توسعه برنامه های موبایل

جلسه پنجم مجازی

بخش اول

سحر صادقی

آموزش کار با صفحه لمسی در اندروید با چند مثال

Gesture در واقع مربوط به حرکات لمس صفحه و کوچک و بزرگ کردن تصاویر است ؛ اندروید انواع خاصی از لمس صفحه را ارائه می دهد که عبارتند از :

- **فشار دادن** (pinch)
- **ضربه ی دوتایی** (double tap)
- **فهرست ها** (scrolls)
- **فشار طولانی مدت** (long presses)
- flinch

اندروید کلاس GestureDetector **را ارائه می دهد تا** event **های** Gesture **را دریافت کرده و به ما بگوید که آیا این رویداد ها با (** Gesture **رفتار) منطبق هستند یا نه. برای استفاده از آن لازم است یک آبجکت از** GestureDetector **ایجاد کنید و سپس یک کلاس دیگر با** GestureDetector.SimpleOnGestureListener **گسترش دهید تا مانند یک شنونده عمل کنید و بعضی از متدها را** override **کنید.**

سینتکس آن را می توانید در زیر ببینید:

```
GestureDetector myG; myG = new GestureDetector(this,new Gesture());     class Gesture extends GestureDetector.SimpleOnGestureListener{    public boolean onSingleTapUp(MotionEvent ev) {    }        public void onLongPress(MotionEvent ev) {    }    public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX,    float distanceY) {    }        public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,    float velocityY) {    } }
```

رفتار: pinch

اندروید برای اداره Gesture (pinch **رفتار ،) کلاس** ScaleGestureDetector **را ایجاد می کند. برای استفاده از این کلاس لازم است که یک آبجکت از این کلاس را نمونه گذاری کنید که سینتکس آن را در زیر می بینید:**

```
ScaleGestureDetector SGD; SGD = new ScaleGestureDetector(this,new ScaleListener()
);
```

اولین پارامتر کانتکست (context) **و دومین پارامتر شنونده ی رویداد** (event listener) **می باشد. ما باید رویداد شنونده را تعریف کرده و از تابع** OnTouchEvent **برای به کار انداختن آن استفاده کنیم. سینتکس آن را در زیر می بینید:**

```
public boolean onTouchEvent(MotionEvent ev) {    SGD.onTouchEvent(ev);    return
true; }  private class ScaleListener extends ScaleGestureDetector.SimpleOnScaleGes
tureListener {    @Override    public boolean onScale(ScaleGestureDetector detecto
r) {       float scale = detector.getScaleFactor();       return true;    } }
```

علاوه بر pinch **، متد های دیگری هم وجود دارند که در مورد رویداد های لمس** (touch event) **بیشتر توضیح می دهند.**

لیست آن ها در زیر ارائه شده اند:

- getEventTime(): **این متد زمان پردازش شدن ، زمان رویداد را به دست می آورد.**
- getFocusX(): **این متد مختصات** X **را از نقطه ی کانونی آخرین حرکت** (gesture) **به دست می آورد.**
- getFocusY(): **این متد مختصات** Y **را از نقطه ی کانونی آخرین حرکت** (gesture) **به دست می آورد.**
- getTimeDelta(): **این متد تفاوت زمانی بین مقیاس رویداد قبلی و مقیاس رویداد کنونی به یک هزارم ثانیه** (milliseconds) **بر می گرداند.**
- isInProgress(): **این متد بولین ، اگر مقیاس یک حرکت** (gesture) **در حال پیشروی باشد ،** true **را باز می گردد.**
- onTouchEvent(MotionEvent event): **این متد** MotionEvents **را قبول کرده و اگر رویداد ها مناسب باشند ، می فرستد.**

مثال:

در اینجا مثالی را می بینید که استفاده از کلاس ScaleGestureDetector **را توضیح می دهد. این مثال یک اپلیکیشن پایه ایجاد می کند که به شما اجازه می دهد تا از طریق** pinch **چیزی را زوم کنید یا از حالت زوم خارج شوید.**

برای آزمایش با این مثال به یک دستگاه واقعی یا یک امولاتور با صفحه ی لمسی فعال نیاز خواهید داشت.

۱. **برای ایجاد یک اپلیکیشن اندروید از** Android studio **تحت پکیج** com.example.gestures **استفاده نمایید.**

۲. **فایل** src/MainActivity.java **را برای افزودن کد لازم تغییر دهید.**

۳. res/layout/activity_main **را برای افزودن کامپونت های** XML **مربوطه تغییر دهید.**

۴. **برنامه را اجرا کرده و یک دستگاه اجرایی اندروید انتخاب کنید و برنامه را روی آن نصب کنید و نتایج را بررسی کنید.**

محتوای فایل اکتیویتی اصلی : src/MainActivity.java

```
package com.example.sairamkrishna.myapplication; import android.app.Activity; im
port android.graphics.Matrix; import android.os.Bundle; import android.view.Motio
nEvent; import android.view.ScaleGestureDetector; import android.widget.ImageView;
public class MainActivity extends Activity {    private ImageView iv;    private M
atrix matrix = new Matrix();    private float scale = 1f;    private ScaleGestureD
etector SGD;    @Override    protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);       setContentView(R.layout.activity_main);
iv=(ImageView)findViewById(R.id.imageView);       SGD = new ScaleGestureDetector(t
his,new ScaleListener());    }    public boolean onTouchEvent(MotionEvent ev) {
SGD.onTouchEvent(ev);       return true;    }    private class ScaleListener exte
nds ScaleGestureDetector.       SimpleOnScaleGestureListener {             @Overr
ide       public boolean onScale(ScaleGestureDetector detector) {          scale *
= detector.getScaleFactor();          scale = Math.max(0.1f, Math.min(scale, 5.0f)
);          matrix.setScale(scale, scale);          iv.setImageMatrix(matrix);
return true;       }    } }
```

محتوای تغییر یافته فایل : res/layout/activity_main.xml

```
<RelativeLayout    xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"    android:layout_width="match_par
ent"    android:layout_height="match_parent"    android:paddingLeft="@dimen/activ
ity_horizontal_margin"    android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"    android:paddingBottom="@di
men/activity_vertical_margin"    tools:context=".MainActivity" >       <TextView
android:text="Gestures         Example" android:layout_width="wrap_content"
android:layout_height="wrap_content"       android:id="@+id/textview"       androi
d:textSize="35dp"       android:layout_alignParentTop="true"       android:layout_
centerHorizontal="true" />          <TextView       android:layout_width="wrap_co
```

```
ntent"        android:layout_height="wrap_content"        android:text="Tutorials po
int"        android:id="@+id/textView"        android:layout_below="@+id/textview"
android:layout_centerHorizontal="true"        android:textColor="#ff7aff24"        a
ndroid:textSize="35dp" />            <ImageView        android:layout_width="wrap_co
ntent"        android:layout_height="wrap_content"        android:id="@+id/imageView
"        android:src="@drawable/abc"        android:scaleType="matrix"        android
:layout_below="@+id/textView"        android:layout_alignParentLeft="true"        an
droid:layout_alignParentStart="true"        android:layout_alignParentBottom="true"
android:layout_alignParentRight="true"        android:layout_alignParentEnd="true"
/>        </RelativeLayout>
```

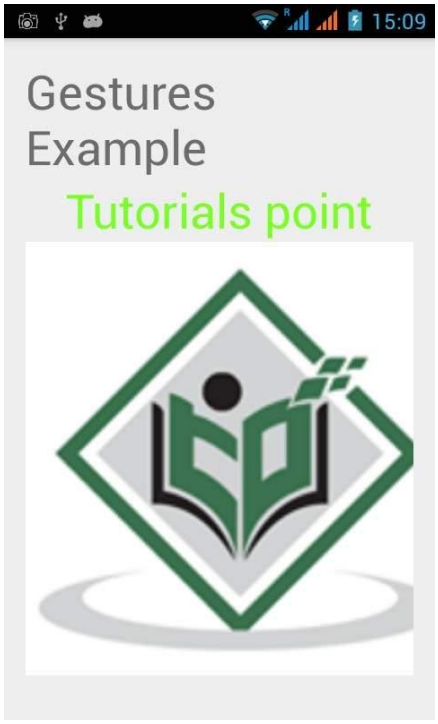
محتوای: res/values/string.xml

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

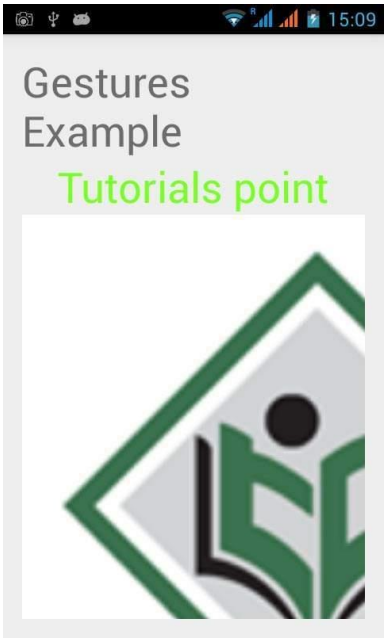
محتوای فایل: AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?> <manifest xmlns:android="http://schemas.an
droid.com/apk/res/android"    package="com.example.sairamkrishna.myapplication" >
<application        android:allowBackup="true"        android:icon="@drawable/ic_lau
ncher"        android:label="@string/app_name"        android:theme="@style/AppTheme
" >            <activity          android:name="com.example.sairamkrishna.myappl
icationMainActivity"          android:label="@string/app_name" >
<intent-filter>            <action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />          </intent-fil
ter>                </activity>        </application> </manifest>
```

اجازه بدهید اپلیکیشن را اجرا کنیم. فرض می کنیم که دستگاه موبایل اندروید خود را به کامپیوتر متصل کرده اید. برای اجرای برنامه از Android studio **، یکی از فایل های اکتیویتی پروژه را باز کرده و روی آیکون** Run **از نوار ابزار کلیک کنید.**



اکنون دو انگشت را روی صفحه ی اندروید قرار داده و آن ها را از هم دور کنید. خواهید دید که تصویر اندروید بزرگتر می شود. مانند تصویر زیر :



حال دوباره دو انگشت را روی صفحه قرار داده و آنها را به هم نزدیک کنید، خواهید دید که تصویر اندروید کوچکتر می شود، مانند تصویر زیر:

چینش ها و ابزار canvas

ساخت و پیاده سازی view های سفارشی و ترکیبی در اندروید

Viewهای پیش فرض محیط اندروید

چارچوب نرم افزاری (framework) **اندروید تعدادی** view **درون ساخته و پیش فرض ارائه می دهد که توسعه دهنده می تواند از آن ها به صورت آماده استفاده نماید. کلاسی که تمامی** view **ها (همان کنترل ها و المان های رابط کاربری) از آن مشتق و ارث بری می شوند،** Viewمی باشد.

Viewها **موظف هستند خود و تمامی المان های داخل خود** (view) **های فرزند که در** ViewGroup **با آن مواجه می شوید) را اندازه گیری، طرح بندی** (layout) **و ترسیم نمایند.**

Viewها همچنین می بایست اطلاعات مربوط به وضعیت (state) UI را ذخیره کرده و eventهایی که با تعامل کاربر با المان های رابط کاربری (لمس نمایشگر) فعال می شوند را مدیریت نمایند.

البته توسعه دهندگان این امکان را هم دارند که viewهای اختصاصی تعریف نموده و آن را در اپلیکیشن خود بکار ببرند.

به منظور تعریف viewهای سفارشی و دلخواه خود می توانید به یکی از روش های زیر اقدام نمایید:

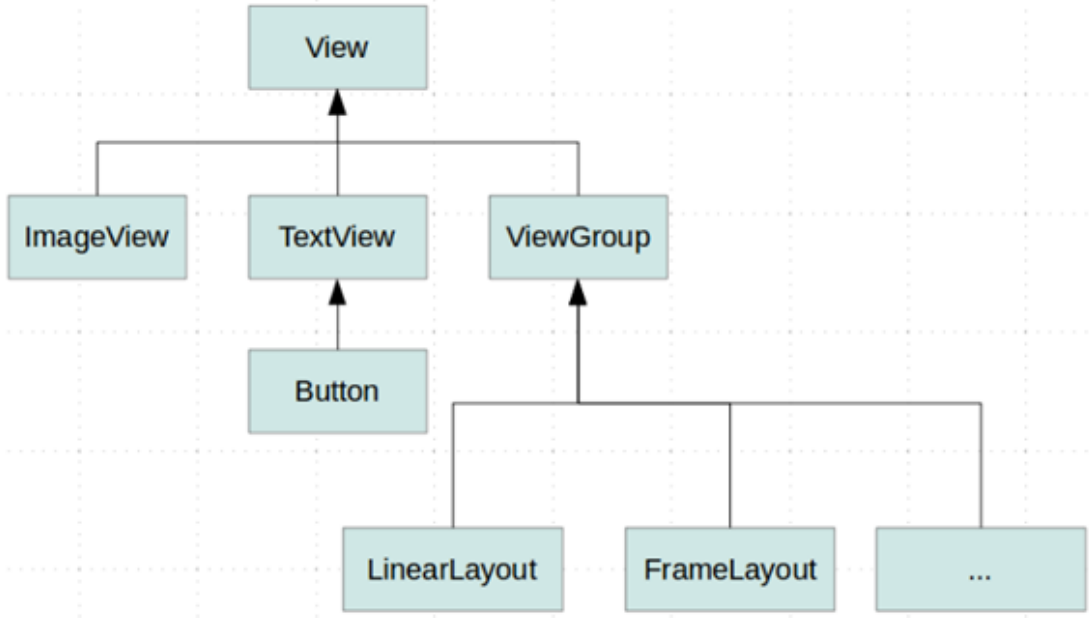
- Compound view تلفیق viewها با اتصال پیش فرض (default wiring)

- Custom view ساخت viewهای اختصاصی و دلخواه خود

با ارث بری از یک viewی آماده همچون Button

با ارث بری از کلاس View

تصویر زیر زنجیره ی ارث بری و سلسله مراتب viewهای پیش فرض اندروید را به نمایش می گذارد.



Viewهای سفارشی معمولا با هدف ارائه ی تجربه ی کاربری ویژه و دلخواه که با viewهای پیش فرض و آماده ی خود اندروید امکان پذیر نیست ساخته می شوند. بعلاوه پیاده سازی viewهای اختصاصی این امکان را برای توسعه دهنده فراهم می کند تا با ابتکار خود کارایی را افزایش دهد برای مثال در خصوص پیاده سازی layout اختصاصی، برنامه نویس می تواند layout manager را با توجه به نیاز خود تنظیم و بهینه نماید.

اندروید چگونه view hierarchy را ترسیم می نماید!

پس از اینکه انتخاب و تمرکز UI **بر روی یک** activity **متمرکز می شود، در همان لحظه بایستی** root node **)گره یا عنصر اصلی و آغازین) سلسه مراتب** layout **خود را در اختیار سیستم اندروید قرار دهد. پس از آن سیستم اندروید فرایند ترسیم را آغاز می نماید.**

روند ترسیم layout **دو مرحله را پشت سر می گذارد:**

- measuring pass (**مرحله ی سنجش و اندازه گیری) – توسط متد** `measure(int, int)` **پیاده سازی می شود. این مرحله به صورت پیمایش از بالای سلسله مراتب** view **تا پایین آن اتفاق می افتد. هر** view **اندازه های خود را ذخیره می کند.**
- layout pass (**مرحله ی تنظیم چیدمان و طرح بندی) – این مرحله توسط متد** layout(int, int, int, int) **پیاده سازی می شود. پیمایش مرحله ی جاری نیز از بالای سلسله مراتب** view **به پایین آن رخ می دهد. طی این مرحله هر** layout manager **مسئول چیدمان و موقعیت دهی فرزندان خود** view) **های داخل خود) می باشد. لازم به ذکر است که متد فوق برای تنظیم موقعیت** view **ها از اندازه های بدست آمده در مرحله ی اول استفاده می کند.**

نکته:

مرحله ی measure **اندازه گیری) و** layout **چیدمان) همیشه همزمان اتفاق می افتند.**

Layout manager **می تواند مرحله ی اندازه گیری را چندین بار اجرا نماید. برای مثال،** LinearLayout **از** attribute **ای به نام** weight **پشتیبانی می کند که فضای خالی باقی مانده را بین** view **ها پخش نموده و** RelativeLayout **تمامی المان یا** view **های فرزند خود را چندین بار اندازه گیری می کند تا** constraint **های تعیین شده در فایل** layout **همگی برآورده شوند.**

View **یا** activity **هر یک می توانند مرحله ی اندازه ی گیری و چیدمان را با فراخوانی متد** requestLayout() **راه اندازی نماید.**

پس از انجام محاسبات مربوط به اندازه گیری و تنظیم چیدمان المان ها، view **ها اقدام به ترسیم خود می نمایند. جهت راه اندازی این عملیات کافی است متد** invalidate() **از کلاس** View **فراخوانی شود.**

استفاده از view **های جدید در فایل های** layout

Viewهای اختصاصی و ترکیبی (پیچیده) می توانند در فایل های layout **تعریف و استفاده شوند. برای نیل به این هدف لازم است اسم** view **ها را به صورت تمام و کامل در فایل ذکر شده قید نمایید. برای مثال می بایست اسم کلاس و پکیج (پوشه ی اصلی پروژه) را ارائه کنید.**

?

```
1  <!--?xml version="1.0" encoding="utf-8" ?-->
2  <linearlayout xmlns:android="http://schemas.android.com/apk/res/android" android:lay
   android:orientation="vertical">
3  <button android:id="@+id/button1" android:layout_width="wrap_content" android:layout_
4  <de.vogella.android.ownview.mydrawview android:id="@+id/myDrawView1" android:layout_
5
6  </de.vogella.android.ownview.mydrawview></button></linearlayout>
```

توجه:

در صورت تمایل می توانید name space **خود را در فایل** layout **اعلان نمایید، مانند** name space **اندروید.**

Alternatively you can also declare you name space in the layout file, similar to the Android name space

تهیه ی تصویر آنی (screenshot) از view ها

تمامی کلاس های View **این قابلیت را دارند از وضعیت و ظاهر کنونی خود تصویر آنی تهیه نمایند.**

?

```
1# Build the Drawing Cache
2view.buildDrawingCache();
3# Create Bitmap
4Bitmap cache = view.getDrawingCache();
5# Save Bitmap
6saveBitmap(cache);
7view.destroyDrawingCache();
```

viewهای ترکیبی(Compound views)

Viewهای ترکیبی (یا کامپونت های ترکیبی) به View های گفته می شوند که از ترکیب چند view دیگر پدیده آمده باشد.

Compound viewها به شما این امکان را می دهند تاAPI ها و توابع اختصاصی خود را جهت بروز رسانی و کوئری گرفتن از اطلاعات مربوط به وضعیت view بکار ببرید.

برای این control یک فایل layout تعریف می کنید و آن را به compound view خود تخصیص می دهید. در پیاده سازیcompound view ، بایستی ارتباط متقابل view ها را تعریف کرده باشید. ابتدا یک فایل layoutتعریف می کنید که اعضا و توابع کلاس ViewGroup مربوطه را به ارث می برد. در این کلاس فایل layoutرا بارگذاری نموده (inflate) و منطق (کد) اتصال و ارتباط View را پیاده سازی می نمایید.

نکته:

برای افزایش کارایی و سرعت اجرا، بهتر است view سفارشی که از کلاس View ارث بری کرده ایجاد نمایید. از این طریق می توانید سلسله مراتب view خود و زیرشاخه های آن را به صورت خطی نمایش دهید (flatten view). چرا که در این حالت ترسیم view به پیمایش کمتری نیاز دارد و در صورتی که به شکل درستی پیاده سازی شود، بسیار سریع تر اجرا خواهد شد.

ساخت view های اختصاصی

پیاده سازی view های اختصاصی

با ارث بری از کلاس View یا یکی از کلاس های مشتق آن(subclass) ، شما می توانید view دلخواه خود را ایجاد نمایید.

برای ترسیم view می توانید متد onDraw() را بکار ببرید. در این متد یک آبجکت Canvas به عنوان ورودی دریافت می کنید. آبجکت نام برده به شما امکان می دهد تا عملیات ترسیم همچون کشیدن خط، دایره، درج متن یا bitmap را در سطح آن انجام دهید. در صورتی که view مجددا ترسیم گردد، شما می توانید متد invalidate()را فراخوانی کنید که خود سبب فراخوانی متد onDraw() این view می شود.

نکته:

در صورت تعریف view های اختصاصی، حتما کلاس ViewConfiguration را بررسی نمایید چرا که این کلاس تعدادی ثابت (constant) درون ساخته دارد که شما می توانید برای تعریف view ها مورد استفاده قرار دهید.

برای ترسیم Views **توسعه دهندگان اغلب از ۲** D Canvas API**استفاده می کنند.**

اندازه گیری view ها

Layout manager**متد** onMeasure() **از** view **مورد نظر را صدا می زند** .View **سپس پارامترهای** layout **را از** layout manager **دریافت می کند** .layout manager **وظیفه ی تعیین اندازه ی تمامی** view **های داخل خود را بر عهده دارد.**

View**می بایست متد** setMeasuredDimension (int, int) **را با نتیجه مورد نظر فراخوانی نماید.**

تعریف layout manager اختصاصی

برای تعریف layout manager **اختصاصی خود می توانید اعضا و توابع کلاس** ViewGroup **را به ارث ببرید. یا انجام کار فوق این امکان برای شما فراهم می شود تا** layout manger **های کارامد و بهینه تری را پیاده سازی نموده یا افکت های دیداری را خلق کنید که در محیط** (platform) **اندروید وجود ندارد.**

یک layout manager **سفارشی، به تبع پیاده سازی توابع** onMeasure() **و** onLayout() **را بازنویسی** (override)**نموده و عملیات محاسبه ی (اندازه ی) المان های فرزند خود را نیز به صورت اختصاصی انجام می دهد. برای مثال ممکن است استفاده از ویژگی سنگین و زمان بر** layout_weight **از کلاس** LinearLayout **را کنار بگذارد.**

نکته:

به منظور محاسبه ی اندازه ی المان محصور (فرزند) کافی است متد measureChildWithMargins() **از کلاس** ViewGroup **را فراخوانی نمایید.**

توصیه می شود تمامی پارامترهای اضافی layout **را در یک کلاس داخلی درون نمونه ی پیاده سازی شده ی خود از** ViewGroup **قرار دهید. برای مثال کلاس** LayoutParams **به عنوان یک ویژگی در کلاس** ViewGroup**ویژگی های** command ، layout**و** parameters **را درون خود کپسوله کرده است و کلاس** LinearLayout**نیز علاوه بر این ویژگی ها، ویژگی** layout_weight **را به این ویژگی اضافه نموده است.**

Life Cycle

eventها و توابع مربوط به مدیریت چرخه ی حیاتwindow

یک view **زمانی نمایش داده می شود که به** layout hierarchy **متصل باشد** layout hierarchy .**نیز خود به تبع به** window **وصل می باشد. هر** view **تعداد زیادی تابع** hook **مدیریت چرخه ی حیات دارد**.