Android移动应用开发基础教程

讲授：葛新

第6章 多媒体

本章主要内容：

- 播放多媒体文件
- 记录声音
- 使用摄像头和相册

6.1 播放多媒体文件

手机已经成为日常人们听音乐和看电影的工具。Android提供了完整的API用于播放多媒体文件，使开发人员可以轻松实现简易的音频和视频播放APP。

本节主要内容：

1. 使用SoundPool播放音效
2. 使用MediaPlay播放音频
3. 使用MediaPlayer播放视频

6.1.1 使用SoundPool播放音效

- SoundPool类可用于管理和播放应用中的音频资源。这些音频资源即可实现包含在应用程序中，也可存放于存储器文件中。通常，SoundPool类只用于播放较短的音频，比如游戏中的各种音效。
- 要使用SoundPool播放音频，首先需创建SoundPool对象。例如：（实例项目：源代码\06\UseSoundPool）

```
if (Build.VERSION.SDK_INT >= 21) {
    SoundPool.Builder builder = new SoundPool.Builder();
    builder.setMaxStreams(2);//设置可加载的音频数量
    //AudioAttributes是一个封装音频各种属性的方法
    AudioAttributes.Builder attrBuilder = new AudioAttributes.Builder();
    attrBuilder.setLegacyStreamType(AudioManager.STREAM_MUSIC);//预设音频类型
    builder.setAudioAttributes(attrBuilder.build());//设置音频类型
    sp = builder.build();//创建SoundPool对象
}
else {//当系统的SDK版本小于21时
    sp = new SoundPool(2, AudioManager.STREAM_SYSTEM, 0);
```

- 在API 21（Android 5.0））之后的版本中，SoundPool()构造方法已经过时了，需用SoundPool.Builder来创建SoundPool对象。SoundPool.Builder对象可执行setMaxStreams()方法设置SoundPool对象中可加载的最大音频数量。setAudioAttributes()方法则用于设置音频的类型。
- SoundPool()构造方法的第一个参数为可加载音频最大数量，第二个参数为音频类型，第三个参数为声音品质（目前无效，用0表示默认值）。

- 获得SoundPool对象后，首先应调用load()方法加载音频资源。load()方法基本格式如下：

  int load(Context context, int resId, int priority)

  int load(String path, int priority)

  int load(AssetFileDescriptor afd, int priority)

  int load(FileDescriptor fd, long offset, long length, int priority)

- 其中，context为当前应用上下文。resId事先存放到应用的res/raw文件中的音频文件的资源ID。priority为优先级，目前无效，1用于与未来版本兼容。path为存储器中音频文件的路径。AssetFileDescriptor为音频asset文件的描述符。在将多个音频存放在一个二进制文件中时，FileDescriptor为该音频文件的描述符，offset指定加载的因为在文件中的开始位置，length指定音频长度。load()方法返回值为加载的音频的ID，在调用其他方法播放、暂停或其他操作处理音频时，用音频ID作为参数。

- 调用load()方法准备好音频资源后，可调用play()方法来播放音频。play()方法基本格式如下：

  play(int soundID, float leftVolume, float rightVolume, int priority, int loop, float rate)

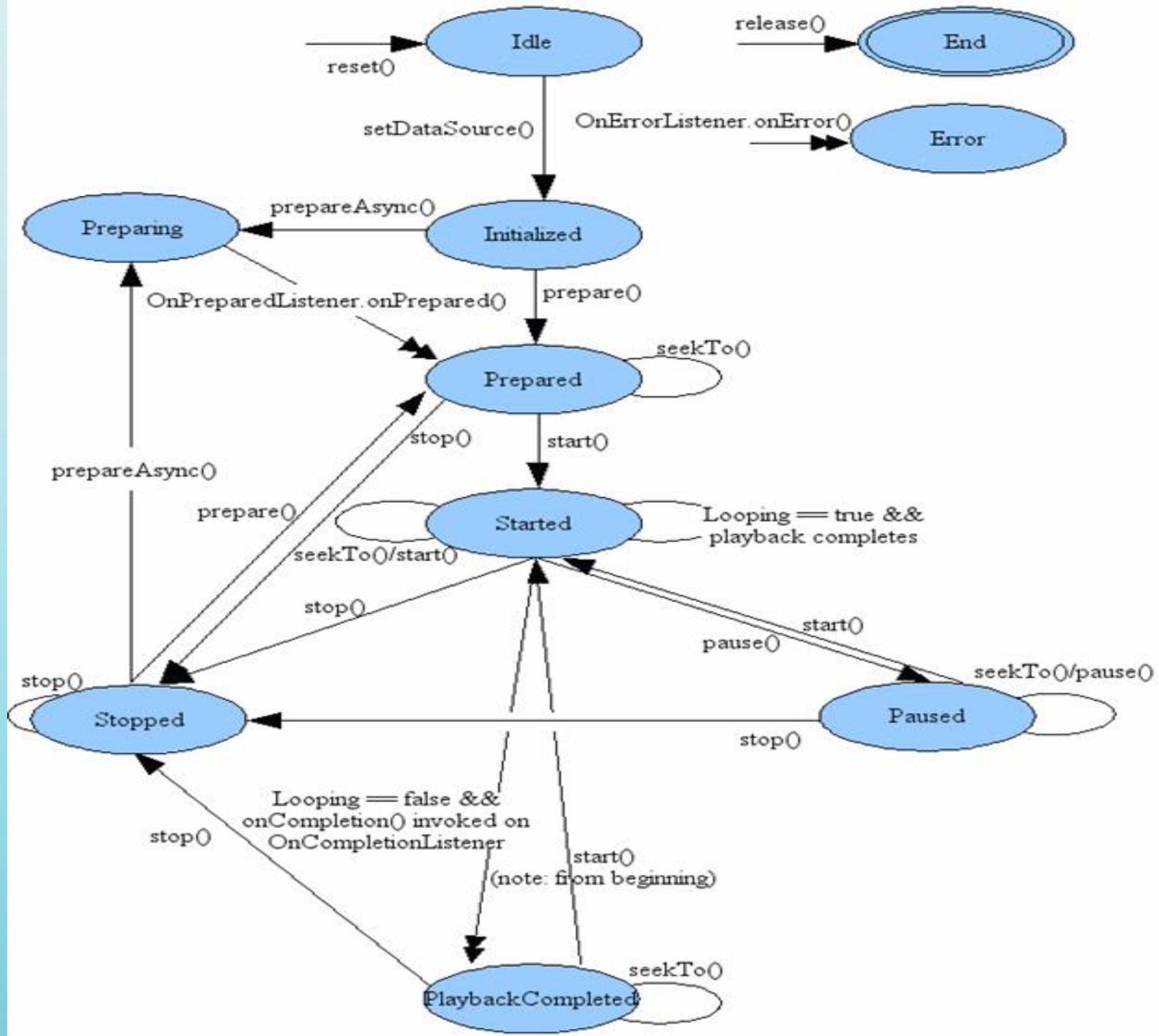
- 其中soundID为load()方法加载音频资源时返回的ID。
- leftVolume和rightVolume分别为左声道音量和右声道音量，取值范围0.0~1.0。priority为优先级，0为最低级。
- loop为重复次数，0表示不重复。
- Rate为播放速率，取值范围0.5~2.0，1.0为正常播放速度。
- 例如，下面的语句播放soundId1对应的音频。

  sp.play(soundId1,1,1,1,0,1);

- 其他SoundPool常用方法如下。
  - pause(int streamID)：暂停播放。
  - release()：释放SoundPool中加载的音频资源。
  - resume(int streamID)：继续播放暂停的音频。
  - setLoop(int streamID, int loop)：设置重复播放次数。
  - setVolume(int streamID, float leftVolume, float rightVolume)：设置音量。
  - stop(int streamID)：停止播放。
  - unload(int soundID)：卸载SoundPool中的音频资源。

6.1.2 使用MediaPlay播放音频

- MediaPlay类提供了音频和视频播放功能，本节先学习用其播放音频。MediaPlay类比SoundPool类提供了更多的音频控制功能，支持更多的音频格式。
- 在使用MediaPlay对象处理音频时，音频可处于多种状态，如图6-1所示（该图引用自：http://developer.android.com/reference/android/media/MediaPlayer.html）
- 在调用MediaPlay对象方法时，音频可在相应的不同状态之间进行转换。


Idle
reset()
release()
End
setDataSource()
OnErrorListener.onError()
Error
Preparing
prepareAsync()
Initialized
OnPreparedListener.onPrepared()
prepare()
seekTo()
Prepared
stop()
start()
prepareAsync()
prepare()
Started
Looping == true &&
playback completes
seekTo()/start()
stop()
start()
pause()
seekTo()/pause()
stop()
Stopped
Paused
stop()
Looping == false &&
onCompletion() invoked on
OnCompletionListener
stop()
start()
(note: from beginning)
seekTo()
PlaybackCompleted

MediaPlay音频控制常用方法

- getCurrentPosition：获得当前播放位置。
- getDuration：获得音频时长。
- isPlaying：判断是否处于播放状态。
- pause：暂停播放。
- prepare：准备音频。本地音频通常不需要准备，远程音频通过准备完成下载和本地缓冲。
- release：释放MediaPlayer对象资源。

MediaPlay音频控制常用方法

- reset：恢复MediaPlayer对象到刚创建状态。
- seekTo：设置播放位置。
- setDataSource：设置音频文件位置。
- setVolume：设置音量。
- start：开始播放。
- stop：停止播放。

使用MediaPlay播放音频的基本步骤

1. 创建MediaPlay对象。
2. 调用setDataSource方法设置音频文件路径。
3. 调用prepare方法加载音频。
4. 调用start方法播放音频。
5. 调用pause方法暂停正在播放的音频。
6. 调用stop方法停止播放，
7. 调用reset方法重置MediaPlay到刚创建时的状态。

6.1.3 使用MediaPlayer播放视频

- MediaPlayer即可用于播放音频，也可用于播放视频，在用法上没有多大区别。只是在播放视频时，应使用SurfaceView控件作为视频的显示容器。
- 下面通过具体的例子说明如何使用MediaPlayer播放视频（上一小节中的UseMediaPlay实例略加修改即可用于播放视频）。（实例项目：源代码\06\UseMediaPlay2）

6.2 记录声音

- MediaRecorder类提供了音频采集功能，使开发者可使用设备的麦克风记录声音。要在应用中实现音频采集功能，首先需要在应用程序的清单文件AndroidManifest.xml添加RECORD_AUDIO权限申请使用麦克风。例如：

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

- Android系统认为使用RECORD_AUDIO权限记录用户声音隐私，是一种“危险”行为。在从Android 6.0（API 23）开始，需要在应用程序运行时动态向用户申请RECORD_AUDIO权限。用户授权后，应用可记录授权，不再重复询问。
- 通常，调用ActivityCompat.requestPermissions()方法来动态申请权限。

修改activity_main.xml主活动布局

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"……>
   <Button  android:text="开始录音"  android:layout_width="match_parent"
      android:layout_height="wrap_content"  android:id="@+id/btStartRecord" />
<Button
android:text="停止录音"  android:layout_width="match_parent"
      android:layout_height="wrap_content"  android:id="@+id/btStopRecord" />
   <Button
      android:text="播放录音"  android:layout_width="match_parent"
      android:layout_height="wrap_content"  android:id="@+id/btPlay" />
   <Button
      android:text="停止播放"  android:layout_width="match_parent"
      android:layout_height="wrap_content"   android:onClick="stopPlay"
      android:id="@+id/btStop" />
</LinearLayout>
```

MainActivity.java，为各个按钮添加单击事件监听器，实现音频的采集和播放控制

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mFileName = getExternalCacheDir().getAbsolutePath();
        mFileName += "/audiorecord.3gp";
        //检查应用是否已经获得授权
        if(ContextCompat.checkSelfPermission(this,
                Manifest.permission.RECORD_AUDIO)
                != PackageManager.PERMISSION_GRANTED){
            //如果没有权限，动态申请授权
            ActivityCompat.requestPermissions(this,
                    new String[]{Manifest.permission.RECORD_AUDIO},1);
        }else {            initMediaRecorder();        }
```

```
Button btStartRecord= (Button) findViewById(R.id.btStartRecord);
        btStartRecord.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {//开始录音
                try {
                    mediaRecorder.prepare();//准备MediaRecorder
                } catch (IOException e) {
                    Log.e(LOG_TAG, "准备MediaRecorder出错啦！");
                }
                mediaRecorder.start();//开始采集音频
            }
        });
```

```
Button btStopRecord= (Button) findViewById(R.id.btStopRecord);
        btStopRecord.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {//停止录音
                mediaRecorder.stop();//停止MediaRecorder
                mediaRecorder.release();//释放MediaRecorder所占资源
                mediaRecorder = null;
            }
        });
```

```
Button btStartPlay= (Button) findViewById(R.id.btPlay);
        btStartPlay.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {//开始播放
                mediaPlayer = new MediaPlayer();
                try {
                    mediaPlayer.setDataSource(mFileName);//设置要播放的音频文件
                    mediaPlayer.prepare();
                    mediaPlayer.start();
                } catch (IOException e) {
                    Log.e(LOG_TAG, "MediaPlayer方法prepare()执行失败！");
                }
            }
        });
```

```
Button btStop= (Button) findViewById(R.id.btStop);
        btStop.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {//停止播放
                mediaPlayer.release();
                mediaPlayer = null;
            }
        });
```

```
public void onRequestPermissionsResult(int requestCode,
                                       @NonNull String[] permissions, @NonNull int[] grantResults) {
    if(requestCode==1){
        if(grantResults.length>0 && grantResults[0]==
                PackageManager.PERMISSION_GRANTED){
            initMediaRecorder();//初始化
        }else{
            Toast.makeText(this,"未获得麦克风访问权限",
                                    Toast.LENGTH_LONG).show();
            finish();
        }
    }
}
```

```
private void initMediaRecorder() {//初始化MediaRecorder
        mediaRecorder = new MediaRecorder();
        mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);//设置音频来源，使用麦克风
        mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);//设置输出格式
        mediaRecorder.setOutputFile(mFileName);//设置音频输出文件
        mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);//设置音频编码方式
    }
```

```
protected void onDestroy() {//应用停止时，释放资源
        if (mediaRecorder != null) {
            mediaRecorder.release();
            mediaRecorder = null;
        }
        if (mediaPlayer != null) {
            mediaPlayer.release();
            mediaPlayer = null;
        }
        super.onDestroy();
    }
```

6.3 使用摄像头和相册

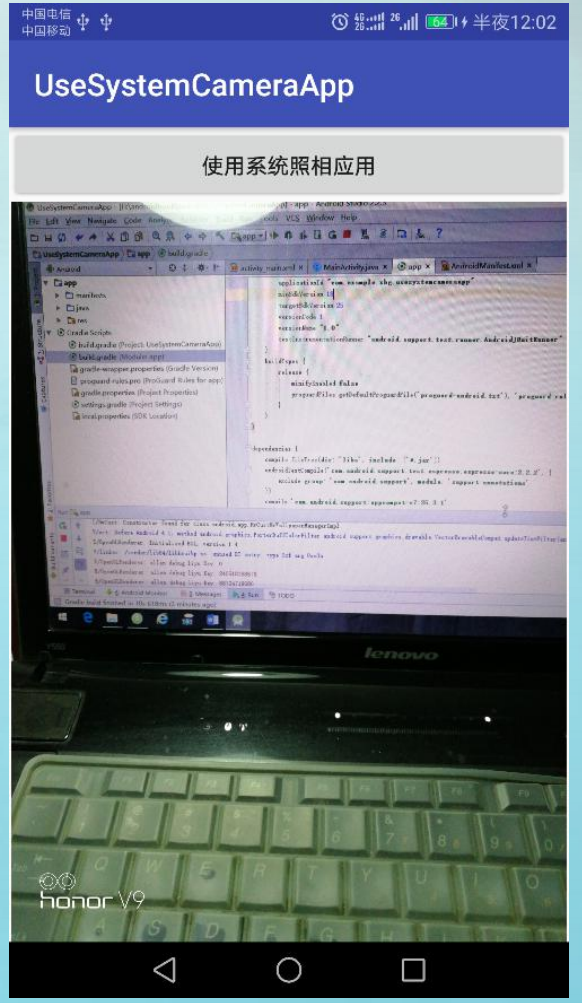
在常用通讯软件（如QQ、微信等）中，经常需要分享图片，下、这些图片可来自于相册或者摄像头拍摄。

本节主要内容：

1. 使用摄像头拍摄相片
2. 选取相册图片

6.3.1 使用摄像头拍摄相片

- 要使用摄像头拍摄下相片，最简单的方式就是调用系统拍照程序。创建一个动作为MediaStore.ACTION_IMAGE_CAPTURE的Intent对象，执行startActivityForResult()启动系统拍照程序，即可使用摄像头进行拍照。
- 下面通过一个例子说明如何调用系统拍照程序完成拍照，并显示拍摄的相片。
- 具体操作步骤如下。（实例项目：源代码\06\UseSystemCameraApp）



在AndroidManifest.xml文件中申请权限

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.xbg.usesystemcameraapp">
<uses-permission android:name=
        "android.permission.WRITE_EXTERNAL_STORAGE"/>
    ……
</manifest>
```

修改MainActivity.java，实现拍照和相片显示

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //使用StrictMode.VmPolicy.Builder检测应用中的FileUriExposure事件
    StrictMode.VmPolicy.Builder builder = new StrictMode.VmPolicy.Builder();
    StrictMode.setVmPolicy(builder.build());
    builder.detectFileUriExposure();
```

```
Button btTakePhoto= (Button) findViewById(R.id.btTakePhoto);
        btTakePhoto.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                try {
                    //创建用于保存所拍照片的文件
                    File sdcard = Environment.getExternalStoragePublicDirectory(
                                                            Environment.DIRECTORY_PICTURES);
                    picFile = new File(sdcard, System.currentTimeMillis() + ".jpg");
                    picFile.createNewFile();
                    Log.e("UseSystemCameraApp",picFile.getName()+"创建成功！");
                }catch(IOException e){
                    e.printStackTrace();
                }
                //使用Intent调用系统摄像头拍照程序
                Intent intent=new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                intent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(picFile));
                startActivityForResult(intent,1);
```

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if(requestCode==1){
            //处理ActivityResult调用返回，将所拍照片显示在ImageView中
            ImageView iv=(ImageView)findViewById(R.id.ivShow);
            iv.setImageURI(Uri.fromFile(picFile));
        }
        super.onActivityResult(requestCode, resultCode, data);
    }
```

6.3.2 选取相册图片

- 选取相册图片与使用系统拍照程序拍照类似，使用Android内置的Activity即可完成。
- 将上一节中的实例项目UseSystemCameraApp略加修改实现相册图片选择。（实例项目：源代码\06\SelectPhoto）
- 首先，创建Intent对象，指定Intent.ACTION_PICK操作用于启动相册。

```
public void onClick(View v) {
        //使用Intent对象来打开相册
        Intent intent = new Intent(Intent.ACTION_PICK,
       android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
        startActivityForResult(intent, 1);
```

- 然后在onActivityResult()方法中处理返回的图片Uri，将图片显示到ImageView中

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if(requestCode==1){//处理从相册选取相片返回结果
            if(resultCode==RESULT_OK){//若用户正确完成相片选择操作返回，进一步处理选择的相片
                Uri uri = data.getData();
                Log.e("图片URI：", uri.toString());
                ContentResolver cr = this.getContentResolver();
                try {
                    Bitmap bitmap = BitmapFactory.decodeStream(cr.openInputStream(uri));
                    ImageView iv=(ImageView)findViewById(R.id.ivShow);
                    /* 将Bitmap设定到ImageView */
                    iv.setImageBitmap(bitmap);
                } catch (FileNotFoundException e) {
                    Log.e("出错了：", e.getMessage(),e);
```