MOBILE APPLICATION DEVELOPMENT

SUBJECT NAME : MOBILE APPLICATION DEVELOPMENT SUBJRECT CODE : CCS51
CLASS : III BSC CS
SEMESTER : ODD

UNIT-V

ADVANCED APPLICATION
Loction based services - android google map - property animation overview Android developers - view animation android developers - animate drawable Graphics android developers - multimedia software - android camera tutorial - Environment sensors android developers.

Content

1. Location based services

You must have used apps like **Google Maps, Waze, MapQuest**, etc. These are the applications that help you track the location of the device. They also provide services like finding nearby restaurants, hospitals, petrol pumps, etc. Even the cab drivers now use maps to locate their route.

As technology is emerging, it becomes quite essential for an android developer to learn about location-based services. Through this article, you will understand the various sections of location-based services(LBS). After understanding the concepts, you will also see an implementation of the same.

1.1 What are Android Location-Based Services?

Location-Based Services(LBS) are present in Android to provide you with features like current location detection, display of nearby places, geofencing, etc. It fetches the location using your device's GPS, Wifi, or Cellular Networks.
To build an app with location-based services, you need to access the Google Play Services Module. After that, you need to use a framework called Location Framework, which has many methods, classes, and interfaces to make your task easier.

1.2 Components of Location-Based Services in Android

The classes and the interfaces present in the Location Framework acts as the essential components for LBS. Those components are as follows:

- **LocationManager Class –** It is used to get Location Service access from the system.
- **LocationListener Interface –** It receives updates from the Location Manager class.
- **LocationProvider –** It is the class that provides us with the location for our devices.
- **Location Class –** Its objects carry information about the location. The information includes latitude, longitude, accuracy, altitude, and speed.

1.3 Location Object

The location objects carry the location of your device. The place is in the form of latitude and longitude. On the location object, you can apply the below methods. The below methods help you to get location and other information regarding the location.

1. float distanceTo(Location destination)

It gives the approximate distance between our current location and the destination location.

2. float getAccuracy()

It gives us the accuracy of our location in metres.

3. double getAltitude()

It gives us the altitude of our place above sea level.

4. double getLatitude()

It gives the latitude coordinate of our place in degrees.

5. double getLongitude()

It gives the latitude coordinate of our place in degrees.
6. float getSpeed()
It gives the speed of our location change.
7. void setAccuracy(float accuracy)
Using setAccuracy(), you can set your custom accuracy in metres.
8. void setAltitude(double altitude)
Using setAltitude(), you can set the altitude of your place from sea level in metres.
9. void setBearing(float bearing)
Using the setBearing() method, you can set location bearing in degrees.
10. void setLatitude(double Latitude)
You can even set your location to some other latitude using the setLatitude() method.
11. void setLongitude(double longitude)
You can even set your location to some other longitude using the setLongitude() method.
12. void setSpeed(float speed)
You can even set speed using the setSpeed() method.
13. void reset()
It is used to reset your set location.
14. boolean hasAccuracy()
It says whether or not the location is accurate.
15. boolean hasAltitude()
It says if the place has an altitude or not
16. boolean hasSpeed()
It is true if the place has a speed attached.
17. boolean hasBearing()
It returns true if the place has a bearing or not.

1.4 Location Quality of Service

Quality of Service(QoS) is enabled through the location request object. Location request object requests the proper and accurate location. You can find below the methods that help in the process.

- **setPriority(int priority) –** It allows us to mark the request with priorities. The higher priority request is executed first.
- **setInterval(long millisecond) –** It allows us to set the intervals on which we seek the location updates.
- **setExpirationDuration(long millisecond) –** It allows us to set the duration of the request after which it shall stop.
- **setExpirationTime(long millisecond) –** It allows us to decide the expiration time of our request.

- **setNumUpdates(int number) –** It allows us to set the number of updates we require for a particular place.

2.Android Google Map

Android provides facility to integrate Google map in our application. Google map displays your current location, navigate location direction, search location etc. We can also customize Google map according to our requirement.

2.1 Types of Google Maps

There are four different types of Google maps, as well as an optional to no map at all. Each of them gives different view on map. These maps are as follow:

1. **Normal:** This type of map displays typical road map, natural features like river and some features build by humans.
2. **Hybrid:** This type of map displays satellite photograph data with typical road maps. It also displays road and feature labels.
3. **Satellite:** Satellite type displays satellite photograph data, but doesn't display road and feature labels.
4. **Terrain:** This type displays photographic data. This includes colors, contour lines and labels and perspective shading.
5. **None:** This type displays an empty grid with no tiles loaded.

2.2 Syntax of different types of map

1. googleMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
2. googleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
3. googleMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
4. googleMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);

2.3 Methods of Google map

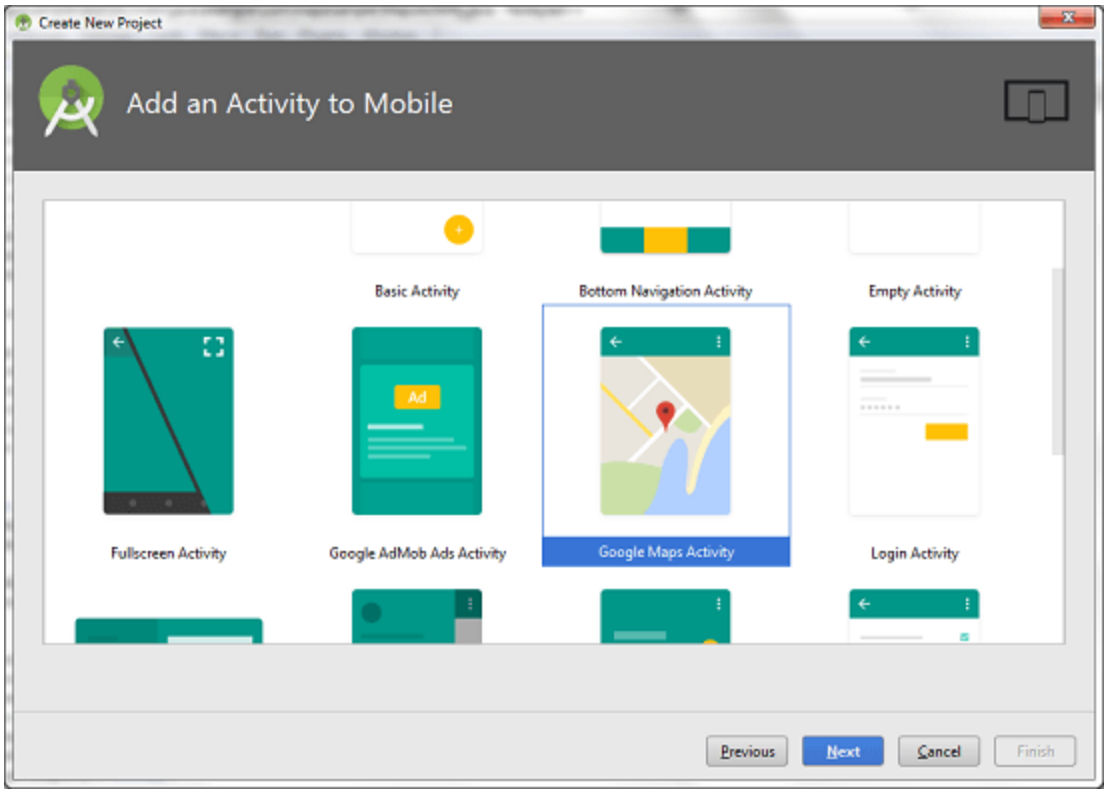
Google map API provides several methods that help to customize Google map. These methods are as following:

Methods	Description
addCircle(CircleOptions options)	This method add circle to map.
addPolygon(PolygonOptions options)	This method add polygon to map.
addTileOverlay(TileOverlayOptions options)	This method add tile overlay to the map.
animateCamera(CameraUpdate update)	This method moves the map according to the update with an animation.

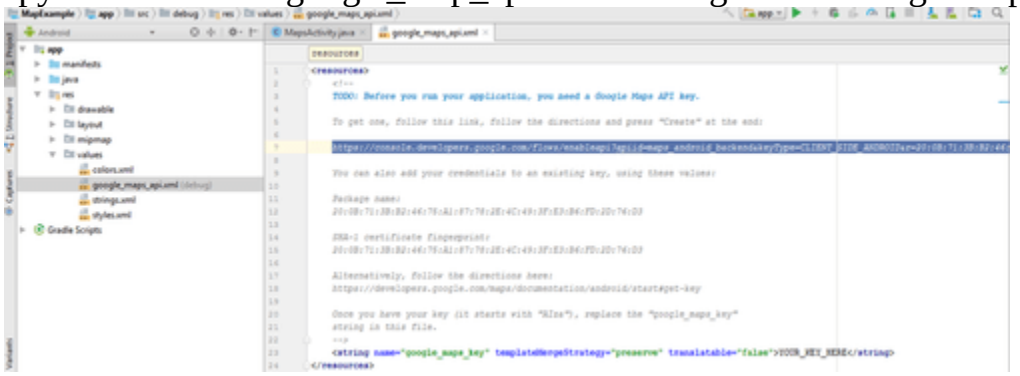
clear()	This method removes everything from the map.
getMyLocation()	This method returns the currently displayed user location.
moveCamera(CameraUpdate update)	This method reposition the camera according to the instructions defined in the update.
setTrafficEnabled(boolean enabled)	This method set the traffic layer on or off.
snapshot(GoogleMap.SnapshotReadyCallback callback)	This method takes a snapshot of the map.
stopAnimation()	This method stops the camera animation if there is any progress.

2.4 Example of Google Map

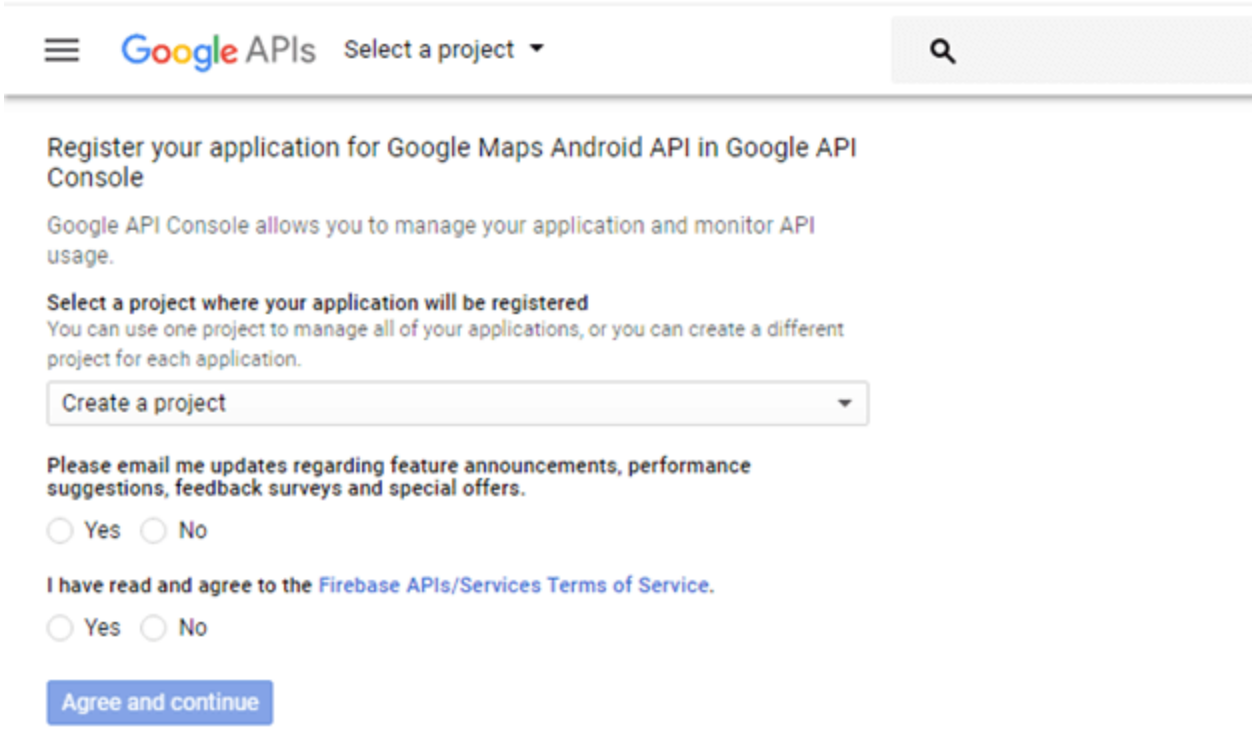
Let's create an example of Google map integrating within our app. For doing this we select Google Maps Activity.



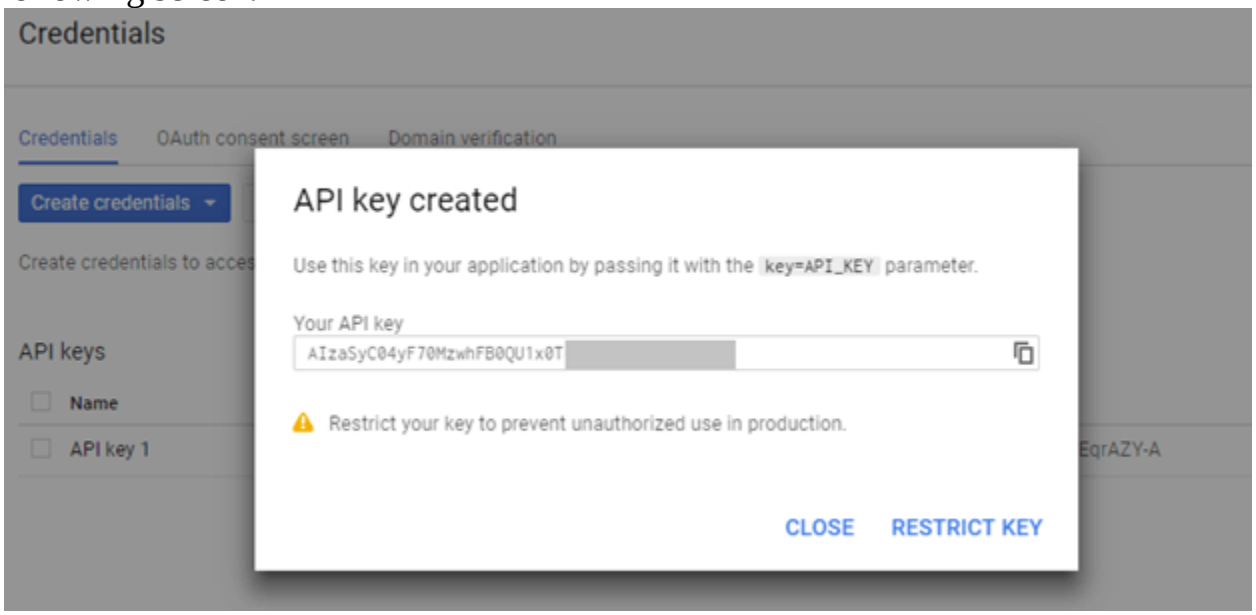
opy the URL from google_map_api.xml file to generate Google map key.



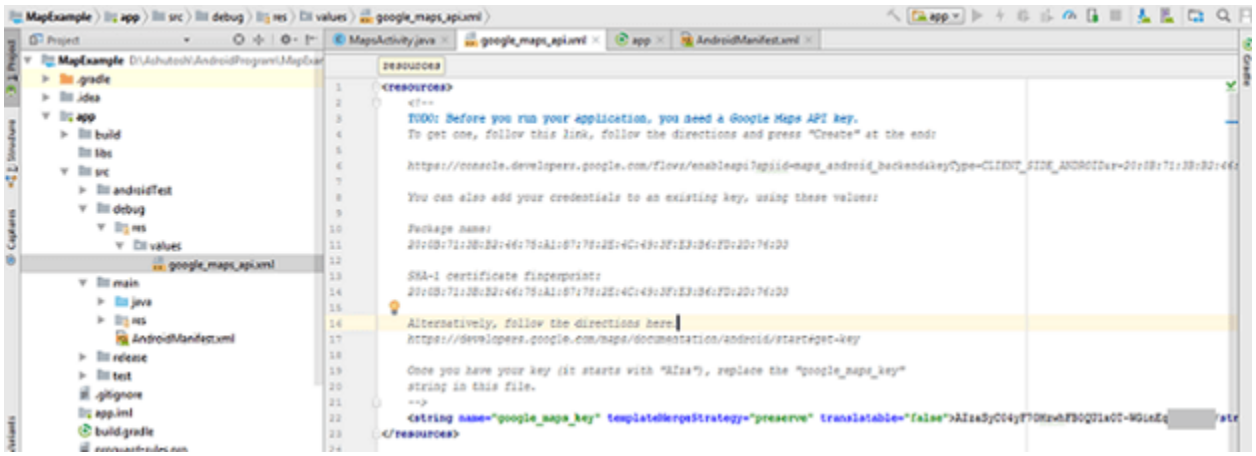
Paste the copied URL at the browser. It will open the following page.



After clicking on Create API key, it will generate our API key displaying the following screen.



Copy this generated API key in our *google_map_api.xml* file



activity_maps.xml

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="example.com.mapexample.MapsActivity" />
```

MapsActivity.java

To get the GoogleMap object in our MapsActivity.java class we need to implement the OnMapReadyCallback interface and override the onMapReady() callback method.

```
package example.com.mapexample;

import android.support.v4.app.FragmentActivity;
import android.os.Bundle;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback{

    private GoogleMap mMap;

```

```
16.    @Override
17.    protected void onCreate(Bundle savedInstanceState) {
18.        super.onCreate(savedInstanceState);
19.        setContentView(R.layout.activity_maps);
20.        // Obtain the SupportMapFragment and get notified when the map is rea
       dy to be used.
21.        SupportMapFragment mapFragment = (SupportMapFragment) getSupp
       ortFragmentManager()
22.                .findFragmentById(R.id.map);
23.        mapFragment.getMapAsync(this);
24.
25.    }
26.
27.    @Override
28.    public void onMapReady(GoogleMap googleMap) {
29.        mMap = googleMap;
30.
31.        // Add a marker in Sydney and move the camera
32.        LatLng sydney = new LatLng(-34, 151);
33.        mMap.addMarker(new MarkerOptions().position(sydney).title("Marker
        in Sydney"));
34.        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
35.
36.    }
37.}
```

Required Permission

Add the following user-permission in AndroidManifest.xml file.

```
1. <uses-
   permission android:name="android.permission.ACCESS_FINE_LOCATIO
   N" />
2. <uses-
   permission android:name="android.permission.ACCESS_COARSE_LOCA
   TION" />
3. <uses-permission android:name="android.permission.INTERNET" />
```

AndroidManifest.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3.     package="example.com.mapexample">
4.     <!--
```

```
5.          The ACCESS_COARSE/FINE_LOCATION permissions are not requi
    red to use
6.          Google Maps Android API v2, but you must specify either coarse or fi
    ne
7.          location permissions for the 'MyLocation' functionality.
8.      -->
9.      <uses-
    permission android:name="android.permission.ACCESS_FINE_LOCATIO
    N" />
10.     <uses-
    permission android:name="android.permission.ACCESS_COARSE_LOCA
    TION" />
11.     <uses-permission android:name="android.permission.INTERNET" />
12.
13.     <application
14.         android:allowBackup="true"
15.         android:icon="@mipmap/ic_launcher"
16.         android:label="@string/app_name"
17.         android:roundIcon="@mipmap/ic_launcher_round"
18.         android:supportsRtl="true"
19.         android:theme="@style/AppTheme">
20.
21.         <meta-data
22.             android:name="com.google.android.geo.API_KEY"
23.             android:value="@string/google_maps_key" />
24.
25.         <activity
26.             android:name=".MapsActivity"
27.             android:label="@string/title_activity_maps">
28.             <intent-filter>
29.                 <action android:name="android.intent.action.MAIN" />
30.
31.                 <category android:name="android.intent.category.LAUNCHER"
    />
32.             </intent-filter>
33.         </activity>
34.     </application>
35.
36. </manifest>
```

METHODS TO GET CURRENT LOCATION

Sr.No.	Callback Methods & Description
1	**abstract void onConnected(Bundle connectionHint)** This callback method is called when location service is connected to the location client successfully. You will use **connect()** method to connect to the location client.
2	**abstract void onDisconnected()** This callback method is called when the client is disconnected. You will use **disconnect()** method to disconnect from the location client.
3	**abstract void onConnectionFailed(ConnectionResult result)** This callback method is called when there was an error connecting the client to the service.

You should create the location client in **onCreate()** method of your activity class, then connect it in **onStart()**, so that Location Services maintains the current location while your activity is fully visible. You should disconnect the client in **onStop()** method, so that when your app is not visible, Location Services is not maintaining the current location. This helps in saving battery power up-to a large extent.

GET UPDATED LOCATION

sr.No.	Callback Method & Description
1	**abstract void onLocationChanged(Location location)** This callback method is used for receiving notifications from the LocationClient when the location has changed.

EXAMPLE CODING

Keeping this application simple we have just added a SupportMapFragment tag inside content_main.xml as shown below:

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android=https://schemas.android.com.apk/res/android

   xmlns:app=https://schemas.android.com.apk/res/-auto

xmlns:tools=https://schemas.android.com/tools

android:layout_width="match_parent"
```

android:layout_height="mtch_parent"

app:layout_behavoiur="@string/appbar_scrolling_view_behaviour"

tools:context="com.journaldev.interatingmaps.MainActivity"

tools:showIn="@layout/activity_main">

<fragment

Android:id="@+id/map"android:name="com.google.android.gms.maps.SupportMapFragment"

Android:layout_width="match_parent"

Android:layout_gravity="center"

Android:layout_height="match+parent"

/>

</RelativeLayout>

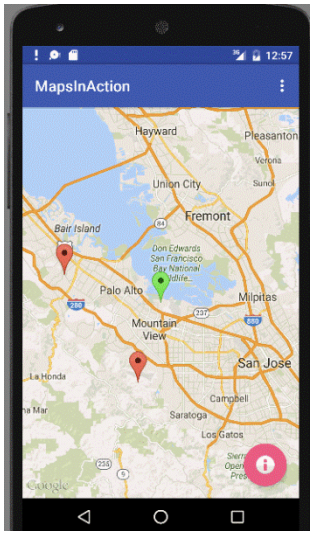
Add the following permission to the AndroidManifest.xml.file.

<uses-permission android:name="amdroid.permission.INTERNET"/>

Build and run this application on the emulator.

Just go to Settings->Apps->Google Play Services nd use that version number in the build.gradle in place of: compile 'com.google.android.gms:play=services:8.3.0'

The following output will be shown:



3.ANIMATION

Android provides a large number of classes and interface for the animation development. Most of the classes and interfaces are given in **android.animation** package.

Android Animation enables you to change the object property and behavior at run time. There are various ways to do animation in android.

The *AnimationDrawable* class provides methods to start and end the animation. Even, you can use time based animation.

Let's have a look at the simple example of android animation.

activity_main.xml

You need to have a view only.

File: activity_main.xml

```
<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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <View
        />

</RelativeLayout>
```

File: logo.xml

Have a image view only.

```
<?xml version="1.0" encoding="utf-8"?>
<ImageView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/anm"
     >

</ImageView>
```

MainActivity class

File: MainActivity.java

```
package com.javatpoint.animation;

import android.os.Bundle;
import android.app.Activity;
import android.graphics.drawable.AnimationDrawable;
import android.widget.ImageView;

public class MainActivity extends Activity {

   ImageView anm;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.logo);
        anm = (ImageView)findViewById(R.id.anm);

        anm.setBackgroundResource(R.drawable.animation);
      // the frame-by-frame animation defined as a xml file within the drawable folder

        /*
         * NOTE: It's not possible to start the animation during the onCreate.
         */
     }
    public void onWindowFocusChanged (boolean hasFocus) {
        super.onWindowFocusChanged(hasFocus);
        AnimationDrawable frameAnimation =
            (AnimationDrawable) anm.getBackground();
        if(hasFocus) {
            frameAnimation.start();
        } else {
            frameAnimation.stop();
```

```
            }
        }

}
```

You need to create animation.xml file inside res/drawable-hdpi directory.

You need to have many images. Here, we are using 14 images and all the 14 images are located inside res/drawable-mdpi directory.

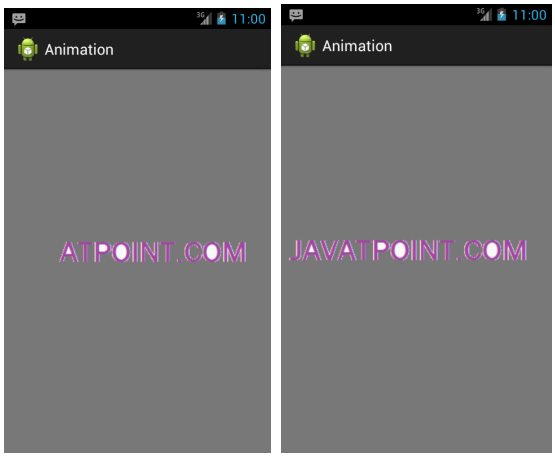
File: animation.xml

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false">

    <item android:drawable="@drawable/frame0" android:duration="120" />
    <item android:drawable="@drawable/frame1" android:duration="120" />
    <item android:drawable="@drawable/frame2" android:duration="120" />
    <item android:drawable="@drawable/frame3" android:duration="120" />
    <item android:drawable="@drawable/frame4" android:duration="120" />
    <item android:drawable="@drawable/frame5" android:duration="120" />
    <item android:drawable="@drawable/frame6" android:duration="120" />
    <item android:drawable="@drawable/frame7" android:duration="120" />
    <item android:drawable="@drawable/frame8" android:duration="120" />
    <item android:drawable="@drawable/frame9" android:duration="120" />
    <item android:drawable="@drawable/frame10" android:duration="120" />
    <item android:drawable="@drawable/frame11" android:duration="120" />
    <item android:drawable="@drawable/frame12" android:duration="120" />
    <item android:drawable="@drawable/frame13" android:duration="120" />
    <item android:drawable="@drawable/frame14" android:duration="120" />
    <item android:drawable="@drawable/frame14" android:duration="120" />
    <item android:drawable="@drawable/frame13" android:duration="120" />
    <item android:drawable="@drawable/frame12" android:duration="120" />
    <item android:drawable="@drawable/frame11" android:duration="120" />
    <item android:drawable="@drawable/frame10" android:duration="120" />
    <item android:drawable="@drawable/frame9" android:duration="120" />
    <item android:drawable="@drawable/frame8" android:duration="120" />
    <item android:drawable="@drawable/frame7" android:duration="120" />
    <item android:drawable="@drawable/frame6" android:duration="120" />
    <item android:drawable="@drawable/frame5" android:duration="120" />
    <item android:drawable="@drawable/frame4" android:duration="120" />
    <item android:drawable="@drawable/frame3" android:duration="120" />
    <item android:drawable="@drawable/frame2" android:duration="120" />
    <item android:drawable="@drawable/frame1" android:duration="120" />
    <item android:drawable="@drawable/frame0" android:duration="120" />
```

35.
36. **</animation-list>**

Output:



3.1View Animation

You can use the view animation system to perform tweened animation on Views. Tween animation calculates the animation with information such as the start point, end point, size, rotation, and other common aspects of an animation.

A tween animation can perform a series of simple transformations (position, size, rotation, and transparency) on the contents of a View object. So, if you have a TextView object, you can move, rotate, grow, or shrink the text. If it has a background image, the background image will be transformed along with the text. The animation package provides all the classes used in a tween animation.

A sequence of animation instructions defines the tween animation, defined by either XML or Android code. As with defining a layout, an XML file is recommended because it's more readable, reusable, and swappable than hard-coding the animation. In the example below, we use XML. (To learn more about defining an animation in your application code, instead of XML, refer to the AnimationSet class and other Animation subclasses.)

The animation instructions define the transformations that you want to occur, when they will occur, and how long they should take to apply. Transformations can be sequential or simultaneous - for example, you can have the contents of a TextView move from left to right, and then rotate 180 degrees, or you can have the text move and rotate simultaneously. Each transformation takes a set of parameters specific for that transformation (starting size and ending size for size change, starting angle and ending angle for rotation, and so on), and also a set of common parameters (for instance, start time and duration). To make several transformations happen simultaneously, give them the same start time; to make them sequential, calculate the start time plus the duration of the preceding transformation.

The animation XML file belongs in the res/anim/ directory of your Android project. The file must have a single root element: this will be either a single <alpha>, <scale>, <translate>, <rotate>, interpolator element, or <set> element that holds groups of these elements (which may include another <set>). By default, all animation instructions are applied simultaneously. To make them occur sequentially, you must specify the startOffset attribute, as shown in the example below.

The following XML from one of the ApiDemos is used to stretch, then simultaneously spin and rotate a View object.

```
<set android:shareInterpolator="false">
  <scale
    android:interpolator="@android:anim/accelerate_decelerate_interpolator"
    android:fromXScale="1.0"
    android:toXScale="1.4"
    android:fromYScale="1.0"
    android:toYScale="0.6"
    android:pivotX="50%"
    android:pivotY="50%"
    android:fillAfter="false"
    android:duration="700" />
  <set android:interpolator="@android:anim/decelerate_interpolator">
    <scale
      android:fromXScale="1.4"
      android:toXScale="0.0"
      android:fromYScale="0.6"
      android:toYScale="0.0"
      android:pivotX="50%"
      android:pivotY="50%"
      android:startOffset="700"
      android:duration="400"
      android:fillBefore="false" />
    <rotate
      android:fromDegrees="0"
      android:toDegrees="-45"
      android:toYScale="0.0"
      android:pivotX="50%"
      android:pivotY="50%"
      android:startOffset="700"
      android:duration="400" />
  </set>
</set>
```

Screen coordinates (not used in this example) are (0,0) at the upper left hand corner, and increase as you go down and to the right.

Some values, such as pivotX, can be specified relative to the object itself or relative to the parent. Be sure to use the proper format for what you want ("50" for 50% relative to the parent, or "50%" for 50% relative to itself).

You can determine how a transformation is applied over time by assigning an Interpolator. Android includes several Interpolator subclasses that specify various speed curves: for instance, AccelerateInterpolator tells a transformation to start slow and speed up. Each one has an attribute value that can be applied in the XML.

With this XML saved as hyperspace_jump.xml in the res/anim/ directory of the project, the following code will reference it and apply it to an ImageView object from the layout.

3.2Android Property Animation — The ValueAnimator

ValueAnimator provides a timing engine for running animation which calculates the animated values and set them on the target objects. By ValueAnimator you can animate a view **width**, **height**, update its **x** and **y** coordinates or even can change its **background**.

It has an interface **ValueAnimator.AnimatorUpdateListener** which is used to receive callbacks on every animation frame.

```
ValueAnimator widthAnimator = ValueAnimator.ofInt(10, 100);
widthAnimator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
   @Override
   public void onAnimationUpdate(ValueAnimator animation) {
      int animatedValue = (int) animation.getAnimatedValue();
      someView.getLayoutParams().width = animatedValue;
      someView.requestLayout();
   }
});
```

In above example, I'm updating the width of a view from **10** to **100**.
In **onAnimationUpdate()** method the animated value is used to update the width of that view. On calling **requestLayout()** the framework will redraw the view according to updated width value (this method is needed to be called on every change of value otherwise the view will not be redrawn according to updated value and effect would not be seen).

Besides this simple example, let's have a look at a little complex one. In this example, I'm going to change **x** and **y coordinates** and **size** of the button on click of a **floating action button**. And result is shown below —

4.Graphics android developer

The **android.graphics.Canvas** can be used to draw graphics in android. It provides methods to draw oval, rectangle, picture, text, line etc.
The **android.graphics.Paint** class is used with canvas to draw objects. It holds the information of color and style.
In this example, we are going to display 2D graphics in android
e: activity_main.xml

```
1. <RelativeLayout xmlns:androclass="http://schemas.android.com/apk/res/an
   droid"
2.     xmlns:tools="http://schemas.android.com/tools"
3.     android:layout_width="match_parent"
4.     android:layout_height="match_parent"
5.     android:paddingBottom="@dimen/activity_vertical_margin"
6.     android:paddingLeft="@dimen/activity_horizontal_margin"
7.     android:paddingRight="@dimen/activity_horizontal_margin"
8.     android:paddingTop="@dimen/activity_vertical_margin"
9.     tools:context=".MainActivity" >
10.
11.    <TextView
12.        android:layout_width="wrap_content"
13.        android:layout_height="wrap_content"
14.        android:text="@string/hello_world" />
15.
16. </RelativeLayout>
```

Activity class

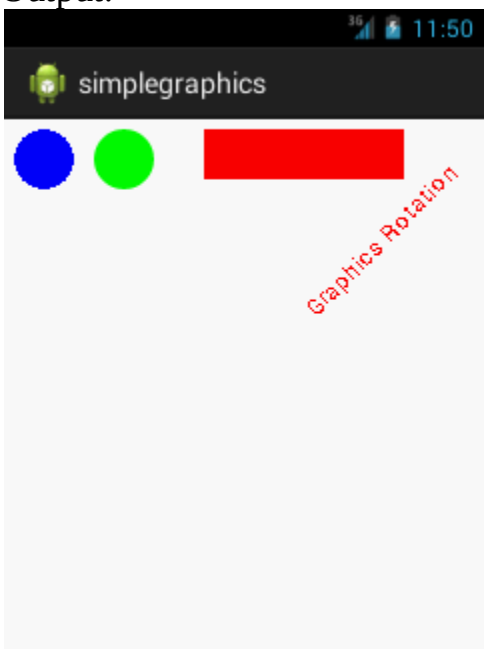
File: MainActivity.java

```
1.  package com.example.simplegraphics;
2.
3.  import android.os.Bundle;
4.  import android.app.Activity;
5.  import android.view.Menu;
6.  import android.content.Context;
7.  import android.graphics.Canvas;
8.  import android.graphics.Color;
9.  import android.graphics.Paint;
10. import android.view.View;
11.
12. public class MainActivity extends Activity {
13.
14.    DemoView demoview;
15.    /** Called when the activity is first created. */
16.    @Override
17.    public void onCreate(Bundle savedInstanceState) {
18.        super.onCreate(savedInstanceState);
19.        demoview = new DemoView(this);
20.        setContentView(demoview);
21.    }
```

```
22.
23.   private class DemoView extends View{
24.       public DemoView(Context context){
25.           super(context);
26.       }
27.
28.       @Override protected void onDraw(Canvas canvas) {
29.           super.onDraw(canvas);
30.
31.           // custom drawing code here
32.           Paint paint = new Paint();
33.           paint.setStyle(Paint.Style.FILL);
34.
35.           // make the entire canvas white
36.           paint.setColor(Color.WHITE);
37.           canvas.drawPaint(paint);
38.
39.           // draw blue circle with anti aliasing turned off
40.           paint.setAntiAlias(false);
41.           paint.setColor(Color.BLUE);
42.           canvas.drawCircle(20, 20, 15, paint);
43.
44.           // draw green circle with anti aliasing turned on
45.           paint.setAntiAlias(true);
46.           paint.setColor(Color.GREEN);
47.           canvas.drawCircle(60, 20, 15, paint);
48.
49.           // draw red rectangle with anti aliasing turned off
50.           paint.setAntiAlias(false);
51.           paint.setColor(Color.RED);
52.           canvas.drawRect(100, 5, 200, 30, paint);
53.
54.           // draw the rotated text
55.           canvas.rotate(-45);
56.
57.           paint.setStyle(Paint.Style.FILL);
58.           canvas.drawText("Graphics Rotation", 40, 180, paint);
59.
60.           //undo the rotate
61.           canvas.restore();
```

```
    }
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}
}
```

Output:



5.MULTIMEDIA SOFTWARE

Android supports multimedia by using the open source multimedia system called *OpenCORE* from PacketVideo Corporation. OpenCORE provides the foundation for Android's media services, which Android wraps in an easy-to-use API. In addition to the OpenCORE framework, the Android platform is migrating to a Google-written multimedia framework named Stagefright. Both frameworks are provided in version 2.2 (Froyo) and in subsequent versions of the SDK. It's anticipated that most, if not all, of the multimedia functionality will be handled by the Stagefright code base.

In this chapter, we'll look at OpenCORE's multimedia architecture and features, and then use it via Android's MediaPlayer API to play audio files, take a picture, play videos, and finally record video and audio from the emulator. To begin, let's look at OpenCORE's multimedia architecture.

Get Android in Action, Second Edition

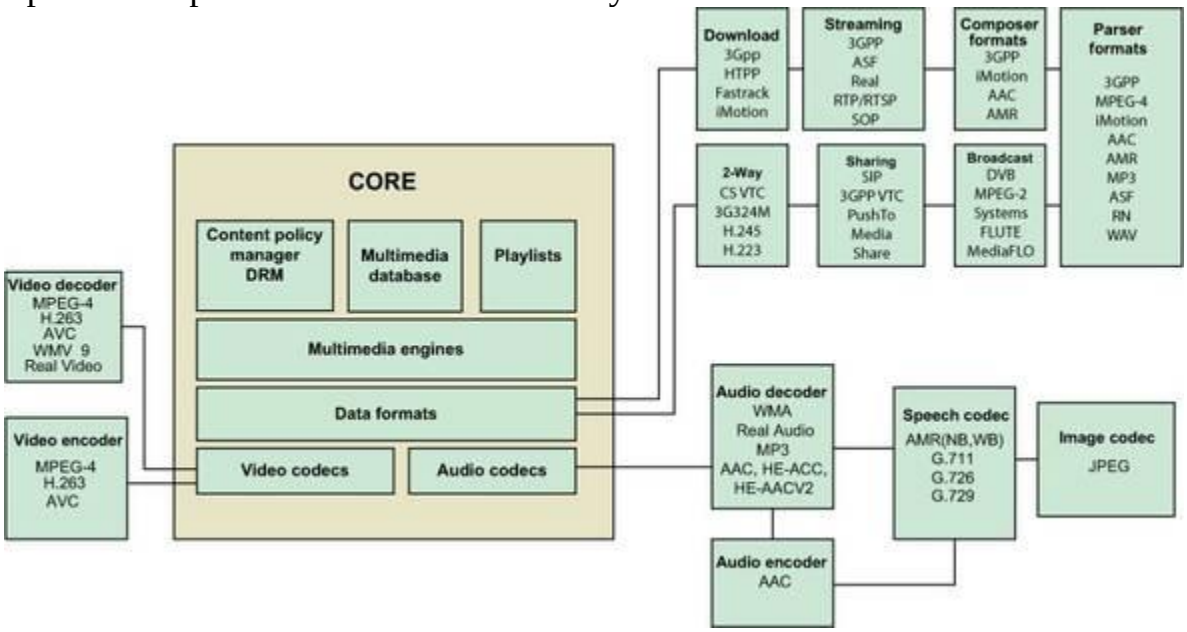
add to cart

5.1 Introduction to multimedia and OpenCORE

Because the foundation of Android's multimedia platform is PacketVideo's Open-CORE, we're going to review OpenCORE's architecture and services. OpenCORE is a Java open source, multimedia platform that supports:

- Interfaces for third-party and hardware media codecs, input and output devices, and content policies
- Media playback, streaming, downloading, and progressive playback, including 3rd Generation Partnership Program (3GPP), Moving Picture Experts Group 4 (MPEG-4), Advanced Audio Coding (AAC), and Moving Picture Experts Group Audio Layer 3(MP3) containers
- Video and image encoders and decoders, including MPEG-4, International Telecommunication Union H.263 video standard (H.263), Advanced Video Coding (AVC H.264), and the Joint Photographic Experts Group (JPEG)
- Speech codecs, including Adaptive Multi-Rate audio codecs AMR-NB and AMR-WB
- Audio codecs, including MP3, AAC, and AAC+
- Media recording, including 3GPP, MPEG-4, and JPEG
- Video telephony based on the 3GPP video conferencing standard 324-M
- PV test framework to ensure robustness and stability; profiling tools for memory and CPU usage

OpenCORE provides all this functionality in a well-laid-out set of services.



5.2 Playing audio

Probably the most basic need for multimedia on a cell phone is the ability to play audio files, whether new ringtones, MP3s, or quick audio notes. Android's Media Player is easy to use. At a high level, all you need to do to play back an MP3 file is follow these steps:

1. Ery rdo MP3 nj vgr re/wrsa directory nj z project (vxnr rzrb xqd nsz cfvc ckb s OYJ rk access files kn xrp roknewt et jec rog itnentre).

2. Treeta s vwn instance kl rdx MediaPlayer pnz ceereefnr xbtg MP3 ud agcliln MediaPlayer.create().

3. Call the MediaPlayer methods prepare() and start().

For'z eotw rhougth nc amepexl xr odtasnmeret ykw mlsipe parj rcse aj. Vartj, cartee s wxn project dlcale MediaPlayer Vmpexal, rpwj nz Activity ecdlal MediaPlayer-Activity. Gwv, cateer z nwk relfod rdnue dcrle/aels tsw. Mx'ff teosr vgt MP3 a nj crqj fodlre. Ext jbar leapemx, wo'ff xzq z ngeirton etl vrb hmxz Hfez 3, ihhcw kbb znc tvreriee klmt MediaPlayer.create. Udnwolao grv Hxzf 3 etmhe navd nzb hns oerth MP3 a xdb yacnf, nsh hdr kmur jn xrp tzw directory. Oxrk, aretce z mplsei Button vtl rgv mcisu ylrpea, cc nhwos jn rog nowgfoill itingsl.

Listing 10.1. main.xml for MediaPlayer Example

<?xml version="1.0" encoding="utf-8"?> <**LinearLayout** xmlns:android="http://schemas.android.com/apk/res/android" android:orientation="vertical" android:layout_width="fill_parent" android:layout_height="fill_parent" > <**TextView** android:layout_width="fill_parent" android:layout_height="wrap_content" android:text="Simple Media Player" /> <**Button** android:id="@+id/playsong" android:layout_width="fill_parent" android:layout_height="wrap_content" android:text="Halo 3 Theme Song" /> </**LinearLayout**>

copy

Mv nykv xr fljf vrb eyt MediaPlayerActivity class, zz snhwo nj qkr iolngfowl islgtni.

Listing 10.2. MediaPlayerActivity.java

```
public class MediaPlayerActivity extends Activity {
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);
        Button mybutton = (Button) findViewById(R.id.playsong);
        mybutton.setOnClickListener(new Button.OnClickListener() {
               public void onClick(View v) {
                   MediaPlayer mp =
 MediaPlayer.create(MediaPlayerActivity.this,
 R.raw.halotheme);
                  mp.start();
                  mp.setOnCompletionListener(new OnCompletionListener(){
                         public void onCompletion(MediaPlayer arg0) {

                         }
                    }
                    );
               }
        }
        );
    }
}
```

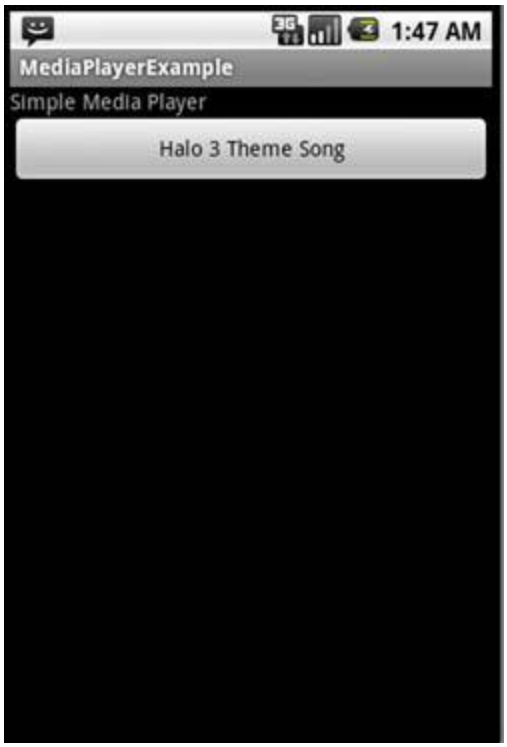
1 Set view and button to play MP3

2 Get context and play MP3

As you can see, playing back an MP3 is easy. In listing 10.2, all we did was use the view that we created in listing 10.1 and map the button, playsong, to mybutton, which we then bound to the setOnClickListener() ❶. Inside the listener, we created the MediaPlayer instance ❷ using the create(Context context, int resourceid) method, which takes our context and a resource ID for our MP3. Finally, we set the setOnCompletionListener, which will perform some task on completion. For the moment, we do nothing, but you might want to change a button's state or provide a notification to a user that the song is over, or ask if the user would like to play another song. If you want to do any of these things, you'd use this method.

Ukw jl ueg lmioecp opr application sun tbn jr, xdb ludhso vkz egnhtismo vfjx figure 10.2. Bfvja ryo uttbon, cnh qgv housld kcut pvr Hefc 3 nkad yedalp dsoz jn uor emulator xjc vqtg kprsesea. Rxg anc czfk orcntlo ukr molvue xl rkg ylacbkpa rgwj rkg evmulo switches xn xrq uvaj xl xur Android emulator phone visualization.

Simple media player example



Kwk zryr wx'vk edloko sr ewp rx ysfb sn audio file, rfo'a xxa wxq qvh anz fhqs s video file.

5.3. Playing video

Playing a video is slightly more complicated than playing audio with the MediaPlayer API, in part because you have to provide a view surface for your video to play on. Android has a VideoView widget that handles that task for you; you can use it in any layout manager. Android also provides a number of display options, including scaling and tinting. So let's get started with playing video by creating a new project called Simple Video Player. Next, create a layout, as shown in the following listing.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
    <VideoView android:id="@+id/video"
    android:layout_width="320px"
    android:layout_height="240px"
    />/
</LinearLayout>
```

1 Add VideoView widget

Cff wx'xk nxyx nj jzrq tislngi cj cqh rop VideoView giedwt ❶.
Bbv VideoView sdporeiv z DJ dgewti jpwr arvg, yfcb, deacvna, irdwen, unc ohetr osbtnut, kiagnm rj ueneysnsrca er hhs thpe vwn. Urko, ow ounk re riewt s class xr pgzf rxb video, hchiw jc snwoh nj org wonfglloi intlsig.

```
public class SimpleVideo extends Activity {

    private VideoView myVideo;
    private MediaController mc;
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        getWindow().setFormat(PixelFormat.TRANSLUCENT);
        setContentView(R.layout.main);
        this.myVideo = (VideoView) findViewById(R.id.video);
        this.myVideo.setVideoPath("sdcard/test.mp4");
        this.mc = new MediaController(this);
        this.mc.setMediaPlayer(this.myVideo);
        this.myVideo.setMediaController(this.mc);
        this.myVideo.requestFocus();
    }
}
```

❶ **Create translucent window**

In this listing, we first created a translucent window, which is necessary for our SurfaceView ❶. Next, we reference the VideoView as a container for playing the video, and use its setVideoPath() to have it look at an SD card for our test MP4. Finally, we set up the MediaController and use the setMediaController() to perform a callback to the VideoView to notify it when our video is finished playing.

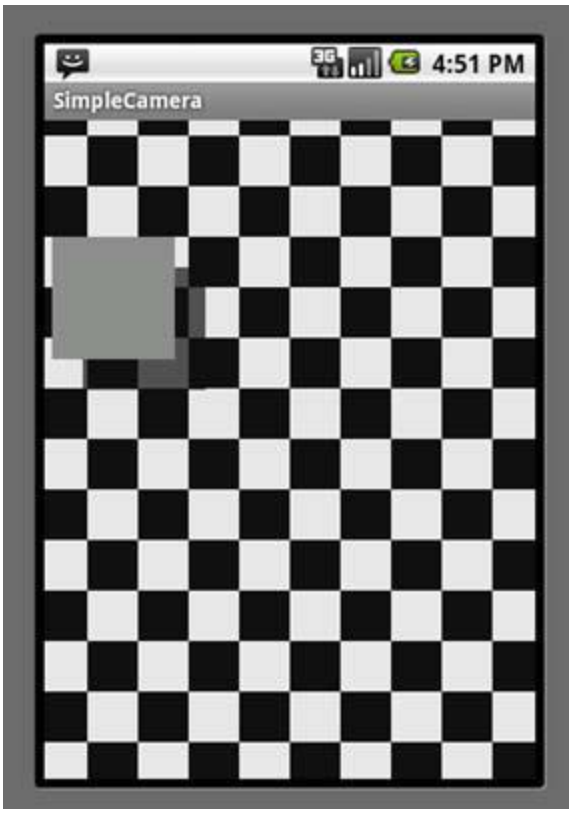
6.ANDROID CAMERA

n important feature of modern cell phones is their ability to take pictures or video using a built-in camera. Some phones even support using the camera's microphone to capture audio. Android, of course, supports all three features and provides a variety of ways to interact with the camera. In this section, we're going to look at how to interact with the camera and take photographs. In the next section, you'll use the camera to take video and save it to an SD card.

Thk'ff do creating z wno project dlealc SmpielYaraem rk ertoadmtnse weq vr tconcen vr z phone 'a aemacr xr uceptar maiegs. Pkt zjru project, wk'ff go ungsi dvr `Camera` class (http://code.google.com/android/reference/android/hardware/Camera.html) rx jor dvr emulator 'c (et phone 'c) rmacae re s `View`. Wcrv el urv xpkz rrus wx'xk ereatcd tlk rjau project seald jywr sihnwog rxb input lmkt rog aamerc, brh rkd nmzj tkvw lte ngkati z upciret ja vonq ph c ngilse method acedll `take-Picture(Camera.ShutterCallback shutter, Camera.PictureCallback raw, Camera.PictureCallback jpeg)`, hcwhi szd erteh kcalcsbla srur wlola pxp rv tcolnor wvq c peucitr jz anetk.

Reefor wo vrb nqs hrfuter nrjv pvr `Camera` class pzn vwd vr cho rou ceaamr, vrf'c arceet ord project. Mk'ff vp creating wrx classes; ebacesu xru cmnj class zj neqf, wk'ff arkeb jr jnrv rwk eossticn. Ptx ory strfi iotcsne, kfok cr ukr olngfoliw intsgil, Braema-Fleapxm.zico.

Test pattern coming from the emulator camera and displayed in the SimpleCamera application



CameraExample.java

```
public class SimpleCamera extends Activity implements
SurfaceHolder.Callback
 {
    private Camera camera;
    private boolean isPreviewRunning = false;
    private SimpleDateFormat timeStampFormat = new
        SimpleDateFormat("yyyyMMddHHmmssSS");
    private SurfaceView surfaceView;
    private SurfaceHolder surfaceHolder;
    private Uri targetResource = Media.EXTERNAL_CONTENT_URI;
    public void onCreate(Bundle icicle)
    {
    super.onCreate(icicle);
    Log.e(getClass().getSimpleName(), "onCreate");
    getWindow().setFormat(PixelFormat.TRANSLUCENT);
    setContentView(R.layout.main);
    surfaceView = (SurfaceView)findViewById(R.id.surface);
    surfaceHolder = surfaceView.getHolder();
    surfaceHolder.addCallback(this);
    surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
}
@Override
public boolean onCreateOptionsMenu(android.view.Menu menu) {
    MenuItem item =
menu.add(0, 0, 0, "View Photos?");
    item.setOnMenuItemClickListener(new
```

1 Create menu to Android's Photo Gallery

```
            MenuItem.OnMenuItemClickListener() {
            public boolean onMenuItemClick(MenuItem item) {
                Intent intent = new Intent(Intent.ACTION_VIEW,
                    SimpleCamera.this.targetResource);
                startActivity(intent);
                return true;
            }
        });
        return true;
    }
    @Override
    protected void onRestoreInstanceState(Bundle savedInstanceState)
{
        super.onRestoreInstanceState(savedInstanceState);
    }
    Camera.PictureCallback mPictureCallbackRaw = new
        Camera.PictureCallback() {
        public void onPictureTaken(byte[] data, Camera c) {
            SimpleCamera.this.camera.startPreview();
        }
    };
    Camera.ShutterCallback mShutterCallback = new Camera.ShutterCallback()
{
        Public void onShutter() {}
        }
    };
```

2 Create PictureCallback

3 Create ShutterCallback

Xqcj igiltns cj dtaarorhigwtsfr. Erzjt, vw zor riaevalsb ltk managing c `surfaceView` chn rvny arv qd xrb `View`. Kvrk, ow reteac z psmiel vmpn znq nmxh ponoti brrz fwjf fotla toxv txy feacsru yvwn roy user lkccis rob Menu btnuto en oqr phone elhiw oru application jc rniungn ❶. Uunej ze fwfj nukx Android 'z erpucit bworrse snq xfr krp user view kry ptoohs ne rkb rcamea. Uvor, wv cereta bkr ftisr `PictureCallback`, ihwhc aj lcedal woyn z icueprt jc itrfs antke ❷. Rdjz sfitr balckcla psceuatr rob `Picture-Callback` 'c fnxu method, `onPictureTaken(byte[] data, Camera camera)`, vr tcuu xdr stw megai data eclrdyit lvmt ogr aacemr. Oxor, kw tareec z `ShutterCallback`, wchhi snz pv dpao wrdj jrz method, `onShutter()`, re fcqh c sduno; tpox ow pnx'r affz prk method ❸. Mv'ff tenniuco rdjw rgo XaarmeLlmeaxp.eczi jn oru krnk glntsii.

CameraExample.java continued

```
@Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {

        ImageCaptureCallback camDemo = null;
        if(keyCode == KeyEvent.KEYCODE_DPAD_CENTER) {
            try {
                String filename = this.timeStampFormat.format(new Date());
                ContentValues values = new ContentValues();
                values.put(MediaColumns.TITLE, filename);
                values.put(ImageColumns.DESCRIPTION,
                    "Image from Android Emulator");
                Uri uri =
                 getContentResolver().insert(
                Media.EXTERNAL_CONTENT_URI, values);
                camDemo = new ImageCaptureCallback(
                    getContentResolver().openOutputStream(uri));
            } catch(Exception ex ){
            }
        }
        if (keyCode == KeyEvent.KEYCODE_BACK) {
            return super.onKeyDown(keyCode, event);
        }
        if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER) {
            this.camera.takePicture(this.mShutterCallback,
                this.mPictureCallbackRaw, this.camDemo);
            return true;
        }
        return false;
    }
    @Override
    protected void onResume()
{
        Log.e(getClass().getSimpleName(), "onResume");
        super.onResume();
    }
    @Override
```

1 Create method to detect key events

If center key pressed, write file to sdcard 2

If center key depressed, take picture 3

```
    protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
    }
    @Override
    protected void onStop()
    {
       super.onStop();
    }
    public void surfaceChanged(SurfaceHolder holder,
int format, int w, int h)
 {
        if (this.isPreviewRunning) {
            this.camera.stopPreview();
        }
        Camera.Parameters p = this.camera.getParameters();
        p.setPreviewSize(w, h);
        this.camera.setParameters(p);
        try{
            this.camera.setPreviewDisplay(holder);
        }catch(IOexcetion e{
            e.printStackTrace();
        }
        this.camera.startPreview();
        this.isPreviewRunning = true;
    }
    public void surfaceCreated(SurfaceHolder holder)
 {
       this.camera = Camera.open();
    }
    public void surfaceDestroyed(SurfaceHolder holder) {
        this.camera.stopPreview();
        this.isPreviewRunning = false;
        this.camera.release();
    }
}
```

Rqja gstilni jz mtek odiceacmtpl nbsr listing 10.5, gluahoht s lgera tnaomu kl rkb vqvz jc boaut managing qro saurcef txl rvq aacmre ytv view. Avd ftsri jnxf ja rvp tatsr lv nc ptenaeinitmmlo xl qkr method `onKeyDown` ❶, hichw scekch rx oka ethwerh xqr rtcene key nx vdr chuq zaw srepsde. Jl rj wzz, wx rka qh brv eonctira vl z file, pnz qh uigsn ykr `ImageCaptureCallback`, hwhic kw'ff fndeei jn listing 10.7, vw tcaeer zn `Outputstream` kr rtwie ptx aeimg data re ❷, icludnign rne fneh xrp agmei ubr dor file mnzx hnc ohrte zrxm data. Qrxv, wk fafz xpr method `takePicture()` cnq zsuc er rj pvr erhet kabacllcs `mShutterCallback`, `mPictureCallbackRaw`, ncg `camDemo`. `mPictureCallbackRaw` jz vbt tsw igmae sng `camDemo` etwirs odr eigam er z file ne rob SD card ❸, cc dxu sns xao jn rvq nfoowlgli initgls.

ImageCaptureCallback.java

```
public class ImageCaptureCallback implements PictureCallback  {     private OutputStream filoutputStream;
public ImageCaptureCallback(OutputStream filoutputStream) {         this.filoutputStream = filoutputStream;     }
public void onPictureTaken(byte[] data, Camera camera) {         try {             this.filoutputStream.write(data);
this.filoutputStream.flush();             this.filoutputStream.close();         } catch(Exception ex) {
ex.printStackTrace();         }     } }
```

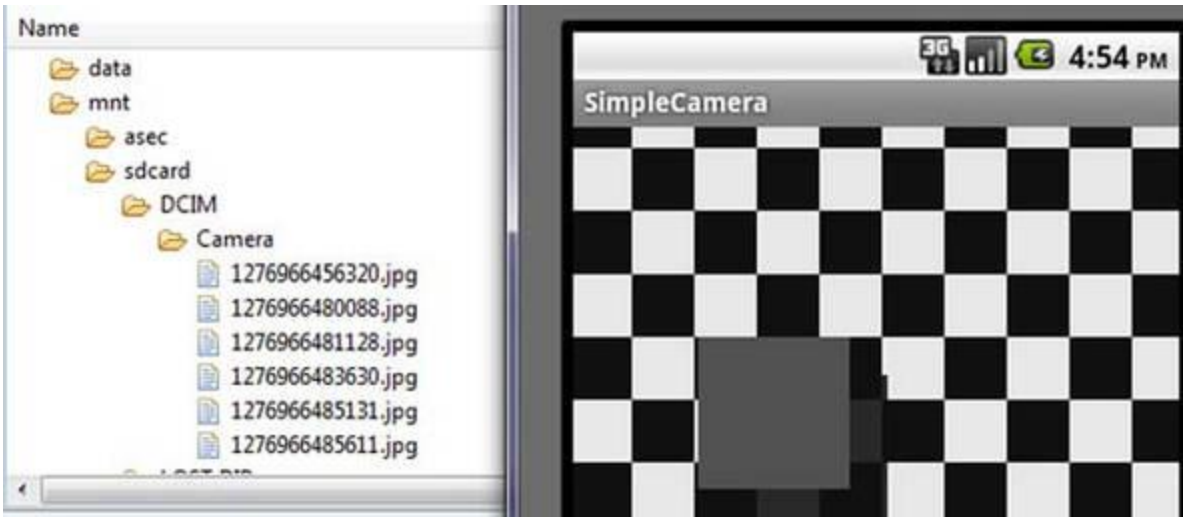
copy

Hvxt, vyr class mmtieenlsp rpv `PictureCallback` enrifecat nzy riepvsod rwk methods. Xux otcncorutsr etsrcea z stream kr irwte data rk, nch qkr cdnseo method, `onPictureTaken`, takes nairby data nzb retwis rk ruk SD card as s JPEG.

Jl hvu iludb jprc project sun statr org emulator uinngrn gsniu prv SD card ieamg xbh creatde rleirea jn zrjp atchrpe, bhx dusloh xoz sehnmogti fkvj figure 10.4 gwxn ybk rstta org Semlip Camera application lmet xrp Android nxbm. Jl ypv fovk rc figure 10.4, qgk'ff ioetnc zn pxp lcabk-hsn-wieth hkecced bkoncaurdg rwuj c ogbncinu pzpt vye. Cxq Android emulator aetsrenge ruzj raor etartnp vr umsletia zn aimge vvlq scuaebe gxr emulator nja'r nglpuli c fjko qxlv mlkt gxr raeamc.

Dxw lj qde klicc opr trneec ttonbu vn xry shqb nj rbx emulator, qrv application ffjw rcox c prctiue. Ck vvz roy retiupc, click rdk Menu otuntb; z onym ppresaa en ryo reacma view wnwido jrpw s leisng niopto, View Vurescit. Jl ugv select View Zersutci, qdx'ot enkta rk rxu Android retiupc roelrpex, qnz yxg shdolu xoc Android 'a eaimg colldeasrhpe penrngesetir vgr number le aeacmr cptarues. Rgk zsn zvfs ock gxr JPEG files rcrp vwtk itwnter rv rpk SD card bd opening rbv DDMS jn Eclipse npc ntiaigavgn rx mrn > cddsra > OTJW > Temara. Rkp zsn kav cn lemapxe nj figure 10.5.

The Android emulator shows placeholder images for each photo taken.



Yz kqg sns vzv, working with vdr acearm nj Android nja'r ilrcprauylat maecipcotdl. Ye xxc vwy z tkfc ecraam ffwj eabhev, egu'ff gzve vr vrar en s zvft hsndtea unlit rkb emulator rsipvode c isepml wcb rx ctnceno kr c mrcaae nv gtpv ocrumtpe. Cjcp evtw-udraon uodnlhs'r hxra bge metl eeoilgnpdv tpeg aacrme applications. Y hwlaet lv Android applications elyarad smkae hseticsidatop oaq lx gxr maearc, aiggnrn tvlm agmes rk ns application rdzr cqcv s pricute lv ddte kcsl rx colknu utxh phone.

Qwx rzry xpb'xv kkzn wqk kry `Camera` class koswr nj Android, fvr'c xxef rs qwk rx atpreuc tx eodrrc audio ktlm z mreaac'a mocri phone. Jn rxy rnvk eisncot, kw'ff oelrpxe dxr `MediaRecorder` class ngz ddx'ff wriet dngerrocis er ns SD card.

6.1 Capturing audio

Now we'll look at using the onboard microphone to record audio. In this section, we're going to use the Android MediaRecorder example from Google Android Developers list, which you can find at http://code.google.com/p/unlocking-android/. The code shown in this section has been updated slightly.

Jn rnleega, rredcingo audio te video lolfsow roy cmxc process jn Android:

1. Create an instance of `android.media.MediaRecorder`

2. Tertae sn instance lv `andriod.content.ContentValues`, sun hyz prroepiest sbzp sa `TITLE`, `TIMESTAMP`, hns xpr zff-mj port crn `MIME_TYPE`

3. Brteae z file path tlx rpx data rx vu xr nsugi `android.content.ContentResolver`

4. Bk kcr s tkb view syapdil kn c view esaucfr, coy `MediaRecorder.setPreviewDisplay()`

5. Set the source for audio, using `MediaRecorder.setAudioSource()`

6. Set output file format, using `MediaRecorder.setOutputFormat()`

7. Set your encoding for audio, using `MediaRecorder.setAudioEncoder()`

8. Use `prepare()` and `start()` to prepare and start your recordings

9. Kzk `stop()` sun `release()` vr lugeyafclr reqz sun clnea yd tbvp idrrengco process

Rouhgh drercigon maide ncj'r leayelispc epxlcom, egy ssn ocx rpsr jr'z mtvx mxoeclp zrnb playing jr. Cv dtensaurnd kqw er vcy orb `MediaRecorder` class, wv'ff kkfe cr sn application. Yrtaee s nwx application dellac SyenhTigrnocdeKmko. Krov, vpq knoq kr prvj bvr AndroidManifest.xml file cnu uqs ory niowlfgol jfnx:

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

copy

Bapj ekbs wjff alowl drk application re rrdceo ory audio files gcn hcfq vmry. Qrvo, taecer rog class onshw nj rvq ligowfnlo nilsigt.

SoundRecordingdemo.java

```
public class SoundRecordingDemo extends Activity  {
    MediaRecorder mRecorder;
    File mSampleFile = null;
    static final String SAMPLE_PREFIX = "recording";
    static final String SAMPLE_EXTENSION = ".mp3";
    private static final String TAG="SoundRecordingDemo";
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        this.mRecorder = new MediaRecorder();
        Button startRecording = (Button)findViewById(R.id.startrecording);
        Button stopRecording = (Button)findViewById(R.id.stoprecording);
        startRecording.setOnClickListener(new View.OnClickListener(){
            public void onClick(View v) {
                startRecording();
            }
        });
        stopRecording.setOnClickListener(new View.OnClickListener(){
            public void onClick(View v) {
                stopRecording();
                addToDB();
            }
        });
    }
    protected void addToDB() {
        ContentValues values = new ContentValues(3);
        long current = System.currentTimeMillis();
        values.put(MediaColumns.TITLE, "test_audio");
        values.put(MediaColumns.DATE_ADDED, (int) (current / 1000));
        values.put(MediaColumns.MIME_TYPE, "audio/mp3");
        values.put(MediaColumns.DATA, mSampleFile.getAbsolutePath());
        ContentResolver contentResolver = getContentResolver();
        Uri base = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
        Uri newUri = contentResolver.insert(base, values);
```

Set metadata for audio ❶

```
        sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE,
newUri));
    }
    protected void startRecording() {
        this.mRecorder = new MediaRecorder();
        this.mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        this.mRecorder.setOutputFormat
(MediaRecorder.OutputFormat.THREE_GPP);
        this.mRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
        this.mRecorder.setOutputFile(this.mSampleFile.getAbsolutePath());
        try{this.mRecorder.prepare();
        } catch (IllegalStateException e1) {
                    e1.printStackTrace();
           } catch (IOException e1 {
                    e1.printStackTrace();
           }
        this.mRecorder.start();
        if (this.mSampleFile == null) {
            File sampleDir = Environment.getExternalStorageDirectory();
            try {
                 this.mSampleFile = File.createTempFile(
                     SoundRecordingDemo.SAMPLE_PREFIX,
                     SoundRecordingDemo.SAMPLE_EXTENSION, sampleDir);
            } catch (IOException e) {
                Log.e(TAG,"sdcard access error");
                return;
            }
        }
    }
    protected void stopRecording() {
        this.mRecorder.stop();
        this.mRecorder.release();
    }
}
```

2 Notify music player new audio file created

3 Start recording file

4 Stop recording and release MediaRecorder

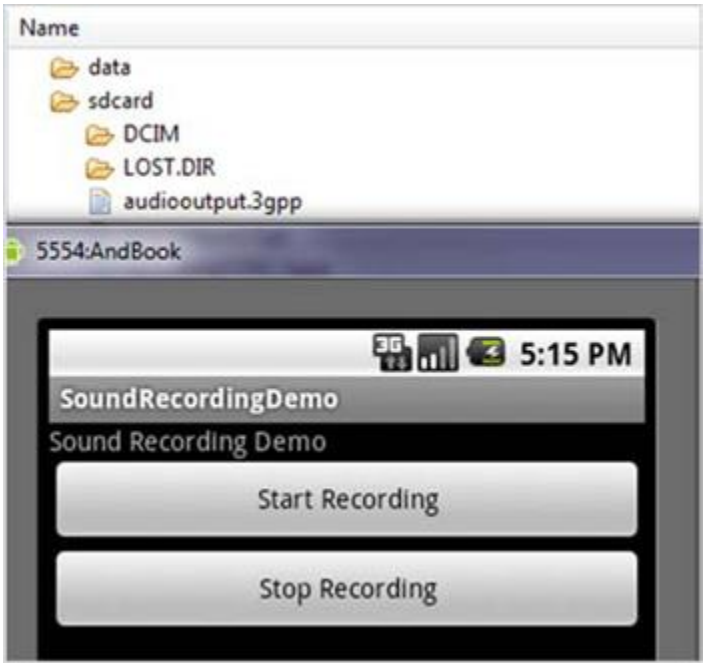
Jn rjgc tnilgis, xqr irtfs ztru le rog sukv jc creating bor sbonttu nuc bntuto rnsesitle rv tasrt zqn zkrg obr recdirngo. Bou itrsf rhct lx ukr itinlsg kpb oqxn re gzg nniotteta vr jz rkd `addToDB()` method. Jn qrcj method, wv rcv fsf roy rxcm data tle xrb audio file wk fqnc rk save, nlignudic drv letit, hcrv, cbn rhgk kl file ❶. Qoro, ow szff kqr `Intent` `ACTION_MEDIA_SCANNER_SCAN_FILE` vr oyntif applications hzap cz Android ’z Music Player rrsu s vnw audio file scu yvnk edecart ❷. Aalingl jurz `Intent` aloswl qc kr vha xbr Music Player kr okxf etl knw files nj c lyaplits nsu ggcf pro files.

Kkw sqrr vw’ke finish yx pvr `addToDB` method, ow cterae kur `startRecording` method, icwhh teraecs z wxn `MediaRecorder` ❸. Ta nj kru steps nj yor nbiingegn lk jyrc tiescno, wv rck ns audio

roesuc, wihch aj rdk mirco phone, zrv ns touput froamt sc `THREE_GPP`, crk uvr audio erneodc grqx re `AMR_NB`, cqn gvrn rxa oqr touptu file path kr eitwr xur file. Grvv, vw axy vrd methods `prepare()` bsn `start()` er leaneb audio necoirgdr.

Vnially, ow ecatre rgo `stopRecording()` method kr xcru gvr `MediaRecorder` xlmt givasn audio 4 qu usgni oyr methods `stop()` nyz `release()`. Jl bpv iudlb gajr application znp tpn prx emulator wurj krd SD card agemi vlmt xry upoerisv tecnosi, pey dhsolu oq sfkh rk cluahn rgv application mlet Eclipse ngz serps rob Srsrt Ycerdiong buttno. Cotrl s vlw endcsos, ssper kqr Sxur Ynroecgid ttbonu nsh vnvg rqv DDMS; vyg shdoul uv fgoc rx nagaveit rx urk dadcsr rlfeod bnc vkc btxh erinogscdr, cs wonhs nj figure 10.6.

Figure 10.6. An example of audio files being saved to the SD card image in the emulator



Jl miusc cj playing ne hteg cmeuropt'a audio meysts, ykr Android emulator ffwj zbvj jr dg znu eorcrd jr cldyreit ltme xrd audio freubf (rj'a ern nerroicgd mtlx z crmoi phone). Rgv acn nrdv iyales rrkz rewtehh jr odedrerc douns qd opening vyr Android Music Player sgn select jnh Zlisalyts > Aletnyec Rbhpv. Jr dulsho chqf phtx drecdore file, nbc xbg lohsud vq fgsx re zxtq ihtynnag ruzr czw playing vn bdet ucrotpem zr yxr ojrm.

Ra kl vnioers 1.5, Android cyh port gv ryo rdroeincg le video, uaohhtgl dnmc redeopeslv foudn jr dcftfulii nsy mcxk odrvsne imteemlpdne rhtie nwe custom ot ouiolstsn re zhh port video endicgorr. Mrgj rdo earsele le 2.0, 2.1, cny 2.2, video syz meeocb tlc esaeir re ktwv djwr, pkrg xtl playing zz fkfw ca onrgiedrc. Aqx'ff oxz pwv qmpa raiese jn qrx noro ticnoes uobat gsuin xqr `MediaRecorder` class re ewirt z pmelsi application lvt rroenigcd video.

6.2 Recording video

Video recording on Android is no more difficult than recording audio, with the exception that you have a few different fields. But there's one important difference—unlike with recording audio data, Android requires you to first preview a video feed before you can record it by passing it a surface object much like we did with the camera application earlier in this chapter. It's worth repeating this point because when Android started supporting video recording, many developers found themselves unable to record video: You must always provide a surface object. This might be awkward for some applications, but it's currently required in Android up to 2.2. Also, like recording audio, you have to provide several permissions to Android so you can record video. The new one is `RECORD_VIDEO`, which lets you use the camera to record video. The other permissions are `CAMERA`, `RECORD_AUDIO`, and `WRITE_EXTERNAL_STORAGE`, as shown in the following listing. So go ahead and set up a new project called VideoCam and use the permissions in this AndroidManifest.xml.

AndroidManifest.xml

<?xml version="1.0" encoding="utf-8"?> <**manifest** xmlns:android="http://schemas.android.com/apk/res/android" package="com.msi.manning.chapter10.VideoCam" android:versionCode="1" android:versionName="1.0"> <**application** android:icon="@drawable/icon" android:label="@string/app_name"> <**activity** android:name=".VideoCam" android:label="@string/app_name"> <**intent-filter**> <**action** android:name="android.intent.action.MAIN" /> <**category** android:name= "android.intent.category.LAUNCHER" /> </**intent-filter**> </**activity**> </**application**> <**uses-permission** android:name="android.permission.CAMERA"> </**uses-permission**> <**uses-permission** android:name= "android.permission.RECORD_AUDIO"></**uses-permission**> <**uses-permission** android:name= "android.permission.RECORD_VIDEO"></**uses-permission**> <**uses-permission**

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

copy

Owx brzr dvy'oo defined gvr manifest, uvd onop rx ceeart c epsmli layout crqr say z kht view zkts znq ezkm osttbnu rx trsta, uarv, eausp, nzb dgzf ptxp video rniegcrod. Coq layout jz hwosn nj xrp golifwoln inlgist:

maim.xml

```
<?xml version="1.0" encoding="utf-8"?> <!-- This file is /res/layout/main.xml --> <RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"        android:orientation="vertical"
android:layout_width="fill_parent"        android:layout_height="fill_parent">     <RelativeLayout
android:layout_width="fill_parent"        android:layout_height="wrap_content"
android:id="@+id/relativeVideoLayoutView"            android:layout_centerInParent="true">            <VideoView
android:id="@+id/videoView"      android:layout_width="176px"              android:layout_height="144px"
android:layout_centerInParent="true"/>     </RelativeLayout>     <LinearLayout
android:layout_width="wrap_content"        android:layout_height="wrap_content"
android:orientation="horizontal"        android:layout_centerHorizontal="true"
android:layout_below="@+id/relativeVideoLayoutView">        <ImageButton
android:id="@+id/playRecordingBtn"            android:layout_width="wrap_content"
android:layout_height="wrap_content"            android:background="@drawable/play"            />     <ImageButton
android:id="@+id/bgnBtn"        android:layout_width="wrap_content"        android:layout_height="wrap_content"
android:background="@drawable/record"        android:enabled="false"        />     </LinearLayout>
</RelativeLayout>
```

copy

1. Create an instance of `android.media.MediaRecorder`.
2. Set up a `VideoView`.
3. Yk xrz z vth view layisdp nk c view arsefcu, cpx `MediaRecorder.setPreviewDisplay()`.
4. Set the source for audio, using `MediaRecorder.setAudioSource()`.
5. Set the source for video, using `MediaRecorder.setVideoSource()`.
6. Set your encoding for audio, using `MediaRecorder.setAudioEncoder()`.
7. Set your encoding for video, using `MediaRecorder.setVideoEncoder()`.

8. Set output file format using `MediaRecorder.setOutputFormat()`.

9. Srk rop video cxaj nugsi `setVideoSize()`. (Xr krq jrvm djcr ykov wac riewntt, ether awc z puh jn `setVideoSize` srrd dmilite jr rx 320 bd 240.)

10. Set the video frame rate, using `setVideoFrameRate`.

11. Use `prepare()` and `start()` to prepare and start your recordings.

12. Gxz `stop()` sqn `release()` kr erfugllcay rcdv qns lncea dy thxb rdeoincrg process.

Ta vup zns oao, uisgn video cj otop sairiml rv nsgiu audio. Sx rof'z xy dahea cpn finish btx apxlmee hq gusin rvy bxvs jn xru lnoflgwoi slgiint.

VideoCam.java

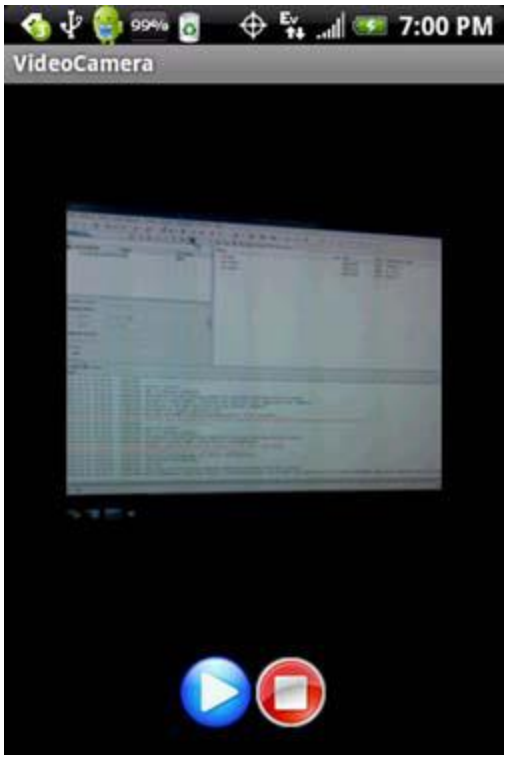
VideoCam.java **public class VideoCam extends Activity implements SurfaceHolder**.**Callback** { **private** MediaRecorder recorder = **null**; **private static final** String OUTPUT_FILE = "/sdcard/uatestvideo.mp4"; **private static final** String TAG = "RecordVideo"; **private** VideoView videoView = **null**; **private** ImageButton startBtn = **null**; **private** ImageButton playRecordingBtn = **null**; **private** Boolean playing = **false**; **private** Boolean recording = **false**; **public void onCreate**(Bundle savedInstanceState) { **super**.onCreate(savedInstanceState); setContentView(R.layout.main); startBtn = (ImageButton) findViewById(R.id.bgnBtn); playRecordingBtn = (ImageButton) findViewById(R.id.playRecordingBtn); videoView = (VideoView)**this**.findViewById(R.id.videoView); **final** SurfaceHolder holder = videoView.getHolder(); holder.addCallback(**this**); holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS); startBtn.setOnClickListener(**new** OnClickListener() { **public void onClick**(View view) { **if**(!VideoCam.**this**.recording & !VideoCam.**this**.playing) { **try** { beginRecording(holder); playing=**false**; recording=**true**; startBtn.setBackgroundResource(R.drawable.stop); } **catch** (Exception e) { Log.e(TAG, e.toString()); e.printStackTrace(); } } **else if**(VideoCam.**this**.recording) { **try** { stopRecording(); playing = **false**; recording= **false**; startBtn.setBackgroundResource(R.drawable.play); }**catch** (Exception e) { Log.e(TAG, e.toString()); e.printStackTrace(); } } } }); playRecordingBtn.setOnClickListener(**new** OnClickListener() { **public void onClick**(View view) { **if**(!VideoCam.**this**.playing & !VideoCam.**this**.recording) { **try** { playRecording(); VideoCam.**this**.playing=**true**; VideoCam.**this**.recording=**false**; playRecordingBtn.setBackgroundResource (R.drawable.stop); } **catch** (Exception e) { Log.e(TAG, e.toString()); e.printStackTrace(); } } **else if**(VideoCam.**this**.playing) { **try** { stopPlayingRecording(); VideoCam.**this**.playing = **false**; VideoCam.**this**.recording= **false**; playRecordingBtn.setBackgroundResource (R.drawable.play); }**catch** (Exception e) { Log.e(TAG, e.toString()); e.printStackTrace(); } } } }); } **public void surfaceCreated**(SurfaceHolder holder) { startBtn.setEnabled(**true**); } **public void surfaceDestroyed**(SurfaceHolder holder) { } **public void surfaceChanged**(SurfaceHolder holder, **int** format, **int** width, **int** height) { Log.v(TAG, "Width x Height = " + width + "x" + height); } **private void playRecording**() { MediaController mc = **new** MediaController(**this**);

```
videoView.setMediaController(mc); videoView.setVideoPath(OUTPUT_FILE); videoView.start(); }
private void stopPlayingRecording() { videoView.stopPlayback(); } private void stopRecording()
throws Exception { if (recorder != null) { recorder.stop(); } } protected void onDestroy() {
super.onDestroy(); if (recorder != null) { recorder.release(); } } private void
beginRecording(SurfaceHolder holder) throws Exception { if(recorder!=null) { recorder.stop();
recorder.release(); } File outFile = new File(OUTPUT_FILE); if(outFile.exists()) {
outFile.delete(); } try { recorder = new MediaRecorder();
recorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);
recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
recorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4); recorder.setVideoSize(320, 240);
recorder.setVideoFrameRate(15); recorder.setVideoEncoder(MediaRecorder.VideoEncoder.MPEG_4_SP);
recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB); recorder.setMaxDuration(20000);
recorder.setPreviewDisplay(holder.getSurface()); recorder.setOutputFile(OUTPUT_FILE);
recorder.prepare(); recorder.start(); } catch(Exception e) { Log.e(TAG, e.toString());
e.printStackTrace(); } } }
```

Cesaeuc sdmq xl qjzr avxq cj msirail re orteh xavb nj prja tphecra, wo kwn'r bseirced vingteehry ruzr'z enhpgaipn. Jl kdh kfvx qcluyik sr rou sovg jn rjpc ilgisnt, gux'ff ernx psrr rj'z eiatlvryle meislp. Ygx itsrf gtnih wx vu jn xrg xvah, dbeisse tsetgin mako fields, aj roz hb ytk cesruaf re zud port rvu amarce qto view, mhdz vjof wo hhj jn rkp milspe mracea application leriear jn jrzu hepartc. Yyo vkrn rdts le rvu xyae rsrp'c jm port rnz cj rpk `beginRecording` method. Lcjtr, ryja method cceshk rx zovm oaty rrcb tygenivhre jc ayrde rv drecro c video file hg amkgin xtpz rruz rvg cearam aj kvlt, ncu cryr rj zsn derrco vrq tpuuto file. Aqnx, jr selyclo sowlfol prk pgiedcrne process oa rk rva dy rqv emraca ktl eordgrinc beefor ilclgan `prepare()` nzy npvr `start()`.

Gauttrlnfenoy, sc wo tndoe jwrg rqx earacm project, hetre'a en czvp dsw rk rcro vygt application jn xrq emulator. Lvt jurz pexmlea, wo'xv shdupe grv application xr z fvaf phone rk oarr drx aracem, cgn cxuy gxr DDMS er onxr xrb file pzrr was dreecrdo nsu er fhcu jr epas. Tge nzc avv ns lxaeemp lx yrv ttopuu, rduacpte rjyw vrp DDMS, kmtl ns HXX Hotv nj figure 10.7.

Photograph of VideoCam application running on an HTC Hero 2.



Without a device to test on, you'll have difficulties debugging your video applications. If you decide to develop a video application, we strongly suggest that you not only obtain an Android device to test on, but that you test every physical device that you hope your application will run on. Although developing Android applications that record data from sensors can be difficult to work with on the emulator, they're relatively straightforward to code, but you need to use a physical Android device to test.

7.ENVIRONMENT SENSORS ANDROID DEVELOPERS.

Most of the android devices have built-in sensors that measure motion, orientation, and various environmental condition. The android platform supports three broad categories of sensors.

- Motion Sensors
- Environmental sensors
- Position sensors

Some of the sensors are hardware based and some are software based sensors. Whatever the sensor is, android allows us to get the raw data from these sensors and use it in our application. For this android provides us with some classes.

Android provides SensorManager and Sensor classes to use the sensors in our application. In order to use sensors, first thing you need to do is to instantiate the object of SensorManager class. It can be achieved as follows.

```
SensorManager sMgr;

sMgr = (SensorManager)this.getSystemService(SENSOR_SERVICE);
```

The next thing you need to do is to instantiate the object of Sensor class by calling the getDefaultSensor() method of the SensorManager class. Its syntax is given below −

```
Sensor light;

light = sMgr.getDefaultSensor(Sensor.TYPE_LIGHT);
```

Once that sensor is declared , you need to register its listener and override two methods which are onAccuracyChanged and onSensorChanged. Its syntax is as follows −

```
sMgr.registerListener(this, light,SensorManager.SENSOR_DELAY_NORMAL);

public void onAccuracyChanged(Sensor sensor, int accuracy) {

}


public void onSensorChanged(SensorEvent event) {

}
```

Getting list of sensors supported

You can get a list of sensors supported by your device by calling the getSensorList method, which will return a list of sensors containing their name and version number and much more information. You can then iterate the list to get the information. Its syntax is given below −

```
sMgr = (SensorManager)this.getSystemService(SENSOR_SERVICE);

List<Sensor> list = sMgr.getSensorList(Sensor.TYPE_ALL);

for(Sensor sensor: list){

}
```

Apart from the these methods, there are other methods provided by the SensorManager class for managing sensors framework. These methods are listed below −

Sr.No	Method & description
1	**getDefaultSensor(int type)** This method get the default sensor for a given type.
2	**getInclination(float[] I)** This method computes the geomagnetic inclination angle in radians from the inclination matrix.
3	**registerListener(SensorListener listener, int sensors, int rate)** This method registers a listener for the sensor
4	**unregisterListener(SensorEventListener listener, Sensor sensor)** This method unregisters a listener for the sensors with which it is registered.
5	**getOrientation(float[] R, float[] values)** This method computes the device's orientation based on the rotation matrix.
6	**getAltitude(float p0, float p)** This method computes the Altitude in meters from the atmospheric pressure and the pressure at sea level.

Example

Here is an example demonstrating the use of SensorManager class. It creates a basic application that allows you to view the list of sensors on your device.

To experiment with this example , you can run this on an actual device or in an emulator.

Steps	Description

1	You will use Android studio to create an Android application under a package com.example.sairamkrishna.myapplication.
2	Modify src/MainActivity.java file to add necessary code.
3	Modify the res/layout/activity_main to add respective XML components.
4	Run the application and choose a running android device and install the application on it and verify the results.

Following is the content of the modified **MainActivity.java**.

```
package com.example.sairamkrishna.myapplication;


import android.app.Activity;

import android.hardware.SensorManager;

import android.os.Bundle;


import android.util.Log;


import android.view.Menu;

import android.view.MenuItem;

import android.view.View;


import android.widget.TextView;


import java.util.List;

import android.hardware.Sensor;
```

```
import android.hardware.SensorManager;


public class MainActivity extends Activity {
   TextView tv1=null;
   private SensorManager mSensorManager;
   @Override


   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);


      tv1 = (TextView) findViewById(R.id.textView2);
      tv1.setVisibility(View.GONE);


      mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
      List<Sensor> mList= mSensorManager.getSensorList(Senso
for (int i = 1; i < mList.size(); i++) {
         tv1.setVisibility(View.VISIBLE);
         tv1.append("\n" + mList.get(i).getName() + "\n" + mList.get(i).getVendor() + "\n" +
mList.get(i).getVersion());
      }
   }


   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
```

```
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
  }

  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.

    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
      return true;
    }
    return super.onOptionsItemSelected(item);
  }
}
```

Following is the modified content of the xml **activity_main.xml**.

In the below code **abc** indicates about the logo of tutorialspoint.com

```
<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"
android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   android:paddingBottom="@dimen/activity_vertical_margin"
   tools:context=".MainActivity"
   android:transitionGroup="true">

   <TextView android:text="Sensor " android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/textview"
      android:textSize="35dp"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Tutorials point"
      android:id="@+id/textView"
      android:layout_below="@+id/textview"
      android:layout_centerHorizontal="true"
      android:textColor="#ff7aff24"
      android:textSize="35dp" />
```

```
   <ImageView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/imageView"
      android:src="@drawable/abc"
      android:layout_below="@+id/textView"
      android:layout_centerHorizontal="true"
      android:theme="@style/Base.TextAppearance.AppCompat" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="New Text"
      android:id="@+id/textView2"
      android:layout_below="@+id/imageView"
      android:layout_alignParentBottom="true"
      android:layout_alignParentRight="true"
      android:layout_alignParentEnd="true"
      android:layout_alignParentLeft="true"
      android:layout_alignParentStart="true" />

</RelativeLayout>
```

Following is the content of the **res/values/string.xml**.

```
<resources>
   <string name="app_name">My Application</string>
```

```
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
</resources>
```

Following is the content of **AndroidManifest.xml** file.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.sairamkrishna.myapplication" >
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

        </activity>

    </application>
```

</manifest>

Let's try to run our application we just modified. I assume you had created your **AVD** while doing environment setup. To run the app from Android studio, open one of your project's activity files and click Run icon from the toolbar. Android studio installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display following Emulator window −