

2019 学年 第 二 学期

# 实 验 报 告

课程名称 Android 应用编程实验

系（部） 信息工程系

专业班级 电子信息科学与技术 17-2

学生姓名 张厚今

学生学号 201723010237

实验名称	多媒体应用-拍照		
实验目的	设计一个简易照相机		
实验内容	利用 Camera 类设计一个简易照相机,通过 SurfaceView 组件实现取景预览,拍照保存等功能,拍照保存时按 camera+序号.jpg 的方式保存图片,序号自动增加,确保拍照保存不会覆盖前面的照片。		
实验仪器设备	电脑、Android Studio、Android 手机		
姓 名	张厚今	组 别	
同组实验者		实验日期	2019 年 5 月
指导教师	魏光村		
批阅意见 及 成 绩			
实 验 步 骤	1 实验原理 2 实验过程记录 3 实验过程中存在的问题及解决方案 4 实验结果 5 源代码 6 待解决问题和设想 7 实验总结 8 部分参考资料		

## 目录

一、实验原理.....	1
二、实验过程记录.....	1
2.1 布局文件.....	1
2.2 控制文件.....	1
2.2.1 实例化对象.....	1
2.2.2 重载构造函数.....	1
2.2.3 编写 mClick 类.....	2
2.2.4 编写 jpegCallback 类.....	3
2.2.5 重载 surfaceCreated 方法.....	3
2.2.6 重载 surfaceChanged 方法.....	4
2.2.7 重载 surfaceDestroyed 方法.....	5
2.3 添加权限.....	5
2.4 安装 APK 测试.....	6
三、实验中存在的问题及解决方案.....	6
3.1 预览倾斜问题及解决过程.....	6
3.2 文件命名问题及解决过程.....	9
3.2.1 问题分析.....	9
3.2.2 测试程序.....	9
3.2.3 安卓测试.....	13
3.3 预览画面卡顿问题及解决过程.....	15
四、实验结果.....	17
五、源代码.....	17
5.1 布局文件.....	17
5.2 程序控制文件.....	18
六、待解决问题和设想.....	22
6.1 待解决问题.....	22
6.2 设想.....	23
七、实验总结.....	23
八、部分参考资料.....	23

## 一、实验原理

该相机程序，需要应用 `surfaceview` 组件来预览摄像头拍摄到的景物，再使用回调接口 `surfaceholder.callback` 监控取景视图。使用照片服务类 `Camera` 实现拍照功能，并通过 `imageView` 组件显示。最后通过 `onPicturetakn` 方法，将拍摄的照片保存至手机。编写 `getFile` 和 `checkFileName` 方法，用来对文件名进行处理，实现了实验对于文件名的要求。

## 二、实验过程记录

### 2.1 布局文件

为实现基本的相机程序效果，使用最简单的线性布局，通过组建的嵌套生成需要的布局。首先指定最外层的线性布局对齐方式为垂直布局，之后再编写 `textView` 组件显示程序的标题，然后嵌套一层水平对齐方式的线性布局，在其中加入两个 `button` 组件用于防止“拍照”和“退出”按钮。之后在第一层线性布局中加入 `ImageView` 组件用于相片的显示，最后加入 `surfaceView` 组件用于取景预览。

### 2.2 控制文件

#### 2.2.1 实例化对象

在 `MainActivity.java` 文件中修改文件，以实现程序的控制。首先在公共类“`MainActivity`”扩展“`Activity`”之后，添加“`implements SurfaceHolder.Callback`”，实现 `Callback` 接口，以处理取景预览功能。之后，如下图所示实例化所需的对象，并声明用于保存图片文件的路径。

```
public class MainActivity extends Activity implements SurfaceHolder.Callback{
    Camera mCamera = null;
    SurfaceView surfaceView;
    SurfaceHolder holder;
    ImageView mImageView;
    Button cameraBtn, exitBtn;
    String path = "/sdcard/test/camera.jpg";    //图片保存路径
```

图 2.1

在上图 2.1 中，分别实例化了用于拍照功能的“`Camera`”对象“`mCamera`”、用于取景预览的“`SurfaceView`”对象“`surfaceView`”、用于回调图片参数的“`SurfaceHolder`”对象“`holder`”、用于显示图片的“`ImageView`”对象“`mImageView`”、用于拍照退出的按钮对象“`cameraBtn`”和“`exitBtn`”，以及用于记录图片保存路径的对象“`path`”。

#### 2.2.2 重载构造函数

对构造函数进行修改，使其实现“关联布局文件和控制文件，注册回调监听器”的功能。如图 2.2 所示：

```
/**
 * Override the onCreate
 * function: 重载构造函数，关联布局文件和控制文件，注册回调监听器
 */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //关联ID
    mImageView = (ImageView)findViewById(R.id.imageView1);
    cameraBtn = (Button)findViewById(R.id.btn1);
    exitBtn = (Button)findViewById(R.id.btn2);
    cameraBtn.setOnClickListener(new mClick()); //设置监听事件
    exitBtn.setOnClickListener(new mClick());
    surfaceView = (SurfaceView)findViewById(R.id.surfaceView1);
    //System.out.println("begin to holder...");
    holder = surfaceView.getHolder();
    holder.addCallback(this);
    holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
}
```

图 2.2

首先，通过 findViewById 方法关联图 1 中的相关组件，并为“cameraBtn”和“exitBtn”分别设置监听事件，最后设置 SurfaceHolder 对象的相关内容。创建 SurfaceHolder 对象“holder”，注册回调监听器并设置 SurfaceHolder 的类型。

### 2.2.3 编写 mClick 类

为按钮的监听事件编写 mClick 类，以实现拍照和退出功能。如图 2.3 所示：

```
/**
 * class: mClick
 * function: 设置按键监听事件，对应拍照和退出按钮
 */
class mClick implements OnClickListener{
    @Override
    public void onClick(View v) {
        if (v == cameraBtn){
            mCamera.takePicture( shutter: null, raw: null, new jpegCallback()); //拍照
        }
        else if (v == exitBtn){
            exit(); //退出程序
        }
    }
}

void exit(){
    mCamera.release();
    mCamera = null;
}
```

图 2.3

上图中，构造了一个继承于 `OnClickListener` 的 `mClick` 类。通过判断点击按钮传入的参数 `v`，可以对拍照和退出进行区分，拍照使用 `takePicture` 方法，退出则调用 `exit()` 函数，对相机资源进行重置。

#### 2.2.4 编写 `jpegCallback` 类

在图 3 编写的 `mClick` 类中，`takePicture` 方法用到了获取照片事件的回调接口 `jpegCallback`，现在需要对 `jpegCallback` 类进行编写。如图 2.4 所示：

```
/**
 * class: jpegCallback
 * function: 实现拍照和保存图片功能
 */
class jpegCallback implements PictureCallback{
    @Override
    public void onPictureTaken (byte[] data, Camera camera){
        Bitmap bitmap = BitmapFactory.decodeByteArray(data, 0, data.length);
        try{
            BufferedOutputStream outputStream = new BufferedOutputStream(new FileOutputStream(path));
            bitmap.compress(Bitmap.CompressFormat.JPEG, 80, outputStream);
            outputStream.flush();
            outputStream.close();
            mImageView.setImageBitmap(bitmap); //在ImageView显示拍照的图片
        }catch (Exception e){
            Log.e( tag: "error", e.getMessage());
        }
    }
}
```

图 2.4

在上图中，编写了继承于 `Camera. PictureCallback` 的类“`jpegCallback`”，通过重载 `onPictureTaken` 函数，实现了拍照、显示图片以及保存图片的功能。

#### 2.2.5 重载 `surfaceCreated` 方法

创建相机对象之后，会默认调用三个构造函数，分别为 `surfaceCreated`、`surfaceChanged` 和 `surfaceDestroyed`，首先重载 `surfaceCreated` 函数，如图 2.5 所示。

```

/**
 * Override the surfaceCreated
 * function: 创建相机时触发，开启相机预览功能
 */
@Override
public void surfaceCreated(SurfaceHolder holder){
    if (mCamera != null){
        ReleaseCamera();    //首先释放相机资源
    }
    mCamera = Camera.open();    //开启摄像头
    //System.out.println("\n\n\nCamera.open() is OK !!!\n\n\n");

    //mCamera = android.hardware.Camera.open();
    //mCamera = Camera.open(Camera.CameraInfo.CAMERA_FACING_BACK);
    //System.out.println("Camera is OK!");
    try{
        mCamera.setPreviewDisplay(holder);    //设置相机预览
    }catch (IOException e){
        System.out.println("预览错误");
    }
}
private void ReleaseCamera()    //重置相机
{
    if(mCamera != null)
    {
        mCamera.release();
        mCamera = null;
    }
}
}
    
```

图 2.5

上图中，首先释放掉相机资源，然后开启摄像头。之后尝试进行相机预览，如果捕获到预览错误信息，则输出报错。其中的 ReleaseCamera()函数用来重置相机，释放相机资源。

### 2.2.6 重载 surfaceChanged 方法

相机画面发生变化时，将触发 surfaceChanged 方法，如下图 2.6 所示重载 surfaceChanged 方法。

```

/**
 * Override the surfaceChanged
 * function: 当画面发生改变时触发，重置相机参数
 * */
@Override
public void surfaceChanged (SurfaceHolder holder, int format, int width, int height){
    //System.out.println("Camera is going to ready...");
    initCamera(); //重置相机参数
}
//设置相机参数
public void initCamera(){
    //System.out.println("here1...");
    Camera.Parameters parameters = mCamera.getParameters();
    parameters.setPictureFormat(PixelFormat.JPEG); //照片格式
    parameters.setPreviewSize( width: 320, height: 240); //预览规格大小
    parameters.setPictureSize( width: 320, height: 240); //图片大小
    mCamera.setParameters(parameters);
    //mCamera.setDisplayOrientation(90); //设置浏览画面水平转90°
    mCamera.startPreview(); //开始预览
}

```

图 2.6

重载的 `surfaceChanged` 方法会调用 `initCamera` 函数，重新设置预览的画面。重置包括预览画面的格式、大小、预览框大小等信息。最后调用 `startPreview` 方法开始预览。

### 2.2.7 重载 `surfaceDestroyed` 方法

关闭相机时，会触发 `surfaceDestroyed` 方法，如下图 2.7 所示重载 `surfaceDestroyed` 方法。

```

/**
 * Override the surfaceDestroyed
 * function: 关闭相机时触发，空
 * */
@Override
public void surfaceDestroyed(SurfaceHolder holder){ } //消灭相机时触发

```

图 2.7

可以看到，重载的 `surfaceDestroyed` 方法并没有实现具体的功能。

## 2.3 添加权限

由于程序中用到了 SD 卡读写和相机的调用，所以需要在程序文件中声明需要申请的权限。包括 SD 卡读取、写入权限，以及相机调用权限。如下图 2.8 所示，在 `AndroidManifest.xml` 文件中添加下列信息，以申请权限。

```

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

```

图 2.8



## 2.4 安装 APK 测试

由于 AndroidStudio 模拟器在电脑上经常卡顿，所以我决定使用安卓手机进行测试。首先将编译好的项目打包成 APK 文件，然后将该 APK 文件发送至手机端安装。安装截图如图 2.9 所示，可以看到安装时会提示该 APP 将获取的手机权限。



图 2.9

点击安装即可完成 APK 文件的安装。

## 三、实验中存在的问题及解决方案

### 3.1 预览倾斜问题及解决过程

安装完成，运行该 APP 程序，点击拍照按钮，可以看到如图 3.1 所示的手机界面。其中上面的小画面是 imageView 组件显示的拍完的照片，下面的大图片是 surfaceView 组件显示的相机预览照片。

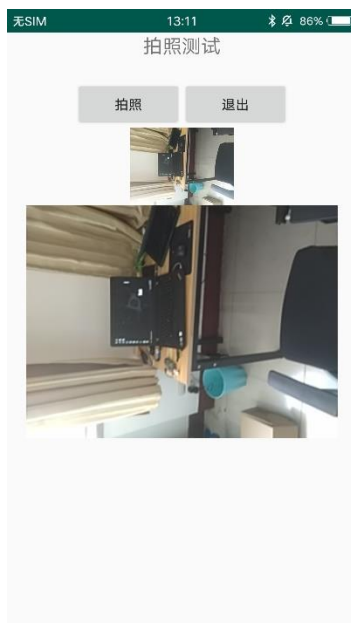


图 3.1

拍摄完成之后，推迟该 APP，并在手机自带的文件管理界面，找到 test 文件夹(如图 3.2)，这是刚刚图片拍摄之后，程序生成的文件夹，用于保存图片文件。在 test 文件夹下，可以看到刚刚生成的 camera.jpg 文件(如图 3.3 所示)，浏览该文件，可以查看之前拍摄的图片(如图 3.4)。



图 3.2

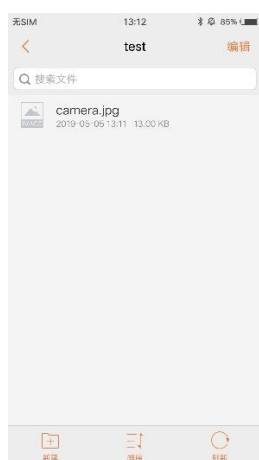


图 3.3

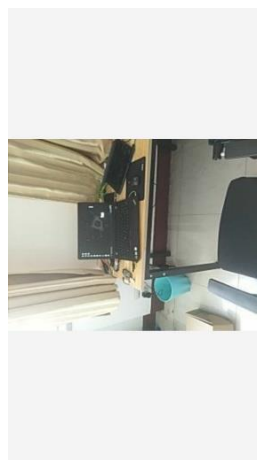


图 3.4

查看该图片，发现拍摄出来的图片，与正常视角相差了  $90^\circ$ ，而且在图片拍摄和预览过程中，图片和正常视角都是不相符的，都旋转了  $90^\circ$ 。

经过查阅资料发现，主要是由于相机传感器像素坐标信息，与手机显示屏像素坐标不相符导致的。图 3.5 展示了两像素坐标的不同之处。



图 3.5

(图片引用于: <https://blog.csdn.net/xx326664162/article/details/53350551>)

相机传感器获取到图像，就确定了这幅图像每个坐标的像素值，但是要显示到手机屏幕上，就需要按照屏幕的坐标系来显示，于是就需要将相机传感器的坐标系逆时针旋转 90 度，才能显示到屏幕的坐标系上。

在安卓程序中，提供了相关的 `setRotation()` 方法和 `setDisplayOrientation()` 方法。其中，`setRotation()` 方法的作用是将相机传感器获取到的位图坐标旋转一定的角度，而 `setDisplayOrientation()` 方法的作用是将预览时的画面旋转一定的角度。通过调用这两个方法，我们就可以实现图片的正常拍摄和预览了。如图 3.6 所示，在原有程序的基础上添加该方法，再生成 APK 文件进行测试。

```
public void initCamera(){
    //System.out.println("here1...");
    Camera.Parameters parameters = mCamera.getParameters();
    parameters.setPictureFormat(PixelFormat.JPEG); //照片格式
    parameters.setPreviewSize( width: 320, height: 240); //预览规格大小
    parameters.setPictureSize( width: 320, height: 240); //图片大小
    parameters.setRotation(90); //设置照片数据旋转90°
    mCamera.setParameters(parameters);
    mCamera.setDisplayOrientation(90); //设置浏览画面水平转90°

    mCamera.startPreview(); //开始预览
}
```

图 3.6

重新安装 APK 文件，测试运行，运行结果如图 3.7 所示。可以看到，预览的图片已经可以正常显示了。然后再到 test 文件夹中查看 camera.jpg 文件(如图 3.8)，同样，拍摄的图片也是可以正常显示了。



图 3.7



图 3.8

## 3.2 文件命名问题及解决过程

目前的程序所能实现的功能，只能拍摄一张照片，如果拍摄多张照片，由于文件名相同，后面的文件就会将前面已存在的文件覆盖掉，很不方便。本次实验要求的是“拍照保存时按 camera+序号. jpg 的方式保存照片，序号自动增加”。

### 3.2.1 问题分析

为了实现这个功能，需要在每次文件保存时，都要考虑我们需要生成的文件名是什么。如果企图保存当前程序生成的图片数量，在关闭 APP 之后，数据不易保存。所以我想的解决办法是：每次生成的文件名都需要经历两次函数，第一个函数需要返回所需文件夹中的所有文件，第二个函数需要判断指定的文件在该文件夹中是否存在，如果存在则返回一个我们需要的、正确的文件名。

### 3.2.2 测试程序

#### ① getFile()函数

为了不破坏原有的安卓程序，我在 Eclipse 中进行相关函数的编写和测试。首先编写第一个函数 getFile()，该函数需要输入文件夹的路径字符串，输出该文件夹下的所有文件名。程序

文件如图 3.9 所示：

```
package hello;
import java.io.File;
import java.util.ArrayList;
public class fix {
    private static ArrayList<String> getFile(String path){
        File file = new File(path);    // 获得指定文件对象
        File[] array = file.listFiles();    // 获得该文件夹内的所有文件
        ArrayList<String> list = new ArrayList<String>();
        for(int i=0;i<array.length;i++){
            if(array[i].isFile()){//如果是文件,只输出文件名字
                list.add(array[i].getName()+"\n");
            }
        }
        return list;
    }
    public static void main(String[] args){
        ArrayList<String> FileList;
        FileList = fix.getFile("C:\\\\Users\\shand\\Desktop\\android");
        System.out.print(FileList);
    }
}
```

图 3.9

该程序，首先通过 path 参数获取指定的文件对象，再通过 listFiles()方法列出该文件夹下，所有的文件，包括文件夹、普通文件等等，并将其保存在 array 数组中。之后逐个判断是否为普通文件，将文件名保存在 list 中，最后返回 list。

现在，将 path 指定为桌面上的 android 文件夹，调用 getFiles 方法，查看程序输出。

图 3.10 是 android 文件夹下的文件信息，图 11 是程序输出的信息。

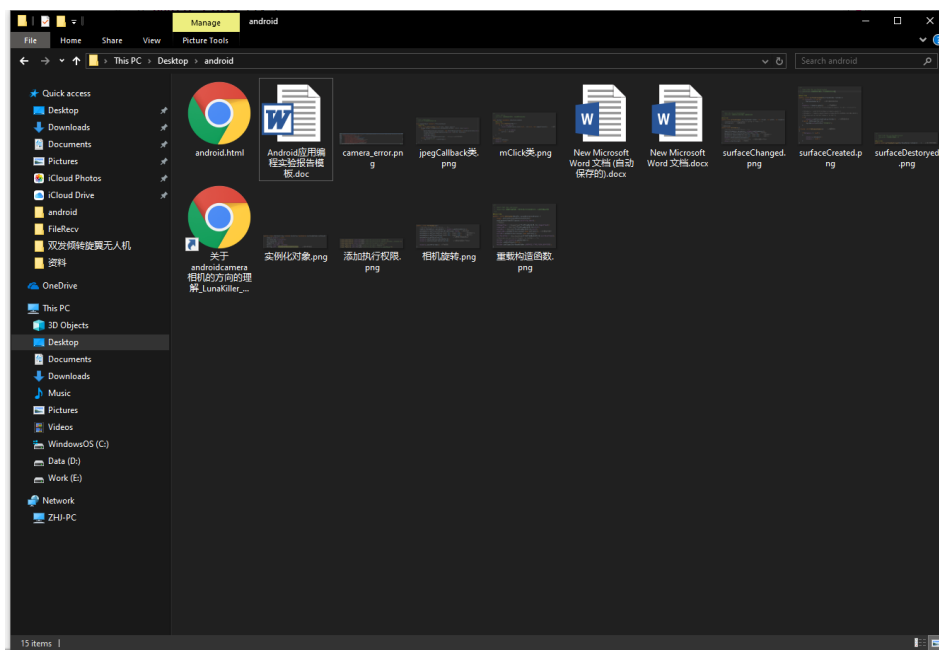
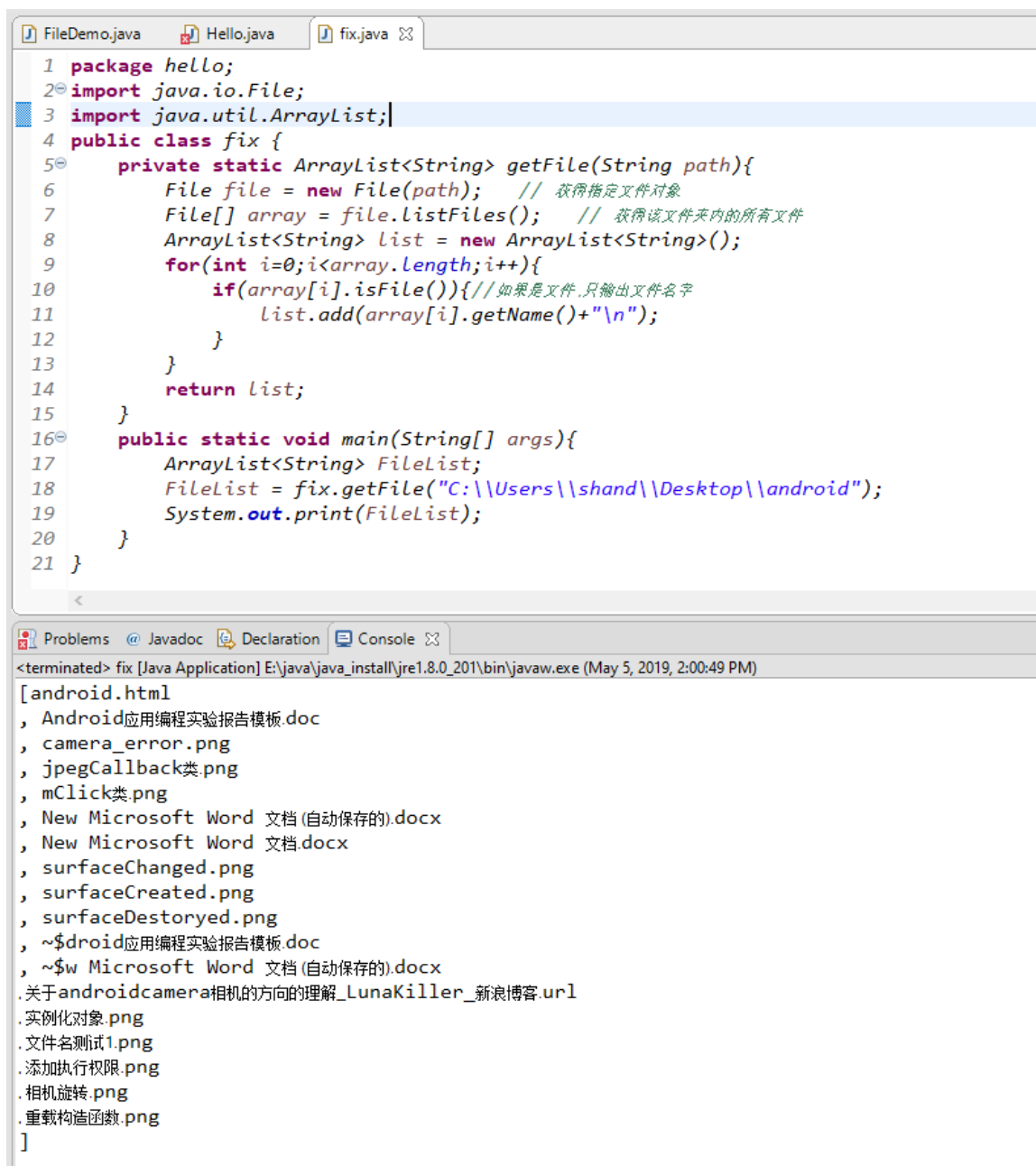


图 3.10



```

1 package hello;
2 import java.io.File;
3 import java.util.ArrayList;
4 public class fix {
5     private static ArrayList<String> getFile(String path){
6         File file = new File(path); // 获得指定文件对象
7         File[] array = file.listFiles(); // 获得该文件夹内的所有文件
8         ArrayList<String> list = new ArrayList<String>();
9         for(int i=0;i<array.length;i++){
10             if(array[i].isFile()){ // 如果是文件,只输出文件名字
11                 list.add(array[i].getName()+"\n");
12             }
13         }
14         return list;
15     }
16     public static void main(String[] args){
17         ArrayList<String> fileList;
18         fileList = fix.getFile("C:\\Users\\shand\\Desktop\\android");
19         System.out.print(fileList);
20     }
21 }
    
```

```

<terminated> fix [Java Application] E:\java\java_install\jre1.8.0_201\bin\javaw.exe (May 5, 2019, 2:00:49 PM)
[android.html
, Android应用编程实验报告模板.doc
, camera_error.png
, jpegCallback类.png
, mClick类.png
, New Microsoft Word 文档 (自动保存的).docx
, New Microsoft Word 文档.docx
, surfaceChanged.png
, surfaceCreated.png
, surfaceDestoryed.png
, ~$droid应用编程实验报告模板.doc
, ~$w Microsoft Word 文档 (自动保存的).docx
, 关于androidcamera相机的方向的理解_LunaKiller_新浪博客.url
, 实例化对象.png
, 文件名测试1.png
, 添加执行权限.png
, 相机旋转.png
, 重载构造函数.png
]
    
```

图 3.11

可以看到，除了临时文件，普通文件均可以正常输出。

## ② checkFileName 方法

下面实现 checkFileName 函数，该方法的作用是：在文件名字符串中，判断所需要的字符串是否存在，如果存在则返回原文件名；不存在则重新调用，利用 index 参数，实现文件名的递增过程。最后返回一个正确的、合理的文件名。

checkFileName 方法的实现过程是：首先将传入的文件名参数根据“.”进行分割，分成“.”之前的字符串+起始索引数字 index+“.”字符+“.”之后的字符串，然后判断整个文件名是否包含于长文件名参数中，如果存在则再次调用该方法，同时起始索引数字加一；如果不存

在，就直接返回上面连接好的文件名。

现在将 `checkFileName` 方法和 `getFile` 进行合并，通过 `main` 函数进行调用测试。首先在桌面的 `android` 目录下新建一个 `mydir` 目录，在 `mydir` 目录下新建一个 `test.txt` 文件，然后运行 `java` 程序进行测试。

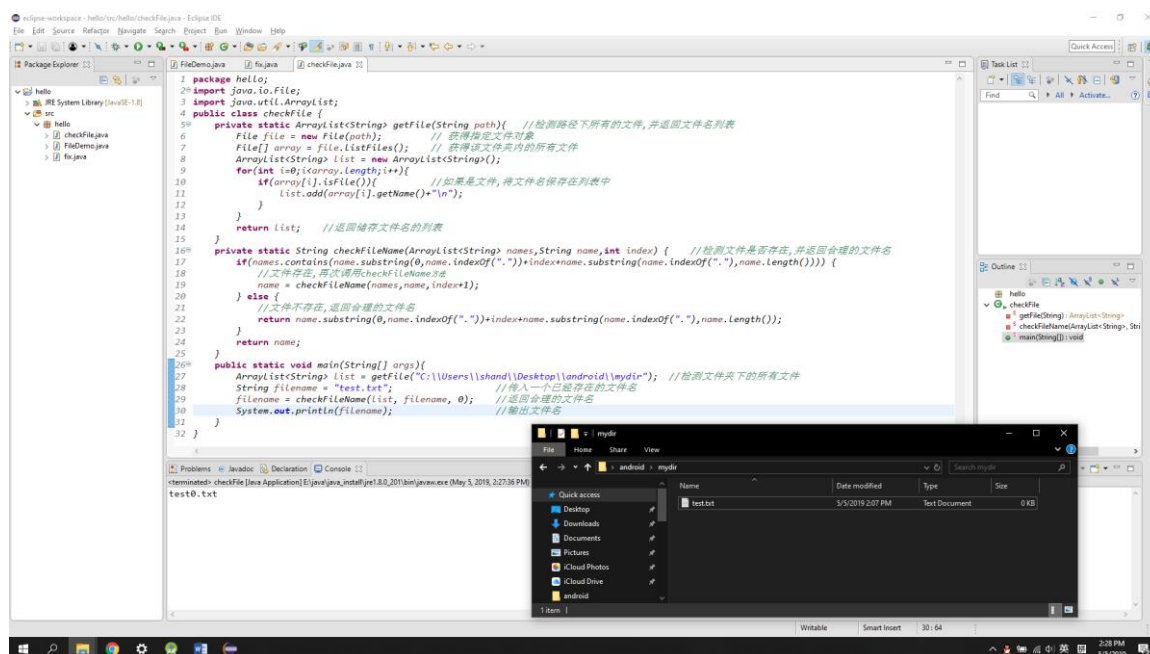


图 3.12

上图 3.12 展示了当前的 `mydir` 目录以及 Java 程序的运行结果。由于起始索引参数 `index` 设置的是 0，而文件夹下只有一个 `test.txt` 文件，所以程序返回的结果“`test.txt`”是正确的。为了测试的准确性，我在 `mydir` 目录下又新建了一个 `test0.txt` 文件，如果程序功能正常，应该是会返回 `test1.txt`。下面运行程序测试：



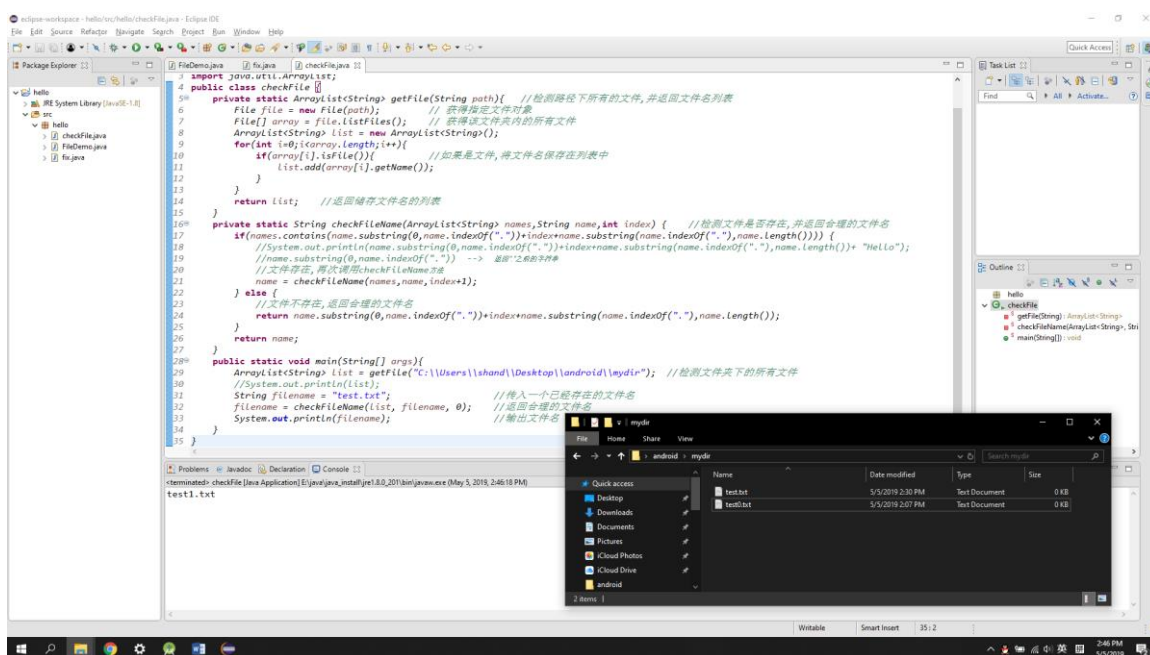


图 3.13

可以看到，程序运行结果正确。

### 3.2.3 安卓测试

现在可以放心地将代码移植到 AndroidStudio 程序中去了。在原有的 onPictureTaken() 方法的基础上，增加红色框中的内容，如图 3.14 所示。其含义为：通过 getFile 方法查找手机 "/sdcard/test" 目录下的所有文件，将结果返回至 list 中。然后通过调用 checkFileName 方法，检查 list 中是否有符合 filename 的文件名(filename 已在之前定义)，checkFileName 方法会返回合理的文件名，再通过字符串连接，将完整的路径保存到 path 字符串中，作为完整的文件名路径。

为了更清晰的显示文件保存的路径和文件名信息，我在布局文件中增加了一个 textview 组件，在 path 生成之后，将 path 输出到 textview 显示，这样就会对文件的路径信息更加直观。

最后，将 getFile 和 checkFileName 方法都移植到 java 文件中，就可以运行测试了。



```
public void onPictureTaken (byte[] data, Camera camera){
    Bitmap bitmap = BitmapFactory.decodeByteArray(data, 0, data.length);
    ArrayList<String> list = getFile( path: "/sdcard/test");
    path = "/sdcard/test/" + checkFileName(list, filename, index: 0);
    textView.setText("文件保存路径: " + path);
    try{
        BufferedOutputStream outStream = new BufferedOutputStream(new FileOutputStream(path));
        bitmap.compress(Bitmap.CompressFormat.JPEG, quality: 80, outStream);
        outStream.flush();
        outStream.close();
        mImageView.setImageBitmap(bitmap); //在ImageView显示拍照的图片
        mCamera.startPreview();
    }catch (Exception e){
        Log.e( tag: "error", e.getMessage());
    }
}
```

图 3.14

同样重新生成 APK 文件测试运行，进行多次拍摄，下面是我拍摄第 8 次时的界面截图(图 3.15)。

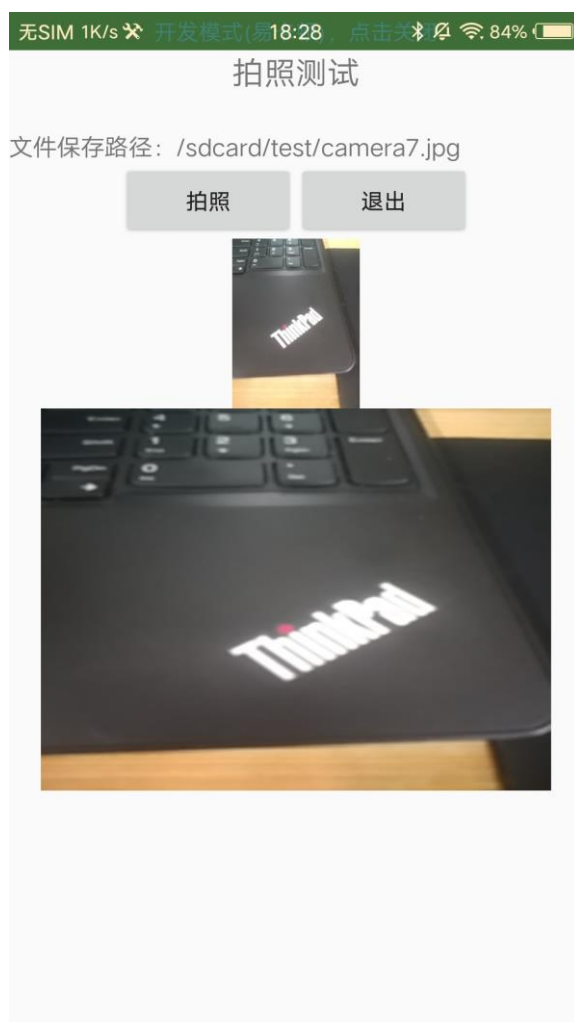


图 3.15

因为文件名起始索引为 0，所以第 8 次生成的文件名应该是 camera7.jpg，可以从 APP 界面中看到文件的保存路径。现在在 test 文件夹下，会有 8 张我刚刚拍摄的图片，命名方式为

camera+i。图 3.16 展示了 test 文件夹下的文件信息，文件的保存也是正常的。

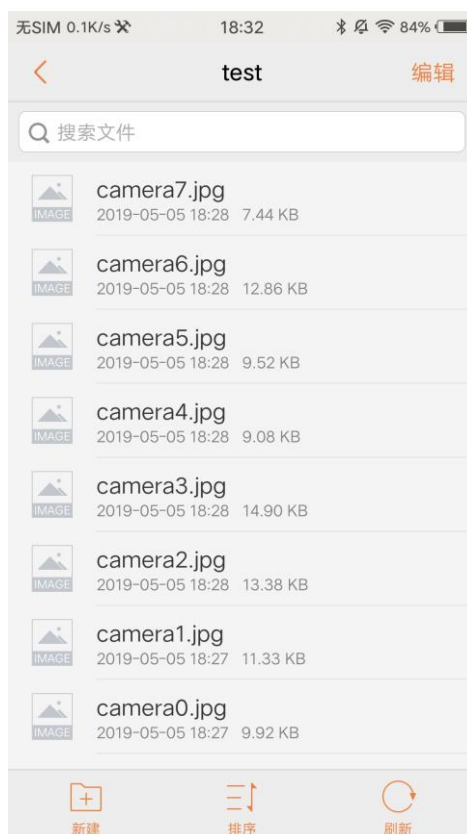


图 3.16

### 3.3 预览画面卡顿问题及解决过程

之前的多次测试中，我发现该程序的一个 bug：在每次点击“拍照”按钮之后，预览画面就会卡住(静止且没有响应)，而且如果此时再次点击”拍照“或”退出“按钮，程序就会意外退出。

经过查阅资料，我了解到，该问题是由以下原因造成的：

在执行拍照命令时，会调用 `camera.takePicture()` 方法，该方法在执行过程中会调用 `camera.stopPreview` 来获取拍摄帧数据，从而进行数据回调。而调用 `camera.stopPreview` 方法，就会暂停相机的预览，并表现为预览画面卡住。如果此时用户点击了按钮的话，也就会再次调用 `camera.takepicture` 方法，由于相机还没有开始预览，没有进行相关进程的初始化，就会出现之前遇到的意外退出问题。

解决的方法也很简单，既然 `camera.takePicture ()` 方法调用了 `camera.stopPreview` 来停止预览，那么只要在 `camera.takePicture()` 方法结束之后，手动调用一次 `camera.startPreview` 方法，来开启相机预览就可以了。

于是进行添加图 3.17 红框中的部分：

```
public void onPictureTaken (byte[] data, Camera camera){
    Bitmap bitmap = BitmapFactory.decodeByteArray(data, offset: 0, data.length);
    ArrayList<String> list = getFile( path: "/sdcard/test");
    path = "/sdcard/test/" + checkFileName(list, filename, index: 0);
    textView.setText("文件保存路径: " + path);
    try{
        BufferedOutputStream outputStream = new BufferedOutputStream(new FileOutputStream(path));
        bitmap.compress(Bitmap.CompressFormat.JPEG, quality: 80, outputStream);
        outputStream.flush();
        outputStream.close();
        mImageView.setImageBitmap(bitmap); //在ImageView显示拍照的图片
        mCamera.startPreview();
    } catch (Exception e){
        Log.e( tag: "error", e.getMessage());
    }
}
```

图 3.17

再次生成 APK 文件进行测试，图 3.18 为测试结果。

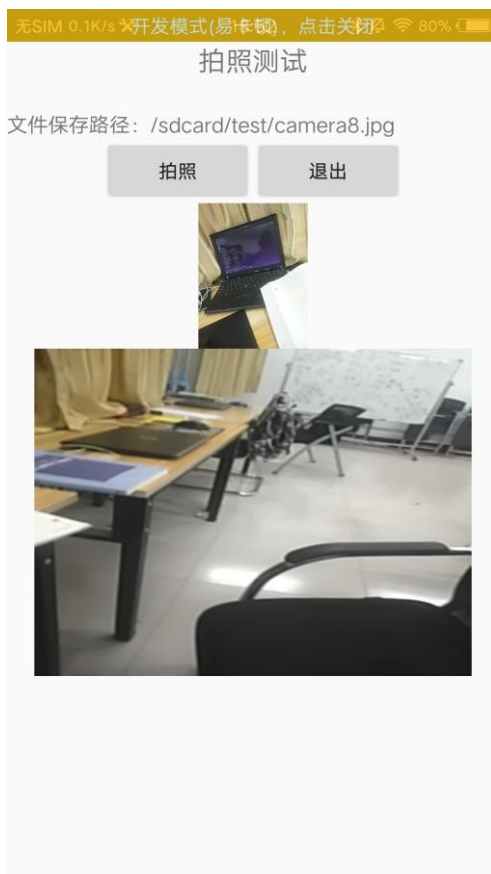


图 3.18

这是我拍摄第 9 张图片之后的界面。上方的 imageView 组件显示的是第 9 张的图片，而下方的 surfaceView 显示的是此时的预览画面。此时如果我连续拍摄，继续点击“拍照”按钮，程序也不会因为 bug 而意外退出了。实现了连续拍照的功能。

## 四、实验结果

现在，程序已经实现了实验所要求的功能，我将目前程序所能实现的效果录制成 GIF 图片并进行上传，程序的效果如下所示：<http://47.95.13.239/Study/Android/test.gif>

## 五、源代码

我已经将整理过的该项目的源代码上传到了 GitHub:

<https://github.com/ZHJ0125/AndroidSimpleCameraApp>

### 5.1 布局文件

```
<?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:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_gravity="center_horizontal"
        android:gravity="center_horizontal"
        android:text="拍照测试"
        android:textSize="20dp"
        tools:context=".MainActivity"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/test"
        android:textSize="15dp"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:gravity="center_horizontal">
        <Button
            android:id="@+id/btn1"
            android:layout_width="110dp"
            android:layout_height="wrap_content"
            android:text="拍照"/>
        <Button
            android:id="@+id/btn2"
            android:layout_width="110dp"
```

```

        android:layout_height="wrap_content"
        android:text="退出" />
    </LinearLayout>
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView1"
        android:layout_gravity="center" />
    <SurfaceView
        android:id="@+id/surfaceView1"
        android:layout_width="320dp"
        android:layout_height="240dp"
        android:layout_gravity="center_horizontal"/>
</LinearLayout>

```

## 5.2 程序控制文件

```

package zhj.com.simplecamera;

import android.os.Bundle;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.PixelFormat;
import android.hardware.Camera;
import android.hardware.Camera.PictureCallback;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.widget.Button;
import android.widget.ImageView;
import android.app.Activity;
import android.widget.TextView;
import java.io.BufferedOutputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.File;
import java.util.ArrayList;

public class MainActivity extends Activity implements SurfaceHolder.Callback{
    Camera mCamera = null;
    SurfaceView surfaceView;
    SurfaceHolder holder;
    ImageView mImageView;

```

```

Button cameraBtn, exitBtn;
TextView textView;
int i = 0;
String filename = "camera.jpg";    //图片文件名
String path = "";    //图片保存路径

/**
 * Override the onCreate
 * function: 重载构造函数, 关联布局文件和控制文件, 注册回调监听器
 */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //关联 ID
    mImageView = (ImageView)findViewById(R.id.imageView1);
    cameraBtn = (Button)findViewById(R.id.btn1);
    exitBtn = (Button)findViewById(R.id.btn2);
    cameraBtn.setOnClickListener(new mClick()); //设置监听事件
    exitBtn.setOnClickListener(new mClick());
    surfaceView = (SurfaceView)findViewById(R.id.surfaceView1);
    textView = (TextView)findViewById(R.id.test);
    //System.out.println("begin to holder...");
    holder = surfaceView.getHolder();
    holder.addCallback(this);
    holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
}

/**
 * class: mClick
 * function: 设置按键监听事件, 对应拍照和退出按钮
 */
class mClick implements OnClickListener{
    @Override
    public void onClick(View v) {
        if (v == cameraBtn){
            mCamera.takePicture(null, null, new jpegCallback());    //拍照
            //surfaceCreated(holder);    //调用构造函数
        }
        else if (v == exitBtn){
            exit(); //退出程序
        }
    }
}

```

```

void exit(){
    mCamera.release();
    mCamera = null;
}

/**
 * class: jpegCallback
 * function: 实现拍照和保存图片功能
 */
public class jpegCallback implements PictureCallback{
    @Override
    public void onPictureTaken (byte[] data, Camera camera){
        Bitmap bitmap = BitmapFactory.decodeByteArray(data, 0, data.length);
        ArrayList<String> list = getFile("/sdcard/test");
        path = "/sdcard/test/" + checkFileName(list, filename, 0);
        textView.setText("文件保存路径: " + path);
        try{
            BufferedOutputStream outputStream = new BufferedOutputStream(new
FileOutputStream(path));
            bitmap.compress(Bitmap.CompressFormat.JPEG, 80, outputStream);
            outputStream.flush();
            outputStream.close();
            mImageView.setImageBitmap(bitmap); //在 ImageView 显示拍照的图片
            mCamera.startPreview();
        }catch (Exception e){
            Log.e("error", e.getMessage());
        }
    }
}

private static ArrayList<String> getFile(String path){ //检测路径下所有的文件,并返回文件名
列表
    File file = new File(path); // 获得指定文件对象
    File[] array = file.listFiles(); // 获得该文件夹内的所有文件
    ArrayList<String> list = new ArrayList<String>();
    for(int i=0;i<array.length;i++){
        if(array[i].isFile()){ //如果是文件,将文件名保存在列表中
            list.add(array[i].getName());
        }
    }
    return list; //返回储存文件名的列表
}

private static String checkFileName(ArrayList<String> names,String name,int index) { //
检测文件是否存在,并返回合理的文件名

```

```

if(names.contains(name.substring(0,name.indexOf("."))+index+name.substring(name.indexOf("."),name.length())) {

//System.out.println(name.substring(0,name.indexOf("."))+index+name.substring(name.indexOf("."),name.length()+ "Hello");
    //name.substring(0,name.indexOf(".")) --> 返回"."之前的字符串
    //文件存在,再次调用 checkFileName 方法
    name = checkFileName(names,name,index+1);
} else {
    //文件不存在,返回合理的文件名
    return
name.substring(0,name.indexOf("."))+index+name.substring(name.indexOf("."),name.length());
}
return name;
}

/**
 * Override the surfaceCreated
 * function: 创建相机时触发, 开启相机预览功能
 * */
@Override
public void surfaceCreated(SurfaceHolder holder){
    if (mCamera != null){
        ReleaseCamera();    //首先释放相机资源
    }
    mCamera = Camera.open();    //开启摄像头
    //System.out.println("\n\n\nCamera.open() is OK !!!\n\n\n");

    //mCamera = android.hardware.Camera.open();
    //mCamera = Camera.open(Camera.CameraInfo.CAMERA_FACING_BACK);
    //System.out.println("Camera is OK!");
    try{
        mCamera.setPreviewDisplay(holder);    //设置相机预览
    }catch (IOException e){
        System.out.println("预览错误");
    }
}

private void ReleaseCamera()    //重置相机
{
    if(mCamera != null)
    {
        mCamera.release();
        mCamera = null;
    }
}

```



```

    }
}

/**
 * Override the surfaceChanged
 * function: 当画面发生改变时触发, 重置相机参数
 * */
@Override
public void surfaceChanged (SurfaceHolder holder, int format, int width, int height){
    //System.out.println("Camera is going to ready...");
    initCamera();    //重置相机参数
}

//设置相机参数
public void initCamera(){
    //System.out.println("here1...");
    Camera.Parameters parameters = mCamera.getParameters();
    parameters.setPictureFormat(PixelFormat.JPEG);    //照片格式
    parameters.setPreviewSize(320, 240);    //预览规格大小
    parameters.setPictureSize(320, 240);    //图片大小
    parameters.setRotation(90);    //设置照片数据旋转 90°
    mCamera.setParameters(parameters);
    mCamera.setDisplayOrientation(90);    //设置浏览画面水平转 90°

    mCamera.startPreview(); //开始预览
}

/**
 * Override the surfaceDestroyed
 * function: 关闭相机时触发, 空
 * */
@Override
public void surfaceDestroyed(SurfaceHolder holder){    }    //消灭相机时触发
}

```

## 六、待解决问题和设想

### 6.1 待解决问题

#### ①功能单一

该 APP 还有很多不完善的地方, 功能非常单一, 只能实现简单的拍照功能。

#### ②版本兼容问题

本次实验中所有的 APP 测试均在安卓 7.1.2 版本的手机上进行，在新建工程时，我将最底支持版本选择为 API 14 和 Android 4.0，但是在 Android 版本为 4.4.2 的手机上测试时，依然会出现很多问题，导致程序异常结束。程序还未在更高 Android 版本的手机上测试过，尚不清楚将会有什么 bug。版本的兼容问题是亟待解决的关键问题之一。

### ③关于连拍功能

报告中已经说明了连续拍照的实现过程，到目前为止，程序已经实现该过程：点击“拍照”按钮 → 拍摄照片 → imageView 显示照片 → surfaceView 启动预览 → 循环拍摄。但在调试过程中我发现，如果点击”拍摄“按钮过快，可能导致相机预览 camera.startPreview 还未开始，就点击了”拍摄“按钮，于是就会导致跟之前相同的问题，让程序意外结束。这也是待解决的问题之一。

## 6.2 设想

### ①版本问题

现在版本问题是最严重的一个问题，因为大家使用的 Android 手机，其 Android 版本必然不同，如果不能解决该问题，可能导致很多手机无法正常运行该程序，或者会出现很多 bug。现在需要查阅一下 Android 版本的资料，并进行兼容性的改进。

### ②加功能

现在功能很单一，以后希望能加上摄像等功能，做到比安卓自带的相机功能更加完善(笑)。

## 七、实验总结

整个实验项目进行的还算顺利，一开始还有很多问题，最后基本上都通过查阅资料以及和同学的交流，解决了问题。

问题还算挺多的，主要是一些细节问题导致的，需要对整个工程的结构，以及各个函数的功能都理清楚，才能更好地解决问题。

调试也有很多技巧，最简便的是使用输出语句判断程序的执行，还有就是通过注释部分功能，判断出问题的位置等等。

## 八、部分参考资料

1. 关于 androidcamera 相机的方向的理解 [http://blog.sina.com.cn/s/blog\\_68f23d9f0102y2cc.html](http://blog.sina.com.cn/s/blog_68f23d9f0102y2cc.html)
2. Java 检测文件名是否重复 <https://blog.csdn.net/u014804332/article/details/80385217>
3. JAVA 中方法的调用 <https://www.imooc.com/article/13423>
4. 预览卡住解决办法思路引导 [https://www.jianshu.com/p/586af3a2dc8d?utm\\_campaign=maleskine&utm\\_content=note&utm\\_medium=seo](https://www.jianshu.com/p/586af3a2dc8d?utm_campaign=maleskine&utm_content=note&utm_medium=seo)

[um=seo\\_notes&utm\\_source=recommendation](#)

5. Android 相机开发和遇到的坑 <https://blog.csdn.net/xx326664162/article/details/53350551>