProvider.getUriForFile(this,
                        "com.example.android.provider",
                        photoFile);
                takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI);
                startActivityForResult(takePictureIntent, REQUEST_TAKE_PHOTO);
            }
        }
    }
}
```

В папке **res/xml** создаём файл **file_paths.xml**. Вместо **ru.alexanderklimov.photo** используйте имя вашего пакета.

```
<?xml version="1.0" encoding="utf-8"?>
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <external-path name="my_images"
path="Android/data/ru.alexanderklimov.photo/files/Pictures" />
</paths>
```

Уведомляем приложение Галерея

Можно сделать фотографию доступной для системы при помощи Media Provider.

Следующий пример демонстрирует метод вызова системного медиа-сканера, чтобы добавить вашу фотографию в базу данных Media Provider, что сделает её видимой в приложении **Галерея** и других приложениях.

```
private void galleryAddPic() {
    Intent mediaScanIntent = new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);
    File file = new File(mCurrentPhotoPath);
    Uri contentUri = Uri.fromFile(file);
    mediaScanIntent.setData(contentUri);
    this.sendBroadcast(mediaScanIntent);
}
```

Декодирование масштабированого изображения

Работа с большими изображениями может вызвать проблемы на устройстве с ограниченным объемом памяти. Если ваше приложение вызывает нехватку памяти после вывода нескольких изображений, вы можете значительно уменьшить объем занимаемой памяти, использовав во время распаковки JPEG-файлов в память масштабирование, которое учитывает размеры вашего View для просмотра картинок:

```
    // Get the dimensions of the view
    int targetW = mImageView.getWidth();
    int targetH = mImageView.getHeight();

    // Get the dimensions of the bitmap
    BitmapFactory.Options bmOptions = new BitmapFactory.Options();
    bmOptions.inJustDecodeBounds = true;
    BitmapFactory.decodeFile(mCurrentPhotoPath, bmOptions);
    int photoW = bmOptions.outWidth;
    int photoH = bmOptions.outHeight;

    // Determine how much to scale down the image
    int scaleFactor = Math.min(photoW/targetW, photoH/targetH);

    // Decode the image file into a Bitmap sized to fill the View
    bmOptions.inJustDecodeBounds = false;
    bmOptions.inSampleSize = scaleFactor;
    bmOptions.inPurgeable = true;

    Bitmap bitmap = BitmapFactory.decodeFile(mCurrentPhotoPath, bmOptions);
    mImageView.setImageBitmap(bitmap);
}
```

Кроме того, в учебном примере рассматривается вывод видео в элемент **VideoView**. Изучайте код самостоятельно.

Снимаем и кадрируем

Рассмотрим ещё один пример, когда мы запустим приложение Камера, а также включим режим кадрирования. Обратите внимание, что основная часть кода остаётся прежней. Я специально даю разные примеры с незначительными изменениями, чтобы вы могли выбрать подходящие приёмы для вашей задачи. Также рекомендую проверять работу с камерой на реальных устройствах, так как многие производители заменяют стандартные методы съёмки своими прошивками и драйверами. В частности, намерение с кадрированием является проблемной, и в интернете многие жалуются на отсутствие поддержки этого способа.

Создадим простенький макет из кнопки для запуска камеры и **ImageView** для вывода кадрированного изображения.

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <Button
        android:id="@+id/buttonCapture"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:onClick="onClick"
        android:text="Запустить камеру" />

    <ImageView
        android:id="@+id/picture"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:background="@drawable/background" />

</LinearLayout>
```

Для большей красоты сделаем задний фон у **ImageView** с закруглёнными углами и обводкой. Для этого в атрибуте **android:background** мы прописали специальный стиль. Создайте папку **res/drawable**, а в ней файл **background.xml** следующего содержания:

```
    android:dither="true" />

    <gradient
        android:angle="90"
        android:centerColor="#00000000"
        android:endColor="#99ffffff"
        android:startColor="#99ffffff" />

    <padding
        android:bottom="10dp"
        android:left="10dp"
        android:right="10dp"
        android:top="10dp" />

    <corners android:radius="5dp" />

    <stroke
        android:width="2dp"
        android:color="#ccffffff" />

</shape>
```

Этот шаг не является обязательным и его можно пропустить.

При нажатии кнопки запускаем приложение Камера и ожидаем результата.

```
public void onClick(View view) {
        try {
                // Намерение для запуска камеры
                Intent captureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                startActivityForResult(captureIntent, CAMERA_REQUEST);
        } catch (ActivityNotFoundException e) {
                // Выводим сообщение об ошибке
                String errorMessage = "Ваше устройство не поддерживает съемку";
                Toast toast = Toast
                                .makeText(this, errorMessage, Toast.LENGTH_SHORT);
                toast.show();
        }
}
```

После того, как пользователь сделал нужный кадр, программа Камера возвращает результат обратно в наше приложение. Результат обрабатывается в методе **onActivityResult()**:

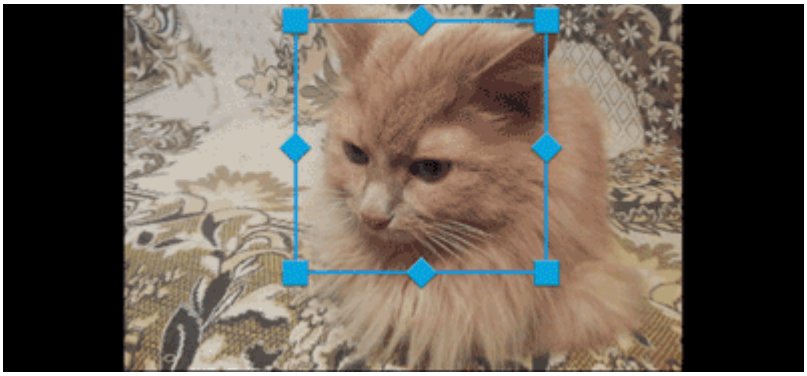
```
        if (resultCode == RESULT_OK) {
                // Вернулись от приложения Камера
                if (requestCode == CAMERA_REQUEST) {
                        // Получим Uri снимка
                        picUri = data.getData();
                        // кадрируем его
                        performCrop();
                }
                // Вернулись из операции кадрирования
                else if(requestCode == PIC_CROP){
                        Bundle extras = data.getExtras();
                        // Получим кадрированное изображение
                        Bitmap thePic = extras.getParcelable("data");
                        // передаём его в ImageView
                        ImageView picView = (ImageView)findViewById(R.id.picture);
                        picView.setImageBitmap(thePic);
                }
        }
}
```

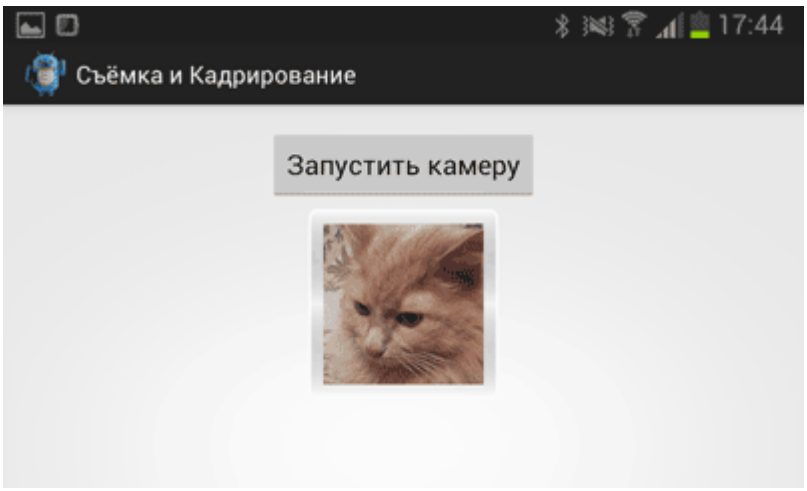
Получив полноразмерное изображение, мы пытаемся откадрировать его. Для этого создадим метод **performCrop()**, который запускает специальное намерение, предназначенное для этих целей. В успешном случае результат снова возвращается в наше приложение, но уже с другим кодом **PIC_CROP**. Теперь мы имеем нужное изображение, которое можно вывести на экран.

При кадрировании мы указываем желаемые размеры (код метода ниже). Если указать слишком больше размеры (больше 400), то результат не возвращается. Попробуйте добавить ещё два параметра:

```
//куда сохраняем
intent.putExtra("output", picUri);
intent.putExtra("outputFormat", "JPEG");
```

Результат работы приложения, когда запускается намерение кадрирования и итоговый результат. Желательно тренироваться на кошках.





Исходник полностью.

▶ Показать код (щелкните мышкой)

Запуск камеры в нужном режиме

Обнаружил, что появилось новое намерение, позволяющее включить камеру в нужном режиме: фотосъёмка или видео.

```
// фотосъёмка
public void capturePhoto() {
    Intent intent = new Intent(MediaStore.INTENT_ACTION_STILL_IMAGE_CAMERA);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(intent, 1);
    }
}

// видеосъёмка
public void capturePhoto() {
    Intent intent = new Intent(MediaStore.INTENT_ACTION_VIDEO_CAMERA);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(intent);
    }
}
```