Мульти-тач

Новые возможности добавляются в платформу с каждой новой версией Android.

В э той части мы сконцентрируемся на таких новых возможностях и на подготов-

ке ваших программ таким образом, чтобы они стали доступны другим пользова-

телям.

В этой главе мы узнаем, как использовать новую возможность мульти-тач в AAnndroid

2.0 «на полную катушку». Затем мы рассмотрим виджеты домашнего экрана,

представленные в Android 1.6, и интерактивные обои, представленные в Android 2.1.

Популярность Android и стремительные темпы его разработки создали проблему

с фрагментацией, поэтому целая глава посвящена работе с различными версиями

и экранными разрешениями, которые могут встретиться на практике. И наконец,

здесь будет раздел, который рассказывает о том, как передать вашу программу

пользователю, публикуя ее на Android Market.

11.1. Введение в мульти-тач

Мульти-тач — это обычное расширение привычного интерфейса с тач-скрином

с использованием двух или нескольких пальцев вместо одного. Ранее1 мы исполь-

зовали жесты, выполняемые одним пальцем, хотя мы их так не называли. Помните,

в подразделе 4.3, «Ввод чисел», мы предоставили пользователю возможность при-

касаться к тайлу в игре Sudoku для того, чтобы изменять его? Это называется *tap*

*gesture* (жест касания). Другой жест называется *drag* (перетаскивание). Это когда

вы удерживаете палец на экране и перемещаете его, что приводит к прокручиванию

содержимого экрана.

Касания, перетаскивания и некоторые другие жесты, выполняемые одним

пальцем, всегда поддерживались в Android. Однако благодаря популярности Apple

iPhone первые пользователи Android жутко завидовали «яблочникам». iPhone

поддерживает мульти-тач, в особенности жест «pinch zoom» (изменение масштаба

изображения двумя пальцами) (рис. 11.1).

1 Некоторые люди используют их больше, чем остальные.

11

11.1. Введение в мульти-тач **201**

**Рис. 11.1.** Три обычных жеста: a — касание, б — перетаскивание,

в — изменение масштаба двумя пальцами

Выполняя изменение масштаба, вы касаетесь экрана двумя пальцами и сводите

их, для того чтобы сделать просматриваемый объект меньше, или разводите — для

увеличения. До Android 2.0 вы должны были использовать неуклюжий элемент

управления масштабированием со значком, который вы нажимали для увеличения

и уменьшения масштаба (см. метод setBuiltInZoomControls() в подразделе 8.3, «Под-

готовка»). Но благодаря поддержке мульти-тач теперь изменение масштаба произ-

водится двумя пальцами и в Android — конечно, если приложение поддерживает

эту технологию.

Обратите внимание на то, что Android 2.2 представил новый класс, называемый

ScaleGestureDetector, который распознает жест изменения масштаба. Однако я решил

его не использовать, для того чтобы обеспечить совместимость с устройствами 2.0

и 2.1. Если вам нужен Android��Android ��2.2��������������������� или выше — обратитесь\_\_ ������������������к онлайновой докум\_е��н-

тации1 за подробностями.

При попытке запустить пример из этой главы на Android��Android ��1.5������� или 1.6���������� он завер-

шится с ошибкой, так как эти версии не поддерживают мульти-тач. Мы узнаем,

как с этим работать в разделе 13.3 «Разворачивание программы на различных API

Android».

1 http://d.android.com/reference/android/view/ScaleGestureDetector. html

**202** Глава 11 • Мульти-тач

**ВНИМАНИЕ! МУЛЬТИ-ОШИБКИ ВПЕРЕДИ**

Мульти-тач в том виде, в котором он реализован в существующих Android-телефонах, полон

ошибок. В действительности недоработок столько, что это граничит с полной неработоспособ-

ностью. ��������������������������������������������������������������������������������API����������������������������������������������������������������������������� постоянно сообщает неверные или невозможные данные о точках касания, особен-

но при переходе от одного пальца к двум при работе с экраном, и наоборот.

На форуме разработчиков вы найдете жалобы на то, что касания пальцами меняют координат-

ные оси X и Y, несколько пальцев иногда распознаются как один. Некоторые из этих проблем

могут относиться к ограничениям аппаратного обеспечения сенсоров касания тач-скринов, ис-

пользованных в определенных телефонах, однако большая часть может быть исправлена или

улучшена путем обновления программного обеспечения.

Путем множества проб и ошибок я смог сделать пример в этой главе работающим, так как

жесты, которые в нем используются, довольно просты. До тех пор, пока Google не признает

и не исправит проблемы с мульти-тач, это, возможно, все, что можно сделать. К счастью,

жест изменения масштаба двумя пальцами кажется единственным мульти-тач-жестом, которым

с удовольствием пользуются большинство людей.

11.2. Создание примера Touch

Для демонстрации технологии мульти-тач мы собираемс я создать простой про-

смотрщик изображений, который позволяет изменять масштаб и прокручивать

изображение. Готовый проект показан на рис. 11.2.

Начнем с создания проекта «���� o,�AndroidHello, Android��»�� \_с�� о\_с����леду\_ющ\_\_�и\_м������ ипара\_ме����трам\_и������� в диа-

логовом окне New Android Project:

Project name: Touch

Build Target: Android 2.2

Application name: Touch

Package name: org.example.touch

Create Activity: Touch

Min SDK Version: 8

Благодаря этим установкам будет создан T�ou���ch..��������java,� с\_о������держащ\_и���������й вашу ос\_-

новную деятельность. Внесем в него правки, для того чтобы он мог отображать

демонстрационное изображение, поместим в него обработчик касаний и добавим

несколько команд импорта, которые будут нужны нам позже.

**Touchv1/src/org/example/touch/Touch.java**

package org.example.touch;

import android.app.Activity;

import android.graphics.Matrix;

import android.graphics.PointF;

import android.os.Bundle;

import android.util.FloatMath;

import android.util.Log;

import android.view.MotionEvent;

import android.view.View;

import android.view.View.OnTouchListener;

import android.widget.ImageView;

public class Touch extends Activity implements OnTouchListener {

private static final String TAG = "Touch" ;

11.2. Создание примера Touch **203**

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

ImageView view = (ImageView) findViewById(R.id.imageView);

view.setOnTouchListener(this);

}

@Override

public boolean onTouch(View v, MotionEvent event) {

// Обработка событий касаний...

}

}

**Рис. 11.2.** Пример реализации технологии касания экрана в простом просмотрщике

изображений с возможностью изменения масштаба двумя пальцами

**204** Глава 11 • Мульти-тач

Сейчас мы заполним метод onTouch(). Для начала нам нужно определить макет

для нашей деятельности.

**Touchv1/res/layout/main.xml**

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

<FrameLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent" >

<ImageView android:id="@+id/imageView"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:src="@drawable/butterfly"

android:scaleType="matrix" >

</ImageView>

</FrameLayout>

Интерфейс представляет собой большой элемент управления ImageView, кото-

рый занимает весь экран. Значение android:src=»@drawable/butterfly» ссылается на

изображение бабочки, использованное в примере. Используйте любой файл форма-

та JPG или PNG по своему выбору; просто поместите его в папку res/drawable-nodpi.

Атрибут android:scaleType=»matrix» показывает, что мы собираемся использовать

матрицу для управления позицией и масштабом изображения. Подробнее мы по-

говорим об этом позже.

Файл AndroidManifest.xml останется нетронутым, за исключением добавления

атрибута android:theme=.

**Touchv1/AndroidManifest.xml**

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

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="org.example.touch"

android:versionCode="1"

android:versionName="1.0" >

<application android:icon="@drawable/icon"

android:label="@string/app\_name"

android:theme="@android:style/Theme.NoTitleBar.Fullscreen" >

<activity android:name=".Touch"

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-sdk android:minSdkVersion="3" android:targetSdkVersion="8" />

</manifest>

@android:style/Theme.NoTitleBar.Fullscreen, как можно понять из его имени, ука-

зывает Android на использование полного экрана без заголовка окна или строки

состояния в верхней части. Если вы запустите приложение сейчас, оно просто по-

кажет изображение.

11.3. Изучение событий касания **205**

11.3. Изучение событий касания

Когда я впервые изучаю новое A�PI,������ то час\_����то и\_с���пол\_ь��зу\_ю���� код,����������� который по\_с����������тавляет от-

ладочную информацию обо всем, что происходит; в итоге я получаю представление

о том, что делает метод, и в каком порядке происходят события. С э того и начнем.

Для начала добавим вызов метода dumpEvent() внутрь onTouch().

**Touchv1/src/org/example/touch/Touch.java**

@Override

public boolean onTouch(View v, MotionEvent event) {

// Записываем события касаний в журнал

dumpEvent(event);

return true; // показываем, что событие было обработано

}

Обратите внимание на то, что нам нужно вернуть true, чтобы показать Android,

что событие было обработано. Далее определим метод dumpEvent(). Его един-

ственный параметр — событие, отладочную информацию по которому мы хотим

сохранить.

**Touchv1/src/org/example/touch/Touch.java**

/\*\* Показывает информацию о событии в окне LogCat для отладочных целей \*/

private void dumpEvent(MotionEvent event) {

String names[] = { "DOWN" , "UP" , "MOVE" , "CANCEL" , "OUTSIDE" ,

"POINTER\_DOWN" , "POINTER\_UP" , "7?" , "8?" , "9?" };

StringBuilder sb = new StringBuilder();

int action = event.getAction();

int actionCode = action & MotionEvent.ACTION\_MASK;

sb.append("event ACTION\_" ).append(names[actionCode]);

if (actionCode == MotionEvent.ACTION\_POINTER\_DOWN

|| actionCode == MotionEvent.ACTION\_POINTER\_UP) {

sb.append("(pid " ).append(

action >> MotionEvent.ACTION\_POINTER\_ID\_SHIFT);

sb.append(")" );

}

sb.append("[" );

for (int i = 0; i < event.getPointerCount(); i++) {

sb.append("#" ).append(i);

sb.append("(pid " ).append(event.getPointerId(i));

sb.append(")=" ).append((int) event.getX(i));

sb.append("," ).append((int) event.getY(i));

if (i + 1 < event.getPointerCount())

sb.append(";" );

}

sb.append("]" );

Log.d(TAG, sb.toString());

}

Вывод данных производится в отладочный журнал Android, который вы можете

увидеть в окне LogCat (см. подраздел 3.10, «Отладка с помощью записи сообщений

в журнал»).

**206** Глава 11 • Мульти-тач

Самый простой способ понять этот код — запустить его. К несчастью, вам не

удастся запустить эту программу в эмуляторе (в действительности удастся, но

эмулятор не поддерживает мульти-тач, поэтому результаты не будут представлять

интереса). Поэтому подключите телефон к USB-порту и запустите пример на нем

(см. подраздел 1.4, «Запуск на реальном телефоне»).

Когда я испытываю это на моем телефоне и делаю несколько быстрых жестов,

я получаю следующие выходные данные:

Стр. 1 event ACTION\_DOWN[#0(pid 0)=135,179]

- event ACTION\_MOVE[#0(pid 0)=135,184]

- event ACTION\_MOVE[#0(pid 0)=144,205]

- event ACTION\_MOVE[#0(pid 0)=152,227]

5 event ACTION\_POINTER\_DOWN(pid 1)[#0(pid 0)=153,230;#1(pid 1)=380,538]

- event ACTION\_MOVE[#0(pid 0)=153,231;#1(pid 1)=380,538]

- event ACTION\_MOVE[#0(pid 0)=155,236;#1(pid 1)=364,512]

- event ACTION\_MOVE[#0(pid 0)=157,240;#1(pid 1)=350,498]

- event ACTION\_MOVE[#0(pid 0)=158,245;#1(pid 1)=343,494]

10 event ACTION\_POINTER\_UP(pid 0)[#0(pid 0)=158,247;#1(pid 1)=336,484]

- event ACTION\_MOVE[#0(pid 1)=334,481]

- event ACTION\_MOVE[#0(pid 1)=328,472]

- event ACTION\_UP[#0(pid 1)=327,471]

Вот как можно объяснить эти события:

В строке 1 мы видим событие ACTION\_DOWN, поэтому пользователь должен был

нажать экран одним пальцем. Палец был расположен в координатах *x=135*,

*y=179*, что близко к верхнему левому углу экрана. Однако совершенно непо-

нятно, что он пытался сделать — касание или перетаскивание.

Далее, начиная со строки 2, поступили несколько событий ACTION\_MOVE, по-

казывающие, что пользователь немного перемещал палец в координатах, о ко-

торых сообщается в событии. (На самом деле довольно сложно прикоснуться

пальцем к экрану, но не переместить его, поэтому вы получите много таких

событий.) Исходя из количества и размера перемещений можно говорить, что

пользователь выполнил жест перетаскивания.

Следующее событие, ACTION\_POINTER\_DOWN, в строке 5 означает, что пользова-

тель нажал экран другим пальцем. pid 1 подразумевает, что было прикосновение

указателя с ID 1 (палец № 1). Палец 0 уже находится на экране, поэтому сейчас

мы отслеживаем два пальца, находящиеся на экране. В теории, Android API

поддерживает одновременно до 256 пальцев, но первый выпуск телефонов на

Android 2.x ограничен двумя1. Координаты обоих пальцев возвращаются как

часть события. Это выглядит похожим на то, что пользователь начал жест из-

менения масштаба двумя пальцами.

1 Хотя идея с 256 пальцами может выглядеть глупой за пределами штаб-квартиры «Людей

в черном», помните, что Android предназначен для широкого круга устройств, а не только

для телефонов. Если у вас есть экран размером со стол, с несколькими работающ ими за

ним людьми, вы легко получите больше касаний экрана, чем пальцев на руке.

11.4. Установки для трансформации изображения **207**

А вот то, что нас интересует. Дальше мы видим серию событий ACTION\_MOVE,

начинающуюся в строке 6. Сейчас у нас два пальца, перемещающиеся по экрану

не так, как раньше. Присмотревшись к координатам, вы увидите, что пальцы

сближаются, а это часть жеста уменьшения масштаба.

В строке 10 мы видим событие ACTION\_POINTER\_UP с pid 0. Это означает, что па-

лец номер 0 был поднят с экрана. Палец 1 все еще здесь. Конечно, это окончание

жеста изменения масштаба.

Мы видим еще пару событий ACTION\_MOVE в строке 11, показывающ их, что

оставшийся палец все еще немного перемещается. Сравнив это с более ранними

событиями перемещения, можно отметить, что здесь сообщается о другом ID

указателя. К несчастью, API обработки касаний настолько кишит ошибками,

что вам не следует всегда на него полагаться (см. врезку: «Внимание: мульти-

ошибки впереди»).

И наконец, в строке 13 мы получаем событие ACTION\_UP, показывающее, что

с экрана убрали последний палец.

Сейчас код для dumpEvent() должен стать чуть более осмысленным. Метод

getAction() возвращает совершенное действие (опускание пальца, поднятие или

перемещение). Младшие 8 бит действия — это код действия, следующие 8 бит —

это ID указателя (пальца), поэтому мы используем двоичное AND (*&*, *И*) и сдвиг

вправо (*>>*) для их разделения.

Затем мы вызываем метод getPointerCount() для того, чтобы увидеть, как много

позиций пальцев включено. getX() и getY() возвращают координаты *X* и *Y* соответственно.

Пальцы могут появляться на экране в любом порядке, поэтому мы

вызываем

getPointerId() для того, чтобы обнаружить, о каком именно пальце идет

речь.

Здесь рассмотрены сырые данные о событиях указателя. Сложность заключает-

ся в их интерпретации и в выполнении действий на основе этих данных.

11.4. Установки для трансформации

изображения

Для перемещения изображения и изменения его масштаба мы будем использовать

маленькие изящные инструменты класса ImageView, которые называются *матрич-*

*ными преобразованиями*. Используя матрицу, мы можем представить любой вид

преобразования, вращения или наклона, который хотим произвести с изображе-

нием. Мы уже включили эти возможности, задав android:scaleType=»matrix» в файле

res/layout/main.xml. В классе Touch нам нужно объявить две матрицы в качестве полей

(одна — для текущего значения, и одна — для исходного значения перед трансфор-

мацией). Мы используем их в методе onTouch() для преобразования изображения.

Нам также нужна переменная mode, которая сообщит, находимся ли мы внутри

жеста перемещения или изменения масштаба, и нам нужны переменные start, mid

и oldDist для контроля за изменением масштаба.

**208** Глава 11 • Мульти-тач

**Touchv1/src/org/example/touch/Touch.java**

public class Touch extends Activity implements OnTouchListener {

// Эти матрицы будут использованы для перемещения изображения

// и изменения его масштаба

Matrix matrix = new Matrix();

Matrix savedMatrix = new Matrix();

// Мы можем находиться в одном из этих 3 состояний

static final int NONE = 0;

static final int DRAG = 1;

static final int ZOOM = 2;

int mode = NONE;

// Сохраним некоторые данные для изменения масштаба

PointF start = new PointF();

PointF mid = new PointF();

float oldDist = 1f;

@Override

public boolean onTouch(View v, MotionEvent event) {

ImageView view = (ImageView) v;

// Запишем сведения о событии прикосновения в журнал

dumpEvent(event);

// Здесь обработаем событие...

switch (event.getAction() & MotionEvent.ACTION\_MASK) {

}

view.setImageMatrix(matrix);

return true; // Показываем, что событие было обработано

}

}

Переменная matrix будет вычислена внутри оператора switch, когда мы создадим

реализацию обработки жестов.

11.5. Реализация обработки жеста

перетаскивания

Жест перетаскивания начинается, когда первый палец касается экрана (ACTION\_

DOWN), и заканчивается, когда его убирают с экрана (ACTION\_UP или ACTION\_

POINTER\_UP).

**Touchv1/src/org/example/touch/Touch.java**

switch (event.getAction() & MotionEvent.ACTION\_MASK) {

case MotionEvent.ACTION\_DOWN:

savedMatrix.set(matrix);

start.set(event.getX(), event.getY());

Log.d(TAG, "mode=DRAG" );

mode = DRAG;

break;

case MotionEvent.ACTION\_UP:

case MotionEvent.ACTION\_POINTER\_UP:

11.6. Реализация обработки жеста изменения масштаба **209**

mode = NONE;

Log.d(TAG, "mode=NONE" );

break;

case MotionEvent.ACTION\_MOVE:

if (mode == DRAG) {

matrix.set(savedMatrix);

matrix.postTranslate(event.getX() - start.x,

event.getY() - start.y);

}

break;

}

Когда жест начинается, мы запоминаем текущее значение матрицы преобра-

зования и стартовую позицию указателя. Каждый раз, когда палец перемещается,

мы начинаем обрабатывать матрицу преобразования с ее исходными значениями

и вызываем метод postTranslate() для добавления вектора преобразования разницы

между текущей и стартовой позицией.

Запустите программу, и вы сможете перемещать изображение по экрану, ис-

пользуя палец. Чистая работа, правда?

11.6. Реализация обработки жеста изменения

масштаба

Жест изменения масштаба похож на предыдущий, за исключением того, что он

начинается, когда второй палец нажимает на экран (ACTION\_POINTER\_DOWN).

**Touchv1/src/org/example/touch/Touch.java**

case MotionEvent.ACTION\_POINTER\_DOWN:

oldDist = spacing(event);

Log.d(TAG, "oldDist=" + oldDist);

if (oldDist > 10f) {

savedMatrix.set(matrix);

midPoint(mid, event);

mode = ZOOM;

Log.d(TAG, "mode=ZOOM" );

}

break;

case MotionEvent.ACTION\_MOVE:

if (mode == DRAG) {

// ...

}

else if (mode == ZOOM) {

float newDist = spacing(event);

Log.d(TAG, "newDist=" + newDist);

if (newDist > 10f) {

matrix.set(savedMatrix);

float scale = newDist / oldDist;

**210** Глава 11 • Мульти-тач

matrix.postScale(scale, scale, mid.x, mid.y);

}

}

break;

Когда мы видим событие опускания на экран второго пальца, мы рассчитыва-

ем и запоминаем расстояние между двумя пальцами. В моем испытании Android

иногда сообщал мне (и делал это неправильно), что пальцы коснулись экрана

почти в одной и той же позиции. Поэтому я добавил проверку, отбрасывающую

события, когда расстояние между пальцами меньше, чем некоторое произвольное

число пикселей. Если оно больше, чем это число, мы запоминаем текущую матри-

цу преобразований, вычисляем середину расстояния между пальцами и начинаем

изменение масштаба.

Если поступают события перемещения в то время, как мы находимся в режиме

изменения масштаба, мы расс читываем расс тояние между пальц ами снова. Если

оно слишком мало, событие игнорируется; в противном случае мы восс танавливаем

матрицу преобразования и изменяем масштаб изображения вокруг центральной

точки.

Масштаб — это просто количественное отображение нового расстояния, делен-

ное на старое расстояние. Если новое расстояние больше (то есть пальцы стали

дальше друг от друга), тогда масштаб будет больше, чем 1, делая изображение

больше. Если оно уменьшается (пальцы сблизились), тогда масштаб будет меньше

единицы, делая изображение меньше. И конечно, если ничего не меняется, масштаб

равен 1, и изображение не трансформируется.

Сейчас определим методы spacing() и midPoint().

Расстояние между двумя точками

Для того чтобы найти, как далеко два пальца расположены друг от друга, мы для

начала создадим вектор (x, y), который представляет собой разницу меду двумя

точками. Затем мы используем формулу Евклидова расстояния для вычисления

промежутка1.

**Touchv1/src/org/example/touch/Touch.java**

private float spacing(MotionEvent event) {

float x = event.getX(0) - event.getX(1);

float y = event.getY(0) - event.getY(1);

return FloatMath.sqrt(x \* x + y \* y);

}

Порядок точек значения не имеет, так как возведение в квадрат любых чисел

даст положительный результат. Обратите внимание, что все вычисления выпол-

няются с использованием типа данных Java float. Хотя некоторые устройства на

Android могут не иметь аппаратного обеспечения для поддержки вычислений

с плавающей точкой, мы делаем подобные вычисления не настолько часто, чтобы

начать беспокоиться о производительности.

1 http://en.wikipedia.org/wiki/Euclidean\_distance

11.7. Вперед>> **211**

Середина расстояния между двумя точками

Нахождение точки, которая расположена точно в середине расстояния между двумя

точками, даже проще:

**Touchv1/src/org/example/touch/Touch.java**

private void midPoint(PointF point, MotionEvent event) {

float x = event.getX(0) + event.getX(1);

float y = event.getY(0) + event.getY(1);

point.set(x / 2, y / 2);

}

Все, что нам нужно, — это взять средние значения координат *X* и *Y*. Для того

чтобы избежать сборки мусора, которая может привести к подтормаживаниям

приложения, мы повторно используем существующий объект для хранения резуль-

татов, вместо того чтобы каждый раз создавать и возвращать новый.

Попытайтесь запустить программу на мобильном телефоне. Перетащите изо-

бражение одним пальцем и измените его масштаб, сводя и разводя два пальца. Для

наилучших результатов не сводите пальцы ближе чем на дюйм. Иначе вас начнут

преследовать ошибки в API, о которых я упоминал выше.

11.7. Вперед>>

В этой главе мы узнали, как использовать мульти-тач A�PII���������� для с\_о����������здания жес\_т�����а из-

менения масштаба при помощи двух пальцев. Есть отличный сайт, называемый

GestureWorks1, который описывает целую библиотеку жестов, реализованных на

платформе AdobeFlash. Вы найдете там идеи других жестов для использования их

в Android-программах, они помогут преодолеть ограничения поддержки мульти-

тач в Android.

Так как мульти-тач использует новые методы, которых не было до Android 2.0,

при попытке запустить этот пример на ранних версиях платформы программа не

будет работать, выдав ошибку «FForor�����ce��� c��lo���������������se�������������» (Принудительное закрытие).� ­К�\_с��ча-

стью, есть способ обойти подобные ограничения, как описано в разделе 13.3 «Разво-

рачивание программ ы на различных API Android ». Вам не удастся научить старый

телефон новым трюкам, но можно, по крайней мере, предотвратить его зависание.

В следующей главе мы исследуем расширения домашнего экрана, в том числе

интерактивные обои.