9 Аудіо, відео, камера

9.1 Відтворення аудіофайлів. Запис аудіо

Для відтворення музики та інших аудіоматеріалів Android надає клас **MediaPlayer**.

Щоб відтворювати аудіо, MediaPlayer повинен знати, який ресурс (файл) потрібно виробляти. Встановити потрібний ресурс для відтворення можна трьома способами:

- у метод `create()` об'єкта MediaPlayer передається id ресурсу, що представляє аудіофайл
- метод `create()` об'єкта MediaPlayer передається об'єкт Uri, що представляє аудіофайл
- у метод `setDataSource()` об'єкту MediaPlayer передається повний шлях до аудіофайлу

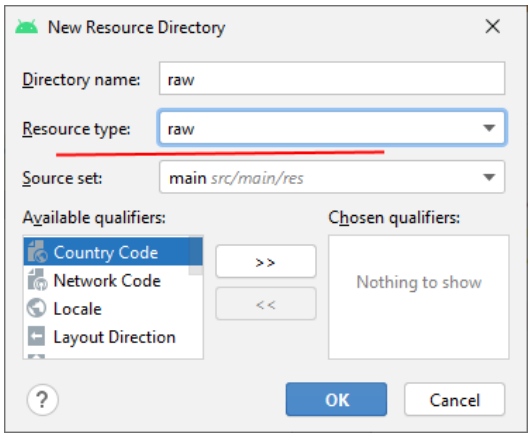
Після встановлення ресурсу викликається метод `prepare()` або `prepareAsync()` (асинхронний варіант prepare()). Цей метод готує аудіофайл до відтворення, витягуючи з нього перші секунди. Якщо ми відтворюємо файл з мережі, краще використовувати prepareAsync().

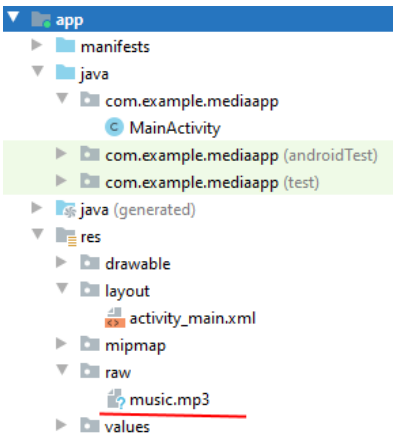
Для керування відтворенням у класі MediaPlayer визначено такі методи:

- `start()`: запускає аудіо
- `pause()`: призупиняє відтворення
- `stop()`: повністю зупиняє відтворення

Отже, створимо новий проект. Як і у випадку з відео, аудіофайл повинен знаходитися в папці **res/raw** , тому додамо в проект Android Studio таку папку. Для цього натиснемо на папку res правою кнопкою миші і в меню виберемо **New -> Android Resource Directory**.

Потім у вікні як тип папки вкажемо **raw** (що також буде використовуватися як назва папки). І скопіюємо в неї якийсь аудіо-файл.





Для керування аудіопотоком визначимо у файлі **activity_main.xml** три кнопки:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/playButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Play"
        android:onClick="play"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@id/pauseButton"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/pauseButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Pause"
        android:onClick="pause"
        app:layout_constraintLeft_toRightOf="@id/playButton"
        app:layout_constraintRight_toLeftOf="@id/stopButton"
        app:layout_constraintTop_toTopOf="parent"/>
    <Button
        android:id="@+id/stopButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Stop"
        android:onClick="stop"
        app:layout_constraintLeft_toRightOf="@id/pauseButton"
        app:layout_constraintRight_toRightOf="parent"
```

```
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

І змінимо код класу **MainActivity** :

```
package com.example.mediaapp;

import androidx.appcompat.app.AppCompatActivity;

import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    MediaPlayer mPlayer;
    Button playButton, pauseButton, stopButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mPlayer= MediaPlayer.create(this, R.raw.music);
        mPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener()
{
            @Override
            public void onCompletion(MediaPlayer mp) {
                stopPlay();
            }
        });
        playButton = findViewById(R.id.playButton);
        pauseButton = findViewById(R.id.pauseButton);
        stopButton = findViewById(R.id.stopButton);

        pauseButton.setEnabled(false);
        stopButton.setEnabled(false);
    }
    private void stopPlay(){
        mPlayer.stop();
        pauseButton.setEnabled(false);
        stopButton.setEnabled(false);
        try {
            mPlayer.prepare();
            mPlayer.seekTo(0);
```

```
42                playButton.setEnabled(true);
43            }
44            catch (Throwable t) {
45                Toast.makeText(this, t.getMessage(), Toast.LENGTH_SHORT).show();
46            }
47        }
48        public void play(View view){
49
50            mPlayer.start();
51            playButton.setEnabled(false);
52            pauseButton.setEnabled(true);
53            stopButton.setEnabled(true);
54        }
55        public void pause(View view){
56
57            mPlayer.pause();
58            playButton.setEnabled(true);
59            pauseButton.setEnabled(false);
60            stopButton.setEnabled(true);
61        }
62        public void stop(View view){
63            stopPlay();
64        }
65        @Override
66        public void onDestroy() {
67            super.onDestroy();
68            if (mPlayer.isPlaying()) {
69                stopPlay();
70            }
71        }
    }
```

Обробник кожної кнопки крім виклику певного методу **MediaPlayer** також перемикає доступність кнопок.

І якщо запуск та призупинення відтворення особливих складнощів не викликає, то при обробці повної зупинки відтворення ми можемо зіткнутися з низкою труднощів. Зокрема, коли ми виходимо з програми - повністю закриваємо його через диспетчер додатків або натискаємо кнопку Назад, то у нас для поточної Activity викликається метод onDestroy, activity знищується, але MediaPlayer продовжує **працювати** . Якщо ми повернемось до програми, то activity буде створена заново, але за допомогою кнопок ми не зможемо керувати відтворенням. Тому в даному випадку перевизначаємо метод onDestroy, в якому завершуємо відтворення.

Для коректного завершення також визначено обробника природного завершення відтворення `OnCompletionListener`, дія якого буде аналогічна натискання на кнопку "Стоп".

Додамо до відтворення індикатор гучності. Для цього у файлі **activity_main.xml** визначимо **SeekBar** :

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/playButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Play"
        android:onClick="play"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@id/pauseButton"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@id/volumeControl" />
    <Button
        android:id="@+id/pauseButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Pause"
        android:onClick="pause"
        app:layout_constraintLeft_toRightOf="@id/playButton"
        app:layout_constraintRight_toLeftOf="@id/stopButton"
        app:layout_constraintTop_toTopOf="parent"/>
    <Button
        android:id="@+id/stopButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Stop"
        android:onClick="stop"
        app:layout_constraintLeft_toRightOf="@id/pauseButton"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <SeekBar
        android:id="@+id/volumeControl"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
```

```
        android:layout_marginTop="32dp"
        app:layout_constraintTop_toBottomOf="@id/playButton"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintLeft_toLeftOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

І далі змінимо код класу **MainActivity** :

```
package com.example.mediaapp;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    MediaPlayer mPlayer;
    Button playButton, pauseButton, stopButton;
    SeekBar volumeControl;
    AudioManager audioManager;

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

        mPlayer=MediaPlayer.create(this, R.raw.music);
        mPlayer.setOnCompletionListener(new
MediaPlayer.OnCompletionListener() {
            @Override
            public void onCompletion(MediaPlayer mp) {
                stopPlay();
            }
        });
        playButton = findViewById(R.id.playButton);
        pauseButton = findViewById(R.id.pauseButton);
        stopButton = findViewById(R.id.stopButton);

        audioManager = (AudioManager)
getSystemService(Context.AUDIO_SERVICE);
```

```
        int maxVolume =
audioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
        int curValue =
audioManager.getStreamVolume(AudioManager.STREAM_MUSIC);

        volumeControl = findViewById(R.id.volumeControl);
        volumeControl.setMax(maxVolume);
        volumeControl.setProgress(curValue);
        volumeControl.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
                audioManager.setStreamVolume(AudioManager.STREAM_MUSIC,
progress, 0);
            }
            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });

        pauseButton.setEnabled(false);
        stopButton.setEnabled(false);
    }
    private void stopPlay(){
        mPlayer.stop();
        pauseButton.setEnabled(false);
        stopButton.setEnabled(false);
        try {
            mPlayer.prepare();
            mPlayer.seekTo(0);
            playButton.setEnabled(true);
        }
        catch (Throwable t) {
            Toast.makeText(this, t.getMessage(), Toast.LENGTH_SHORT).show();
        }
    }

    public void play(View view){

```

```
85              mPlayer.start();
86              playButton.setEnabled(false);
87              pauseButton.setEnabled(true);
88              stopButton.setEnabled(true);
89          }
90          public void pause(View view){
91
92              mPlayer.pause();
93              playButton.setEnabled(true);
94              pauseButton.setEnabled(false);
95              stopButton.setEnabled(true);
96          }
97          public void stop(View view){
98              stopPlay();
99          }
100         @Override
            public void onDestroy() {
                super.onDestroy();
                if (mPlayer.isPlaying()) {
                    stopPlay();
                }
            }
        }
```

Для керування гучністю звуку застосовується клас **AudioManager** . А в за допомогою дзвінка `audioManager.setStreamVolume(AudioManager.STREAM_MUSIC, progress, 0);` як другий параметр можна передати потрібне значення гучності.

9.2 Відтворення відеофайлів. Запис відео

Для роботи з відеоматеріалами у стандартному наборі віджетів Android визначено клас VideoView, що дозволяє відтворювати відео.

Які типи відеофайлів можна використовувати? Android підтримує більшість поширених типів відеофайлів, зокрема 3GPP (.3gp), WebM (.webm), Matroska (.mkv), MPEG-4 (.mp4).

VideoView може працювати як з роликами, розміщеними на мобільному пристрої, так і відеоматеріалами з мережі. У цьому випадку використовуємо відеоролик, розміщений локально. Для цього додамо до проекту якийсь відеоролик. Зазвичай відеоматеріали поміщають у проекті папку **res/raw** . За замовчуванням проект не містить такої папки, тому додамо до каталогу res підпапку raw. Для цього натиснемо на папку res правою кнопкою миші і в меню виберемо **New -> Android Resource Directory** :

Потім у вікні як тип папки вкажемо **raw** (що також буде використовуватися як назва папки). Після додавання папки raw скопіюємо в неї якийсь відеофайл.

Тепер визначимо функціонал його відтворення. Для цього у файлі **activity_main.xml** вкажемо наступний код:

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
3       xmlns:android="http://schemas.android.com/apk/res/android"
4       xmlns:app="http://schemas.android.com/apk/res-auto"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent">
7
8       <Button
9           android:id="@+id/playButton"
10          android:layout_width="0dp"
11          android:layout_height="wrap_content"
12          android:text="Play"
13          android:onClick="play"
14          app:layout_constraintBottom_toTopOf="@id/videoPlayer"
15          app:layout_constraintLeft_toLeftOf="parent"
16          app:layout_constraintRight_toLeftOf="@id/pauseButton"
17          app:layout_constraintTop_toTopOf="parent" />
18      <Button
19          android:id="@+id/pauseButton"
20          android:layout_width="0dp"
21          android:layout_height="wrap_content"
22          android:text="Pause"
23          android:onClick="pause"
24          app:layout_constraintBottom_toTopOf="@id/videoPlayer"
25          app:layout_constraintLeft_toRightOf="@id/playButton"
26          app:layout_constraintRight_toLeftOf="@id/stopButton"
27          app:layout_constraintTop_toTopOf="parent"/>
28      <Button
29          android:id="@+id/stopButton"
30          android:layout_width="0dp"
31          android:layout_height="wrap_content"
32          android:text="Stop"
33          android:onClick="stop"
34          app:layout_constraintBottom_toTopOf="@id/videoPlayer"
35          app:layout_constraintLeft_toRightOf="@id/pauseButton"
36          app:layout_constraintRight_toRightOf="parent"
37          app:layout_constraintTop_toTopOf="parent" />
38      <VideoView android:id="@+id/videoPlayer"
```

```
        android:layout_height="0dp"
        android:layout_width="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@id/playButton"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Для керування відтворенням відео тут визначено три кнопки: для запуску відео, для паузи та його зупинки.

Також змінимо код **MainActivity** :

```
package com.example.mediaapp;

import androidx.appcompat.app.AppCompatActivity;

import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.VideoView;

public class MainActivity extends AppCompatActivity {

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

        videoPlayer = findViewById(R.id.videoPlayer);
        Uri myVideoUri= Uri.parse( "android.resource://" + getPackageName() + "/"
+ R.raw.cats);
        videoPlayer.setVideoURI(myVideoUri);
    }

    public void play(View view){
        videoPlayer.start();
    }
    public void pause(View view){
        videoPlayer.pause();
    }
    public void stop(View view){
        videoPlayer.stopPlayback();
        videoPlayer.resume();
    }
```

```
    }
```

По-перше, щоб керувати потоком відтворення, нам треба отримати об'єкт VideoView:`videoPlayer = findViewById(R.id.videoPlayer);`

Щоб вказати джерело відтворення, потрібний об'єкт **Uri** . В даному випадку за допомогою виразу `Uri myVideoUri= Uri.parse("android.resource://" + getPackageName() + "/" + R.raw.cats);` отримуємо адресу відео усередині пакета програми.

Рядок URI має ряд частин: спочатку йде `Uri-схема`(`http://`або як тут `android.resource://`), потім назва пакета, одержуване через метод getPackageName(), і далі безпосередньо назва ресурсу відео з папки res/raw , яке збігається з назвою файлу.

Потім цей Uri встановлюється у VideoPlayer: `videoPlayer.setVideoURI(myVideoUri);`

Щоб керувати відеопотоком, обробники натискання кнопок викликають відповідну дію:

```
1     public void play(View view){
2             videoPlayer.start();
3         }
4         public void pause(View view){
5             videoPlayer.pause();
6         }
7         public void stop(View view){
8             videoPlayer.stopPlayback();
9             videoPlayer.resume();
10        }
```

Метод `videoPlayer.start()`починає або продовжує відтворення.

Метод `videoPlayer.pause()`зупиняє відео.

Метод `videoPlayer.stopPlayback()`повністю зупиняє відео.

Метод `videoPlayer.resume()`дозволяє знову розпочати відтворення відео з початку після його повної зупинки.

При запуску програми ми зможемо за допомогою кнопок керувати відтворенням.

MediaController

За допомогою класу **MediaController** ми можемо додати до VideoView додаткові елементи керування. Для цього змінимо код MainActivity:

```
1   package com.example.mediaapp;
2
3   import androidx.appcompat.app.AppCompatActivity;
```

```
4
5   import android.net.Uri;
6   import android.os.Bundle;
7   import android.view.View;
8   import android.widget.VideoView;
9   import android.widget.MediaController;
10
11  public class MainActivity extends AppCompatActivity {
12
13      VideoView videoPlayer;
14      @Override
15      protected void onCreate(Bundle savedInstanceState) {
16          super.onCreate(savedInstanceState);
17          setContentView(R.layout.activity_main);
18
19          videoPlayer = findViewById(R.id.videoPlayer);
20          Uri myVideoUri= Uri.parse( "android.resource://" + getPackageName() + "/"
21  + R.raw.cats);
22          videoPlayer.setVideoURI(myVideoUri);
23          MediaController mediaController = new MediaController(this);
24          videoPlayer.setMediaController(mediaController);
25          mediaController.setMediaPlayer(videoPlayer);
26      }
27
28      public void play(View view){
29          videoPlayer.start();
30      }
31      public void pause(View view){
32          videoPlayer.pause();
33      }
34      public void stop(View view){
35          videoPlayer.stopPlayback();
36          videoPlayer.resume();
37      }
    }
```

І якщо ми запустимо програми, то при дотику до VideoView внизу з'являться інструменти для керування відео. У принципі тепер і кнопки, які ми створили раніше, не потрібні.

Відтворення файлу з Інтернету

VideoView підтримує відтворення файлу з Інтернету. Але щоб це стало можливо, необхідно у файлі **AndroidManifest.xml** встановити дозвіл **android.permission.INTERNET** , тому що ми отримуємо дані з інтернету:

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

Далі змінимо клас MainActivity:

```
package com.example.mediaapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.VideoView;

public class MainActivity extends AppCompatActivity {

    VideoView videoPlayer;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        videoPlayer = findViewById(R.id.videoPlayer);
        videoPlayer.setVideoPath("http://techslides.com/demos/sample-videos/small.mp4");
    }

    public void play(View view){
        videoPlayer.start();
    }
    public void pause(View view){
        videoPlayer.pause();
    }
    public void stop(View view){
        videoPlayer.stopPlayback();
        videoPlayer.resume();
    }
}
```

Тут нам треба в метод `videoPlayer.setVideoPath()` передати інтернет-адресу файлу, що відтворюється.

9.3 Використання камери

Для безпосереднього керування камерою пристрою потрібно набагато більше коду, ніж для запиту зображень або відео з існуючих програм камери.

Отримання екземпляра об'єкта Camera є першим кроком у процесі безпосереднього керування камерою. Як і власна програма Android Camera,

рекомендований спосіб отримати доступ до камери – відкрити Cameraв окремому потоці, який запускається з onCreate(). Такий підхід є гарною ідеєю, оскільки це може зайняти деякий час і може призвести до затримки потоку інтерфейсу користувача. У більш базовій реалізації відкриття камери можна відкласти до onResume() методу, щоб полегшити повторне використання коду та зберегти простий потік керування.

Виклик Camera.open() створює виняток, якщо камера вже використовується іншою програмою, тому ми загортаємо її в блок try.

```
private fun safeCameraOpen(id: Int): Boolean {
    return try {
        releaseCameraAndPreview()
        mCamera = Camera.open(id)
        true
    } catch (e: Exception) {
        Log.e(getString(R.string.app_name), "failed to open Camera")
        e.printStackTrace()
        false
    }
}

private fun releaseCameraAndPreview() {
    preview?.setCamera(null)
    mCamera?.also { camera ->
        camera.release()
        mCamera = null
    }
}
```

Починаючи з рівня API 9, структура камери підтримує кілька камер. Якщо ви використовуєте застарілий API і викликаєте open() без аргументів, ви отримуєте першу задню камеру.

Створіть попередній перегляд камери

Для зйомки зазвичай потрібно, щоб ваші користувачі бачили попередній перегляд об’єкта зйомки перед тим, як натиснути кнопку затвора. Для цього ви можете використати SurfaceView для попереднього перегляду того, що знімає датчик камери.

Попередній перегляд класу

Щоб розпочати відображення попереднього перегляду, вам потрібен клас попереднього перегляду. Для попереднього перегляду потрібна реалізація

інтерфейсу android.view.SurfaceHolder.Callback, який використовується для передачі даних зображення від обладнання камери до програми.

```
class Preview(
        context: Context,
        val surfaceView: SurfaceView = SurfaceView(context)
) : ViewGroup(context), SurfaceHolder.Callback {

    var mHolder: SurfaceHolder = surfaceView.holder.apply {
        addCallback(this@Preview)
        setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS)
    }
    ...
}
```

Клас попереднього перегляду має бути переданий об'єкту, Cameraперш ніж можна буде запустити попередній перегляд живого зображення, як показано в наступному розділі.

Встановіть і запустіть попередній перегляд

Екземпляр камери та відповідний попередній перегляд мають бути створені в певному порядку, причому об'єкт камери має бути першим. У наведеному нижче фрагменті процес ініціалізації камери інкапсульовано таким чином, що Camera.startPreview() викликається методом setCamera() щоразу, коли користувач щось робить, щоб змінити камеру. Попередній перегляд також потрібно перезапустити в surfaceChanged()методі зворотного виклику класу попереднього перегляду.

```
fun setCamera(camera: Camera?) {
    if (mCamera == camera) {
        return
    }

    stopPreviewAndFreeCamera()

    mCamera = camera

    mCamera?.apply {
        mSupportedPreviewSizes = parameters.supportedPreviewSizes
        requestLayout()

        try {
            setPreviewDisplay(holder)
        } catch (e: IOException) {
            e.printStackTrace()
        }
```

```
        // Important: Call startPreview() to start updating the preview
        // surface. Preview must be started before you can take a picture.
        startPreview()
    }
}
```

Змінити налаштування камери

Налаштування камери змінюють спосіб зйомки камерою, від рівня масштабування до компенсації експозиції. Цей приклад змінює лише розмір попереднього перегляду; дивіться вихідний код програми Камера, щоб дізнатися про багато іншого.

```
override fun surfaceChanged(holder: SurfaceHolder, format: Int, w: Int, h: Int) {
    mCamera?.apply {
        // Now that the size is known, set up the camera parameters and begin
        // the preview.
        parameters?.also { params ->
            params.setPreviewSize(previewSize.width, previewSize.height)
            requestLayout()
            parameters = params
        }

        // Important: Call startPreview() to start updating the preview surface.
        // Preview must be started before you can take a picture.
        startPreview()
    }
}
```

Встановіть орієнтацію попереднього перегляду

Більшість програм камери блокують дисплей у ландшафтному режимі, оскільки це природна орієнтація датчика камери. Цей параметр не заважає вам робити фотографії в портретному режимі, оскільки орієнтація пристрою записується в заголовку EXIF. Цей setCameraDisplayOrientation() метод дозволяє змінити спосіб відображення попереднього перегляду, не впливаючи на спосіб запису зображення. Однак в Android до рівня API 14 ви повинні зупинити попередній перегляд перед зміною орієнтації, а потім перезапустити його.

Зробити знімок

Використовуйте цей Camera.takePicture() метод, щоб зробити знімок після запуску попереднього перегляду. Ви можете створювати Camera.PictureCallback об’єкти Camera.ShutterCallbackта передавати їх у Camera.takePicture().

Якщо ви хочете постійно захоплювати зображення, ви можете створити, Camera.PreviewCallbackякий реалізує onPreviewFrame(). Для чогось між

ними можна захопити лише вибрані кадри попереднього перегляду або налаштувати відкладену дію для виклику takePicture().

Перезапустіть попередній перегляд

Після того, як знімок зроблено, ви повинні перезапустити попередній перегляд, перш ніж користувач зможе зробити інший знімок. У цьому прикладі перезапуск здійснюється шляхом перевантаження кнопки спуску затвора.

```
fun onClick(v: View) {
   previewState = if (previewState == K_STATE_FROZEN) {
      camera?.startPreview()
      K_STATE_PREVIEW
   } else {
      camera?.takePicture(null, rawCallback, null)
      K_STATE_BUSY
   }
   shutterBtnConfig()
}
```

Зупиніть попередній перегляд і відпустіть камеру

Після того, як ваша програма буде виконана за допомогою камери, настав час очистити. Зокрема, ви повинні звільнити Camera об’єкт, інакше ви ризикуєте призвести до збою інших програм, включаючи нові екземпляри вашої власної програми.

Коли слід зупинити попередній перегляд і відпустити камеру? Що ж, знищення поверхні попереднього перегляду є досить хорошим натяком на те, що настав час зупинити попередній перегляд і відпустити камеру, як показано в цих методах із класу Preview.

```
override fun surfaceDestroyed(holder: SurfaceHolder) {
   // Surface will be destroyed when we return, so stop the preview.
   // Call stopPreview() to stop updating the preview surface.
   mCamera?.stopPreview()
}

/**
 * When this function returns, mCamera will be null.
 */
private fun stopPreviewAndFreeCamera() {
   mCamera?.apply {
      // Call stopPreview() to stop updating the preview surface.
      stopPreview()

      // Important: Call release() to release the camera for use by other
      // applications. Applications should release the camera immediately
```

```
        // during onPause() and re-open() it during onResume()).
        release()

        mCamera = null
    }
}
```

Раніше ця процедура також була частиною методу setCamera(), тому ініціалізація камери завжди починається з зупинки попереднього перегляду.