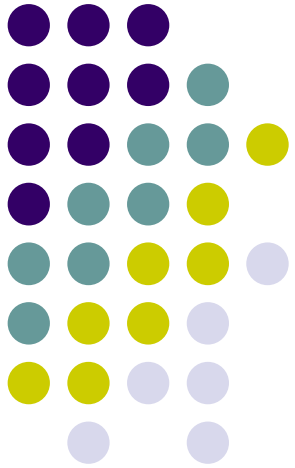
Εφαρμοσμένα Πληροφοριακά Συστήματα

11



Διαχείριση κάμερας

Ιωάννης Έλληνας

Λήψη Εικόνας

- Λήψη εικόνων με την κλάση *Camera*. Άδεια χρήσης και ιδιότητες της κάμερας στο αρχείο μανιφέστου της εφαρμογής.

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

- Άλλες ιδιότητες που μπορεί να δηλωθούν είναι:
  - *android.hardware.camera.flash* – Χρήση του φλας της συσκευής.
  - *android.hardware.camera.front* – Χρήση της πρόσθιας κάμερας.

- Λήψη φωτογραφίας με τη δημιουργία μιας πρόθεσης με ενέργεια “MediaStore.ACTION_IMAGE_CAPTURE”, ενσωματώνοντας σε αυτήν ένα αρχείο σαν “MediaStore.EXTRA_OUTPUT”. Η πρόθεση εκδηλώνεται με τη μέθοδο *startActivityForResult()* και τα αποτελέσματα αναμένονται στη μέθοδο επανάκλησης *onActivityResult()*.

- Δημιουργία αρχείου στην εσωτερική κάρτα μνήμης *sdcard0* για αποθήκευση. Εκδήλωση πρόθεσης για λήψη της φωτογραφίας, ενσωμάτωση στην πρόθεση της ανάλυσης του ονόματος του αρχείου και καλείται η δραστηριότητα του συστήματος με αυτήν την ενέργεια και κώδικα αίτησης PICTURE_CODE, ο οποίος είναι ένας ακέραιος αριθμός.

```
File imgFile = new File(Environment.getExternalStorageDirectory().getAbsolutePath() + "/photo.jpg");
Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
Uri fileUri = Uri.fromFile(imgFile);
intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);
startActivityForResult(intent, PICTURE_CODE);
```

- Μόλις ληφθεί η φωτογραφία:

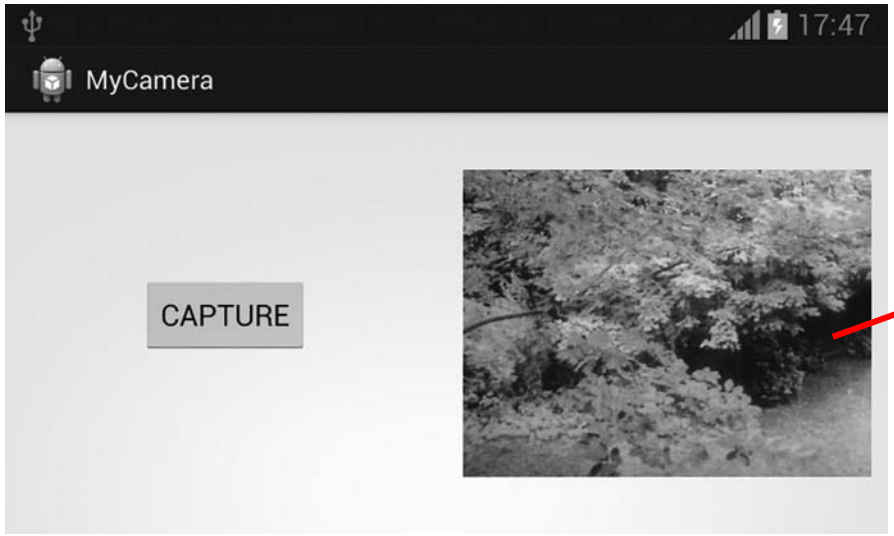
```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
 if (requestCode == PICTURE_CODE) {
   if (resultCode == RESULT_OK) {
    ImageView img = (ImageView) findViewById(R.id.imageView1);
    Uri fileUri_2 = Uri.fromFile(imgFile);
    img.setImageURI(fileUri_2);
   }
 }
}
```

- Με την ενεργοποίηση της κάμερας προκαλείται επανενεργοποίηση της δραστηριότητας με αποτέλεσμα το *fileUri* να μηδενιστεί και να χαθεί η διαδρομή του αρχείου με την εικόνα. Έτσι, ο κώδικας που υπάρχει στη μέθοδο *onActivityResult()* δίνει το σφάλμα "*NullPointerException*". Έτσι, η παράμετρος *data* επιστρέφει μηδενική και προσδιορίζουμε τη διαδρομή του αρχείου που περιέχει την εικόνα με τη βοήθεια του *fileUri_2* αποθηκεύοντας και ανακαλώντας την τιμή του *fileUri* με τις μεθόδους *onSaveInstanceState()* και *onRestoreInstanceState().*

```
@Override
protected void onSaveInstanceState(Bundle outState) {
  super.onSaveInstanceState(outState);
  // save file url
  outState.putParcelable("file_Uri", fileUri);
}

@Override
protected void onRestoreInstanceState(Bundle savedInstanceState) {
  super.onRestoreInstanceState(savedInstanceState);
  // recall file url
  fileUri_2 = savedInstanceState.getParcelable("file_Uri");
}
```

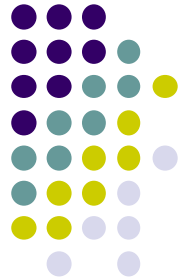
- Η μέθοδος αποθήκευσης και ανάκλησης πραγματοποιείται με τη μέθοδο "κλειδί-τιμή" μέσω των μεθόδων *putParcelable()* και *getParcelable().*

Παράδειγμα λήψης (MyCamera)



Αρχείο

Λειτουργία φωτογραφικής μηχανής

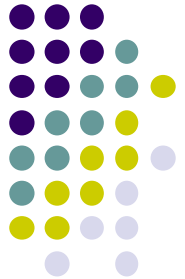


- Το περιγραφικό αρχείο *activity_main.xml* περιέχει ένα ImageView για την απεικόνιση της φωτογραφίας και ένα πλήκτρο για τη λήψης της ενώ ο προσανατολισμός είναι οριζόντιος.
- Το αρχείο μανιφέστου περιέχει την άδεια χρήσης της κάμερας, τη δήλωση του υλικού της κάμερας, την άδεια εγγραφής στην εξωτερική κάρτα μνήμης της συσκευής (sdcard0) η οποία βρίσκεται στο εσωτερικό της και την οθόνη σε οριζόντιο προσανατολισμό.

```
<uses-feature android:name="android.hardware.camera" />
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission. WRITE_EXTERNAL_STORAGE" />
……
<activity
  android:name="gr.mybook.mycamera.MainActivity"
  android:label="@string/app_name"
  android:screenOrientation="landscape" >
……
```

```
package gr.mybook.mycamera;

import ……

public class MainActivity extends Activity {
  private static final int PICTURE_CODE=1;

  private ImageView img;
  private Uri fileUri;
  private File imgFile;

  //directory for images
  private static final String DIRECTORY_NAME = "MyPhotos";

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

    img=(ImageView) findViewById(R.id.imageView1);
  }

  public void capture(View view) {
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    // store file in root directory of sdcard0
    // imgFile = new File(Environment.getExternalStorageDirectory(). getAbsolutePath() + "/photo.jpg");

    // create folder in root of sdcard0
    // File myDirectory = new File(Environment.getExternalStorageDirectory() + File.separator + DIRECTORY_NAME);
```

```
// create folder inside folder Pictures of sdcard0
    File myDirectory = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES), DIRECTORY_NAME);
    // Create directory if it does not exist
    if (!myDirectory.exists()) {
      if (!myDirectory.mkdirs()) {
        Log.d(DIRECTORY_NAME, "Creation Failure "+ DIRECTORY_NAME);
      }
    }

    // date and time stamp for image name
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss", Locale.getDefault()).format(new Date());
    // filename
    imgFile = new File(myDirectory.getPath() + File.separator + "IMG_" + timeStamp + ".jpg");
    fileUri=Uri.fromFile(imgFile);
    intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);
    startActivityForResult(intent, PICTURE_CODE);
  }

  @Override
  protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    // save file url
    outState.putParcelable("file_Uri", fileUri);
  }

  @Override
  protected void onRestoreInstanceState(Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
    // recall file url
    fileUri = savedInstanceState.getParcelable("file_Uri");
  }
```



```
@Override
 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
  if (requestCode == PICTURE_CODE) {
   if (resultCode == RESULT_OK) {
    try {
     // bimatp factory
     BitmapFactory.Options options = new BitmapFactory.Options();
     // subsampling image
     // options.inSampleSize = 4;
     final Bitmap bitmap = BitmapFactory.decodeFile(fileUri.getPath(), options);
     img.setImageBitmap(bitmap);

     // fileUri = Uri.fromFile(imgFile);
     // img.setImageURI(fileUri);
    } catch(NullPointerException e) {}
   }
  }
 }
}
```

Έλεγχος κάμερας

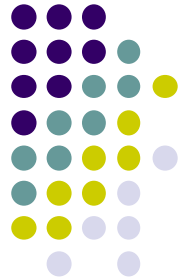
- Η πρόσβαση σε μια κάμερα της κινητής συσκευής είναι η δημιουργία ενός στιγμιοτύπου της κλάσης *Camera* στη μέθοδο *onResume()* και το άνοιγμα της κάμερας σε ένα ξεχωριστό νήμα δίνοντας την ταυτότητά της. Στη μέθοδο *onPause()* πρέπει να ελευθερώνεται η κάμερα, αφού πρώτα σταματήσει η προεπισκόπηση (preview), τη λειτουργία της οποίας σηματοδοτεί η σημαία *fPreview*.

```
@Override
protected void onResume() {
  super.onResume();
  mCamera = Camera.open(id);
}
@Override
protected void onPause() {
  if (fPreview) {
    mCamera.stopPreview();
  }
  if (mCamera != null) {
    mCamera.release();
    mCamera = null;
    fPreview = false;
    super.onPause();
  }
}
```

- Η προεπισκόπηση γίνεται με την κλάση *SurfaceView* και τη διασύνδεση *SurfaceHolder.Callback*, η οποία μεταφέρει τα δεδομένα από το υλικό της συσκευής φωτογράφησης στην εφαρμογή και ενημερώνει για τις μεταβολές που πρόκειται να γίνουν στην *SurfaceView* με μεθόδους επανάκλησης. Επομένως πρωταρχικά χρειάζεται μια οθόνη η οποία να υποστηρίζει την κλάση προεπισκόπησης, η οποία χρησιμοποιείται σαν ένας καμβάς απεικόνισης όλων των γραφικών. Το περιγραφικό αρχείο που θα φιλοξενήσει την εικόνα σε ολόκληρη την οθόνη μπορεί να έχει την εξής μορφή:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent" >
  <SurfaceView
    android:id="@+id/camerapreview"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"/>
</LinearLayout>
```

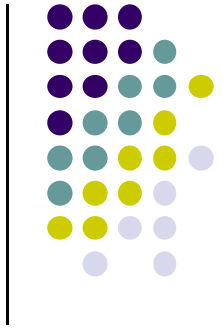
- Επειδή η προεπισκόπηση και λήψη φωτογραφιών είναι πιο βολική με οριζόντια τοποθέτηση της κάμερας, πρέπει αυτό να δηλωθεί στο αρχείο μανιφέστου.



- Η εγγραφή της διασύνδεσης *SurfaceHolder.Callback* και η δήλωση των buffers για την αποθήκευση των δεδομένων λήψης της κάμερας πραγματοποιείται ως εξής:

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

  mSurfaceView = (SurfaceView) findViewById(R.id.camerapreview);
  mHolder = mSurfaceView.getHolder();
  mHolder.addCallback(surfaceCallback);
  mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
}
```

- Η προεπισκόπηση μπορεί να αποτελεί ξεχωριστή κλάση η οποία είναι υποκλάση της *SurfaceView* με τη διασύνδεση *SurfaceHolder.Callback* ή να βρίσκεται στην ίδια δραστηριότητα με τις μεθόδους επανάκλησης *surfaceCreated()*, *surfaceChanged()*, *surfaceDestroyed()*.
- Η μέθοδος επανάκλησης *surfaceCreated()* συνδέει την κάμερα με την *SurfaceView* μέσω του *SurfaceHolder* και εκκινεί την προεπισκόπηση, η μέθοδος *surfaceChanged()* ενεργοποιείται όταν μεταβληθούν οι διαστάσεις της επιφάνειας προεπισκόπησης, ενώ η μέθοδος επανάκλησης *surfaceDestroyed()* τερματίζει όποια επισκόπηση βρίσκεται σε εξέλιξη.



```
SurfaceHolder.Callback surfaceCallback=new SurfaceHolder.Callback() {

  public void surfaceCreated(SurfaceHolder holder) {
    try {
      mCamera.setPreviewDisplay(holder);
      mCamera.startPreview();
    }   catch (IOException exception) { }
  }

  public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
    // stop preview before changes
    If (fPreview) {
      mCamera.stopPreview();
      fPreview = false;
    }
    // make any changes in SurfaceView and restart preview
    if (mCamera != null) {
      try {
        mCamera.setPreviewDisplay(holder);
        mCamera.startPreview();
        fPreview = true;
      } catch (Exception e) {                  }
    }
  }

  public void surfaceDestroyed(SurfaceHolder holder) {
    mCamera.stopPreview();
  }
};
```

- Η αλλαγή των ρυθμίσεων της κάμερας, μεταβολή του μεγέθους της προεπισκόπησης, του επιπέδου zoom και άλλων, πραγματοποιείται με επέμβαση στη μέθοδο επανάκλησης *surfaceChanged()* ως εξής:

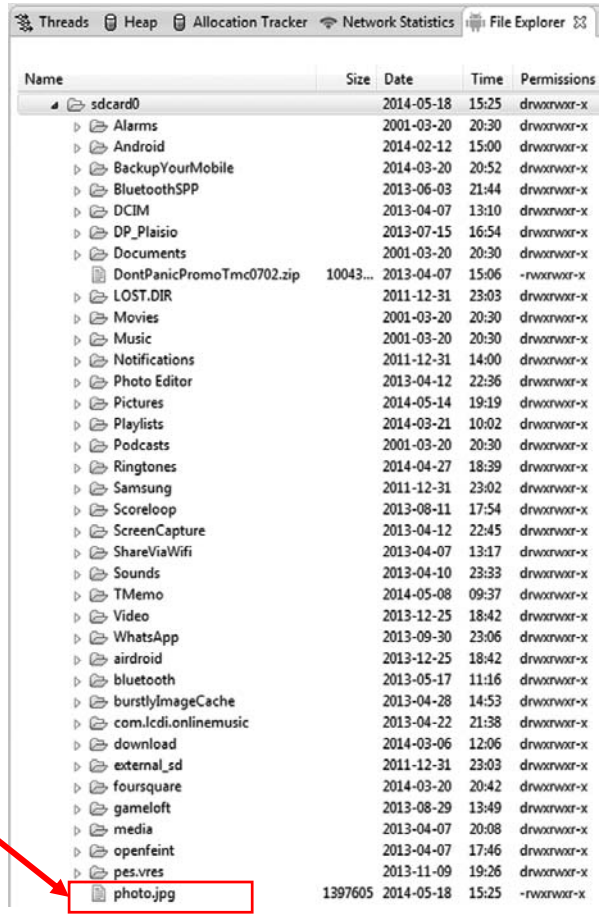
```
public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {
  Camera.Parameters parameters = mCamera.getParameters();

  // TODO any change of display size
  Camera.Size mPreviewSize = changePreviewSize(width, height, parameters);

  parameters.setPreviewSize(mPreviewSize.width, mPreviewSize. height);
  mCamera.setParameters(parameters);
  mCamera.startPreview();
  fPreview = true;
}
```

- Αφού γίνει η προεπισκόπηση, μπορεί να ληφθεί μια φωτογραφία με τη μέθοδο *Camera.takePicture()*. Η μέθοδος αυτή προκαλεί μια ασύγχρονη λήψη της εικόνας προκαλώντας την κλήση διαφόρων μεθόδων επανάκλησης καθώς η λήψη της εικόνας βρίσκεται σε εξέλιξη. Η μορφή της είναι:

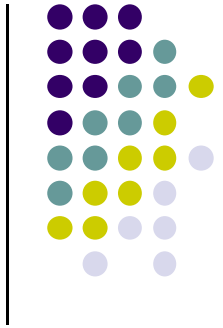
takePicture (Camera.ShutterCallback shutter, Camera.PictureCallback raw, Camera.PictureCallback postview, Camera.PictureCallback jpeg)

- **ShutterCallback** – Η μέθοδος καλείται αφού έχει ληφθεί η εικόνα. Μπορεί να χρησιμοποιηθεί από τον χρήστη για τη δημιουργία ενός ήχου λήψης της εικόνας.
- **PictureCallback raw** – Προκαλείται όταν υπάρχουν τα δεδομένα της εικόνας και παρέχει έναν πίνακα από bytes. Τα δεδομένα θα είναι μηδενικά αν δεν έχει δημιουργηθεί buffer κατάλληλου μεγέθους.
- **PictureCallback postview** – Προκαλείται όταν μια προ-επισκοπημένη εικόνα είναι διαθέσιμη, έτσι ο χρήστης να πραγματοποιήσει κάποια ενέργεια όπως να προσδιορίσει κάποιο αντικείμενο πάνω στην εικόνα.
- **PictureCallback jpeg** - Προκαλείται όταν υπάρχουν τα δεδομένα της εικόνας σε συμπιεσμένη μορφή και παρέχει έναν πίνακα από bytes.

Παράδειγμα ελέγχου
(MyCameraControl)



- Το περιγραφικό αρχείο *activity_main.xml* περιέχει ένα *SurfaceView* που καταλαμβάνει ολόκληρη την οθόνη.



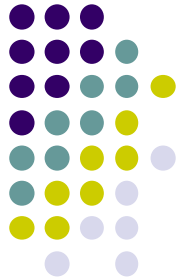
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent" >
  <SurfaceView
    android:id="@+id/camerapreview"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
</LinearLayout>
```

- Το νέο περιγραφικό αρχείο, *control.xml*, περιγράφει το πλήκτρο λήψης της φωτογραφίας. Το πλήκτρο αυτό τοποθετείται στο κάτω δεξιό μέρος της οριζόντιας οθόνης.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:gravity="bottom" >
  <Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text=" CAPTURE "
    android:layout_gravity="right"
    android:layout_margin="10dp"
    android:onClick="capture"/>
</LinearLayout>
```

- Αρχείο μανιφέστου:

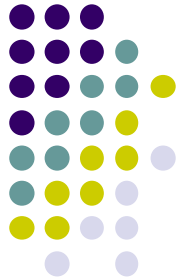
```
<uses-feature android:name="android.hardware.camera" />
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
……
<activity
  android:name="gr.mybook.mycamera.MainActivity"
  android:label="@string/app_name"
  android:screenOrientation="landscape" >
……
```

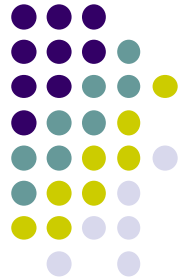


```
package gr.mybook.mycameracontrol;
import ……

public class MainActivity extends Activity {
  Camera mCamera;
  SurfaceView mSurfaceView;
  SurfaceHolder mHolder;
  boolean fPreview = false;
  LayoutInflater controlInflater = null;
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    getWindow().setFormat(PixelFormat.UNKNOWN);
    mSurfaceView = (SurfaceView)findViewById(R.id.camerapreview);
    mHolder = mSurfaceView.getHolder();
    mHolder.addCallback(surfaceCallback);
    mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

    controlInflater = LayoutInflater.from(getBaseContext());
    View viewControl = controlInflater.inflate(R.layout.control, null);
    LayoutParams layoutParamsControl = new LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.FILL_PARENT);
    this.addContentView(viewControl, layoutParamsControl);
  }

 @Override
  protected void onResume() {
    super.onResume();
    mCamera = Camera.open();
    mCamera.startPreview();
    fPreview = true;
  }
```

```
@Override
protected void onPause() {
  if (fPreview) {
    mCamera.stopPreview();
  }
  mCamera.release();
  mCamera = null;
  fPreview = false;
  super.onPause();
}

public void capture(View view) {
  mCamera.takePicture(myShutterCallback, myPictureCallback_RAW, myPictureCallback_JPG);
}

ShutterCallback myShutterCallback = new ShutterCallback() {
  @Override
  public void onShutter() {
    // TODO Auto-generated method stub
}};

PictureCallback myPictureCallback_RAW = new PictureCallback() {
  @Override
  public void onPictureTaken(byte[] arg0, Camera arg1) {
    // TODO Auto-generated method stub
}};
```

```
PictureCallback myPictureCallback_JPG = new PictureCallback() {
  @Override
  public void onPictureTaken(byte[] arg0, Camera arg1) {
    new PictureTask().execute(arg0);
    mCamera = arg1;
    mCamera.startPreview();
    fPreview=true;
 }};

class PictureTask extends AsyncTask<byte[], String, String> {
  @Override
  protected String doInBackground(byte[]... jpeg) {
    File photo=new File(Environment.getExternalStorageDirectory(). getAbsolutePath() + "/photo.jpg");

    try {
      FileOutputStream fos=new FileOutputStream(photo.getPath());
      fos.write(jpeg[0]);
      fos.close();
    }
    catch (IOException e) { }
    return(null);
  }
}

SurfaceHolder.Callback surfaceCallback=new SurfaceHolder. Callback() {
  public void surfaceCreated(SurfaceHolder holder) {
    // no-op -- wait until surfaceChanged()
  }
```





```
public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
    if(fPreview) {
      mCamera.stopPreview();
      fPreview = false;
    }
    if (mCamera != null) {
      try {
        mCamera.setPreviewDisplay(mHolder);
        mCamera.startPreview();
        fPreview = true;
      } catch (Exception e) {}
    }
  }

  public void surfaceDestroyed(SurfaceHolder holder) {
    // no-op
  }
 };
}
```