توسعه برنامه های موبایل

جلسه چهارم مجازی

بخش سوم

سحر صادقی

صفحه لمسی یا Touchscreen چیست؟

صفحه لمسی یا تاچ اسکرین (Touchscreen) نوعی دستگاه ورودی و خروجی است که به کاربر اجازه می‌دهد از طریق لمس صفحه با کمک یک یا چند انگشت و یا با کمک یک استایلوس) نوعی ابزار شبیه قلم) با آنچه روی صفحه مشاهده می کند ارتباط برقرار کرده و یا آن را کنترل نماید.

هر چند امروزه صفحات لمسی را می‌توانید به شکلی کاملا فراگیر در گوشه و اطراف خود مشاهده کنید اما شاید تصور این موضوع چندان ساده نباشد که نخستین صفحات لمسی سال‌ها پیش در دهه ۱۹۶۰ میلادی طراحی شدند و تا به امروز تغییرات بسیاری را پشت سر گذاشته‌اند. به طور کلی در این صفحات با یکپارچه سازی سنسورهای لمسی و صفحه های نمایش، این امکان برای کاربران فراهم می‌شود که به جای استفاده از ابزارهایی نظیر ماوس و یا تاچ پد بتوانند به طور مستقیم با آنچه در صفحه می‌بینند تعامل برقرار کنند.

امروزه استفاده از تاچ اسکرین ها تنها به گوشی‌های هوشمند و تبلت ها محدود نمی‌شود بلکه در بسیاری از رایانه های قابل حمل، ساعت های هوشمند، کتاب خوان های الکترونیک، دستگاه‌های ثبت رأی الکترونیک، دستیارهای دیجیتال شخصی، کنسول های بازی، خودپردازها و ... نیز می‌توانید حضور صفحات لمسی را مشاهده کنید. حتی در نمایشگرهای راهنمای حمل و نقل، موزه‌ها و به طور کلی در هر موقعیتی که استفاده از صفحه کلید و ماوس، چندان تجربه مناسب، دقیق یا سریعی را به کاربر القا نمی کند معمولا از تاچ اسکرین‌ها استفاده می‌شود.

یک لپ تاپ مجهز به صفحه لمسی

فناوری‌های به کار رفته در صفحات لمسی

در صفحه‌های لمسی از فناوری‌های مختلفی برای شناسایی نقاط لمس شده استفاده می‌شود که از جمله آن‌ها می‌توان به موارد زیر اشاره نمود:

صفحه لمسی مقاومتی :(Resistive Touchscreen) مهم‌ترین اجزای تشکیل دهنده صفحات لمسی مقاومتی، دو لایه بسیار نازک است که با فاصله اندکی روی یکدیگر قرار گرفته‌اند (البته این‌ها تنها لایه‌های موجود در صفحات مقاومتی نیستند). به هنگام لمس لایه انعطاف‌پذیر بیرونی، بخش‌های داخلی این دو لایه که با ماده‌ای رسانا پوشیده شده است به یکدیگر متصل می‌شوند و این امر به ایجاد یک مدار الکتریکی و در نهایت تعیین مختصات نقطه لمس شده می‌انجامد. با توجه به اینکه صفحات لمسی مقاومتی به فشار حساس هستند تماس حاصل می‌تواند توسط نوک انگشت یا هر ابزار مناسب دیگری (از جمله استایلوس) نیز انجام شود. این نوع صفحه ها در مقایسه با سایر صفحات لمسی ارزان‌تر هستند و انرژی کمتری مصرف می‌کنند. به علاوه تقریبا با هر شیئی (نوک یک قلم و حتی انگشت‌هایی که با یک دستکش معمولی پوشیده شده‌اند) قابل استفاده هستند و در برابر مایعات و گرد و غبار نیز مقاوم می‌باشند. با اینحال به دلیل تعدد لایه‌ها از وضوح کمتری برخوردارند و در برابر اشیاء تیز آسیب‌پذیرتری بیشتری دارند.

صفحه لمسی خازنی :(Capacitive Touchscreen) در صفحات لمسی خازنی، برخورد آرام انگشت با صفحه (بدون نیاز به فشار دادن) منجر به تغییر میدان الکترواستاتیک صفحه می‌شود. در این صفحات، خواص رسانایی بدن باعث این تغییر میدان می‌شود و در نهایت به تشخیص نقطه لمس شده در صفحه منتج می‌شود. برخلاف صفحات مقاومتی، از آنجایی که دستکش‌های معمولی رسانا نیستند نمی‌توان از آن‌ها هنگام کار با صفحه لمسی خازنی استفاده نمود؛ در عوض، کاربر می‌تواند از دستکش‌ها یا استایلوس‌های ویژه خازنی (به جای انگشت برهنه) برای کار با این نوع صفحات لمسی استفاده کند. گفتنی است این صفحات در مقایسه با صفحات مقاومتی از وضوح بیشتری برخوردار هستند. اغلب صفحات خازنی امروزی، امکان لمس همزمان چند نقطه (Multi-touch) را برای کاربر فراهم می‌آورند.

صفحات لمسی مبتنی بر امواج آکوستیک سطحی(Surface Acoustic Wave) ، مجموعه‌ای از امواج اولتراسونیک در پنل خود ایجاد می‌کنند. هنگام لمس صفحه، بخشی از این موج جذب شده و به این ترتیب مختصات نقطه لمس شده مشخص می‌شود. این صفحات نیز می‌توانند در مقابل لمس با انگشت (حتی در صورت پوشیدن دستکش) و استایلوس، واکنش مناسب نشان دهند.

صفحات لمسی فروسرخ (Infrared) گونه‌ای دیگر از صفحات لمسی هستند که در لبه‌های آن‌ها مجموعه‌ای از LEDهای فروسرخ و حسگرهای نوری تعبیه شده است. لمس یک نقطه در این صفحات باعث قطع پرتوهای فروسرخ در آن نقطه شده و در نتیجه تشخیص مختصات آن نقطه توسط حسگرها امکان‌پذیر می‌شود.

در صفحات لمسی مبتنی بر تشخیص پالس آکوستیک(Acoustic Pulse Recognition) ، موج صوتی به وجود آمده در اثر لمس صفحه، مورد بررسی قرار گرفته و سیگنال حاصل از آن با لیستی از سیگنال‌های ناشی از لمس نقاط مختلف صفحه مقایسه می‌شود و به این ترتیب مختصات نقطه لمس شده شناسایی می‌شود.

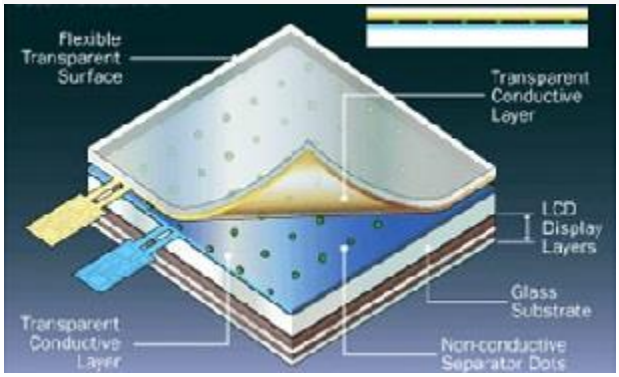
نکات قابل ذکر در مورد صفحات لمسی

یکی از معیارهای مهم در تعیین کیفیت صفحات لمسی، دقت صفحه می‌باشد. به عبارت بهتر کاربر باید بتواند به آسانی و به شکل نسبتا دقیقی همان چیزی که می خواهد را انتخاب کند. به علاوه با توجه به اینکه در صفحات لمسی، برخورد انگشت با صفحه به مراتب بیشتر از صفحات غیرلمسی است در برخی از تاچ اسکرین ها از پوشش‌های مخصوصی برای کاهش اثر انگشت جذب شده روی صفحه استفاده می‌شود.

با ظهور و گسترش صفحات لمسی، واسط های کاربری و حرکات لمسی ویژه‌ای برای بهبود تعامل کاربران با ابزارهای مجهز به این صفحات معرفی شد. هرچند صفحه کلیدهای فیزیکی و ماوس‌های رایانه‌ای طرفداران خاص خود را (حتی هنگام کار کردن با تبلت ها) دارند اما انجام بسیاری از حرکات پرکاربرد (نظیر بزرگنمایی و چرخاندن) با کمک صفحات لمسی بسیار آسانتر و سریعتر از روش معادل آن در واسط های کاربری غیرلمسی (و با کمک ماوس و صفحه کلید) می‌باشد. جالب است بدانید واکنش صفحات لمسی در برابر حرکات کاربر (نظیر لرزش دستگاه هنگام آغاز عمل دراگ کردن اشیای موجود در صفحه) می‌تواند تجربه کاربری بهتری به او منتقل نماید.

نمایشگرهای لمسی چگونه کار می‌کنند؟

این روزها وقتی از نمایشگرهای صفحه لمسی سخن به میان می‌آید، بی‌اختیار فکرها متوجه تلفن‌های همراه هوشمند و رایانه‌های لوحه‌ای (تبلت‌ها) می‌شود. سال ۱۳۸۶ که اپل، آیفون را به بازار عرضه کرد به تبدیل محصولات تجملی چون تبلت‌ها و تلفن‌های همراه هوشمند به صنایع میلیاردی دلاری سرعت بخشید و رقابت نرم‌افزاری سیستم‌های عامل iOS اندروید و ویندوزفون به صورت دیوانه‌وار اقبال عمومی از این صفحات را افزایش داده است.



به گزارش اورژانس IT باشگاه خبرنگاران،بیشتر گجت‌های لمسی موفق در چند سال گذشته در یک چیز مشترک بوده‌اند؛ یک صفحه لمسی خازنی که (capactive) دارای قابلیت تشخیص ورودی‌های متعدد در یک زمان است. به این ترتیب تعامل با یک موبایل جدید اندرویدی درست مانند تعامل با مدل اولیه یک آیفون است. با آن‌که زیربنای فناوری ساخت آنها یکسان است؛ اما موارد اختلاف بین آنها فهرستی بلند بالاست. صفحه نمایش‌های لمسی امروزی که در تبلت‌ها و تلفن‌های همراه به کار می‌روند، از نظر ساخت کمی با صفحه لمسی‌های خازنی که پیشتر ذکر کردیم، تفاوت دارند.

آنها از لایه‌های متعددی تشکیل شده‌اند. در بالا شما یک لایه شیشه‌ای یا پلاستیکی را برای حفاظت از کل لایه‌های مجتمع صفحه لمسی دارید. این لایه معمولا از جنسی ضد خش مانند شیشه Corning's Gorilla glass ساخته می‌شود تا صفحه موبایلتان در مجاورت با محتویات داخل جیب شما آسیب نبیند. لایه زیرین آن لایه خازنی است که مسئولیت هدایت مقدار ناچیزی الکتریسیته را به عهده داشته و روی لایه نازک شیشه‌ای دیگری قرار دارد. در قسمت زیر آن خود پنل LCD

جا گرفته است. هنگامی که انگشت شما به عنوان یک جسم رسانای طبیعی صفحه را لمس می‌کند، با میدان الکتریکی لایه خازنی تداخل پیدا می‌کند.

داده حاصله از این کار به یک تراشه کنترل‌کننده که موقعیت محل تماس (اغلب فشار) را ثبت می‌کند، انتقال یافته و از سیستم عامل می‌خواهد مطابق با آن پاسخ دهد. این مقدمات به خودی خود به طور دقیق فقط می‌تواند یک نقطه تماس را در یک زمان کشف کرده و با لمس دو نقطه یا بیشتر، موقعیت نقطه تماس را غلط تفسیر کرده یا اصلا حس نمی‌کند. برای ثبت نقاط لمس متعدد به صورت مشخص، لایه خازنی باید متشکل از دو لایه مختلف باشد؛ یکی از الکترودهای فرستنده سود ببرد و دیگری از الکترودهای گیرنده. این لایه‌های الکترود به صورت یک شبکه توری روی صفحه نمایش گجت آرایش گرفته‌اند.

هنگامی که انگشت شما صفحه را لمس می‌کند، بین آن با سیگنال الکتریکی بین الکترودهای گیرنده و فرستنده تداخل ایجاد می‌شود. زمانی که انگشت شما به عنوان یک رسانای الکتریکی، صفحه را لمس می‌کند، با میدان الکتریکی که به وسیله الکترودهای فرستنده ایجاد شده و به سوی الکترودهای گیرنده ارسال می‌شود، تداخل پیدا کرده و وسیله به این ترتیب یک لمس راثبت می‌کند. به دلیل آرایش شبکه میله‌ی الکترودی کنترل‌کننده می‌تواند بیش از یک نقطه لمس را به طور همزمان تشخیص دهد.

بیشتر موبایل‌ها و رایانه‌های لوحه‌ای امروزی بین دو تا ده نقطه تماس را به صورت همزمان مورد پشتیبانی قرار می‌دهند. سطوح چند لمسی صفحه نمایش‌ها حرکت‌های اشاره‌ای مانند نیشگون گرفتن (gestures) روی صفحه جهت بزرگنمایی و چرخاندن تصویر را ممکن می‌کنند. هدایت و ناوبری در داخل سیستم عامل یک موبایل امروزی از امور ضروری است که بدون توانایی صفحه نمایش‌ها برای تشخیص همزمان لمس‌های متعدد ممکن نیست. امروزه اینها پایه و اساس تلفن‌های همراه هوشمند، تبلت‌ها و رایانه‌های دسکتاپ (PC) لمسی هستند.

امروزه ما پوشش مخصوص اولئوفوبیک (oleophobic) یا به زبان ساده ضدچربی را داریم که به لایه رویی صفحه نمایش‌ها افزوده شده است. این لایه صفحه نمایش را از چربی باقیمانده در اثر انگشت محافظت می‌کند. ما شاهد دو نگارش از شیشه ضد خش و نشکن گوریلا گلس هستیم که لایه شیشه‌ای محافظ را نازک‌تر و نسبت به خط و خش مقاوم‌تر کرده است. آخرین فناوری هم in-cell است که لایه خازنی لمسی را در خود LCD جاسازی کرده است و مجموعه‌های هماهنگی از شبکه‌های میله‌های الکترودی را به کار می‌برد.

این فناوری باعث کاهش ضخامت کلی و پیچیدگی اجزای تشکیل‌دهنده صفحات نمایش لمسی شده است. با این فناوری گجت قادر است محل نقاط لمس متعدد را به صورت همزمان و دقیقا تشخیص دهد؛ اما هیچ یک از این تغییرات از اهمیت اساسی فناوری چند نقطه لمسی خازنی اصلی برخوردار نبوده و فقط شرایط باریک‌تر و سبک‌تر شدن تلفن‌های همراه را فراهم کرده‌اند.

آینده در دست فناوری هوا لمسی (AirTouch)

محقق‌ها بتازگی موفق به ساخت یک قاب (frame) خالی بسیار حساس و دارای ظرفیت پاسخگویی بسیار بالای لمسی شده‌اند. حسگرهای مادون قرمز و LED تصور واضحی از هر شیئی را که در این قاب قرار بگیرد، ممکن می کنند.
این فناوری روش گرانقیمتی برای تبدیل صفحه نمایش‌های بزرگ به صفحه نمایش‌های تعاملی چند لمسی است. برای سال‌ها این فناوری محدود مانده و درجا می‌زد تا این که بتازگی یک نمونه آزمایشی از این فناوری را ساخته‌اند که ZeroTouchنامیده شده و گرچه در ظاهر مانند یک قاب خالی به نظر می‌آید، اما سرشار از قابلیت‌های پیشرفته باورنکردنی است و یک نمونه ۲۸ اینچی که گوشه‌های صدفی دارد، می‌تواند هر چه که داخلش قرار بگیرد از جمله تماس انگشتان دست و حرکات قسمت‌های دیگر بدن را نیز در یک شبکه دو بعدی نوری تشخیص دهد.

جمع‌بندی

موفقیت تلفن‌های همراه و تبلت‌های لمسی دو اثر بزرگ از خود به جا گذاشته است؛ اول این که کاربران شروع به پذیرش لمسی حتی در جاهایی که این فناوری جایی نداشته است، کرده‌اند. در روی دیگر سکه، تولیدکنندگان سخت‌افزارها دنبال تکرار موفقیت گجت‌های لمسی در حوزه‌های دیگری چون رایانه‌های دسکتاپ، لپ‌تاپ‌ها و الترابوک‌های دارای سیستم عامل کروم و ویندوز ۸ و حتی ساعت‌های مچی هستند. همچنین کنسول‌های بازی مانند PlayStation Vita۴ ، PlayStationسونی و وی‌یو (Wii U) شرکت نینتندو صفحه لمسی‌ها را به عنوان دومین رابط کاربری ورودی سیستم علاوه بر دسته‌های کنترل بازی اصلی‌شان وارد میدان کرده‌اند. رایانه‌های یکپارچه (All-in-one) یا آیو (AiO) نیز از نمونه‌های حضور فناوری لمسی در رایانه‌های دسکتاپ هستند؛ اما در آینده نزدیک دیگر نیازی به لمس هیچ صفحه یا شیئی برای تعامل با گجت یا رایانه‌تان ندارید، چرا که همه این تعامل‌ها را با لمس کردن هوا انجام خواهید داد!

جسچرها و حرکات لمسی در اندروید

ساخت یک اپلیکیشن تعاملی تنها با onClick امکان پذیر نیست و SDKاندروید از حرکات متنوعی پشتیبانی می کند. بدین ترتیب توسعه دهندگان قادر به طراحی روش های جدیدی جهت تعامل کاربران با اپلیکیشن می گردند. در اندروید از کلاس GestureDetector به منظور تشخیص ایونت های حرکتی مقدماتی و از MotionEventبرای مدیریت حرکات پیچیده تر استفاده می شود.

1. GestureDetector

کلاس GestureDetector **ایونت های حرکتی را که به مجموعه خاصی از حرکات کاربر ارتباط پیدا می کنند دریافت می نماید. این حرکات شامل ضربه زدن، کشیدن افقی و عمودی، فشار کوتاه و طولانی بر روی صفحه، دوبار ضربه زدن و اسکرول ها می شود** .GestureDetector **در مدیریت تعاملات استاندارد کاربری عملکرد قدرتمندی دارد و ست کردن** SimpleOnGestureLister **در عناصر رابط کاربری اندروید به سادگی صورت می پذیرد.**

این کلاس به اورراید کردن متدهایی موردنیاز از GestureLister **پرداخته و بدون نیاز به تشخیص دستی حرکت صورت گرفته از سوی کاربر به اجرای آن می پردازد.**

تشخیص جسچرهای متداول

مثال زیر نمونه ای از جسچرهای متداول کلاس GestureDetector **می باشد. در گام نخست لازم است یک نمونه از کلاس** GestureDetector **را در داخل کلاس اکتیویتی خود تعریف کنید.**

```
GestureDetector mGestureDetector;
```

در داخل کلاس اکتیویتی یک کلاس جاوا بسازید که دارای رابط های GestureDetector.OnGestureListener**و** GestureDetector.OnDoubleTapList **به منظور تشخیص حرکات پایه اندروید باشد.**

```
class Android_Gesture_Detector implements GestureDetector.OnGestureListener,
        GestureDetector.OnDoubleTapListener {
    @Override
    public boolean onDown(MotionEvent e) {
        Log.d("Gesture ", " onDown");
        return true;
    }
  @Override
    public boolean onSingleTapConfirmed(MotionEvent e) {
        Log.d("Gesture ", " onSingleTapConfirmed");
        return true;
    }
  @Override
    public boolean onSingleTapUp(MotionEvent e) {
        Log.d("Gesture ", " onSingleTapUp");
        return true;
    }
    @Override
    public void onShowPress(MotionEvent e) {
        Log.d("Gesture ", " onShowPress");
    }
  @Override
    public boolean onDoubleTap(MotionEvent e) {
        Log.d("Gesture ", " onDoubleTap");
```

```
        return true;
      }
      @Override
      public boolean onDoubleTapEvent(MotionEvent e) {
        Log.d("Gesture ", " onDoubleTapEvent");
        return true;
      }
      @Override
      public void onLongPress(MotionEvent e) {
        Log.d("Gesture ", " onLongPress");
      }
    @Override
    public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY) {
        Log.d("Gesture ", " onScroll");
        if (e1.getY() < e2.getY()){
          Log.d("Gesture ", " Scroll Down");
        }
       if(e1.getY() > e2.getY()){
          Log.d("Gesture ", " Scroll Up");
        }
        return true;
    }
      @Override
      public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {
    if (e1.getX() < e2.getX()) {
          Log.d("Gesture ", "Left to Right swipe: "+ e1.getX() + " - " + e2.getX());
          Log.d("Speed ", String.valueOf(velocityX) + " pixels/second");
     }
        if (e1.getX() > e2.getX()) {
          Log.d("Gesture ", "Right to Left swipe: "+ e1.getX() + " - " + e2.getX());
    Log.d("Speed ", String.valueOf(velocityX) + " pixels/second");
        }
        if (e1.getY() < e2.getY()) {
        Log.d("Gesture ", "Up to Down swipe: " + e1.getX() + " - " + e2.getX());
          Log.d("Speed ", String.valueOf(velocityY) + " pixels/second");
        }
        if (e1.getY() > e2.getY()) {
          Log.d("Gesture ", "Down to Up swipe: " + e1.getX() + " - " + e2.getX());
    Log.d("Speed ", String.valueOf(velocityY) + " pixels/second");
        }
        return true;
      }
}
```

کلاس Android_Gesture_Detector **به رهگیری تمامی جسچرهای مقدماتی اجرا شده درون خود می پردازد، اما به منظور استفاده از این جسچرها باید اقدام به اورراید نمودن متد** onTouchEvent() **جهت رهگیری ایونت های تاچ نموده و آنها را به کلاس** Android_Gesture_Detector **تغییر مسیر دهید.**

```
@Override
public boolean onTouchEvent(MotionEvent event) {
   mGestureDetector.onTouchEvent(event);
   return super.onTouchEvent(event);
   // Return true if you have consumed the event, false if you haven't.
   // The default implementation always returns false.
```

```
}
```

قسمت زیر را به متد onCreate() **بیفزایید، با این کار** gesture detector **اندروید قادر به تشخیص تعاملات کاربر می گردد.**

```
// Create an object of the Android_Gesture_Detector  Class
Android_Gesture_Detector  android_gesture_detector  =  new Android_Gesture_Detector();
// Create a GestureDetector
mGestureDetector = new GestureDetector(this, android_gesture_detector);
```

تست کردن کلاس gesture

اپلیکیشن را به اجرا در آورده و با آن کار کنید و نتیجه را که در پنجره Android Studio LogCat **به نمایش در آمده مشاهده کنید.**

```
D/Gesture: Left to Right swipe: 222.0 - 443.5
D/Speed: 4637.715 pixels/second
D/Gesture: Up to Down swipe: 222.0 - 443.5
D/Speed: 5615.9194 pixels/second
```

پس از این مرحله می توانید به ساخت عملکرد اپلیکیشن درون متدهای اوررایدشده بپردازید.

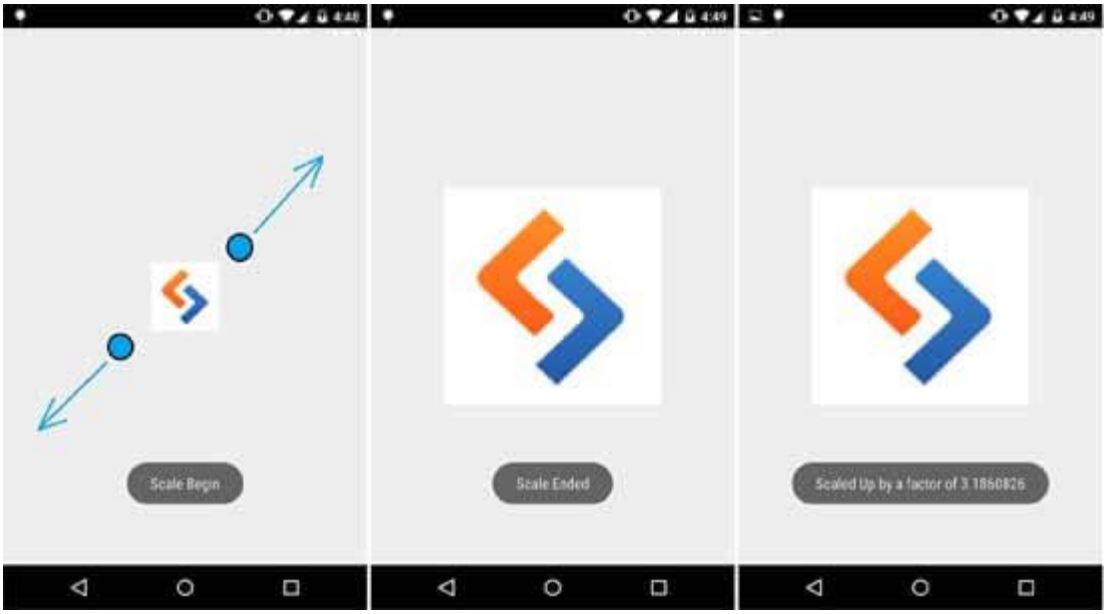
Pinch Gesture

قابلیت تغییر اندازه عناصر رابط کاربری در صفحه یکی دیگر از حرکات کاربردی در هر اپلیکیشن می باشد که با کمک کلاس ScaleGestureDetector **جهت تغییر اندازه ویوها قادر به استفاده از این ویژگی می باشیم. در کد زیر از کلاس** ScaleListener **جهت اجرای حرکت** pinch **در** imageView **استفاده شده و می توان با حرکت دادن انگشت عناصر صفحه را کوچک و بزرگ کرد.**

```
import android.app.Activity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.ScaleGestureDetector;
import android.widget.ImageView;
import android.widget.Toast;
public class MainActivity extends Activity {
   private ImageView imageView;
   private float scale = 1f;
   private ScaleGestureDetector detector;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      imageView=(ImageView)findViewById(R.id.imageView);
      detector = new ScaleGestureDetector(this,new ScaleListener());
   }
public boolean onTouchEvent(MotionEvent event) {
//  re-route the Touch Events to the ScaleListener class
      detector.onTouchEvent(event);
      return super.onTouchEvent(event);
   }
```

```
   private class ScaleListener extends
ScaleGestureDetector.SimpleOnScaleGestureListener {
      float onScaleBegin = 0;
      float onScaleEnd = 0;
      @Override
      public boolean onScale(ScaleGestureDetector detector) {
         scale *= detector.getScaleFactor();
         imageView.setScaleX(scale);
         imageView.setScaleY(scale);
         return true;
      }
      @Override
      public boolean onScaleBegin(ScaleGestureDetector detector) {
         Toast.makeText(getApplicationContext(),"Scale Begin" ,Toast.LENGTH_SHORT).show();
         onScaleBegin = scale;
         return true;
      }
      @Override
      public void onScaleEnd(ScaleGestureDetector detector) {
         Toast.makeText(getApplicationContext(),"Scale Ended",Toast.LENGTH_SHORT).show();
         onScaleEnd = scale;
         if (onScaleEnd > onScaleBegin){
            Toast.makeText(getApplicationContext(),"Scaled Up by a factor of " + String.valueOf(
onScaleEnd / onScaleBegin ), Toast.LENGTH_SHORT ).show();
         }
         if (onScaleEnd < onScaleBegin){
            Toast.makeText(getApplicationContext(),"Scaled Down by a factor of " +
String.valueOf( onScaleBegin / onScaleEnd ), Toast.LENGTH_SHORT ).show();
         }
         super.onScaleEnd(detector);
      }
   }
}
```

کلاس ScaleListener **سه متد** onScale ، onScaleBegin **و** onScaleEnd **را اورراید می کند. در حین اجرای متد** onScale **تغییر اندازه** iamgeView **بر پایه مقیاسی از حرکت انگشت بر روی صفحه نمایش گوشی به اجرا در می آید**.



2. Motion Event

Android Simple GestureDetector **برای حرکات ساده و مقدماتی بسیار کاربردی است، اما برای حرکاتی که شامل دو یا تعداد بیشتری تاچ هستند به متد دیگری نیاز دارید. لزومی به استفاده از ایونت های تاچ در تمامی اپلیکیشن ها احساس نمی شود اما در صورت تمایل به ساخت یک اپلیکیشن کاملا تعاملی استفاده از این کلاس تنها راه حل است. وظیفه اصلی کلاس** MotionEvent **گرفتن ایونت های تاچ در یک اکتیویتی و اورراید کردن فراخوانی** onTouchEvent() **است و در ویوها با استفاده از متد** setOnTouchListener() **که برای گوش دادن به ایونت های تاچ در ویو استفاده می شود کار خود را به انجام می رساند. همانطور که در مثال زیر نیز نمایش داده شده می توان با اجرای** View.onTouachListener **در کلاس اکتیویتی ایونت های تاچ را در ویو گرفت.**

اندروید به تولید ایونت های تاچ زیر در هر جایی که صفحه با یک یا چند انگشت لمس شود می پردازد.

ACTION_DOWN

برای نخستین اشاره گری که صفحه را لمس می کند تاچ جدید شروع می شود.

ACTION_MOVE

یک تغییر در حرکت تاچ صورت گرفته و انگشت در حال حرکت است.

ACTION_UP

آخرین اشاره گر صفحه را ترک می کند.

ACTION_POINTER_DOWN

برای اشاره گرهای اضافه ای که پس از اشاره گر اول وارد صفحه می شوند یعنی مولتی تاچ.

ACTION_POINTER_UP

زمانی که یک اشاره گر غیراولیه بالا می آید (مولتی تاچ).

ACTION_CANCEL

ایونت تاچ کنسل شده و چیز دیگری کنترل ایونت را بر عهده می گیرد.

جهت نمایش کاربرد کلاس MotionEvent کار را با مثالی ادامه می دهیم. حرکات به منظور تعامل با رابط کاربری اپلیکیشن اند، بنابراین به شرح مثالی از یک اپلیکیشن که ویوها را در لی اوت اصلی با کمک یک حرکت انگشت جابجا می کند، اندازه ویو را با کمک حرکت دو انگشت بزرگ می کند و به کمک سه انگشت می چرخاند می پردازیم.

بیایید کار را با این ایونت های تاچ آغاز نماییم، یک اکتیویتی اندروید بسازید که دارای View.OntouchListenerباشد. آی دی rootLayout را در فایل اکتیویتی به لی اوت Relative اصلی ست کنید. هدف افزودن عناصر imageView در لی اوت روت می باشد که هرکدام دارای قابلیت عملکرد بر اساس یک ایونت تاچ هستند. جهت افزودنimageView های جدید بر روی لی اوت دکمه ای را که با هر بار کلیک تابع Add_Image() را فراخوانی می کند افزوده ایم.

این تابع یک imageView می سازد و یک منبع برای آن ست می کند، یک مجموعه از پارامترهای لی اوت را می سازد و آنها را به imageView متصل می نماید، سپس این imageView را به لی اوت روت اضافه کرده و Touch Listener را به این ویو ست می کند.

```
private void Add_Image() {
    final ImageView iv = new ImageView(this);
    iv.setImageResource(R.drawable.image);
    RelativeLayout.LayoutParams layoutParams = new RelativeLayout.LayoutParams(150,
150);
    iv.setLayoutParams(layoutParams);
    RootLayout.addView(iv, layoutParams);
    iv.setOnTouchListener(this);
  }
```

مهم ترین بخش TouchListener ویو است. این لیسنر اکشن ها را بر اساس ایونت های تاچی که در بالا شرح داده شد به اجرا در می آورد.

در حین اجرای ایونت ACTION_DOWN مکان های X و Y را دریافت می کنیم و حاشیه های مربوطه را حذف می نماییم تا به نقطه دقیق در ویو که تاچ بر روی آن صورت گرفته برسیم.

در حین اجرای ایونت ACTION_UP یک ایونت دابل کلیک کاستوم جهت پاک کردن ویویی که این ایونت بر روی آن انجام شده اجرا شده است.

در حین ACTION_MOVE اکشن هایی برای امکانات جالبی که در بالا شرح داده شد به اجرا در آمده است. یکی از ویژگی های جالب درباره کلاس MotionEvent قابلیت تشخیص تعداد انگشت هایی است که صفحه را لمس کرده اند و برای این کار از متد getPointerCount() استفاده شده است.

یک انگشت – جابجایی ویو

LayoutParamsمربوط به ویو را بگیرید و آن را بر اساس حرکت انگشت بر روی صفحه ریست کنید. حاشیه سمت چپ برابر با مکان ایونت تاچ منهای مقدار X از تاچ درون ویو که در حین ACTION_DOWN ست شده است می باشد، بنابراین ویو بر اساس مقدار X از انگشت که بر روی لی اوت قرار گرفته جابجا می شود و خود را بر اساس مکان تاچ درون ویو تراز می کند.

همین منطق برای حاشیه بالایی نیز صدق می کند، اما بر اساس محور Y. مقدار ۵۰۰ برای حاشیه های راست و پایین ست شده است، بنابراین لمس حاشیه های راست و پایین لی اوت اندازه را کم نمی کند و تنها موجب پنهان شدن آن قسمت می گردد.

```
RelativeLayout.LayoutParams Params = (RelativeLayout.LayoutParams)
view.getLayoutParams();
            Params.leftMargin = X - Position_X;
            Params.topMargin = Y - Position_Y;
            Params.rightMargin = -500;
            Params.bottomMargin = -500;
            view.setLayoutParams(Params);
```

دو انگشت – بزرگ کردن ویو

با گرفتن LayoutParams ویو عرض و طول آن را تغییر دهید. در حین عملیات تغییر اندازه حاشیه های ویو تغییر پیدا نمی کنند و عملیات تغییر اندازه بر اساس حرکت انگشت اشاره صورت می پذیرد، بنابراین اندازه ها با توجه به مکان قرارگیری انگشت اشاره بر روی لی اوت ست می شوند. مقادیر ثابت Position_X و Position_Yتاثیری بر روی عملیات تغییر اندازه ندارند و تنها برای موارد دیداری مورد استفاده قرار می گیرند.

```
RelativeLayout.LayoutParams layoutParams1 = (RelativeLayout.LayoutParams)
view.getLayoutParams();
            layoutParams1.width = Position_X +(int)event.getX();
            layoutParams1.height = Position_Y + (int)event.getY();
            view.setLayoutParams(layoutParams1);
```

سه انگشت – چرخش ویو

چرخش ویو به سادگی صورت می پذیرد. چرخش فعلی ویو را بگیرید و یک مقدار ثابت از نوع شناور را به آن بیفزایید و در حین ACTION_MOVE با حرکت سه انگشت با استفاده از متد setRotation() مجددا آن را برای ویو ست کنید.

```
view.setRotation(view.getRotation() + 10.0f);
```

به خاطر داشته باشید که عملیات تغییر اندازه ویو پس از ایونت چرخش به درستی کار نخواهد کرد زیرا محورهای ویو به وضعیت چرخشی کنونی چرخانده شده اند.

کد کامل برای اکتیویتی MotionEvent به صورت زیر می باشد:

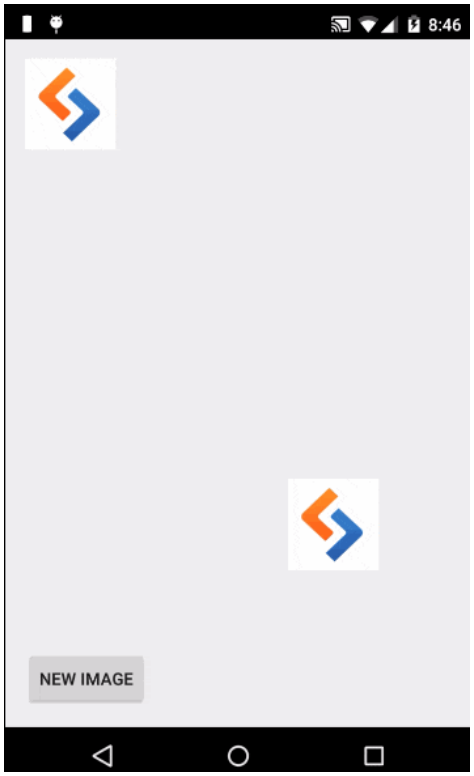
```
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.RelativeLayout;
public class MainActivity extends Activity implements View.OnTouchListener {
   int clickCount;
   private ViewGroup RootLayout;
   private int Position_X;
   private int Position_Y;
```

```
  long startTime = 0 ;
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    RootLayout = (ViewGroup) findViewById(R.id.rootLayout);
    //new image
    Button NewImage =  (Button)findViewById(R.id.new_image_button);
    NewImage.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        Add_Image();
      }
    });
    clickCount = 0;
  }
  private void Add_Image() {
    final ImageView iv = new ImageView(this);
    iv.setImageResource(R.drawable.image);
    RelativeLayout.LayoutParams layoutParams = new RelativeLayout.LayoutParams(150,
150);
    iv.setLayoutParams(layoutParams);
    RootLayout.addView(iv, layoutParams);
    iv.setOnTouchListener(this);
  }
  public boolean onTouch(final View view, MotionEvent event) {
    final int X =  (int) event.getRawX();
    final int Y =  (int) event.getRawY();
    int pointerCount = event.getPointerCount();
      switch (event.getAction() & MotionEvent.ACTION_MASK) {
        case MotionEvent.ACTION_DOWN:
          RelativeLayout.LayoutParams layoutParams = (RelativeLayout.LayoutParams)
view.getLayoutParams();
          Position_X = X - layoutParams.leftMargin;
          Position_Y = Y - layoutParams.topMargin;
          break;
        case MotionEvent.ACTION_UP:
          if (startTime == 0){
            startTime = System.currentTimeMillis();
          }else {
            if (System.currentTimeMillis() - startTime < 200) {
              AlertDialog.Builder  builder = new AlertDialog.Builder(MainActivity.this);
              builder.setMessage("Are  you sure you want to delete this?");
              builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                  view.setVisibility(View.GONE);
                }
              });
              builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                  dialog.dismiss();
                }
```

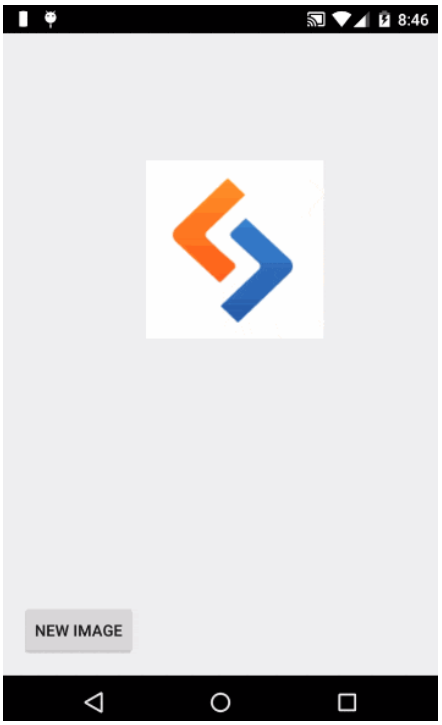
```
                    });
                    AlertDialog alertDialog = builder.create();
                    alertDialog.show();
                }
                startTime = System.currentTimeMillis();
            }
            break;
        case MotionEvent.ACTION_POINTER_DOWN:
            break;
    case MotionEvent.ACTION_POINTER_UP:
            break;
        case MotionEvent.ACTION_MOVE:
            if (pointerCount == 1){
                RelativeLayout.LayoutParams Params = (RelativeLayout.LayoutParams)
view.getLayoutParams();
                Params.leftMargin = X - Position_X;
                Params.topMargin = Y - Position_Y;
                Params.rightMargin = -500;
                Params.bottomMargin = -500;
                view.setLayoutParams(Params);
            }
            if (pointerCount == 2){
                RelativeLayout.LayoutParams layoutParams1 =  (RelativeLayout.LayoutParams)
view.getLayoutParams();
                layoutParams1.width = Position_X +(int)event.getX();
                layoutParams1.height = Position_Y + (int)event.getY();
                view.setLayoutParams(layoutParams1);
            }
            //Rotation
            if (pointerCount == 3){
                //Rotate the ImageView
                view.setRotation(view.getRotation() + 10.0f);
            }
            break;
        }
// Schedules a repaint for the root Layout.
      RootLayout.invalidate();
      return true;
    }
}
```

تست اپلیکیشنMotionEvent

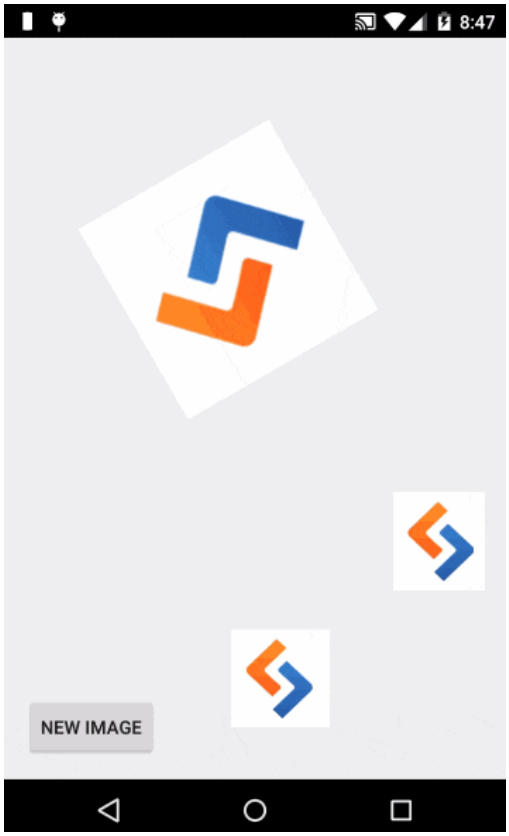
ویوی جابجایی



ویوی تغییر اندازه



ویوی چرخش



جمع بندی

در اینجا دو نوع اصلی از کلاس های جسچر در اندروید شرح داده شد و با کمک دانش مقدماتی که در اختیار شما قرار گرفت می توانید با این جسچرها کار کنید و به ارتقای تجربه کاربری اپلیکیشن خود بپردازید.