# 手机应用平台软件开发

## 14、多媒体应用

## Color类

Android颜色使用4个数字表示，分别代表透明度、红色、绿色和蓝色（Alpha、Red、Green、Blue），每个数字8位，因此常用32位整数表示。

- Int color=Color.BLUE;//Color类的静态常量；
- Color=Color.argb(127,255,0,255);

## Paint类

Paint类是Android本机图形库中最重要的类之一。它包含样式、颜色以及绘制任何图形（包括位图、文本和几何图形）所需提供的其他信息。

```
cPaint.setColor(Color.LTGRAY);
//代码使用了浅灰色的预定义颜色值;
```

## Canvas类

Canvas类是代表可在其上绘制的画布。最初，其上没有任何内容。Android中的现实屏是由Activity类的对象支配的，Activity类的对象引用View类的对象，而View类的对象又引用Canvas类的对象，通过覆写View.onDraw()方法，可以在指定的画布上绘图。

# Path类

包含一组用于绘制2D图形的矢量绘图命令，例如画线条、画矩形和画曲线等。Path类简单来说就是封装了一系列由线段（方形、多边形等）、2次曲线（圆、椭圆等）和3次曲线复合而成的几何轨迹集合的类。

借助于Canvas的drawPath(path, paint)方法可以将这些轨迹集合绘制出来，另外一个paint参数则是用于设置画笔风格，例如当使用此方法绘制一个圆时，若paint的风格为filled，则会绘制一个实心圆，若风格为stroked，则会绘制一个空心圆，还可以使用paint在path中加入文字。

# Path类基本方法

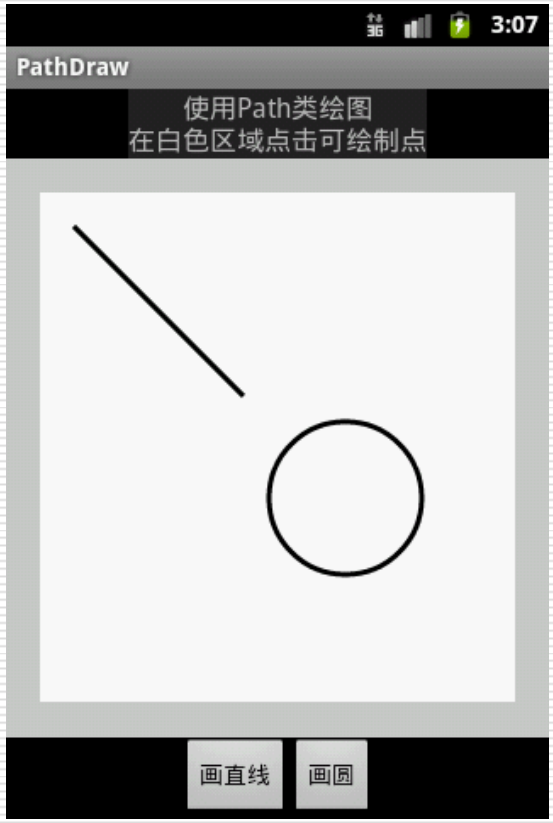
- addArc()：用于绘制一段圆弧；
- addCircle()：用于绘制圆形；
- addOval()：用于绘制椭圆；
- addPath()：用于绘制一组轨迹；
- addRect()：用于绘制矩形；
- addRoundRect()：用于绘制圆角矩形；

可以使用这些提供好的方法为基础通过组合绘制出各种图形。另外Path还提供了一些直接操作元线段的方法，例如moveTo()、lineTo()等方法。

# Path绘制2D图形示例




初始界面




画出直线和圆


通过点击画曲线

# Path绘制2D图形示例



为了实现能够在一块特定的区域内进行绘图，项目首先实现了一个用于绘制图像的视图类PathView，该类实现了用于向Path中添加一条线段的方法drawLine()；用于添加一个圆形的方法drawCircle()；用于绘制Path的方法showPath()，该方法在重写的父类方法onDraw()中被调用；用于向Path中添加点的方法drawPoint()；还通过重写onTouchEvent()方法实现了对触摸（点击）事件的响应，从而可以通过触摸（点击）来绘制一系列的点。下面是PathView类的一些变量和构造函数。

# Path绘制2D图形示例



```
public class PathView extends View {
    private Paint mPaint = new Paint(Paint.ANTI_ALIAS_FLAG); //抗锯齿
    private Path mPath;//声明Path类对象
    private boolean mCurDown;//指示屏幕是否被按下（点击）
    private int mCurX;//被点击点的坐标
    private int mCurY;
    private final Rect mRect = new Rect();//该矩形用于控制局部更新区域
    public PathView(Context context, AttributeSet attrs) {
        super(context, attrs);
        setFocusable(true);  //设置视图可获得焦点
        setFocusableInTouchMode(true);//设置视图可获取触摸事件
        mPaint.setStyle(Paint.Style.STROKE);//绘制成线
        mPaint.setStrokeWidth(3);//设置绘制线宽
        mPath = new Path();//创建Path实例
    }
```

# Path绘制2D图形示例

SUN YAT-SEN UNIVERSITY

drawLine()，用于绘制直线的方法。

```
public void drawLine(){//向Path中添加一条线段供绘制
    mPath.moveTo(20, 20);
    mPath.lineTo(120, 1200);
    invalidate();//立即更新视图
}
```

drawCircle()，用于绘制圆圈的方法。

```
public void drawCircle(){//向Path中添加一个圆形供绘制
    mPath.addCircle(180, 180, 45, Path.Direction.CCW);
    invalidate();
}
```

# Path绘制2D图形示例



showPath()，显示绘制的图形。

```
private void showPath(Canvas canvas, int x, int y, Path.FillType ft,Paint paint)
{
    canvas.translate(x, y);
    canvas.clipRect(0, 0, 280, 300);//可以绘制图形的区域
    canvas.drawColor(Color.WHITE);//为该区域着色
    mPath.setFillType(ft);//设置fill方式
    canvas.drawPath(mPath, paint);//绘制Path
}
```

onDraw()，用于更新显示。

```
protected void onDraw(Canvas canvas) {
    Paint paint = mPaint;
    canvas.drawColor(0xFFCCCCCC);//为Canvas着色
    canvas.translate(20, 20);//Canvas向左下方向偏移（20，20）
    paint.setAntiAlias(true);//抗锯齿
    paint.setColor(Color.BLACK);//设置画笔颜色
    paint.setPathEffect(null);//不采用特效
    showPath(canvas, 0, 0, Path.FillType.WINDING, paint
}
```

# Path绘制2D图形示例



onTouchEvent()，处理触摸事件；drawPoint()，画点。

```
public boolean onTouchEvent(MotionEvent event) {
    int action = event.getAction();
    mCurDown = action == MotionEvent.ACTION_DOWN
            || action == MotionEvent.ACTION_MOVE;
    int N = event.getHistorySize();
    for (int i=0; i<N; i++) {
      drawPoint(event.getHistoricalX(i), event.getHistoricalY(i));
    }
    drawPoint(event.getX(), event.getY());//添加当前被触摸的点
    return true;
}
private void drawPoint(float x, float y) {
    mCurX = (int)x - 20;
    mCurY = (int)y - 20;
    if (mCurDown) {//添加当前被触摸点并刷新视图显示
        mPath.addCircle(mCurX, mCurY, 1, Path.Direction.CCW);
        mRect.set((int)x-3, (int)y-3, (int)x+3, (int)y+3);
        invalidate(mRect);  //只更新矩形区域
    }
}
```

# 触摸事件

- 在手机上运行的应用程序，效率是非常重要的。若Android界面框架不能产生足够多的触摸事件，则应用程序就不能够很精确的描绘触摸屏上的触摸轨迹
- 若Android界面框架产生了过多的触摸事件，虽然能够满足精度的要求，但却降低了应用程序效率
- Android界面框架使用了“打包”的解决方法。在触点移动速度较快时会产生大量的数据，每经过一定的时间间隔便会产生一个ACTION_MOVE事件，在这个事件中，除了有当前触点的相关信息外，还包含这段时间间隔内触点轨迹的历史数据信息，这样既能够保持精度，又不至于产生过多的触摸事件。

# 触摸事件

- 通常情况下，在ACTION_MOVE的事件处理函数中，都先处理历史数据，然后再处理当前数据

```
1.  private int ProcessHistory(MotionEvent event)
2.      {
3.          int historySize = event.getHistorySize();
4.          for (int i = 0; i < historySize; i++) {
5.                  long time = event.getHistoricalEventTime(i);
6.                  float pressure = event.getHistoricalPressure(i);
7.                  float x = event.getHistoricalX(i);
8.                  float y = event.getHistoricalY(i);
9.                  float size = event.getHistoricalSize(i);
10.
11.                 // 处理过程.....
12.         }
13.         return historySize;
14.     }
```

## 触摸事件

- 第3行代码获取了历史数据的数量
- 然后在第4行至12行中循环处理这些历史数据
- 第5行代码获取了历史事件的发生时间
- 第6行代码获取历史事件的触点压力
- 第7行和第8行代码获取历史事件的相对坐标
- 第9行获取历史事件的触点尺寸
- 在第14行返回历史数据的数量，主要是用于界面显示

- Android模拟器并不支持触点压力和触点尺寸的模拟，所有触点压力恒为1.0，触点尺寸恒为0.0
- Android模拟器上也无法产生历史数据，历史数据量一直显示为0

# Path绘制2D图形示例



由于该示例中还需要在Activity中添加一些其他的UI控件，所以需要在布局文件中将PathView作为一个控件元素使用，使之成为界面组成的一部分。

```
<FrameLayout android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:layout_weight="1">
   <com.android.example.pathdraw.PathView
      android:id="@+id/pathdrawview"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent" />
</FrameLayout>
```

# Android的动画

SUN YAT-SEN UNIVERSITY

## Android 平台提供了两类动画

- 补间动画（Tween Animation），即通过对场景里的对象不断做图像变换(平移、缩放、旋转)产生动画效果
- 帧动画（Frame Animation），即顺序播放事先做好的图像，跟电影类似。类似GIF

# 补间动画简介



- Tween 动画通过对 View 的内容完成一系列的图形变换 (包括平移、缩放、旋转、改变透明度)来实现动画效果。
- 预先定义一组指令，这些指令指定了图形变换的类型、触发时间、持续时间。这些指令可以是以 XML 文件方式定义，也可以是以源代码方式定义。程序沿着时间线执行这些指令就可以实现动画效果。

# 补间动画简介

SUN YAT-SEN UNIVERSITY

- 动画的进度使用 Interpolator 控制
  - LinearInterpolator 实现了匀速效果
  - Accelerateinterpolator 实现了加速效果
  - DecelerateInterpolator 实现了减速效果
- 还可以定义自己的 Interpolator 子类，实现抛物线、自由落体等物理效果。

# 补间动画简介

SUN YAT-SEN UNIVERSITY

- 动画的运行模式有两种
  - 独占模式：

    即程序主线程进入一个循环，根据动画指令不断刷新屏幕，直到动画结束；
  - 中断模式：

    即有单独一个线程对时间计数，每隔一定的时间向主线程发通知，主线程接到通知后更新屏幕；

# 补间动画简介



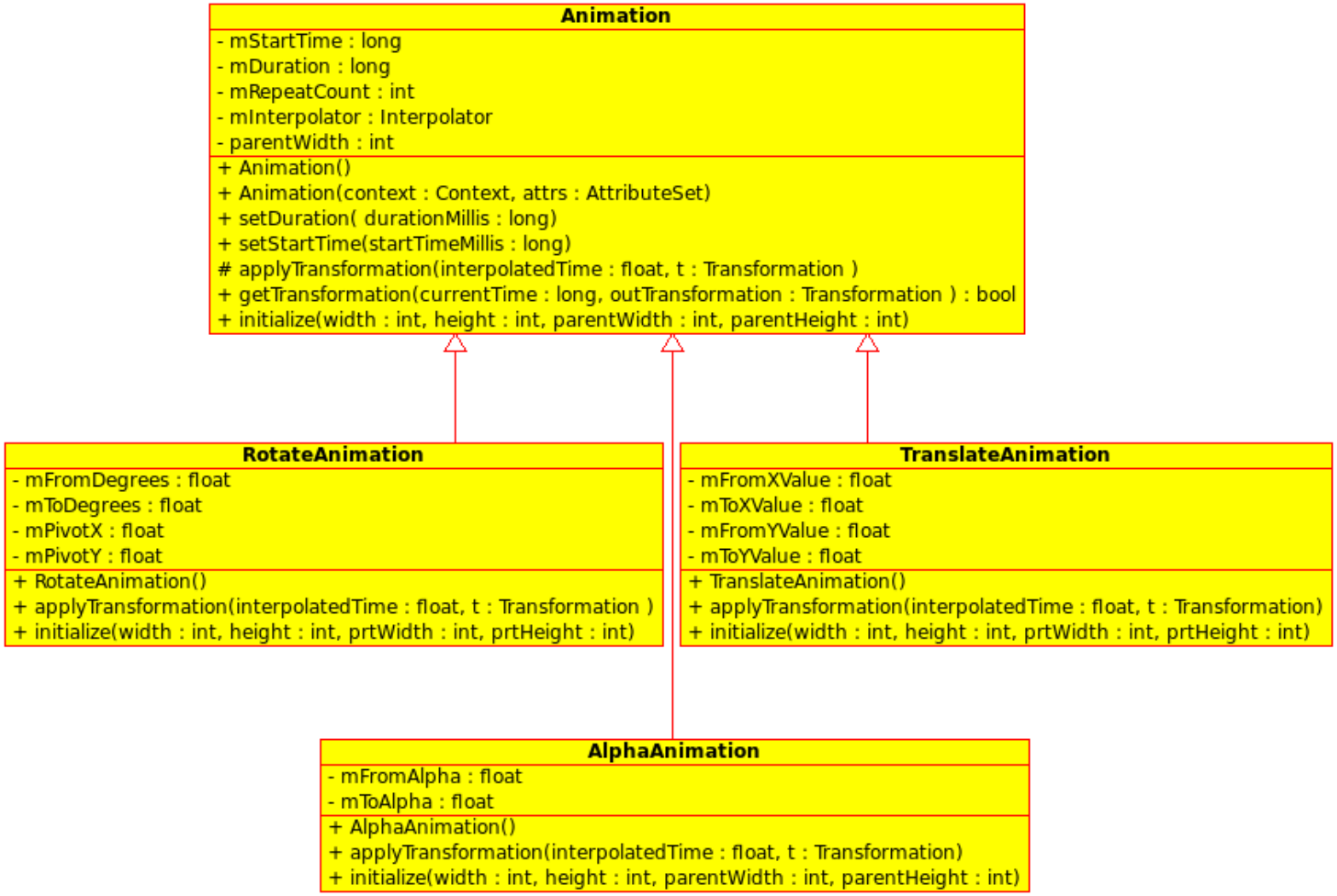
- 图形变换通过仿射矩阵实现。图形变换是线性代数基本知识。
- 每种变换都是一次矩阵运算。在 Android 中，Canvas 类中包含当前矩阵，当调用 Canvas.drawBitmap(bmp, x, y, Paint)绘制时，android会先把bmp做一次矩阵运算，然后将运算的结果显示在 Canvas 上。这样，编程人员只需不断修改 Canvas 的矩阵并刷新屏幕，View 里的对象就会不停的做图形变换，动画就形成了。

# Animation类及其子类

- Animation 类及其子类是动画的核心模块，它实现了各种动画效果，如平移、缩放、旋转、改变透明度等。

# Animation类及其子类

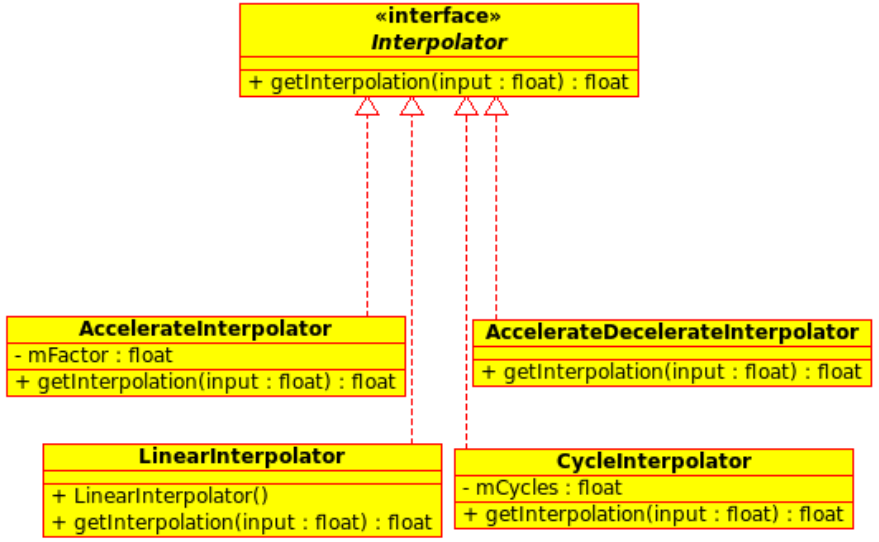




# Interpolator 类及其子类



- SDK的描述：An interpolator defines the rate of change of an animation. This allows the basic animation effects (alpha, scale, translate, rotate) to be accelerated, decelerated, repeated, etc。简而言之Interpolator就是一个“变化率”，一个基本动画的“变化率”。
- Interpolator 定义了动画的变化速度，可以实现匀速、正加速、负加速



# Interpolator 类及其子类



- Interpolator接口有一个抽象方法getInterpolation(float input)，由此SDK中扩展了另外几个常用Interpolator类，分别是：
  - AccelerateInterpolator：动画从开始到结束，变化率是一个加速的过程。
  - DecelerateInterpolator：动画从开始到结束，变化率是一个减速的过程。
  - CycleInterpolator：动画从开始到结束，变化率是循环给定次数的正弦曲线。
  - AccelerateDecelerateInterpolator：动画从开始到结束，变化率是先加速后减速的过程。
  - LinearInterpolator：动画从开始到结束，变化率是线性变化。

# Interpolator 类及其子类



- 对于 LinearInterpolator，变化率是个常数，即 f (x) = x.

```
public class LinearInterpolator implements Interpolator {

    public LinearInterpolator() {
    }

    public LinearInterpolator(Context context, AttributeSet attrs) {
    }

    public float getInterpolation(float input) {
        return input;
    }
}
```

# Interpolator 类及其子类



- 对于 AccelerateInterpolator，开始变化很慢，然后逐渐变快，即 f(x) = x*x 或者 f(x) = pow(x, 2*mFactor)

```
public float getInterpolation(float input) {
    if (mFactor == 1.0f) {
        return input * input;
    } else {
        return (float)Math.pow(input, mDoubleFactor);
    }
}
```

# Interpolator 类及其子类



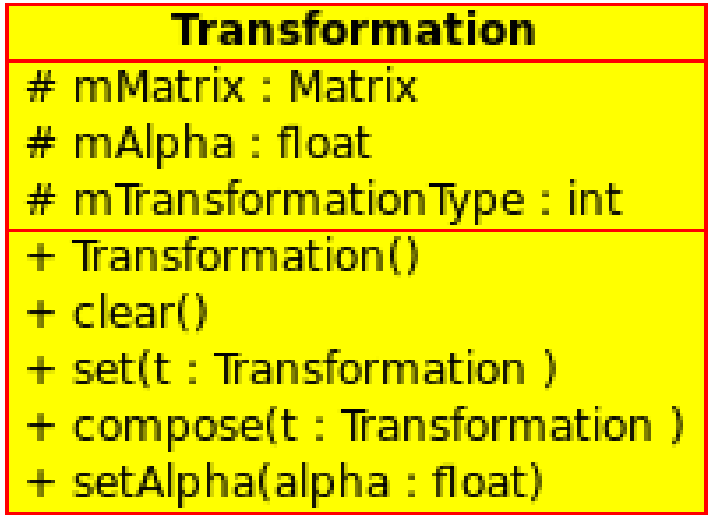
- AccelerateDecelerateInterpolator，变化率开始和结束都很慢，但中间很快，即 f(x) = (cos ((x+1)*PI) / 2.0f) + 0.5f.

```
public float getInterpolation(float input) {
    return (float)(Math.cos((input + 1) * Math.PI) / 2.0f) + 0.5f;
}
```

# Transformation 类

SUN YAT-SEN UNIVERSITY

- Transformation 记录了仿射矩阵 Matrix，动画每触发一次，会对原来的矩阵做一次运算， View 的 Bitmap 与这个矩阵相乘就可实现相应的操作(旋转、平移、缩放等)。
- Transformation 类封装了矩阵和 alpha 值，它有两个重要的成员，一是 mMatrix，二是 mAlpha。



# View中对Animation的实现



- view 创建动画对象，设置动画属性，调用invalidate 刷新屏幕，启动动画；
- invalidate 方法触发了 onDraw 函数；
- 在 onDraw 函数中：
  - 调用动画的 getTransformation 方法，得到当前时间点的矩阵
  - 将该矩阵设置成 Canvas 的当前矩阵
  - 调用 canvas 的 drawBitmap 方法，绘制屏幕。
  - 判断 getTransformation 的返回值，若为真，调用 invalidate 方法，刷新屏幕进入下一桢；若为假，说明动画完成。

# 如何使用Animation？



## 1、在xml文件中定义Animation

- 在资源文件夹中创建xml文件/res/anim/anim.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
        android:interpolator="@android:anim/accelerate_interpolator">
    <translate android:fromXDelta="100%p" android:toXDelta="0" android:duration="1000"></translate>
</set>
```

- 在代码中通过AnimationUtil.load加载这个xml文件创建Animation对象
- 再通过View.startAnimation开始动画

# 如何使用Animation？



## 2、通过代码动态创建Animation对象

- 位移，形变等Animation和Interpolator的使用，参考相应SDK
- 当有多种Animation同时进行时，会用到AnimationSet

```
myTextView = (TextView)findViewById(R.id.myTextView);
Animation animation = new AlphaAnimation(0.0f, 1.0f);
animation.setDuration(3000);

myTextView.setAnimation(animation);
```

# 多媒体开发—播放音频

SUN YAT-SEN UNIVERSITY

Android支持的音频格有:AAC, WAV, MP3, WMA, OGG, MIDI。在模拟器上只支持OGG,WAV和MP3格式。借助于Android提供的MediaPlayer类可以快速的完成播放一段音频的代码实现，创建方式有两种：

1. 使用静态方法MediaPlayer.create创建，通过参数使播放器与资源相关联起来，再使用start()方法开始播放指定的音频文件，代码如下：

```
MediaPlayer mediaPlayer = MediaPlayer.create(this, R.raw.tmp);
 mediaPlayer.start();
```

2. 使用构造方法MediaPlayer()创建一个播放器对象，然后使用播放器的setDataSource()方法将音频资源相关联，与静态方法创建播放器不同的是，使用构造方法创建的播放器还需要首先使用prepare()方法，然后再使用start()方法开始播放，否则会抛出一个播放器状态不正常的异常。代码如下：

```
MediaPlayer mp = new MediaPlayer();
mp.setDataSource(PATH_TO_FILE);
mp.prepare();
mp.start();
```

# 多媒体开发—录制音频

SUN YAT-SEN UNIVERSITY

录制音频资源最方便的方式是使用MediaRecorder类，通过设置录制音频来源（通常是设备默认的麦克风）就能方便的录制语音，具体步骤如下：

1. 实例化android.media.MediaRecorder对象；
2. 使用MediaRecorder.setAudioSource()方法来设置音频资源；这将会很可能使用到MediaRecorder.AudioSource.MIC；
3. 使用MediaRecorder.setOutputFormat()方法设置输出文件格式；
4. 用MediaRecorder.setAudioEncoder()方法来设置音频编码；
5. 使用setOutputFile()方法设置输出的音频文件；
6. 使用prepare()和start()方法开始录制音频，通过stop()和release()方法完成一段音频的录制。

# 多媒体开发—录制音频

SUN YAT-SEN UNIVERSITY

```
/* 实例化MediaRecorder对象 */
recorder = new MediaRecorder();
/* 设置麦克风 */
recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
/* 设置输出文件的格式 */
recorder.setOutputFormat(MediaRecorder.OutputFormat.DEFAULT);
/* 设置音频文件的编码 */
recorder.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);
/* 设置输出文件的路径 */
recorder.setOutputFile(mRecAudioFile.getAbsolutePath());
/* 准备 */
recorder.prepare();
/* 开始 */
recorder.start();
/* 停止录音 */
recorder.stop();
/* 释放MediaRecorder */
 recorder.release();
```

# 多媒体开发—录制音频

由于录制音频需要使用麦克风，因此需要在AndroidManifest.xml文件中声明使用麦克风的权限：

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

# 多媒体开发—视频

Android原生系统所支持解码的视频编码格式有：H.263（后缀.3gp和.mp4）、H.264 AVC（3.0+版本，后缀.3gp和.mp4等等）、MPEG-4 SP（后缀.3gp）和VP8（2.3.3+版本，后缀.webm），其中H.263和H.264是Android支持的编码格式。

与音频的使用方式非常相似，不同点是音频本身并不会表现为用户界面，而视频则需要成为用户界面的一部分，视频播放只要使用VideoView类就可以实现，而视频录制也是借助于MediaRecorder类，不同之处是此处的数据来源由麦克风变为摄像头，另外再将输出格式及编码方式做相应修改，为了实现有实用性的视频录制功能，还需要增加一个用于显示实时录制图像的视图。

# 多媒体开发—播放视频

SUN YAT-SEN UNIVERSITY

使用VideoView播放视屏的代码如下，还包括了使用系统提供的播放控制器MediaController与VideoView进行绑定，从而控制视频的播放/暂停、快进/快退的简单操作。

```
Context context = getApplicationContext();
VideoView mVideoView = new VideoView(context);
mVideoView = (VideoView) findViewById(R.id.surface_view);
mVideoView.setVideoURI(Uri.parse(path));
MediaController mc = new MediaController(this);
mc.setAnchorView(mVideoView);
mVideoView.setMediaController(mc);
mVideoView.requestFocus();
mVideoView.start();
```

# 多媒体开发—录制视频

SUN YAT-SEN UNIVERSITY

录制视频的基本实现与录制音频的方式相似，于可用性方面的考虑，需要提供实时观察录制区域的显示。

通常为了实现对摄像头捕获图像的预览，实现了一个Preview类，该类继承自SurfaceView并实现了SurfaceHolder.Callback接口，SurfaceView可以理解为可嵌入到界面布局中的一块用于图像绘制的区域，通常用于摄像头预览、游戏界面、3D绘图等等，VideoView就是SurfaceView的一个子类。每一个SurfaceView都有一个与之绑定的SurfaceHolder类对象用于控制SurfaceView的一些属性，可以通过SurfaceView的getHolder()方法获取到SurfaceHolder实例，SurfaceHolder.Callback接口则提供了用于在SurfaceView创建、更改和被销毁时所调用的回调方法，通过实现这个接口来进行SurfaceView的初始化，更新及销毁的工作。

# 多媒体开发—录制视频（Preview类）

SUN YAT-SEN UNIVERSITY

Preview类主要实现了SurfaceHolder.Callback接口的三个回调方法，并且加入了用于监听开始录制和停止录制的按键事件方法，以及用于控制视频开始录制和停止录制的方法。下面是Preview类的一些变量和构造函数。

```
public class Preview extends SurfaceView implements SurfaceHolder.Callback{
    private SurfaceHolder holder = null;
    private static boolean isRecording = false;
    /* 录制的视频文件名及存放路径 */
    private File mRecVideoFile;
    private File mRecVideoPath;
    /* MediaRecorder对象 */
    private MediaRecorder mMediaRecorder;
    /* 录制视频文件名的前缀 */
    private String strTempFile = "A_VideoRecordTest_";
    public Preview(Context context) {
      super(context);
      holder = this.getHolder();//获取SurfaceHolder对象
      holder.addCallback(this);//添加回调接口
      holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);//缓冲类型
      holder.setFixedSize(400, 300);//设置视图大小
    }
……
```

# 多媒体开发—录制视频（Preview类）

SUN YAT-SEN UNIVERSITY

surfaceCreated()方法，初始化视频预览界面。

```
public void surfaceCreated(SurfaceHolder holder) {
   if(mMediaRecorder==null){
      mRecVideoFile = new File("/mnt/sdcard/VideoRecordTempFile.3gp");
      mMediaRecorder = new MediaRecorder();
mMediaRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);
    mMediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
      mMediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
      mMediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
      mMediaRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.H263);
      mMediaRecorder.setOutputFile(mRecVideoFile.getAbsolutePath());
      mMediaRecorder.setPreviewDisplay(holder.getSurface());
      try {
          mMediaRecorder.prepare();
      } catch (IllegalStateException e) {
         e.printStackTrace();
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

# 多媒体开发—录制视频（Preview类）

onKeyDown()方法，处理键被按下的事件。

```
public boolean onKeyDown(int keyCode, KeyEvent event) {
   switch(keyCode){
   case KeyEvent.KEYCODE_DPAD_CENTER:{  //方向导航键中键
      if(mMediaRecorder !=null){
         if(isRecording){
            finishRecordVideo();
            isRecording = false;
         } else{
            startRecordVideo();
            isRecording = true;}
      }
      break;
   }
   case KeyEvent.KEYCODE_BACK:{
      System.exit(0);
     }}
   return super.onKeyDown(keyCode, event);
}
```

# 多媒体开发—录制视频（Preview类）

SUN YAT-SEN UNIVERSITY

startRecordVideo()方法，开始录制视频(保存视频文件)。

```
public void startRecordVideo(){
    try{
    /* 创建视频文件 */
    mRecVideoFile = File.createTempFile(strTempFile, ".3gp", mRecVideoPath);
    /* 设置输出文件的路径 */
    mMediaRecorder.setOutputFile(mRecVideoFile.getAbsolutePath());
    mMediaRecorder.setPreviewDisplay(holder.getSurface());
    /* 准备 */
    mMediaRecorder.prepare();
    /* 开始 */
    mMediaRecorder.start();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

实现了Preview后，在主Activity中使用setContentView()方法将视图设置为Preview即可。

# Questions?