INSTITUTO FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA

Introdução à Programação Android

Prof. Samir Bonho

Aula 8

Florianópolis, 16 de Abril de 2014

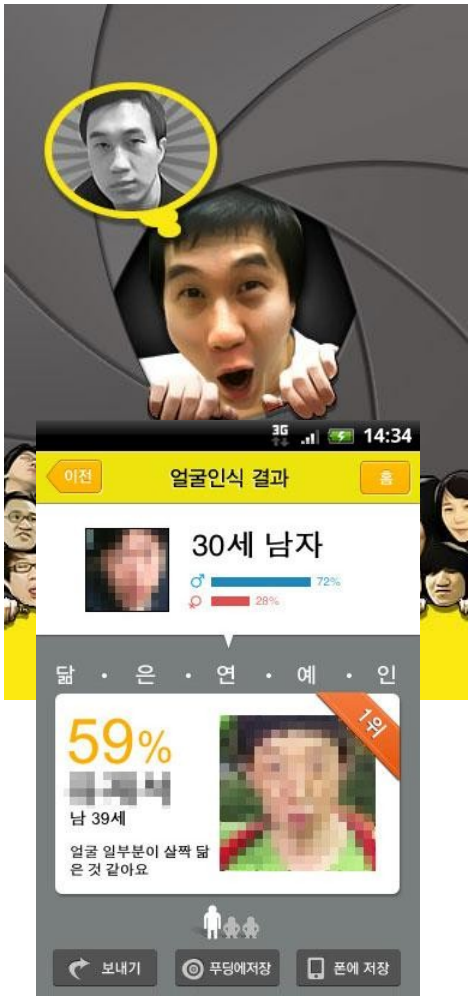
Sumário

- ✓ Camera
- ✓ Fontes:
  - ✓ Android Developers. Disponível em http://developer.android.com/
  - ✓ California State University -Department of Math & Computer Science

Cameras

Android: Camera

- O que você pode fazer utilizando a camera?



Permissão para utilização da Android Camera

- Sempre colocar no arquivo AndroidManifest.xml

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera" />
```

Duas opções de acesso à camera

- http://developer.android.com/guide/topics/me

- Opção 1: Criando um Intent *linkado* com a App da Camera
  - utilizando-se a aplicação de câmera nativa que já existe no Android para tirar a foto e capturar o arquivo para dentro da aplicação: aqui o controle da camera e parâmetros são feitos pela app nativa.

Duas opções de acesso à camera

- Opção 2: Criando os controles da camera dentro da aplicação
  - Acessando diretamente as classes que controlam o hardware da câmera.

Classes envolvidas no acesso à camera

- Opção 1: através de Intent acessando a app nativa da Camera
  - Intent As ações MediaStore.ACTION_IMAGE_CAPTURE or MediaStore.ACTION_VIDEO_CAPTURE podem ser utilizadas para captura da imagem e arquivo.

- Opção 2: através do uso de classes na própria aplicação
  - Camera API principal para o controle da camera. Esta classe é usada para tirar fotos ou gravar videos.
  - SurfaceView Utilizada para mostrar um preview em tempo real da imagem da camera.
  - MediaRecorder Utilizada para gravações de videos.

OPÇÃO1: CRIANDO UM INTENT PARA CHAMAR A APLICAÇÃO NATIVA DA CAMERA

Utiliza a aplicação nativa da camera e retorna a foto/video diretamente para o aplicativo para que seja armazenada como um arquivo pertencente ao seu *datapath.*

Opção1: Via Intent

1. **Crie o Intent** – Crie um Intent que requisite uma foto ou video utilizando os seguintes tipos de intent:
   - MediaStore.ACTION_IMAGE_CAPTURE - Solicita uma foto da aplicação nativa da camera.
   - MediaStore.ACTION_VIDEO_CAPTURE – Solicita um video do aplicativo nativo da camera.
1. **Inicialize o Intent** - Utilize o método startActivityForResult() para executar o intent. Depois da inicialização, a aplicação nativa da camera aparecerá para que o usuário tire a foto ou faça o video.

2. **Receive the Intent Result** – Configure o método onActivityResult() para receber o *callback* do intent da camera. Esta função é chamada após a finalização da foto/vídeo.

Opção1: Via Intent

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="grewe.example.Camera.Simple"
    android:versionCode="1"
    android:versionName="1.0">
  <uses-sdk android:minSdkVersion="12" />
  <uses-permission android:name="android.permission.CAMERA"></uses-permission>
  <uses-feature android:name="android.hardware.camera" />
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"></uses-permission>
  <application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".CameraSimpleActivity"
          android:label="@string/app_name">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

Opção1: Via Intent

```
/** Called when the activity is first created. --- CREATES INTENT AND STARTS IT */
@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);

   //get the picture taken by external existing Camera App right
   //away at start...could do instead as a result of Event handling
   // create Intent to take a picture and return control to the calling application
   Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);    //Create Intent

    //setup File URL for location for image
   fileUri = getOutputMediaFileUri(MEDIA_TYPE_IMAGE); // create a file to save the image
   intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri); // set the image file name
   // start the image capture Intent
   startActivityForResult(intent, CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE);
}
```

Opção1: Via Intent

```
//METHOD to Get Intents Recieved --in this case when camera app done
//   will put message in TextView on main apps interface
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {

            // say that we saved image
            TextView t = (TextView)findViewById(R.id.textView_Results);
             t.setText("Success");

        } else if (resultCode == RESULT_CANCELED) {
            // User cancelled the image capture
            TextView t = (TextView)findViewById(R.id.textView_Results);
            t.setText("Camera Cancelled");
        } else {
            // Image capture failed, advise user
            TextView t = (TextView)findViewById(R.id.textView_Results);
            t.setText("Failure");
        }
```

Opção1: Via Intent

```
    if (requestCode == CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            // Video captured and saved to fileUri specified in the Intent
            Toast.makeText(this, "Video saved to:\n" +  data.getData(),
    Toast.LENGTH_LONG).show();
        } else if (resultCode == RESULT_CANCELED) {
            // User cancelled the video capture
        } else {
            // Video capture failed, advise user
        }
    }
}
```

OPÇÃO 2: A PRÓPRIA APLICAÇÃO EXIBE INTERFACE E CONTROLES DA CAMERA

É necessária a utilização de classes que forneçam o acesso à camera.

Interessante para implementação de funções não existentes no aplicativo nativo.

Opção 2: Classes necessárias

- Camera: Classe base para acesso às funcionalidades da camera.
- SurfaceView Mostra uma preview em tempo real da imagem focalizada na camera.
- MediaRecorder Utilizada para gravação de videos.

Opção 2: Diferentes cameras

- Dispositivos Android podem ter mais de uma camera.
  - Versões Android 2.3 (API Level 9) e posteriores:
    - Camera.getNumberOfCameras()
      - Verificação do número de cameras disponível
    - Camera.getCameraInfo()
      - Informação sobre a posição da camera (frente ou costas), orientação da imagem, etc.

Opção 2: Passos

1. **Detecção e acesso** - Implemente a codificação para verificar a existência de cameras e solicitar acesso ao hardware.
2. **Preview da imagem** – Codificação da classe que fará a pré-visualização da imagem da camera. Esta classe deve estender a classe-mãe SurfaceView e implementar sua interface SurfaceHolder
3. **Construção do layout de pré-visualização** - Layout para visualização do preview e exibição dos controles da camera (zoom, filtros, botões de ação, etc).
4. **Implementação de *Listeners*** - Codificação dos *listeners* para a interface de controle da aplicação: captura as ações do usuário
5. **Captura and armazenamento dos arquivos**
6. **Liberação da camera** – Após o uso do hardware, a aplicação deve liberar todos recursos alocados para que outros processo s possam utilizar a camera.

Opção 2: Passo 2 – Preview Interface

```
import android.graphics.Canvas;
import android.graphics.Rect;
import android.view.Surface;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.hardware.Camera;
import android.content.*;
import java.io.IOException;
import android.util.Log;


public class CameraPreview extends SurfaceView implements SurfaceHolder.Callback {


private SurfaceHolder mHolder;
private Camera mCamera;
```

Opção 2: Passo 2 – Preview Interface

```
public CameraPreview(Context context, Camera camera) {
    super(context);
    mCamera = camera;

    // Install a SurfaceHolder.Callback so we get notified when the
    // underlying surface is created and destroyed.
    mHolder = getHolder();
    mHolder.addCallback(this);
    // deprecated setting, but required on Android versions prior to 3.0
    mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
  }
 public void surfaceCreated(SurfaceHolder holder) {
    // The Surface has been created, now tell the camera
        //where to draw the preview.
    try {
      mCamera.setPreviewDisplay(holder);
      mCamera.startPreview();
    } catch (IOException e) {
      Log.d(TAG, "Error setting camera preview: " + e.getMessage());
    }
  }
```

Opção 2: Passo 2 – Preview Interface

```
public void surfaceDestroyed(SurfaceHolder holder) {
    // empty. Take care of releasing the Camera preview in your
     activity.
    if (mCamera != null)
        mCamera.stopPreview();
  }
public void surfaceChanged(SurfaceHolder holder, int format,
     int w, int h) {
    // If your preview can change or rotate, take care of those
     events here.
    // Make sure to stop the preview before resizing or
     reformatting it.
    if (mHolder.getSurface() == null){
      // preview surface does not exist
      return;
    }
    // stop preview before making changes
    try {
       mCamera.stopPreview();
    } catch (Exception e){
      // ignore: tried to stop a non-existent preview
    }
     // make any resize, rotate or reformatting
    changes here
try {
        mCamera.setPreviewDisplay(mHolder);
        mCamera.startPreview();

     } catch (Exception e){
    //     Log.d(TAG, "Error starting camera
    preview: " + e.getMessage());
     }
  }

}
```

Exercícios