LẬP TRÌNH DI ĐỘNG

Bài 10: dịch vụ đa phương tiện trên android

Nhắc lại nội dung bài trước

- Khái niệm, ưu điểm và nhược điểm của khả năng làm việc đa luồng (multithread)
- Tiếp cận của android trong lập trình đa luồng
  - Duy trì một luồng chính phụ trách giao diện (UI thread), kiểm soát nghiêm ngặt việc tương tác với UI thread
  - Các API hỗ trợ, cho phép thực hiện các tác vụ nền (background works) một cách đơn giản
- Sử dụng handler: cơ chế giao tiếp kiểu ủy thác
- Sử dụng AsyncTask: thực hiện việc theo mẫu
- Các kĩ thuật kinh điển của java: Timer và Thread

Nội dung

1. Giao diện lập trình đa phương tiện (media API)
2. Cơ sở dữ liệu đa phương tiện (MediaStore)
3. Làm việc với audio
    1. Record
    2. Playback
4. Tổng hợp tiếng nói (text-to-speech)
5. Làm việc với video
6. Làm việc với camera

Phần 1

Giao diện lập trình đa phương tiện (media API)

Media API

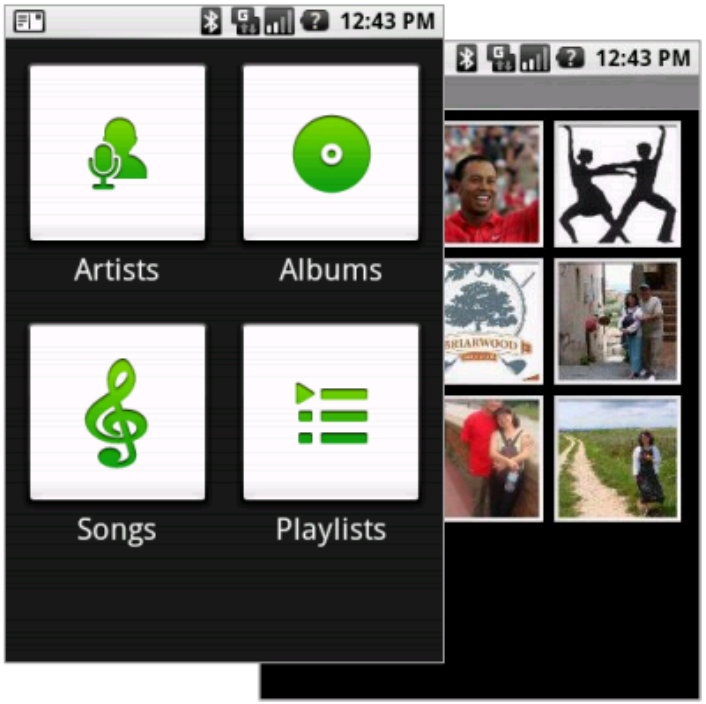
- Android cung cấp nhiều class hỗ trợ tập media phong phú, gồm cả âm thanh, hình ảnh và video
- Chia làm 2 nhóm recorder và playback
- Các lớp thư viện này đều dễ dàng sử dụng trong phát triển ứng dụng và hoàn toàn miễn phí (rất quan trọng)

Media API

- Tuy dễ sử dụng và miễn phí nhưng số lượng và các loại codec của android không quá nhiều, chưa có cơ chế mở rộng các công nghệ mới
- Android có rất nhiều loại thiết bị và nhiều nhà cung cấp khác nhau, vì vậy viết một ứng dụng chạy tốt với mọi loại camera, màn hình, micro là rất phức tạp

Media API

- Media API hỗ trợ đa dạng các loại tệp tin media:
  - Các tệp tin media được lưu trữ bên ngay bên trong ứng dụng (các file resources)
  - Các file media độc lập có trong bộ nhớ trong của máy
  - Các file media lưu trong thẻ nhớ SDCARD (cần quyền truy cấp thẻ sd-card)
  - Các file media trên mạng

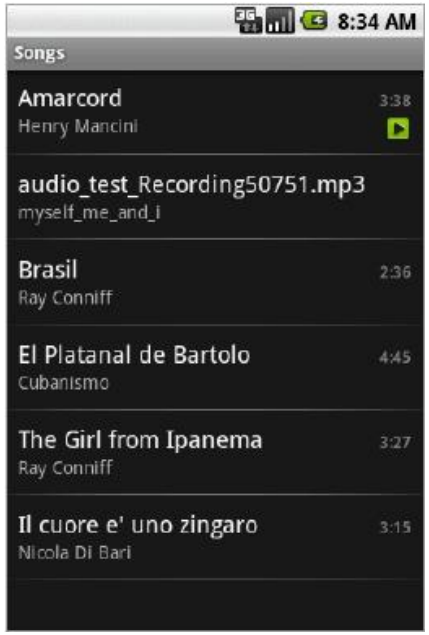


Phần 2

Cơ sở dữ liệu đa phương tiện (MediaStore)

MediaStore

- Android OS có provider chuẩn chứa thông tin về các file media trong thiết bị
- Các ứng dụng tạo media (chụp ảnh, quay video) chủ động thêm các file media vào provider này



MediaStore.Audio

MediaStore.Images

MediaStore.Video

MediaStore

Có thể xem cấu trúc của MediaStore bằng cách xem file CSDL sau /data/data/com.android.providers.media/databases/internal.db

```
// Muốn mở các video thì dùng Uri chuẩn EXTERNAL_CONTENT_URI
Uri p = android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
Intent myIntent = new Intent(Intent.ACTION_VIEW, p);
startActivity(myIntent);

// Trường hợp muốn mở để chọn media thì sử dụng ACTION_PICK
Uri x = android.provider.MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
Intent myIntent = new Intent(Intent.ACTION_PICK, x);
startActivityForResult(myIntent);
```

Phần 3

Làm việc với audio

MediaPlayer & MediaRecorder

- Lớp MediaPlayer hỗ trợ việc playback các file audio và video
- Lớp MediaRecorder hỗ trợ việc ghi âm (ghi hình) và chuyển thành các file audio (video)
  - Chú ý: việc record phụ thuộc vào việc phần cứng được hỗ trợ hay không
- Android OS chưa có cơ chế cài đặt thêm các codec mới cho audio và video, trường hợp ứng dụng muốn chạy format mới thì cần tự làm việc với i/o stream, surface view và audio stream

MediaPlayer

- MediaPlayer được sử dụng cho việc chơi lại (playback) các file audio, video và stream
- MediaPlayer hiểu cả các giao thức video internet
- Có thể xem các protocol được hỗ trợ bởi MediaPlayer trong link sau: http://developer.android.com/guide/appendix/media-formats.html

MediaPlayer – Useful Methods

public methods	
static MediaPlayer	create(Context context, Uri uri) convenience method to create a MediaPlayer for a given Uri
static MediaPlayer	create(Context context, int resid) convenience method to create a Media Player for a given resource id
boolean	isPlaying() checks whether the Media Player is playing
void	pause() pauses play back
void	prepare() prepares the player for play back, synchronously

MediaPlayer – Useful Methods

public methods	
void	setLooping(boolean looping) sets the player to be looping or non-looping
void	setVolume(float leftVolume, float rightVolume) sets the volume on this player
void	start() starts or resumes play back
void	stop() stops play back after playback has been stopped or paused

Phần 3.1

Làm việc với audio: record

Audio Recording

- Để ghi âm, sử dụng MediaRecorder
  1. Khởi tạo đối tượng recorder thông qua hàm khởi tạo
  2. Khởi tạo đối tượng android.content.ContentValues, truyền các giá trị TITLE, TIMESTAMP và MIME_TYPE để lưu trữ
  3. Tạo đường dẫn đến file lưu trữ
  4. Thiết lập audio source với MediaRecorder.setAudioSource()

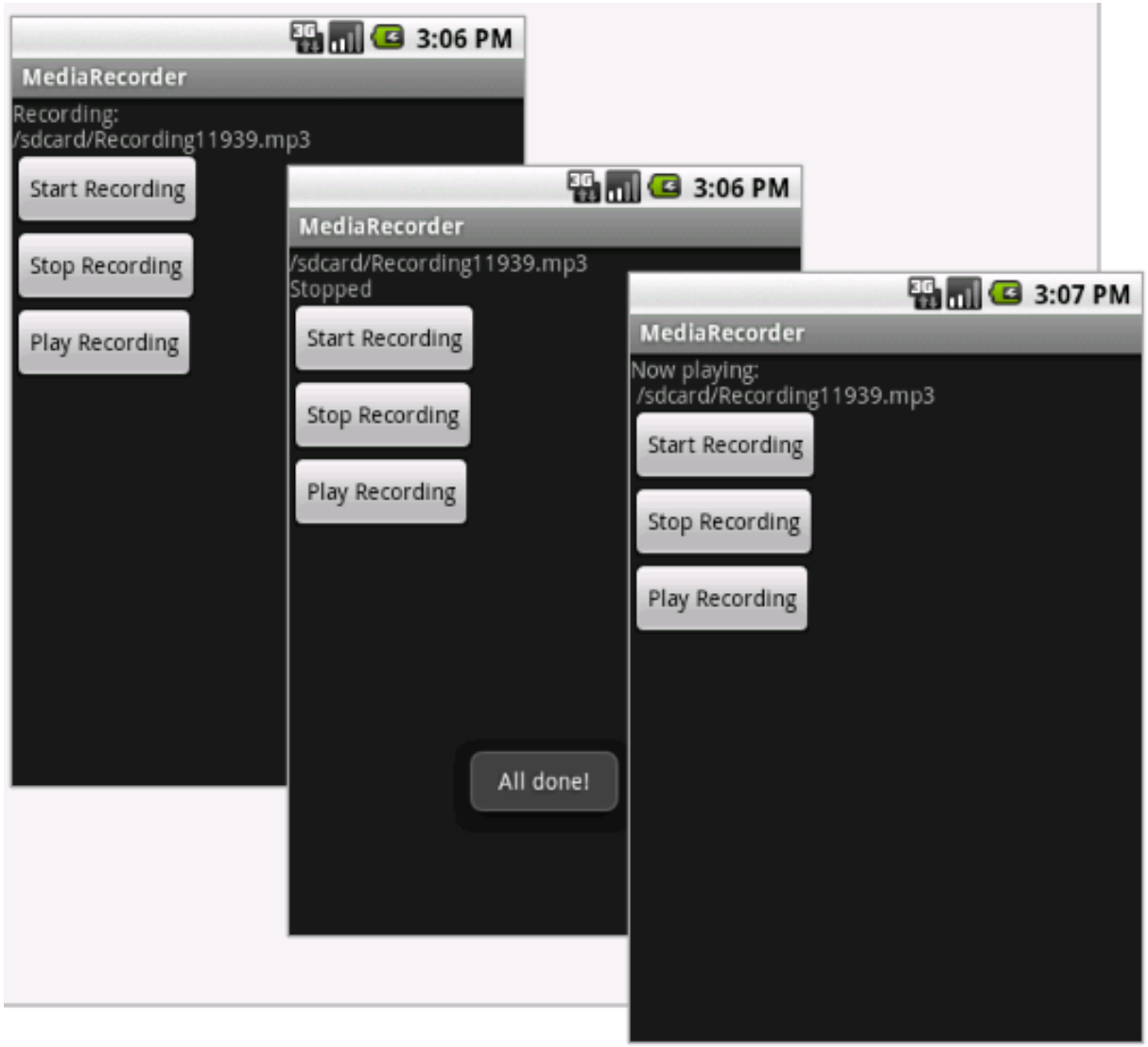
Audio Recording

5. Cấu hình kiểu format
   MediaRecorder.setOutputFormat()
6. Chọn kiểu mã hóa
   MediaRecorder.setAudioEncoder()
7. Chuẩn bị codec bằng gọi phương thức prepare()
8. Bắt đầu ghi âm với phương thức start() và dừng với stop()
9. Giải phóng bộ nhớ khi kết thúc, gọi release()

Audio Recording

```
// SET UP AND RECORD AN AUDIO FILE
recorder = new MediaRecorder();
// BASIC DATA NEEDED TO ADD RECORDING TO THE AUDIO MEDIASTORE PROVIDER
ContentValues values = new ContentValues(5);
     values.put(MediaStore.MediaColumns.TITLE, SOME_NAME_HERE);
     values.put(MediaStore.MediaColumns.TIMESTAMP, System.currentTimeMillis());
     values.put(MediaStore.MediaColumns.MIME_TYPE, recorder.getMimeContentType());
     values.put(AudioColumns.IS_MUSIC, true);
     values.put(AudioColumns.ARTIST, "myself");

ContentResolver contentResolver = new ContentResolver();
Uri base = MediaStore.Audio.INTERNAL_CONTENT_URI;
Uri newUri = contentResolver.insert(base, values);

// USE MICROPHONE, 3GP FORMAT, AMR CODEC (SPEECH RECORDING)
recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
recorder.setOutputFile(path);
// GET READY, GO ...
recorder.prepare();
recorder.start();
```

Audio Recording – example



Audio Recording – example

```
public class MyAudioRecorder extends Activity {
    MediaRecorder myRecorder;
    File mSampleFile = null;
    TextView txtMsg;
    static final String SAMPLE_PREFIX = "Recording";
    static final String SAMPLE_EXTENSION = ".mp3";
    private static final String TAG = "<<MyAudioRecorder>>";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        txtMsg = (TextView)findViewById(R.id.txtMsg);
        myRecorder = new MediaRecorder();
```

Audio Recording – example

```
Button start = (Button) findViewById(R.id.startRecording);
start.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        startRecording();
    }
});

Button stop = (Button) findViewById(R.id.stopRecording);
stop.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        stopRecording();
        addToMediaStoreDB();
    }
});
```

Audio Recording – example

```
    Button play = (Button) findViewById(R.id.playRecording);
    play.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            try {
                String name = mSampleFile.getAbsolutePath();
                txtMsg.setText("Now playing:\n " + name);
                MediaPlayer mp = new MediaPlayer();
                mp.setDataSource(name);
                mp.prepare();
                mp.start();
            } catch (Exception e) {}
        }
    }
} // onCreate
```

Audio Recording – example

```
protected void startRecording() {
    try {
        if (this.mSampleFile == null) {
            File dir = Environment.getExternalStorageDirectory();
            try {
                this.mSampleFile = File.createTempFile(
                            MyAudioRecorder.SAMPLE_PREFIX,
                            MyAudioRecorder.SAMPLE_EXTENSION, dir);
            } catch (IOException e) {
                return;
            }
        }
        txtMsg.setText("Recording: \n" +
                            mSampleFile.getCanonicalPath());
```

Audio Recording – example

```
        myRecorder = new MediaRecorder();
        myRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        myRecorder.setOutputFormat(MediaRecorder.OutputFormat.
                                                    THREE_GPP);
        myRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.
                                                      AMR_NB);
        myRecorder.setOutputFile(this.mSampleFile.
                                                 getAbsolutePath());
        myRecorder.prepare();
        myRecorder.start();
    } catch (Exception e) { }
} // startRecording
```

Audio Recording – example

```
protected void stopRecording() {
    try {
        myRecorder.stop();
        myRecorder.release();
    } catch (IllegalStateException e) {}
}
protected void addToMediaStoreDB() {
    try {
        int now = (int) (System.currentTimeMillis() / 1000);
        ContentValues newValues = new ContentValues(6);
        newValues.put(MediaColumns.TITLE, mSampleFile.getName());
        newValues.put(MediaColumns.DATE_ADDED, now);
```

Audio Recording – example

```
        newValues.put(MediaColumns.MIME_TYPE, "audio/mpeg");
        newValues.put(AudioColumns.IS_MUSIC, true);
        newValues.put(AudioColumns.ARTIST, "myself");
        newValues.put(MediaColumns.DATA,
                                    mSampleFile.getAbsolutePath());
        ContentResolver contentResolver = getContentResolver();
        Uri base = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
        Uri nUri = contentResolver.insert(base, newValues);
        sendBroadcast(new
            Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, nUri));
    } catch (Exception e) { }
} // addToMediaStoreDB
```

Phần 3.2

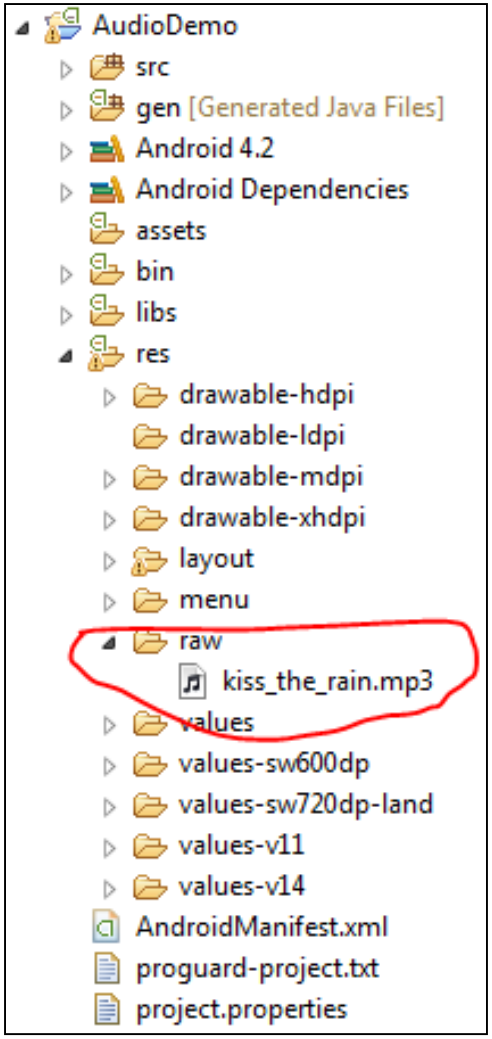
Làm việc với audio: playback

Chơi audio từ resource

1. Đặt file nhạc trong thư mục res/raw của dự án
2. Tạo đối tượng MediaPlayer để chơi nhạc (thông qua hàm static create)
3. Gọi prepare để khởi tạo các thông số cần thiết và bắt đầu chơi nhạc bằng phương thức start()
   - Muốn dừng chơi: gọi phương thức stop()
   - Muốn tạm dừng: sử dụng phương thức pause()

```
MediaPlayer mp = MediaPlayer.create(context, R.raw.sound_1);
mp.prepare();
mp.start();
```

MediaPlayer – example



MediaPlayer – example

```
public class MyPlayer extends Activity {
    MediaPlayer mp = null;
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);
    }
    // xử lý button STOP
    public void btnStop(View v) {
        if (null == mp) return;
        if (mp.isPlaying()) {
            mp.stop();
            mp = null;
        }
  }
```

MediaPlayer – example

```
    // xử lý button PLAY
    public void btnPlay(View v) {
        try {
            mp = MediaPlayer.create(MyPlayer.this, R.raw.kiss_the_rain);
            mp.start();
            mp.setOnCompletionListener(new OnCompletionListener() {
                public void onCompletion(MediaPlayer arg0) {
                    // xử lý khi đã chơi xong
                }
            });
        } catch (Exception e) { }
    }
}
```

Chơi audio từ File/Stream

- Khởi tạo MediaPlayer thông qua hàm khởi tạo
- Gọi phương thức setDataSource(string url) với url là địa chỉ file hay đường dẫn trên internet
- Gọi prepare() để khởi tạo codec phù hợp
- Gọi start() để bắt đầu chơi
- Khi muốn tạm dừng hay dừng hẳn gọi các phương thức pause() hay stop()

```
String PATH_TO_FILE = "/sdcard/my_favorite_song.mp3";
MediaPlayer mp = new MediaPlayer();
mp.setDataSource(PATH_TO_FILE);
mp.prepare();
mp.start();
```

MediaPlayer – example

```
final Button b = (Button) findViewById(R.id.Play);
b.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
                // TODO Auto-generated method stub
                player  = new MediaPlayer();
                try {
                        player.setDataSource(Environment.getExternalStorageDirectory()
                                        .getPath()+ "/Music/test.mp3");
                        player.prepareAsync();
                        player.setOnPreparedListener( new  MediaPlayer.OnPreparedListener(){

                                @Override
                                public void onPrepared(MediaPlayer mp) {
                                        // TODO Auto-generated method stub
                                        mp.start();
                                }
                        });
                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }
});
```

Một số chú ý khi playback

- MediaPlayer cần được reset hoặc release rồi mới được chơi lại nếu trước đó bạn stop
- MediaPlayer chạy ngầm, vì thế nếu bạn đóng activity (finish) thì audio vẫn chạy ngầm (và không có cách nào dừng nó), vì thế nên dùng System.exit để kết thúc ứng dụng
- Có thể chơi cùng lúc nhiều MediaPlayer và có thể thiết lập các mức volume khác nhau cũng như các nguồn ra khác nhau cho từng MediaPlayer (ví dụ chơi 2 file audio và mỗi file đổ âm thanh ra một phía của tai nghe)

Một số chú ý khi playback

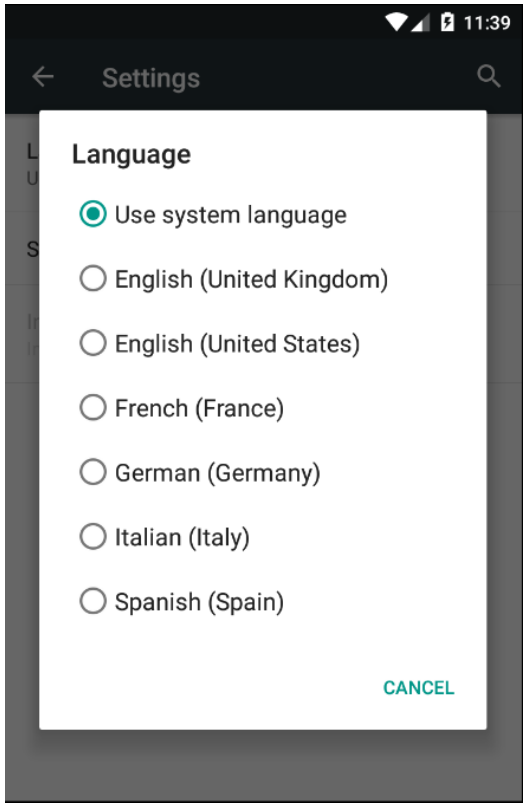
- Muốn ứng dụng chạy ngầm và sau khi ứng dụng quay trở lại vẫn tiếp tục điều khiển được Audio cũ, ta nên sử dụng service
- Muốn tương tác với phím điều khiển âm lượng:
  - Đăng kí broadcast receiver: android.intent.action.MEDIA_BUTTON
  - Điều chỉnh âm thanh bằng phương thức setVolume(left, right), trong đó giá trị left/right là số thực nằm trong khoảng từ 0.0f đến 1.0f
- Sử dụng AudioManager trong trường hợp muốn tương tác nhiều hơn với phần cứng audio

Phần 4

Tổng hợp tiếng nói (text-to-speech)

Text to speech

- Cho phép chuyển đổi từ văn bản sang âm thanh
- Hỗ trợ một số ngôn ngữ (tùy thuộc vào engine trên thiết bị)
- Có một số engine hỗ trợ tiếng Việt tạm ổn như vnSpeak TTS
- Thông thường trên các máy sẽ cài Pico TTS hoặc Google TTS Engine
- TTS rất hữu ích với các ứng dụng có nhu cầu dùng tiếng nói đơn giản và không quan trọng ngữ điệu



Text to speech

- Một số chú ý khi làm việc với TTS
  - Cần kiểm tra xem thiết bị có TTS hay không?
  - Nếu có thì khởi tạo thành công hay không? TTS có hỗ trợ ngôn ngữ và quốc gia bạn muốn hay không?
  - Hiệu chỉnh âm lượng, tốc độ phát âm
  - Có thể chuyển hướng phát âm ra luồng khác hoặc file
  - Có thể tùy biến âm địa phương hoặc khai thác một số API ẩn của engine
- Hai class hữu ích:
  - TextToSpeech: phát âm
  - TextToSpeechService: customize engine

Text to speech

```
// gọi activity mặc định để kiểm tra có dữ liệu cho TTS chưa
Intent intent = new Intent(Engine.ACTION_CHECK_TTS_DATA);
startActivityForResult(intent, CODE);

// xử lý dữ liệu do activity trả về
protected void onActivityResult(int req, int result, Intent data) {
    if (req == CODE)
        if (result != Engine.CHECK_VOICE_DATA_PASS)
            startActivity(new Intent(Engine.ACTION_INSTALL_TTS_DATA));
        else {
            // dữ liệu đã có, tạo đối tượng TTS mặc định
        }
}
```

Text to speech

```
TextToSpeech talker = new TextToSpeech(this, new OnInitListener() {
    public void onInit(int status) {
        talker.speak("Hi! There!", TextToSpeech.QUEUE_FLUSH, null);
    }
}
TextToSpeech tts = new TextToSpeech(this, new OnInitListener() {
    public void onInit(int status) {
        if (status != TextToSpeech.SUCCESS) return;
        tts.setLanguage(Locale.ENGLISH);
        tts.setPitch(0.8f);
        tts.setSpeechRate(1.0f);
    }
});
```

Phần 5

Làm việc với video

Video playback

- Android OS có 2 cách để chơi lại các tập tin video
  - Sử dụng VideoView kết hợp với MediaController
  - Sử dụng MediaPlayer và SurfaceView
- Chơi lại video không yêu cầu quyền gì đặc biệt, nhưng nếu file video ở ngoài internet, thì ứng dụng cần có quyền truy cập internet
- Phương pháp thứ 2 cho phép lập trình viên thiết lập các bộ filter cho hình ảnh phát ra thông qua hàm setPreviewCallback(filter), cần có kiến thức tốt về video nếu muốn viết filter

VideoView + MediaController

- VideoView là view dùng để hiển thị dữ liệu video
- VideoView cung cấp các hàm để điều khiển quá trình chơi video: start, pause, suspend, resume, stopPlayback, seekTo(millis)
- MediaController là widget cung cấp các điều khiển cơ bản cho video, ngoài ra cũng cho lập trình viên tùy biến điều khiển các nút next và prev
- VideoView và MediaController được thiết kế để làm việc với nhau và cùng đáp ứng trải nghiệm người dùng khi chơi video (xử lý các sự kiện chạm)

VideoView + MediaController

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="${packageName}.${activityClass}" >
    <VideoView
        android:id="@+id/videoView1"
        android:layout_width="fill_parent"
        android:layout_height="match_parent"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true" />
</RelativeLayout>
```

VideoView + MediaController

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    VideoView videoView = (VideoView) findViewById(R.id.videoView1);
    videoView.setMediaController(new MediaController(this));
    videoView.setVideoURI(Uri.parse("android.resource://" +
        getPackageName() + "/" + R.raw.teamwork));
    videoView.start();
}
```

VideoView + MediaController

- MediaController được mặc định là ẩn, không cần đặt lên layout khi thiết kế
- MediaController dùng hàm setAnchorView(v) để xác định nó sẽ được gắn vào view nào khi xuất hiện, view mặc định chính là VideoView mà nó điều khiển
- MediaController được ẩn đi sau 5s nếu không có tác động, nếu cần thay đổi những mặc định này thì nên viết lại class MediaController
- VideoView mặc định giữ nguyên tỉ lệ khung hình của video mà nó chạy; muốn thiết lập tỉ lệ khác, chỉ cần điều chỉnh kích thước của VideoView

MediaPlayer + SurfaceView

- MediaPlayer là bộ giải mã, luôn chạy ngầm, hỗ trợ nhiều chuẩn và giao thức video, audio, mạng
- SurfaceView là view được thiết kế với mục đích để thể hiện các hình ảnh cần có tốc độ cập nhật cao, đặc biệt thích hợp với việc thể hiện dữ liệu từ video, camera và hoạt hình
- SurfaceView loại bỏ các chi tiết phức tạp của view nhưng cũng có những hạn chế nhất định
  - Thread có cơ chế cập nhật thẳng vào SurfaceView không cần qua đồng bộ
  - SurfaceView luôn chiếm một vùng màn hình và không thể bị che hoặc có bóng mờ

MediaPlayer + SurfaceView

```
public class MainActivity extends Activity
    implements SurfaceHolder.Callback, OnPreparedListener {

    private MediaPlayer mediaPlayer;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        SurfaceView vidSurface = new SurfaceView(this);
        vidSurface.getHolder().addCallback(this);
        setContentView(vidSurface);
    }
    public void surfaceChanged(SurfaceHolder s, int a, int b, int c) {
    }
    public void surfaceDestroyed(SurfaceHolder arg0) {
    }
```

MediaPlayer + SurfaceView

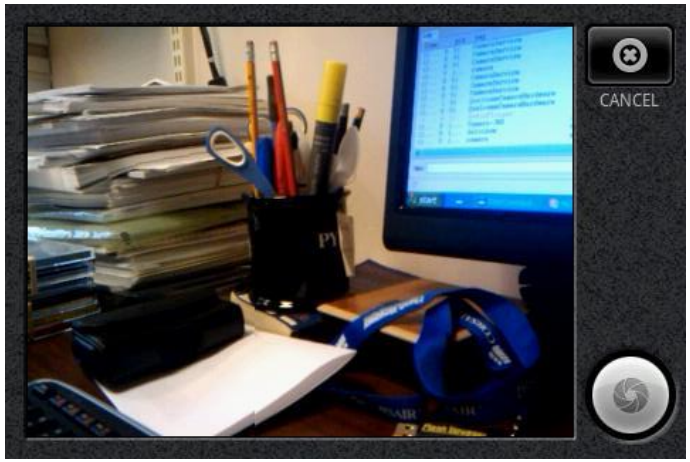
```
    public void surfaceCreated(SurfaceHolder arg0) {
        try {
            mediaPlayer = new MediaPlayer();
            mediaPlayer.setDisplay(arg0);
            mediaPlayer.setDataSource(this,
                Uri.parse("android.resource://" +
                    getPackageName() + "/" + R.raw.teamwork));
            mediaPlayer.prepare();
            mediaPlayer.setOnPreparedListener(this);
            mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
        } catch (Exception e) { }
    }

    public void onPrepared(MediaPlayer mp) { mediaPlayer.start(); }
}
```

Phần 6

Làm việc với camera

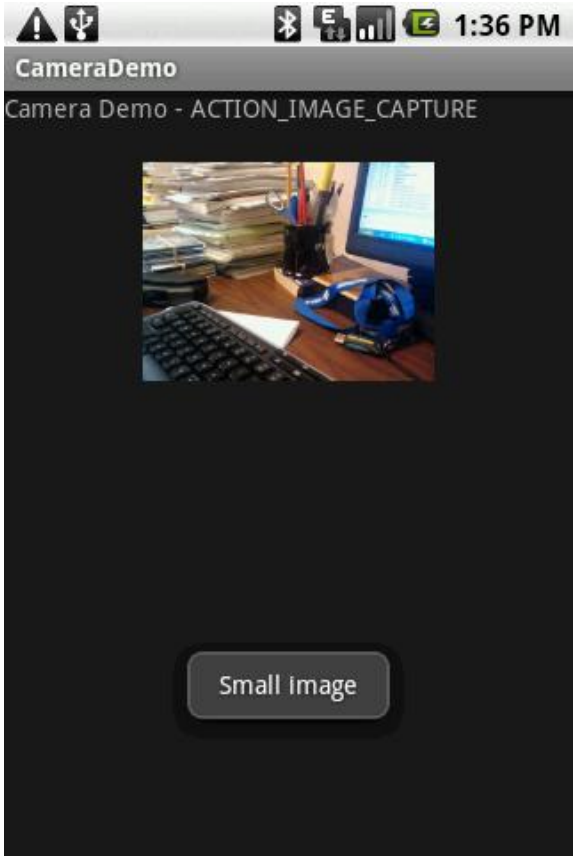
Chụp ảnh bằng camera activity

1. Previewing



2. After shutter is pressed

These views are Provided by the Built-in ACTION_IMAGE_CAPTURE *Intent.*



3. Image transferred from CAMERA to the application

Chụp ảnh bằng camera activity

```
public class CameraDemo extends Activity {
    ImageView mImageView;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mImageView = (ImageView) findViewById(R.id.mImageView);
        Intent mIntent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        startActivityForResult(mIntent, 101);
    }
    private void showToast(Context context, String text) {
        Toast.makeText(context, text, 1).show();
    }
```

Chụp ảnh bằng camera activity

```
    protected void onActivityResult(int req, int res, Intent i) {
        super.onActivityResult(req, res, i);
        if (res == RESULT_CANCELED) return;
        if (req == 101) {
            Bundle b = i.getExtras();
            Bitmap bm = (Bitmap) b.get("data");
            mImageView.setImageBitmap(bm);
        }
    }
}
```

Camera

- Lớp Camera hỗ trợ kết nối và ngắt kết nối tới dịch vụ camera, để cho phép bạn có thể: chụp, quay và sử dụng các tiện ích camera cung cấp
- Dùng hàm open() để lấy về một đối tượng Camera
- Cần khai báo trong Android Manifest để cấp quyền và các tính năng sử dụng Camera
- Xem demo ứng dụng chụp ảnh để hiểu rõ hơn

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

The takePicture Method

```
public final void takePicture ( Camera.ShutterCallback  shutter,
                                Camera.PictureCallback  raw,
                                Camera.PictureCallback  jpeg )
```

Triggers an asynchronous image capture. The camera service will initiate a series of *callbacks* to the application as the image capture progresses.

1. The *shutter callback* occurs after *the image is captured*. This can be used to trigger a sound to let the user know that image has been captured.
2. The *raw callback* occurs when *the raw image data is available* (NOTE: the data may be null if the hardware does not have enough memory to make a copy).
3. The *jpeg callback* occurs when the *compressed image is available*. If the application does not need a particular callback, a null can be passed instead of a callback method.

Parameters

shutter	callback after the image is captured, may be null
raw	callback with raw image data, may be null
jpeg	callback with jpeg image data, may be null