# 多媒体编程（底层都是用到了c编写的）

* 文本
* 图片
* 音频
* 视频

# 图片

## 图片在计算机中的大小，这里我们使用bmp格式计算，因为它是无损压缩，而jpg是有损压缩，所以算出来的可能会不一样

### 图片总大小 = 图片总像素（尺寸800\*400=320000像素） \* 每个像素占用的大小

### 图片分为：单色位图，16色位图， 256色位图， 24位位图

### 单色位图：只能表示2种颜色

* 使用两个数字：0和1
* 使用一个长度为1的二进制数字就可以表示了
* 每个像素占用1/8个字节（一个字节 = 1个8位 = 长度为8的二进制数字）
* 计算：320000 \* （1/8） = 320000 / 8 = 40000 个字节
* 然而实际上时40000个字节还多一点，因为图片除了保存像素信息之外，还有其他信息：拍摄时间，相机型号，PS版本号…

### 16色位图：能表示16种颜色

* 需要16个数字：0-15，二进制：0000 – 1111
* 使用一个长度为4的二进制数字就可以表示了
* 每个像素占用1/2个字节
* 计算：320000 \* （1/2） = 320000 / 2 = 160000 个字节

### 24位位图：能表示16777215种颜色

* 每个像素占用24位，也就是3个字节，一个字节8位，所在叫24位位图
* R：0-255,需要一个长度为8的二进制数字，占用1个字节
* G：0-255,需要一个长度为8的二进制数字，占用1个字节
* B：0-255,需要一个长度为8的二进制数字，占用1个字节
* 计算：320000 \* 3 = 960000个字节

# 加载大图片

* 计算机把图片所有像素信息全部解析出来（与jpg等格式无关，用上面方法计算出来的总大小是多少就是多少），保存至内存
* Android保存图片像素信息，是用ARGB保存。
* 也就是每个像素占用4个字节，也就是32位。
* 如果尺寸是800\*400，那么图片解析出来的大小就是：320000 \* 4 = 1280000个字节
* 所以，加载一个图片格式为jpg，大小为1,815,556字节的图片。
  + 当然解析的时候，和这两个数据是没关系的。而是根据图片尺寸，和每个像素多少位决定的。
  + 如果这个图片的尺寸为：2400 \* 3200，且是24位位图（Android用的是ARGB, 所以每个像素占32位）。
  + 那么解析出来的大小就是：7680000 \* 4 = 30720000个字节， 30720000/1204/1204 = 29.296M。
  + 所以加载图片过程中需要用到的内存大小是29M。而我们手机如果只有16M的内存，会造成内存溢出。
* 解决方案：把图片缩放到适配屏幕的大小
  + 如果屏幕大小是320\*480，总像素153600
  + 图片宽高2400\*3200，总像素7680000
  + 如果不缩放，直接加载，那么屏幕肯定显示不下，肯定会内存溢出
  + 计算：图片宽/手机宽，图片高/手机高。得到缩放比例。也就是宽高分别缩放多少倍。计算结果取整数，如果计算结果其中一个是小于1的，那么证明图片比屏幕还小，所以就不用缩放了
    - 缩就是除多少倍，放就是乘多少倍
    - 宽：2400 / 320 = 7
    - 高：3200 / 480 = 6
    - 宽高比例不一致，图片会变形。到底用哪一个呢。选大的。
    - 原因是：如果用6的，那么高3200 / 6=533，宽：2400/6=400。明显，图片宽高是比手机宽高还要大一点的。虽然大一点不会内存溢出，但是会浪费像素。所以不合适
      * 而使用7缩放，那么高3200 / 7 = 457， 宽:2400/7=342。那么只有图片宽比屏幕宽大一点。浪费的像素更小。所以使用大的

# 创建图片副本：在内存中创建一个图片的复制文件

## 作用：

* Bitmap bm = BitmapFactory.decodeFile(Environment.getExternalStorageDirectory() + "/dog.jpg", opts);，得到sd卡上的dog.jpg图片
* 这样得到的图片是只读的，不能修改
* 如果要往这张图片上加特效（水印…），就需要创建一个副本（可读可写）来进行操作

## 选一个图片比较小的来做试验。这样就不会有内存溢出，就不用去缩放图片了

* 原图跟复制图同时在屏幕上显示，出现一张图片变小了。是因为被另一张图片挤压导致第二张的空间不够大导致的

# 图片加简单特效

# 触摸事件

## 画画板

### 原理

* 在一个图片上作画。把手指经过的每两个坐标点之间画一条直线

## 改变画笔颜色和宽度

## 保存图片

* 会发现保存后图片在图库没有显示（实际上已经保存在手机了）。
* 因为：系统每次收到SD卡就绪广播时，都会去遍历sd卡的所有文件和文件夹，把遍历到的所有多媒体文件都在MediaStore数据库保存一个索引，这个索引包含多媒体文件的文件名、路径、大小、长度和艺术家
* 图库、音乐、视频程序每次打开时，其实不会去遍历sd卡寻找多媒体文件，而是直接从MediaStore数据库中读取多媒体文件，通过库中的索引找到对应的多媒体文件后，把文件显示在界面
* 系统开机或者点击加载sd卡按钮时，系统会发送sd卡就绪广播，我们也可以手动发送就绪广播

## 撕衣服

* 原理：两个图片叠加，把上一层图片划过的坐标置位透明色即可

# 音乐播放器

* + 用的是android.media.MediaPlayer;这个类来播放音乐
  + android.media.MediaPlayer状态图



## 第一版：播放本地音乐，没有进度条

## 第二版: 播放网络音乐，没有进度条

## 第三版：增加进度条

* 进度条控件使用：SeekBar，因为可拖动

|  |
| --- |
| <SeekBar  android:id=*"@+id/sb"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"wrap\_content"*  android:max=*"1000"*  android:progress=*"0"*/> |

* 在服务中，使用计时器，不断的获取当前播放进度，传给消息队列。然后在MAinActivity中，刷新进度条
* 为SeekBar设置进度改变侦听，在拖动后获取到拖动后的进度。使用中间人间接调用服务中的方法，改变播放进度

# 视频播放

* 视频显示在SurfaceView控件里面
* 双缓冲技术
* 重量级组件
* 只要不可见，就不会创建，可见时，才会创建
* 只要不可见，就会销毁

|  |
| --- |
| <SurfaceView  android:id=*"@+id/sv"*  android:layout\_width=*"wrap\_content"*  android:layout\_height=*"wrap\_content"*/> |

## 第一版：还是使用android.media.MediaPlayer操作视频

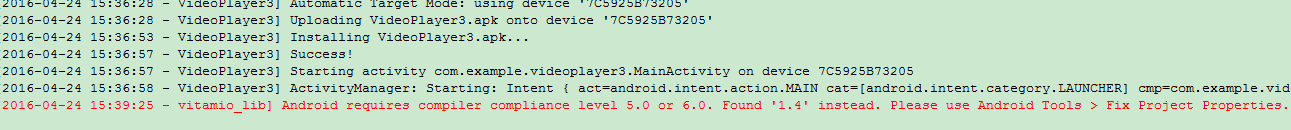
* 支持的格式很少很少（3gp..）

## 第二版：使用Android原生的VideoView

* 高度封装了第一版的代码

## 第三版：使用Vitamio框架：

* FFMPEG
  + 开源免费的音视频编解码器
* Vitamio
  + 封装了FFMPEG的视频播放框架
  + 对外提供的api全部都是javaApi
* 导入一个java项目vitamio\_lib。他所在的目录要与开发项目同目录
* 如果有这个错误的话，按照提示，选择修复即可



* 播放控件：io.vov.vitamio.widget.VideoView
* 检测是否支持Vitamtio
  + if (!LibsChecker.checkVitamioLibs(this)) {return;}
* 增加一个默认的播放控件（有进度条，暂停，播放等...）
  + voidView.setMediaController(new MediaController(this));
* 还要配置一个activity
  + <activity android:name="io.vov.vitamio.activity.InitActivity"></activity>

# 拍照摄像

* 原本拍照摄像是需要权限的。但是我们这里是启动拍照摄像程序，而不是我们来进行拍照摄像。所以不需要权
* 拍照的action：MediaStore.ACTION\_IMAGE\_CAPTURE
* 摄像的action：MediaStore.ACTION\_VIDEO\_CAPTURE