Привет, Android!

РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Эд Бурнет

ПИТЕР®

Эд Бурнет

Привет, Android! Разработка мобильных приложений

Перевели с английского А. Заика, А. Севостьянова

Заведующий редакцией	*А. Кривцов*
Руководитель проекта	*А. Кривцов*
Ведущий редактор	*Ю. Сергиенко*
Научный редактор	*А. Пасечник*
Художественный редактор	*Л. Адуевская*
Корректор	*Н. Викторова*
Верстка	*Е. Егорова*

ББК 32.973.2-018.2
УДК 004.451

Э. Бурнет

Б91 Привет, Android! Разработка мобильных приложений. — СПб.: Питер, 2012. — 256 с.: ил.

ISBN 978-5-459-01015-2

Операционная система Android установлена внутри миллионов сотовых телефонов, коммуникаторов, планшетных компьютеров, нетбуков и смартбуков. Она разработана Google и Open Handset Alliance на основе ядра Linux.

С помощью этой книги вы сможете создать собственное приложение для Android всего за несколько минут! Затем мы перейдем к более сложному примеру: программированию игры Android Sudoku. Шаг за шагом дописывая код игры, вы изучите основы программирования для Android.

Вы научитесь писать приложения для аудио- и видеоконтента, работать с графикой, используя 2D и 3D OpenGL, обрабатывать веб-страницы и веб-сервисы, хранить данные с SQLite.

Вы также узнаете, как продавать ваши приложения на Android Market.

Если вы любите писать программы больше, чем читать о них, то эта книга для вас.

ISBN 978-1934356562 англ.
ISBN 978-5-459-01015-2

ООО «Мир книг», 198206, Санкт-Петербург, Петергофское шоссе, 73, лит. А29.
Налоговая льгота — общероссийский классификатор продукции ОК 005-93, том 2; 95 3005 — литература учебная.
Подписано в печать 12.09.11. Формат 70x100/16. Усл. п. л. 20,640. Тираж 2000. Заказ 0000.
Отпечатано по технологии CtP в ОАО «Печатный двор» им. А. М. Горького.
197110, Санкт-Петербург, Чкаловский пр., 15.

Оглавление

ЧАСТЬ IV. СЛЕДУЮЩЕЕ ПОКОЛЕНИЕ

Отзывы читателей о книге «Привет, Android»

Изучайте разработку приложений для Android с помощью этого полного, но легкого в освоении введения в платформу Android. Каждая страница этой книги преподносит отлично приготовленный и богатый материал для разработчиков, впервые знакомящихся с Android. Очень скоро вы будете писать приложения для Android самостоятельно.

Марко Гарджента (Marco Gargenta),
CEO, maracana.com

Третье издание книги «Привет, Android» позволит вам в кратчайшие сроки освоить разработку приложений для Android, начиная с основных понятий и заканчивая публикацией ваших приложений на Android Market. Эд делится своими обширными знаниями в данной области и касается даже таких тем, информацию по которым найти весьма непросто. Среди них — использование технологии мульти-тач (multi-touch) и OpenGL. Это руководство обязательно нужно прочитать тем, кто начинает захватывающее путешествие в мир разработки приложений для Android.

Диего Торрес Милано (Diego Torres Milano),
эксперт по Android, блогер

Я полностью удовлетворен книгой «Привет, Android», с ее помощью я нашел верное направление, приведшее меня к выпуску моих первых двух приложений на Android Market.

Натан Рэпп (Nathan Rapp),
основатель KMBurrito Designs

«Привет, Android» — это не только первое знакомство, но и приглашение для новичков и профессионалов в мир разработки для платформы Android.

Майкл Мартин PMP (Michael Martin PMP),
основатель GoogleAndBlog и Mobile Martin

Благодарности

Мне хотелось бы поблагодарить множество людей, которые сделали возможным выход этой книги, в том числе — читателей предыдущих изданий за все их замечательные предложения; моего редактора, Сюзанну Пфэлзер (Susannah Pfalser) за ее внимание к деталям; Хавьера Колладо (Javier Collado), Мэрилин Харрет (Marilynn Huret) и Стефана Нотеберга (Staffan Nöteberg) за ценные комментарии; и особенно Лизу, Майкла и Кристофера за их бесконечное терпение и поддержку.

Предисловие

Android — это программное обеспечение с открытым кодом для мобильных телефонов, созданное компаниями Google и Open Handset Alliance. Android установлен на миллионах мобильных телефонов и других устройств, и это делает его основной платформой для разработчиков программного обеспечения. Являетесь ли вы любителем или профессиональным программистом, делаете ли вы свое дело ради удовольствия или ради денег, для вас настало время побольше узнать о разработках под Android. Эта книга поможет сделать первый шаг.

Что делает Android особенным?

На рынке уже существует множество мобильных платформ, в том числе Symbian, iPhone, Windows Mobile, BlackBerry, Java Mobile Edition, Linux Mobile (LiMo) и другие. Когда я рассказываю людям об Android, они часто задают один и тот же вопрос: «Зачем нам нужен другой мобильный стандарт? А особенно тот, в котором нет „ничего для себя!“?»

Хотя многие фишки Android появились раньше, он стал первой средой, которая совместила в себе следующие особенности:

- *По-настоящему открытая, бесплатная платформа разработки ПО, основанная на Linux и открытом коде.* Она нравится разработчикам сотовых телефонов, так как они могут использовать и модифицировать ее под свои нужды, не платя лицензионных отчислений. Разработчики любят ее, потому что знают, что эта платформа перспективна и не замкнута на единственного разработчика, который в любом момент может разориться или поглотиться другой компанией.
- *Архитектура, основанная на компонентах, несущая в себе идеи, распространенные в Интернете.* Различные части приложения могут быть использованы не так, как это было изначально задумано разработчиками. Вы можете даже заменить встроенные компоненты на собственные, улучшенные версии. Эта его особенность позволяет достичь нового уровня креативности в области разработки мобильных приложений.

- *Множество изначально встроенных возможностей*. Сервисы определения местоположения, основанные на GPS или алгоритмах триангуляции базовых станций, позволяют управлять пользовательскими приложениями в зависимости от физического расположения абонента. Полнофункциональная SQL-база данных позволяет программам, требующим подключения к Сети, работать с локальным хранилищем данных в промежутках между синхронизациями. Браузер и карту можно встроить непосредственно в приложения. Все эти встроенные возможности помогают увеличить функциональность программ и снизить затраты на разработку.
- *Автоматическое управление жизненным циклом приложения*. Программы изолированы друг от друга множеством уровней прав доступа, которые помогают обеспечить такой уровень стабильности системы, который был недоступен ранее для смартфонов. Конечный пользователь больше не беспокоится о том, активно ли приложение, не закрывает одни приложения для того, чтобы запустить другие. Android изначально оптимизирован для устройств с низким энергопотреблением и малым количеством памяти, чего нельзя сказать о выпущенных ранее платформах.
- *Высококачественные графика и звук*. Качественная, сглаженная двумерная (2D) векторная графика и анимация, вдохновленные идеями Flash, смешанные с возможностями 3D-ускорения OpenGL, предоставляют широкие возможности новым видам игровых и деловых приложений. В систему встроены кодеки для проигрывания наиболее распространенных аудио- и видеоформатов, в том числе форматов H.264 (AVC), MP3 и AAC.
- *Переносимость между широким диапазоном существующего и будущего аппаратного обеспечения*. Все программы для ваших мобильных устройств написаны на Java и исполняются на виртуальной машине Android Dalvik, а это значит, что ваш код можно переносить между устройствами, основанными на архитектурах ARM, x86 и другими. В систему включена поддержка множества способов ввода данных, таких как клавиатура, тач-скрин, трекбол. Пользовательский интерфейс может быть настроен под любое разрешение и любую ориентацию экрана.

Android предлагает вам по-новому взглянуть на взаимодействие мобильных приложений с пользователем и техническую базу, делающую это возможным. Но лучшая часть Android — это приложения, которые вы собираетесь писать для этой платформы. Эта книга поможет вам взять отличный старт.

Для кого предназначена эта книга?

Единственное требование к читателю этой книги — понимание базовых основ программирования на Java или на похожем объектно-ориентированном языке (в крайнем случае на C#). Вам не понадобится опыт разработки приложений для мобильных устройств. Напротив, будет лучше, если вы попытаетесь забыть подобный опыт. Android настолько отличается от всего остального, что лучше начинать работу с ним «с чистого листа».

Что вы найдете в этой книге?

Книга «Привет, Android» разделена на четыре части. Материалы книги расположены по принципу «от простого к сложному» или от более общих к специализированным аспектам Android.

Через несколько глав проходит сквозной пример: игра Android Sudoku. Постепенно добавляя в игру новые возможности, вы изучите множество сторон программирования под Android, в том числе пользовательский интерфейс, мультимедийные возможности и жизненный цикл программы на Android.

В части I мы начнем с введения в Android. Именно здесь вы узнаете, как установить эмулятор Android и как использовать интегрированную среду разработки (IDE) для того, чтобы написать вашу первую программу. Затем мы расскажем о жизненном цикле Android-приложений. Программирование для Android немного отличается от того, что вы делали раньше, поэтому убедитесь в том, что вы поняли это, прежде чем двигаться дальше.

Часть II предоставляет сведения о пользовательском интерфейсе Android, двумерной графике, мультимедийных компонентах и простом доступе к данным. Эти возможности будут использованы в большинстве программ, которые вы напишете.

Часть III представляет собой более глубокое погружение в платформу Android. Здесь вы узнаете о том, как соединить вашу программу с внешним миром, о сервисах, основанных на определении местоположения, встроенной базе данных SQLLite и трехмерной графике.

Часть IV завершает картину беседой об использовании продвинутых методов ввода данных, включающих мульти-тач и расширение стартового экрана виджетами и интерактивными обоями. Наконец, мы расскажем, как сделать ваше приложение совместимым с множеством Android-устройств и версий платформы и затем опубликуем его на Android Market.

В конце книги вы найдете приложение, которое иллюстрирует разницу между платформами Android и Java Standard Edition (SE), а также список литературы.

Что нового в третьем издании?

Третье издание этой книги обновлено для поддержки всех версий Android — от версии 1.5 до 2.2 и других. Здесь содержится краткий перечень новых возможностей, представленных в каждой из версий, и ссылки на разделы, которые раскрывают эти возможности.

Новое в версии Cupcake

В Android 1.5 (Cupcake) представлено множество улучшений платформы Android, в том числе поддержка экранной клавиатуры, видеозаписей, виджетов. Имеется более 1000 различий в Android API между версиями 1.1 и 1.5[1]. О виджетах речь пойдет в разделе 12.1 «Привет, виджет».

[1] http://d.android.com/sdk/api_diff/3/changes.html

Новое в версии Donut

В Android 1.6 (Donut) добавлена поддержка экранов высокой и низкой плотности размещения пикселей плюс множество менее значительных изменений, которые не затрагивают большинства разработчиков[1]. Вы можете узнать о том, как поддерживать эти различные форм-факторы устройств, в разделе 13.5 «Все экраны, большие и маленькие»

Новое в версии Eclair

В Android 2.0 (Eclair) добавлены: поддержка технологии мульти-тач, виртуальных клавиш, централизованного управления учетными записями, API-синхронизации; обработка подключения к док-станции; HTML5 и многое другое[2]. Версию 2.0 быстро заменил Android 2.1 (его тоже называют Eclair), который содержит все нововведения версии 2.0 плюс несколько исправлений[3]. Технология мульти-тач обсуждается в главе 11 «Мульти-тач».

Новое в версии Eclair MR1

В Android 2.1 (Eclair Maintenance Release 1) добавлены интерактивные обои, лучшая поддержка HTML5 и другие, менее значительные, улучшения[4]. Расширения стартового экрана, в том числе интерактивные обои и виджеты, представлены в главе 12 «Нет места лучше дома».

Новое в версии Froyo и в других версиях

Android 2.2 (Froyo) поддерживает установку приложений на внешний носитель (SD-карты), более быструю виртуальную машину Java, API OpenGL ES 2.0 и многое другое[5]. В разделе 13.6 «Установка на SD-карту» объясняется, как настроить вашу программу таким образом, чтобы ее можно было установить на внешний носитель, и описываются ситуации, к которых это следует и не следует делать.

Android 1.5 (или более новые версии) доступен сейчас на всех поставляемых Android-устройствах. Он установлен на все новые устройства, и, по заверениям Google, почти все более старые устройства обновлены до версии 1.5 или более поздней. Обратитесь к Android Device Dashboard[6] для того, чтобы узнать, какую долю рынка занимают устройства на различных версиях платформы. Версии 1.1 или более ранние в книге не рассматриваются.

Обратите внимание на то, что пока платформа на всех устройствах будет обновлена, это может занять некоторое время, если вообще произойдет. Так, глава

---

[1] http://d.android.com/sdk/api_diff/4/changes.html
[2] http://d.android.com/sdk/api_diff/5/changes.html
[3] http://d.android.com/sdk/api_diff/6/changes.html
[4] http://d.android.com/sdk/api_diff/7/changes.html
[5] http://d.android.com/sdk/api_diff/8/changes.html
[6] http://d.android.com/resources/dashboard/platform-versions.html

13 «Написав однажды, протестируй везде» рассказывает о том, как создавать программы, которые поддерживают множество версий платформы. Все примеры из этой книги протестированы на версиях платформы от 1.5 до 2.2.

Онлайн-ресурсы

На веб-сайте этой книги (http://pragprog.com/titles/eband3/hello-android) вы найдете следующее:

- полный исходный код всех программ, использованных в этой книге;
- форум, где вы можете напрямую связаться с автором и другими Android-разработчиками (будем надеяться, что форум окажется полным до краев!).

Вы можете скачать с сайта издательства «Питер» (www.piter.com) архив с дополнительными материалами к этой книге.

Используйте представленные исходные коды для разработки собственных приложений.

Вперед >>

Хотя многие авторы ожидают, что вы будете читать их книги не отрываясь, я знаю, что вы не собираетесь так делать. Вы захотите узнать ровно столько, чтобы начать работать над своим приложением, а затем, возможно, позднее вернетесь к прочитанному и узнаете что-то еще, что позволит вам завершить очередной этап работы. Поэтому я попытался предоставить вам небольшую помощь для того, чтобы вы не заблудились в обилии материала.

Каждая глава в этой книге завершается разделом «Вперед >>». Этот раздел содержит некоторые указания о том, куда вам следует идти дальше в том случае, если вам нужно читать книгу не по порядку. Также вы найдете там указания на другие источники информации, такие как книги и документация в Интернете, в том случае, если вы захотите узнать больше по интересующей вас теме.

Итак, чего же вы ждете? Следующая глава — глава 1 «Быстрый старт» — окунет вас прямо в глубины разработки вашей первой программы для Android. Глава 2 «Ключевые концепции» сделает шаг назад и ознакомит вас с основными концепциями и философией Android, а глава 3 «Разработка пользовательского интерфейса» углубится в пользовательский интерфейс, который является основной частью большинства программ на Android.

Ваша главная цель заключается в том, чтобы сделать ваши приложения доступными для продажи или бесплатной загрузки на Android Market. Когда вы будете к этому готовы, глава 14 «Публикация на Android Market» покажет вам, как сделать этот заключительный шаг.

От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу электронной почты comp@piter.com (издательство «Питер», компьютерная редакция).

На веб-сайте издательства (www.piter.com) вы можете скачать архив с дополнительными материалами к этой книге.

Мы будем рады узнать ваше мнение!

На веб-сайте издательства http://www.piter.com вы найдете подробную информацию о наших книгах.

I

Введение в Android

1 Быстрый старт

Android, опираясь на повсеместную распространенность мобильных телефонов, сочетает в себе преимущества программного обеспечения с открытым исходным кодом, корпоративную поддержку Google и других членов Open Handset Alliance, таких, как Motorola, HTC, Verison и AT&T. Образно говоря, это мобильная платформа, которую вы не имеете права не изучить.

К счастью, начать разработку приложений для Android довольно просто. Вам даже не нужен телефон на Android — нужен лишь компьютер, где вы можете установить Android SDK и эмулятор сотового телефона.

В этой главе я покажу вам, как установить все необходимые инструменты разработчика, после чего мы начнем ими пользоваться для того, чтобы создать работающее приложение — традиционное «Hello, World» для платформы Android.

1.1. Установка инструментов

Набор инструментов разработчика для Android (SDK — Software Development Kit) работает под Windows, Linux и Mac OS X. Приложения, которые вы создадите, естественно, можно будет установить на любое Android-устройство.

Прежде чем вы начнете создавать текст программы, вы должны установить на свой ПК платформу Java, IDE (Integrated Development Environment — Интегрированную среду разработки) и Android SDK.

Java 5.0+

Для начала вам понадобится копия платформы Java. Все инструменты разработки под Android нуждаются в ней, и программы, которые вы пишете, будут использовать язык Java. Нам нужен JDK 5 или 6.

Недостаточно иметь лишь среду времени выполнения (JRE — Java Runtime Environment), вам понадобится полный комплект для разработчика. Я рекомендую скачать последнюю версию Sun JDK SE 6.0 с сайта Sun[1]. Практика показывает,

[1] http://java.sun.com/javase/downloads

что 32-битная версия предпочтительнее (см. врезку «32 бита против 64 битов»). Пользователи Mac OS X должны выбрать последнюю версию Mac OS X и JDK с веб-сайта Apple.

Чтобы удостовериться в том, что у вас есть нужная версия, запустите команду java–version из окна командной строки. Вот что получилось у меня, когда я запустил ее:

```
C:\> java -version
java version "1.6.0_14"
Java(TM) SE Runtime Environment (build 1.6.0_14-b08)
Java HotSpot(TM) Client VM (build 14.0-b16, mixed mode, sharing)
```

Вы должны увидеть что-то подобное с версией «1.6. ...» или более поздней.

Eclipse

Далее следует установить среду разработки для Java, если у вас она еще не установлена. Я рекомендую Eclipse, так как она, во-первых, бесплатная, а во-вторых, используется и поддерживается разработчиками Google, которые создали Android.

Минимальная рекомендуемая версия Eclipse — 3.3.1, но вы всегда должны использовать самые свежие версии программных продуктов. Перейдите к странице загрузки Eclipse[1], выберите Eclipse IDE for Java Developers. Обратите внимание на то, что вам нужно больше, чем просто стандартная, «классическая» платформа Eclipse SDK. Загрузите установочный пакет во временную папку, распакуйте его (обычно хватает двойного щелчка на имени файла) и переместите все распакованные файлы в отдельную папку (например, в Windows это C:\Eclipse, в Mac OS X это /Applications/Eclipse).

Если вы не хотите пользоваться Eclipse (всегда найдется кто-то, отличающийся от других), обратитесь за поддержкой других IDE, таких как NetBeans и JetBrains IDEA, к соответствующим сообществам. Или, если вы действительно являетесь ныне почти вымершим программистом-динозавром, то можете полностью забыть про IDE и пользоваться инструментами командной строки[2]. В этой книге мы будем пользоваться Eclipse. Если вы выбрали что-то другое, вам придется делать поправки на особенности другой интегрированной среды.

Android SDK Starter Package

Начиная с версии 2.0 Android SDK разбит на две части: SDK Starter Package и SDK Components. Для начала воспользуйтесь веб-браузером, чтобы загрузить Starter Package. Страница загрузки Android[3] содержит пакеты для Windows, Mac OS X и Linux. После загрузки выбранного пакета распакуйте .zip-файл в папку для хранения временных файлов.

---

[1] http://www.eclipse.org/downloads

[2] Обратитесь к странице http://d.android.com/guide/developing/tools для получения документации по инструментам командной строки

[3] http://d.android.com/sdk

32 БИТА ПРОТИВ 64 БИТ

Если вы пользуетесь 64-битной версией Windows, вы можете захотеть установить 64-битную версию Java Development Kit вместо 32-битной версии. Старые версии Eclipse имели проблемы при работе с 64-битным JDK, которые исправлены в Eclipse 3.6 («Helios»)[1]. Однако некоторые браузеры все еще нуждаются в 32-битных версиях Java-плагинов. За исключением тех случаев, когда вам действительно нужна 64-битная Java, лучше использовать 32-битные версии и Java и Eclipse.

По умолчанию SDK распаковывается в папку с именем android-sdk-[операционная система]_[платформа]. Например, папка с 32-разрядной SDK для Windows будет иметь имя android-sdk-windows_x86. Переместите эту папку в удобное место, например в папку C:\Google или /Applications/Google. Запомните полный путь папке с SDK, чтобы иметь возможность обратиться к ней позже из Eclipse как к *установочному каталогу (install directory)* SDK.

Программы, необходимые для работы Eclipse или SDK, не нуждаются в специальной установке, но я рекомендую вам добавить пути к папкам tools и platform-tools в переменную PATH.

Android SDK Components

Далее запустите программу Setup SDK. В Windows запустите файл SDK Setup.exe или SDK Manager.exe. В Linux или Mac OS X запустите программу tools/android, выберите Available Packages, установите флажки напротив каждого пакета и нажмите Install Selected.

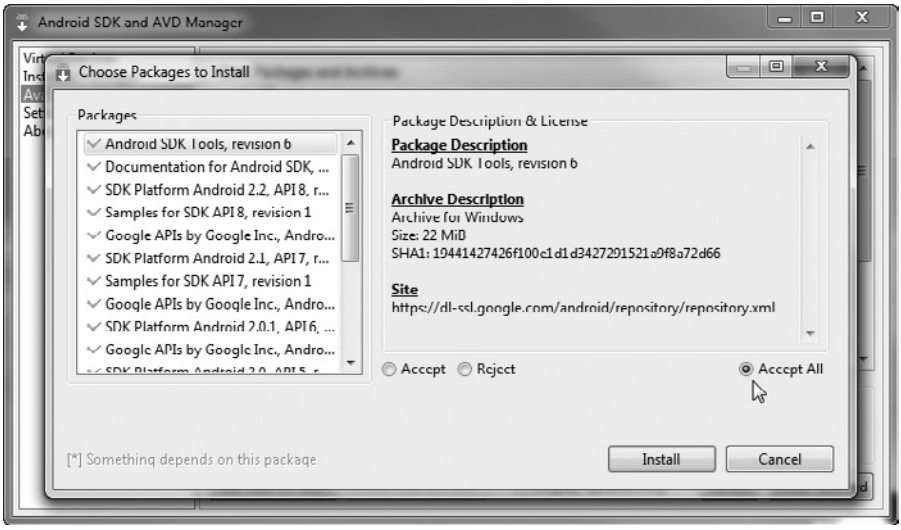

Рис. 1.1. Установка Android SDK Components

[1] https://bugs.eclipse.org/bugs/show_bug.cgi?id=293969

Программа Setup отобразит список доступных компонентов, включая документацию, платформы, дополнительные библиотеки, USB-драйверы (см. рис. 1.1). Выберите команду Accept All и щелкните на кнопке Install. Все перечисленные компоненты будут загружены и установлены в установочную папку SDK. Обратите внимание на то, что на установку может уйти довольно много времени. Для того чтобы ускорить этот процесс, вы можете включить или отключить отдельные компоненты вместо того, чтобы устанавливать их все.

Если вы столкнулись с ошибкой HTTP SSL, отмените загрузку, выберите команду Settings в основном окне SDK и AVD Manager. Выберите параметр Force https:// sources to be fetched using HTTP:// и щелкните на кнопке Save&Apply. Выйдите из программы установки и запустите ее снова.

Следующий шаг — запуск Eclipse и ее настройка.

Плагины для Eclipse

С целью облегчения процесса разработки Google выпустил плагин для Eclipse, который называется *Android Development Toolkit* (ADT). Чтобы установить плагин, выполните следующие шаги (обратите внимание на то, что эти указания даны для Eclipse 3.5, — в других версиях меню и команды могут отличаться).

1. Запустите Eclipse (eclipse.exe в Windows или eclipse в Mac OS X или Linux). Если вас спросят о рабочей папке (workspace), просто согласитесь с параметрами по умолчанию и нажмите OK.
2. Выберите команду меню Help ▸ Install New Software. Посмотрите врезку *Вопрос/ответ*, если у вас возникнет ошибка подключения.
3. Щелкните на ссылке Available Software Sites в появившемся диалоговом окне.

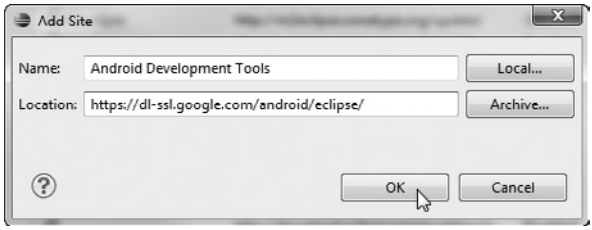

Рис. 1.2. Установка Android Development Toolkit

4. Нажмите кнопку Add... (Добавить).
5. Введите адрес сайта обновлений *Android Development Tools*: https://dl-ssl.google.com/android/eclipse/. Заполненное диалоговое окно должно выглядеть так, как показано на рис. 1.2.
6. Щелкните на кнопке OK, чтобы вернуться к списку сайтов, затем щелкните на кнопке Test Connection, чтобы проверить адрес сайта, который вы только что ввели. Если у вас возникли проблемы с этим сайтом, попробуйте использовать протокол HTTP вместо HTTPS. Когда вы убедитесь в правильности адреса,

нажмите OK еще раз, чтобы вернуться в диалоговое окно Install New Software (Установка нового программного обеспечения).

7. Введите android в поле Work With и нажмите Return. В списке ниже должна появиться запись Developer Tools.
8. Установите флажок около Developer Tools и нажмите Next. Если вы получили сообщение об ошибке на данном этапе, вы, возможно, установили не ту версию Eclipse. Я настоятельно рекомендую пользоваться либо преднастроенной версией Eclipse IDE for Java Developers, либо Eclipse IDE for Java EE Development версии 3.5 или более новой.

   Если у вас другой дистрибутив Eclipse, то для работы с редакторами Android вам понадобится установить плагин Web Standard Tools (WST) со всеми его предустановками.

   Обратитесь к веб-страничке[1] Web Tools Platform за дополнительными инструкциями и ссылками для загрузки. Все это уже встроено в рекомендованные пакеты, упомянутые ранее.

ВОПРОС/ОТВЕТ

Появляется сообщение: «Connection Error» («Ошибка соединения»). И что мне делать?

Если у вас возникла ошибка соединения, причина, скорее всего, кроется в брандмауэре, который установлен вашим системным администратором. Для того чтобы обойти брандмауэр, нужно ввести в конфигурационную информацию Eclipse адрес вашего прокси-сервера. Это тот же самый прокси-сервер, который используется для вашего веб-браузера, но, к несчастью, Eclipse не настолько продвинута, чтобы самостоятельно извлекать нужные параметры из его настроек.

Чтобы ввести в Eclipse данные о прокси-сервере, выполните команду Window ▸ Preferences ▸ General ▸ Network Connections (Окно ▸ Установки ▸ Общие ▸ Сетевые соединения) или Eclipse ▸ Preferences (Eclipse ▸ Установки) на Mac OS X, включите Manual proxy configuration (Ручная настройка прокси-сервера), введите имя сервера и номер порта и нажмите OK. Если вы не видите этого параметра, возможно, вы запустили устаревшую версию Eclipse. Попытайтесь поискать в меню Preferences ▸ Install/Update (Установки ▸ Инсталляция/Обновление) или поищите настройки по ключевому слову *proxy* (прокси-сервер).

9. Посмотрите список компонентов, которые будут установлены, нажмите Next, примите лицензионные соглашения и затем нажмите Finish, чтобы начать процесс загрузки и установки.
10. После успешного завершения установки перезапустите Eclipse.
11. Возможно, при следующем запуске Eclipse выведет несколько сообщений об ошибках, так как вы еще не сообщили среде разработке, где расположен Android SDK. Выберите команду меню Window ▸ Preference ▸ Android (Eclipse ▸ Preferences на Mac OS) и введите туда путь к папке, в которую установлен SDK, — тот путь, который вы запомнили ранее. Нажмите OK.

[1] http://www.eclipse.org/webtools

Вот и все! К счастью, вы должны проделать все это только один раз (или, по крайней мере, не чаще, чем появляется новая версия Eclipse или ADT). Сейчас, когда все необходимое установлено, настало время написать вашу первую программу.

1.2. Создание первой программы

ADT поставляется со встроенной демонстрационной программой, или шаблоном, поэтому мы собираемся использовать его для создания буквально за несколько секунд простой программы «Hello, Android». Приготовьте секундомер. Все готово? Начали!

Выполните команду меню File ▸ New ▸ Project, чтобы открыть диалоговое окно New Project. Далее выберите Android ▸ Android Project и нажмите Next.

Введите в окно нового проекта следующую информацию:

```
Project name: HelloAndroid
Build Target: Android 2.2
Application name: Hello, Android
Package name: org.example.hello
Create Activity: Hello
Min SDK Version: 8
```

То, что получилось в итоге, должно выглядеть так, как показано на рис. 1.3.

Нажмите на кнопку Finish. Плагин Android создаст проект и заполнит некоторые строки по умолчанию. Когда проект будет готов к исполнению, Eclipse скомпонует и упакует его. Если вы столкнулись с ошибкой отсутствия папок с исходниками, выберите команду Project ▸ Clean, чтобы исправить ее.

Итак, система позаботилась о создании программы; сейчас все готово к тому, чтобы попробовать ее запустить. Для начала мы запустим ее на эмуляторе Android.

1.3. Запуск программы на эмуляторе

Для того чтобы запустить вашу Android-программу, перейдите в окно Package Explorer, щелкните правой кнопкой мыши на проекте HelloAndroid и выберите команду Run As ▸ Android Application. Если вы следовали за нами при работе над этим примером, то, скорее всего, увидите диалоговое окно с описанием ошибки, похожее на рис. 1.4. Оно указывает на то, что мы не сообщили эмулятору, какой именно телефон эмулировать.

Создание AVD

Вам нужно создать Android Virtual Device (AVD — виртуальное устройство Android), используя либо Eclipse, либо команду *android avd*[1]. Проще использовать Eclipse, поэтому выберите Yes в диалоговом окне Android AVD Error, чтобы открыть окно AVD Manager. Позже вы можете открыть менеджер снова, выбрав команду Window ▸ Android SDK and AVD Manager.

[1] http://d.android.com/guide/developing/tools/avd.html

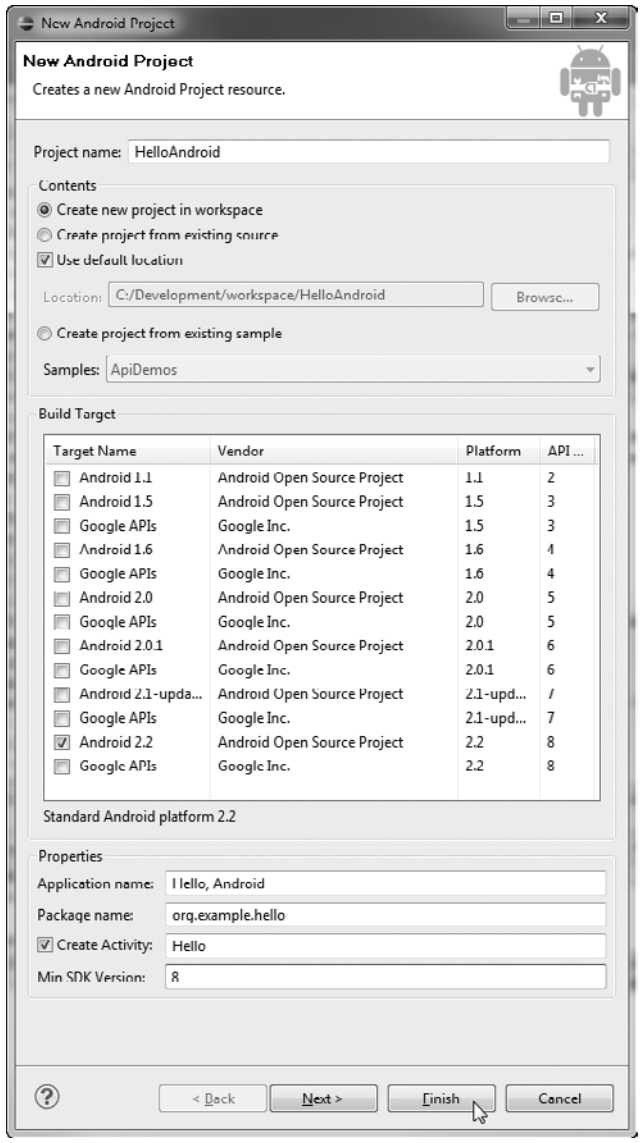

Рис. 1.3. Новый проект для платформы Android

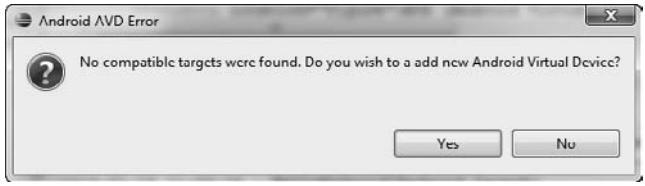

Рис. 1.4. Отсутствие Android Virtual Device (AVD — Виртуального устройства Android)

ПОДДЕРЖАНИЕ ПЛАГИНОВ В ХОРОШЕМ СОСТОЯНИИ

Плагины Android Eclipse находятся в постоянной разработке, поэтому изменения в них происходят гораздо чаще, чем в Android SDK. Версии, которые вы загрузили, могут отличаться от тех, которые я использовал, когда писал эту книгу, и они могут содержать некоторые, скажем так, отличительные особенности. Я рекомендую вам проверять сайт с плагинами ежемесячно для того, чтобы быть в курсе появления любых новых разработок и исправлений ошибок.

Щелкните на кнопке New... и заполните поля для нового AVD следующим образом:

```
Name: em22
Target: Android 2.2 - API Level 8
SDCard: 64
Skin: Default (HVGA)
```

Этим вы сообщаете Eclipse, что нужно создать стандартное устройство, именуемое *em22*, которое имеет прошивку Android 2.2 (Froyo) и дисплей с разрешением half-VGA (320×480). Будет также выделено место под 64-мегабайтную виртуальную SD-карту.

Закончив установку, вы должны увидеть что-то наподобие рис. 1.6. Так как с тех пор, как написана эта книга, инструменты SDK были обновлены, картинка на вашем экране может выглядеть немного по-другому.

Щелкните на кнопке Create AVD, чтобы создать виртуальное устройство. Через несколько секунд вы увидите сообщение о том, что устройство создано. Нажмите OK, выберите AVD, затем нажмите Start... и затем Launch, чтобы запустить эмулятор. По окончании процесса закройте AVD Manager.

CUPCAKE VS. DONUT VS. ECLAIR VS. FROYO VS. GINGERBREAD

Версии Android, которые выполняются на вашем эмуляторе (или на обычном телефоне), должны быть совместимы с целевой платформой (build target) вашей программы. Например, если вы попробуете запустить приложение, совместимое с Android 2.2 (FroYo) или Android 2.3 (Gingerbread) на телефоне с Android 1.5 (Cupcake), это приложение не будет работать, так как телефоны с Android 1.5 могут исполнять приложения либо для версии 1.5, либо для более ранних версий платформы. Телефоны с прошивкой Android 2.3, напротив, могут работать с приложениями, созданными для версий платформы 2.3, 2.2, 2.1, 2.0.1, 2.0, 1.6, 1.5 и более ранних. И пройдет еще какое-то время, прежде чем на основной массе телефонов система будет обновлена до более новой версии.

Тогда почему бы просто не использовать в качестве целевой платформы Android 1.5? К несчастью, приложения, написанные для 1.5, не всегда отображаются корректно на больших и маленьких экранах, которыми оборудованы телефоны, несущие на борту Android 1.6. Но есть простой способ сделать ваши программы совместимыми со всеми версиями Android. Смотрите главу 13 «Написав однажды, протестируй везде» для получения сведений по этому вопросу.

ЭКОНОМИЯ ВРЕМЕНИ

Запуск эмулятора занимает немало времени и ресурсов. Подумайте об этом в таком ключе: когда вы включаете сотовый телефон, он загружается, как и любая вычислительная система. Закрытие эмулятора можно сравнить с выключением телефона или с вытаскиванием из него батареек. Поэтому не выключайте его ради экономии своего времени!

Оставьте окно эмулятора открытым, пока открыто окно Eclipse. В следующий раз, когда вы запустите программу на Android, Eclipse поймет, что эмулятор уже готов, и просто отправит на него новую программу для запуска.

Рис. 1.5. Запуск программы «Hello, Android»

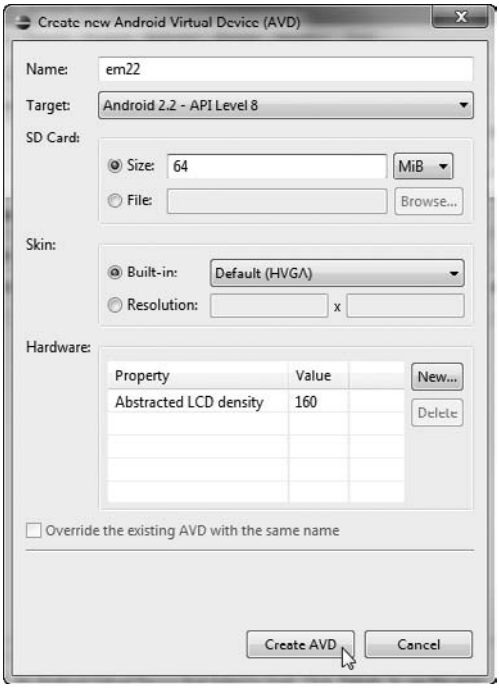

Рис. 1.6. Создание AVD в Eclipse

Попробуем снова

После того как у вас появилось правильно настроенное AVD, появится окно эмулятора Android, загрузится ОС Android. Если это происходит впервые, весь процесс может занять минуту или две, но будьте терпеливы. Возможно, вам понадобится щелкнуть правой кнопкой мыши на значке проекта и выбрать команду Run As ▸ Android Application снова. Если вы увидите сообщение об ошибке, в котором говорится, что приложение не отвечает, выберите команду, которая позволяет продолжить ожидание. Если вы увидите хранитель экрана, проведите по нему мышью в указанном направлении для разблокирования устройства.

Eclipse отправит копию вашей программы на эмулятор для исполнения на нем. На экране вашего компьютера появляется экран приложения, а это значит, что ваша программа «Hello, Android» уже работает (рис. 1.5). Вот и все! Примите поздравления с успешным написанием вашей первой программы для Android.

1.4. Запуск на реальном телефоне

Запуск Android-программ на физических устройствах, таких как Droid или Nexus One, в процессе разработки почти ничем не отличается от запуска их на эмуляторе. Вам следует включить USB-отладку на телефоне (запустив приложение Settings и выбрав Applications ▸ Development ▸ USB Debugging), установить драйвер устройства Android USB (только для Windows) и подключить телефон к вашему компьютеру, используя USB-кабель, который поставляется в комплекте с телефоном[1].

Закройте окно эмулятора, если оно все еще открыто. Пока к компьютеру подключен телефон, Eclipse будет загружать и запускать приложения на телефоне, а не на эмуляторе. Однако вы можете переключаться между эмулятором и телефоном, для этого отредактируйте ваши настройки запуска (они находятся в меню Run ▸ Run Configuraions), выберите закладку Target и измените параметр Target selection mode в соответствии с вашими предпочтениями. Установка режима Manual приведет к тому, что Eclipse будет предлагать вам выбор устройства каждый раз, когда вы запускаете программу. Если ваш телефон не появляется в списке устройств, это обычно указывает на проблемы с USB-драйвером или с минимальным уровнем SDK, необходимым для запуска программы, и его реальным значением. Вам может понадобиться внести изменения в файл AdobeManifest.xml, если на телефоне не установлена последняя версия Android (см. раздел 13.2, «Компоновка для множества версий Android»).

Если вы решитесь предложить написанное вами приложение другим пользователям, вы должны будете выполнить еще несколько шагов. Глава 14, «Публикация на Android Market», детально раскрывает этот вопрос.

---

[1] Обратитесь к странице http://d.android.com/guide/developing/device.html для того, чтобы найти свежие драйверы устройств и руководства по установке.

1.5. Вперед >>

Благодаря плагинам для Eclipse создание каркасной программы для Android занимает всего несколько секунд. В главе 3 «Разработка пользовательского интерфейса» мы начнем создавать оболочку для каркасного приложения, превращая ее в реальную программу — игру Sudoku. Этот пример будет использован на протяжении нескольких глав для демонстрации возможностей API Android.

Но прежде чем двигаться дальше, потратьте несколько минут на чтение главы 2 «Ключевые концепции»). Только после того, как вы разберетесь с основными идеями, такими как *деятельности* (activities) и *жизненные циклы* (life cycles), вы можете продолжить работу.

Хотя использование Eclipse для создания Android-программ вовсе не обязательно, я настоятельно рекомендую именно эту среду разработки.

2 Ключевые концепции

После начального знакомства с Android давайте посмотрим, как он работает. Некоторые концепции Android могут показаться знакомыми — такие, как ядро Linux (Linux kernel), OpenGL и база данных SQL. Другие вы увидите впервые — такие, как концепция жизненного цикла приложений Android.

Разберитесь в этих ключевых концепциях для того, чтобы писать корректно работающие программы для Android, а поэтому, если вы решили прочитать всего одну главу из этой книги, прочитай именно эту.

2.1. Общая картина

Для начала посмотрим на общую архитектуру системы — ключевые уровни и компоненты, которые составляют стек ПО с открытым исходным кодом Android. На рис. 2.1 вы видите Android «с высоты птичьего полета», далее мы рассмотрим его подробнее.

Каждый уровень использует сервисы, работу которых обеспечивают нижележащие уровни. По стеку Android мы будем подниматься, начиная с самых нижних ступеней.

Ядро Linux

Android построен на прочном и надежном фундаменте: на ядре Linux. Созданную Линусом Торвальдсом (Linus Torvalds) в 1991 году Linux сегодня можно найти практически везде, в любых устройствах — от наручных часов до суперкомпьютеров. В нашем случае Linux обеспечивает уровень абстракций между оборудованием и остальными частями стека Android.

С точки зрения внутренней архитектуры Android использует Linux для управления памятью, процессами, сетевым взаимодействием и другими возможностями операционной системы. Как жители многоэтажного дома не видят фундамент дома, так и пользователи телефонов на Android никогда не увидят Linux, а ваши программы никогда не будут обращаться к службам Linux напрямую. Однако вы, как разработчик, должны знать об этом.

Некоторые утилиты, которые понадобятся при разработке, взаимодействуют с Linux. Например, команда[1] ddb shell запускает командный интерпретатор Linux, в котором осуществляется ввод других команд для выполнения на устройстве. Отсюда производится проверка файловой системы Linux, просматриваются активные процессы и так далее, в зависимости от ограничений безопасности.

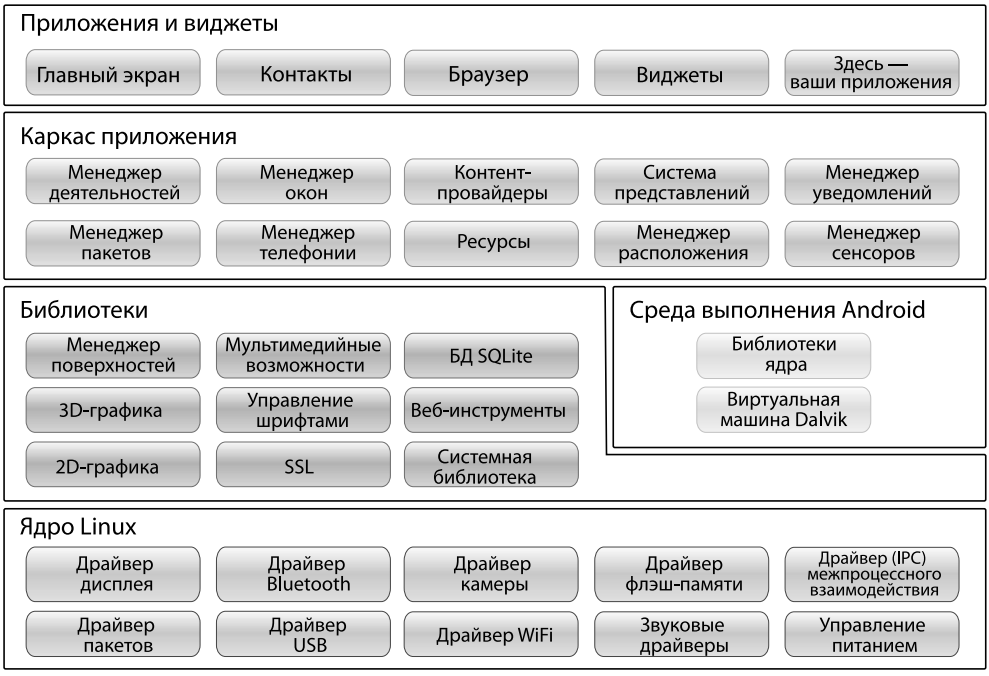

Рис. 2.1. Системная архитектура Android

Библиотеки

Уровень, следующий за уровнем ядра, содержит исходные библиотеки Android. Эти разделяемые библиотеки написаны на C или на C++, скомпилированы для конкретной аппаратной архитектуры и предустановлены на устройство разработчиком телефона.

Рассмотрим некоторые наиболее важные исходные библиотеки:

- *Менеджер поверхностей*. Android использует композитный менеджер окон, похожий на Vista или Compiz, но более простой. Вместо того чтобы выводить графические данные непосредственно в буфер экрана, команды отображения графики формируют закадровые битовые массивы, которые затем объединяются с другими массивами для того, чтобы сформировать изображение, которое видит пользователь. Это позволяет системе создавать различные интересные эффекты, например полупрозрачные окна и градиентные переходы.

[1] http://d.android.com/guide/developing/tools/adb.html

- *2D и 3D графика*: В Android двух- и трехмерные графические элементы комбинируются в единый пользовательский интерфейс. Библиотека будет использовать возможности аппаратного 3D-ускорения, если устройство ими оснащено, или быстрый программный рендеринг, если нет. Смотрите главу 4 «Введение в 2D графику» и главу 10 «3D графика в OpenGL».
- *Медиа-кодеки*. Android может проигрывать видеоролики и фильмы, записывать и воспроизводить аудиофрагменты в различных форматах, в том числе AAC, AVC (H.264), H.263, MP3, и MPEG4. В главе 5 «Мультимедиа» приведен интересный пример.
- *База данных SQL*. Android имеет «легковесную» встраиваемую реляционную базу данных[1] SQLite, эта же база данных используется в Firefox и в Apple iPhone[2]. Используйте этот механизм для постоянного хранения данных ваших приложений. Смотрите главу 9 «Работа с SQL» — там приведен пример.
- *Браузер*. Для быстрого отображения HTML-контента Android использует библиотеку WebKit[3]. Тот же механизм используется в браузере Google Chrome, браузере Apple Safari, в Apple iPhone и платформе Nokia S60. Обратитесь к главе 7 «Объединенный мир» за примером.

Эти библиотеки не являются отдельными приложениями. Они существуют только для того, чтобы их могли вызывать высокоуровневые программы. Начиная с версии 1.5 Android позволяет писать и внедрять свои собственные библиотеки, используя Native Development Toolkit (NTK). Разработка исходных библиотек выходит за рамки данной книги, но если вы заинтересовались, почитайте об этом в Интернете[4].

Среда выполнения Android

Среда выполнения Android также находится над ядром и включает в себя *виртуальную машину Dalvik* и *библиотеки ядра Java*.

Виртуальная машина (VM) Dalvik — это виртуальная машина Java в исполнении Google, оптимизированная для мобильных устройств. Весь код, который вы создаете для Android, пишется на Java и выполняется внутри виртуальной машины. Dalvik имеет следующие отличия от обычной Java-машины:

- Dalvik VM запускает файлы .dex, которые конвертируются при компиляции из стандартных файлов .class и .jar. Файлы .dex более компактны и эффективны, чем файлы классов, что является важным соображением, если принять во внимание ограничения памяти и энергопотребления устройств, для которых предназначен Android.
- Библиотеки ядра Java, которые поставляются с Android, отличаются и от библиотек Java Standard Edition (Java SE), и от библиотек Java Mobile Edition

[1] http://www.sqlite.org

[2] На сайте http://www.zdnet.com/blog/burnette/iphone-vs-android-development-day-1/682 вы найдете материалы по сравнению процессов разработки для iPhone и Android.

[3] http://webkit.org

[4] http://d.android.com/sdk/ndk

(Java ME). Однако они очень похожи. В приложении А приводится сравнение Android и стандартных библиотек Java.

ВОПРОС/ОТВЕТ

Что такое Dalvik?

Dalvik — это виртуальная машина (VM), разработанная и написанная Дэном Бернштейном (Dan Bornstein) из Google. Ваш код компилируется в машинно-независимые инструкции, называемые *байт-кодом (bytecodes)*, который затем исполняется на мобильном устройстве с помощью Dalvik VM.

Хотя формат байт-кода несколько специфичен, Dalvik — это виртуальная машина Java, оптимизированная для небольшого количества памяти. Это позволяет множеству экземпляров VM запускаться одновременно и пользоваться всеми преимуществами родительской операционной системы (Linux) в плане безопасности и изоляции процессов.

Бернштейн назвал Dalvik в память о рыбацкой деревне в Исландии, где некогда жили его предки.

Каркас приложений

Уровень каркаса приложений находится над библиотеками и средой выполнения. Он обеспечивает высокоуровневые строительные блоки, которые вы будете использовать для создания приложений. Фреймворк поставляется предустановленным вместе с Android, но при необходимости вы можете расширять его с помощью собственных компонентов.

Наиболее важные части фреймворка:

- *Менеджер деятельностей*: контролирует жизненный цикл приложения (см. раздел 2.2 «Оно живое!») и поддерживает возможности пользовательской навигации по стеку деятельностей.
- *Контент-провайдеры*: эти объекты включают в себя данные, которые нужно разделить между приложениями, такие как, например, контактные сведения. Смотрите раздел 2.3, «Контент-провайдеры».
- *Менеджер ресурсов*. Ресурсы — это все, что поставляется вместе с вашей программой, но не является кодом (см. раздел 2.4 «Использование ресурсов»).
- *Менеджер расположения*. Телефон на Android всегда знает, где он находится (см. главу 8 «Определение местоположения и использование сенсоров»).
- *Менеджер уведомлений*. Здесь весьма ненавязчиво представляются события, такие как поступление сообщения, напоминание о встрече, информация о приближении какого-либо события, уведомление о вторжении инопланетян и многое другое.

ПРИМЕЧАНИЕ

Включать и расширять

Одно из уникальных и мощных качеств Android заключается в том, что все приложения «играют на одном поле». Я имею в виду то, что системные приложения используют те же самые публичные API, которые используете вы. При желании вы можете попросить Android заменить стандартные приложения на ваши.

Приложения и виджеты

Самый высокий уровень в диаграмме архитектуры Android — это уровень приложений и виджетов. Это верхушка айсберга Android. Конечный пользователь видит только эти программы, не подозревая о существовании подводной части айсберга. Но как разработчику для Android вам следует об этом знать.

Приложения — это программы, которые могут занимать весь экран и взаимодействовать с пользователем. С другой стороны, виджеты (их иногда называют *гаджетами*) занимают лишь небольшой участок рабочего экрана.

Основная часть этой книги посвящена разработке приложений, так как именно их написанием вы и будете заниматься. Разработке виджетов посвящена глава 12, «Нет места лучше дома».

При покупке телефона с Android пользователь получает множество предустановленных стандартных системных приложений, в том числе:

- программу для набора номеров;
- почтовый клиент;
- список контактов;
- веб-браузер;
- Android Market.

Используя Android Market, пользователь может загружать новые программы для запуска на своем телефоне. И именно в эту область вы сейчас входите. Закончив читать эту книгу, вы сможете написать свое собственное Killer application — захватчик рынка ПО для Android.

Сейчас давайте подробнее рассмотрим жизненный цикл приложений Android. Он немного отличается от жизненного цикла приложений в большинстве систем.

2.2. Оно живое!

На обычном рабочем столе Linux или Windows вы запускаете множество приложений и просматриваете результаты их работы в отдельных окнах. Одно из рабочих окон «активировано», то есть владеет фокусом ввода, однако все программы равноправны между собой. По своему желанию вы можете переключаться между ними и перемещать их, для того чтобы видеть свои действия и закрывать программы, которые не нужны.

Android работает не так.

В Android есть приложение переднего плана, активное приложение, которое обычно занимает весь экран, кроме строки состояния. Когда пользователь включает свой телефон, первое приложение, которое он видит, — это программа Home (Домашний экран), рис. 2.2.

Когда пользователь запускает программу, Android начинает ее исполнение и делает ее активной. Из этого приложения пользователь может вызвать другое приложение или другой экран в том же самом приложении, и так далее. Все эти программы и экраны записываются в *стек приложений* (application stack) системным

Менеджером деятельностей. В любое время пользователь может нажать кнопку Back и вернуться к предыдущему экрану в стеке. С точки зрения пользователя, это работает почти так же, как перемещение по истории просмотров в веб-браузере, где нажатие кнопки Back возвращает его на предыдущую страницу.

Рис. 2.2. Приложение Home

Процесс != Приложение

Изнутри каждый экран пользовательского интерфейса представлен классом Activity (см. раздел 2.3, «Деятельности»). Каждая деятельность имеет собственный жизненный цикл. Приложение — это одна или несколько деятельностей плюс процесс Linux, содержащий их. Это звучит весьма просто, не так ли? Но не обольщайтесь; я послал вам крученый мяч.

В Android приложения могут быть «живыми» даже в том случае, если процессы «убиты». Говоря другим языком, цикл жизненной активности приложения не привязан к жизненному циклу процессов. Процессы — это одноразовые контейнеры для деятельностей. Возможно, это отличается от всего, к чему вы привыкли в других системах, поэтому давайте посмотрим на процессы поближе, прежде чем продолжить.

Жизненные циклы богатых и знаменитых

В течение жизни каждая деятельность программы Android может находиться в одном из нескольких состояний, как показано на рис. 2.3. Вы, разработчик, не контролируете состояние программы. Всем этим управляет система. Однако вы получаете уведомления, когда состояние изменяется, благодаря вызовам метода *onXX()*.

Вы переопределяете эти методы в классе Activity, и Android вызывает их в соответствующее время:

- onCreate(Bundle): этот метод вызывается при первом запуске деятельности. Используйте его для начальной инициализации, например для создания пользовательского интерфейса. onCreate() имеет один параметр, который может быть либо установлен в *null*, либо содержать информацию о состоянии, ранее сохраненную методом onSaveInstanceState().

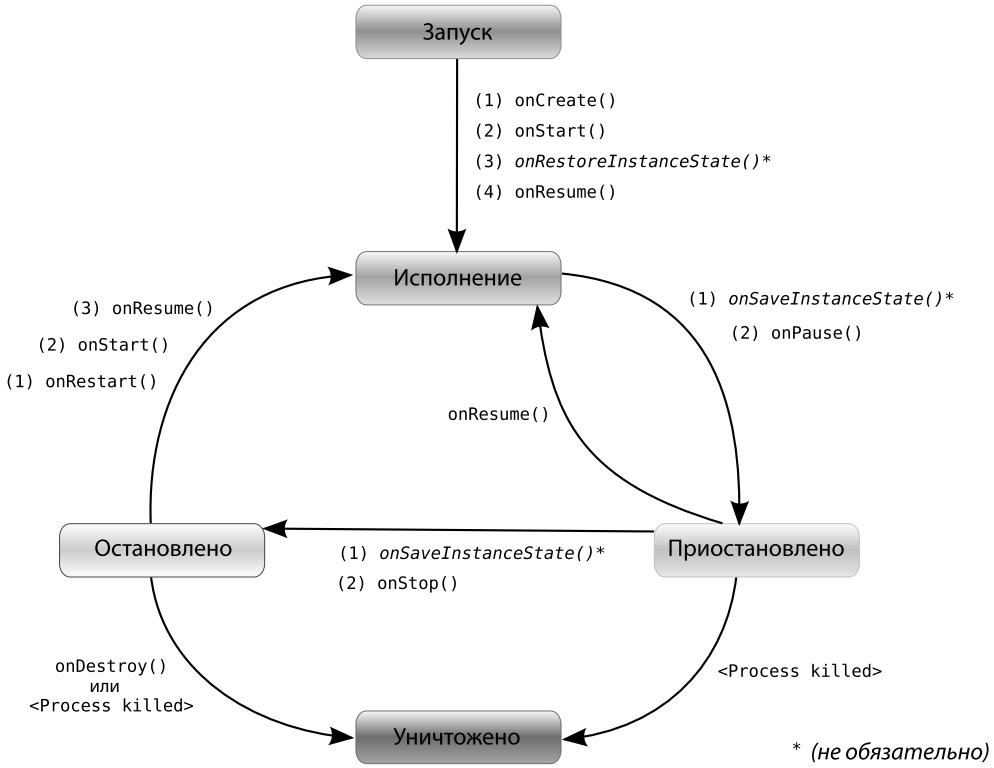

Рис. 2.3. Жизненный цикл деятельностей Android

- onStart(): этот метод указывает на то, что приложение готово для показа пользователю.
- onResume(): этот метод вызывается, когда деятельность может начать взаимодействие с пользователем. Это неплохое место кода для запуска анимации или музыки.
- onPause(): этот метод запускается, когда деятельность готова перейти в фоновый режим, обычно по причине того, что другая деятельность была запущена перед ней. Это место кода, где вам следует выполнить сохранение состояния программы, такого как, например, измененные записи базы данных.

- onStop(): вызывается, когда деятельность больше не видна пользователю и на некоторое время не востребована. Если в системе мало памяти, onStop() может быть никогда не вызван (система может просто завершить ваш процесс).

ОТКРЫВАНИЕ И ЗАКРЫВАНИЕ КЛАВИАТУРЫ

Я предлагаю быстрый способ тестирования корректности работы вашего кода по сохранению состояния приложения. В текущей версии Android изменение ориентации экрана (между портретными и ландшафтными режимами) приводит к тому, что система проходит через процессы сохранения состояния деятельности, приостановки, остановки, уничтожения, создания нового состояния деятельности из сохраненного состояния. На мобильном телефоне T-Mobile G1, например, эти процессы вызывает открытие и закрытие клавиатуры. На эмуляторе Android эти операции выполнит комбинация Ctrl+F11 или нажатие клавиш 7 или 9 на цифровой клавиатуре.

- onRestart(): вызов этого метода указывает на то, что ваша деятельность показана пользователю и вышла из состояния остановки.
- onDestroy(): вызывается непосредственно перед тем, как деятельность будет уничтожена. Если в системе мало памяти, onDestroy() может быть никогда не вызван (система может просто завершить процесс).
- onSaveInstanceState(Bundle): Android вызывает этот метод для того, чтобы разрешить деятельности сохранить свое предыдущее состояние, такое как позиция курсора в текстовом поле. Обычно не требуется переопределять его, так как стандартный вариант метода сохраняет состояние всех пользовательских элементов управления автоматически.
- onRestoreInstanceState(Bundle): этот метод вызывается, когда деятельность повторно инициализирована из состояния, ранее сохраненного методом onSaveInstanceState(). Стандартная реализация метода восстанавливает состояние пользовательского интерфейса.

Неактивные деятельности могут быть остановлены, или процессы Linux, которые их обслуживают, могут быть уничтожены в любое время, для того чтобы освободить ресурсы для новых деятельностей. Это вполне обычное дело, поэтому очень важно, чтобы ваше приложение разрабатывалось с самого начала с учетом возможности такого события. В некоторых случаях вызов метода onPause() может быть последним методом, вызванным в деятельности, поэтому именно здесь нужно сохранить данные, которые вы не хотите потерять до следующего запуска приложения.

В дополнение к управлению жизненным циклом приложения фреймворк Android предоставляет множество строительных блоков, которые могут быть использованы при создании вашего приложения. Давайте взглянем на них.

2.3. Строительные блоки

В Android SDK определено несколько объектов, с которыми должен быть хорошо знаком каждый разработчик. Наиболее важные из них — это *деятельности* (activities), *намерения* (intents), *сервисы* (services) и *контент-провайдеры* (content

providers). Позже вы познакомитесь с примерами их использования, а сейчас мне хотелось бы кратко их представить.

Деятельности

Деятельность — это окно или экран пользовательского интерфейса. Приложение может определить одну или несколько деятельностей для поддержки различных стадий работы программы. Как говорилось в разделе 2.2 «Оно живое!», каждая деятельность ответственна за сохранение своего состояния, которое может быть восстановлено позднее как часть жизненного цикла приложения (см. раздел 3.3 «Создание стартового экрана», в котором вы найдете соответствующий пример). Деятельности расширяют класс Context, поэтому вы можете использовать их для получения глобальной информации о приложении.

Намерения

Намерение — это механизм для описания одного действия, такого как «выбрать фотографию», «позвонить домой» или «открыть двери модуля[1]». В Android почти все работает благодаря намерениям, и существует множество возможностей по замене или повторному использованию компонентов. Обратитесь к разделу 3.5 «Создание информационного окна», в котором вы найдете пример работы с намерениями.

Допустим, есть намерение «отправить электронную почту». Если ваше приложение должно отправить почту, активируйте это намерение. Или, при написании нового приложения для работы с электронной почтой, зарегистрируйте деятельность для обработки этого намерения и замените предустановленную программу электронной почты. Когда пользователь попытается отправить электронную почту, ваша программа будет вызвана вместо стандартной.

Сервисы

Сервис — это задача, которая выполняется в фоновом режиме без прямого взаимодействия с пользователем. Сервисы похожи на демоны Unix. Например, рассмотрим музыкальный проигрыватель. Проигрывание музыки может быть запущено по намерению, но вы хотите, чтобы она играла даже тогда, когда пользователь переместился в другую программу. Для этого код, который выполняет проигрывание музыки, должен находиться внутри сервиса. Позже другая деятельность может подключиться к этому сервису и сообщить ему, что следует переключить или остановить воспроизведение. Android поставляется с множеством встроенных сервисов вместе с удобными API для доступа к ним. В разделе 12.2 «Интерактивные обои» используется сервис для рисования анимированной картинки в качестве фона домашнего экрана.

---

[1] «Открой двери модуля, ХЭЛ!» (Open the pod bay doors, HAL!) — фраза из фильма «Космическая одиссея 2001». — *Примеч. ред.*

Контент-провайдеры

Контент-провайдер — это набор данных, «завернутый» в пользовательский интерфейс API для чтения и записи. Это лучший способ разделять глобальные данные *между приложениями*. Например, Google предоставляет контент-провайдер для адресной книги. Вся информация здесь — имена, адреса, номера телефонов и так далее — может быть использована любыми приложениями, которым она нужна. Смотрите пример в разделе 9.5 «Использование ContentProvider».

2.4. Использование ресурсов

Ресурс — это локализованная текстовая строка, изображение или другой небольшой объем информации, в котором нуждается программа, не являющийся программным кодом. При сборке все ресурсы встраиваются в программу. Ресурсы используются для локализации продуктов или для поддержки устройств различных типов (см. раздел 3.4 «Использование альтернативных ресурсов».

Вы можете создавать и хранить ресурсы в папке res внутри проекта. Компилятор ресурсов Android (aapt)[1] обрабатывает ресурсы в соответствии с именем подпапки, в которой они расположены, и форматом файла. Например, графические файлы формата PNG и JPG должны находиться в папке, название которой начинается с res/drawable, а XML-файлы, которые описывают варианты компоновки экрана, должны располагаться в папке, имя которой начинается с res/layout. Добавляйте суффиксы для отдельных языков, ориентации экрана, плотности пикселей и так далее (см. раздел 13.5 «Все экраны, большие и маленькие»).

Компилятор ресурсов сжимает и упаковывает ресурсы и затем создает класс, называющийся *R*, который содержит идентификаторы, использованные при обращении к этим ресурсам в программе. Такой подход немного отличается от использования стандартных ресурсов Java, обращаться к которым можно по ключевым строкам. Выполнение операций с ресурсами подобным образом позволяет Android быть уверенным в том, что все ваши ссылки верны, и экономит пространство, не храня все ключи ресурсов. Eclipse использует похожий метод для хранения и доступа к ресурсам в своих плагинах.

Мы рассмотрим пример кода для доступа к ресурсам в главе 3 «Разработка пользовательского интерфейса».

2.5. Безопасность и защищенность

Как упоминалось ранее, каждое приложение работает в собственном процессе Linux. Аппаратное обеспечение не позволяет одному процессу получать доступ к памяти другого процесса. Более того, каждому приложению присваивается ин-

[1] http://d.android.com/guide/developing/tools/aapt.html

дивидуальный идентификатор. Любые файлы, созданные им, не могут быть прочитаны или переписаны другим приложением.

Кроме того, доступ к определенным критическим операциям ограничен, и следует особым образом запрашивать разрешение на их использование в файле, который называется AndroidManifest.xml. Когда приложение устанавливается, менеджер пакетов либо предоставляет, либо не предоставляет разрешения, основываясь на сертификатах и, если нужно, на ответах пользователя. Вот некоторые из наиболее распространенных разрешений, которые вам понадобятся:

- INTERNET: доступ к Интернету.
- READ_CONTACTS: чтение (но не запись) данных адресной книги пользователя.
- WRITE_CONTACTS: запись (но не чтение) в адресную книгу пользователя.
- RECEIVE_SMS: отслеживание входящих SMS-сообщений (текстовых).
- ACCESS_COARSE_LOCATION: использование средств грубого определения местоположения, например вышек сотовой связи или точек доступа Wi-Fi.
- ACCESS_FINE_LOCATION: использование более точных средств определения местоположения, таких как GPS.

Например, для отслеживания входящих SMS-сообщений добавьте следующие строки в файл манифеста:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.google.android.app.myapp" >
    <uses-permission android:name="android.permission.RECEIVE_SMS" />
</manifest>
```

Android может также запрещать доступ к целым частям системы. Используя теги XML в AndroidManifest.xml, вы можете ограничить круг пользователей, которым разрешено запускать деятельность, запускать сервис или подключаться к нему, создавать широковещательные намерения или получать доступ к данным контент-провайдера. Этот вид управления выходит за рамки данной книги, чтобы узнать больше, почитайте онлайновую справку по модели безопасности Android[1].

2.6. Вперед >>

Концепции, представленные в этой главе, будут использоваться на протяжении всей книги. В главе 3 «Разработка пользовательского интерфейса» мы будем пользоваться деятельностями и методами жизненного цикла для создания демонстрационного приложения. Глава 4, «Введение в 2D-графику», будет использовать некоторые из классов, предназначенных для работы с графикой в библиотеках Android. Медиа-кодеки будут рассмотрены в главе 5 «Мультимедиа», а тема контент-провайдеров будет раскрыта в главе 9 «Работа с SQL».

[1] http://d.android.com/guide/topics/security/security.html

II

Основы Android

- **Глава 3. Разработка пользовательского интерфейса**
- **Глава 4. Введение в 2D-графику**
- **Глава 5. Мультимедиа**
- **Глава 6. Хранение локальных данных**

3 Разработка пользовательского интерфейса

В главе 1 «Быстрый старт» мы использовали плагин Android для Eclipse, чтобы быстро, за несколько минут, собрать простую программу «Hello, Android». В части II мы создадим более серьезный пример: игру Sudoku (Судоку). Постепенно добавляя к этой игре новые возможности, вы узнаете о многих аспектах программирования для Android. Начнем с пользовательского интерфейса.

Полные примеры кода, использованного в этой книге, можно найти на странице http://pragprog.com/titles/eband3 или скачать с сайта издательства «Питер» (www.piter.com) архив с дополнительными материалами.

3.1. Введение в демонстрационную программу Sudoku

Sudoku является прекрасной демонстрационной программой для Android, так как игра сама по себе очень проста. У вас есть таблица 9×9, и вы пытаетесь заполнить ячейки числами так, чтобы каждый столбец, каждая строка и каждая область размером 3×3 содержали числа от 1 до 9 только по одному разу. Когда игра начинается, некоторые числа (*заданные*) уже находятся в ячейках. Все, что остается делать игроку, — это заполнить оставшиеся ячейки. Настоящая головоломка судоку имеет лишь одно уникальное решение.

В судоку обычно играют с карандашом и бумагой, но компьютеризированные версии игры также весьма популярны. В бумажной версии игры очень легко сделать ошибку в самом начале, и если это произойдет, вам придется вернуться назад и стереть большую часть своей работы. В версии Android вы можете менять значки в ячейках так часто, как хочется, без необходимости постоянно использовать стирательную резинку.

Android Sudoku (рис. 3.1) также предлагает подсказки, которые могут взять на себя некоторую часть рутинной работы по решению головоломки. В крайнем

случае система может даже решить головоломку за вас, но от этого вы не получите никакого удовольствия, не так ли? Поэтому нам нужно сбалансировать количество подсказок и сложность игры, мы не будем делать игру слишком простой.

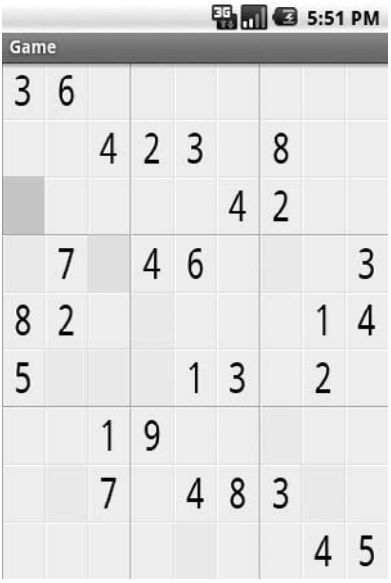

Рис. 3.1. Демонстрационная программа Sudoku для Android

О СУДОКУ

Многие люди думают, что судоку — это один из видов древней японской игры, но это не так. Хотя похожие головоломки можно найти во французских журналах XIX столетия, большинство экспертов сходятся во мнении, что современный вариант судоку разработал американский архитектор Говард Гарнс (Howard Garns). Игра называлась Number Place и была впервые опубликована в США в 1978 году в Dell Magazines.

3.2. Декларативная разработка

Пользовательский интерфейс может быть разработан с помощью одного из двух методов: *процедурного* (procedural) и *декларативного* (declarative). *Процедурная* разработка заключается в использовании программного кода. Например, если вы разрабатываете приложение Swing, вы пишете код на Java для создания и манипулирования всеми объектами пользовательского интерфейса, такими как JFrame и JButton. Таким образом, в Swing применяется процедурный метод.

Декларативный дизайн, с другой стороны, не включает в себя никакого кода. При разработке простой веб-страницы вы используете HTML, язык разметки, похожий на XML, который описывает то, что вы хотите видеть на странице, но не то, как эта страница должна себя вести. HTML — это декларативный язык.

Android пытается оседлать разрыв между процедурным и декларативным мирами, позволяя создавать пользовательские интерфейсы, используя и тот и другой стиль программирования. Основой вашей работы может быть Java-код, или же вы можете почти полностью полагаться на XML-описания. Посмотрите документацию для любого Android-компонента, предназначенного для построения пользовательского интерфейса, — вы увидите и вызовы Java API, и соответствующие декларативные XML-атрибуты, выполняющие те же функции.

Что же выбрать? И тот и другой путь правильный, но Google советует использовать декларативный XML всегда, когда это возможно. XML-код короче и проще для понимания, чем аналогичный Java-код, и это вряд ли изменится в будущих версиях.

А теперь давайте посмотрим, как использовать полученные знания для создания стартового экрана для Sudoku.

3.3. Создание стартового экрана

Мы начнем с каркасной программы для Android, созданной плагином Eclipse. Точно так же, как вы действовали в разделе 1.2 «Создание первой программы», создавая новый проект «Hello, Android», используйте следующие установки:

```
Project name: Sudoku
Build Target: Android 2.2
Application name: Sudoku
Package name: org.example.sudoku
Create Activity: Sudoku
Min SDK Version: 8
```

В реальной программе, конечно, вы используете свои собственные наименования. Помните, что имя пакета крайне важно. Каждое приложение в системе должно иметь уникальное имя пакета. Однажды выбранное имя не так-то просто изменить, потому что оно используется в очень многих местах.

Я предпочитаю оставлять окно эмулятора Android включенным все время и запускать программу после каждого изменения, поскольку это занимает всего несколько секунд. Если вы поступите так и запустите сейчас программу, то увидите пустой экран, который содержит лишь слова: «Hello World, Sudoku». Наш первый шаг состоит в том, чтобы заменить его на стартовый экран для игры, с кнопками, которые позволят игроку начать новую игру, продолжить предыдущую, получить информацию об игре и выйти. Итак, что же нужно изменить, чтобы сделать это?

Как говорилось в главе 2 «Ключевые концепции», приложения для Android — это коллекции самостоятельных деятельностей, каждая из которых определяет экран пользовательского интерфейса. При создании проекта Sudoku плагин Android создал деятельность в Sudoku.java:

Sudokuv0/src/org/example/sudoku/Sudoku.java

```
package org.example.sudoku;
import android.app.Activity;
import android.os.Bundle;
```

```
public class Sudoku extends Activity {
    /** Вызывается при первом создании деятельности. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Android вызывает метод onCreate() вашей деятельности для ее инициализации. Вызов setContentView() заполняет экран деятельности, используя Android view. Widget.

Мы могли бы использовать несколько строк кода на Java и, возможно, один или два дополнительных класса, чтобы определить пользовательский интерфейс, используя процедурный подход. Но вместо этого плагин выбрал декларативный путь, и мы продолжим движение в этом направлении. В ранее представленном коде R.Layout.Main — это идентификатор ресурса, который ссылается на файл main.xml в папке res/layout (рис. 3.2). Файл main.xml декларирует параметры пользовательского интерфейса в XML, и именно этот файл нам нужно модифицировать. Во время исполнения программы Android анализирует и создает (*разворачивает*) экземпляр ресурса, определенный в этой записи, и устанавливает его в качестве вьювера для текущей деятельности.

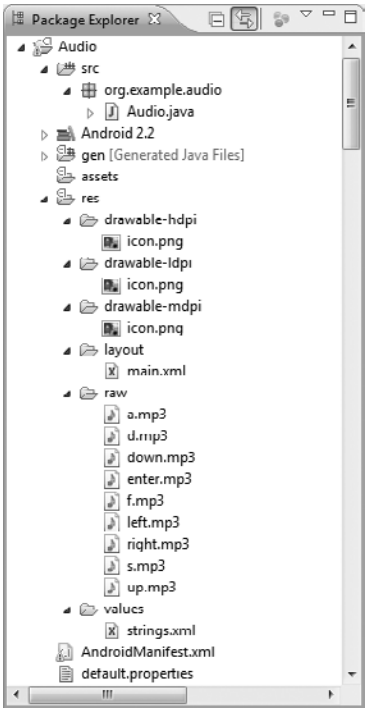

Рис. 3.2. Первоначальные ресурсы в проекте Sudoku

Важно отметить, что классом R плагин Android для Eclipse управляет автоматически. При размещении файла в любой подпапке папки res плагин замечает изменение и добавляет ID (идентификатор) ресурса в файл R.java в папке gen за вас. Если вы переместите или измените файл ресурса, R.java будет автоматически синхронизирован с этими изменениями. Если вы откроете файл в редакторе, то он будет выглядеть примерно так:

Sudokuv0/gen/org/example/sudoku/R.java

```
/* AUTO-GENERATED FILE. DO NOT MODIFY.
*
* Этот класс был автоматически создан
* инструментом aapt из данных ресурсов, которые он нашел.
* Его не следует модифицировать вручную.
*/
package org.example.sudoku;
public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

Шестнадцатеричные цифры — это просто целые числа, которые менеджер ресурсов Android использует для загрузки реальных данных — строк и других ресурсов, которые собираются в вашем проекте. Вам не нужно беспокоиться об их значениях. Просто помните, что это указатели, которые ссылаются на данные, но не объекты, которые содержат данные. Эти объекты не будут развернуты до тех пор, пока они не будут нужны. Обратите внимание, что почти все программы для Android, включая базовый фреймворк Android, содержат класс R. Обратитесь к онлайновой документации по android.R, для того чтобы узнать обо всех встроенных ресурсах, которыми вы можете пользоваться[1].

Итак, теперь мы будем модифицировать файл main.xml. Давайте проанализируем исходные определения для того, чтобы увидеть, что именно мы будем менять. Дважды щелкните на main.xml в Eclipse, чтобы открыть его. В зависимости от настроек Eclipse увидите либо визуальный редактор интерфейса, либо XML-редактор. В текущей версии ADT визуальный редактор не очень полезен, поэтому щелкните на main.xml или на вкладке Source (Код) в нижней части, для того чтобы увидеть код XML. Первая строка main.xml выглядит так:

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

[1] http://d.android.com/reference/android/R.html

Файлы XML в Android начинаются с этой строки. Она просто сообщает компилятору о том, что файл имеет формат XML и использует кодировку UTF-8. В тексте UTF-8 любой байт со значением меньше 128 изображает символ ASCII с тем же кодом. Для изображения прочих символов, например японских иероглифов, используются управляющие коды (escape codes).

ВОПРОС/ОТВЕТ

Почему Android использует XML? Эффективен ли он?

Android оптимизирован для мобильных устройств с ограниченным количеством памяти и вычислительной мощности, поэтому вам может показаться странным то, что он так широко использует XML. Кроме того, XML содержит многословные описания, а формат, который легко читается человеком, не отличается краткостью и эффективностью, правильно?

Хотя вы видите XML, когда пишете программу, плагин Eclipse вызывает компилятор ресурсов Android, aapt, для переработки XML в сжатый двоичный формат. Именно в этом формате, а не в виде оригинального текста XML, данные хранятся на устройстве.

Далее мы видим ссылку на <LinearLayout>:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <!-- ... -->
</LinearLayout>
```

Макет (способ размещения контента) — это контейнер для одного или более объектов-потомков, он располагает их на экране в пределах прямоугольника родительского объекта. Вот наиболее часто используемые способы размещения контента, предоставляемые Android:

- FrameLayout: располагает объекты-потомки так, что первый находится в верхнем левом углу экрана. Обычно такое расположение используется для просмотра в виде вкладок или переключателей изображения.
- LinearLayout: располагает вложенные объекты в один столбец или строку. Это наиболее распространенный способ размещения контента, вы часто будете пользоваться им.
- RelativeLayout: размещает объекты-потомки определенным образом по отношению к другим объектам или к родительскому объекту. Это часто используется при построении форм.
- TableLayout: располагает вложенные объекты по строкам и столбцам наподобие HTML-таблицы.

Некоторые параметры совпадают для всех вариантов расположения контента:

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

Эта команда определяет пространство имен XML для Android. Пространство имен следует определять лишь один раз, в первом XML-теге файла.

```
android:layout_width="fill_parent", android:layout_height="fill_parent"
```

Занимает всю ширину и высоту родительского объекта (в данном случае, окна). Возможные значения — fill_parent и wrap_content.

Внутри тега <LinearLayout> находится один виджет-потомок:

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
```

Эти команды определяют простую текстовую надпись. Заменим этот текст другим и добавим несколько кнопок. Вот наша первая попытка:

Sudokuv1/res/layout/main1.xml

```
<?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="@string/main_title" />
    <Button
        android:id="@+id/continue_button"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/continue_label" />
    <Button
        android:id="@+id/new_button"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/new_game_label" />
    <Button
        android:id="@+id/about_button"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/about_label" />
    <Button
        android:id="@+id/exit_button"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/exit_label" />
</LinearLayout>
```

Обратите внимание: если вы увидите предупреждение редактора о пропущенных грамматических описаниях (DTD или XML-схемах), просто не обращайте на них внимания или отключите сообщения об ошибках, открыв диалоговое окно Windows ▸ Preferences и установив параметр XML ▸ XML Files ▸ Validation ▸ Indicate when no grammar is specified в значение Ignore. Закройте окно проекта и выполните команду Project ▸ Clean для того, чтобы полностью избавиться от предупреждений.

В этом примере мы представляем новый синтаксис, @+id/resid. Вместо того чтобы обращаться к ID ресурса, объявленному где-то еще, вы можете создать новый ID ресурса, на который могут ссылаться другие. Например, @+id/about_button определяет ID для кнопки About (Об игре), которую мы будем использовать позже чтобы выполнить некоторые действия, когда пользователь нажмет эту кнопку.

Также, вместо жесткого программистского английского текста в файле определения расположения элементов интерфейса, мы используем синтаксис вида @string/resid, чтобы сослаться на строку, находящуюся в файле res/values/strings.xml. У вас могут быть различные версии этого и других файлов ресурсов, выбор которых может основываться на соображениях локализации или на других параметрах, таких как разрешение или ориентация экрана.

Откройте файл strings.xml, если нужно, переключитесь на закладку strings.xml в нижней части окна и введите следующее:

Sudokuv1/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Sudoku</string>
    <string name="main_title">Android Sudoku</string>
    <string name="continue_label">Continue</string>
    <string name="new_game_label">New Game</string>
    <string name="about_label">About</string>
    <string name="exit_label">Exit</string>
</resources>
```

Сохраните strings.xml, Eclipse перекомпилирует проект. Когда вы запустите программу, вы должны увидеть что-то наподобие рис. 3.3.

Поскольку это третье издание книги, я неплохо представляю себе, где именно люди могут столкнуться с проблемами. И, кстати, это может произойти прямо сейчас. Вы внесли много изменений, поэтому не удивляйтесь, если получите сообщение об ошибке вместо стартового экрана. Не паникуйте: просто просмотрите раздел 3.10 «Отладка», для того чтобы получить совет о том, как диагностировать проблему. Обычно путеводная нить для распутывания проблемы ждет вас в *LogCat*. Иногда выбор команды Project ▸ Clean помогает все исправить. Если у вас все еще что-то не получается, посетите веб-форум этой книги, и кто-нибудь там будет счастлив вам помочь[1].

Текущий экран вполне читаем, однако в него не помешает внести некоторые косметические улучшения. Сделаем мелкий текст покрупнее и выровняем его по центру, уменьшим размер кнопок и используем другой фоновый цвет. Вот определения цвета, которые вам следует поместить в res/values/colors.xml:

Sudokuv1/res/values/colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="background">#3500ffff</color>
</resources>
```

[1] http://forums.pragprog.com/forums/152

Рис. 3.3. Первая версия стартового экрана

А вот новый код для макета:

Sudokuv1/res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@color/background"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:padding="30dip"
android:orientation="horizontal" >
<LinearLayout
    android:orientation="vertical"
    android:layout_height="wrap_content"
    android:layout_width="fill_parent"
    android:layout_gravity="center" >
    <TextView
        android:text="@string/main_title"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_gravity="center"
        android:layout_marginBottom="25dip"
        android:textSize="24.5sp" />
    <Button
        android:id="@+id/continue_button"
        android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
        android:text="@string/continue_label" />
    <Button
        android:id="@+id/new_button"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/new_game_label" />
    <Button
        android:id="@+id/about_button"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/about_label" />
    <Button
        android:id="@+id/exit_button"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/exit_label" />
    </LinearLayout>
</LinearLayout>
```

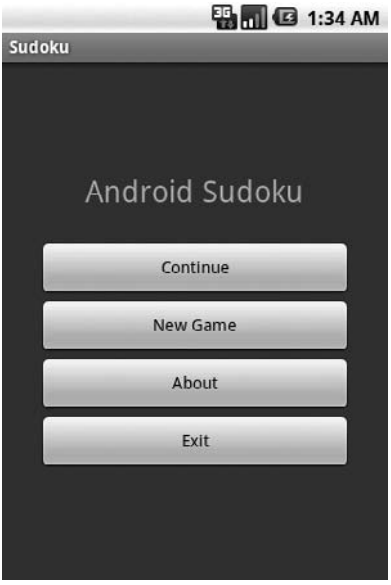

Рис. 3.4. Стартовый экран с новым расположением элементов

Результат показан на рис. 3.4. Этот новый экран лучше всего смотрится в портретном режиме (когда экран больше по размеру в высоту, чем в ширину), но как насчет ландшафтной (широкоэкранной) ориентации? Пользователь может переключать режимы в любое время, например открывая и закрывая клавиатуру или переворачивая телефон набок, поэтому мы должны это учитывать.

3.4. Использование альтернативных ресурсов

В качестве эксперимента попробуем переключить эмулятор в ландшафтный режим (Ctrl+F1, клавиши 7 или 9 на клавиатуре или Fn+Ctrl+F11 на Mac). Упс! Кнопка Exit ушла за пределы экрана (рис. 3.5). Как же нам это исправить?

Вы можете попытаться настроить расположение элементов таким образом, чтобы они хорошо смотрелись при разных ориентациях экрана. К несчастью, не всегда это возможно или приводит к появлению весьма странно выглядящих экранов. Хорошим решением будет создание отдельного макета для ландшафтного режима экрана. Этим мы сейчас и займемся.

Создайте файл с именем res/layout-land/main.xml (обратите внимание на суффикс -land), который содержит следующее определение расположения элементов:

Sudokuv1/res/layout-land/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="@color/background"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:padding="15dip"
    android:orientation="horizontal" >
    <LinearLayout
        android:orientation="vertical"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:layout_gravity="center"
        android:paddingLeft="20dip"
        android:paddingRight="20dip" >
        <TextView
            android:text="@string/main_title"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_gravity="center"
            android:layout_marginBottom="20dip"
            android:textSize="24.5sp" />
        <TableLayout
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_gravity="center"
            android:stretchColumns="*" >
            <TableRow>
                <Button
                    android:id="@+id/continue_button"
                    android:text="@string/continue_label" />
                <Button
                    android:id="@+id/new_button"
                    android:text="@string/new_game_label" />
```

```
            </TableRow>
            <TableRow>
                <Button
                    android:id="@+id/about_button"
                    android:text="@string/about_label" />
                <Button
                    android:id="@+id/exit_button"
                    android:text="@string/exit_label" />
            </TableRow>
        </TableLayout>
    </LinearLayout>
</LinearLayout>
```

Рис. 3.5. В ландшафтном режиме мы не видим кнопку Exit

Рис. 3.6. Использование специально созданного ландшафтного макета позволяет нам видеть все кнопки

В этом коде использован режим размещения TableLayout для создания двух колонок кнопок. Теперь запустите программу снова (рис. 3.6). Все кнопки видны даже в ландшафтном режиме. Вы можете скопировать тот же файл в res/layout-ldpi для поддержки режима низкого разрешения.

Суффиксы ресурсов можно использовать для создания альтернативной версии любого ресурса, не обязательно макета размещения элементов. Например, вы можете использовать их для того, чтобы хранить локализованные версии текстовых строк на различных языках. Поддержка экранов различной плотности в Android

сильно зависит от суффиксов ресурсов (см. раздел 13.5 «Все экраны, большие и маленькие»).

ВОПРОС/ОТВЕТ

Что такое Dips и Sps?

Исторически сложилось так, что программисты всегда занимались разработкой компьютерных интерфейсов, работая с пикселями. Например, вы можете сделать поле шириной в 300 пикселей, предусмотреть 5-пиксельное пространство между колонками и определить значки размером 16×16 пикселей. Проблема заключается в том, что если вы запустите такую программу на новых дисплеях с большим количеством точек на дюйм (dpi, dots per inch), пользовательский интерфейс будет выглядеть слишком мелко. В частности, это приведет к тому, что с ним будет почти невозможно работать.

Единицы измерения, не зависящие от разрешения, помогают решить эту проблему. Android поддерживает следующие единицы измерения:

px (pixels, пиксели): точки на экране, минимальные единицы изображения.

in (inches, дюймы): размеры, которые можно измерить линейкой.

mm (millimeters, миллиметры): такие размеры тоже можно измерить линейкой.

pt (points, пункты): 1/72 дюйма.

dp (density-independent pixels, пиксели, не зависящие от разрешения): абстрактная единица измерения, основанная на пиксельной плотности экрана. На дисплее с разрешением 160 точек на дюйм 1 dp=1 px.

dip: синоним для dp, обычно используемый в примерах от Google.

sp (scale independent pixels, пиксели, не зависящие от масштаба): похожи на dp, но в масштабе, измененном в соответствии с пользовательскими предустановками размера шрифта.

Для того чтобы сделать интерфейс подходящим для любого текущего и будущего типа экрана, я рекомендую всегда пользоваться единицами измерения sp для размера текста и dip для всего остального. Вы также должны обдумать использование векторной графики вместо точечных изображений (см. главу 4 «Введение в 2D-графику»).

3.5. Создание информационного окна

Когда пользователь выберет кнопку About, это значит, что он либо прикоснется к ней (если у него есть тач-скрин), либо перейдет к ней с помощью D-Pad (джойстика) или трекбола и нажмет выбранную кнопку, — мы хотели бы показать всплывающее окно с информацией о Sudoku.

После просмотра текста пользователь может нажать клавишу Back, чтобы закрыть окно. Мы можем выполнить это несколькими способами:

- определить новую деятельность (Activity) и запустить ее;
- использовать класс AlertDialog и показать его;
- использовать подкласс класса Android Dialog и показать его.

Для этого примера давайте определим новую деятельность. Так же как основная деятельность Sudoku, деятельность About нуждается в файле макета. Мы назовем его res/layout/about.xml:

Sudokuv1/res/layout/about.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dip" >
    <TextView
        android:id="@+id/about_content"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/about_text" />
</ScrollView>
```

Нам нужна лишь одна версия этого макета, так как он хорошо выглядит и при портретной и при ландшафтной ориентации экрана.

Теперь добавим строки для заголовка диалогового окна About и текст, который оно содержит в res/values/strings.xml:

Sudokuv1/res/values/strings.xml

```
    <string name="about_title">About Android Sudoku</string>
    <string name="about_text">\
Sudoku is a logic-based number placement puzzle.
Starting with a partially completed 9x9 grid, the
objective is to fill the grid so that each
row, each column, and each of the 3x3 boxes
(also called <i>blocks</i>) contains the digits
1 to 9 exactly once.
</string>
```

Обратите внимание на то, что строковые ресурсы могут содержать простое HTML-форматирование и распространяться на несколько строк. Если в этом тексте кое-что вас удивило, то знайте, что символ обратной косой черты (\) в строке about_text предотвращает появление дополнительного пустого пространства перед первым словом.

Деятельность About должна быть определена в файле About.java. Все, что нужно сделать, — это переопределить метод onCreate() и вызвать setContentView(). Для того чтобы создать новый класс в Eclipse, воспользуйтесь командой File ▸ New ▸ Class. Задайте следующие параметры:

```
Source folder: Sudoku/src
Package: org.example.sudoku
Name: About
```

Отредактируйте текст класса следующим образом:

Sudokuv1/src/org/example/sudoku/About.java

```
package org.example.sudoku;
import android.app.Activity;
import android.os.Bundle;
public class About extends Activity {
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.about);
    }
}
```

Далее мы должны все это привязать к кнопке About в классе Sudoku. Начнем с добавления нескольких команд импорта, которые нам понадобятся в Sudoku.java:

Sudokuv1/src/org/example/sudoku/Sudoku.java

```
import android.content.Intent;
import android.view.View;
import android.view.View.OnClickListener;
```

В методе onCreate() добавим код для вызова findViewById(), чтобы обнаружить элемент управления Android по ID ресурса, и добавим код для вызова setOnClickListener(), чтобы сообщить Android, какой объект следует «вытащить», когда пользователь коснется элемента управления или щелкнет по нему.

Sudokuv1/src/org/example/sudoku/Sudoku.java

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    // Устанавливаем обработчики нажатий для всех кнопок
    View continueButton = findViewById(R.id.continue_button);
    continueButton.setOnClickListener(this);
    View newButton = findViewById(R.id.new_button);
    newButton.setOnClickListener(this);
    View aboutButton = findViewById(R.id.about_button);
    aboutButton.setOnClickListener(this);
    View exitButton = findViewById(R.id.exit_button);
    exitButton.setOnClickListener(this);
}
```

Пока мы здесь, сделаем то же самое для каждой кнопки. Напомним, что константы типа R.id.about_button создаются плагином Eclipse в файле R.java, когда он видит код @+id/about_button в файле res/layout/main.xml.

Методу setOnClickListener() нужно передать объект, который реализует интерфейс Java OnClickListener. Мы передаем переменную this, поэтому лучше всего убедиться, что текущий класс (Sudoku) поддерживает реализацию этого интерфейса, иначе мы получим ошибку компиляции. OnClickListener имеет один метод, который называется onClick(), поэтому мы добавим этот метод в наш класс[1].

[1] Если вы — эксперт в области Java, вы можете удивиться, почему мы не используем анонимный внутренний класс для обработки нажатий. Вы можете это сделать, но, по словам разработчиков Android, каждый новый внутренний класс занимает дополнительно 1 Кбайт памяти.

Sudokuv1/src/org/example/sudoku/Sudoku.java

```
public class Sudoku extends Activity implements OnClickListener {
    // ...
    public void onClick(View v) {
        switch (v.getId()) {
        case R.id.about_button:
            Intent i = new Intent(this, About.class);
            startActivity(i);
            break;
        // Здесь можно разместить код для других кнопок...
        }
    }
}
```

Для того чтобы запустить деятельность в Android, мы для начала должны создать экземпляр класса Intent. Существуют два типа намерений: *public* (именованные) намерения, которые регистрируются системой и могут быть вызваны из любого приложения, и *private* (анонимные) намерения, которые используются внутри отдельного приложения. Для этого примера нам нужен лишь последний тип. Если вы загрузите программу и нажмете кнопку About, то получите ошибку (рис. 3.7). Что случилось?

Рис. 3.7. У нас проблема

Мы забыли об одном важном шаге: каждая деятельность должна быть объявлена в AndroidManifest.xml. Для того чтобы это сделать, произведите двойной щелчок на файле, чтобы его открыть, переключитесь в режим XML, если нужно, выбрав вкладку AndroidManifest.xml в нижней части окна, и добавьте новый тег <activity> после первого подобного закрывающего тега:

```
<activity android:name=".About"
    android:label="@string/about_title" >
</activity>
```

Теперь, если вы сохраните файл манифеста, запустите программу снова и нажмете кнопку About, вы должны увидеть что-то вроде рис. 3.8. Нажмите кнопку Back (Esc на эмуляторе), когда убедитесь в том, что все в порядке.

Все это выглядит неплохо, но не лучше было бы, если бы мы могли видеть исходный экран под текстом окна About?

3.6. Применение тем

Тема (theme) — это набор стилей, который переопределяет внешний вид и поведение виджетов Android. Создатели тем были вдохновлены *каскадными таблицами стилей* (Cascading Style Sheets, CSS), использующимися для веб-страниц, — они отделяют содержимое экрана от его представления или стиля. Android поставляется с несколькими темами, к которым вы можете обращаться по именам[1], или вы можете создать собственную тему как подкласс существующих и переопределить их значения по умолчанию.

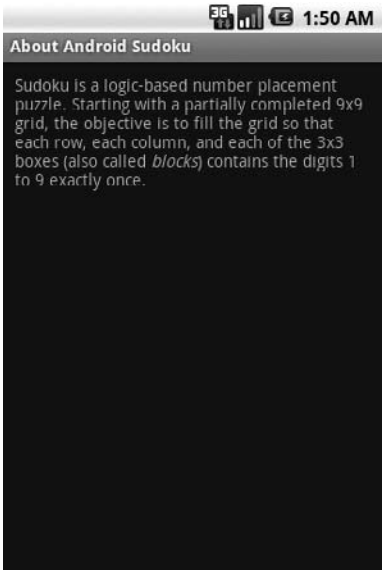

Рис. 3.8. Первая версия экрана About

Мы можем определить нашу собственную тему в res/values/styles.xml, но для этого примера мы просто воспользуемся возможностями одной из предустановленных тем. Откройте файл AndroidManifest.xml в редакторе и измените определение деятельности About так, чтобы она имела свойство, связанное с темой.

Sudokuv1/AndroidManifest.xml

```
<activity android:name=".About"
    android:label="@string/about_title"
    android:theme="@android:style/Theme.Dialog" >
</activity>
```

Префикс @android перед именем стиля означает, что это ссылка к ресурсу, определенному Android, а не к одному из тех, которые определены в вашей программе.

[1] См. http://d.android.com/reference/android/R.style.html — строки, начинающиеся с «Theme_».

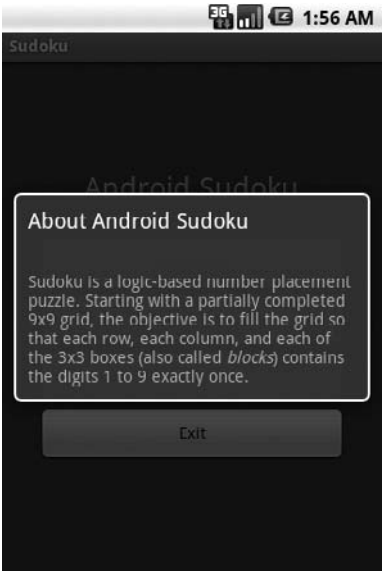

Рис. 3.9. Экран About после применения темы для диалоговых окон

Запустите программу снова, окно About сейчас выглядит так, как показано на рис. 3.9.

Многие программы нуждаются в меню и в установке параметров, поэтому следующие два раздела посвящены их определению.

3.7. Добавление меню

Android поддерживает два типа меню. Первый — это меню, которое появляется, когда вы нажимаете физическую кнопку Menu. Второй — это контекстное меню, которое всплывает, когда вы прикасаетесь пальцем к экрану и удерживаете его несколько секунд (или нажимаете и удерживаете трек-болл или центральную кнопку D-pad).

ВОПРОС/ОТВЕТ

Почему бы не использовать элемент управления HTML?

Android поддерживает внедрение веб-браузера прямо в элементы управления посредством класса WebView (см. раздел 7.2, «Веб-браузером с вьювером»). Так почему бы просто не использовать это для окна About?

В принципе, вы можете так и сделать. WebView поддерживает гораздо более богатое форматирование, чем простой TextView, но он имеет некоторые ограничения (такие, как невозможность использования прозрачного фона). Также WebView — это ресурсоемкий виджет, который работает медленнее и требует бóльших ресурсов, чем TextView. Для ваших собственных приложений используйте любой подходящий для ваших нужд способ.

Для начала сделаем так, что когда пользователь нажимает на клавишу Menu, программа открывает меню типа того, которое изображено на рис. 3.10. Начнем с определения нескольких строк, которые мы будем использовать позже:

Sudokuv1/res/values/strings.xml

```
<string name="settings_label">Settings...</string>
<string name="settings_title">Sudoku settings</string>
<string name="settings_shortcut">s</string>
<string name="music_title">Music</string>
<string name="music_summary">Play background music</string>
<string name="hints_title">Hints</string>
<string name="hints_summary">Show hints during play</string>
```

Затем определим меню, используя XML в res/menu/menu.xml:

Sudokuv1/res/menu/menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/settings"
        android:title="@string/settings_label"
        android:alphabeticShortcut="@string/settings_shortcut" />
</menu>
```

Сейчас нам нужно модифицировать класс Sudoku для вызова меню, которое мы только что определили. Для этого нам понадобятся еще несколько команд импорта.

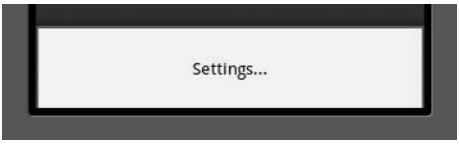

Рис. 3.10. Меню содержит один элемент — Settings (Настройки)

Sudokuv1/src/org/example/sudoku/Sudoku.java

```
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
```

Теперь мы переопределим метод Sudoku.onCreateOptionsMenu():

Sudokuv1/src/org/example/sudoku/Sudoku.java

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu, menu);
    return true;
}
```

getMenuInflater() возвращает экземпляр MenuInflater, который мы используем для чтения определения меню в XML и его отображения. Когда пользователь вы-

бирает любой элемент меню, вызывается onOptionsItemSelected(). Вот определение для этого метода:

Sudokuv1/src/org/example/sudoku/Sudoku.java

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    case R.id.settings:
        startActivity(new Intent(this, Prefs.class));
        return true;
    // Здесь можно разместить дополнительные элементы ...
    }
    return false;
}
```

Prefs — это класс, который мы собираемся определить, он будет отображать все наши предустановки и позволит пользователю их менять.

3.8. Добавление установок

Android предоставляет нам удобный механизм для определения всех установок вашего приложения и способа их отображения, практически не используя код. Вы определяете предустановки в файле ресурса, называемом res/xml/settings.xml:

Sudokuv1/res/xml/settings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android" >
    <CheckBoxPreference
        android:key="music"
        android:title="@string/music_title"
        android:summary="@string/music_summary"
        android:defaultValue="true" />
    <CheckBoxPreference
        android:key="hints"
        android:title="@string/hints_title"
        android:summary="@string/hints_summary"
        android:defaultValue="true" />
</PreferenceScreen>
```

Программа Sudoku имеет две установки: одна — для фоновой музыки, вторая — для показа подсказок. Ключи — это неизменяемые строки, которые будут использованы во внутренних механизмах базы данных предустановок Android

Далее определим класс Prefs и расширим им PreferenceActivity:

Sudokuv1/src/org/example/sudoku/Prefs.java

```
package org.example.sudoku;
import android.os.Bundle;
import android.preference.PreferenceActivity;
```

```
public class Prefs extends PreferenceActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.settings);
    }
}
```

Метод addPreferencesFromResource() читает определения установок из XML и разворачивает их на экране текущей активности. Все тяжелые операции отрабатываются в классе PreferenceActivity.

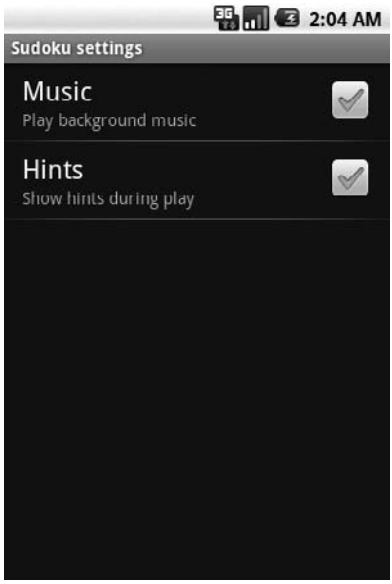

Рис. 3.11. Здесь не так много того, на что можно смотреть, но мы получили это даром

Не забудьте зарегистрировать деятельность Pref в AndroidManifest.xml:

Sudokuv1/AndroidManifest.xml

```
<activity android:name=".Prefs"
    android:label="@string/settings_title" >
</activity>
```

Теперь вернитесь к Sudoku, нажмите клавишу Menu, выберите пункт Settings и посмотрите с восхищением на появившуюся страницу установок Sudoku (рис. 3.11). Попытайтесь изменить значения и выйдите из программы, а затем вернитесь и убедитесь в том, что они все еще установлены.

Код, который позволяет читать установки и делать что-нибудь с ними, обсуждается в другой главе (глава 6, «Хранение локальных данных»). Сейчас давайте перейдем к кнопке New Game (Новая игра).

3.9. Начало новой игры

Если вы играли в Sudoku, то знаете, что иногда она бывает легкой, а иногда — нереально сложной. Мы хотим, чтобы при нажатии пользователем кнопки New Game всплыло диалоговое окно, которое предлагает ему выбрать между тремя уровнями сложности. Выбор из списка значений реализуется в Android довольно просто.

Для начала нам понадобится добавить еще несколько строк в res/values/strings.xml:

Sudokuv1/res/values/strings.xml

```
<string name="new_game_title">Difficulty</string>
<string name="easy_label">Easy</string>
<string name="medium_label">Medium</string>
<string name="hard_label">Hard</string>
```

Создайте список уровней сложности как ресурс-массив в res/values/arrays.xml:

Sudokuv1/res/values/arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <array name="difficulty">
        <item>@string/easy_label</item>
        <item>@string/medium_label</item>
        <item>@string/hard_label</item>
    </array>
</resources>
```

Нам также нужно добавить еще несколько команд импорта в классе Sudoku:

Sudokuv1/src/org/example/sudoku/Sudoku.java

```
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.util.Log;
```

Добавьте код в оператор switch метода onClick() для обработки щелчка на кнопке New Game:

Sudokuv1/src/org/example/sudoku/Sudoku.java

```
case R.id.new_button:
    openNewGameDialog();
    break;
```

Метод openNewGameDialog() занимается созданием пользовательского интерфейса для списка уровней сложности:

Sudokuv1/src/org/example/sudoku/Sudoku.java

```
private static final String TAG = "Sudoku" ;
private void openNewGameDialog() {
    new AlertDialog.Builder(this)
        .setTitle(R.string.new_game_title)
```

```
        .setItems(R.array.difficulty,
        new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialoginterface,
                int i) {
            startGame(i);
        }
    })
    .show();
}

private void startGame(int i) {
    Log.d(TAG, "clicked on " + i);
    // Запустите игру здесь...
}
```

Рис. 3.12. Диалоговое окно выбора уровней сложности

Метод setItems() принимает два параметра: ID ресурса из списка параметров и обработчик, который будет вызван, когда один из параметров будет выбран.

Когда вы запустите программу и нажмете на кнопку New Game, вы увидите диалоговое окно, как на рис. 3.13.

Сейчас мы не собираемся начинать игру, вместо этого, когда мы выбираем уровень сложности, мы просто выведем отладочное сообщение, используя метод Log.d() и передавая ему строку tag и сообщение для вывода.

3.10. Отладка

В Android можно применять те же приемы отладки программ, которые вы использовали при программировании для других платформ. Эти приемы включают вывод сообщений в журнал и пошаговое исполнение программ в отладчике.

Отладка с помощью записи сообщений в журнал

Класс Log предоставляет несколько статических методов для вывода сообщений нескольких уровней важности в системный журнал Android:

- Log.e(): ошибки;
- Log.w(): предупреждения;
- Log.i(): информация;
- Log.d(): отладка;
- Log.v(): подробности;
- Log.wtf(): Что за ужасная ошибка[1]!

Пользователи никогда не видят этот журнал, но, как разработчик, вы можете просматривать его несколькими способами. В Eclipse откройте окно LogCat, выбрав команду Window ▸ Show View ▸ Other... ▸ Android ▸ LogCat (рис. 3.13). Просматриваемые элементы можно отфильтровать по уровню важности или по тегам, которые вы задали при вызове соответствующего метода.

Если вы не пользуетесь Eclipse, то можете увидеть те же самые данные, выполнив команду adb logcat из папки SDK platform-tools[2]. Я рекомендую выполнить эту команду в отдельном окне и держать его открытым все время работы с эмулятором. Оно не вступит в конфликт ни с какими другими мониторами.

Сложно переоценить полезность журнала ошибок Android при разработке ПО. Помните ошибку, которую мы видели ранее в диалоговом окне About (см. рис. 3.7.)? Если вы откроете окно LogCat на этой точке, вы увидите это сообщение: «ActivityNotFoundException: Unable to find explicit activity class...have you declared this activity in your AndroidManifest.xml?» (ActivityNotFoundException: Невозможно найти заданный класс деятельности...объявили ли вы эту деятельность в вашем AndroidManifest.xml?» Нет ничего проще.

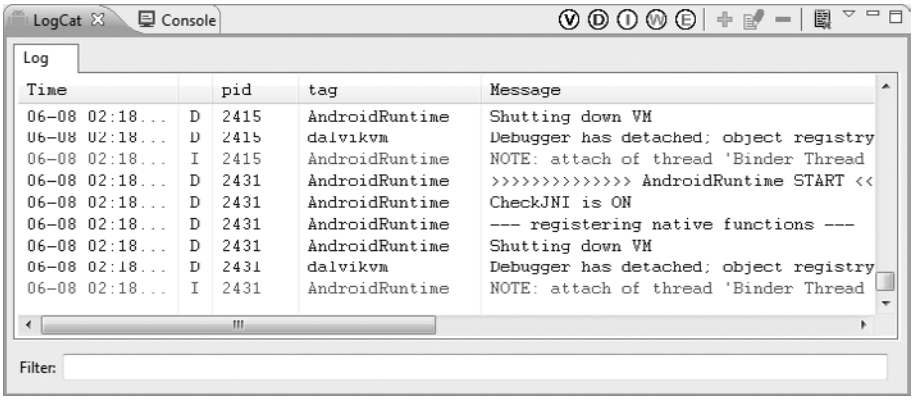

Рис. 3.13. Отладочные записи в окне LogCat

[1] Начиная с Android 2.2.

[2] http://d.android.com/guide/developing/tools/adb.html

Отладка с помощью отладчика

В добавление к сообщениям в журнале вы можете использовать отладчик Eclipse для того, чтобы устанавливать точки останова, пошагово исполнять программу или просматривать ее состояние. Для начала разрешите отладку в вашем проекте, добавив команду android:debuggable="true" в файл AndroidManifest.xml[1]:

Sudokuv1/AndroidManifest.xml

```
<application android:icon="@drawable/icon"
    android:label="@string/app_name"
    android:debuggable="true" >
```

Теперь, просто щелкните правой кнопкой мыши на проекте и выберите команду Debug As ▸ Android Application.

3.11. Выход из игры

Игре на самом деле не нужна кнопка Exit, так как пользователь может просто нажать на кнопку Back или на кнопку Home, чтобы сделать что-то другое. Но я хочу добавить такую кнопку, чтобы показать, как завершить деятельность.

Добавьте следующие строки в оператор switch метода onClick():

Sudokuv1/src/org/example/sudoku/Sudoku.java

```
case R.id.exit_button:
    finish();
    break;
```

Когда выбрана кнопка Exit, мы вызываем метод Finish(). Он прекращает деятельность и возвращает управление к следующей деятельности в стеке приложений Android (обычно — к экрану Home).

3.12. Вперед >>

Ух ты, сколько всего в одной главе! Вы начали с каркасного приложения, затем узнали, как использовать файлы макетов для организации пользовательского интерфейса и файлы ресурсов Android для работы с текстом, цветами и так далее. Вы добавили элементы управления, такие как кнопки и текстовые поля, применили темы для изменения внешнего вида программы и даже добавили меню и установки для организации взаимодействия с программой.

Android — это сложная система, но вы не должны знать ее досконально для того, чтобы начать работать. Если вам нужна помощь, сотни страниц руководств в Ин-

[1] Это необязательно, если вы пользуетесь эмулятором, но нужно для отладки на реальном устройстве. Не забудьте удалить эту команду, прежде чем публиковать ваш код.

тернете помогут углубиться в подробности всех классов и методов, использованных здесь[1]. Другой замечательный ресурс для поиска советов и секретов — это Planet Android[2]. И конечно, если вы зашли в тупик, вы всегда можете начать общаться на форуме этой книги[3]. Другие читатели и я будем счастливы вам помочь.

В главе 4 «Введение в 2-D графику» будет рассмотрен графический API Android для рисования изображения игры Sudoku.

[1] Для просмотра онлайновой документации откройте подпапку docs папки, в которую установлен Android SDK, или зайдите на страницу http://d.android.com/guide.

[2] http://www.planetandroid.com

[3] http://forums.pragprog.com/forums/152

4 Введение в 2D-графику

Мы пока еще обсудили лишь основные концепции и философию Android, поговорили о том, как создать простой пользовательский интерфейс с несколькими кнопками и диалоговым окном. Вы уже владеете всеми этими возможностями. Но кое-что упущено... что же? О, конечно, развлечения!

Хорошая графика может добавить нам хорошего настроения и увлекательности любому приложению. Android делает доступными самые мощные графические библиотеки для мобильных устройств. Фактически это два вида библиотек: одни — для двумерной графики, второй — для трехмерной[1].

В этой главе мы обсудим 2D-графику и применим полученные знания для создания одной из частей демонстрационной игры Sudoku. Глава 10, «3D-графика в OpenGL», посвящена созданию 3D-графики с использованием библиотеки OpenGL ES.

Для того чтобы облегчить чтение данной главы, я вынес весь код, который не имеет отношения к графике, в отдельный раздел (раздел 4.4. «Конец истории»). Именно здесь вы сможете найти методы для реализации логики игры Sudoku и обработки линий головоломки. Если ваша цель заключается в том, чтобы получить и запустить итоговый вариант Sudoku, вы должны ввести или загрузить этот код. Однако если вашей задачей является лишь изучение общих концепций графики в Android, вы можете пропустить эти дополнительные методы.

4.1. Основы

Android поддерживает полнофункциональные исходные библиотеки двумерной графики в пакете android.graphics. Изучив основы таких классов, как Color и Canvas, вы сразу сможете самостоятельно рисовать.

---

[1] Разработчиками Android рассматривалась возможность включения функционала для четырехмерной графики, но из-за нехватки времени этот функционал не был реализован.

Класс Color (Цвет)

Цвета в Android описываются четырьмя числами, по одному для каждого из каналов: альфа-канала, красного, зеленого и синего (ARGB — Alpha, Red, Green, Blue). Каждый из компонентов является восьмибитным целым, то есть может принимать 256 различных значений, поэтому цвет обычно упаковывается в 32-битное целое число. Для большей эффективности в коде Android используются целые числа вместо экземпляров класса Color.

Красный, зеленый и синий цвета не нуждаются в особом представлении, а вот сущность альфа-канала требует дополнительных пояснений. *Альфа-канал* (Alpha) — это мера прозрачности. Самое низкое значение, *0*, указывает на то, что элемент полностью прозрачен. Неважно, какие значения имеют компоненты RGB, если A равно 0. Самое высокое значение, 255, указывает на то, что элемент полностью непрозрачен. Промежуточные значения используются для задания просвечивающих или, другими словами, полупрозрачных цветов. Они позволяют видеть немного из того, что находится под объектом, который нарисован на переднем фоне.

Для того чтобы создать цвет, вы можете использовать одну из статических констант класса Color, например:

```
int color = Color.BLUE; // синий цвет
```

или, если вы знаете значения для альфа-канала, красной, зеленой и синей компонент, то можете использовать один из существующих статических методов, например:

```
// Полупрозрачный пурпурный
color = Color.argb(127, 255, 0, 255);
```

Лучше будет объявить все ваши цвета в ресурсном файле XML. Это облегчит наши действия в том случае, если позже мы решим что-то поменять:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="mycolor">#7fff00ff</color>
</resources>
```

Вы можете обращаться к цветам по именам в других XML-файлах, как мы делали в главе 3, или использовать имена в Java-коде, например:

```
color = getResources().getColor(R.color.mycolor);
```

Метод getResource() возвращает класс ResourceManager() для текущей деятельности, а getColor() запрашивает у менеджера поиск цвета, заданного в ID ресурса.

Класс Paint (Рисование)

Один из важнейших классов графической библиотеки Android — это класс Paint. Он хранит стили, цвета и другую информацию, необходимую для рисования графических объектов, в том числе битовые изображения, текст, геометрические фигуры.

Обычно когда вы рисуете что-то на экране и хотите нарисовать это сплошным цветом, вы устанавливаете этот цвет с помощью метода Paint.setColor().

Например:

```
cPaint.setColor(Color.LTGRAY);
```

Здесь использовано предопределенное значение для светло-серого цвета.

Объект Canvas (Холст)

Класс Canvas представляет собой поверхность, на которой вы рисуете. Обычно «холсты» изначально не имеют никакого содержимого, как пустые пленки для проекторов. Методы класса Canvas позволяют рисовать линии, прямоугольники, круги или другие произвольные графические объекты на плоскости.

В Android экран дисплея захватывается деятельностью (Activity), которая содержит окно просмотра, или вьювер (View), которое в свою очередь содержит холст (Canvas). У вас есть возможность рисовать на этих холстах, переопределяя метод View.onDraw(). Единственный параметр для onDraw() — это объект Canvas, на котором вы рисуете.

Вот пример деятельности, названной Graphics, которая содержит окно просмотра, называемое GraphicsView:

```
public class Graphics extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new GraphicsView(this));
    }
    static public class GraphicsView extends View {
        public GraphicsView(Context context) {
            super(context);
    }
    @Override
    protected void onDraw(Canvas canvas) {
        // Здесь начинаются команды рисования
    }
}
```

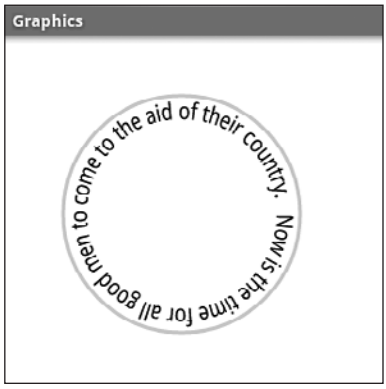

Рис. 4.1. Рисование текста по периметру окружности

Мы собираемся разместить некоторые команды рисования в методе onDraw() в следующем разделе.

Класс Path (Контур)

Класс Path хранит набор команд рисования векторных объектов, таких как линии, прямоугольники и кривые. Вот пример задания контура в виде окружности:

```
circle = new Path();
circle.addCircle(150, 150, 100, Direction.CW);
```

Здесь задана окружность с центром в позиции x=150, y=150 и радиусом в 100 пикселей. Сейчас, когда мы задали контур, давайте используем его для рисования окружности с обводкой, а также текста с ее внутренней стороны:

```
private static final String QUOTE = "Now is the time for all " +
            "good men to come to the aid of their country." ;
    canvas.drawPath(circle, cPaint);
    canvas.drawTextOnPath(QUOTE, circle, 0, 20, tPaint);
```

Вы можете видеть результат на рис.4.1. Так как окружность была нарисована по часовой стрелке (Direction.CW), текст также выводится в этом направлении.

Если вы хотите получить что-нибудь действительно забавное, Android предоставляет множество классов PathEffect, которые позволяют выполнять всяческие трюки вроде случайных изменений контура, сглаживания всех линейных отрезков контура с использованием кривых или разбиение контура на сегменты, а также другие эффекты.

Класс Drawable (Визуализация)

В Android класс Drawable используется для визуализации таких элементов, как битовые изображения или сплошные цвета, которые предназначены только для отображения на экране. Вы можете комбинировать объекты этого класса с другими графическими элементами или использовать их для виджетов пользовательского интерфейса (например, в качестве фона для кнопки или элемента View).

Объекты визуализации могут принимать различные формы:

- Bitmap: изображение PNG или JPG.
- NinePath: масштабируемое PNG-изображение, такое название появилось из-за того, что изначально изображение делится на девять частей. Оно используется как фоновое изображение для кнопок, которые могут менять размер.
- Shape: команды рисования векторных объектов, основанные на Path. Это один из заменителей SVG (Scalable Vector Graphics — масштабируемой векторной графики).
- Layers: контейнеры для дочерних объектов визуализации, которые выводятся один над другим в определенном порядке.
- States: контейнер, который показывает один из объектов-потомков, основанных на исходном объекте (битовой маске). Их используют для создания различных вариантов выделения кнопок и установки фокуса на них.

- Levels: контейнер, показывающий лишь один из объектов-потомков класса Drawable, основываясь на его уровне (целом числе). Это может быть использовано для индикатора заряда батареи или силы сигнала.
- Scale: контейнер для одного объекта-потомка класса Drawable, который изменяет размер на основе текущего заданного значения. Может быть использован для создания просмотрщика картинок с возможностью увеличения.

Объекты визуализации часто определяют в XML. Вот типичный пример, где объект визуализации определяет градиент от одного цвета к другому (в данном случае — от белого к серому). Угол задает направление градиента (270 градусов означают движение сверху вниз). Этот градиент будет использован для фона вьювера:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <gradient
        android:startColor="#FFFFFF"
        android:endColor="#808080"
        android:angle="270" />
</shape>
```

Для того чтобы воспользоваться приведенным примером, мы можем либо сослаться на данное определение в XML с помощью атрибута android:background=, либо вызвать метод setBackgroungResource() в методе вьювера onCreate(), как здесь:

```
setBackgroundResource(R.drawable.background);
```

Эта команда добавляет в наш пример GraphicsView интересный градиентный фон, как показано на рис. 4.2.

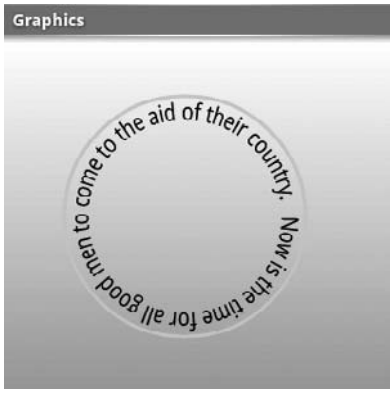

Рис. 4.2. Использование градиентного фона, описанного с помощью XML

НЕМНОГО О СУДОКУ

Через несколько лет после того, как игра была опубликована в США под названием Number Place, ее идею подобрал японский издатель Николи (Nikoli), который дал ей более звучное имя Sudoku (в переводе с японского — «одно число»). После этого игра распространилась по всему миру, но это уже другая история. К сожалению, создатель игры Гарнс (Garns) умер в 1989 году, до того, как у него появилась бы возможность увидеть, как его детище стало всемирной сенсацией.

Объекты Drawable могут быть расположены в различных папках в зависимости от плотности экрана, для которой они рассчитаны (см. раздел 3.4 «Использование альтернативных ресурсов»).

4.2. Добавление графики к Sudoku

Сейчас пришло время использовать то, что мы узнали, в нашей демонстрационной игре Sudoku. Когда мы остановились в конце главы 3, Sudoku имела стартовый экран, диалоговое окно с информацией о программе и все, что нужно для старта новой игры. Но отсутствовала одна очень важная вещь: сама игра! Мы будем использовать графическую 2D-библиотеку, для того чтобы создать эту часть программы.

Начало игры

Сначала нам нужен код, который начинает игру. Метод startGame() принимает один параметр — индекс уровня сложности, выбранный из списка.

Вот новое определение метода:

Sudokuv2/src/org/example/sudoku/Sudoku.java

```
private void startGame(int i) {
    Log.d(TAG, "clicked on " + i);
    Intent intent = new Intent(this, Game.class);
    intent.putExtra(Game.KEY_DIFFICULTY, i);
    startActivity(intent);
}
```

Игровая часть Sudoku будет реализована в дополнительной деятельности, названной Game, поэтому мы создаем новое намерение, чтобы ее вызвать. Мы размещаем уровень сложности в области extraData, принадлежащей намерению, и вызываем метод startActivity(), чтобы запустить новую деятельность.

Зона extraData — это список пар ключ/значение, который передается вместе с намерением. Ключи — это строки, а значения могут быть любыми примитивными типами данных, массивами примитивов, объектами класса Bundle или подклассами классов Serializable или Parceable.

Определение класса Game

Вот описание деятельности Game:

Sudokuv2/src/org/example/sudoku/Game.java

```
package org.example.sudoku;
import android.app.Activity;
import android.app.Dialog;
import android.os.Bundle;
import android.util.Log;
import android.view.Gravity;
import android.widget.Toast;
```

```
public class Game extends Activity {
    private static final String TAG = „Sudoku" ;
    public static final String KEY_DIFFICULTY =
        „org.example.sudoku.difficulty" ;
    public static final int DIFFICULTY_EASY = 0;
    public static final int DIFFICULTY_MEDIUM = 1;
    public static final int DIFFICULTY_HARD = 2;
    private int puzzle[];
    private PuzzleView puzzleView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d(TAG, „onCreate" );
        int diff = getIntent().getIntExtra(KEY_DIFFICULTY,
        DIFFICULTY_EASY);
        puzzle = getPuzzle(diff);
        calculateUsedTiles();
        puzzleView = new PuzzleView(this);
        setContentView(puzzleView);
        puzzleView.requestFocus();
    }
    // ...
}
```

Метод onCreate() берет уровень сложности из намерения и выбирает конкретную головоломку для загрузки в игру. Затем он создает экземпляр класса PuzzleView, устанавливает PuzzleView в качестве нового содержимого для вьювера. Так как это полностью настраиваемый вьювер, легче сделать это в коде, чем в XML.

Метод getPuzzle() (его определение приведено далее, в разделе 4.4, «Разное») ищет новую головоломку, основываясь на заданном уровне сложности. Метод calculateUsedTiles() (определенный в разделе 4.4, «Создание игровой логики») использует правила Sudoku, чтобы выяснить для каждой клетки в таблице девять на девять, какое число не подходит для неё, так как оно появляется где-нибудь в горизонтальном или вертикальном направлении или в квадрате три на три.

КАКОГО ОНО ВСЕ-ТАКИ РАЗМЕРА?

Источник обычной ошибки, которую делают новички разработки для Android, — это использование ширины и высоты вьювера внутри его конструктора. Когда вызывается конструктор вьювера, Android еще не знает, насколько большим будет вьювер, то есть размеры установлены в ноль. Реальный размер рассчитывается на стадии размещения, которая происходит после конструирования, но до того, как что-либо будет нарисовано. Вы можете использовать метод onSizeChanged() для того, чтобы узнать значения, когда они будут известны, или методы getWidth() и getHight(), так же, как и метод onDrow().

Так как речь идет о деятельности, мы должны зарегистрировать ее в AndroidManifest.xml:

Sudokuv2/AndroidManifest.xml

```
<activity android:name=".Game"
    android:label="@string/game_title" />
```

Также нам нужно добавить еще несколько строковых ресурсов в res/value/strings.xml:

Sudokuv2/res/values/strings.xml

```
<string name="game_title">Game</string>
<string name="no_moves_label">No moves</string>
<string name="keypad_title">Keypad</string>
```

Определение класса PuzzleView

Далее нам нужно определить класс PuzzleView. Вместо того чтобы использовать XML-макет, сделаем это полностью на Java.

Вот каркас класса:

Sudokuv2/src/org/example/sudoku/PuzzleView.java

```
package org.example.sudoku;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Rect;
import android.graphics.Paint.FontMetrics;
import android.graphics.Paint.Style;
import android.util.Log;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.view.View;
import android.view.animation.AnimationUtils;
public class PuzzleView extends View {
    private static final String TAG = "Sudoku" ;
    private final Game game;
    public PuzzleView(Context context) {
        super(context);
        this.game = (Game) context;
        setFocusable(true);
        setFocusableInTouchMode(true);
    }
    // ...
}
```

В конструкторе мы сохраняем ссылку на класс Game и устанавливаем параметр, позволяющий пользователю осуществлять ввод данных во вьювер. Внутри PuzzleView нам нужно применить метод onSizeChanging(). Он вызывается после того, как вьювер создан и Android знает все размеры.

Sudokuv2/src/org/example/sudoku/PuzzleView.java

```
private float width; // ширина одного тайла
private float height; // высота одного тайла
private int selX; // координата x выделенной области
private int selY; // координата y выделенной области
private final Rect selRect = new Rect();
@Override
protected void onSizeChanged(int w, int h, int oldw, int oldh) {
    width = w / 9f;
    height = h / 9f;
    getRect(selX, selY, selRect);
    Log.d(TAG, "onSizeChanged: width " + width + ", height "
        + height);
    super.onSizeChanged(w, h, oldw, oldh);
}
private void getRect(int x, int y, Rect rect) {
    rect.set((int) (x * width), (int) (y * height), (int) (x
        * width + width), (int) (y * height + height));
}
```

Мы используем onSizeChanged() для вычисления размера каждого тайла (элемента игрового изображения) на экране (1/9 от полного размера вьювера по ширине и высоте). Обратите внимание на то, что это — число с плавающей запятой, поэтому вполне возможно, что мы получим дробное количество пикселей. selRect — это прямоугольник, которым мы воспользуемся позже, чтобы отслеживать положение курсора.

ДРУГОЙ СПОСОБ СДЕЛАТЬ ЭТО

Когда я писал этот пример, я перепробовал несколько различных подходов, таких как использование кнопки для каждого тайла или объявление решетки классов ImageView в XML. После многих неудачных попыток я обнаружил, что подход с использованием одного вьювера для всего пазла и рисования линий и чисел внутри него обеспечивает наивысшую скорость и простоту. Хотя и у него есть свои недостатки, такие как необходимость рисовать выделение и явная обработка событий клавиатуры и тач-скрина. Когда вы разрабатываете свою собственную программу, я рекомендую сначала испробовать стандартные виджеты и вьюверы и переходить к собственным рисованным объектам, только если стандартные виджеты вам не подходят.

На данный момент мы создали вьювер для игры и мы знаем, какого он размера. Следующий шаг заключается в рисовании решетки из линий, которые разделяют клетки на игровой доске.

Рисование игровой доски

Android вызывает метод onDraw() вьювера каждый раз, когда любая его часть нуждается в обновлении. Проще говоря, onDraw() делает вид, что вы воссоздаете весь экран с нуля. В реальности вы можете рисовать только на маленькой части вьювера, определенной прямоугольником холста. Android сам заботится о создании зоны рисования для вас.

Начнем с определения нескольких новых цветов в res/values/color.xml, с которыми можно будет экспериментировать.

Sudokuv2/res/values/colors.xml

```
<color name="puzzle_background">#ffe6f0ff</color>
<color name="puzzle_hilite">#ffffffff</color>
<color name="puzzle_light">#64c6d4ef</color>
<color name="puzzle_dark">#6456648f</color>
<color name="puzzle_foreground">#ff000000</color>
<color name="puzzle_hint_0">#64ff0000</color>
<color name="puzzle_hint_1">#6400ff80</color>
<color name="puzzle_hint_2">#2000ff80</color>
<color name="puzzle_selected">#64ff8000</color>
```

Вот основной каркас метода onDraw():

Sudokuv2/src/org/example/sudoku/PuzzleView.java

```
@Override
protected void onDraw(Canvas canvas) {
    // Рисование фона...
    Paint background = new Paint();
    background.setColor(getResources().getColor(
        R.color.puzzle_background));
    canvas.drawRect(0, 0, getWidth(), getHeight(), background);
    // Рисование игровой доски...
    // Рисование чисел...
    // Рисование подсказок...
    // Рисование выделения...
}
```

Первый параметр — это объект Canvas, на котором нужно будет рисовать. Приведенный код рисует фон для игры, используя цвет puzzle_background.

Теперь давайте добавим код для рисования решетки из линий на доске:

Sudokuv2/src/org/example/sudoku/PuzzleView.java

```
// Рисование доски...
// Определение цветов для линий решетки
Paint dark = new Paint();
dark.setColor(getResources().getColor(R.color.puzzle_dark));
Paint hilite = new Paint();
hilite.setColor(getResources().getColor(R.color.puzzle_hilite));
Paint light = new Paint();
light.setColor(getResources().getColor(R.color.puzzle_light));
// Рисование вспомогательных линий решетки
for (int i = 0; i < 9; i++) {
    canvas.drawLine(0, i * height, getWidth(), i * height,
        light);
    canvas.drawLine(0, i * height + 1, getWidth(), i * height
        + 1, hilite);
    canvas.drawLine(i * width, 0, i * width, getHeight(),
        light);
```

```
    canvas.drawLine(i * width + 1, 0, i * width + 1,
        getHeight(), hilite);
    }
// Рисование основных линий решетки
for (int i = 0; i < 9; i++) {
    if (i % 3 != 0)
        continue;
    canvas.drawLine(0, i * height, getWidth(), i * height,
        dark);
    canvas.drawLine(0, i * height + 1, getWidth(), i * height
        + 1, hilite);
    canvas.drawLine(i * width, 0, i * width, getHeight(), dark);
    canvas.drawLine(i * width + 1, 0, i * width + 1,
        getHeight(), hilite);
}
```

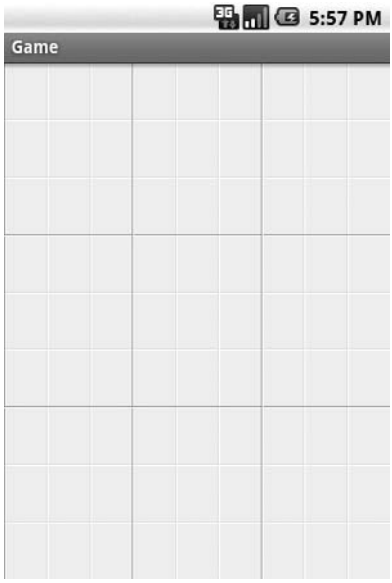

Рис. 4.3. Рисование линий решетки с использованием трех оттенков серого для создания эффекта объема

В коде использованы три разных цвета для линий решетки: светлый цвет между тайлами, темный между блоками три на три и цвет подсветки по краям каждого тайла для того, чтобы придать им легкий объемный эффект. Порядок, в котором выводятся линии, важен, так как линии, нарисованные позже, выводятся поверх линий, выведенных ранее. Вы можете увидеть, на что это похоже, на рис. 4.3. Далее нам понадобятся числа для расстановки их внутри ячеек решетки.

Рисование чисел

Следующий код выводит числа головоломки поверх тайлов. Некоторая сложность заключается в том, чтобы каждое число расположилось точно в центре соответствующего тайла и приняло подходящий размер.

Sudokuv2/src/org/example/sudoku/PuzzleView.java

```
// Рисование чисел...
// Определение цвета и стиля для чисел
Paint foreground = new Paint(Paint.ANTI_ALIAS_FLAG);
foreground.setColor(getResources().getColor(
    R.color.puzzle_foreground));
foreground.setStyle(Style.FILL);
foreground.setTextSize(height * 0.75f);
foreground.setTextScaleX(width / height);
foreground.setTextAlign(Paint.Align.CENTER);
// Рисование числа в центре тайла
FontMetrics fm = foreground.getFontMetrics();
// Центровка по оси X: использование выравнивания (и координаты центральной
точки)
float x = width / 2;
// Центровка по оси Y: сначала измеряем повышение/понижение
float y = height / 2 - (fm.ascent + fm.descent) / 2;
for (int i = 0; i < 9; i++) {
    for (int j = 0; j < 9; j++) {
        canvas.drawText(this.game.getTileString(i, j), i
            * width + x, j * height + y, foreground);
    }
}
```

Мы вызываем метод getTileString() (определенный в разделе 4.4, «Разное»), чтобы понять, какое число нам нужно отображать. Для нахождения размера числа мы устанавливаем размер шрифта в три четверти от высоты тайла, а соотношение сторон делаем таким же, как у тайла. Мы не можем использовать абсолютные значения в пикселях или пунктах, так как мы хотим, чтобы программа работала с любыми разрешениями экрана.

Для того чтобы задать позицию каждого числа, мы центруем его по измерениям x и y. Позицию x определить довольно просто — просто разделите ширину тайла на два. Но для координаты y мы должны слегка уменьшить начальную позицию так, чтобы центральная точка тайла стала центральной точки буквы, а не ее базовой линии (линии, проходящей под символом). Мы используем класс FontMetrics из графической библиотеки для того, чтобы указать, какое вертикальное пространство должен занимать символ в целом, затем делим полученное число на два, для того чтобы внести поправку. Вы можете видеть результат на рис. 4.4.

Созданный нами механизм позволяет выполнять начальное заполнение головоломки (вводить исходные данные). Следующий шаг заключается в том, чтобы позволить пользователю вводить цифры, которые он подбирает для решения задачи, в пустые ячейки.

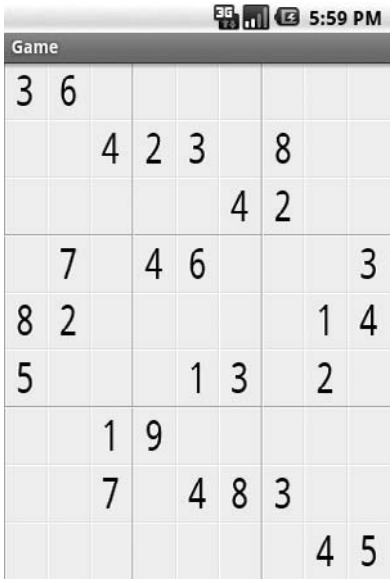

Рис. 4.4. Символы, отцентрованные внутри тайлов

4.3. Обработка ввода

Одна отличительная особенность программирования под Android в сравнении, скажем, с программированием под iPhone состоит в том, что телефоны на Android существуют в различных форм-факторах и размерах и имеют множество средств для ввода данных. Они могут иметь клавиатуру, джойстик, тач-скрин или некоторые сочетания этих средств управления.

Хорошая программа на Android должна быть готова к поддержке любого доступного аппаратного способа ввода данных, так же как она должна быть готова к поддержке любого разрешения экрана.

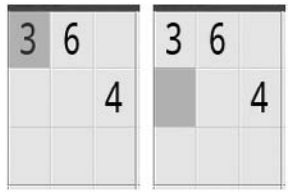

Рис. 4.5. Рисование и перемещение выделения

Задание и обновление выделенной области

Для начала мы должны создать небольшой курсор, который показывает игроку выделенный тайл. Выделенный тайл — это тот, который может быть изменен, когда пользователь введет число. Следующий код рисует выделение в методе onDraw():

Sudokuv2/src/org/example/sudoku/PuzzleView.java

```
// Рисование выделения...
Log.d(TAG, "selRect=" + selRect);
Paint selected = new Paint();
selected.setColor(getResources().getColor(
    R.color.puzzle_selected));
canvas.drawRect(selRect, selected);
```

Мы используем прямоугольник выделения, параметры которого вычислены ранее в методе onSizeChanged(), для рисования полупрозрачного цветного прямоугольника поверх выделенного тайла.

Далее мы обеспечиваем способ перемещения выделения, переопределяя метод onKeyDown():

Sudokuv2/src/org/example/sudoku/PuzzleView.java

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    Log.d(TAG, "onKeyDown: keycode=" + keyCode + ", event="
        + event);
    switch (keyCode) {
    case KeyEvent.KEYCODE_DPAD_UP:
        select(selX, selY - 1);
        break;
    case KeyEvent.KEYCODE_DPAD_DOWN:
        select(selX, selY + 1);
        break;
    case KeyEvent.KEYCODE_DPAD_LEFT:
        select(selX - 1, selY);
        break;
    case KeyEvent.KEYCODE_DPAD_RIGHT:
        select(selX + 1, selY);
        break;
    default:
        return super.onKeyDown(keyCode, event);
    }
    return true;
}
```

Если пользователь имеет джойстик (D-pad) и нажимает кнопки «вверх», «вниз», «вправо» или «влево», мы вызываем select() для перемещения курсора выделения в выбранном направлении.

А как насчет трекбола? Мы можем переопределить метод onTrackBallEvent(), но на деле оказывается, что если вы не обрабатываете события трекбола, Android ав-

томатически транслирует их в события джойстика. Поэтому мы можем оставить его непереопределенным для этого примера.

Внутри метода select() мы вычисляем новые координаты *x* и *y* выделения и затем используем getRect() для вычисления нового прямоугольника выделения.

Sudokuv2/src/org/example/sudoku/PuzzleView.java

```
private void select(int x, int y) {
    invalidate(selRect);
    selX = Math.min(Math.max(x, 0), 8);
    selY = Math.min(Math.max(y, 0), 8);
    getRect(selX, selY, selRect);
    invalidate(selRect);
}
```

Обратите внимание на два вызова *invalidate()*. Первый сообщает Android о том, что пространство, покрытое старым прямоугольником выделения (слева на рис. 4.5), должно быть перерисовано. Второй вызов *invalidate()* говорит о том, что новую область выделения (справа на рисунке) тоже нужно перерисовать. В реальности мы ничего здесь не рисуем.

Вот очень важный момент: никогда не вызывайте функции рисования нигде, кроме метода *onDraw()*. Вместо этого используйте метод *invalidate()*, чтобы отметить область как грязную, недействительную (*dirty*). Менеджер окон собирает все «грязные» прямоугольники и в некоторый момент вызывает onDraw(). Недействительные области становятся областями обрезки, таким образом, обновление экрана оптимизировано в расчете только на те области, которые меняются.

ОПТИМИЗАЦИЯ ОБНОВЛЕНИЯ

В ранней версии этого примера я делал недействительным весь экран. Таким образом, каждое нажатие кнопки приводило к необходимости перерисовывать всю головоломку. Это вызывало заметные «подтормаживания». Изменение кода таким образом, чтобы он помечал недействительным лишь небольшие участки экрана, привело к ускорению работы программы.

Теперь давайте снабдим игрока возможностью вводить новое число в выбранное поле.

Ввод чисел

Для обработки ввода с клавиатуры мы добавим несколько строк в оператор case в методе onKeyDown() для чисел от 0 до 9 (0 или пробел означает стирание числа).

Sudokuv2/src/org/example/sudoku/PuzzleView.java

```
case KeyEvent.KEYCODE_0:
case KeyEvent.KEYCODE_SPACE: setSelectedTile(0); break;
case KeyEvent.KEYCODE_1: setSelectedTile(1); break;
case KeyEvent.KEYCODE_2: setSelectedTile(2); break;
case KeyEvent.KEYCODE_3: setSelectedTile(3); break;
case KeyEvent.KEYCODE_4: setSelectedTile(4); break;
```

```
case KeyEvent.KEYCODE_5: setSelectedTile(5); break;
case KeyEvent.KEYCODE_6: setSelectedTile(6); break;
case KeyEvent.KEYCODE_7: setSelectedTile(7); break;
case KeyEvent.KEYCODE_8: setSelectedTile(8); break;
case KeyEvent.KEYCODE_9: setSelectedTile(9); break;
case KeyEvent.KEYCODE_ENTER:
case KeyEvent.KEYCODE_DPAD_CENTER:
    game.showKeypadOrError(selX, selY);
    break;
```

Для поддержки джойстика мы проверяем нажатие клавиши Enter или центральной кнопки джойстика в onKeyDown() и показываем всплывающую клавиатуру, которая позволяет пользователю выбрать, какое именно число нужно разместить в ячейке.

Для тач-скринов мы переопределяем метод onTouchEvent() и выводим на экран клавиатуру, которая будет создана позже:

Sudokuv2/src/org/example/sudoku/PuzzleView.java

```
@Override
public boolean onTouchEvent(MotionEvent event) {
    if (event.getAction() != MotionEvent.ACTION_DOWN)
        return super.onTouchEvent(event);
    select((int) (event.getX() / width),
        (int) (event.getY() / height));
    game.showKeypadOrError(selX, selY);
    Log.d(TAG, "onTouchEvent: x " + selX + ", y " + selY);
    return true;
}
```

В конце концов все дороги ведут к вызову метода setSelectedTile() для изменения числа в ячейке:

Sudokuv2/src/org/example/sudoku/PuzzleView.java

```
public void setSelectedTile(int tile) {
    if (game.setTileIfValid(selX, selY, tile)) {
        invalidate();// можно изменить подсказки
    } else {
        // Число не подходит для этого тайла
        Log.d(TAG, "setSelectedTile: invalid: " + tile);
    }
}
```

Метод showKeypadOrError() рассмотрен в разделе 4.4, «Создание экранной клавиатуры» метод setTileIfValid() определен в разделе 4.4, «Создание игровой логики».

Обратите внимание на то, что мы вызываем invalidate() без параметров. Этот вызов помечает весь экран как недействительный, что нарушает мой собственный совет, данный в примечании! Однако в данном случае это необходимо, так как любые добавленные или удаленные цифры могут изменить подсказки, которые мы создадим в следующем разделе.

Добавление подсказок

Можно ли немного помочь пользователю, не решая за него всю головоломку? Как насчет того, чтобы раскрасить фон каждого тайла в зависимости от того, сколько возможных вариантов расстановки чисел он может иметь? Добавьте следующий код в onDraw() перед кодом рисования выделения:

Sudokuv2/src/org/example/sudoku/PuzzleView.java

```
// Нарисовать подсказки...
// Выбрать цвет подсказки, основываясь на количестве оставшихся ходов
Paint hint = new Paint();
int c[] = { getResources().getColor(R.color.puzzle_hint_0),
    getResources().getColor(R.color.puzzle_hint_1),
    getResources().getColor(R.color.puzzle_hint_2), };
Rect r = new Rect();
for (int i = 0; i < 9; i++) {
    for (int j = 0; j < 9; j++) {
        int movesleft = 9 - game.getUsedTiles(i, j).length;
        if (movesleft < c.length) {
            getRect(i, j, r);
            hint.setColor(c[movesleft]);
            canvas.drawRect(r, hint);
        }
    }
}
```

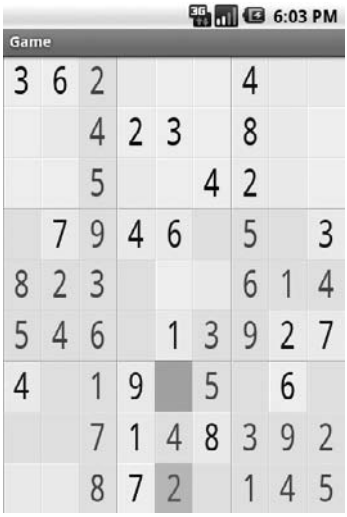

Рис. 4.6. Тайлы раскрашены по-разному в зависимости от того, как много возможных значений могут быть в них введены

Мы используем три состояния для обозначения отсутствия хода, одного и двух возможных ходов. Если ходов нет, это означает, что игрок сделал что-то не так и нуждается в возврате к более раннему состоянию игры.

Результат будет выглядеть, как показано на рис. 4.6. Найдете ли вы ошибку (или ошибки), допущенные игроком[1]?

Встряска

Что, если пользователь ввел очевидно неверное число, такое как число, которое уже имеется в блоке три на три? Просто для забавы сделаем так, чтобы экран вздрагивал каждый раз при неверном ходе. Для начала мы добавим вызов в setSelectedTile() при вводе неправильного числа.

Sudokuv2/src/org/example/sudoku/PuzzleView.java

```
Log.d(TAG, "setSelectedTile: invalid: " + tile);
startAnimation(AnimationUtils.loadAnimation(game,
    R.anim.shake));
```

Этот код загружает и выполняет ресурс, который называется R.anim.shake, определенный в res/anim/shake.xml, который «встряхивает» экран в течение 1000 миллисекунд (1 секунды) на 10 пикселей из стороны в сторону.

Sudokuv2/res/anim/shake.xml

```
<?xml version="1.0" encoding="utf-8"?>
<translate
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromXDelta="0"
    android:toXDelta="10"
    android:duration="1000"
    android:interpolator="@anim/cycle_7" />
```

Количество повторений анимации, ее скорость и ускорение контролируются интерполятором анимации, определенным в XML.

Sudokuv2/res/anim/cycle_7.xml

```
<?xml version="1.0" encoding="utf-8"?>
<cycleInterpolator
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:cycles="7" />
```

В частности, здесь задано, что анимация будет повторяться семь раз.

4.4. Конец истории

Теперь давайте вернемся назад и свяжем вместе несколько свободных концов, начиная с класса Keypad. Эти части нужны для компиляции и работы программы, но ничего не делают с ее графической составляющей. Если вам это неинтересно, можете пропустить материал до раздела 4.5 «Дополнительные улучшения».

[1] В верхнем ряду центрального блока два числа введены неверно.

Создание экранной клавиатуры

Экранная клавиатура удобна для телефонов, которые не имеют обычной клавиатуры. Она отображает набор цифр от 1 до 9 в деятельности, появляющейся поверх головоломки. Основная цель диалогового окна клавиатуры заключается в возврате цифры, выбранной пользователем.

В дополнительных материалы к книге представлен макет пользовательского интерфейса из res/layout/keypad.xml:

Sudokuv2/res/layout/keypad.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/keypad"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:stretchColumns="*" >
    <TableRow>
        <Button android:id="@+id/keypad_1"
            android:text="1" >
        </Button>
        <Button android:id="@+id/keypad_2"
            android:text="2" >
        </Button>
        <Button android:id="@+id/keypad_3"
            android:text="3" >
        </Button>
    </TableRow>
    <TableRow>
        <Button android:id="@+id/keypad_4"
            android:text="4" >
        </Button>
        <Button android:id="@+id/keypad_5"
            android:text="5" >
        </Button>
        <Button android:id="@+id/keypad_6"
            android:text="6" >
        </Button>
    </TableRow>
    <TableRow>
        <Button android:id="@+id/keypad_7"
            android:text="7" >
        </Button>
        <Button android:id="@+id/keypad_8"
            android:text="8" >
        </Button>
        <Button android:id="@+id/keypad_9"
            android:text="9" >
```

```
        </Button>
    </TableRow>
</TableLayout>
```

Далее определим класс Keypad.

Вот его код:

Sudokuv2/src/org/example/sudoku/Keypad.java

```
package org.example.sudoku;
import android.app.Dialog;
import android.content.Context;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
public class Keypad extends Dialog {
    protected static final String TAG = "Sudoku" ;
    private final View keys[] = new View[9];
    private View keypad;
    private final int useds[];
    private final PuzzleView puzzleView;
    public Keypad(Context context, int useds[], PuzzleView puzzleView) {
        super(context);
        this.useds = useds;
        this.puzzleView = puzzleView;
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setTitle(R.string.keypad_title);
        setContentView(R.layout.keypad);
        findViews();
        for (int element : useds) {
            if (element != 0)
                keys[element - 1].setVisibility(View.INVISIBLE);
        }
        setListeners();
    }
    // ...
}
```

Если отдельные цифры не подходят для ячейки (например, та же цифра уже появлялась в данной строке), мы делаем цифру невидимой в ячейке экранной клавиатуры, в результате пользователь не может ее выбрать (рис. 4.7).

Метод findViews() выбирает и сохраняет вьюверы для всех кнопок экранной клавиатуры и главного окна клавиатуры:

Sudokuv2/src/org/example/sudoku/Keypad.java

```
private void findViews() {
    keypad = findViewById(R.id.keypad);
    keys[0] = findViewById(R.id.keypad_1);
    keys[1] = findViewById(R.id.keypad_2);
```

```
    keys[2] = findViewById(R.id.keypad_3);
    keys[3] = findViewById(R.id.keypad_4);
    keys[4] = findViewById(R.id.keypad_5);
    keys[5] = findViewById(R.id.keypad_6);
    keys[6] = findViewById(R.id.keypad_7);
    keys[7] = findViewById(R.id.keypad_8);
    keys[8] = findViewById(R.id.keypad_9);
}
```

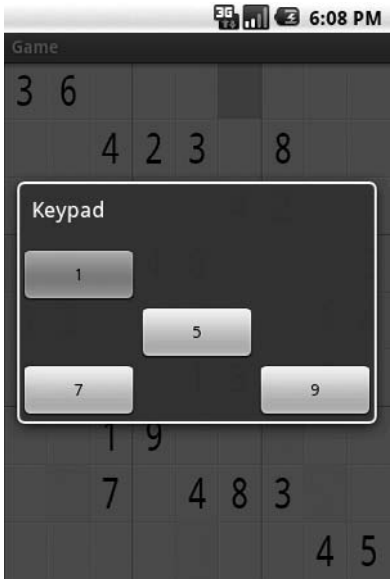

Рис. 4.7. Неправильные значения скрыты в окне экранной клавиатуры

Метод setListeners() перебирает все клавиши экранной клавиатуры и устанавливает обработчики для каждой из них. Также он устанавливает обработчики для всего окна клавиатуры.

Sudokuv2/src/org/example/sudoku/Keypad.java

```
private void setListeners() {
    for (int i = 0; i < keys.length; i++) {
        final int t = i + 1;
        keys[i].setOnClickListener(new View.OnClickListener(){
            public void onClick(View v) {
                returnResult(t);
            }});
    }
    keypad.setOnClickListener(new View.OnClickListener(){
        public void onClick(View v) {
            returnResult(0);
        }});
}
```

Когда игрок выбирает одну из кнопок на экранной клавиатуре, вызывается метод returnResult() с цифрой этой кнопки. Если игрок выбрал место, на котором нет кнопки, вызывается метод returnResult() с нулем, что указывает на то, что тайл должен быть очищен.

onKeyDown() вызывается, когда игрок использует обычную клавиатуру для ввода чисел:

Sudokuv2/src/org/example/sudoku/Keypad.java

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    int tile = 0;
    switch (keyCode) {
    case KeyEvent.KEYCODE_0:
    case KeyEvent.KEYCODE_SPACE: tile = 0; break;
    case KeyEvent.KEYCODE_1: tile = 1; break;
    case KeyEvent.KEYCODE_2: tile = 2; break;
    case KeyEvent.KEYCODE_3: tile = 3; break;
    case KeyEvent.KEYCODE_4: tile = 4; break;
    case KeyEvent.KEYCODE_5: tile = 5; break;
    case KeyEvent.KEYCODE_6: tile = 6; break;
    case KeyEvent.KEYCODE_7: tile = 7; break;
    case KeyEvent.KEYCODE_8: tile = 8; break;
    case KeyEvent.KEYCODE_9: tile = 9; break;
    default:
        return super.onKeyDown(keyCode, event);
    }
    if (isValid(tile)) {
        returnResult(tile);
    }
    return true;
}
```

Если цифра подходит для текущего тайла, то вызывается returnResult(), иначе данные игнорируются.

Метод isValid() проверяет, является ли заданная цифра подходящей для текущей позиции:

Sudokuv2/src/org/example/sudoku/Keypad.java

```
private boolean isValid(int tile) {
    for (int t : useds) {
        if (tile == t)
            return false;
    }
    return true;
}
```

Если цифра находится в массиве useds, значит, она не подходит, так как она уже использована в текущей строке, столбце или блоке.

Метод returnResult() вызывается для возврата цифры, выбранной в вызванной деятельности:

Sudokuv2/src/org/example/sudoku/Keypad.java

```
private void returnResult(int tile) {
    puzzleView.setSelectedTile(tile);
    dismiss();
}
```

Мы вызываем метод PuzzleView.setSelectedTile(),чтобы изменить текущий тайл головоломки. Метод dismiss вызывает закрытие диалогового окна Keypad. Сейчас, когда у нас есть диалоговое окно Keypad, вызовем класс Game и извлечем результат.

Sudokuv2/src/org/example/sudoku/Game.java

```
protected void showKeypadOrError(int x, int y) {
    int tiles[] = getUsedTiles(x, y);
    if (tiles.length == 9) {
        Toast toast = Toast.makeText(this,
            R.string.no_moves_label, Toast.LENGTH_SHORT);
        toast.setGravity(Gravity.CENTER, 0, 0);
        toast.show();
    } else {
        Log.d(TAG, "showKeypad: used=" + toPuzzleString(tiles));
        Dialog v = new Keypad(this, tiles, puzzleView);
        v.show();
    }
}
```

Для того чтобы решить, какие цифры можно выбирать, мы передаем конструктору класса Keypad массив, содержащий все цифры, которые уже были использованы.

Создание игровой логики

Оставшаяся часть кода в Game.java отвечает за игровую логику, в частности за определение верных и неверных ходов в соответствии с правилами игры. Метод setTileIfValid() — ее ключевая часть. По заданным координатам x и y и новому значению для тайла он изменяет тайл лишь в том случае, если представленное значение допустимо.

Sudokuv2/src/org/example/sudoku/Game.java

```
protected boolean setTileIfValid(int x, int y, int value) {
    int tiles[] = getUsedTiles(x, y);
    if (value != 0) {
        for (int tile : tiles) {
            if (tile == value)
                return false;
        }
    }
    setTile(x, y, value);
    calculateUsedTiles();
    return true;
}
```

Для определения правильных ходов мы создадим массив для каждого тайла в таблице. Для каждой позиции массив хранит список заполненных тайлов, которые видимы в настоящее время. Если число появляется в списке, это значит, что оно не подходит для текущего тайла. Метод getUsedTiles() получает этот список для заданной позиции тайла.

Sudokuv2/src/org/example/sudoku/Game.java

```
private final int used[][][] = new int[9][9][];
protected int[] getUsedTiles(int x, int y) {
    return used[x][y];
}
```

Массив использованных тайлов рассчитывать довольно накладно, поэтому мы кэшируем этот массив и пересчитываем его лишь тогда, когда это нужно, вызывая метод calculateUsedTiles().

Sudokuv2/src/org/example/sudoku/Game.java

```
private void calculateUsedTiles() {
    for (int x = 0; x < 9; x++) {
        for (int y = 0; y < 9; y++) {
            used[x][y] = calculateUsedTiles(x, y);
            // Log.d(TAG, "used[" + x + "][" + y + "] = "
            // + toPuzzleString(used[x][y]));
        }
    }
}
```

Метод calculateUsedTiles() просто вызывает calculateUsedTiles(x, y) для каждой позиции решетки девять на девять:

Sudokuv2/src/org/example/sudoku/Game.java

```
1     private int[] calculateUsedTiles(int x, int y) {
-             int c[] = new int[9];
-             // горизонтальная
-             for (int i = 0; i < 9; i++) {
5                 if (i == x)
-                     continue;
-                 int t = getTile(i, y);
-                 if (t != 0)
-                     c[t - 1] = t;
10            }
-             // вертикальная
-             for (int i = 0; i < 9; i++) {
-                 if (i == y)
-                     continue;
15                int t = getTile(x, i);
-                 if (t != 0)
-                     c[t - 1] = t;
-             }
-             // та же клетка блока
20            int startx = (x / 3) * 3;
```

```
-             int starty = (y / 3) * 3;
-             for (int i = startx; i < startx + 3; i++) {
-                 for (int j = starty; j < starty + 3; j++) {
-                     if (i == x && j == y)
25                       continue;
-                     int t = getTile(i, j);
-                     if (t != 0)
-                         c[t - 1] = t;
-                 }
30            }
-             // сжатие
-             int nused = 0;
-             for (int t : c) {
-                 if (t != 0)
35                   nused++;
-             }
-             int c1[] = new int[nused];
-             nused = 0;
-             for (int t : c) {
40                if (t != 0)
-                     c1[nused++] = t;
-             }
-             return c1;
-     }
```

Мы начинаем с массива из девяти нулей. В строке 4 мы проверяем все тайлы, принадлежащие тому же горизонтальному ряду, что и текущий, и, если тайл не пустой, мы сохраняем его цифру в массиве.

В строке 12 мы делаем то же самое для всех тайлов вертикальной строки, в строке 20 — то же самое для тайлов блока три на три.

Последний шаг, который начинается в строке 32, — это сжатие нулей в массиве перед его возвращением. Мы делаем это для того, чтобы команда array.length могла быть использована для быстрого определения количества заполненных тайлов, которые видимы в текущей позиции.

Разное

Для завершения реализации нам понадобятся еще несколько служебных функций и переменных. easyPuzzle, mediumPuzzle и hardPuzzle — наши закодированные головоломки Sudoku для простого, среднего и высокого уровней сложности соответственно.

Sudokuv2/src/org/example/sudoku/Game.java

```
private final String easyPuzzle =
    "360000000004230800000004200" +
    "070460003820000014500013020" +
    "001900000007048300000000045" ;
private final String mediumPuzzle =
```

```
    "650000070000506000014000005" +
    "007009000002314700000700800" +
    "500000630000201000030000097" ;
private final String hardPuzzle =
    "009000000080605020501078000" +
    "000000700706040102004000000" +
    "000720903090301080000000600" ;
```

Команда getPuzzle() просто получает уровень сложности и возвращает соответствующую головоломку.

Sudokuv2/src/org/example/sudoku/Game.java

```
private int[] getPuzzle(int diff) {
    String puz;
    // Нужно сделать: продолжение предыдущей игры
    switch (diff) {
    case DIFFICULTY_HARD:
        puz = hardPuzzle;
        break;
    case DIFFICULTY_MEDIUM:
        puz = mediumPuzzle;
        break;
    case DIFFICULTY_EASY:
    default:
        puz = easyPuzzle;
        break;
    }
    return fromPuzzleString(puz);
}
```

Позже мы изменим метод getPuzzle() для того, чтобы реализовать возможность продолжения игры.

Метод toPuzzleString() конвертирует игру из массива целых чисел в строку. fromPuzzleString() выполняет обратное преобразование.

Sudokuv2/src/org/example/sudoku/Game.java

```
static private String toPuzzleString(int[] puz) {
    StringBuilder buf = new StringBuilder();
    for (int element : puz) {
        buf.append(element);
    }
    return buf.toString();
}
static protected int[] fromPuzzleString(String string) {
    int[] puz = new int[string.length()];
    for (int i = 0; i < puz.length; i++) {
        puz[i] = string.charAt(i) - '0' ;
    }
    return puz;
}
```

Метод getTile() принимает в качестве параметров x и y и возвращает число, которое сейчас находится в тайле с этой позицией. Если это ноль, значит, тайл пуст.

Sudokuv2/src/org/example/sudoku/Game.java

```
private int getTile(int x, int y) {
    return puzzle[y * 9 + x];
}
private void setTile(int x, int y, int value) {
    puzzle[y * 9 + x] = value;
}
```

Метод getTileString() используется при отображении тайла. Он возвращает строку со значением тайла или пустую строку, если тайл пуст.

Sudokuv2/src/org/example/sudoku/Game.java

```
protected String getTileString(int x, int y) {
    int v = getTile(x, y);
    if (v == 0)
        return "" ;
    else
        return String.valueOf(v);
}
```

Только тогда, когда все эти части станут одним целым, вы получите игру Sudoku, в которую можно играть. Попытайтесь проверить работу программы. Как и в любом другом коде, в нашем есть возможности для его оптимизации.

4.5. Дополнительные улучшения

Хотя код, представленный в этой главе, вполне подходит для игры Sudoku, более сложные программы нуждаются в более аккуратном кодировании в смысле выжимания из устройства максимума производительности. В особенности весьма критичен в плане производительности метод onDraw(), поэтому в нем нужно выполнять как можно меньше операций.

Вот некоторые идеи, касающиеся того, как можно ускорить этот метод:

- Если возможно, избегайте выполнять создание объектов в методе onDraw().
- Задавайте различные значения, например цветовые константы, в другом месте (например, в конструкторе вьювера).
- Создавайте объекты Paint заранее и используйте существующие экземпляры в onDraw().
- Для значений, которые используются много раз, таких как ширина экрана, которую возвращает getWidth(), получайте значения в начале работы метода и затем используйте их локальные копии.

В качестве дополнительного упражнения я предлагаю подумать о том, как улучшить графическую составляющую игры Sudoku. Например, можно добавить небольшой фейерверк, если игрок соберет головоломку или развернет тайлы, как

это делает Ванна Уайт (Vanna White). Анимированный фон головоломки также может быть интересным. Дайте волю своему воображению. Если вы хотите сделать программу высшего качества, любые мелочи, как эта, могут добавить изюминку к программе, которая иначе была бы самой обычной.

В главе 5 «Мультимедиа» мы расширим программу музыкальным сопровождением, а в главе 6 «Хранение локальных данных» рассмотрим сохранение состояния игры и в итоге, создадим кнопку Continue (Продолжить).

4.6. Вперед >>

В этой главе мы лишь слегка коснулись графических возможностей Android. «Родная» 2D-библиотека весьма обширна, поэтому при написании программ пользуйтесь всплывающими подсказками, автозавершением и Javadoc, которые предоставляет плагин Android для Eclipse. Онлайновая документация по пакету android.graphics[1] предоставит вам дополнительную информацию.

Если вашей программе нужна более совершенная графика, вы, может быть, захотите забежать вперед и прочитать главу 10 «3D графика в OpenGL». Там вы найдете информацию о том, как использовать трехмерную графическую библиотеку Android, которая основана на стандарте OpenGL ES. В противном случае переходите к следующей главе, где вы познакомитесь с удивительным миром видео и музыки в Android.

[1] http://d.android.com/reference/android/graphics/package-summary.html

5 Мультимедиа

Помните телевизионную рекламу Apple с силуэтами людей, которые безудержно танцуют под звуки своих iPod'ов? Вы наверняка хотели бы, чтобы ваша программа оказывала аналогичное воздействие[1]. Музыка, звуковые эффекты и видео могут сделать программу более увлекательной и притягательной, чем если бы она содержала только простой текст и изображения.

Эта глава покажет, как добавить мультимедийные возможности к вашему Android-приложению. Вы, скорее всего, не заставите пользователей прыгать по комнате, но если все будет сделано правильно, вы, как минимум, заставите их улыбнуться.

5.1. Проигрывание аудио

Была темная грозовая ночь... Прогремел стартовый выстрел, и они начали... Толпа неистовствовала, когда Стейт забил трехочковый на последней секунде.

Звуки пронизывают все вокруг и задают тон нашим эмоциям. Думайте о звуке как о еще одном пути к умам пользователей. Так же как при использовании графики на дисплее, чтобы донести какую-нибудь информацию до пользователя, вы можете использовать аудио для усиления и закрепления произведенного впечатления.

Android поддерживает вывод звуков и музыки посредством класса MediaPlayer в пакете android.media[2]. Протестируем его на простом примере, который воспроизводит звуки при нажатии кнопки на клавиатуре или на джойстике.

Мы начнем с создания проекта «Hello, Android», используя следующие параметры в диалоговом окне New Android Project:

---

[1] Конечно, нормальные люди старше восьми лет вряд ли смогут так танцевать... за исключением, возможно, того случая, когда мои дети засунули ящерицу мне в... хотя ладно, я отвлекся.

[2] http://d.android.com/guide/topics/media

```
Project name: Audio
Build Target: Android 2.2
Application name: Audio
Package name: org.example.audio
Create Activity: Audio
Min SDK Version: 8
```

Далее, нам понадобятся несколько звуков. Для этого примера я создал собственные звуки с помощью программы Sound Recorder (Звукозапись) (Start ▸ All Programs ▸ Accessories ▸ Sound Recorder (Пуск ▸ Все программы ▸ Стандартные ▸ Звукозапись)) в Windows 7 и недорогой гарнитуры. После установки подходящего уровня звука я записал каждый аудиофрагмент, выбрал File ▸ Save as... (Файл ▸ Сохранить как...) в меню, щелкнул кнопкой Change (Изменить) и выбрал формат файла, который поддерживается Android (рис. 5.1). На веб-сайте этой книги можно найти звуковые файлы и исходный код этого примера.

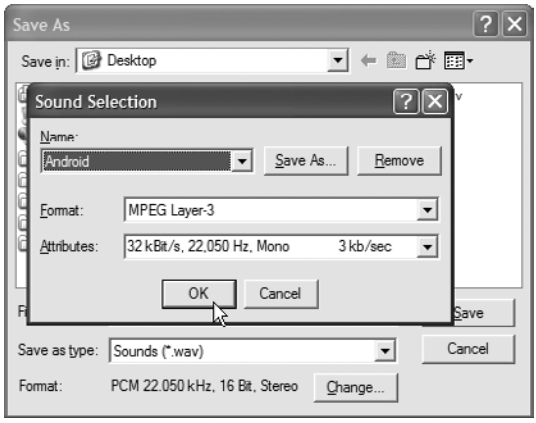

Рис. 5.1. Сохранение звуковых эффектов в сжатом формате, который может проигрывать Android

Скопируйте звуковые файлы в папку res/raw вашего проекта. Как вы помните из раздела 2.4. «Использование ресурсов», простое копирование файлов в папку res приводит к тому, что плагин Android Eclipse производит соответствующие изменения в Java-коде класса *R*. После того как вы все сделаете, проект должен выглядеть так, как показано на рис. 5.2.

Сейчас пришло время настроить деятельность Audio. Для начала мы объявим поле, которое называется mp, для хранения экземпляра класса MediaPlayer. В этой программе мы собираемся работать в каждый момент времени с одним экземпляром MediaPlayer.

Audio/src/org/example/audio/Audio.java

```
package org.example.audio;
import android.app.Activity;
import android.media.AudioManager;
import android.media.MediaPlayer;
```

```
import android.os.Bundle;
import android.view.KeyEvent;
public class Audio extends Activity {
    private MediaPlayer mp;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        setVolumeControlStream(AudioManager.STREAM_MUSIC);
    }
}
```

Метод setVolumeControlStream() сообщает Android, что когда пользователь нажимает клавиши для увеличения или уменьшения громкости во время выполнения приложения, оно должно изменять громкость музыки или других медиаматериалов, вместо того чтобы изменять громкость звонка.

Далее, нам нужно перехватить нажатие клавиш и проиграть подходящие звуки. Мы сделаем это, переопределив метод Activity.onKeyDown().

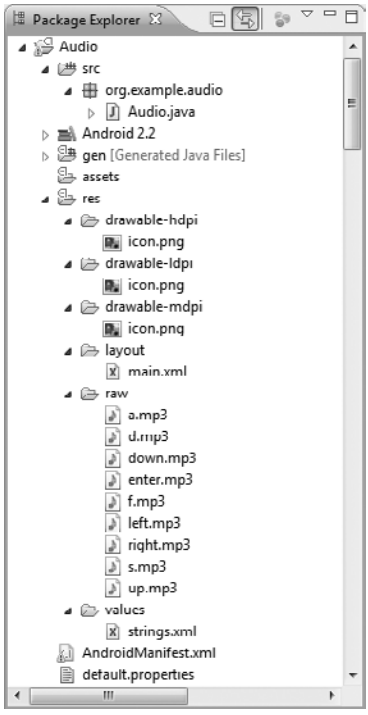

Рис. 5.2. Копирование аудиофайлов в папку res/raw вашего проекта

Audio/src/org/example/audio/Audio.java

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

```
-         int resId;
-         switch (keyCode) {
5         case KeyEvent.KEYCODE_DPAD_UP:
-             resId = R.raw.up;
-             break;
-         case KeyEvent.KEYCODE_DPAD_DOWN:
-             resId = R.raw.down;
10            break;
-         case KeyEvent.KEYCODE_DPAD_LEFT:
-             resId = R.raw.left;
-             break;
-         case KeyEvent.KEYCODE_DPAD_RIGHT:
15            resId = R.raw.right;
-             break;
-         case KeyEvent.KEYCODE_DPAD_CENTER:
-         case KeyEvent.KEYCODE_ENTER:
-             resId = R.raw.enter;
20            break;
-         case KeyEvent.KEYCODE_A:
-             resId = R.raw.a;
-             break;
-         case KeyEvent.KEYCODE_S:
25            resId = R.raw.s;
-             break;
-         case KeyEvent.KEYCODE_D:
-             resId = R.raw.d;
-             break;
30        case KeyEvent.KEYCODE_F:
-             resId = R.raw.f;
-             break;
-         default:
-             return super.onKeyDown(keyCode, event);
35        }
-
-         // Высвобождение ресурсов от предыдущих вызовов MediaPlayer
-         if (mp != null) {
-             mp.release();
40        }
-
-         // Создание нового объекта MediaPlayer для проигрывания этого звука
-         mp = MediaPlayer.create(this, resId);
-         mp.start();
45
-         // Указание того, что нажатие данной клавиши было обработано
-         return true;
-     }
```

Первая часть кода метода занимается выбором ресурса в зависимости от того, какая клавиша была нажата. Затем, в строке 39 мы используем метод release(), чтобы остановить любые звуки, которые уже проигрываются, и освободить любые

ресурсы, связанные со старым экземпляром MediaPlayer. Если вы забыли сделать это, программа завершится с ошибкой (смотрите врезку).

В строке 43 мы используем метод create(), чтобы создать новый экземпляр MediaPlayer, используя выбранный звуковой ресурс, и вызываем метод start(), чтобы начать его проигрывать. Метод start() асинхронен, он возвращает управление немедленно, независимо от того, как долго длится звук. Вы можете использовать метод setOnCompletionListener(), чтобы узнать, когда завершится проигрывание клипа.

КОГДА ПРОИСХОДЯТ ОШИБКИ

Занимаясь мультимедийным программированием какое-то время, вы, возможно, обнаружите, что MediaPlayer Android иногда ведет себя неустойчиво. Его реализация в новых версиях Android улучшена по сравнению с предшественниками, но он все еще может завершиться с ошибкой от малейшей неточности. Одна из причин, по которой это происходит, заключается в том, что MediaPlayer — это преимущественно системное приложение с небольшой прослойкой Java-кода поверх него. Родной системный код проигрывателя оптимизирован для целей производительности и не содержит разветвленной системы контроля ошибок.

К счастью, жесткий контроль Android за процессами Linux предотвращает повреждения системы, которые могли бы стать последствием «падения» плеера. Эмулятор (или телефон, если вы запускаете программу на реальном устройстве) и другие приложения продолжают нормально работать. Пользователь же увидит, что его приложение завершилось, возможно, с выводом диалогового окна, содержащего информацию об ошибке.

Однако при разработке вы можете получить гораздо больше диагностической информации, которая поможет понять, что пошло не так. Сообщения и результаты отслеживания ошибок будут выведены в системный журнал Android, который доступен в окне LogCat в Eclipse. Также можно воспользоваться командой addb logcat (см. раздел 3.10, «Отладка с помощью записи сообщений в журнал»).

ВОПРОС/ОТВЕТ

Какие форматы аудио поддерживает Android?

Итак, есть заявленная на бумаге поддержка и есть поддержка форматов на эмуляторе и проигрывание на реальном устройстве. Если говорить о заявленных в документации форматах, поддерживаемых Android, то это следующие типы аудиофайлов (этот список подвержен изменениям в новых выпусках платформы):

- WAV (Несжатый формат PCM — Pulse Code Modulation);
- AAC (Формат Apple iPod, незащищенный);
- MP3 (MPEG-3);
- WMA (Windows media audio);
- AMR (Речевой кодек);
- OGG (Ogg vorbis);[1]
- MIDI (Цифровые ноты, только инструменты).

В реальности оказывается, что только файлы форматов OGG, WAV и MP3 нормально работают на эмуляторе, и лишь их я могу порекомендовать для разработки приложений. Исходный аудиоформат Android — это 16-битное стерео с частотой дискретизации в 44,1 кГц. Однако так как битрейт WAV-файлов довольно велик, придерживайтесь форматов OGG или MP3 (монофонических — для голоса, стереофонических — для музыки). OGG-файлы работают лучше для коротких клипов вроде игровых звуковых эффектов.

Держитесь подальше от необычных битрейтов вроде 8 кГц, так как артефакты ресэмплинга делают воспроизведение таких звуков просто ужасным. Используйте частоту

[1] http://www.vorbis.com

дискретизации 11 кГц, 22 кГц или 44,1 кГц для наилучших результатов. Помните, что хотя телефон может иметь маленький динамик, многие из ваших пользователей пользуются наушниками (такими, как в iPod), поэтому логично, чтобы ваше аудио проигрывалось в высоком качестве.

Если сейчас вы запустите программу и затем нажмете одну из кнопок (например, кнопку Enter в центре D-пада), вы должны услышать звук. Если этого не происходит, проверьте регулятор громкости (не смейтесь) или посмотрите отладочные сообщения в окне LogCat. Если вы запускаете программу на телефоне без клавиатуры, D-пада или трекбола, нажмите и удерживайте кнопку Menu, чтобы вызвать экранную клавиатуру.

Обратите внимание на то, что в некоторых случаях вывод звука может сопровождаться помехами или задержками. Попробуйте различные форматы звуковых файлов (например, OGG вместо MP3) и более низкий битрейт. Вы также можете попробовать использовать класс SounPool, который хорошо поддерживает одновременное проигрывание нескольких аудиопотоков. Он полон ошибок и плохо документирован в релизе 1.0., но уже в 1.5. выглядит достаточно стабильным.

Наш следующий трюк будет заключаться в том, чтобы проиграть видеоклип, воспользовавшись лишь одной строкой программного кода.

ВОПРОС/ОТВЕТ

Какие форматы видео можно просматривать в Android?

Вот что официально поддерживается:

- MP4 (MPEG-4, низкий битрейт);
- H.263 (3GP);
- H.264 (AVC).

В Android 1.5 H.263 является рекомендованным видеоформатом, так как его поддерживают все аппаратные платформы, и он достаточно эффективен при кодировании и декодировании. Он также совместим с другими устройствами, такими как iPhone. Используйте программу типа QuickTime Pro[1], чтобы конвертировать видео из одного формата в другой. Выбирайте как можно более низкое разрешение и битрейт в целях экономии места, однако не делайте их слишком низкими, так как это может серьезно повредить качеству.

5.2. Проигрывание видео

Видео — это больше, чем просто набор сменяющих друг друга картинок. Оно содержит звуковую дорожку, и звук должен быть точно синхронизирован с изображением.

Класс Android MediaPlayer работает с видео так же, как и с обычным аудио. Разница заключается лишь в том, что вам нужно создать Surface (поверхность), которую плеер будет использовать для того, чтобы создавать изображения. Используйте методы start() и stop() для управления воспроизведением.

[1] http://www.apple.com/quicktime/pro

Однако я не собираюсь приводить здесь еще один пример MediaPlayer, поскольку есть более простой способ вставлять видео в приложение — использовать класс VideoView. Для того чтобы продемонстрировать этот способ, создадим новый проект Android, который назовем Video, используя следующие параметры:

```
Project name: Video
Build Target: Android 2.2
Application name: Video
Package name: org.example.video
Create Activity: Video
Min SDK Version: 8
```

Измените макет (res/layout/main.xml) следующим образом:

Videov1/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" >
    <VideoView
        android:id="@+id/video"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_gravity="center" />
</FrameLayout>
```

Откройте файл Video.java и измените метод onCreate:

Videov1/src/org/example/video/Video.java:

```
package org.example.video;
import android.app.Activity;
import android.os.Bundle;
import android.widget.VideoView;
public class Video extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Заполнение окна просмотра из ресурса
        setContentView(R.layout.main);
        VideoView video = (VideoView) findViewById(R.id.video);
        // Загрузка и начало воспроизведения видеофайла
        video.setVideoPath("/data/samplevideo.3gp" );
        video.start();
    }
}
```

Метод setVideoPath() открывает файл, масштабирует его по размерам контейнера, сохраняя соотношение сторон, и начинает его проигрывать.

Сейчас нам нужно загрузить в проект что-нибудь для проигрывания. Для того чтобы сделать это, выполните (в Windows) следующую команду[1]:

[1] Это команда adb в папке SDK platform-tools.

```
C:\> adb push c:\code\samplevideo.3gp /data/samplevideo.3gp
1649 KB/s (369870 bytes in 0.219s)
```

Рис. 5.3. Видео очень просто встроить с помощью VideoView

Найдите файл samplevideo.3gp на сайте издательства «Питер» (www.piter.com) в архиве с дополнительными материалами или самостоятельно создайте собственный видеофайл. Папка (/data) использована здесь лишь для демонстративных целей, ее не следует использовать для хранения медиафайлов. В данной конфигурации пример будет работать лишь на эмуляторе, так как на реальных устройствах эта папка защищена.

Помните, что Android не обращает внимания на то, какое расширение вы даете файлу. Также вы можете загружать и выгружать файлы в Eclipse, используя окно File Explorer в виде Android, однако я нахожу использование командной строки более удобным для простых задач типа этой.

И еще одно: мы хотим, чтобы видео занимало весь экран, включая заголовок окна и строку состояния. Для того чтобы сделать это, достаточно задать подходящую тему в AndroidManifest.xml.

Videov1/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.example.video"
    android:versionCode="1"
    android:versionName="1.0" >
    <application android:icon="@drawable/icon"
            android:label="@string/app_name" >
        <activity android:name=".Video"
            android:label="@string/app_name"
            android:theme="@android:style/Theme.NoTitleBar.Fullscreen" >
            <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>
```

Если все было сделано правильно, после запуска программы вы должны увидеть и услышать видеоклип (рис. 5.3). Попробуйте повращать экран, чтобы удостовериться в том, что все воспроизводится и в портретном и в ландшафтном режимах. Вуаля! Как быстро мы справились.

Сейчас давайте доработаем пример Sudoku, добавим в него немного музыки для настроения.

5.3. Добавление звуков в Sudoku

В этом разделе мы собираемся воспользоваться полученными знаниями, чтобы добавить фоновую музыку в разрабатываемую нами игру Sudoku. Одна композиция будет играть при отображении стартового экрана, вторая — в процессе игры. Этот пример предназначен не только для того, чтобы показать, как проигрывать музыку, он затрагивает некоторые важные принципы, касающиеся жизненного цикла программы.

ВОПРОС/ОТВЕТ

Почему воспроизведение видео начинается сначала, когда я поворачиваю экран?

В Android подразумевается по умолчанию, что ваша программа ничего не знает о поворотах экрана. Для того чтобы обработать возможные изменения ресурсов, Android уничтожает и пересоздает вашу деятельность из кода. Это подразумевает повторный вызов onCreate(), что, в свою очередь, приводит к повторному началу воспроизведения видео (именно так написан этот пример).

Такое поведение вполне оправданно для 90% приложений, поэтому большинство разработчиков об этом не беспокоятся. Это еще и неплохой способ для тестирования жизненного цикла вашего приложения и кода сохранения и восстановления его состояния (см. раздел 2.2 «Оно живое!»). Однако можно поступить умнее и оптимизировать отработку изменения состояния экрана.

Самый простой способ — применить в вашей деятельности метод onRetainNonConfigurationInstance() для сохранения данных между вызовами onDestroy() и onCreate(). Когда программа начинает исполнение снова, вы используете getLastNonConfigurationInstance() в новом экземпляре деятельности для того, чтобы восстановить эту информацию. Вы можете сохранить все, включая ссылки на текущее намерение и выполняющиеся процессы.

Более сложный путь заключается в использовании свойства android:configChanges= в файле AndroidManifest.xml, чтобы сообщить Android, какие изменения вы хотите поддерживать. Например, если вы установите его в значение keyboardHidden|orientation, Android не будет уничтожать и восстанавливать вашу деятельность, когда пользователь откроет клавиатуру. Вместо этого он вызовет метод onConfigurationChanged(Configuration) и будет полагать, что вы знаете что делаете[1].

[1] Подробности вы можете найти на странице http://d.android.com/reference/android/app/Activity.html#ConfigurationChanges

Чтобы добавить музыку к стартовому экрану, нам следует переопределить два метода в классе Sudoku:

Sudokuv3/src/org/example/sudoku/Sudoku.java

```
@Override
protected void onResume() {
    super.onResume();
    Music.play(this, R.raw.main);
}
@Override
protected void onPause() {
    super.onPause();
    Music.stop(this);
}
```

Если вы вспомните раздел 2.2 «Оно живое!», метод onResume() вызывается, когда деятельность готова к взаимодействию с пользователем. Это отличное место для начала проигрывания музыки, поэтому мы помещаем вызов Music.play() здесь. Скоро мы определим класс Music.

ВОПРОС/ОТВЕТ

Нужно ли нам использовать фоновые сервисы (Background Service) для проигрывания музыки?

Мы не рассказали подробно о классе Android Service, но вы можете увидеть его использование в некоторых примерах, содержащих проигрывание музыки, в Интернете. В основном Service — это способ запуска фоновых процессов, которые могут работать даже тогда, когда ваша текущая деятельность завершится. Сервисы похожи на демоны Linux, однако это не одно и то же. Если вы пишете обычный музыкальный проигрыватель и хотите, чтобы воспроизведение музыки продолжалось, когда вы читаете почту или просматриваете веб-страницы, тогда, конечно, Service отлично подойдет. В большинстве же случаев необходимо завершить проигрывание музыки, когда программа закончит работу, поэтому вам не нужно использовать класс Service.

R.raw.main ссылается на res/raw/main.mp3. Найдите соответствующие звуковые файлы в проекте Sudokuv3 в загружаемых примерах на сайте издательства.

Метод onPause() — это двойной фиксатор для метода onResume(). Android приостанавливает текущую деятельность, прежде чем начать новую: так, в Sudoku, когда вы начинаете новую игру, деятельность Sudoku приостанавливается, и затем запускается деятельность Game(). Метод onPause() также вызывают, когда пользователь нажимает кнопку Back или Home. Все это — места, где мы хотели бы остановить нашу музыку, поэтому мы вызываем метод Music.stop() в onPause().

Теперь давайте сделаем что-то в этом роде с музыкой в деятельности Game:

Sudokuv3/src/org/example/sudoku/Game.java

```
@Override
protected void onResume() {
    super.onResume();
    Music.play(this, R.raw.game);
}
```

```
@Override
protected void onPause() {
    super.onPause();
    Music.stop(this);
}
```

НЕМНОГО О SUDOKU

Существуют десятки вариантов Sudoku, хотя ни один из них не сумел превзойти по популярности оригинал. Существует вариант, в котором используются таблицы размером шестнадцать на шестнадцать, с шестнадцатеричными цифрами. Другой, называемый Gattai 5 или Samurai Sudoku, использует пять таблиц девять на девять, которые перекрываются в угловых областях.

Если вы сравните это с тем, что мы делали в классе Sudoku, то заметите, что мы ссылаемся на другой звуковой ресурс, R.raw.game (res/raw/game.mp3).

Финальный фрагмент музыкального пазла — это класс Music, который будет управлять классом MediaPlayer, использованным для проигрывания текущего звукового сопровождения:

Sudokuv3/src/org/example/sudoku/Music.java

```
1     package org.example.sudoku;
-
-         import android.content.Context;
-         import android.media.MediaPlayer;
5
-         public class Music {
-             private static MediaPlayer mp = null;
-
-             /** Остановка старой композиции и начало новой */
10            public static void play(Context context, int resource) {
-                 stop(context);
-                 mp = MediaPlayer.create(context, resource);
-                 mp.setLooping(true);
-                 mp.start();
15            }
-
-             /** Остановка проигрывания музыки */
-             public static void stop(Context context) {
-                 if (mp != null) {
20                    mp.stop();
-                     mp.release();
-                     mp = null;
-                 }
-             }
25        }
```

Метод play() первоначально вызывает метод stop() для остановки любой музыки, которая сейчас проигрывается. Далее мы создаем новый экземпляр MediaPlayer, используя MediaPlayer.create(), передавая контекст и ID ресурса.

После того, как у нас появился проигрыватель, мы устанавливаем параметр, который включает циклическое повторение музыки, и начинаем воспроизведение. Метод start() мгновенно начинает работать.

Метод stop(), код которого начинается в строке 18, прост. После небольшой защитной проверки, которая позволяет нам убедиться, что у нас действительно есть объект MediaPlayer, с которым мы будем работать, мы вызываем его методы stop() и release(). Метод MediaPlayer.stop()останавливает воспроизведение музыки (а вы чего ожидали?). Метод release() освобождает системные ресурсы, связанные с проигрывателем. Так как это родные ресурсы, мы не можем просто ждать, пока обычный сборщик мусора Java их утилизирует. Отказ от использования release() — это прекрасный способ добиться того, чтобы ваша программа неожиданно падала из-за непонятных ошибок (со мной, конечно, такого не случалось; я просто говорю, чтобы вы помнили об этом).

А вот и время для забавы — попробуйте поиграть в Sudoku с внесенными изменениями. Протестируйте ее любыми способами, которые сможете себе вообразить, такими как переключение между разными деятельностями, нажатие кнопок Back и Home в различные моменты игры, запуск игры снова, когда она уже запущена и находится на различных стадиях исполнения, поворот дисплея и так далее. Соответствующее управление жизненным циклом программы иногда превращается в кошмар, однако ваши пользователи оценят усилия.

5.4. Вперед >>

В этой главе мы рассмотрели воспроизведение аудио- и видеоклипов с использованием Android SDK. Мы не говорили о записи, так как большинство программ не нуждаются в этом, но если вам случиться быть исключением, посмотрите описание класса MediaRecorder в онлайновой документации[1].

В главе 6 «Хранение локальных данных» вы узнаете о некоторых простых способах, с помощью которых программы на Android могут хранить данные между вызовами. Если вам это не нужно, переходите к главе 7 «Объединенный мир», где рассказывается о доступе в Сеть.

[1] http://d.android.com/reference/android/media/MediaRecorder.html

6 Хранение локальных данных

До сих пор мы пытались написать (и писали) приложения, которые не нуждаются в хранении данных при выходе из них. Они загружаются, запускаются и завершают работу, не оставляя после себя никаких следов. Однако большая часть реальных программ нуждается в сохранении устойчивых состояний, будь это простая установка размера шрифта, пикантное фото с последнего корпоратива или план здорового питания на следующую неделю. Что бы это ни было, Android позволяет постоянно хранить данные на мобильном устройстве для последующего использования и защищает их от случайного или злонамеренного доступа других программ.

Ваше приложение может хранить данные, используя несколько различных способов в зависимости от размера данных, их структуры, времени хранения и того, будут ли они использоваться другими программами. В этой главе мы разберем три простых способа хранения локальных данных: предустановки API, сохранение состояния объекта и хранение файлов во флэш-памяти. В главе 9 «Работа с SQL» мы изучим более совершенные способы, которые используют встроенную систему управления базами данных SQLite.

6.1. Добавление пункта Options в Sudoku

В разделе 3.7 «Добавление меню» мы использовали метод onCreateOptionsMenu() для добавления меню, содержащего один элемент, на главный экран Sudoku. Когда пользователь нажимает клавишу Menu и выбирает пункт Settings..., из кода запускается деятельность Prefs, которая позволяет пользователю менять параметры игры. Так как Prefs расширяет PreferenceActivity, значения настроек хранятся в области предустановок программы, но изначально мы ничего с ними не делали. Сейчас мы собираемся с ними поработать.

НЕМНОГО О SUDOKU

Существует 6 670 903 752 021 072 936 960 возможных вариантов решения классического судоку. Если исключить повторения, которые являются простыми поворотами таблицы решений, отражениями, перенумерацией и т. д., вы получите «всего лишь» 5 472 730 538 решений.

Для начала давайте изменим класс Prefs, добавив пару методов, получающих данные, которые восстанавливают текущие значения для наших двух параметров. Вот его новое определение:

Sudokuv4/src/org/example/sudoku/Prefs.java

```
package org.example.sudoku;
import android.content.Context;
import android.os.Bundle;
import android.preference.PreferenceActivity;
import android.preference.PreferenceManager;
public class Prefs extends PreferenceActivity {
    // Названия параметров и значения по умолчанию
    private static final String OPT_MUSIC = "music" ;
    private static final boolean OPT_MUSIC_DEF = true;
    private static final String OPT_HINTS = "hints" ;
    private static final boolean OPT_HINTS_DEF = true;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.settings);
    }
    /** Получить текущее значение для музыкального параметра */
    public static boolean getMusic(Context context) {
        return PreferenceManager.getDefaultSharedPreferences(context)
            .getBoolean(OPT_MUSIC, OPT_MUSIC_DEF);
    }
    /** Получить текущее значение для параметра подсказок */
    public static boolean getHints(Context context) {
        return PreferenceManager.getDefaultSharedPreferences(context)
            .getBoolean(OPT_HINTS, OPT_HINTS_DEF);
    }
}
```

Обратите внимание на то, чтобы ключи параметров (музыка и подсказки) совпадали с ключами, использованными в res/xml/settings.xml.

Метод Music.play() был модифицирован для проверки музыкальной предустановки:

Sudokuv4/src/org/example/sudoku/Music.java

```
public static void play(Context context, int resource) {
    stop(context);
    // Начать проигрывать музыку только в том случае, если это не выключено
в настройках
    if (Prefs.getMusic(context)) {
        mp = MediaPlayer.create(context, resource);
        mp.setLooping(true);
        mp.start();
    }
}
```

И метод PuzzleView.onDraw() также нуждается в модификации для проверки настроек подсказок.

Sudokuv4/src/org/example/sudoku/PuzzleView.java

```
if (Prefs.getHints(getContext())) {
    // Рисование подсказок...
}
```

Если getHints() возвратит истину, мы будем рисовать подсказки, как показано на рис. 4.6. В противном случае мы просто пропустим эту часть кода.

Далее я покажу, как использовать предустановки API для того, чтобы хранить кое-что помимо настроек.

6.2. Продолжение старой игры

В любое время игрок может решить выйти из игры Sudoku и заняться чем-то еще. Возможно, мимо проходит его руководитель, а может быть, ему позвонили по телефону или сработало напоминание о важной встрече. Какой бы ни была причина, мы хотим позволить игроку вернуться к игре позже и продолжить ее с того места, на котором его прервали.

Для начала нам нужно где-нибудь сохранить текущее состояние головоломки. Предустановки API можно использовать не только для хранения параметров игры; они могут хранить любые небольшие отдельные блоки информации, которая имеет отношение к вашей программе. В данном случае состояние головоломки может быть сохранено в виде строки из девяноста одного символа, по одному на каждую клетку.

В классе Game мы начнем с определения пары констант: одной — для данных головоломки и одной — флага, который позволит нам определить, следует ли продолжать предыдущую игру или начинать новую.

Sudokuv4/src/org/example/sudoku/Game.java

```
private static final String PREF_PUZZLE = "puzzle" ;
protected static final int DIFFICULTY_CONTINUE = -1;
```

Далее нам нужно сохранить текущую головоломку, в какой бы момент игра ни была приостановлена. Смотрите раздел 2.2 «Оно живое!», чтобы найти описание метода onPause() и других методов жизненного цикла программы.

Sudokuv4/src/org/example/sudoku/Game.java

```
@Override
protected void onPause() {
    super.onPause();
    Log.d(TAG, "onPause" );
    Music.stop(this);
    // Сохранение текущей головоломки
    getPreferences(MODE_PRIVATE).edit().putString(PREF_PUZZLE,
        toPuzzleString(puzzle)).commit();
}
```

Сейчас головоломка сохранена, но как нам прочитать сохраненные данные? Помните, что когда начинается игра, вызывается метод getPuzzle() и в качестве параметра передается уровень сложности. Мы воспользуемся этим механизмом и для продолжения игры.

Sudokuv4/src/org/example/sudoku/Game.java

```
private int[] getPuzzle(int diff) {
    String puz;
    switch (diff) {
    case DIFFICULTY_CONTINUE:
        puz = getPreferences(MODE_PRIVATE).getString(PREF_PUZZLE,
            easyPuzzle);
        break;
        // ...
    }
    return fromPuzzleString(puz);
}
```

Все, что нам нужно, — это добавить проверку флага DIFFICULTY_CONTINUE. Если он установлен, тогда, вместо того чтобы начинать новую игру, мы прочитаем данные той, которую сохранили в предустановках.

Далее нам нужно заставить что-нибудь делать кнопку Continue на главном экране (см. рис. 3.4.). Вот где мы это сделаем:

Sudokuv4/src/org/example/sudoku/Sudoku.java

```
public void onClick(View v) {
    switch (v.getId()) {
    case R.id.continue_button:
        startGame(Game.DIFFICULTY_CONTINUE);
        break;
        // ...
    }
}
```

Мы добавили ветвь case в Sudoku.onClick() для вызова startGame(). При нажатии кнопка Continue. startGame() передает параметр, содержащий сложность игры, в деятельность Game, и Game.onCreate() вызывает Intent.getIntExtra() для чтения уровня сложности и передачи его в getPuzzle() (вы можете увидеть код, делающий это, в разделе 4.2, «Начало игры»).

Осталось сделать лишь одно: восстановить игру из сохраненной информации, когда пользователь переключается между деятельностями (например, когда во время игры стартует другая деятельность, после чего пользователь опять возвращается к деятельности Game). Следующая модификация метода Game.onCreate() позаботится об этом:

Sudokuv4/src/org/example/sudoku/Game.java)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    // ...
```

```
    // Если деятельность перезапущена, продолжить игру в следующий раз
    getIntent().putExtra(KEY_DIFFICULTY, DIFFICULTY_CONTINUE);
}
```

На это закончим наш разговор о предустановках. Теперь давайте посмотрим на сохранение состояния запущенного экземпляра игры.

6.3. Запоминание текущей позиции курсора

Если вы измените ориентацию экрана в процессе выполнения Sudoku, то заметите, что она «забывает» позицию курсора. Это происходит потому, что мы используем собственное окно просмотра PuzzleView. Обычно окна в Android запоминают собственное состояние автоматически, но как только мы самостоятельно создаем программное окно, нам приходится позаботиться об этом самим.

В отличие от предустановленного состояния состояние запущенной программы постоянно изменяется. Оно находится в классе Bundle стека приложений Android. Состояние запущенной программы предназначено для сохранения небольших порций информации, таких как положение курсора.

Вот что мы должны сделать, чтобы включить данную функцию в нашу программу:

Sudokuv4/src/org/example/sudoku/PuzzleView.java

```
1     import android.os.Bundle;
-         import android.os.Parcelable;
-
-         public class PuzzleView extends View {
5             private static final String SELX = "selX" ;
-             private static final String SELY = "selY" ;
-             private static final String VIEW_STATE = "viewState" ;
-             private static final int ID = 42;
-
10             public PuzzleView(Context context) {
-                 // ...
-                 setId(ID);
-             }
-
15             @Override
-             protected Parcelable onSaveInstanceState() {
-                 Parcelable p = super.onSaveInstanceState();
-                 Log.d(TAG, „onSaveInstanceState" );
-                 Bundle bundle = new Bundle();
20                bundle.putInt(SELX, selX);
-                 bundle.putInt(SELY, selY);
-                 bundle.putParcelable(VIEW_STATE, p);
-                 return bundle;
-             }
```

```
25              @Override
-               protected void onRestoreInstanceState(Parcelable state) {
-                   Log.d(TAG, „onRestoreInstanceState" );
-                   Bundle bundle = (Bundle) state;
-                   select(bundle.getInt(SELX), bundle.getInt(SELY));
30              super.onRestoreInstanceState(bundle.getParcelable(VIEW_STATE));
-               }
-               // ...
-           }
```

В строке 5 мы определяем константы для ключей сохранения и восстановления позиции курсора. Нам нужно сохранять и координату *X*, и координату *Y* и вдобавок любое состояние, необходимое для базового класса View.

Как часть обработки Activity.onSaveInstanceState() Android спускается по иерархии окон просмотра и вызывает View.onSaveInstanceState() в каждом из окон, ID которого он найдет. То же самое происходит для метода onRestoreInstanceState(). Обычно этот идентификатор (ID) берется из XML, но после того, как PuzzleView был создан посредством программного кода, мы должны установить ID самостоятельно. Мы задаем произвольный номер в строке 8 (подойдет любое положительное значение) и затем используем метод setid() в строке 12, чтобы присвоить его.

Метод onSaveInstanceState() определен в строке 16. Мы вызываем суперкласс, чтобы получить его состояние, и затем сохраняем его и наше состояния в Bundle. Невозможность вызывать суперкласс приводит к ошибке времени выполнения.

Позже вызывается onRestoreInstanceState() (строка 26) для извлечения сохраненной нами информации. Мы получаем собственные координаты *X* и *Y* из Bundle и затем вызываем суперкласс, чтобы позволить получить ему все, что необходимо. После того как произведены эти изменения, позиция курсора запоминается PuzzleView так же, как и для любого другого окна просмотра Android.

Теперь давайте посмотрим, как хранить данные в старых добрых файлах.

6.4. Доступ к внешней файловой системе

В недрах Android работает Linux, поэтому здесь имеется и реальная смонтированная файловая система с корневым каталогом и всем остальным. Файлы хранятся в энергонезависимой флэш-памяти, встроенной в устройство, поэтому они не теряются, когда телефон выключают.

Вы можете пользоваться в программе всеми обычными подпрограммами ввода-вывода Java из пакета java.io, с учетом того, что ваши процессы имеют ограниченные разрешения, которые не позволяют им внести беспорядок в данные других приложений. На самом деле основное место, куда они могут получить доступ, — это приватная папка пакета, созданная во время установки (/data/data/packagename).

В классе Context есть несколько вспомогательных методов (и ими же класс Activity расширяет каждую из деятельностей), которые позволяют читать и запи-

сывать данные в эту папку. Вот некоторые из них, которые наверняка вам понадобятся:

deleteFile()	Удаляет приватный файл. Возвращает true при успешном завершении действия, в противном случае — false
fileList()	Возвращает список файлов в приватной зоне приложения в массиве String()
openFileInput()	Открывает приватный файл для чтения, возвращает java.io.FileInputStream
openFileOutput()	Открывает приватный файл для записи. Возвращает java.io.FileOutputStream

Однако так как эта внешняя память ограниченна, я рекомендую хранить здесь лишь небольшие объемы данных, например мегабайт или, в крайнем случае, два, и тщательно обрабатывать ошибки ввода-вывода на тот случай, если свободное пространство будет исчерпано.

К счастью, внешняя память — это не единственное хранилище данных, с которым вы можете работать.

ВСЕ В СЕМЬЕ

Если вы вспомните раздел 2.5 «Безопасность и защищенность», каждое приложение обычно получает свой собственный ID пользователя во время установки. Этот идентификатор пользователя позволяет читать и записывать данные в приватной папке приложения. Однако если два приложения подписаны[1] одним и тем же цифровым сертификатом, Android подразумевает то, что они поступили от одного и того же разработчика, и дает им один и тот же ID пользователя.

С одной стороны, это позволяет им разделять любые виды данных друг с другом, если это нужно, но с другой — это также подразумевает, что они нуждаются в особом внимании, чтобы не помешать друг другу.

6.5. Доступ к SD-карте

Некоторые устройства на Android включают слот для установки дополнительной флэш-памяти, обычно в виде Secure Digital (SD) карт. Эти карты памяти, если они присутствуют, гораздо больше по объему, чем встроенная память, и поэтому они отлично подходят для хранения многомегабайтных музыкальных и видеофайлов. Их нельзя использовать для хранения программного кода, но все приложения могут читать и записывать файлы на них.

В разделе 5.2 «Проигрывание видео» мы загружали пример видео в папку /data эмулируемого устройства. Это неподходящее место для таких файлов с тех пор,

[1] http://d.android.com/guide/topics/security/security.html#signing

как мы пришли к выводу, что не следует размещать большие файлы во внутренней файловой системе. Сейчас я продемонстрирую лучшее решение.

Первый шаг заключается в том, чтобы создать и отформатировать виртуальную SD-карту, которую мы можем «подключить» к эмулятору. К счастью, мы уже сделали это — если вы помните, в разделе 1.3, «Создание AVD», когда мы создавали виртуальное устройство *em22*, мы добавили ему виртуальную SD-карту объемом 64 Мб. Вы можете сделать ее любого размера, но если вы сделаете ее слишком маленькой, это может привести к зависанию эмулятора; если же вы сделаете ее слишком большой, это приведет к напрасной трате свободного места на жестком диске компьютера.

Далее давайте скопируем видеофайл на SD-карту:

```
C:\> adb push c:\code\samplevideo.3gp /sdcard/samplevideo.3gp
1468 KB/s (369870 bytes in 0.246s)
```

Теперь нам нужно модифицировать метод onCreate() класса Video для проигрывания видео с SD-карты вместо папки /data:

Videov2/src/org/example/video/Video.java

```
// Загрузка и начало проигрывания видеофайла
video.setVideoPath("/sdcard/samplevideo.3gp" );
video.start();
```

Теперь попробуйте запустить программу. Видео должно нормально проигрываться.

Обратите внимание на то, что начиная с Android 1.6 вы должны запросить разрешение WRITE_EXTERNAL_STORAGE в файле manifest, если хотите записывать данные на SD-карту из вашего приложения. Чтение с карты не нуждается в особых разрешениях.

Начиная с Android 2.2 ваше приложение может использовать метод Context.getEnternalFilesDir() для получения доступа к папке внешней файловой системы, где оно может разместить файлы для постоянного хранения. Android удалит эти файлы, когда приложение будет деинсталлировано.

6.6. Вперед>>

В этой главе мы рассмотрели несколько основных способов хранения локальных данных на платформе Android. Этого должно быть достаточно для начала, но для структурированных данных, таких как список телефонов или рецептов, вам понадобится кое-что более совершенное. Обратитесь к главе 9 «Работа с SQL» за указаниями, касающимися использования встроенной в Android базы данных SQLite и за информацией о том, как разделять информацию для совместного использования приложениями с применением контент-провайдеров. Инструкции по установке приложений на внешние носители вы найдете в разделе 13.6 «Установка на SD-карту».

Мы подошли к концу части II этой книги. На примере Sudoku вы ознакомились со всеми основами программирования под Android, включая разработку пользовательского интерфейса, использование 2D-графики, аудио, видео и хранения простых данных.

Сейчас пришло время оставить Sudoku позади и перейти от основ к более сложным особенностям программирования.

III

За пределами основ

7 Объединенный мир

В следующих главах мы рассмотрим более сложные темы, такие как доступ к Сети и сервисы, основанные на определении местоположения. Вы сможете написать множество полезных приложений и без этого, но шаг за пределы основных возможностей Android сделает ваши программы полезнее, добавит им функциональности при минимуме усилий с вашей стороны.

Для чего вы используете мобильный телефон? Если не брать в расчет телефонные звонки, все больше и больше людей используют свои телефоны как мобильные интернет-устройства. Аналитики предсказывают, что в ближайшие несколько лет мобильные телефоны вытеснят настольные компьютеры и станут устройством номер один для доступа в Интернет[1]. Причем в некоторых странах это уже произошло[2].

Телефоны на Android прекрасно приспособлены для использования в новом объединенном мире мобильного Интернета. Во-первых, Android имеет полнофункциональный веб-браузер, основанный на проекте с открытым кодом WebKit[3]. Он имеет тот же движок, что и Google Chrome, Apple iPhone и Safari, но с некоторыми «наворотами». Android позволяет использовать браузер как компонент прямо внутри вашего приложения.

Во-вторых, Android предоставляет вашим программам доступ к стандартным сетевым сервисам, таким как сокеты TCP/IP. Это позволяет использовать веб-сервисы от Google, Yahoo, Amazon и многих других источников в Интернете.

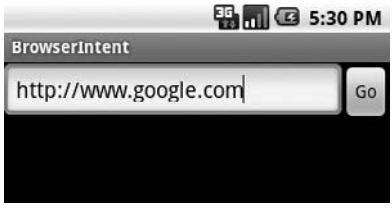

Рис. 7.1. Открытие браузера с использованием намерения Android

[1] http://archive.mobilecomputingnews.com/2010/0205.html
[2] http://www.comscore.com/press/release.asp?press=1742
[3] http://webkit.org

В данной главе вы узнаете, как воспользоваться преимуществами этих и других возможностей, на примере четырех демонстрационных программ:

- *BrowserIntent*: демонстрирует открытие внешнего веб-браузера с использованием намерения Android.
- *BrowserWiev*: показывает, как встроить браузер непосредственно в ваше приложение.
- *LocalBrowser*: разъясняет, как JavaScript во встроенном WebView и Java-код вашей Android-программы могут взаимодействовать друг с другом.
- *Translate*: использует связывание данных, поточность и веб-сервисы для достижения весьма занимательной цели.

7.1. Просмотр ресурсов Интернета с помощью намерения

Самое простое, что вы можете сделать с помощью сетевого API Android, — это открыть в браузере выбранную веб-страничку. Вы можете сделать это, чтобы обеспечить открытие ссылки на вашу домашнюю страничку из программы или чтобы получить доступ к серверному приложению, такому как система заказов. В Android все это занимает три строки программного кода.

Для демонстрации давайте напишем новый пример, названный BrowserIntent, имеющий поле ввода, в которое мы можем ввести URL, и кнопку Go, которую вы нажмете, чтобы открыть браузер с этим URL (рис. 7.1). Начнем с создания нового проекта «Hello, Android» со следующими значениями в мастере New Project:

```
Project name: BrowserIntent
Build Target: Android 2.2
Application name: BrowserIntent
Package name: org.example.browserintent
Create Activity: BrowserIntent
Min SDK Version: 8
```

Как только у вас будет каркасная программа, измените файл макета (res/layout/main.xml) следующим образом:

BrowserIntent/res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <EditText
        android:id="@+id/url_field"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1.0"
```

```
        android:lines="1"
        android:inputType="textUri"
        android:imeOptions="actionGo" />
    <Button
        android:id="@+id/go_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/go_button" />
</LinearLayout>
```

Этот код определяет наши два элемента управления — EditText и Button.

В EditText мы задали android:layout_weight="1.0", чтобы текстовая область заполнила все горизонтальное пространство слева от кнопки, также мы установили android:lines=»1» для ограничения высоты элемента управления одной строкой. Обратите внимание на то, что это не влияет на количество текста, которое пользователь может здесь ввести, а влияет лишь на то, как этот текст будет отображаться.

Android 1.5 представил поддержку для экранной клавиатуры и других альтернативных способов ввода данных. Параметры для android:inputType=»textUri» и android:imeOptions=»actionGo» — это подсказка, в каком виде должна появляться экранная клавиатура. Она дает Android указание заменить стандартную клавиатуру другой, которая имеет удобные кнопки для ввода *.com* и */* в веб-адресах и кнопку Go, которая открывает веб-страницу[1].

Как обычно, текст, который может быть прочитан человеком, следует поместить в файл ресурса, res/values/strings.xml.

BrowserIntent/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">BrowserIntent</string>
    <string name="go_button">Go</string>
</resources>
```

Далее нам нужно написать код для метода onCreate() в классе BrowserIntent. Это то место, где мы собираемся построить пользовательский интерфейс и подключить все поведение программы. Если вам лень все это набирать вручную, полный исходный код примера доступен в Интернете, на веб-сайте книги[2].

BrowserIntent/src/org/example/browserintent/BrowserIntent.java

```
1     package org.example.browserintent;
-
-         import android.app.Activity;
-         import android.content.Intent;
5         import android.net.Uri;
```

[1] Посетите страницу http://d.android.com/reference/android/widget/TextView.html и http://android-developers.blogspot.com/2009/04/updating-applications-for-on-screen.html, чтобы найти больше информации о параметрах ввода данных.

[2] http://pragprog.com/titles/eband3

```
-         import android.os.Bundle;
-         import android.view.KeyEvent;
-         import android.view.View;
-         import android.view.View.OnClickListener;
10         import android.view.View.OnKeyListener;
-         import android.widget.Button;
-         import android.widget.EditText;
-
-         public class BrowserIntent extends Activity {
15             private EditText urlText;
-             private Button goButton;
-
-             @Override
-             public void onCreate(Bundle savedInstanceState) {
20                 super.onCreate(savedInstanceState);
-                 setContentView(R.layout.main);
-
-                 // Получает обработчики для всех элементов пользовательского
                  //интерфейса
-                 urlText = (EditText) findViewById(R.id.url_field);
25                 goButton = (Button) findViewById(R.id.go_button);
-
-                 // Установка обработчиков событий
-                 goButton.setOnClickListener(new OnClickListener() {
-                     public void onClick(View view) {
30                         openBrowser();
-                     }
-                 });
-                 urlText.setOnKeyListener(new OnKeyListener() {
-                     public boolean onKey(View view, int keyCode, KeyEvent
event) {
35                         if (keyCode == KeyEvent.KEYCODE_ENTER) {
-                            openBrowser();
-                            return true;
-                         }
-                         return false;
40                     }
-                 });
-             }
-         }
```

Внутри onCreate() мы вызываем setContentView() в строке 21 для загрузки вьювера из его определения в ресурсе макета, а затем мы обращаемся к findViewById() в строке 24, чтобы получить обработчики для двух элементов управления пользовательского интерфейса.

Строка 28 указывает Android на запуск некоторого кода, когда пользователь нажимает кнопку Go, либо прикасаясь к ней, либо переходя к ней с помощью кнопок джойстика и нажимая его центральную кнопку. Когда это происходит, мы вызываем метод openBrowser(), который сейчас и определим.

Если пользователь вводит адрес и нажимает кнопку Enter (если такая есть на его телефоне), мы хотим, чтобы браузер открылся так же, как при нажатии кнопки Go. Чтобы это сделать, мы определяем обработчик в строке 33, который будет вызываться каждый раз, когда пользователь вводит символы в поле ввода. Если введен символ Enter, мы вызываем метод openBrowser() для открытия браузера; иначе возвращаем false, чтобы позволить текстовому элементу управления обычным образом обработать ввод текста.

А вот и та часть программы, появления которой мы ожидаем: метод openBrowser(). Как мы и обещали, здесь всего три строки:

BrowserIntent/src/org/example/browserintent/BrowserIntent.java

```
/** Открытие браузера с URL, заданном в текстовом поле */
private void openBrowser() {
    Uri uri = Uri.parse(urlText.getText().toString());
    Intent intent = new Intent(Intent.ACTION_VIEW, uri);
    startActivity(intent);
}
```

Первая строка получает адрес веб-страницы в виде строки (например, http://www.android.com) и конвертирует его в универсальный идентификатор ресурса (Universal Resource Identifier, URI).

Обратите внимание на то, что не следует опускать часть URL *http://*, когда вы будете испытывать свое приложение. Если вы это сделаете, программа завершится с ошибкой, так как Android не будет знать, как ему обрабатывать адрес. В реальной программе следует предусмотреть добавление этого префикса, если пользователь его забудет.

Следующая строка создает новый класс Intent с действием ACTION_VIEW и передает ему только что созданный класс Uri в качестве объекта, который мы хотим просмотреть. И наконец, мы вызываем метод startActivity(), чтобы сделать запрос на выполнение этого действия.

Когда запускается деятельность Browser, мы создаем ее собственное окно просмотра (рис. 7.2), и наша программа приостанавливается. Если пользователь нажмет клавишу Back в этот момент, окно браузера закроется, и ваше приложение продолжит работу. Но что, если вы хотите видеть что-то одновременно и в пользовательском интерфейсе, и на веб-странице? Android позволяет сделать это, используя класс WebView.

Рис. 7.2. Просмотр веб-страницы с помощью браузера по умолчанию

7.2. Веб-браузер с вьювером

На настольном компьютере веб-браузер — это большое, сложное, требовательное к памяти приложение с богатым набором функций, таких как закладки, подключаемые модули, флэш-анимация, вкладки, полосы прокрутки, возможности распечатки и так далее.

Когда я работал над проектом Eclipse и кто-то предлагал заменить некоторые обычные текстовые поля на встроенный веб-браузер, я полагал, что этот кто-то не прав. Не больше ли смысла, спорил я, просто расширить текстовое поле возможностями отображения курсива, таблиц или чего угодно, отсутствующего в нем?

Но оказалось, что этот кто-то не был не прав:

- Веб-браузер может быть (сравнительно) эффективным, если вы опустите все, кроме основного движка рендеринга.
- Если вы расширите функционал обычного текстового поля, добавляя все больше и больше возможностей, которыми обладает браузер, вы в итоге придете либо к чрезмерно усложненному, «раздутому» полю для просмотра текста, либо к маломощному браузеру.

Android предоставляет оболочку для движка браузера WebKit, которая называется WebView и предоставляет все возможности реального браузера с небольшим объемом задействованных ресурсов в 1 Мбайт. И хотя 1 Мбайт — это немало для компактного устройства, во многих случаях использование WebView оправданно.

WebView работает в основном так же, как и другие окна просмотра Android, за исключением того, что он имеет несколько дополнительных методов, специфичных для браузера. Я собираюсь создать интегрированную версию предыдущего примера. Она будет называться BrowserView вместо BrowserIntent, так как использует встроенный объект View вместо Intent. Начнем с создания нового проекта «Hello, Android», используя следующие установки:

```
Project name: BrowserView
Build Target: Android 2.2
Application name: BrowserView
Package name: org.example.browserview
Create Activity: BrowserView
Min SDK Version: 8
```

Файл макета для BrowserView похож на такой же для BrowserIntent, за исключением того, что мы добавим WebView в его нижней части:

BrowserView/res/layout/main.xml

```
<?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" >
```

```
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" >
        <EditText
            android:id="@+id/url_field"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1.0"
            android:lines="1"
            android:inputType="textUri"
            android:imeOptions="actionGo" />
        <Button
            android:id="@+id/go_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/go_button" />
    </LinearLayout>
    <WebView
        android:id="@+id/web_view"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1.0" />
</LinearLayout>
```

Мы используем два элемента управления LinearLayout, чтобы все отображалось на правильных позициях. Крайний элемент управления делит экран на верхнюю и нижнюю части: верхняя часть занята текстовым полем и кнопкой, нижняя часть содержит WebView. Внутренний LinearLayout — тот же самый, что и раньше; он лишь создает текстовое поле, выровненное по левому краю, и кнопку с выравниванием по правому краю.

Метод onCreate() для BrowserView не изменился, за исключением того, что он содержит дополнительный вьювер для поиска:

BrowserView/src/org/example/browserview/BrowserView.java

```
import android.webkit.WebView;
// ...
public class BrowserView extends Activity {
    private WebView webView;
    // ...
    @Override
    public void onCreate(Bundle savedInstanceState) {
        // ...
        webView = (WebView) findViewById(R.id.web_view);
        // ...
    }
}
```

Однако метод openBrowser() выглядит по-другому:

BrowserView/src/org/example/browserview/BrowserView.java

```
/** Открывает браузер с URL, заданном в текстовом поле */
private void openBrowser() {
    webView.getSettings().setJavaScriptEnabled(true);
    webView.loadUrl(urlText.getText().toString());
}
```

Метод loadUrl() запускает загрузку и отображение движком браузера веб-страницы, находящейся по заданному адресу. Он возвращает результат немедленно даже в том случае, если реальная загрузка занимает некоторое время (если она вообще когда-нибудь закончится).

Не забудьте обновить строковые ресурсы:

BrowserView/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">BrowserView</string>
    <string name="go_button">Go</string>
</resources>
```

Мы должны сделать еще одно изменение в программе. Добавьте следующую строку в AndroidManifest.xml перед тегом <application>:

BrowserView/AndroidManifest.xml

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

Если вы этого не сделаете, Android не даст вашему приложению доступа к Интернету и вы увидите ошибку «Web page not available».

Попробуйте запустить программу, введите подходящий веб-адрес, начинающийся с *http://*; когда вы нажмете Return или щелкните по кнопке Go, веб-страница должна появиться на экране (рис. 7.3).

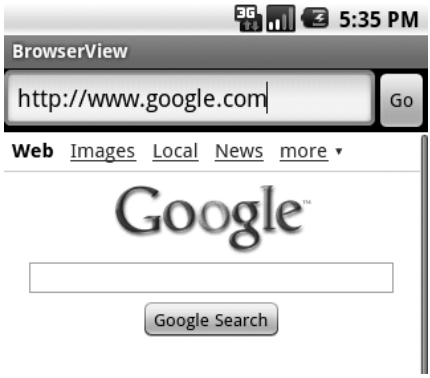

Рис. 7.3. Встраивание браузера с использованием WebView

ВОПРОС/ОТВЕТ

Почему BrowserIntent не нуждается в <user-permission>

Предыдущий пример, BrowserIntent, просто инициирует намерение, запрашивающее другое приложение для просмотра веб-страницы. Это другое приложение (браузер) и должно запрашивать разрешение на соединение с Интернетом в своем AndroidManifest.xml.

WebView имеет десятки других методов, которые вы можете использовать для управления отображением контента или для получения сообщений об изменениях состояния браузера.

Вы можете найти их полный список в онлайновой документации к WebView. Далее приведены методы, которые, вам, скорее всего, понадобятся:

- addJavascriptInterface(): предоставляет доступ к Java-объектам из кода JavaScript (подробнее об этом читайте в следующем разделе);
- createSnapshot(): создает скриншот текущей страницы;
- getSettings(): возвращает объект WebSettings, который используется для управления настройками;
- loadData(): загружает заданную текстовую строку в браузер;
- loadDataWithBaseURL(): загружает заданные данные, используя базовый URL;
- loadUrl(): загружает веб-страницу с заданного URL;
- setDownloadListener(): регистрирует обратные вызовы для событий загрузки, таких как, например, загрузка пользователем файлов .zip или .apk файла;

ВОПРОС/ОТВЕТ

Опасно ли позволять JavaScript вызывать Java?

Если вы разрешили веб-странице доступ к локальным ресурсам или вызов функций, находящихся за пределами «песочницы» браузера, тщательно продумайте возможные последствия для безопасности. Например, вы не хотите создавать метод, позволяющий JavaScript читать данные из любого места файловой системы, так как это может раскрыть ваши приватные данные вредоносному сайту, который получил бы информацию об именах ваших файлов.

Вот несколько идей, которые следует постоянно держать в голове. Первая: не полагайтесь на безопасность вслепую. Установите ограничение на страницы, которые могут запускать ваши методы, и на то, что эти методы могут делать. И помните золотое правило безопасности: не управляйте выходом, управляйте входом. Говоря другими словами, не пытайтесь проверить все непонятное, что кто-то предлагает вам сделать (например, неправильные символы в запросе). Вы можете что-то упустить. Вместо этого запрещайте все и пропускайте только то, в безопасности чего вы абсолютно уверены.

- setWebChromeClient(): регистрирует обратные вызовы для событий, которые требуется выполнить за пределами области просмотра WebView, таких как обновление заголовка или индикатора прогресса или открытие диалогового окна JavaScript;

- setWebViewClient(): позволяет приложению устанавливать перехватчики в браузере для перехвата событий, таких как загрузка ресурсов, нажатие кнопок и запросы на авторизацию;
- stopLoading(): останавливает загрузку текущей страницы.

Одна из наиболее полезных вещей, которую вы можете сделать с помощью элемента управления WebView, заключается в обмене сообщениями между ним и содержащим его приложением Android. Посмотрим на эту возможность поближе.

7.3. Из JavaScript в Java и обратно

Устройство Android может делать массу замечательных вещей, таких как хранение локальных данных, рисование графических объектов, проигрывание музыки, совершение звонков и определение собственного местоположения. Было бы прекрасно, если бы существовала возможность получать доступ к этой функциональности с веб-страниц, не так ли? С помощью встроенного WebView вы сможете это сделать.

Ключ к этим возможностям — метод addJavaScriptInterface() в классе WebView. Вы можете использовать его для того, чтобы расширить DOM (Document Object Model — объектная модель документа) внутри встроенного браузера и для определения новых объектов, к которым может получить доступ JavaScript-код. Когда программа на JavaScript вызывает методы таких объектов, это на самом деле приводит к вызову методов в Android-программе.

Также вы можете вызвать методы JavaScript из Android-программы. Все, что вам нужно для этого сделать, — использовать метод loadUrl(), передавая ему URL формы *javascript:code-to-execute*. Вместо того чтобы перейти к новой странице, браузер исполнит заданное выражение на JavaScript внутри текущей страницы. Вы можете вызывать методы, менять значение переменных JavaScript, изменять просматриваемый документ — все, что вам нужно.

Для демонстрации взаимодействия между JavaScript в WebView и Java в программе Android давайте создадим программу, часть которой написана на HTML/JavaScript, а часть — на Android (рис. 7.4). Верхняя часть окна приложения — это элемент управления WebView, нижняя часть — это TextView и Button из пользовательского интерфейса Android. Когда вы щелкаете на кнопке или на ссылках, это приводит к обмену командами между двумя средами.

Начнем с создания программы «Hello, Android», используя следующие параметры:

```
Project name: LocalBrowser
Build Target: Android 2.2
Application name: LocalBrowser
Package name: org.example.localbrowser
Create Activity: LocalBrowser
Min SDK Version: 8
```

Пользовательский интерфейс нашей программы будет разделен на две части. Первая часть определена в файле макета Android res/layout/main.xml.

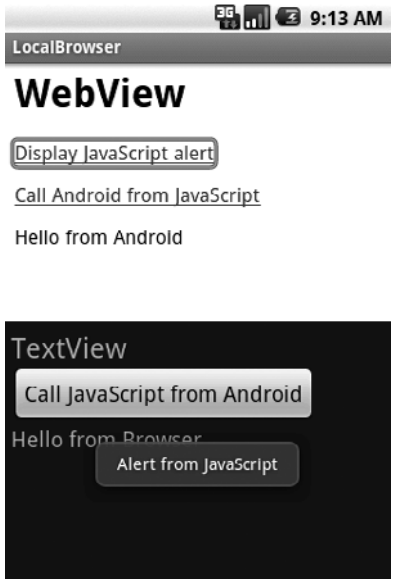

Рис. 7.4. Взаимодействие между Android и встроенным WebView

LocalBrowser/res/layout/main.xml

```
<?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" >
    <WebView
        android:id="@+id/web_view"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1.0" />
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1.0"
        android:padding="5sp" >
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textSize="24sp"
            android:text="@string/textview" />
```

```
        <Button
            android:id="@+id/button"
            android:text="@string/call_javascript_from_android"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="18sp" />
        <TextView
            android:id="@+id/text_view"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textSize="18sp" />
    </LinearLayout>
</LinearLayout>
```

Вторая часть представляет собой файл index.html, который будет загружен в WebView. Этот файл расположен в папке assets, а не в папке res, так как это некомпилируемый ресурс. Все, что находится в папке assets, копируется без изменений в локальное хранилище данных при установке программы. Эта папка предназначена для хранения локальных копий HTML-страниц, изображений и скриптов, которые браузер может просматривать без подключения к Сети.

LocalBrowser/assets/index.html

```
1     <html>
-         <head>
-         <script language="JavaScript">
-             function callJS(arg) {
5                 document.getElementById('replaceme').innerHTML = arg;
-             }
-         </script>
-         </head>
-         <body>
10         <h1>WebView</h1>
-         <p>
-         <a href="#" onclick="window.alert('Alert from JavaScript')">
-             Display JavaScript alert</a>
-         </p>
15         <p>
-         <a href="#" onclick="window.android.callAndroid('Hello from Browser')">
-             Call Android from JavaScript</a>
-         </p>
-         <p id="replaceme">
20         </p>
-         </body>
-         </html>
```

Строка 4 index.html определяет функцию callJS(), которую наша программа вызовет позже. Она принимает в качестве входного параметра строковой аргумент и вставляет его в тег replaceme в строке 19.

На рис. 7.4 вы видите две HTML-ссылки, которые определены начиная со строки 12. Первая просто вызывает стандартную функцию window.alert() для открытия

окна, показывающего короткое сообщение. Вторая ссылка, в строке 16, вызывает метод callAndroid() в объекте window.android(). Если вы загрузите эту страницу в обычный веб-браузер, window.android() будет не определена. Но с того момента, как мы внедрили браузер в приложение Android, мы можем самостоятельно объявлять объекты так, чтобы страница могла их использовать.

Далее мы вернемся к коду для Android в классе LocalBrowser. Вот основной каркас, включающий команды импорта, которые понадобятся нам позже:

LocalBrowser/src/org/example/localbrowser/LocalBrowser.java

```
Стр. 1     package org.example.localbrowser;
-
-          import android.app.Activity;
-          import android.os.Bundle;
5          import android.os.Handler;
-          import android.util.Log;
-          import android.view.View;
-          import android.view.View.OnClickListener;
-          import android.webkit.JsResult;
10          import android.webkit.WebChromeClient;
-          import android.webkit.WebView;
-          import android.widget.Button;
-          import android.widget.TextView;
-          import android.widget.Toast;
15
-          public class LocalBrowser extends Activity {
-              private static final String TAG = "LocalBrowser" ;
-              private final Handler handler = new Handler();
-              private WebView webView;
20              private TextView textView;
-              private Button button;
-
-              @Override
-              public void onCreate(Bundle savedInstanceState) {
25                  super.onCreate(savedInstanceState);
-                  setContentView(R.layout.main);
-
-                  // Поиск элементов управления Android на экране
-                  webView = (WebView) findViewById(R.id.web_view);
30                  textView = (TextView) findViewById(R.id.text_view);
-                  button = (Button) findViewById(R.id.button);
-                  // Продолжение onCreate следует...
-              }
-          }
```

Обратите внимание на инициализацию объекта Handler в строке 18. Вызов JavaScript поступает в специальный поток, выделенный для браузера, но вызов пользовательского интерфейса Android может быть выполнен только из главного потока (GUI). Мы используем класс Handler для выполнения перехода.

Для вызова кода Java Android из JavaScript определите обычный Java-объект с помощью одного или нескольких методов, как показано далее:

LocalBrowser/src/org/example/localbrowser/LocalBrowser.java

```
/** Объект, доступный JavaScript */
private class AndroidBridge {
    public void callAndroid(final String arg) { // must be final
        handler.post(new Runnable() {
            public void run() {
                Log.d(TAG, "callAndroid(" + arg + ")" );
                textView.setText(arg);
            }
        });
    }
}
```

Когда JavaScript вызывает метод callAndroid(), приложение создает новый объект Runnable и отправляет его в очередь исполнения главного потока, используя Handler.post(). Как только у главного потока появляется возможность, он активирует метод run(), который вызывает setText() для изменения текста в объекте TextView. Сейчас — самое время связать все воедино в методе onCreate(). Для начала мы активируем JavaScript (по умолчанию он выключен) и зарегистрируем наш мост к JavaScript:

LocalBrowser/src/org/example/localbrowser/LocalBrowser.java

```
// Включение JavaScript во встроенном браузере
webView.getSettings().setJavaScriptEnabled(true);
// Предоставление Java-объекта JavaScript в браузере
webView.addJavascriptInterface(new AndroidBridge(),
    "android" );
```

Теперь мы создадим анонимный объект WebChromeClient и зарегистрируем его с помощью метода setWebChromeClient().

LocalBrowser/src/org/example/localbrowser/LocalBrowser.java

```
// Задание функции, которая будет вызвана, когда JavaScript попытается
// открыть окно сообщения
webView.setWebChromeClient(new WebChromeClient() {
    @Override
    public boolean onJsAlert(final WebView view,
        final String url, final String message,
        JsResult result) {
        Log.d(TAG, "onJsAlert(" + view + ", " + url + ", "
            + message + ", " + result + ")" );
        Toast.makeText(LocalBrowser.this, message, 3000).show();
        result.confirm();
        return true; // Я обработал это
    }
});
```

Термин *chrome* здесь относится ко всему, что происходит вокруг окна браузера. Если это полнофункциональное клиентское приложение, мы должны обрабатывать команды навигации, закладки, меню и так далее. В данном случае все, что нам нужно, — это изменить поведение кода JavaScript, когда браузер попытается открыть сообщение JavaScript (используя window.alert()). Внутри onJsAlert() мы используем класс Android Toast для создания окна сообщения, которое появится на небольшое время (в данном случае — на 3000 миллисекунд, или на 3 секунды).

Как только мы завершим настройку WebView, мы можем использовать loadUrl() для загрузки локальной веб-страницы:

LocalBrowser/src/org/example/localbrowser/LocalBrowser.java

```
// Загрузка веб-страницы из локального ресурса
webView.loadUrl("file:///android_asset/index.html" );
```

Ссылки вида *file:///android_asset/filename* (обратите внимание на три слэша) имеют особое значение для движка браузера Android. Как вы уже могли догадаться, они ссылаются на файлы в папке assets. В нашем случае мы загружаем файл index.html, определенный ранее.

Вот файл res/values/strings.xml для примера LocalBrowser.

LocalBrowser/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">LocalBrowser</string>
    <string name="textview">TextView</string>
    <string name="call_javascript_from_android">
        Call JavaScript from Android
    </string>
</resources>
```

Последнее, что нужно сделать, — настроить кнопку в нижней части экрана таким образом, чтобы она могла вызывать JavaScript (вызов JavaScript из Java).

LocalBrowser/src/org/example/localbrowser/LocalBrowser.java

```
// Эта функция будет вызвана, когда пользователь нажмет
// кнопку в Android-части приложения
button.setOnClickListener(new OnClickListener() {
    public void onClick(View view) {
        Log.d(TAG, "onClick(" + view + ")" );
        webView.loadUrl("javascript:callJS('Hello from Android')" );
    }
});
```

Для того чтобы это сделать, мы устанавливаем процесс, отслеживающий нажатия на кнопку, используя setOnClickListener(). Когда кнопка нажата, вызывается onClick(), который выполняется и вызывает WebView.loadUrl(), передавая ему выражение JavaScript для вычисления в браузере. Выражение — это вызов функции callJS(), определенной в index.html.

Сейчас запустите программу и протестируйте ее работу. Когда вы щелкнете на «Display JavaScript alert», появится окно сообщения Android. Когда вы щелкнете на

«Call Android from JavaScript», в текстовом элементе управления Android появится строка «Hello from browser». И наконец, когда вы нажмете кнопку «Call JavaScript from Android», строка «Hello from Android» будет отправлена в браузер и вставлена в HTML, где она будет отображена в конце веб-страницы.

Иногда не требуется отображать веб-страницу, но вы хотите получить доступ к каким-либо веб-сервисам или другим серверным ресурсам. В следующем разделе я покажу, как это сделать.

7.4. Использование веб-сервисов

Android обеспечивает полный набор стандартных сетевых API Java, таких как пакет java.net.Http.URLConnection, который вы можете использовать в программах. В этом случае сложность заключается в создании асинхронных вызовов так, чтобы пользовательский интерфейс вашей программы был доступен в любое время.

Подумаем о том, что может произойти, если вы просто произведете блокирующий сетевой вызов в главном потоке (GUI). До тех пор, пока вызов не возвратит результат (а этого может не произойти никогда), ваше приложение не сможет отвечать ни на какие события пользовательского интерфейса, такие как нажатие клавиш или кнопок. Для пользователя это выглядит как зависание, и вам следует этого избегать.

Пакет java.util.concurent отлично подходит для работы такого рода. Изначально созданный Дугом Ли (Doug Lea) как отдельная библиотека и позже встроенный в Java 5, этот пакет поддерживает программирование параллельных вычислений на более высоком уровне, чем стандартный класс Java Thread. Класс ExecutorService управляет одним или несколькими потоками, и все, что вам нужно сделать, — это передать задачи (экземпляры Runnable или Callable) исполнителю для запуска. В этом случае будет возвращен экземпляр класса Future, который является ссылкой на некоторые пока неизвестные будущие значения, которые будут возвращены вашими задачами (если будут). Вы можете ограничить количество потоков, которые создаются, и можете прервать выполнение задач, если в этом возникнет необходимость.

Для того чтобы проиллюстрировать эти идеи, давайте создадим маленькую забавную программку, которая вызывает Google Translation API. Вы когда-нибудь смеялись над странными переводами на иностранные языки и с них, особенно над переводами, произведенными компьютером? Эта программа позволит пользователю ввести фразу на одном языке, запросить у Google перевод на другой язык, и затем попросить у Google перевести результат обратно на исходный язык. В идеале, результат должен представлять собой те же самые исходные слова, но так бывает не всегда, это показано на рис. 7.5.

Для использования этой программы просто выберите исходный и целевой язык, а затем начните вводить фразу. Пока вы вводите текст, программа будет использовать веб-сервис Google Translate для перевода текста «на» и «с» целевого языка.

Рис. 7.5. Работа над совершенствованием машинного перевода все еще продолжается

Для создания этого приложения начнем с программы «Hello, Android», используя следующие параметры:

```
Project name: Translate
Build Target: Android 2.2
Application name: Translate
Package name: org.example.translate
Create Activity: Translate
Min SDK Version: 8
```

Как только этот пример получит доступ к Интернету, чтобы выполнить вызов веб-сервиса, нам понадобится запросить у Android соответствующее разрешение.

ТРУДНОСТИ ПЕРЕВОДА

Когда я впервые размышлял над этим примером, я считал, что не возникнет проблемы с получением забавных результатов. К несчастью (или к счастью, в зависимости от вашей точки зрения), интерактивный переводчик Google делает свою работу хорошо для большинства языков. Если вы найдете какие-нибудь особо забавные случаи, когда переводчик действительно все путает, пожалуйста, отправьте их на дискуссионный форум сайта книги (http://pragprog.com/titles/eband3), чтобы и другие повеселились.

Добавьте эту строку в файл AndroidManifest.xml перед тегом XML <application>:

Translate/AndroidManifest.xml

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

Макет для этого примера немного сложнее, чем обычно, здесь мы будем использовать окно просмотра TableLayout. TableLayout позволяет организовывать вьюверы

в строки и столбцы, управляя выравниванием и настройкой размера колонок для того, чтобы они соответствовали содержимому. Это похоже на использовании тегов *<table>* и *<tr>* в HTML.

Translate/res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <TableLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:stretchColumns="1"
        android:padding="10dip" >
        <TableRow>
            <TextView android:text="@string/from_text" />
            <Spinner android:id="@+id/from_language" />
        </TableRow>
        <EditText
            android:id="@+id/original_text"
            android:hint="@string/original_hint"
            android:padding="10dip"
            android:textSize="18sp" />
        <TableRow>
            <TextView android:text="@string/to_text" />
            <Spinner android:id="@+id/to_language" />
        </TableRow>
        <TextView
            android:id="@+id/translated_text"
            android:padding="10dip"
            android:textSize="18sp" />
        <TextView android:text="@string/back_text" />
        <TextView
            android:id="@+id/retranslated_text"
            android:padding="10dip"
            android:textSize="18sp" />
    </TableLayout>
</ScrollView>
```

В этом примере у нас есть шесть строк, каждая строка содержит одну или две колонки. Заметьте, что если в строке присутствует лишь один вьювер, вам не нужно использовать TableRow в качестве контейнера для него. Также нет необходимости использовать android:layout_width= и android:layout_height= в каждом вьювере, как вы это делали с LinearLayout.

Класс Spinner — это кое-что новое, чего мы раньше не видели. Он похож на комбинированное поле ввода (combo box) в другом наборе элементов интерфейса. Пользователь выделяет поле выбора (spinner) (например, прикасаясь к нему), и появляется список доступных для выбора значений. В этом примере мы собираемся использовать данный элемент управления для выбора из списка языков.

Настоящий список хранится в качестве ресурса Android в файле res/values/arrays.xml::

Translate/res/values/arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <array name="languages">
        <item>Bulgarian (bg)</item>
        <item>Chinese Simplified (zh-CN)</item>
        <item>Chinese Traditional (zh-TW)</item>
        <item>Catalan (ca)</item>
        <item>Croatian (hr)</item>
        <item>Czech (cs)</item>
        <item>Danish (da)</item>
        <item>Dutch (nl)</item>
        <item>English (en)</item>
        <item>Filipino (tl)</item>
        <item>Finnish (fi)</item>
        <item>French (fr)</item>
        <item>German (de)</item>
        <item>Greek (el)</item>
        <item>Indonesian (id)</item>
        <item>Italian (it)</item>
        <item>Japanese (ja)</item>
        <item>Korean (ko)</item>
        <item>Latvian (lv)</item>
        <item>Lithuanian (lt)</item>
        <item>Norwegian (no)</item>
        <item>Polish (pl)</item>
        <item>Portuguese (pt-PT)</item>
        <item>Romanian (ro)</item>
        <item>Russian (ru)</item>
        <item>Spanish (es)</item>
        <item>Serbian (sr)</item>
        <item>Slovak (sk)</item>
        <item>Slovenian (sl)</item>
        <item>Swedish (sv)</item>
        <item>Ukrainian (uk)</item>
    </array>
</resources>
```

Здесь определен список, который называется languages, содержащий большинство языков, распознаваемых Google Translate API. Заметим, что каждое значение имеет длинное имя (например, *Spanish*) и короткое имя (например, *es*). Мы будем использовать короткие имена при передаче информации о языке переводчику.

Теперь приступим к модификации класса Translate. Вот его основной каркас:

Translate/src/org/example/translate/Translate.java

```
1     package org.example.translate;
-
-         import java.util.concurrent.ExecutorService;
```

```
-          import java.util.concurrent.Executors;
5          import java.util.concurrent.Future;
-          import java.util.concurrent.RejectedExecutionException;
-
-          import android.app.Activity;
-          import android.os.Bundle;
10          import android.os.Handler;
-          import android.text.Editable;
-          import android.text.TextWatcher;
-          import android.view.View;
-          import android.widget.AdapterView;
15          import android.widget.ArrayAdapter;
-          import android.widget.EditText;
-          import android.widget.Spinner;
-          import android.widget.TextView;
-          import android.widget.AdapterView.OnItemSelectedListener;
20
-          public class Translate extends Activity {
-              private Spinner fromSpinner;
-              private Spinner toSpinner;
-              private EditText origText;
25              private TextView transText;
-              private TextView retransText;
-
-              private TextWatcher textWatcher;
-              private OnItemSelectedListener itemListener;
30
-              private Handler guiThread;
-              private ExecutorService transThread;
-              private Runnable updateTask;
-              private Future transPending;
35
-              @Override
-              public void onCreate(Bundle savedInstanceState) {
-                  super.onCreate(savedInstanceState);
-
40                   setContentView(R.layout.main);
-                  initThreading();
-                  findViews();
-                  setAdapters();
-                  setListeners();
45              }
-          }
```

После объявления нескольких переменных мы определяем метод onCreate(), начинающийся в строке 37, для инициализации работы с потоками и инициализации пользовательского интерфейса. Не беспокойтесь, мы напишем код этих методов в порядке рассмотрения примера.

Метод findViews(), вызванный из строки 42, просто получает обработчики всех элементов пользовательского интерфейса, определенных в файле макета.

Translate/src/org/example/translate/Translate.java

```
private void findViews() {
    fromSpinner = (Spinner) findViewById(R.id.from_language);
    toSpinner = (Spinner) findViewById(R.id.to_language);
    origText = (EditText) findViewById(R.id.original_text);
    transText = (TextView) findViewById(R.id.translated_text);
    retransText = (TextView) findViewById(R.id.retranslated_text);
}
```

Метод setAdapters(), вызываемый из onCreate() в строке 43, определяет источники данных для полей выбора.

Translate/src/org/example/translate/Translate.java

```
private void setAdapters() {
    // Список для поля выбора берется из ресурса,
    // Пользовательский интерфейс поля выбора использует стандартный макет
    ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
        this, R.array.languages,
        android.R.layout.simple_spinner_item);
    adapter.setDropDownViewResource(
        android.R.layout.simple_spinner_dropdown_item);
    fromSpinner.setAdapter(adapter);
    toSpinner.setAdapter(adapter);
    // Автоматический выбор двух элементов полей выбора
    fromSpinner.setSelection(8); // English (en)
    toSpinner.setSelection(11); // French (fr)
}
```

ВОПРОС/ОТВЕТ

Действительно ли необходимы все эти задержки и работа с потоками?

Одна из причин, по которой вам следует поступать именно так, заключается в необходимости избежать слишком большого количества вызовов внешней веб-службы. Представьте себе, что происходит, когда пользователь вводит слово scissors. Программа видит вводимое слово по одному символу за раз, сначала s, потом — c, затем i и так далее, возможно, со стиранием некоторых символов, так как никто не помнит, как правильно пишется слово scissors. Вы действительно хотите производить вызов веб-службы для каждого отдельного символа? Нет, конечно. Не говоря о ненужной нагрузке на сервер, такое решение будет весьма требовательным в плане энергопотребления. Каждый запрос требует, чтобы радиомодуль устройства отправил и принял несколько пакетов данных, которые используют некоторую часть энергии, запасенной в батареях. Вы захотите подождать, пока пользователь завершит ввод, прежде чем отправить запрос, но как сообщить системе, что он завершил ввод?

Алгоритм, использованный здесь, заключается в том, что, как только пользователь ввел символ, будет инициирован отложенный запрос. Если он не вводит других символов, секундная задержка заканчивается, и запрос проходит. Иначе, первый запрос удаляется из очереди запросов, прежде чем он уйдет на сервер. Если запрос уже обрабатывается, мы пытаемся прервать его. То же самое происходит при смене языков, за исключением того, что мы используем меньшую задержку. Хорошая новость заключается в том, что я все это уже сделал за вас, используйте тот же самый шаблон в ваших собственных асинхронных программах.

В Android Adapter — это класс, который связывает источники данных (в данном случае — массив языков, определенный в arrays.xml) с элементами пользовательского интерфейса (в данном случае — с полем выбора). Мы используем стандартный макет, который предлагается Android для отдельных элементов в списке и для выпадающего окна, которое вы видите, выделяя поле выбора.

Далее мы устанавливаем обработчики пользовательского интерфейса в подпрограмме setListeners() (вызываемой из строки 44 onCreate()):

Translate/src/org/example/translate/Translate.java

```
private void setListeners() {
    // Объявляем обработчики событий
    textWatcher = new TextWatcher() {
        public void beforeTextChanged(CharSequence s, int start,
            int count, int after) {
            /* Не делаем ничего */
        }
        public void onTextChanged(CharSequence s, int start,
            int before, int count) {
            queueUpdate(1000 /* миллисекунд */);
        }
        public void afterTextChanged(Editable s) {
            /* Не делаем ничего */
        }
};
itemListener = new OnItemSelectedListener() {
    public void onItemSelected(AdapterView parent, View v,
        int position, long id) {
        queueUpdate(200 /* миллисекунд */);
    }
    public void onNothingSelected(AdapterView parent) {
        /* Не делаем ничего */
    }
};
// Устанавливаем обработчики событий на интерфейсные виджеты
origText.addTextChangedListener(textWatcher);
fromSpinner.setOnItemSelectedListener(itemListener);
toSpinner.setOnItemSelectedListener(itemListener);
}
```

Мы задаем два обработчика: один, который вызывается, когда текст для перевода изменяется, и один, вызываемый при смене языка. queueUpdate() помещает отложенный запрос на обновление в список дел главного потока, используя Handler. Мы произвольным образом использовали задержку в 1000 миллисекунд для события изменения текста и 200-миллисекундную задержку для изменения языка.

Запрос на обновление определен внутри метода initThreading():

Translate/src/org/example/translate/Translate.java

```
1     private void initThreading() {
-             guiThread = new Handler();
```

```
-             transThread = Executors.newSingleThreadExecutor();
5             // Эта задача выполняет перевод и обновление экрана
-             updateTask = new Runnable() {
-                 public void run() {
-                 // Получить текст для перевода
-                 String original = origText.getText().toString().trim();
10
-                 // Отменить предыдущий перевод при его наличии
-                 if (transPending != null)
-                     transPending.cancel(true);
-
15                  // Проверить, не пуста ли строка для перевода
-                 if (original.length() == 0) {
-                     transText.setText(R.string.empty);
-                     retransText.setText(R.string.empty);
-                 } else {
20                      // Показать пользователю, что мы что-то делаем
-                     transText.setText(R.string.translating);
-                     retransText.setText(R.string.translating);
-
-                     // Начать перевод сейчас, но не ожидать его
25                      try {
-                         TranslateTask translateTask = new TranslateTask(
-                             Translate.this, // ссылка на деятельность
-                             original, // исходный текст
-                             getLang(fromSpinner), // с языка
30                               getLang(toSpinner) // на язык
-                         );
-                         transPending = transThread.submit(translateTask);
-                     } catch (RejectedExecutionException e) {
-                         // Невозможно начать новую задачу
35                          transText.setText(R.string.translation_error);
-                         retransText.setText(R.string.translation_error);
-                     }
-                 }
-             }
40         };
-    }
```

У нас есть два потока: главный поток Android, используемый для пользовательского интерфейса, и поток перевода, который мы создадим для выполнения работы по переводу. Мы представили первый с помощью Android Handler, а второй с помощью Java ExecutorService.

Строка 6 определяет задачу обновления, которая будет запланирована с помощью метода queueUpdate(). Когда он запустится, он тут же получит текущий текст для перевода и затем приготовится отправить задание на перевод в поток перевода. Он отменит любой перевод, который уже выполняется (в строке 13), позаботится о том случае, когда текстовая строка для перевода пуста (строка 17), и заполнит два текстовых поля, где позже появится переведенный текст, строкой «Translating...» (строка 21). Этот текст будет заменен позже на переведенный текст.

Наконец, в строке 26 мы создаем экземпляр TranslateTask, передаем ему ссылку на деятельность Translate так, чтобы его можно было вызвать позже для изменения строки, содержащей исходный текст и короткие имена двух языков, указанных в полях выбора. Строка 32 передает новую задачу в поток перевода, возвращая ссылку на возвращаемое значение Future. В этом случае мы на самом деле не имеем возвращенного значения до тех пор, пока TranslateTask не изменит GUI напрямую, но мы используем ссылку на Future снова в строке 13 для того, чтобы отменить перевод, если в этом возникнет необходимость.

Для окончательной отделки класса Translate существует несколько удобных функций, использованных в других местах:

Translate/src/org/example/translate/Translate.java

```
/** Извлекаем код языка из текущего поля выбора */
private String getLang(Spinner spinner) {
    String result = spinner.getSelectedItem().toString();
    int lparen = result.indexOf(,(, );
    int rparen = result.indexOf(,)‘ );
    result = result.substring(lparen + 1, rparen);
    return result;
}
/** Запрашивает начало обновления после короткой задержки */
private void queueUpdate(long delayMillis) {
    // Отменяет предыдущее обновление, если оно еще не началось
    guiThread.removeCallbacks(updateTask);
    // Начинает обновление, если ничего не произойдет через несколько миллисекунд
    guiThread.postDelayed(updateTask, delayMillis);
}
/** Изменяет текст на экране (вызывается из другого потока) */
public void setTranslated(String text) {
guiSetText(transText, text);
}
/** Изменяет текст на экране (вызывается из другого потока) */
public void setRetranslated(String text) {
    guiSetText(retransText, text);
}
/** Все изменения в GUI должны быть выполнены в потоке GUI */
private void guiSetText(final TextView view, final String text) {
    guiThread.post(new Runnable() {
        public void run() {
            view.setText(text);
        }
    });
}
```

Метод getLand() определяет, какой элемент выделен в текущий момент в поле выбора, получает строку для этого элемента и выбирает из нее короткий код языка, нужный для Translate API.

queueUpdate() помещает запрос на обновление в очередь запросов главного потока, но сообщает ему о небольшом ожидании до его реального запуска. Если в очереди уже есть запрос, он удаляется.

Методы setTranslated() и setRtranslated() будут использованы TranslateTask для обновления пользовательского интерфейса, когда результаты перевода вернутся от веб-сервиса. Они оба вызывают приватную функцию, которая называется guiSetText(); эта функция использует метод Handler.post() для того, чтобы запросить у главного потока GUI обновление текста в элементах управления TextView. Этот дополнительный шаг необходим, так как вы не можете вызвать функцию пользовательского интерфейса из потока, не относящегося к пользовательскому интерфейсу, и guiSetText() будет вызвана потоком перевода.

Вот файл res/values/strings.xml для примера Translate:

Translate/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Translate</string>
    <string name="from_text">From:</string>
    <string name="to_text">To:</string>
    <string name="back_text">And back again:</string>
    <string name="original_hint">Enter text to translate</string>
    <string name="empty"></string>
    <string name="translating">Translating...</string>
    <string name="translation_error">(Translation error)</string>
    <string name="translation_interrupted">(Translation
        interrupted)</string>
</resources>
```

Наконец, вот определение класса TranslateTask:

Translate/src/org/example/translate/TranslateTask.java

```
package org.example.translate;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import org.json.JSONException;
import org.json.JSONObject;
import android.util.Log;
public class TranslateTask implements Runnable {
    private static final String TAG = "TranslateTask" ;
    private final Translate translate;
    private final String original, from, to;
    TranslateTask(Translate translate, String original, String from,
        String to) {
        this.translate = translate;
        this.original = original;
        this.from = from;
        this.to = to;
    }
    public void run() {
```

```
        // Переводит исходный текст на целевой язык
        String trans = doTranslate(original, from, to);
        translate.setTranslated(trans);
        // Теперь переводит то, что мы получили, обратно на исходный язык
        // В идеале перевод будет идентичным оригиналу, но обычно это не так.
        String retrans = doTranslate(trans, to, from); // меняем
        translate.setRetranslated(retrans);
}
/**
* Вызываем Google Translation API для перевода строки с одного
* языка на другой. Подробности об API вы можете найти на:
* http://code.google.com/apis/ajaxlanguage
*/
private String doTranslate(String original, String from,
        String to) {
    String result = translate.getResources().getString(
        R.string.translation_error);
        HttpURLConnection con = null;
        Log.d(TAG, "doTranslate(" + original + ", " + from + ", "
            + to + ")" );
        try {
            // Проверка, не была ли прервана задача
            if (Thread.interrupted())
                throw new InterruptedException();
            // Построение запроса RESTful для Google API
            String q = URLEncoder.encode(original, "UTF-8" );
            URL url = new URL(
            "http://ajax.googleapis.com/ajax/services/language/translate"
                + "?v=1.0" + "&q=" + q + "&langpair=" + from
                + "%7C" + to);
            con = (HttpURLConnection) url.openConnection();
            con.setReadTimeout(10000 /* миллисекунды */);
            con.setConnectTimeout(15000 /* миллисекунды */);
            con.setRequestMethod("GET" );
            con.addRequestProperty("Referer" ,
                "http://www.pragprog.com/titles/eband3/hello-android" );
            con.setDoInput(true);
            // Запуск запроса
            con.connect();
            // Проверка, не была ли прервана задача
            if (Thread.interrupted())
                throw new InterruptedException();
            // Чтение результатов запроса
            BufferedReader reader = new BufferedReader(
                new InputStreamReader(con.getInputStream(), "UTF-8" ));
            String payload = reader.readLine();
            reader.close();
            // Обработка для получения переведенного текста
            JSONObject jsonObject = new JSONObject(payload);
            result = jsonObject.getJSONObject("responseData" )
```

```
                .getString("translatedText" )
                .replace("&#39;" , "'" )
                .replace("&amp;" , "&" );
            // Проверка, не была ли прервана задача
            if (Thread.interrupted())
                throw new InterruptedException();
        } catch (IOException e) {
                Log.e(TAG, "IOException" , e);
        } catch (JSONException e) {
            Log.e(TAG, "JSONException" , e);
        } catch (InterruptedException e) {
            Log.d(TAG, "InterruptedException" , e);
            result = translate.getResources().getString(
                R.string.translation_interrupted);
        } finally {
            if (con != null) {
                con.disconnect();
            }
        }
        // Все выполнено
        Log.d(TAG, " -> returned " + result);
        return result;
    }
}
```

Это отличный пример вызова веб-сервиса RESTful с помощью HttpURLConnection, разбора результата в формате JavaScript Object Notation (JSON) и обработки всех видов сетевых ошибок и запросов на прерывание операции. Я не собираюсь объяснять все это детально, так как данный код не содержит ничего специфического для Android, за исключением нескольких сообщений об ошибках.

7.5. Вперед>>

В этой главе мы рассмотрели множество основополагающих приемов: от открытия простой веб-страницы до использования асинхронных веб-сервисов. Программирование на HTML/JavaScript выходит за рамки данной книги, но есть несколько хороших ресурсов по этой теме. Если вы собираетесь углубиться в параллельное программирование с использованием классов, таких как ExecutorService, я рекомендую книгу Java Concurrency in Practice [Goe06], которую написал Brian Goetz.

Следующая глава посвящена обсуждению нового уровня интерактивности с использованием сервисов определения географического положения устройства и его положения в пространстве. Если вы хотите узнать больше об источниках данных и связывании данных, переходите к главе 9 «Работа с SQL».

8 Определение местоположения и использование сенсоров

Платформа Android использует множество различных технологий. Большинство из них — новые, но некоторые можно было встретить и раньше. Уникальность Android заключается в том, что эти технологии работают совместно. В данной главе мы обсудим следующее:

- информирование о местоположении с использованием недорогих GPS-устройств;
- акселерометр, встроенный в мобильные устройства, подобный тому, который можно найти в игровом пульте Nintendo Wii;
- смешанные данные, часто представляющие собой комбинацию карты с другой информацией.

Несколько популярных программ для Android используют эти концепции для лучшего соответствия ожиданиям пользователей. Например, приложение[1] Locale может адаптировать установки вашего телефона на основании местоположения. Вы постоянно забываете включить виброзвонок, когда находитесь на работе или в кинотеатре? Locale может позаботиться об этом, используя Android Location API, которое здесь описано.

8.1. Места, места, места

Сегодня, когда тридцать один спутник вращается вокруг Земли, для них нет лучшего занятия, чем помочь вам найти дорогу в ближайшую булочную. Глобальная система позиционирования (GPS, Global Positioning System), изначально разработанная для военных целей, но затем преобразованная для гражданского использования, передает прецизионные сигналы точного времени на наземные приемники, как тот, который расположен в вашем Android-телефоне. При хорошем приеме

[1] http://www.androidlocale.com

и небольшом объеме вычислений GPS-чип может определить вашу позицию с точностью до 15 метров[1].

ВОПРОС/ОТВЕТ

Может ли GPS позволить кому-нибудь следить за моими перемещениями?

Нет. GPS-приемник — это только приемник. GPS-чип и любая программа, запускаемая на вашем Android-устройстве, знают ваше местоположение. Но если ни одна из этих программ не передает преднамеренно эту информацию, никто не сможет использовать данную технологию, чтобы вас отыскать.

В дополнение к GPS Android может производить вычисление вашего местонахождения, основываясь на информации от близлежащих базовых станций сотовых операторов, и если вы подсоединены к точке доступа Wi-Fi, он может использовать и ее тоже. Помните, что все эти источники данных о местоположении в некоторых местах ненадежны. Когда вы перемещаетесь, например, внутри здания, только отраженный GPS-сигнал сможет достичь вашего приемника (да и то не всегда).

Для демонстрации возможностей Android по определению местоположения давайте напишем тестовую программу, которая просто отображает ваше текущее положение и обновляет его состояние на экране, когда вы передвигаетесь. Скриншот этой программы представлен на рис. 8.1.

Рис. 8.1. Тестирование LocationManager

[1] Вам необязательно знать, как работает GPS, чтобы пользоваться этой технологией, но если вам любопытно, посмотрите сайт http://electronics.howstuffworks.com/gadgets/travel/gps.htm.

Где я?

Начнем с создания приложения «Hello, Android», используя следующие параметры в мастере New Project:

```
Project name: LocationTest
Build Target: Android 2.2
Application name: LocationTest
Package name: org.example.locationtest
Create Activity: LocationTest
Min SDK Version: 8
```

Доступ к информации о местоположении закрыт разрешениями Android. Для обеспечения доступа к ним добавьте следующие строки в файл AndroidManifest.xml перед тегом <application>:

LocationTest/AndroidManifest.xml

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

В этом примере поддерживается как использование точных источников информации о местоположении, таких как GPS, так и работа с менее точными ресурсами, такими как триангуляция показателей базовых станций сотовых операторов.

В пользовательском интерфейсе мы собираемся выводить все данные о местоположении в большой прокручивающийся TextView, который определен в res/layout/main.xml.

LocationTest/res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <TextView
        android:id="@+id/output"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
</ScrollView>
```

Когда подготовительные работы выполнены, мы можем начинать программирование. Вот каркас класса LocationTest и метода onCreate(). (Пока не обращайте внимания на ссылку на LocationListener в строке 15, мы вернемся к ней позже).

LocationTest/src/org/example/locationtest/LocationTest.java

```
1     package org.example.locationtest;
-
-         import java.util.List;
-
5         import android.app.Activity;
```

```
-         import android.location.Criteria;
-         import android.location.Location;
-         import android.location.LocationListener;
-         import android.location.LocationManager;
10         import android.location.LocationProvider;
-         import android.os.Bundle;
-         import android.widget.TextView;
-
-         public class LocationTest extends Activity implements
15           LocationListener {
-            private LocationManager mgr;
-            private TextView output;
-            private String best;
-
20           @Override
-            public void onCreate(Bundle savedInstanceState) {
-                super.onCreate(savedInstanceState);
-                setContentView(R.layout.main);
-
25               mgr = (LocationManager) getSystemService(LOCATION_SERVICE);
-                output = (TextView) findViewById(R.id.output);
-
-                log("Location providers:" );
-                dumpProviders();
30
-                Criteria criteria = new Criteria();
-                best = mgr.getBestProvider(criteria, true);
-                log("\nBest provider is: " + best);
-
35               log("\nLocations (starting with last known):" );
-                Location location = mgr.getLastKnownLocation(best);
-                dumpLocation(location);
-            }
-         }
```

Начальная точка пути к сервисам Android, определяющим местоположение, — вызов getSystemService() в строке 25. Он возвращает класс LocationManager, который мы сохраним в поле для использования позже.

В строке 29 мы вызываем наш метод dumpProviders() для вывода списка всех поставщиков навигационной информации в системе.

Далее нам нужно выбрать одного из провайдеров для дальнейшего использования. Мне доводилось видеть примеры, где просто выбирается первый из доступных, но я советую использовать метод getBestProvider(), как показано здесь. Android будет подбирать наилучшего поставщика данных в соответствии с параметром Criteria, который вы ему передадите (строка 31). Если у вас есть какие-либо ограничения по стоимости, мощности, точности и так далее, это именно то место, где вам следует о них заявить. В нашем примере ограничений нет.

В зависимости от поставщика информации вычисление вашего местоположения может занять у устройства некоторое время. Возможно, это будут несколько секунд,

минута или даже больше. Однако Android запоминает последнюю возвращенную позицию, поэтому мы можем послать запрос и немедленно вывести ее в строке 36. Эта информация может быть устаревшей — например, если устройство ранее было выключено и перемещено, — но это лучше, чем ничего.

Знание того, где мы, — это лишь половина удовольствия. Куда мы идем дальше?

Обновление информации о местоположении

Для того чтобы Android мог сообщить об изменении местоположения, вызовите метод requestLocationUpdates() объекта LocationManager(). Для экономии энергии батарей мы хотим производить обновления лишь тогда, когда программа активна. Поэтому нам следует перехватить методы жизненного цикла деятельности Android, переопределяя методы onResume() и onPause().

LocationTest/src/org/example/locationtest/LocationTest.java

```
@Override
protected void onResume() {
    super.onResume();
    // Запуск обновлений (документация рекомендует задержку >= 60000 мс)
    mgr.requestLocationUpdates(best, 15000, 1, this);
}
@Override
protected void onPause() {
    super.onPause();
    // Остановка обновлений для экономии энергии, когда приложение приостановлено
    mgr.removeUpdates(this);
}
```

Когда работа приложения возобновляется, мы вызываем метод requestLocationUpdates() для запуска процесса обновления. Этот метод принимает четыре параметра: имя источника данных, задержку (вы не сможете получать обновления слишком часто), минимальное расстояние (изменения позиции на меньшее расстояние игнорируются) и объект LocationListener.

Когда приложение приостанавливается, мы вызываем метод removeUpdates() для остановки процесса получения обновлений. Источник данных (чип) будет переведен в режим пониженного энергопотребления, если в течение некоторого времени он не используется.

Сейчас вы знаете, почему LocationTest использует LocationListener, — мы можем просто передать ссылку на деятельность, вместо того чтобы создавать новый объект-обработчик. Это сэкономит нам около 1 Кбайт памяти во время выполнения программы.

Вот определение четырех методов, которые нужны данному интерфейсу:

LocationTest/src/org/example/locationtest/LocationTest.java

```
public void onLocationChanged(Location location) {
    dumpLocation(location);
}
public void onProviderDisabled(String provider) {
```

```
    log("\nProvider disabled: " + provider);
}
public void onProviderEnabled(String provider) {
    log("\nProvider enabled: " + provider);
}
public void onStatusChanged(String provider, int status,
    Bundle extras) {
    log("\nProvider status changed: " + provider + ", status="
        + S[status] + ", extras=" + extras);
}
```

Самый важный из них — onLocationChanged().

Как следует из его имени, он вызывается каждый раз, когда провайдер сообщает об изменении позиции устройства. Методы onProviderDisabled(), onProviderEnabled(), и onStatusChanged() могут быть использованы для переключения на другие источники данных в том случае, если первый выбранный окажется недоступным.

Код оставшихся методов LocationTest — log(), dumpProviders() и dumpLocation() — не очень интересен, но мы приведем его здесь для полноты изложения:

LocationTest/src/org/example/locationtest/LocationTest.java

```
// Определение имен, которые может прочитать пользователь
private static final String[] A = { "invalid" , "n/a" , "fine" , "coarse" };
private static final String[] P = { "invalid" , "n/a" , "low" , "medium" ,
    "high" };
private static final String[] S = { "out of service" ,
    "temporarily unavailable" , "available" };
/** Запись строки в окно вывода */
private void log(String string) {
    output.append(string + "\n" );
}
/** Запись информации от всех провайдеров навигационных данных */
private void dumpProviders() {
    List<String> providers = mgr.getAllProviders();
    for (String provider : providers) {
        dumpProvider(provider);
    }
}
/** Запись информации от отдельного поставщика навигационных данных */
private void dumpProvider(String provider) {
    LocationProvider info = mgr.getProvider(provider);
    StringBuilder builder = new StringBuilder();
    builder.append("LocationProvider[" )
        .append("name=" )
        .append(info.getName())
        .append(",enabled=" )
        .append(mgr.isProviderEnabled(provider))
        .append(",getAccuracy=" )
        .append(A[info.getAccuracy() + 1])
        .append(",getPowerRequirement=" )
        .append(P[info.getPowerRequirement() + 1])
```

```
        .append(",hasMonetaryCost=" )
        .append(info.hasMonetaryCost())
        .append(",requiresCell=" )
        .append(info.requiresCell())
        .append(",requiresNetwork=" )
        .append(info.requiresNetwork())
        .append(",requiresSatellite=" )
        .append(info.requiresSatellite())
        .append(",supportsAltitude=" )
        .append(info.supportsAltitude())
        .append(",supportsBearing=" )
        .append(info.supportsBearing())
        .append(",supportsSpeed=" )
        .append(info.supportsSpeed())
        .append("]" );
    log(builder.toString());
}
/** Описание местоположения, которое может принимать значение null */
private void dumpLocation(Location location) {
    if (location == null)
        log("\nLocation[unknown]" );
    else
        log("\n" + location.toString());
}
```

Если вы не хотите вводить весь этот текст с клавиатуры, найдите примеры для загрузки на сайте издательства «Питер» (www.piter.com).

Заметки об эмуляции

Если вы запустите пример LocationTest на реальном устройстве, он покажет ваше текущее местоположение. На эмуляторе используется виртуальный GPS-модуль, который всегда возвращает ту же самую позицию до тех пор, пока вы не измените ее. Сделаем это сейчас.

В Eclipse вы можете изменить местоположение виртуального устройства, используя окно Emulator Control (Window ▸ Show View ▸ Other... ▸ Android ▸ Emulator Control). Прокрутите окно к его нижней части и найдите поля для ручного ввода долготы и широты. Когда вы щелкнете на кнопке Send, Eclipse отправит новую позицию на эмулируемое устройство, и вы увидите его отображенным в любой программе, которая следит за местоположением.

Также вы можете запустить программу Dalvik Debug Monitor Service (DDMS) за пределами Eclipse и отправить изменение виртуальной позиции оттуда. В дополнение к ручному изменению позиции вы можете использовать путь, записанный во внешний файл. Посмотрите документацию к DDMS для получения более подробной информации[1]. С помощью провайдеров навигационных данных Android вы можете определить, где находитесь, но лишь довольно грубо, в глобальном плане.

[1] http://d.android.com/guide/developing/tools/ddms.html

Если вы хотите получить больше локальной информации — например, сведения о наклоне устройства или о температуре, используйте другой API. Это — тема следующего раздела.

8.2. Выжимаем все из сенсоров

Предположим, вы пишете симулятор автогонок и вам нужно предоставить игроку способ управлять машиной на экране. Один из способов — это использование кнопок, как в автомобильных играх на Sony PlayStation или Nintendo DS. Нажмите правую кнопку для поворота руля вправо, нажмите левую кнопку для движения влево и нажмите еще какую-нибудь кнопку, чтобы прибавить газу. Это нормально работает, но это не слишком естественно.

Видели ли вы, как кто-нибудь играет в подобные игры? Неосознанно они наклоняются из стороны в сторону, проходя крутой поворот, дергают джойстик, сталкиваясь с другой машиной, наклоняются вперед, увеличивая скорость, и назад, давя на тормоз. Было бы замечательно, если бы эти движения имели какое-то воздействие на игровой процесс? Теперь это возможно.

Использование сенсоров

Android SDK поддерживает множество различных типов сенсоров:

- TYPE_ACCELEROMETER: измеряет ускорение в пространстве по осям X, Y, Z;
- TYPE_LIGHT: сообщает о том, насколько хорошо освещено окружающее пространство;
- TYPE_MAGNETIC_FIELD: возвращает значение магнитного поля в пространстве по осям X, Y, Z;
- TYPE_ORIENTATION: измеряет повороты, наклоны и вращения устройства;
- TYPE_PRESSURE: воспринимает текущее атмосферное давление;
- TYPE_PROXIMITY: дает информацию о расстоянии от сенсора до других объектов;
- TYPE_TEMPERATURE: измеряет температуру окружающей среды.

Конечно, не все устройства поддерживают весь этот функционал[1].

Пример SensorTest, доступный на сайте книги, демонстрирует использование Sensor API. Класс Android SensorManager похож на LocationManager, за исключением того, что обновление данных он производит быстрее, возможно, проводя сотни измерений в секунду. Для того чтобы получить доступ к сенсорам, вы, в первую очередь, вызываете метод getSystemService(), как здесь:

[1] К несчастью, из Android 1.5 удалена поддержка сенсора TRICORDER, который превращает ваше устройство в полнофункциональный трикордер из Star Trek. К черту, Джим, я программист, а не создатель «пасхальных яиц».

SensorTest/src/org/example/sensortest/SensorTest.java

```
private SensorManager mgr;
    // ...
    mgr = (SensorManager) getSystemService(SENSOR_SERVICE);
```

Затем вы вызываете registerListener() в методе onResume(), чтобы начать получать обновленные данные, и вызываете unregisterListener() в методе onPause(), чтобы прекратить их получать.

Интерпретация показаний сенсоров

Сервис сенсоров вызывает onSensorChanged() каждый раз при изменении значений.

Это должно выглядеть примерно так:

SensorTest/src/org/example/sensortest/SensorTest.java

```
public void onSensorChanged(SensorEvent event) {
    for (int i = 0; i < event.values.length; i++) {
        // ...
    }
}
```

Все сенсоры возвращают массив значений с плавающей запятой. Размер массива зависит от особенностей сенсора: например, TYPE_TEMPERATURE возвращает лишь одно значение — температуру в градусах Цельсия. Возможно, вам не понадобится использовать все полученные от сенсора данные. Например, если вам нужны лишь сведения о магнитном азимуте, вы можете использовать первое число, возвращенное сенсором TYPE_ORIENTATION.

Превращение данных сенсоров (особенно это касается акселерометра) в осмысленную информацию может выглядеть как черная магия. Вот несколько советов, которые следует помнить:

- Показания акселерометра выглядят весьма неровными. Вам нужно сгладить эти данные, используя какой-нибудь вид взвешенных средних, но вы не должны сглаживать их слишком сильно, иначе ваш интерфейс будет выглядеть запаздывающим и неотзывчивым.
- Данные от сенсоров приходят неравномерно. Вы получаете несколько значений в строке, потом следует короткая пауза, за которой вы получите гораздо большее количество измерений. Не ожидайте спокойного, ровного потока данных.
- Попробуйте идти впереди пользователя, предсказывая его будущие действия. Предположим, последние три измерения показали начало вращения устройства вправо, следующее — немного быстрее, чем предыдущее. Вы можете предположить, с некоторой долей вероятности, каким будет следующее движение, и отреагировать, основываясь на вашем предсказании.

Сложней всего использовать сенсоры в играх стиля экшн, которым нужна прямая связь между тем, как пользователь перемещает устройство, и тем, что происходит на экране. К сожалению, эмулятор не очень подходит для таких вещей.

Замечания об эмуляторе

В соответствии с информацией Google **невозможно в полной мере протестировать** сенсоры, используя эмулятор. Большинство компьютеров не имеют датчиков освещенности, GPS-чипов или встроенных компасов. Весьма вероятно, что при запуске программы SensorTest на эмуляторе она не отобразит никаких результатов. Однако проект, который называется OpenIntents[1], предоставляет альтернативный сенсорный API, который можно вызывать исключительно для тестовых целей.

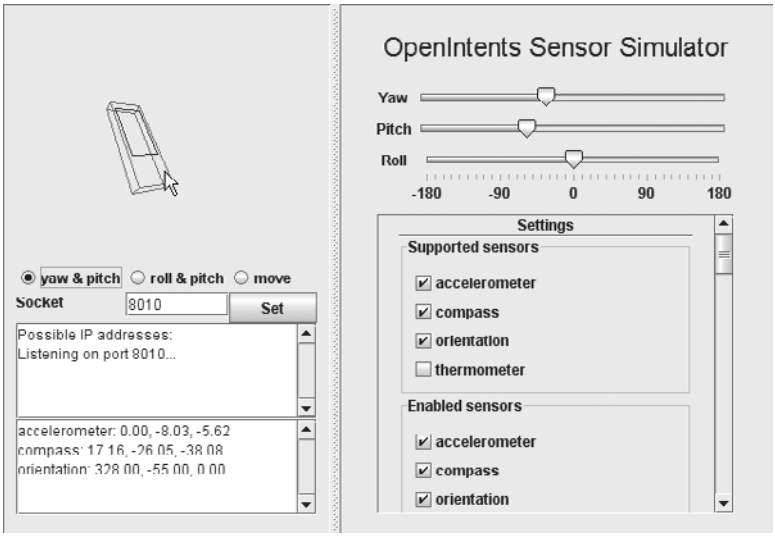

Рис. 8.2. Эмуляция сенсоров с помощью Sensor Simulator

Он работает следующим образом: вы подсоединяете эмулятор к приложению Sensor Simulator, запущенному на вашем компьютере. Симулятор показывает изображение виртуального телефона и позволяет перемещать его по экрану при помощи мыши (рис. 8.2), а затем передает эти движения вашей Android-программе, запущенной на эмуляторе. Если ваш рабочий компьютер действительно имеет сенсоры (как Apple MacBook) или вы можете подсоединить игровой джойстик Wii по Bluetooth, Sensor Simulator может использовать их в качестве источника данных.

Обратной стороной медали является то, что вам необходимо изменить исходный код, чтобы все заработало. Обратитесь к сайту OpenIntents для получения более подробной информации, если вы решите испытать эту программу. Я советую забыть об эмуляции сенсоров и тестировать все на реальном устройстве. Отлаживайте ваши алгоритмы до тех пор, пока они не заработают так, как надо.

[1] http://www.openintents.org

Сейчас вы знаете о низкоуровневых вызовах, позволяющих получить информацию о местоположении, и о запросах к сенсорам, возвращающим данные, такие как магнитный азимут. Для определенных приложений вы можете забыть это все и просто использовать Google Maps API.

8.3. Взгляд с высоты птичьего полета

Одним из первых приложений-захватчиков рынка для Ajax был Google Maps[1] (Карты Google). Используя JavaScript и объект XmlHttpRequest, инженеры Google создали просмотрщик карт, который позволял плавно перемещать, увеличивать и уменьшать карты и запускался в любом современном браузере без дополнительных подключаемых модулей. Эта идея была быстро скопирована другими фирмами, такими как Microsoft и Yahoo, но версия Google все еще самая лучшая.

Вы можете использовать эти веб-карты в Android с помощью встроенного элемента управления WebView, о котором мы говорили в разделе 7.2 «Веб-браузер с вьювером». Но архитектура вашего приложения будет серьезно усложнена, поэтому Google предлагает элемент управления MapView.

Встраивание MapView

MapView можно встроить в Android-приложение с помощью всего нескольких строк кода. Он обеспечивает основную функциональность Google Maps, а также возможности для добавления ваших собственных функций (рис. 8.3).

Рис. 8.3. Встроенная карта показывает ваше текущее местоположение

[1] http://maps.google.com

Класс MapView может также обеспечить связь с источниками данных о местоположении и сенсорами. Он может показывать текущее положение на карте и даже отображать компас, показывая направление вашего движения. Создадим демонстрационную программу, чтобы показать некоторые из его возможностей.

Для начала создадим приложение «Hello, Android», используя следующие значения в окне мастера:

```
Project name: MyMap
Build Target: Google APIs (Platform: 2.2)
Application name: MyMap
Package name: org.example.mymap
Create Activity: MyMap
Min SDK Version: 8
```

Обратите внимание на то, что мы используем Google APIs в качестве целевой среды, вместо Android 2.2. Это из-за того, что Google Maps APIs не является частью обычной поставки Android. Отредактируйте файл макета, замените его содержимое на MapView, который занимает весь экран.

MyMap/res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/frame"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <com.google.android.maps.MapView
        android:id="@+id/map"
        android:apiKey="MapAPIKey"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:clickable="true" />
</LinearLayout>
```

Замените MapApiKey на ключ Google Maps API, который вы получили от Google[1]. Обратите внимание на то, что мы используем полное имя (com.google.android.maps.MapView), так как MapView не является стандартным классом Android. Нам также нужно добавить тег <uses-library> в элемент <application> файла AndroidManifest.xml:

MyMap/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.example.mymap"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-permission
```

[1] http://code.google.com/android/maps-api-signup.html

```
        android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission
        android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission
        android:name="android.permission.INTERNET" />
    <application android:icon="@drawable/icon"
        android:label="@string/app_name" >
    <activity android:name=".MyMap"
        android:label="@string/app_name" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
    </activity>
    <uses-library android:name="com.google.android.maps" />
    </application>
    <uses-sdk android:minSdkVersion="3" android:targetSdkVersion="8" />
</manifest>
```

Если вы пропустите тег <uses-library>, то получите сообщение об ошибке ClassNotFoundException во время исполнения программы.

В дополнение к провайдерам навигационных данных различной точности класс MapView нуждается в доступе к Интернету для вызова сервера Google и получения изображения карт. Они будут автоматически кэшированы в папку вашего приложения.

Вот каркас класса MyMap:

MyMap/src/org/example/mymap/MyMap.java

```
package org.example.mymap;
import android.os.Bundle;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import com.google.android.maps.MyLocationOverlay;
public class MyMap extends MapActivity {
    private MapView map;
    private MapController controller;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        initMapView();
        initMyLocation();
    }
    @Override
    protected boolean isRouteDisplayed() {
        // Нужен MapActivity
        return false;
    }
}
```

Самое важное здесь то, что ваша деятельность расширена MapActivity. Класс MapActivity управляет фоновыми процессами, соединяется с Интернетом для получения изображений карты, занимается кэшированием, выполняет анимацию, заботится о жизненном цикле приложения и о многом другом. Все, что вам нужно, — правильно его настроить и запустить.

Подготовка

Первое, что нам нужно сделать — это вызвать findViewById() для получения доступа к MapView и его контейнеру. Мы можем сделать это в методе initMapView().

MyMap/src/org/example/mymap/MyMap.java

```
private void initMapView() {
    map = (MapView) findViewById(R.id.map);
    controller = map.getController();
    map.setSatellite(true);
    map.setBuiltInZoomControls(true);
}
```

Метод getController() возвращает объект MapController, который мы будем использовать для перемещения и масштабирования карты. setSatellite() переключает карту в спутниковый режим, и setBuiltInZoomControl()[1] включает стандартные элементы управления масштабом. Класс MapView позаботится о том, чтобы сделать эти элементы управления видимыми, когда пользователь перемещает карту, и постепенно сделает прозрачными, когда перемещение закончится.

Последний шаг — сообщить MapView о том, что он должен следовать за вашей позицией в методе initMyLocation().

ВОПРОС/ОТВЕТ

Почем MapView находится в пакете com.google.android.maps, а не в android.maps?

Любой код, находящийся в пакетах android.*, — это часть ядра Android. Он открыт и доступен на любом Android-устройстве. В противоположность ему карты являются собственностью Google и поставщиков данных, которые получают плату от Google за геологическую информацию и изображения. Google бесплатно предоставляет API до тех пор, пока вы соблюдаете определенные условия[2]. Если ограничения вас не устраивают, вы можете разработать свои собственные средства просмотра и найти собственные источники данных, но это непросто и недешево.

MyMap/src/org/example/mymap/MyMap.java

```
private void initMyLocation() {
    final MyLocationOverlay overlay = new MyLocationOverlay(this, map);
    overlay.enableMyLocation();
    //overlay.enableCompass(); // не работает в эмуляторе
    overlay.runOnFirstFix(new Runnable() {
        public void run() {
```

[1] Представлено в Android 1.5.

[2] http://code.google.com/apis/maps/terms.html

```
            // Показывает текущее местоположение, изменяя масштаб
            controller.setZoom(8);
            controller.animateTo(overlay.getMyLocation());
        }
    });
    map.getOverlays().add(overlay);
}
```

Android предоставляет класс MyLocationOverlay, который делает большую часть тяжелой работы. Накладываемый объект — это что-то, что выводится поверх карты, в данном случае — это пульсирующая точка, показывающая ваше текущее местоположение. Вызывайте enableMyLocations(), чтобы сообщить накладываемому объекту о том, что он должен начать следить за обновлениями местоположения и вызвать enableCompass() для оперативного отслеживания обновлений данных от компаса.

Метод runOnFirstFix() сообщает наложенному объекту, как ему следует действовать, когда он в первый раз получит данные от провайдера навигационных данных. В этом случае мы устанавливаем масштаб и запускаем анимацию, которая перемещает позицию на карте из любого места туда, где мы сейчас расположены.

Если вы запустите программу сейчас, вы должны увидеть картинку, подобную рис. 8.3. Прикоснитесь к экрану и перетащите изображение для передвижения по карте; используйте кнопки изменения масштаба, чтобы рассмотреть карту подробнее. Когда вы будете передвигаться с телефоном, точка на карте должна следовать за вашим местоположением.

Замечания об эмуляторе

При первом запуске программы MyMap на эмуляторе вы можете столкнуться с ошибкой Android AVD. Следуйте указаниями в разделе 1.3, «Создание AVD», для того, чтобы создать новое AVD для целевой платформы «Google APIs (Goole Inc.) — API Level 8», назвав его *em22google*.

На эмуляторе вы сначала увидите карту мира в крупном масштабе без точки, показывающей ваше расположение. Как и ранее, используйте окно Emulator Control в Eclipse (или отдельную программу DDMS) для того, чтобы передать произвольные GPS-данные демонстрационному приложению.

Когда программа будет запущена в эмуляторе, вкладка компаса показана не будет, так как сенсор компаса не эмулируется.

8.4. Вперед>>

Эта глава познакомила вас с восхитительным новым миром определения местоположения и мобильными вычислениями, рассчитанными на знание положения пользователя. Эти технологии в комбинации с другими, не менее популярными, такими как внедрение широкополосного мобильного Интернета, и экспоненциальным ростом вычислительной мощности и объема хранимых данных приведут

к революционным изменениям в способах нашего взаимодействия с компьютерами и друг с другом.

Иной подход заключается в восприятии мира через зрение и слух. Android предоставляет класс[1] Camera для создания фотографий с использованием встроенной камеры (если она есть), но вы также можете использовать ее для других целей, таких как чтение штрихкодов. Класс[2] MediaRecorder позволяет записывать и хранить аудиоклипы. Описание этих возможностей выходит за рамки данной книги, но если они нужны в ваших программах, обратитесь к онлайновой документации.

Если говорить о хранении данных, следующая глава покажет, как использовать SQL для хранения структурированной информации (например, журнала путешествия, фотографий и заметок) локально на мобильном телефоне. Если это не ваша область интересов, переходите к главе 10 «3D-графика в OpenGL», чтобы узнать, как раскрыть незаметный с первого взгляда 3D-потенциал Android.

[1] http://d.android.com/reference/android/hardware/Camera.html
[2] http://d.android.com/reference/android/media/MediaRecorder.html

9 Работа с SQL

В главе 6 «Хранение локальных данных» мы исследовали хранение данных в предустановках и в обычных файлах. Это неплохо работает, пока количество данных невелико или все данные имеют один и тот же тип (например, картинки или аудиофайлы). Однако существует лучший способ хранения больших количеств структурированных данных: реляционная база данных.

Последние тридцать лет базы данных были основным продуктом корпоративной разработки приложений, но лишь до тех пор, пока они были слишком дорогими и громоздкими для использования в более мелких масштабах. Ситуация изменилась с появлением маленьких встроенных систем, одна из которых включена в платформу Android.

Эта глава покажет, как использовать встроенную в Android систему управления базами данных SQLite. Вы так же узнаете, как использовать систему связывания данных Android, чтобы соединить источники данных с пользовательским интерфейсом. И наконец, вы посмотрите на класс ContentProvider, который позволяет двум приложениям совместно использовать одни и те же данные.

9.1. Введение в SQLite

SQLite[1] — это маленькая, но мощная система управления базами данных, созданная доктором Ричардом Хиппом (Dr. Richard Hipp) в 2000 году. Это, по-видимому, самая распространенная SQL-система управления базами данных в мире. Помимо Android, SQLite можно обнаружить в Apple iPhone, телефонах на Symbian, в Mozilla Firefox, Skype, PHP, Adobe AIR, Mac OS X, Solaris и много где еще.

Существуют три причины, по которым эта система столь популярна:

- она бесплатная. Авторы разместили ее в свободном доступе и не берут плату за ее использование;
- она маленькая. Текущая версия занимает примерно 150 Кбайт, что весьма соответствует ограничениям памяти телефонов Android;

[1] http://www.sqlite.org

- ей не нужна установка или администрирование. У нее нет сервера, конфигурационного файла, и она не нуждается в администраторе баз данных.

ЛИЦЕНЗИЯ SQLITE

Исходный код SQLite не содержит лицензии, так как он находится в свободном доступе. Вместо лицензии в коде содержатся напутствия:

May you do good and not evil. (Желаем вам делать добро, а не зло.)

May you find forgiveness for yourself and forgive others. (Желаем вам найти прощение для себя и прощать других.)

May you share freely, never taking more than you give. (Желаем вам легко делиться, никогда не беря больше, чем вы отдаете.)

База данных SQLite — это обычный файл. Вы можете брать этот файл, перемещать и даже копировать его на другую систему (например, с телефона на рабочий компьютер), и она будет отлично работать. Android хранит файл в папке /data/data/packagename/databases (рис. 9.1). Вы можете использовать команду adb в окне File Explorer (Window ▸ Show View ▸ Other... ▸ Android ▸ File Explorer) для его просмотра, перемещения или удаления.

Вместо того чтобы вызывать подпрограммы Java I/O для доступа к этому файлу, вы запускаете команды Structured Query Language (SQL — структурированный язык запросов). Посредством вспомогательных классов и удобных методов Android скрывает часть деталей, но вам нужно немного знать об SQL, чтобы пользоваться этими инструментами.

9.2. SQL 101

Если вы пользовались Oracle, SQL Server, MySQL, DB2 или другими системами управления базами данных, тогда SQL должен быть вашим старым знакомым. Вы можете пропустить этот раздел и сразу перейти к разделу 9.3 «Привет, база данных». Специально для остальных мы быстро повторим материал.

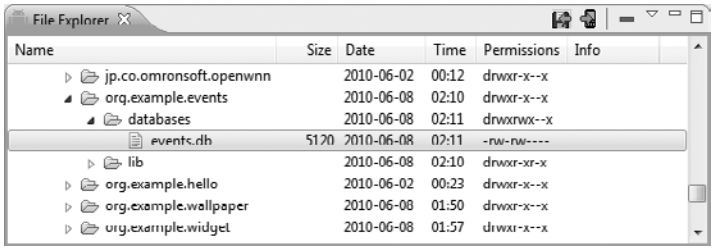

Рис. 9.1. SQLite хранит всю базу данных в одном файле

При использования SQL-базы данных вы отправляете SQL-запросы и получаете результаты. Существует три основных вида SQL-запросов: DDL, Modification и Query.

DDL-запросы

Файл базы данных может иметь любое количество таблиц. Таблица состоит из строк, а каждая из строк может иметь определенное количество столбцов. Каждый столбец в таблице имеет имя и тип данных (текстовая строка, число и т. д.). Вы определяете эти таблицы и имена столбцов в начале работы, запуская запрос Data Definition Language (DDL — язык определения данных). Вот запрос, который создает таблицу с тремя столбцами:

SQLite/create.sql

```
create table mytable (
    _id integer primary key autoincrement,
    name text,
    phone text );
```

Один из столбцов обозначен как PRIMARY KEY (первичный ключ), то есть уникальное число, которое однозначно идентифицирует строку. AUTOINCREMENT подразумевает то, что база данных будет добавлять 1 к ключу каждой предыдущей записи для того, чтобы соблюсти их уникальность. По соглашению, первый столбец всегда называется _id. Столбец _id — это не жесткое требование SQLite, однако позже, когда мы захотим использовать объект Android ContentProvider, нам это понадобится.

Заметьте, что в SQLite, в отличие от многих других баз данных, типы данных столбцов — это лишь подсказка. Если вы попытаетесь сохранить строку в столбце, предназначенном для хранения целых чисел, или наоборот, это будет нормально работать. Авторы SQLite говорят об этом как об особенности базы данных, а не как об ошибке.

Modification-запросы

SQL предоставляет множество выражений запросов, которые позволяют вставлять, удалять и изменять записи в базе данных. Например, для того чтобы добавить несколько телефонных номеров, вы можете использовать следующий код:

SQLite/insert.sql

```
insert into mytable values(null, 'Steven King' , '555-1212' );
insert into mytable values(null, 'John Smith' , '555-2345' );
insert into mytable values(null, 'Fred Smitheizen' , '555-4321' );
```

Значения задаются в том же порядке, который вы использовали в запросе CREATE TABLE. Мы задаем NULL для _id, так как SQLite вычисляет это значение для нас.

Query-запросы

После того как данные были помещены в таблицу, вы запускаете запросы к таблице, используя выражение SELECT. Например, если вы хотите получить третью запись, вы должны отправить следующий запрос:

SQLite/selectid.sql

```
select * from mytable where(_id=3);
```

Скорее всего, вам захочется найти номер телефона по имени его владельца. Вот как вы можете найти все записи, содержащие «Smith» в имени:

SQLite/selectwhere.sql

```
select name, phone from mytable where(name like "%smith%" );
```

Помните, что SQL чувствителен к регистру. Ключевые слова, имена столбцов и даже строки для поиска могут быть указаны в верхнем или в нижнем регистре.

Сейчас вы уже знаете достаточно про SQL, чтобы представлять себе возможные опасности при работе с ним. Посмотрим, как использовать эти знания в работе: построим простую программу.

9.3. Привет, база данных

Для демонстрации возможностей SQLite создадим маленькое приложение, которое называется Events и занимается хранением записей в базе данных и выводом их на экран. Мы собираемся начать с каркаса приложения и добавлять к нему новые возможности. Создайте новую программу «Hello, Android», используя следующие значения в мастере проектов:

```
Project name: Events
Build Target: Android 2.2
Application name: Events
Package name: org.example.events
Create Activity: Events
Min SDK Version: 8
```

Как обычно, вы можете загрузить полный исходный код с веб-сайта издательства «Питер» (www.piter.com).

Нам нужно кое-что сохранить в нескольких константах, описывающих базу данных, поэтому создадим интерфейс Constants:

Eventsv1/src/org/example/events/Constants.java

```
package org.example.events;
import android.provider.BaseColumns;
public interface Constants extends BaseColumns {
    public static final String TABLE_NAME = "events" ;
    // Столбцы в базе данных Events
    public static final String TIME = "time" ;
    public static final String TITLE = "title" ;
}
```

Каждое событие будет сохранено в виде строки в таблице events. Каждая строка будет иметь столбцы _id, time и title. _id — это первичный ключ, объявленный в интерфейсе BaseColumns, который мы расширили. time и title будут использованы для хранения отметки времени и названия события соответственно.

Использование SQLiteOpenHelper

Далее мы создадим вспомогательный класс, названный EventsData, для представления базы данных. Этот класс расширяет возможности класса SQLiteOpenHelper, который управляет созданием базы данных и ее версиями. Все, что вам нужно, — это создать конструктор и переопределить пару методов:

Eventsv1/src/org/example/events/EventsData.java

```
1    package org.example.events;
-
-        import static android.provider.BaseColumns._ID;
-        import static org.example.events.Constants.TABLE_NAME;
5        import static org.example.events.Constants.TIME;
-        import static org.example.events.Constants.TITLE;
-        import android.content.Context;
-        import android.database.sqlite.SQLiteDatabase;
-        import android.database.sqlite.SQLiteOpenHelper;
10
-        public class EventsData extends SQLiteOpenHelper {
-            private static final String DATABASE_NAME = "events.db" ;
-            private static final int DATABASE_VERSION = 1;
15           /** Создает вспомогательный объект для базы данных Events */
-            public EventsData(Context ctx) {
-                super(ctx, DATABASE_NAME, null, DATABASE_VERSION);
-            }
-
20           @Override
-            public void onCreate(SQLiteDatabase db) {
-                db.execSQL("CREATE TABLE " + TABLE_NAME + " (" + _ID
-                    + " INTEGER PRIMARY KEY AUTOINCREMENT, " + TIME
-                    + " INTEGER," + TITLE + " TEXT NOT NULL);" );
25           }
-
-            @Override
-            public void onUpgrade(SQLiteDatabase db, int oldVersion,
-                int newVersion) {
30               db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
-                onCreate(db);
-            }
-        }
```

Код конструктора начинается в строке 16. DATABASE_NAME — это реальное имя файла базы данных (events.db), которое мы будем использовать, а DATABASE_VERSION — это просто заданный нами номер. Если это будет реальная программа, вы можете увеличивать номер версии всякий раз, когда производите значительные изменения в структуре базы данных (например, добавляете новый столбец).

Когда вы попытаетесь получить доступ к базе данных в первый раз, SQLiteOpenHelper обнаружит, что она еще не существует, и вызовет метод onCreate(), чтобы создать ее. В строке 21 мы переопределяем его и запускаем запрос SQL CREATE

TABLE. Благодаря этому будут созданы таблица events и база данных events.db, которая содержит эту таблицу.

Когда Android обнаружит, что вы ссылаетесь на старую базу данных (основываясь на номере версии), он вызовет метод onUpgrade() (строка 28). В этом примере мы просто удаляем старую таблицу, но вы можете сделать здесь и кое-что получше. Например, вы можете запустить SQL-команду ALTER TABLE, чтобы добавить столбец в существующую таблицу базы данных.

Создание основной программы

В программе Events мы использовали базу данных SQLite для хранения событий, и она показывала их в виде строк внутри TextView.

ВОПРОС/ОТВЕТ

Почему константы и интерфейс?

Это технологии, характерные для Java. Не знаю, как вы, но я не люблю повторять имя класса каждый раз, поэтому использую константы. Например, я хочу лишь ввести TIME, а не Constant.TIME. Обычно, для того чтобы это сделать, Java использует интерфейсы. Классы могут наследовать интерфейс Constants и затем опускать имя интерфейса, ссылаясь на любые поля. Если вы посмотрите на интерфейс BaseColumn, вы увидите, что программисты под Android пользуются тем же приемом.

Начиная с Java 5 существует лучший способ: статические импорты. Именно этот метод я буду использовать в классе EventsData и в других классах в этой главе. Поскольку Constants — это интерфейс, вы можете использовать его либо по-старому, либо по-новому — как вам больше нравится.

К несчастью, как известно, Eclipse не очень хорошо поддерживает статические импорты, поэтому если вы будете использовать их в своих программах, Eclipse может не вставить команды импорта автоматически в нужные места. Вот небольшой совет для пользователей Eclipse: введите шаблон статического импорта после команды пакета (например, import static org.example.events.Constants.*;) для того, чтобы программа нормально скомпилировалась. Позже вы можете использовать импорты Source ▸ Organize для расширения набора шаблонных символов и видов команд импорта. Будем надеяться, это будет реализовано более удобно в будущих версиях Eclipse.

Отредактируйте файл макета (layout/main.xml), как показано далее:

Eventsv1/res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <TextView
        android:id="@+id/text"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
</ScrollView>
```

Здесь объявлен TextView с понятным ID text (R.id.text в коде), который помещен в ScrollView на тот случай, если будет выведено слишком много событий для одного экрана. Вы можете видеть, как это выглядит, на рис. 9.2.

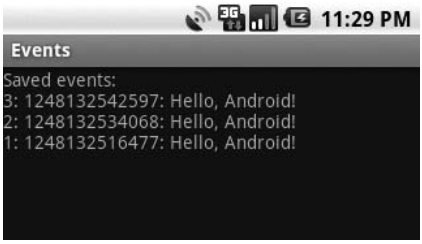

Рис. 9.2. Первая версия программы показывает записи базы данных в TextView

Основная программа — это метод onCreate() в деятельности Events. Вот его каркас:

Eventsv1/src/org/example/events/Events.java

```
1     package org.example.events;
-
-         import static android.provider.BaseColumns._ID;
-         import static org.example.events.Constants.TABLE_NAME;
5         import static org.example.events.Constants.TIME;
-         import static org.example.events.Constants.TITLE;
-         import android.app.Activity;
-         import android.content.ContentValues;
-         import android.database.Cursor;
10         import android.database.sqlite.SQLiteDatabase;
-         import android.os.Bundle;
-         import android.widget.TextView;
-
-         public class Events extends Activity {
15             private EventsData events;
-
-             @Override
-             public void onCreate(Bundle savedInstanceState) {
-                 super.onCreate(savedInstanceState);
20                  setContentView(R.layout.main);
-                 events = new EventsData(this);
-                 try {
-                     addEvent("Hello, Android!" );
-                     Cursor cursor = getEvents();
25                      showEvents(cursor);
-                 } finally {
-                     events.close();
-                 }
-             }
30         }
```

В строке 20 метода onCreate() мы устанавливаем макет для этого окна просмотра. Затем мы создаем экземпляр класса EventsData в строке 21 и начинаем блок try. Если вы посмотрите далее, на строку 27, вы можете увидеть, что мы закрываем базу данных внутри блока finally. Так, если даже ошибки возникнут где-то посередине, база данных будет закрыта.

Таблица events не будет интересна, если в ней не будет событий, поэтому в строке 23 мы вызываем метод addEvent() для добавления в нее событий. Каждый раз, когда вы запускаете программу, вы получаете новое событие. Вы можете добавить меню, команды, вводимые жестами или с клавиатуры, для создания других событий, если хотите, но я оставил это в качестве упражнения для читателя.

В строке 24 мы вызываем метод getEvents() для получения списка событий, и, наконец, в строке 25 мы вызываем метод showEvents() для показа списка событий пользователю.

Весьма просто, так ведь? Теперь давайте создадим эти новые методы, которые мы уже использовали.

Добавление строки

Метод addEvent() помещает новую запись в базу данных, используя переданную ему строку в качестве заголовка события.

Eventsv1/src/org/example/events/Events.java

```
private void addEvent(String string) {
    // Вставляем новую запись в источник данных Event.
    // Вы должны делать что-то подобное для удаления или обновления.
    SQLiteDatabase db = events.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(TIME, System.currentTimeMillis());
    values.put(TITLE, string);
    db.insertOrThrow(TABLE_NAME, null, values);
}
```

Если нам понадобится модифицировать данные, мы вызовем getWritableDatabase() для получения обработчиков чтения/записи для базы данных events. Обращения к базе данных кэшируются, поэтому вы можете вызывать этот метод так часто, как вам нужно.

Далее мы заполним объект ContentValues сведениями о текущем времени и заголовком события и передадим его методу insertOrThrow() для выполнения реального запроса SQL INSERT. Вам не нужно передавать ID записи, так как SQLite создаст его и возвратит из вызова метода.

Как можно понять из имени, insertOrThrow() может выбрасывать исключение (типа SQLException), если произойдет ошибка. Его не нужно объявлять с помощью ключевого слова throws, так как это RuntimeException (исключение времени выполнения), а не проверенное исключение. Однако если хотите, можете обработать его в блоке try/catch, как и любое другое исключение. Если вы не обработаете его и это

окажется ошибкой, программа будет завершена и информация об этом будет записана в журнал Android.

По умолчанию, как только вы добавите новые данные, база данных будет объявлена. Если вам по каким-либо причинам требуется групповое добавление или отложенная модификация данных, обратитесь к веб-сайту SQLite за подробностями.

Выполнение запросов

Метод getEvents() выполняет запрос к базе данных на получение списка событий:

Eventsv1/src/org/example/events/Events.java

```
private static String[] FROM = { _ID, TIME, TITLE, };
private static String ORDER_BY = TIME + " DESC" ;
private Cursor getEvents() {
    // Производит управляемый запрос. Activity обработает закрытие
    // и повторный запрос указателя, когда это нужно.
    SQLiteDatabase db = events.getReadableDatabase();
    Cursor cursor = db.query(TABLE_NAME, FROM, null, null, null,
        null, ORDER_BY);
    startManagingCursor(cursor);
    return cursor;
}
```

Нам не нужно изменять базу данных для запроса, поэтому мы вызываем getReadableDatabase() для получения доступа только для чтения. Затем мы вызываем метод query() для проведения реального запроса SQL SELECT. FROM — это массив столбцов, который нам нужен, а ORDER_BY сообщает SQLite о том, что необходимо возвратить результаты в порядке от более новых к более старым.

Хотя мы не используем их в этом примере, метод query() имеет параметры, которые задают условия WHERE, GROUP BY и HAVING. В действительности query() — это лишь удобство для программистов. В соответствии с собственными предпочтениями вы можете построить запрос SELECT собственными силами, в виде строки, и использовать метод rawQuery() для его исполнения. В любом случае, возвращаемое значение — это объект Cursor, который представляет результирующий набор данных.

Cursor похож на Iterator Java или на ResultSet JDBC. Вы вызываете его внутренние методы, чтобы получить информацию о текущей строке, затем вы вызываете другой метод для перемещения к следующей строке. Мы скоро узнаем, как им пользоваться, когда будем отображать результаты.

Последний шаг — это вызов startManagingCursor(), который сообщает деятельности о том, что она должна позаботиться о жизненном цикле указателя, основываясь на жизненном цикле деятельности. Например, когда деятельность приостановлена, он автоматически деактивирует указатель и затем повторит запрос, когда деятельность будет перезапущена. Когда деятельность завершается, все управляемые указатели будет закрыты.

Отображение результатов запроса

Последний метод, который нам осталось определить, — это showEvent(). Эта функция принимает Cursor на входе и форматирует выходные данные таким образом, чтобы пользователь мог их воспринять.

Eventsv1/src/org/example/events/Events.java

```
1     private void showEvents(Cursor cursor) {
-             // Собираем все в одну большую строку
-             StringBuilder builder = new StringBuilder(
-                 "Saved events:\n" );
5             while (cursor.moveToNext()) {
-                 // Можно использовать getColumnIndexOrThrow() для получения
индексов
-                 long id = cursor.getLong(0);
-                 long time = cursor.getLong(1);
-                 String title = cursor.getString(2);
10                 builder.append(id).append(": " );
-                 builder.append(time).append(": " );
-                 builder.append(title).append("\n" );
-             }
-             // Отображение на экране
15             TextView text = (TextView) findViewById(R.id.text);
-             text.setText(builder);
-         }
```

В этой версии Events мы просто собираемся создать большую строку (см. строку 3), которая будет хранить все элементы событий, разделенные переводами строки. Так делать не рекомендуется, но сейчас это работает.

Строка 5 вызывает метод Cursor.moveToNext() для перемещения к следующей строке в наборе данных. Когда вы впервые получаете объект Cursor, указатель расположен перед первой записью, поэтому вызов moveToNext() предоставляет первую запись. Мы работаем в цикле до тех пор, пока moveToNext() не возвратит false, что означает, что строк больше нет.

Внутри цикла (строка 7) мы вызываем методы getLong() и getString(), чтобы извлечь данные из интересующих нас столбцов, и затем присоединяем найденные значения к строке (строка 10). Есть и другой метод в Cursor — getColumnIndexOrThrow(), который мы можем использовать для получения индексов колонок (значения 0, 1 и 2 передаются в getLong() и getString()). Однако этот метод работает медленнее, поэтому, если вам это нужно, вам следует вызывать его вне цикла и запоминать индексы самостоятельно.

Как только строки данных будут обработаны, мы ищем TextView из layout/main.xml и помещаем большую строку в него (строка 15).

Если сейчас вы запустите пример, вы должны увидеть что-то вроде рис. 9.2. Примите поздравления с вашей первой программой под Android, работающей с базой данных! Хотя, конечно, здесь много чего можно доработать.

Что произойдет, если в списке будут тысячи или миллионы событий? Программа будет очень медленной, ей может не хватить памяти в попытке построить строку,

способную все это вместить. Что, если вы захотите дать пользователю возможность выбора одного события и выполнения с ним каких-либо действий? Если все будет в одной строке, вы этого сделать не сможете. К счастью, Android обеспечивает лучший путь: связывание данных.

9.4. Связывание данных

Связывание данных позволяет подключить модель (данные) к вашему окну просмотра с помощью всего нескольких строк кода. С целью демонстрации связывания данных мы модифицируем пример Events для использования ListView, который ограничен результатами запроса к базе данных. Для начала нам нужно расширить класс Events до ListActivity вместо Activity:

Eventsv2/src/org/example/events/Events.java

```
import android.app.ListActivity;
// ...
public class Events extends ListActivity {
    // ...
}
```

Далее следует изменить способ отображения событий в методе Events.showEvents().

Eventsv2/src/org/example/events/Events.java

```
import android.widget.SimpleCursorAdapter;
// ...
    private static int[] TO = { R.id.rowid, R.id.time, R.id.title, };
    private void showEvents(Cursor cursor) {
        // Настройка связывания данных
        SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
            R.layout.item, cursor, FROM, TO);
        setListAdapter(adapter);
    }
```

Обратите внимание на то, что этот код гораздо меньше, чем предыдущий (две строки против десяти). Первая строка создает SimpleCursorAdapter для объекта Cursor, а вторая сообщает ListActivity о необходимости использования нового адаптера. Адаптер работает как посредник, соединяя окно просмотра с его источником данных.

Если вы помните, мы впервые использовали адаптер в примере Translate (см. Translate.setAdapters() в разделе 7.4 «Использование веб-сервисов»). В этом примере мы используем ArrayAdapter, так как источник данных — это массив, определенный в XML. Для него мы используем SimpleCursorAdapter, так как источник данных — это объект Cursor, который создан из запроса к базе данных.

Конструктор для SimpleCursorAdapter принимает пять параметров:

- context: ссылка на текущую деятельность;
- layout: ресурс, который определяет окно просмотра для отдельного списка элементов;

- cursor: указатель набора данных;
- from: список имен столбцов, из которых поступают данные;
- to: список областей просмотра, куда поступают данные.

Макет из списка элементов определен в layout/item.xml. Обратите внимание на то, что определения областей просмотра для ID, time, title передаются в массиве TO.

Eventsv2/res/layout/item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal"
    android:padding="10sp" >
    <TextView
        android:id="@+id/rowid"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView
        android:id="@+id/rowidcolon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=": "
        android:layout_toRightOf="@id/rowid" />
    <TextView
        android:id="@+id/time"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/rowidcolon" />
    <TextView
        android:id="@+id/timecolon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=": "
        android:layout_toRightOf="@id/time" />
    <TextView
        android:id="@+id/title"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:ellipsize="end"
        android:singleLine="true"
        android:textStyle="italic"
        android:layout_toRightOf="@id/timecolon" />
</RelativeLayout>
```

Это выглядит более сложным, чем есть на самом деле. Все, что мы делаем, — это помещаем ID, time и title в одну строку с двоеточиями между полями. Я добавил не-

большие дополнения и форматирование для того, чтобы улучшить внешний вид данных.

Наконец, нам нужно изменить макет для деятельности в layout/main.xml. Вот новая версия:

Eventsv2/res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <!—Обратите внимание на то, что встроенные id для 'list' и 'empty' -->
    <ListView
        android:id="@android:id/list"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView
        android:id="@android:id/empty"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/empty" />
</LinearLayout>
```

Так как деятельность расширяет ListActivity, Android ищет два особых идентификатора в файле макета. Если список их имеет, отображается окно просмотра android:id/list; в противном случае отображается вьювер android:id/empty. Так, если здесь нет элементов, вместо пустого экрана пользователь видит сообщение «No events!».

Вот строковые ресурсы, которые нам нужны (см. дополнительные материалы к книге Eventsv2/res/values/strings.xml):

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Events</string>
    <string name="empty">No events!</string>
</resources>
```

Для того чтобы увидеть окончательный результат, посмотрите на рис. 9.3. В качестве упражнения подумайте о том, как можно расширить это приложение сейчас, когда у вас есть реальный список для экспериментов. Например, когда пользователь выберет событие, вы можете открыть окно подробного просмотра, отправить событие в техподдержку или, возможно, удалить выбранное событие и все события, находящиеся ниже него, из базы данных.

Однако есть еще одна маленькая проблема с этим примером. Другие приложения не могут добавлять данные о событиях в базу или даже просматривать их! Для этого нам нужно использовать механизм Android ContentProvider.

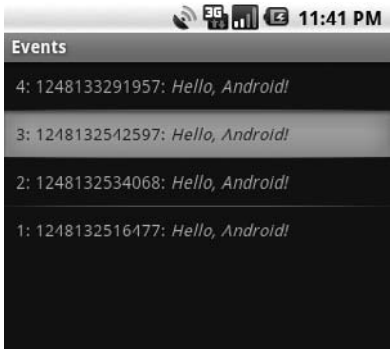

Рис. 9.3. Эта версия использует ListActivity для связывания данных

9.5. Использование ContentProvider

В модели безопасности Android (см. обсуждение в разделе 2.5 «Безопасность и защищенность») файлы, записанные одним приложением, нельзя прочитать или изменить из другого приложения. Каждая программа имеет свой собственный идентификатор пользователя Linux, папку с данными (data/data/packagename) и собственную защищенную область памяти. Программы Android могут взаимодействовать друг с другом двумя способами:

- *Inter-Process Communication (IPC — межпроцессная коммуникация)*: один процесс объявляет произвольный API, используя Android Interface Definition Language (AIDL — язык Android для определения интерфейсов) и интерфейс IBinder. Параметры эффективно и безопасно передаются между процессами при вызове API. Эта продвинутая техника работы используется для удаленных вызовов процедуры в фоновом процессе Service[1].
- *ContentProvider (контент-провайдеры)*: процессы регистрируются в системе как провайдеры определенного вида данных. Когда подобная информация запрашивается, они вызываются Android через фиксированные API для получения или изменения данных любым способом, который им подходит. Эта технология, которую мы собираемся использовать в примере Events.

Любая информация, управляемая ContentProvider, адресуется посредством URI, выглядящего примерно так:

```
content://authority/path/id
```

где:

- content:// — стандартный требуемый префикс;
- authority — имя провайдера. Здесь рекомендовано использование полного квалификационного имени пакета для предотвращения конфликтов имен;

[1] IPC, сервисы и связывание данных выходят за рамки данной книги. Для того чтобы найти дополнительную информацию, обратитесь к http://d.android.com/guide/developing/tools/aidl.html, http://d.android.com/reference/android/app/Service.html, и http://d.android.com/reference/android/os/IBinder.html.

- path — виртуальная папка внутри провайдера, которая определяет вид запрашиваемых данных;
- id — первичный ключ отдельной запрошенной записи. Для запрашивания всех записей определенного типа опустите id и завершающий слэш.

Android поставляется с несколькими встроенными провайдерами, в их числе[1]:

- content://browser;
- content://contacts;
- content://media;
- content://settings.

В целях демонстрации использования ContentProvider преобразуем пример Events для использования одного из провайдеров. Для нашего провайдера Events будут правильными следующие URI:

```
content://org.example.events/events/3 — отдельное событие с _id=3
content://org.example.events/events — все события
```

Для начала нам нужно добавить две дополнительные константы в Constant.java:

Eventsv3/src/org/example/events/Constants.java

```
import android.net.Uri;
// ...
    public static final String AUTHORITY = "org.example.events" ;
    public static final Uri CONTENT_URI = Uri.parse("content://"
        + AUTHORITY + "/" + TABLE_NAME);
```

Файлы макетов (main.xml и item.xml) не нуждаются в изменениях, поэтому следующий шаг заключается во внесении небольших изменений в класс Event.

Изменение основной программы

Основная программа (метод Events.onCreate()) в действительности станет немного проще, так как здесь не будет объекта базы данных, с которым нужно работать:

Eventsv3/src/org/example/events/Events.java

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    addEvent("Hello, Android!" );
    Cursor cursor = getEvents();
    showEvents(cursor);
}
```

Нам не нужен блок try/finally, и мы можем удалить ссылки на EventData.

[1] Для того чтобы увидеть обновленный список, обратитесь к странице http://d.android.com/reference/android/provider/package-summary.html. Вместо использования строк применяйте документированные константы, такие как Browser.BOOKMARKS_URI. Обратите внимание на то, что доступ к некоторым провайдерам требует особых разрешений, которые следует запросить в файле манифеста.

Добавление строки

Две строки изменились в addEvent(). Вот новая версия:

Eventsv3/src/org/example/events/Events.java

```
import static org.example.events.Constants.CONTENT_URI;
    private void addEvent(String string) {
        // Вставляем новую запись в источник данных Events.
        // Вы можете сделать что-то подобное для удаления и обновления.
        ContentValues values = new ContentValues();
        values.put(TIME, System.currentTimeMillis());
        values.put(TITLE, string);
        getContentResolver().insert(CONTENT_URI, values);
}
```

Вызов getWritableDatabase() удален, вызов insertOrThrow() заменен на getContentResolver().insert(). Вместо указателя на базу данных мы используем контентный URI.

Выполнение запроса

Метод getEvents() тоже упрощен при использовании ContentProvider:

Eventsv3/src/org/example/events/Events.java

```
private Cursor getEvents() {
    // Выполнение управляемого запроса. Activity обработает завершение программы
    // и заново поставит в очередь указатель, когда это будет нужно.
    return managedQuery(CONTENT_URI, FROM, null, null, ORDER_BY);
}
```

Здесь мы используем метод Activity.managedQuery(), передавая ему контентный URI, список интересующих нас столбцов и порядок, в котором они должны быть отсортированы.

Удаляя все ссылки на базу данных, мы разделяем клиентское приложение Events и поставщик данных Events. Клиентская часть упрощается, но сейчас мы должны создать то, чего у нас не было раньше.

9.6. Создание реализации ContentProvider

ContentProvider — это высокоуровневый объект, как Activity, который нужно объявить для использования в системе. Так, первым шагом при его создании будет добавление его в файл AndroidManifest.xml перед тегом <activity> (в качестве объекта-потомка <application>):

Eventsv3/AndroidManifest.xml

```
<provider android:name=".EventsProvider"
    android:authorities="org.example.events" />
```

android:name — имя класса (присоединенное к имени пакета в манифесте), android:authorities — строка, использованная в контентном URI.

Далее мы создаем класс EventsProvider, который должен расширить ContentProvider. Вот его основной каркас:

Eventsv3/src/org/example/events/EventsProvider.java

```
package org.example.events;
import static android.provider.BaseColumns._ID;
import static org.example.events.Constants.AUTHORITY;
import static org.example.events.Constants.CONTENT_URI;
import static org.example.events.Constants.TABLE_NAME;
import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.text.TextUtils;
public class EventsProvider extends ContentProvider {
    private static final int EVENTS = 1;
    private static final int EVENTS_ID = 2;
    /** Тип MIME каталога событий */
    private static final String CONTENT_TYPE
        = "vnd.android.cursor.dir/vnd.example.event" ;
    /** Тип MIME отдельного события */
    private static final String CONTENT_ITEM_TYPE
        = "vnd.android.cursor.item/vnd.example.event" ;
    private EventsData events;
    private UriMatcher uriMatcher;
    // ...
}
```

По соглашению мы используем vnd.example вместо org.example в MIME-типе[1]. EventsProvider поддерживает два типа данных:

- EVENTS (MIME-тип CONTENT_TYPE): каталог или список событий;
- EVENTS_ID (MIME-тип CONTENT_ITEM_TYPE): отдельное событие.

В понятиях URI разница заключается в том, что первый тип не предусматривает использования ID, а второй — предусматривает. Мы используем класс Android UriMatcher для обработки URI и получения сведений о спецификации клиента. И мы повторно используем класс EventsData для управления реальной базой данных в контент-провайдере.

В интересах экономии места я не собираюсь приводить остальной код класса, но вы можете загрузить полный пример с веб-сайта издательства «Питер» (www.piter.com). Все три версии примера Example можно найти в .zip-файле с исходным кодом.

[1] Multipurpose Internet Mail Extensions (MIME) — интернет-стандарт для описания типа любого вида контента.

Последняя версия примера Events внешне выглядит точно так же, как предыдущая (рис. 9.3). Однако внутри вы сейчас имеете фреймворк для хранения событий, которые могут быть использованы другими приложениями в системе, даже теми, которые написаны другими разработчиками.

9.7. Вперед>>

В этой главе мы узнали, как хранить данные в базе данных SQL Android. Если вы хотите сделать с помощью SQL что-нибудь более серьезное, вам стоит поискать дополнительную информацию о запросах и выражениях, часть из которых мы здесь рассмотрели. Книги, такие как SQL Pocket Guide [Gen06] Jonathan Gennick или The Definitive Guide to SQLite [Owe06] Mike Owens, будут неплохим вложением денег, но помните, что синтаксис SQL и функции слегка отличаются у различных баз данных.

Другая возможность для хранения данных в Android — это *db4o*[1]. Эта библиотека больше, чем SQLite, и использует другую лицензию (GNU Public License), но она бесплатна и, возможно, вам легче будет с ней работать, особенно если вы не знаете SQL.

SimpleCursorAdapter, представленный в этой главе, может быть настроен для того, чтобы отображать не только обычный текст. Например, вы можете отображать звездочки рейтинга или искры либо другие объекты, основываясь на данных в объекте Cursor. Поищите информацию[2] по ViewBinder в документации к SimpleCursorAdapter.

А теперь поговорим о совершенно другом... Следующая глава будет посвящена 3D-графике с использованием OpenGL.

[1] http://www.db4o.com/android

[2] http://d.android.com/reference/android/widget/SimpleCursorAdapter.html

3D-графика в OpenGL

10

Двумерная графика отлично подходит для большинства программ, но иногда бывает необходимо дополнительное измерение для придания приложению глубины, интерактивности или реализма, недостижимого в 2D. Для таких случаев Android предлагает библиотеку трехмерной графики, основанную на стандарте OpenGL ES. В этой главе мы рассмотрим концепции 3D и создадим демонстрационную программу, которая использует OpenGL.

10.1. Основы 3D-графики

Мир трехмерен, но мы постоянно видим его в двух измерениях. Когда вы смотрите телевизор или просматриваете картинку в книге, 3D-изображения располагаются или проецируются на 2D-поверхность (экран телевизора или страница книги).

Проведите простой эксперимент: закройте один глаз и посмотрите в окно. Что вы видите? Свет солнца отражается от объектов, находящихся снаружи, проходит через оконное стекло и попадает в ваш глаз — так, что вы можете его воспринять. В понятиях графики, сцена из внешнего мира проецируется на окно (или *viewport* — окно просмотра). Если кто-нибудь заменит ваше окно на высококачественную фотографию, она будет выглядеть так же до тех пор, пока вы не переместитесь.

В зависимости от того, как близко глаз расположен к окну и насколько окно велико, вы можете видеть лишь ограниченное пространство внешнего мира. Это называется полем зрения (field of view). Если вы проведете воображаемые линии от глаза к четырем углам окна и дальше, вы получите пирамиду как на рис. 10.1. Она называется *view frustum* (пирамида видимости) (латинское слово для «разбитого на куски»). С целью повышения производительности усеченная пирамида видимости обычно ограничивается дальними и ближними плоскостями отсечения. Вы можете видеть все внутри пирамиды, но ничего вне ее.

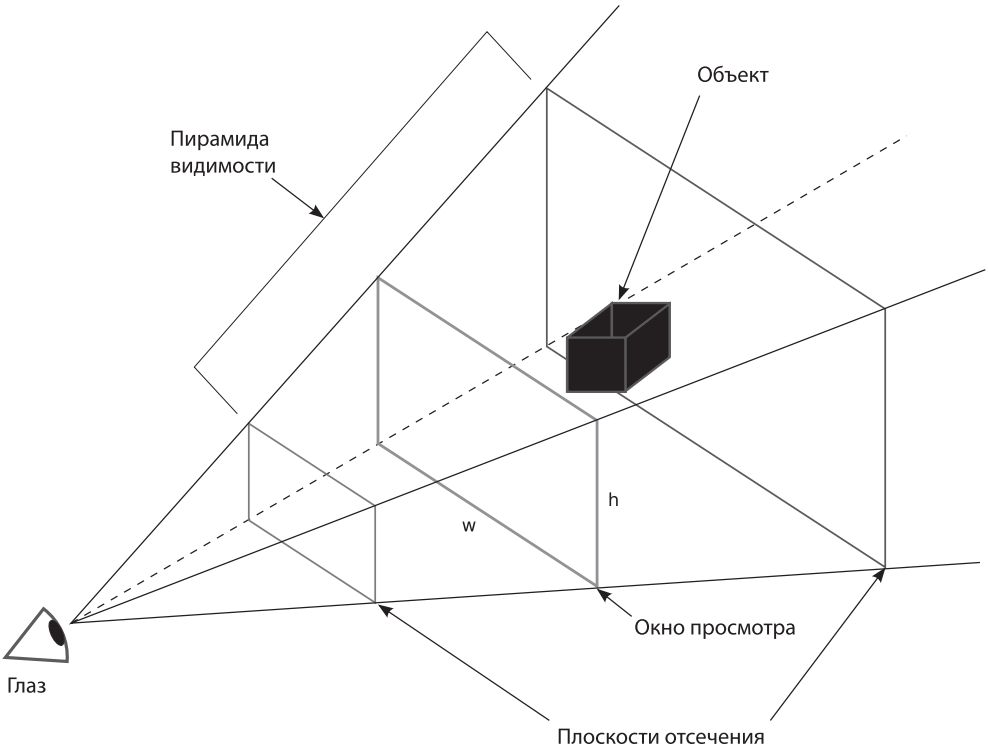

Рис. 10.1. Просмотр трехмерной сцены

В трехмерной компьютерной графике экран компьютера действует как окно просмотра. И ваша задача заключается в том, чтобы убедить пользователя в том, что это — окно в другой мир, который находится по ту сторону стекла. Графическая библиотека OpenGL — это API, который позволяет решить эту задачу.

10.2. Введение в OpenGL

Технология OpenGL[1] была разработана компанией Silicon Graphics в 1992 году. Она обеспечивает унифицированный интерфейс для программистов, который позволяет пользоваться возможностями аппаратного обеспечения любого производителя. OpenGL использует уже знакомые вам концепции, такие как окно просмотра, освещение, и пытается скрыть как можно большую часть аппаратного уровня от разработчика.

Так как технология OpenGL была разработана для настольных компьютеров, она слишком громоздка, чтобы подойти мобильным устройствам. Поэтому в Android применяется разновидность OpenGL, которая называется OpenGL for

[1] http://www.opengl.org

Embedded Systems (OpenGL ES)[1]. Этот стандарт был создан Khronos Group, индустриальным консорциумом компаний, таких как Intel, AMD, Nvidia, Nokia, Samsung и Sony. Та же самая библиотека (с небольшими отличиями) доступна теперь на основных мобильных платформах, включая Android, Symbian и iPhone.

СПАСИБО ВАМ, ДЖОН КАРМАК

OpenGL оказался весьма успешным проектом, но не нашел поначалу широкого применения. В 1995 году Microsoft представила конкурента, который назывался Direct3D. Благодаря доминирующей позиции Microsoft на рынке и значительным вложениям в дизайн и разработку программного обеспечения через некоторое время оказалось, что Direct3D превратился в стандарт де-факто для игровой индустрии. Однако один человек, Джон Кармак (John Carmack), один из основателей id Software, отказался подчиниться. Его популярнейшие игры Doom и Quake почти единолично принудили производителей аппаратного обеспечения поддерживать OpenGL-драйверы для PC в актуальном состоянии. Сейчас пользователи Linux, Mac OS X и мобильных устройств могут сказать спасибо Джону и id Software за то, что они помогли поддерживать стандарт OpenGL в актуальном состоянии.

Каждый язык имеет собственные языковые привязки к OpenGL ES, и Java — не исключение. Язык привязки Java определен в Java Specification Request (JSR) 239[2]. Android следует этому стандарту настолько точно, насколько это возможно; существует огромное количество книг и документов по JSR 239 и OpenGL ES с полным описанием всех его классов и методов.

Теперь давайте посмотрим, как создать простую OpenGL-программу на Android.

10.3. Создание OpenGL-программы

Начнем с создания нового проекта «Hello, Android», как в разделе 1.2 «Создание первой программы», но в этот раз введем следующие значения в диалоговое окно New Android Project:

```
Project name: OpenGL
Build Target: Android 2.2
Application name: OpenGL
Package name: org.example.opengl
Create Activity: OpenGL
Min SDK Version: 8
```

ВО ВСЕХ ЛИ ТЕЛЕФОНАХ ЕСТЬ ПОДДЕРЖКА 3D-ГРАФИКИ?

И да, и нет. Некоторые дешевые устройства, которые работают на Android, могут не иметь аппаратного обеспечения, поддерживающего трехмерную графику. Однако программный интерфейс OpenGL у них есть. Все 3D-функции эмулируются программно. Ваша программа будет работать, но это будет гораздо медленнее, чем на устройствах с аппаратным ускорением 3D. По этой причине было бы неплохо предоставить пользователю возможности отключать определенные детали и спецэффекты, которые занимают ресурсы, но не являются необходимыми для программы. Поэтому если пользователь запускает вашу программу на медленном устройстве, он должен иметь возможность пожертвовать «красивостями» ради производительности.

[1] http://www.khronos.org/opengles

[2] http://jcp.org/en/jsr/detail?id=239

Благодаря этим параметрам будет создан OpenGL.java, содержащий вашу основную деятельность. Внесите в нее правки и измените ее таким образом, чтобы она ссылалась на ваше окно просмотра, названное GLView:

OpenGL/src/org/example/opengl/OpenGL.java

```
package org.example.opengl;
import android.app.Activity;
import android.os.Bundle;
public class OpenGL extends Activity {
    GLView view;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        view = new GLView(this);
        setContentView(view);
    }
    @Override
    protected void onPause() {
        super.onPause();
        view.onPause();
    }
    @Override
    protected void onResume() {
        super.onResume();
        view.onResume();
    }
}
```

Мы переопределяем методы onPause() и onResume() так, чтобы они могли вызывать методы с теми же именами в окне просмотра.

Нам не нужен макет (res/layout/main.xml), поэтому удалите его. Сейчас давайте определим класс нашего окна просмотра:

OpenGL/src/org/example/opengl/GLView.java

```
package org.example.opengl;
import android.content.Context;
import android.opengl.GLSurfaceView;
class GLView extends GLSurfaceView {
    private final GLRenderer renderer;
    GLView(Context context) {
        super(context);
        // Раскомментируйте следующую строку, чтобы включить проверку ошибок
и запись в журнал
        //setDebugFlags(DEBUG_CHECK_GL_ERROR | DEBUG_LOG_GL_CALLS);
        renderer = new GLRenderer(context);
        setRenderer(renderer);
    }
}
```

GLSurfaceView — это новый класс, представленный в Android 1.5, который значительно упрощает использование OpenGL в Android. Он обеспечивает связь для

соединения OpenGL ES с системой просмотра и жизненным циклом деятельности. Он заботится о подборе подходящего формата пиксельного фрейм-буфера и управляет различными процессами рендеринга для обеспечения плавной анимации. Все, что нужно GLView, — это расширить GLSurfaceView и определить обработчик рендеринга для окна просмотра.

В следующем разделе мы заполним пространство экрана цветом.

10.4. Рендеринг сцены

Как мы видели в разделе 4.2, «Рисование игровой доски», 2D-библиотека Android вызывает метод onDraw() окна просмотра каждый раз, когда ей нужно перерисовать область экрана. OpenGL действует несколько по-другому.

В OpenGL ES на Android рисование выделено в класс рендеринга, который отвечает за инициализацию и рисование всего экрана. Определим его сейчас.

Вот каркас для класса GLRenderer:

OpenGL/src/org/example/opengl/GLRenderer.java

```
package org.example.opengl;
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
import android.content.Context;
import android.opengl.GLSurfaceView;
import android.opengl.GLU;
import android.util.Log;
class GLRenderer implements GLSurfaceView.Renderer {
    private static final String TAG = "GLRenderer" ;
    private final Context context;
    GLRenderer(Context context) {
        this.context = context;
    }
    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
        // ...
    }
    public void onSurfaceChanged(GL10 gl, int width, int height) {
        // ...
    }
    public void onDrawFrame(GL10 gl) {
        // ...
    }
}
```

GLRenderer реализует интерфейс GLSurfaceView.Renderer, который имеет три метода. Начнем с метода onSurfaceCreated(), который вызывается, когда Surface (поверхность, аналог Canvas в обычной двумерной графике) создается или пересоздается.

OpenGL/src/org/example/opengl/GLRenderer.java

```
1     public void onSurfaceCreated(GL10 gl, EGLConfig config) {
2        // Настройка нужных нам параметров OpenGL
```

```
3         gl.glEnable(GL10.GL_DEPTH_TEST);
4         gl.glDepthFunc(GL10.GL_LEQUAL);
5         gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
6
7         // Необязательно: отключаем сглаживание для повышения производительности
8         // gl.glDisable(GL10.GL_DITHER);
9       }
```

В строке 3 мы настраиваем пару параметров OpenGL. OpenGL имеет десятки параметров, которые могут быть включены или выключены с помощью glEnable() и glDisable(). Среди наиболее часто используемых следующие:

Параметр	Описание
GL_BLEND	Смешивает входящие цветовые значения с цветами, которые уже находятся в буфере цветов
GL_CULL_FACE	Игнорирует полигоны, основываясь на их повороте (по часовой стрелке или против часовой стрелки) в координатах окна. Это малозатратный способ устранить фоновые объекты
GL_DEPTH_TEST	Производит сравнение глубин и обновляет буфер глубины. Пиксели, которые расположены дальше, чем уже нарисованные, игнорируются
GL_LIGHTi	Включает значение освещенности *i* при расчете яркости и цвета объектов
GL_LIGHTING	Включает освещение и расчет материалов
GL_LINE_SMOOTH	Рисует сглаженные линии (линии без зазубрин)
GL_MULTISAMPLE	Производит мультисэмплинг для достижения целей антиалиасинга и других эффектов
GL_POINT_SMOOTH	Рисует сглаженные точки
GL_TEXTURE_2D	Использует текстуры для рисования поверхностей

Все параметры по умолчанию выключены, за исключением GL_DITHER и GL_MULTISAMPLE. Учтите: все, что вы включаете, требует некоторых затрат вычислительной мощности.

Далее давайте заполним метод onSurfaceCharged(). Этот метод вызывается единожды после того, как Surface создан, и затем всегда, когда изменяется размер Surface:

OpenGL/src/org/example/opengl/GLRenderer.java

```
Стр. 1     public void onSurfaceChanged(GL10 gl, int width, int height) {
2             // Определяет пирамиду видимости для окна просмотра
3             gl.glViewport(0, 0, width, height);
4             gl.glMatrixMode(GL10.GL_PROJECTION);
```

```
5             gl.glLoadIdentity();
6             float ratio = (float) width / height;
7             GLU.gluPerspective(gl, 45.0f, ratio, 1, 100f);
8         }
```

Здесь мы настраиваем пирамиду просмотра нашего окна, устанавливая несколько параметров OpenGL. Заметим, что мы вызываем вспомогательную функцию GLU.gluPerspective() в строке 7. Последние два аргумента — это расстояние от глаза до ближней и дальней плоскости отсечения (см. рис. 10.1).

Пришло время что-нибудь нарисовать. Метод onDrawFrame() вызывается снова и снова в процессе рендеринга, созданном классом GLSurfaceView.

OpenGL/src/org/example/opengl/GLRenderer.java

```
public void onDrawFrame(GL10 gl) {
    // Очищаем экран, заполняя его черным цветом
    gl.glClear(GL10.GL_COLOR_BUFFER_BIT
        | GL10.GL_DEPTH_BUFFER_BIT);
    // Располагаем модель таким образом, чтобы мы могли ее видеть
    gl.glMatrixMode(GL10.GL_MODELVIEW);
    gl.glLoadIdentity();
    gl.glTranslatef(0, 0, -3.0f);
    // Другие команды рисования располагаются здесь...
}
```

Для начала мы заполняем экран черным цветом. Мы очищаем буферы цвета и глубины. Всегда помните о том, что нужно очищать и тот и другой, в противном случае вы получите весьма странные результаты, возникшие из-за того, что в буфере осталась информация о глубине от предыдущего кадра. Мы также устанавливаем начальную позицию для команд рисования, которые будут завершены в следующем разделе.

ВЕРСИЯ 1.ЧТО?

OpenGL ES 1.0 основана на полной версии OpenGL 1.3, а ES 1.1 основана на OpenGL 1.5. JSR 239 имеет две версии: исходную 1.0 и обновленную версию 1.0.1. Здесь так же присутствуют некоторые расширения OpenGL ES, в которые я не углубляюсь. Все версии Android имеют реализацию JSR 239 1.0.1 с OpenGL 1.0 и некоторые — 1.1. Для большинства программ стандарт JSR вполне подходит, поэтому именно его мы используем в этой главе.

Начиная с Android 2.2 посредством пакета android.opengl[1] поддерживается OpenGL ES 2.0. Вы также можете вызвать его из Native Development Kit (NDK)[2]. OpenGL ES 2.0 определена относительно полной спецификации OpenGL 2.0 и придает особое значение шейдерам и программируемым 3D-конвейерам. Пока нет стандарта JSR для OpenGL 2.0 ES, и программный интерфейс не имеет обратной совместимости с 1.0.

Запустите программу сейчас, и вы получите рис. 10.2. Если вы думаете, что это глупо — рисовать один и тот же черный экран снова и снова в цикле, вы правы. В этом появится больше смысла позже, когда мы поговорим об анимации, поэтому сейчас просто следуйте за мной.

[1] http://d.android.com/reference/android/opengl/GLES20.html
[2] http://d.android.com/sdk/ndk

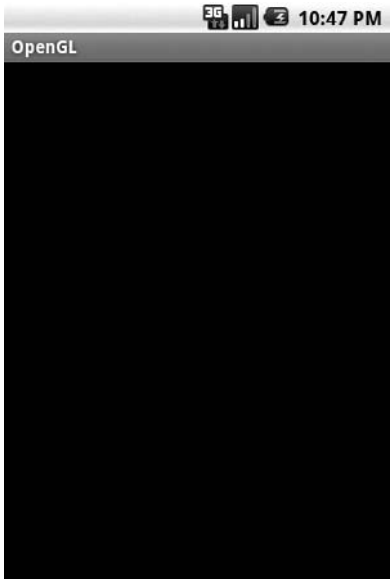

Рис. 10.2. Было проделано немало работы для получения черного экрана

Давайте двигаться дальше и нарисуем что-нибудь более интересное. Для начала нам нужно определить то, что мы будем рисовать (модель).

10.5. Построение модели

В зависимости от сложности объекта вы обычно создаете его, используя инструменты графического дизайна, и импортируете в вашу программу. Для этого примера мы просто определим простую модель в коде: это будет куб.

OpenGL/src/org/example/opengl/GLCube.java

```
1     package org.example.opengl;
-
-         import java.nio.ByteBuffer;
-         import java.nio.ByteOrder;
5         import java.nio.IntBuffer;
-
-         import javax.microedition.khronos.opengles.GL10;
-
-         import android.content.Context;
10         import android.graphics.Bitmap;
-         import android.graphics.BitmapFactory;
-         import android.opengl.GLUtils;
-
-         class GLCube {
15           private final IntBuffer mVertexBuffer;
```

```
-             public GLCube() {
-                 int one = 65536;
-                 int half = one / 2;
-                 int vertices[] = {
20                      // Передняя сторона
-                     -half, -half, half, half, -half, half,
-                     -half, half, half, half, half, half,
-                      // Задняя сторона
-                     -half, -half, -half, -half, half, -half,
25                      half, -half, -half, half, half, -half,
-                      // Левая сторона
-                     -half, -half, half, -half, half, half,
-                     -half, -half, -half, -half, half, -half,
-                      // Правая сторона
30                      half, -half, -half, half, half, -half,
-                      half, -half, half, half, half, half,
-                      // Верхняя сторона
-                     -half, half, half, half, half, half,
-                     -half, half, -half, half, half, -half,
35                      // Нижняя сторона
-                     -half, -half, half, -half, -half, -half,
-                      half, -half, half, half, -half, -half, };
-
-                 // Буфер, передаваемый в функцию gl*Pointer(), должен быть
40               // прямым, то есть он должен быть расположен на исходной куче
-                 // где сборщик мусора не сможет переместить его.
-                 //
-                 // Буферы с типами данных, состоящими из нескольких байтов
-                 // (например, short, int
-                 // float), должны иметь порядок байтов, установленный
-                 // в соответствии с исходным порядком
45               ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length *
4);
-                 vbb.order(ByteOrder.nativeOrder());
-                 mVertexBuffer = vbb.asIntBuffer();
-                 mVertexBuffer.put(vertices);
-                 mVertexBuffer.position(0);
50            }
-
-             public void draw(GL10 gl) {
-                 gl.glVertexPointer(3, GL10.GL_FIXED, 0, mVertexBuffer);
-
55               gl.glColor4f(1, 1, 1, 1);
-                 gl.glNormal3f(0, 0, 1);
-                 gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 0, 4);
-                 gl.glNormal3f(0, 0, -1);
-                 gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 4, 4);
60
-                 gl.glColor4f(1, 1, 1, 1);
-                 gl.glNormal3f(-1, 0, 0);
```

```
-                gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 8, 4);
-                gl.glNormal3f(1, 0, 0);
65               gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 12, 4);
-
-                gl.glColor4f(1, 1, 1, 1);
-                gl.glNormal3f(0, 1, 0);
-                gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 16, 4);
70               gl.glNormal3f(0, -1, 0);
-                gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 20, 4);
-            }
-        }
```

Массив vertices в строке 19 определяет углы куба в координатной модели с фиксированными точками (см. врезку «Фиксированная точка против плавающей»). Каждая сторона куба — это квадрат, который состоит из двух треугольников. Мы используем обычную модель рисования OpenGL, которая называется *triangle strips* (ленты треугольников). В этом режиме мы задаем две начальные точки, и после этого каждая следующая точка определяет треугольник с углами в предыдущих двух точках. Это быстрый способ получить много геометрических фигур с помощью графического аппаратного обеспечения.

Обратите внимание на то, что каждая точка имеет три координаты (x, y и z). Оси x и y расположены справа и сверху, соответственно, ось z выходит из экрана в направлении точки наблюдения.

ФИКСИРОВАННАЯ ТОЧКА ПРОТИВ ПЛАВАЮЩЕЙ

OpenGL ES обеспечивает интерфейсы вычислений с фиксированной точкой (integer) и с плавающей точкой. Методы для работы с фиксированной точкой оканчиваются на букву x, а с плавающей — на f. Например, можно использовать либо glColor4x(), либо glColor4f() для установки четырех компонентов цвета.

Числа с фиксированной точкой масштабируются к 2^{16}, или к 65 536. Так, 32 768 с фиксированной точкой эквивалентно 0.5f. Говоря другим языком, целая часть использует наиболее значимые два байта из четырехбайтного целого (int), где дробная часть использует менее значимые два байта. Это несколько отличается от способа, которым исходная 2D-библиотека Android использует целые числа, поэтому будьте осторожны.

В простом примере, как этот, не имеет особого значения, используете ли вы вычисления с фиксированной или с плавающей точкой, поэтому используйте их как взаимозаменяемые, так, как вам удобно. Однако помните, что некоторые устройства на Android не имеют аппаратного обеспечения для поддержки вычислений с плавающей точкой, поэтому вычисления с фиксированной точкой будут быстрее. С другой стороны, некоторые разработчики сообщают, что такая конфигурация работает гораздо медленнее, чем эмуляция вычислений с плавающей точкой. Ваш путь может изменяться.

Я советую сначала кодировать с использованием плавающей точки, так как это легче в плане программирования. Затем оптимизировать медленные части с использованием вычислений с фиксированной точкой позже, если в этом возникнет необходимость.

В методе рисования (строка 52) мы используем вертексный (вершинный) буфер, созданный конструктором, и рисуем шесть разных наборов треугольников (для шести сторон куба). В реальной программе вы захотите скомбинировать вызовы в одну или две группы треугольников, так как чем меньшее количество раз вызывается OpenGL, тем быстрее работает программа.

Сейчас давайте используем наш новый класс в GLRenderer:

OpenGL/src/org/example/opengl/GLRenderer.java

```
private final GLCube cube = new GLCube();
public void onDrawFrame(GL10 gl) {
    // ...
    // Рисуем модель
    cube.draw(gl);
}
```

Запустите программу, и вы увидите восхитительную картинку как на рис. 10.3. Во всяком случае, она гораздо привлекательнее черного экрана.

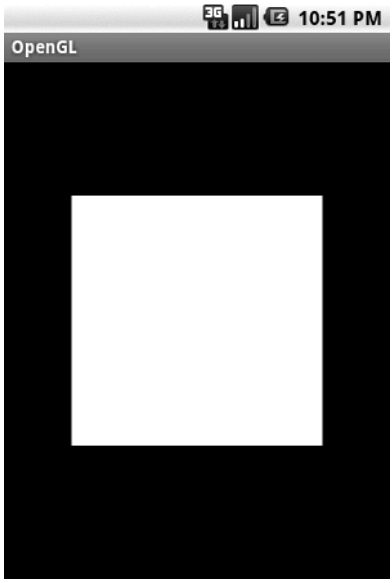

Рис. 10.3. Рисование куба без теней

10.6. Свет, камера...

В реальной жизни у вас есть источники света, такие как солнце, фары, факелы или сияющие потоки лавы. OpenGL позволяет определить до восьми источников света на сцене. Есть две части освещения — свет и то, что светится в его лучах. Начнем со света.

Все библиотеки 3D-графики поддерживают три вида освещения:

- *Ambient* (постоянный, окружающий свет): общее освещение, которое освещает всю сцену и даже объекты, которые отвернуты от света. Важно иметь хотя бы небольшой постоянный свет, для того чтобы видеть детали объектов даже тогда, когда они находятся в тени.

- *Diffuse* (диффузный, рассеянный свет): мягкое направленное освещение, такое, которое можно получить от флуоресцентных ламп. Основное освещение сцены обычно исходит от диффузных источников.
- *Specular* (отраженный, точечный свет): яркий свет, обычно от ярких точечных источников. В комбинации с блестящими материалами он дает отражения (блеск), который добавляет реализма.

Один источник света может обеспечивать все три вида освещения. Эти значения входят в вычисления освещенности, которые определяют цвет и яркость каждого пикселя на экране.

Источники света определены в методе GLRenderer.onSurfaceCreated():

OpenGL/src/org/example/opengl/GLRenderer.java

```
float lightAmbient[] = new float[] { 0.2f, 0.2f, 0.2f, 1 };
float lightDiffuse[] = new float[] { 1, 1, 1, 1 };
float[] lightPos = new float[] { 1, 1, 1, 1 };
gl.glEnable(GL10.GL_LIGHTING);
gl.glEnable(GL10.GL_LIGHT0);
gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_AMBIENT, lightAmbient, 0);
gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_DIFFUSE, lightDiffuse, 0);
gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_POSITION, lightPos, 0);
```

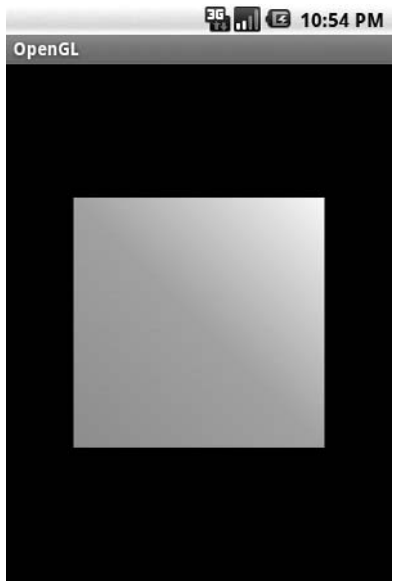

Рис. 10.4. Освещение сцены

В коде мы объявили один источник света в позиции (1, 1, 1). Это белый однонаправленный свет, который имеет светлый диффузный компонент и слабый компонент окружающего света. В этом примере мы не используем точечный свет.

Далее нам нужно сообщить OpenGL о материале, из которого сделан наш куб. Свет по-разному отражается от различных материалов, таких как металл, пластик или бумага. Для симуляции этого эффекта в OpenGL добавьте этот код в метод onSurfaceCreated(), чтобы определить, как материал взаимодействует с тремя типами света: окружающим, диффузным и точечным:

OpenGL/src/org/example/opengl/GLRenderer.java

```
float matAmbient[] = new float[] { 1, 1, 1, 1 };
float matDiffuse[] = new float[] { 1, 1, 1, 1 };
gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_AMBIENT,
    matAmbient, 0);
gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_DIFFUSE,
    matDiffuse, 0);
```

Объект появится и будет иметь матовую поверхность, как если бы он был сделан из бумаги (рис. 10.4). Верхний правый угол куба расположен ближе всего к источнику света и поэтому выглядит светлее.

10.7. Мотор!

До этого момента куб просто висел на экране без движения. Это слишком скучно, поэтому давайте его подвигаем. Для того чтобы сделать это, нам нужна пара изменений в наших методах onSurfaceCreated() и onDrawFrame() в GLRenderer.

OpenGL/src/org/example/opengl/GLRenderer.java

```
private long startTime;
private long fpsStartTime;
private long numFrames;
public void onSurfaceCreated(GL10 gl, EGLConfig config) {
    // ...
    startTime = System.currentTimeMillis();
    fpsStartTime = startTime;
    numFrames = 0;
}
public void onDrawFrame(GL10 gl) {
    // ...
    // Задаем угол вращения как функцию времени
    long elapsed = System.currentTimeMillis() - startTime;
    gl.glRotatef(elapsed * (30f / 1000f), 0, 1, 0);
    gl.glRotatef(elapsed * (15f / 1000f), 1, 0, 0);
    // Рисуем модель
    cube.draw(gl);
}
```

Этот код немного поворачивает куб каждый раз, проходя главный цикл. В частности, каждую секунду он поворачивается на 30° вокруг оси x и на 15°вокруг оси y. В результате мы получаем замечательный, плавно вращающийся куб (рис. 10.5).

10.8. Применение текстур

Хотя наша сцена выглядит гораздо интереснее, никто не спутает ее с реальной жизнью. Объекты, с которыми мы постоянно сталкиваемся в реальности, имеют текстуры вроде грубой поверхности кирпичной стены или гравийной садовой дорожки. У вас есть ламинированный стол? Ламинат, выглядящий как дерево, — это лишь фотография текстуры дерева, которая приклеена на поверхность из менее дорогого материала вроде пластика или древесно-стружечной плиты.

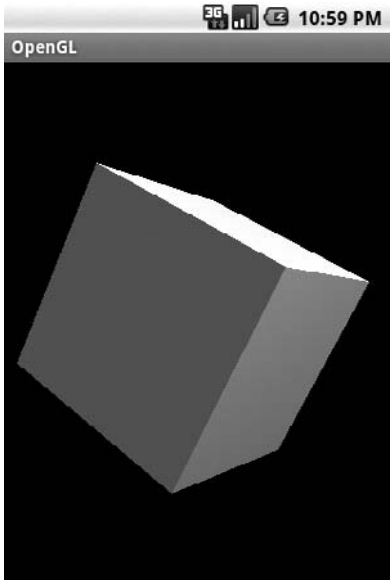

Рис. 10.5. Вращение куба

АНИМАЦИЯ, ОСНОВАННАЯ НА ВРЕМЕНИ

Первая версия этого примера сохраняла направление текущего угла вращения и просто увеличивала его в каждой итерации цикла. Вы можете придумать причину, почему это была плохая идея?

С тех пор как Android стал работать на различных устройствах, вы не можете заранее знать, сколько времени займет рисование отдельного кадра. Это может занять полсекунды или 1/100 секунды. Если вы будете перемещать объект на фиксированное расстояние в каждом кадре, тогда, на медленных устройствах, он будет перемещаться слишком медленно, а на быстрых — слишком быстро. Привязывая расстояние перемещения к тому, сколько времени прошло, вы достигнете предсказуемых результатов на любом устройстве. Более быстрое аппаратное обеспечение выведет более плавную анимацию, но объект попадет из точки A в точку B за то же самое время.

Мы собираемся сделать то же самое с нашим кубом, используя картинку. К несчастью, код, который это делает, довольно длинный. Не беспокойтесь, если вы его не вполне понимаете.

OpenGL/src/org/example/opengl/GLCube.java

```
private final IntBuffer mTextureBuffer;
public GLCube() {
    int texCoords[] = {
        // Передняя сторона
        0, one, one, one, 0, 0, one, 0,
        // Задняя сторона
        one, one, one, 0, 0, one, 0, 0,
        // Левая сторона
        one, one, one, 0, 0, one, 0, 0,
        // Правая сторона
        one, one, one, 0, 0, one, 0, 0,
        // Верхняя сторона
        one, 0, 0, 0, one, one, 0, one,
        // Нижняя сторона
        0, 0, 0, one, one, 0, one, one, };
    // ...
    ByteBuffer tbb = ByteBuffer.allocateDirect(texCoords.length * 4);
    tbb.order(ByteOrder.nativeOrder());
    mTextureBuffer = tbb.asIntBuffer();
    mTextureBuffer.put(texCoords);
    mTextureBuffer.position(0);
    }
    static void loadTexture(GL10 gl, Context context, int resource) {
        Bitmap bmp = BitmapFactory.decodeResource(
            context.getResources(), resource);
        GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, bmp, 0);
        gl.glTexParameterx(GL10.GL_TEXTURE_2D,
            GL10.GL_TEXTURE_MIN_FILTER, GL10.GL_LINEAR);
        gl.glTexParameterx(GL10.GL_TEXTURE_2D,
            GL10.GL_TEXTURE_MAG_FILTER, GL10.GL_LINEAR);
        bmp.recycle();
    }
}
```

Сейчас нам нужно сообщить OpenGL об использовании координат текстур. Добавьте это в начало метода draw():

OpenGL/src/org/example/opengl/GLCube.java

```
gl.glEnable(GL10.GL_TEXTURE_2D); // обход ошибки 3623
gl.glTexCoordPointer(2, GL10.GL_FIXED, 0, mTextureBuffer);
```

И наконец, нам нужно вызывать метод loadTexture() в GLRenderer. Добавьте эти строки в конец метода onSurfaceCreated():

OpenGL/src/org/example/opengl/GLRenderer.java

```
// Включаем использование текстур
gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
gl.glEnable(GL10.GL_TEXTURE_2D);
// Загружаем текстуру куба из растрового изображения
GLCube.loadTexture(gl, context, R.drawable.android);
```

Этот код включает использование текстур и координат текстур и затем вызывает наш метод loadTexture(), передавая ему контекст Activity и ID ресурса так, что он сможет загрузить изображение текстуры.

R.drawable.android — это PNG-файл 128×128 пикселей, который я скопировал в res/drawable-nodpi/android.png. Вы найдете его в дополнительных материалах к книге. Обратите внимание на то, что число 128 нигде не появляется, поэтому вы можете легко заменить изображение на большее или меньшее.

Вы можете видеть результаты нашей работы на рис. 10.6.

Рис. 10.6. Применение текстуры

10.9. Ку-ку

Просто для развлечения давайте сделаем куб полупрозрачным. Добавьте это в GLRenderer.onSurfaceCreated():

OpenGL/src/org/example/opengl/GLRenderer.java

```
boolean SEE_THRU = true;
// ...
if (SEE_THRU) {
    gl.glDisable(GL10.GL_DEPTH_TEST);
    gl.glEnable(GL10.GL_BLEND);
    gl.glBlendFunc(GL10.GL_SRC_ALPHA, GL10.GL_ONE);
}
```

Здесь выключена проверка глубины, так как мы хотим видеть и скрытые части объекта, так же как и те, которые находятся на переднем плане. Здесь тоже включен режим смешивания, который позволяет основывать прозрачность объектов на их альфа-канале (канале прозрачности). В результате задние грани куба можно видеть через передние (рис. 10.7).

Рис. 10.7. Последняя версия: полупрозрачный куб

Я оставляю вам для самостоятельной работы возможности по включению и выключению прозрачности. Поэкспериментируйте с различными режимами смешивания для получения интересных эффектов.

10.10. Измерение плавности

Насколько плавной выглядит плавная анимация? Плавность игры или другой программы, требовательной к графике, определяется тем, насколько быстро она может обновлять состояние экрана. Обычно ее измеряют, подсчитывая количество отображаемых экранов или кадров в секунду (FPS — frames per second). Каждый человек имеет свое представление о «плавности». 15–30 FPS иногда приемлемо для тех, кто играет от случая к случаю, но более серьезные игроки ожидают более высоких показателей, — таких как 60 FPS и выше. Я рекомендую приложить все усилия для достижения устойчивого уровня в 60 FPS. Это соответствует максимальной скорости обновления экрана для большинства LCD-панелей, а кроме того,

это скорость, используемая популярными игровыми платформами, такими как Sony PlayStation и PlayStation Portable (PSP).

Обратите внимание: это не всегда возможно, так как некоторые телефоны на Android имеют ограничение на скорость вывода кадров, что подразумевает, что их аппаратное обеспечение поддержки 3D-графики менее мощное в сравнении с разрешением экрана. В зависимости от того, что вы рисуете, они могут быть просто не в состоянии вывести пиксели на экран достаточно быстро для достижения 60 FPS. Первое поколение телефонов с дисплеями 800×480+, таких как Nexus One и Droid Sholes, страдали от этой проблемы. Но выпускаются и быстрые устройства, которые такой проблемы не имеют.

Высокие показатели вывода кадров — это непростая задача, так как на 60 FPS у вас есть лишь 1/60 секунды (16,67 миллисекунды) между вызовами onDrawFrame() для того, чтобы успеть сделать все, что нужно, включая любую анимацию, физические и игровые расчеты, плюс время, которое займет рисование сцены для текущего кадра. Единственный способ узнать, достигли ли вы целевого FPS, — измерить его.

Для того чтобы это сделать, попробуйте добавить этот кусок кода в конец метода onDrawFrame():

OpenGL/src/org/example/opengl/GLRenderer.java

```
numFrames++;
long fpsElapsed = System.currentTimeMillis() - fpsStartTime;
if (fpsElapsed > 5 * 1000) { // every 5 seconds
    float fps = (numFrames * 1000.0F) / fpsElapsed;
    Log.d(TAG, "Frames per second: " + fps + " (" + numFrames
        + " frames in " + fpsElapsed + " ms)" );
    fpsStartTime = System.currentTimeMillis();
    numFrames = 0;
}
```

Каждые пять секунд этот код выводит средний FPS в системный журнал Android (см. раздел 3.10, «Отладка с помощью записи сообщений в журнал»). Если эти показатели падают ниже ваших целевых значений, тогда поработайте над своим алгоритмом и попробуйте снова. Продолжайте экспериментировать до тех пор, пока не достигнете цели. Программа для сбора информации, такая как traceview[1], также может оказаться полезной. Вы должны избегать попыток отображать эти сведения на экране поверх других графических объектов, так как эти действия могут уменьшить показатели.

Если вы испытаете это сейчас, то заметите, что эмулятор работает гораздо медленнее, чем реальное устройство. В моих тестах я видел примерно 12 FPS на эмуляторе и значение, близкое к 60 FPS, на реальном телефоне. Отсюда можно извлечь следующий урок: не доверяйте в тестах производительности эмулятора.

[1] http://d.android.com/guide/developing/tools/traceview.html

10.11. Вперед>>

В этой главе вы узнали, как пользоваться библиотекой Android для работы с трехмерной графикой. Поскольку Android использует библиотеку трехмерной графики, основанную на стандарте OpenGL ES API и признанную индустриальным стандартом, если вы захотите узнать об этой технологии больше, вы найдете огромное количество информации. Я особенно порекомендовал бы Javadoc спецификации[1] стандарта JSR 239 API. Другие советы по работе с графикой ищите в конференции[2] Google I/O.

[1] http://java.sun.com/javame/reference/apis/jsr239

[2] http://code.google.com/events/io/2009/sessions/WritingRealTimeGamesAndroid.html и http://code.google.com/events/io/2010/sessions/writing-real-time-games-android.html

IV

Следующее поколение

11 Мульти-тач

Новые возможности добавляются в платформу с каждой новой версией Android. В этой части мы сконцентрируемся на таких новых возможностях и на подготовке ваших программ таким образом, чтобы они стали доступны другим пользователям.

В этой главе мы узнаем, как использовать новую возможность мульти-тач в Android 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. Ранее[1] мы использовали жесты, выполняемые одним пальцем, хотя мы их так не называли. Помните, в подразделе 4.3, «Ввод чисел», мы предоставили пользователю возможность прикасаться 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.

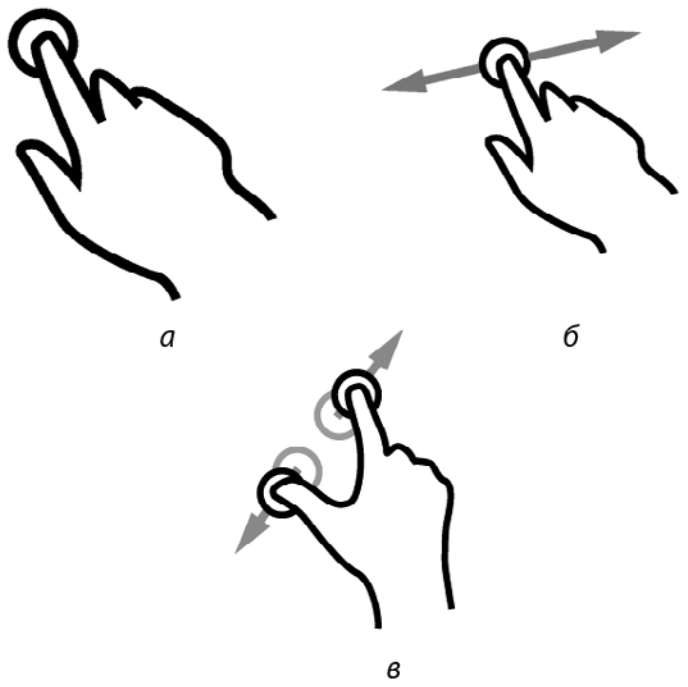

Рис. 11.1. Три обычных жеста: *а* — касание, *б* — перетаскивание, *в* — изменение масштаба двумя пальцами

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, «Подготовка»). Но благодаря поддержке мульти-тач теперь изменение масштаба производится 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 2.2 или выше — обратитесь к онлайновой документации[1] за подробностями.

При попытке запустить пример из этой главы на Android 1.5 или 1.6 он завершится 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

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 неработоспособностью. API постоянно сообщает неверные или невозможные данные о точках касания, особенно 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 координатные оси X и Y, несколько пальцев иногда распознаются как один. Некоторые из этих проблем могут относиться к ограничениям аппаратного обеспечения сенсоров касания тач-скринов, использованных 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 собираемся создать простой просмотрщик изображений, который позволяет изменять масштаб и прокручивать изображение. Готовый проект показан на рис. 11.2.

Начнем с создания проекта «Hello, Android» со следующими параметрами в диалоговом 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 Touch.java, содержащий вашу основную 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" ;
```

```
    @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

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, который занимает весь экран. Значение android:src=»@drawable/butterfly» ссылается на изображение бабочки, использованное в примере. Используйте любой файл формата 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 поговорим 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, указывает Android на использование полного экрана без заголовка окна или строки состояния в верхней части. Если вы запустите приложение сейчас, оно просто покажет изображение.

11.3. Изучение событий касания

Когда я впервые изучаю новое API, то часто использую код, который поставляет отладочную информацию обо всем, что происходит; в итоге я получаю представление о том, что делает метод, и в каком порядке происходят события. С этого 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 единственный 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»).

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 непонятно, что он пытался сделать — касание или перетаскивание.
- Далее, начиная со строки 2, поступили несколько событий ACTION_MOVE, показывающие, что пользователь немного перемещал палец в координатах, о которых 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 пользователь 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 двумя[1]. Координаты обоих пальцев возвращаются как часть события. Это выглядит похожим на то, что пользователь начал жест изменения 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 работающими за ним людьми, вы легко получите больше касаний экрана, чем пальцев на руке.

- 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 палец 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, показывающих, 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 заключается 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 *матричными преобразованиями*. Используя матрицу, мы можем представить любой вид преобразования, вращения или наклона, который хотим произвести с изображением. 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 трансформацией). 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.

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:
```

```
    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 преобразования 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, используя 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;
```

```
            matrix.postScale(scale, scale, mid.x, mid.y);
        }
}
break;
```

Когда мы видим событие опускания на экран второго пальца, мы рассчитываем 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 матрицу 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 рассчитываем расстояние между пальцами снова. Если оно слишком мало, событие игнорируется; в противном случае мы восстанавливаем матрицу преобразования и изменяем масштаб изображения вокруг центральной точки.

Масштаб — это просто количественное отображение нового расстояния, деленное 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 промежутка[1].

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 выполняются 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

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

Попытайтесь запустить программу на мобильном телефоне. Перетащите изображение 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 API для создания жеста изменения масштаба при помощи двух пальцев. Есть отличный сайт, называемый GestureWorks[1], 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 «Force close» (Принудительное закрытие). К счастью, есть способ обойти подобные ограничения, как описано в разделе 13.3 «Разворачивание программы 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.

[1] http://gestureworks.com

12 Нет места лучше дома

Неважно, как глубоко вы погрузились в игру или другую Android-программу, вы всегда можете нажать кнопку Home и увидеть хорошо знакомый домашний экран Android. Это центральное место, откуда можно воспользоваться услугами веб-браузера, сделать телефонный звонок, открыть электронное письмо, запустить любое приложения и заняться другими развлечениями с Android.

Поскольку вы собираетесь проводить здесь немало времени, Android позволяет настроить домашний экран в соответствии со своими предпочтениями множеством различных способов: например, установить статические фоновое изображение или изменить положения значков. В этой главе вы узнаете еще пару способов настройки: виджеты и интерактивные обои. Сейчас щелкните каблуками ваших башмачков и давайте начнем.

12.1. Привет, виджет

Представленные в Android 1.5 (Cupcake) виджеты — это миниатюрные окна приложений, которые можно встроить в домашний экран. Несколько виджетов предоставляет сам Android — это аналоговые часы, пульт управления проигрыванием музыки и еще один, показывающий картинки. Множество разработчиков создали интересные виджеты для отображения информации о погоде, вывода заголовков новостей, гороскопов и многого другого. Вы тоже сможете их создавать. Данный раздел покажет, как это сделать.

Создание вашего первого виджета

Для этого примера мы собираемся создать виджет, который показывает текущую дату. Конечный результат показан на рис. 12.5.

К несчастью, не существует особого средства Eclipse для создания виджетов, поэтому мы для начала создадим обычное приложение «Hello, Android», как мы это уже делали в подразделе 1.2 «Создание первой программы», и затем его настроим. Выберите команду меню File ▸ New ▸ Project..., чтобы открыть диалоговое окно New Project. Теперь выберите Android ▸ Android Project и нажмите кнопку Next. Введите следующую информацию:

```
Project name: Widget
Build Target: Android 2.2
Application name: Widget
Package name: org.example.widget
Min SDK Version: 8
```

Вместо того чтобы вводить имя в поле Activity, мы оставим это поле пустым и снимем флажок напротив пункта Create Activity. После завершения этих действий вы получите картинку как на рис. 12.1.

Щелкните Finish. Плагин Android создаст проект и наполнит его некоторыми файлами по умолчанию. Эти файлы не подходят для проекта виджета, поэтому давайте их отредактируем.

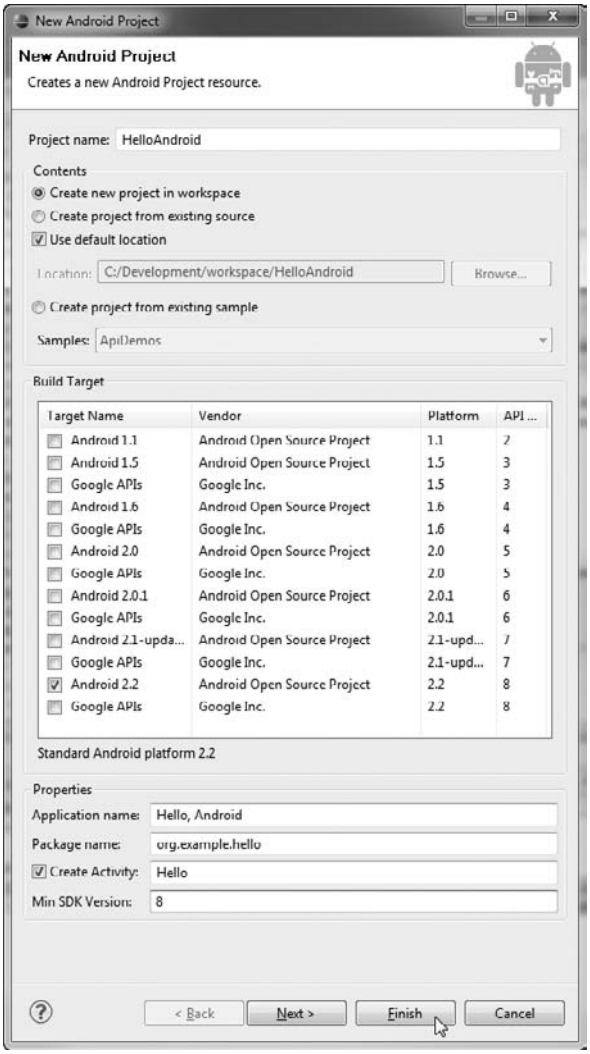

Рис. 12.1. Новый проект виджета Android

Вызываем все виджеты!

Нашей первой остановкой будет AndroidManifest.xml. Хотя технически возможно (на практике так часто и делают) поместить виджеты и деятельности в одно и то же приложение, в этом примере будет помещен только виджет. Нам не нужен тег <activity> для определения виджета. Вот отредактированный файл манифеста:

Widget/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.example.widget"
    android:versionCode="1"
    android:versionName="1.0" >
    <application android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <!—Широковещательный приемник, который обработает обновления AppWidget
-->
        <receiver android:name=".Widget"
        android:label="@string/widget_name" >
        <intent-filter>
            <action android:name=
                "android.appwidget.action.APPWIDGET_UPDATE" />
        </intent-filter>
    <meta-data android:name="android.appwidget.provider"
            android:resource="@xml/widget" />
        </receiver>
    </application>
    <uses-sdk android:minSdkVersion="3" android:targetSdkVersion="8" />
</manifest>
```

Тег <meta-data> сообщает Android о том, что он может найти определение виджета в res/xml/widget.xml.

Вот его определение:

Widget/res/xml/widget.xml

```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider
        xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="146dip"
    android:minHeight="72dip"
    android:updatePeriodMillis="1800000"
    android:initialLayout="@layout/main"
    />
```

Этот код определяет минимальный размер виджета, как часто его следует обновлять (подробности об этом позже) и ссылку на его стартовый макет.

Макет определяется в res/layout/main.xml:

Widget/res/layout/main.xml

```
<?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"
    android:background="@drawable/widget_bg"
    >
<TextView android:id="@+id/text"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="@string/hello"
    android:textSize="18sp"
    android:gravity="center"
    android:textColor="@android:color/black"
    />
</LinearLayout>
```

Обратите внимание, что этот макет почти такой же, какой мы использовали для примера «Hello, Android», за исключением того, что эта версия задает текст черного цвета, выровненный по центру, и фоновое изображение.

Растягиваем по размеру

Для фонового изображения нам следует использовать любой объект Android Drawable, который имеет сплошной цвет или представляет собой растровое изображение (см. подраздел 4.1 «Основы»). Мы даже могли бы оставить его выключенным для того, чтобы получить полностью прозрачный фон. Но для этого примера я хотел показать, как использовать изображение NinePath.

Изображение NinePath — это растягиваемое PNG-изображение, которое часто используют для фона кнопок с изменяемым размером. Вы используете инструмент Draw 9-path[1], включенный в SDK, для создания таких изображений, как показано на рис. 12.2.

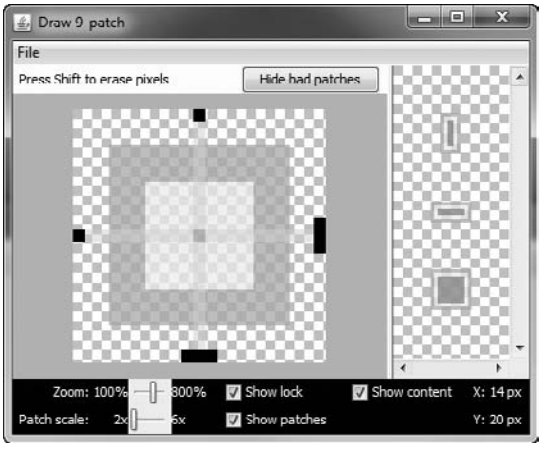

Рис. 12.2. Определение растягиваемого фона с помощью инструмента Draw 9-path

[1] http://d.android.com/guide/developing/tools/draw9patch.html

Однопиксельная граница вокруг реального изображения содержит дополнительную информацию о том, как растягивать изображение и вставлять дополнительный контент. Линии на нижнем и правом краях сообщают Android, в каком направлении следует перемещаться контенту. Если содержимое не подходит для заданной области (что обычно и случается), тогда линии слева и сверху сообщают Android, какие строки и столбцы пикселей следует скопировать, чтобы растянуть изображение.

Для готового файла следует использовать расширение .9.png, и мы должны разместить его в папке проекта res/drawable.

Рис. 12.3. Настройка вашего домашнего экрана с помощью виджетов

Следующее, что нам нужно, — это класс Widget.

Охватить и растянуть

Widget нуждается в расширении класса AppWidgetProvider, предоставляемого Android. Используя этот класс, мы без затрат получаем множество встроенных функциональных возможностей, что всегда хорошо. Вот определение класса Widget:

Widget/src/org/example/widget/Widget.java

```
package org.example.widget;
import android.appwidget.AppWidgetProvider;
public class Widget extends AppWidgetProvider {
    // ...
}
```

Мы вернемся к заполнению этого класса очень скоро, но сейчас мы можем просто использовать стандартное поведение, реализуемое AppWidgetProvider.

Наконец, давайте избавимся от нескольких надоедливых сообщений об ошибках, определив следующие строковые значения:

Widget/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World!</string>
    <string name="app_name">Widget</string>
    <string name="widget_name">Widget</string>
</resources>
```

Все, что нам теперь осталось, — запустить программу.

Запуск виджета

Чтобы запустить ваш новый виджет, перейдите в окно Package Explorer, щелкните правой кнопкой по проекту Widget и выберите команду Run As ▸ Android Application. К сожалению, вы не увидите никаких сообщений, которые бы могли как-то отобразить, как Eclipse компилирует и устанавливает виджет на ваш эмулятор или на устройство.

Чтобы увидеть новый виджет, откройте контекстное меню домашнего экрана: нажмите и удерживайте палец (или мышь) на домашнем экране. Появится меню с перечислением всех видов объектов, которые вы можете добавить (рис. 12.3).

Выберите Widgets из меню, затем выберите виджет, названный Widget (оригинально, не правда ли?). В итоге ваш виджет должен появиться на экране (рис. 12.4).

Рис. 12.4. Привет, Виджет

Попытайтесь перемещать его по экрану, нажимая и удерживая палец на виджете. Поверните дисплей, чтобы увидеть, как виджет автоматически изменил размер. Для удаления виджета переместите его на значок корзины в верхней части экрана.

Он хорош, не правда ли? Но давайте сделаем его еще лучше, отобразив текущую дату и время в нашем виджете.

Обновление виджета

В Android 1.5 виджет-хост (программа вроде домашнего экрана, которая может содержать виджеты) отправляет сообщения всем своим виджет-детям, когда виджеты должны что-то отобразить. Android использует для передачи сообщений широковещательные намерения. В этом случае намерение выглядит как android.appwidget.action.APPWIDGET_UPDATE.

Ранее, когда мы настраивали файл AndroidManifest.xml, мы сообщили Android, что можем принимать намерения и делать с ними что-то интересное. Сейчас пришло время заполнить класс Widget для того, чтобы это сделать:

Widget/src/org/example/widget/Widget.java

```
1     package org.example.widget;
-
-         import java.text.SimpleDateFormat;
-         import java.util.Date;
5
-         import android.appwidget.AppWidgetManager;
-         import android.appwidget.AppWidgetProvider;
-         import android.content.Context;
-         import android.widget.RemoteViews;
10
-         public class Widget extends AppWidgetProvider {
-             // ...
-             // Определяет формат строки для даты
-             private SimpleDateFormat formatter = new SimpleDateFormat(
15                 "EEEEEEEEE\nd MMM yyyy" );
-
-             @Override
-             public void onUpdate(Context context,
-                 AppWidgetManager appWidgetManager, int[] appWidgetIds) {
20                  // Получаем и форматируем текущую дату
-                 String now = formatter.format(new Date());
-
-                 // Изменяем текст в виджете
-                 RemoteViews updateViews = new RemoteViews(
25                      context.getPackageName(), R.layout.main);
-                 updateViews.setTextViewText(R.id.text, now);
-                 appWidgetManager.updateAppWidget(appWidgetIds, updateViews);
-
-                 // Не обязательно, на самом деле, просто привычка
```

```
30                  super.onUpdate(context, appWidgetManager, appWidgetIds);
-               }
-           }
```

Когда бы ни поступило намерение APPWIDGET_UPDATE, Android вызывает наш метод onUpdate(). В строке 21 мы форматируем текущую дату, используя SimpleDateFormat, созданный в строке 14. Это позволяет отобразить день недели, день месяца, название месяца и год в первой строке виджета и час, минуту, секунду и миллисекунду во второй строке.

Далее в строке 24 мы создаем экземпляр RemoteWiews для нашего нового макета просмотра, который отобразит виджет. Так случилось, что этот пример, когда обновляется виджет, использует тот же макет R.layout.main, что и при его старте. Строка 26 заменяет исходный текст «Hello, World» текущей датой и временем.

Наконец, в строке 27 мы отправляем наш обновленный вьювер для замены текущего содержимого виджета. Вызов super.onUpdate() в строке 30 — лишь для чистоты кода.

Удалите виджет с домашнего экрана и переустановите его из Eclipse. Когда вы вернете его обратно на домашний экран, перед вами будет что-то вроде рис. 12.5.

Рис. 12.5. Отображение даты и времени

Частота обновлений регулируется параметром android:updatePeriodMillis= в res/xml/widget.xml. Установим значение 1 800 000 миллисекунд — это 30 минут, то есть наш виджет будет обновляться дважды в час. Заметьте, что это лишь приблизительное число. Реальные события обновления могут быть отложены (возможно, на длительное время), так как на телефоне могут происходить другие события.

Google рекомендует обновлять виджеты нечасто, например раз в день или раз в час, чтобы сохранять заряд батарей. Начиная с Android 1.6 параметр

android:updatePeriodMillis= не принимает значения меньше получаса. Если требуются более частые обновления, установите ваш собственный таймер, используя класс AlarmManager. Это будет упражнением для читателя.

Уходим в отрыв

Сейчас, после знакомства с основами создания виджета, несть числа интересным штуковинам, которые вы можете создать. Если вам нужны более широкие возможности, такие как реакции на события, фоновые процессы и задание исходной деятельности для конфигурации, обратитесь к онлайновой документации[1]. Но постарайтесь делать ваши виджеты простыми и полезными. Разумно используя виджеты, вы сделаете работу с Android для ваших пользователей более персонализированной и динамичной.

Королевство виджетов для вас маловато, развернуться негде? Тогда следуйте по дороге, вымощенной желтым кирпичом в страну интерактивных обоев.

12.2. Интерактивные обои

Обычные обои просто отображаются. Они хорошо выглядят, но никогда не меняются. Скука.

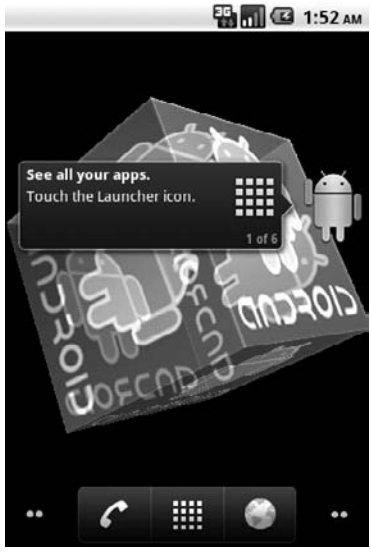

Рис. 12.6. Пример с обоями повторно использует код из главы про OpenGL

Интерактивные обои — это новая возможность Android 2.1 (Éclair Maintenance Release 1). Они позволяют заменить скучные статические изображения обоев всем

[1] http://d.android.com/guide/topics/appwidgets

чем угодно — от живой пульсирующей визуализации музыкальной композиции до тихого медитативного пруда, который отвечает на прикосновения спокойными волнами.

Отображение текущей погоды, слайд-шоу, игра Magic 8 Balls и пиротехнические эффекты — лишь несколько возможных направлений. Давайте отодвинем занавес для того, чтобы увидеть, как делается волшебство.

Создание проекта Wallpaper

В этом примере мы создадим интерактивные обои, которые отображают вращающийся куб, используя OpenGL. Финальный результат показан на рис. 12.6. Начнем с создания нового проекта Android, используя эти значения в мастере проекта:

```
Project name: Wallpaper
Build Target: Android 2.2
Application name: Wallpaper
Package name: org.example.wallpaper
Min SDK Version: 8
```

Как и в случае с проектом виджета, оставим поле с именем деятельности пустым и снимем флажок около параметра Create Activity (Создавать деятельность). После того как проект создан, нам нужно выполнить некоторые небольшие изменения в файле AndroidManifest.xml. Вот на что он должен быть похож:

Wallpaper/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.example.wallpaper"
    android:versionCode="1"
    android:versionName="1.0" >
    <application android:label="@string/app_name">
        <service android:name=".Wallpaper"
            android:label="@string/service_name"
            android:permission="android.permission.BIND_WALLPAPER" >
        <intent-filter>
            <action android:name=
                "android.service.wallpaper.WallpaperService" />
            </intent-filter>
            <meta-data android:name="android.service.wallpaper"
                android:resource="@xml/wallpaper" />
        </service>
    </application>
    <uses-sdk android:minSdkVersion="7" android:targetSdkVersion="8" />
</manifest>
```

Здесь есть новый тег <service>. Он задает сервис Android, который будет выполняться в фоновом режиме и реагировать на события. Атрибут android:permission= означает, что любая программа, которая вызывает наш сервис, нуждается в заданном разрешении. Программа Android Home screen (домашний экран) уже имеет набор нужных разрешений, поэтому она нормально работает.

Тег <intent-filter> сообщает Android, какой это тип сервиса, и тег <meta-data> дает ему знать о том, где искать дополнительную информацию об обоях. Установка android:resource=»@xml/wallpaper» ссылается на файл res/xml/wallpaper.xml — новый файл, который вам следует создать сейчас, используя следующий код:

Wallpaper/res/xml/wallpaper.xml

```
<?xml version="1.0" encoding="utf-8"?>
<wallpaper xmlns:android="http://schemas.android.com/apk/res/android"
    android:author="@+string/author"
    android:description="@string/description"
    android:thumbnail="@drawable/thumbnail" />
```

Метаданные обоев задают автора обоев (это вы), короткое описание того, как они работают, и картинку значка. Изображение и описание появятся в списке, когда пользователь захочет выбрать обои для отображения.

Перед тем как мы пойдем дальше, давайте определим строки, которые нам нужны для проекта в файле res/values/strings.xml:

Wallpaper/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Wallpaper</string>
    <string name="service_name">Hello, Android!</string>
    <string name="author">Hello, Android!</string>
    <string name="description">Sample live wallpaper
        from Hello, Android!</string>
</resources>
```

Удалите файл макета, res/layout/main.xml, так как мы не будем его использовать. Мы наполним содержимым класс Wallpaper (Wallpaper.java) после того, как рассмотрим некоторые подробности о сервисах Android.

Введение в сервисы

Одна из характерных особенностей Android — это возможность запускать программы в фоновом режиме. Для того чтобы отличать их от деятельностей, видимых пользователю, Android называет эти программы *сервисами* (services).

Главный класс Java для сервисов — потомок класса Service. Сервисы имеют жизненный цикл, похожий на деятельности (см. раздел 2.2 «Оно живое!»), но он немного проще. Здесь есть метод onCreate(), который вызывается при первом создании сервиса, и метод onDestroy(), который вызывается при уничтожении сервиса.

Между ними Android вызывает метод onStartCommand() (onStart() до версии 2.0), где клиент запрашивает старт сервиса. Сервисы могут быть *прикрепляемыми* и *неприкрепляемыми*, в зависимости от того, хотите ли вы, чтобы сервис оставался в памяти между вызовами.

Существует несколько других методов, которые вы при желании можете реализовать, например, при нехватке памяти. Смотрите онлайновую документацию для того, чтобы узнать все чудовищные подробности[1].

В примере с обоями нам не нужно беспокоиться об этих методах, так как они все обрабатываются классом WallpaperService, который является подклассом Service.

Наш основной класс нуждается в расширении WallpaperService, как здесь:

Wallpaper/src/org/example/wallpaper/Wallpaper.java

```
package org.example.wallpaper;
import android.service.wallpaper.WallpaperService;
public class Wallpaper extends WallpaperService {
    private class MyEngine extends Engine {
        // Здесь будет реализация системы...
    }
    @Override
    public Engine onCreateEngine() {
        return new MyEngine();
    }
}
```

Все, что нам остается сделать, — реализовать метод onCreateEngine(), который имеет одну строку. Его единственная цель — создать и возвратить другой класс, который называется MyEngine.

Создание механизма отрисовки

Класс MyEngine — это внутренний класс Wallpaper, поэтому в Java он объявляется внутри фигурных скобок класса. MyEngine расширяет класс Engine, предоставленный Android. Вот набросок класса MyEngine с методами-заглушками:

Wallpaper/src/org/example/wallpaper/Wallpaper.java

```
private class MyEngine extends Engine {
    @Override
    public void onCreate(final SurfaceHolder holder) {
        super.onCreate(holder);
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
    };
    @Override
    public void onSurfaceCreated(final SurfaceHolder holder) {
        super.onSurfaceCreated(holder);
    }
```

[1] http://d.android.com/reference/android/app/Service.html

```
    @Override
    public void onSurfaceDestroyed(final SurfaceHolder holder) {
        super.onSurfaceDestroyed(holder);
    }
    @Override
    public void onSurfaceChanged(final SurfaceHolder holder,
        final int format, final int width, final int height) {
        super.onSurfaceChanged(holder, format, width, height);
    }
    @Override
    public void onVisibilityChanged(final boolean visible) {
        super.onVisibilityChanged(visible);
    }
    @Override
    public void onOffsetsChanged(final float xOffset,
        final float yOffset, final float xOffsetStep,
        final float yOffsetStep, final int xPixelOffset,
        final int yPixelOffset) {
        super.onOffsetsChanged(xOffset, yOffset, xOffsetStep,
            yOffsetStep, xPixelOffset, yPixelOffset);
    }
}
```

Обратите внимание на то, что каждый метод должен всегда вызывать метод своего надкласса. Используйте Eclipse, чтобы создавать эти заглушки в редакторе Java, выделяя имя класса MyEngine, щелкая правой кнопкой мыши по Source ▸ Override/Implement Methods и выбирая методы, которые вы хотите создать. Тем не менее есть одно различие между тем, что создает Eclipse, и тем, что показано ранее. Я добавил ключевое слово *final* к каждому параметру метода. Это нам понадобиться позднее для того, чтобы внутренние классы внутри этих методов могли получать доступ к параметрам. Если вы забудете и пропустите их, компилятор даст об этом знать.

В течение жизненного цикла механизма отрисовки Android вызывает эти методы в особом порядке. Вот полная последовательность:

```
onCreate
    onSurfaceCreated
        onSurfaceChanged (1+ вызывается в любом порядке)
        onOffsetsChanged (0+ вызывается в любом порядке)
        onVisibilityChanged (0+ вызывается в любом порядке)
    onSurfaceDestroyed
onDestroy
```

Мы заполним все эти методы ближе к концу главы.

Также нам нужно еще несколько директив import для предотвращения ошибок компилятора. Я обычно позволяю Eclipse создать их для меня в процессе кодирования (используя Content Assist (Ctrl+Пробел), или Quick Fix (Ctrl+F1), или в пакетном режиме командой Source ▸ Organize Imports (Ctrl+Shift+0). Но вы можете просто ввести их с клавиатуры. Вот полный список:

Wallpaper/src/org/example/wallpaper/Wallpaper.java

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import javax.microedition.khronos.egl.EGL10;
import javax.microedition.khronos.egl.EGL11;
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.egl.EGLContext;
import javax.microedition.khronos.egl.EGLDisplay;
import javax.microedition.khronos.egl.EGLSurface;
import javax.microedition.khronos.opengles.GL10;
import android.service.wallpaper.WallpaperService;
import android.view.SurfaceHolder;
```

Далее нам нужно позаимствовать немного кода для рисования нашего куба.

Повторное использование кода OpenGL

Вам необязательно использовать OpenGL для интерактивных обоев, но мне это нравится, так как это работает быстрее, чем обычные 2D-библиотеки Android (см. главу 4 «Введение в 2D-графику»). К тому же у нас есть отличный пример с вращающимся трехмерным кубом, уже написанный в другой главе (глава 10 «3D-графика в OpenGL»).

Возьмите три файла из этого проекта: GLCube.java, GLRenderer.java и android.png. Поместите Java-файлы в пакет org.example.wallpaper (другими словами, в папку org/example/wallpaper), а PNG-файл — в папку res/drawable-nodpi.

Если вы еще не проработали эту главу, для загрузки всех необходимых файлов воспользуйтесь веб-сайтом издательства «Питер», чтобы скачать дополнительные материалы. Распакуйте архив с исходным кодом во временную папку и скопируйте файлы из проекта OpenGL.

Не вносите какие-либо изменения в исходный код, за исключением имени пакета в верхней части обоих Java-файлов, который должен выглядеть следующим образом:

Wallpaper/src/org/example/wallpaper/GLRenderer.java

```
package org.example.wallpaper;
```

Сейчас мы лишь разберемся с тем, как вызывать код из нашего нового проекта обоев.

Создание и уничтожение механизма отрисовки

Для начала давайте определим несколько полей для использования механизма отрисовки. Поместите это в начало класса MyEngine:

Wallpaper/src/org/example/wallpaper/Wallpaper.java

```
private GLRenderer glRenderer;
private GL10 gl;
private EGL10 egl;
private EGLContext glc;
```

```
private EGLDisplay glDisplay;
private EGLSurface glSurface;
private ExecutorService executor;
private Runnable drawCommand;
```

Самая важная переменная здесь — executor. В Java исполнитель — это объект, который может запускать сниппеты (которые называются *runnables* (запускаемые подпрограммы)) асинхронно, в других процессах. Когда механизм визуализации обоев создается впервые, мы собираемся «пригласить» одного такого исполнителя для обработки всего взаимодействия с OpenGL.

Нам нужно сделать это, так как OpenGL может быть вызвана лишь из одного потока. Сервис создает поток для вывода в фоновом режиме в любом случае, и мы не можем вызвать часть OpenGL-кода в фоновом процессе и часть — в активном процессе. Поэтому мы используем исполнитель для всех вызовов. Определение метода onCreate() показывает, как его инициализировать.

Wallpaper/src/org/example/wallpaper/Wallpaper.java

```
@Override
public void onCreate(final SurfaceHolder holder) {
    super.onCreate(holder);
    executor = Executors.newSingleThreadExecutor();
    drawCommand = new Runnable() {
        public void run() {
            glRenderer.onDrawFrame(gl);
            egl.eglSwapBuffers(glDisplay, glSurface);
            if (isVisible()
                && egl.eglGetError() != EGL11.EGL_CONTEXT_LOST) {
                executor.execute(drawCommand);
            }
        }
    };
}
```

Помимо исполнителя, мы также создаем запускаемую подпрограмму, названную drawCommand, которая будет использована позже, для рисования кадров анимации куба. Это — анонимный внутренний класс Java, поэтому он имеет доступ ко всем полям и параметрам с ключевым словом *final* их родительских классов. Обратите внимание, что мы еще не инициализировали их переменные (такие, как glRenderer и glDisplay), но это нормально, поскольку мы лишь определяем этот код для того, чтобы запустить его позже, после того, как все будет готово.

Помимо onCreate() существует onDestroy(). onDestroy() вызывается, когда Android завершает работу механизма визуализации обоев. Все, что ему нужно, — завершить работу исполнителя, которого мы создали в методе onCreate():

Wallpaper/src/org/example/wallpaper/Wallpaper.java

```
@Override
public void onDestroy() {
    executor.shutdownNow();
    super.onDestroy();
};
```

Обратите внимание на то, что мы вызываем super.onDestroy() в конце метода, а не в начале, как мы делали в большинстве случаев. Это соответствует стандарту Java — позволить надклассу навести после себя порядок — зеркальное отражение способа, которым он был создан. Я не знаю, насколько это необходимо в данном случае, но, следуя соглашению, мы можем даже об этом не думать.

Управление поверхностью

В течение жизненного цикла механизма визуализации Android создаст объект Surface, представляющий фон домашнего экрана, на котором механизм сможет выводить изображения. Когда поверхность создается, вызывается метод onSurfaceCreated(). Мы используем эту возможность для инициализации OpenGL и нашего класса GLRenderer, который мы скопировали из другого проекта:

Wallpaper/src/org/example/wallpaper/Wallpaper.java

```
@Override
public void onSurfaceCreated(final SurfaceHolder holder) {
    super.onSurfaceCreated(holder);
    Runnable surfaceCreatedCommand = new Runnable() {
        @Override
        public void run() {
            // Инициализация OpenGL
            egl = (EGL10) EGLContext.getEGL();
            glDisplay = egl.eglGetDisplay(EGL10.EGL_DEFAULT_DISPLAY);
            int[] version = new int[2];
            egl.eglInitialize(glDisplay, version);
            int[] configSpec = { EGL10.EGL_RED_SIZE, 5,
                EGL10.EGL_GREEN_SIZE, 6, EGL10.EGL_BLUE_SIZE,
                5, EGL10.EGL_DEPTH_SIZE, 16, EGL10.EGL_NONE };
            EGLConfig[] configs = new EGLConfig[1];
            int[] numConfig = new int[1];
            egl.eglChooseConfig(glDisplay, configSpec, configs,
                1, numConfig);
            EGLConfig config = configs[0];
            glc = egl.eglCreateContext(glDisplay, config,
                EGL10.EGL_NO_CONTEXT, null);
            glSurface = egl.eglCreateWindowSurface(glDisplay,
                config, holder, null);
            egl.eglMakeCurrent(glDisplay, glSurface, glSurface,
                glc);
            gl = (GL10) (glc.getGL());
            // Инициализация рендеринга
            glRenderer = new GLRenderer(Wallpaper.this);
            glRenderer.onSurfaceCreated(gl, config);
        }
    };
    executor.execute(surfaceCreatedCommand);
}
```

Было бы неплохо со стороны Android предоставить вспомогательный класс, чтобы скрыть некоторые из шаблонных деталей OpenGL от программиста (что-то наподобие GLSurfaceView, но для обоев), но пока просто внимательно скопируйте код.

Метод onSurfaceDestroyed() вызывается, когда фоновое изображение закрывается. Это подходящее место для того, чтобы завершить работу всех инструментов OpenGL, которые мы инициализировали ранее:

Wallpaper/src/org/example/wallpaper/Wallpaper.java

```
@Override
public void onSurfaceDestroyed(final SurfaceHolder holder) {
    Runnable surfaceDestroyedCommand = new Runnable() {
        public void run() {
            // Освобождаем ресурсы OpenGL
            egl.eglMakeCurrent(glDisplay, EGL10.EGL_NO_SURFACE,
                EGL10.EGL_NO_SURFACE, EGL10.EGL_NO_CONTEXT);
            egl.eglDestroySurface(glDisplay, glSurface);
            egl.eglDestroyContext(glDisplay, glc);
            egl.eglTerminate(glDisplay);
        };
    };
    executor.execute(surfaceDestroyedCommand);
    super.onSurfaceDestroyed(holder);
}
```

До тех пор, пока поверхность существует, Android вызывает метод onSurfaceChanged() , чтобы сообщить о ее ширине и высоте.

Wallpaper/src/org/example/wallpaper/Wallpaper.java

```
@Override
public void onSurfaceChanged(final SurfaceHolder holder,
        final int format, final int width, final int height) {
    super.onSurfaceChanged(holder, format, width, height);
    Runnable surfaceChangedCommand = new Runnable() {
        public void run() {
            glRenderer.onSurfaceChanged(gl, width, height);
        };
    };
    executor.execute(surfaceChangedCommand);
}
```

Пока что поверхность не видна пользователю, поэтому мы все еще ничего на ней не рисовали. Это следует исправить.

Делаем обои видимыми

После инициализации WallpaperService инициализируются Engine и Surface (вот так так!), единственное, что осталось, — это сделать поверхность видимой. Когда она для этого готова, Android вызывает метод onVisibilityChanged() с логическим параметром, который сообщает, следует ли сделать ее видимой или невидимой:

Wallpaper/src/org/example/wallpaper/Wallpaper.java

```
@Override
public void onVisibilityChanged(final boolean visible) {
    super.onVisibilityChanged(visible);
    if (visible) {
        executor.execute(drawCommand);
    }
}
```

Если она видима, все, что нам нужно сделать, — поставить в очередь вызов запускаемой подпрограммы drawCommand, которая будет рисовать один кадр анимации и, если поверхность все еще видима, ставить ее в очередь снова и снова. Для экономии энергии батарей очень важно обновлять обои только тогда, когда они видимы.

Сейчас вы можете испытать пример на эмуляторе или на реальном устройстве. Щелкните правой кнопкой по проекту, выберите команду Run As ▸ Android Application (Запустить как ▸ Приложение Android). Как и в случае с виджетами, обои лишь устанавливаются на устройство, но не запускаются на выполнение из Eclipse. Для того чтобы их действительно запустить, перейдите к устройству или к эмулятору, нажмите и удерживайте палец (или мышь) на домашнем экране. Появится меню со списком типов объектов, которые можно добавить (рис. 12.3).

Выберите Wallpaper (Обои) из меню и затем Live wallpaper (Интерактивные обои). Появится список обоев. Выберите те, которые называются *Hello, Android!* И обои должны начать выполняться в режиме предпросмотра (если этого не произойдет, следуйте инструкциям раздела 3.10 «Отладка» для диагностики проблемы). Прикоснитесь к кнопке Set wallpaper (Установить обои) для того, чтобы разместить обои на домашнем экране, или нажмите кнопку Back (Назад) для того, чтобы вернуться к списку. Для того чтобы увидеть интерактивные обои в действии, см. рис. 12.6.

Обработка пользовательского ввода

При использовании обычных, статических обоев и перемещении домашнего экрана влево или вправо при помощи пальца обои также перемещаются. Однако при попытке проделать это с интерактивными обоями ничего не произойдет.

Для того чтобы получить такую функциональность, мы должны реализовать метод onOffsetChanged():

Wallpaper/src/org/example/wallpaper/Wallpaper.java

```
@Override
public void onOffsetsChanged(final float xOffset,
        final float yOffset, final float xOffsetStep,
        final float yOffsetStep, final int xPixelOffset,
        final int yPixelOffset) {
    super.onOffsetsChanged(xOffset, yOffset, xOffsetStep,
        yOffsetStep, xPixelOffset, yPixelOffset);
    Runnable offsetsChangedCommand = new Runnable() {
```

```
        public void run() {
            if (xOffsetStep != 0f) {
                glRenderer.setParallax(xOffset - 0.5f);
            }
        };
    };
    executor.execute(offsetsChangedCommand);
}
```

Если xOffsetStep установлен в 0, это означает, что перемещение отключено (например, в режиме предварительного просмотра). Для других значений мы сдвигаем вид, основываясь на переменной xOffset. Если смещение равно 0, пользователь перемещает экран влево, если 1 — он переместил все вправо. Сможете догадаться, что означает 0,5?

Сейчас я беру обратно мои слова о том, что не следует модифицировать GLRenderer. Так как пример OpenGL не принимает команды пользователя, мы добавим дополнительные поля и метод setParallax(), который устанавливает их в классе GLRenderer:

Wallpaper/src/org/example/wallpaper/GLRenderer.java

```
class GLRenderer implements GLSurfaceView.Renderer {
    // ...
    private float xOffset;
    public void setParallax(float xOffset) {
        this.xOffset = -xOffset;
    }
}
```

Затем внутри метода GLRenderer.onDrawFrame() мы изменим одну строку для использования нового поля для перемещения модели немного влево или вправо:

Wallpaper/src/org/example/wallpaper/GLRenderer.java

```
public void onDrawFrame(GL10 gl) {
    // ...
    // Располагаем модель таким образом, чтобы мы могли ее видеть
    gl.glMatrixMode(GL10.GL_MODELVIEW);
    gl.glLoadIdentity();
    gl.glTranslatef(xOffset, 0, -3.0f);
    // Здесь располагаются другие команды рисования...
}
```

Мы можем просто менять направление взгляда, но легче переместить куб. Вот и все! Сейчас испытайте пример снова, и вы сможете легко перемещать экран вправо или влево.

Есть еще пара способов взаимодействия с интерактивными обоями: *команды* и *события касания*. Для поддержки команд реализуйте метод onCommand(), для поддержки необработанных событий касаний — метод onTouchEvent(). Я оставляю вам это в качестве самостоятельного упражнения.

12.3. Вперед>>

В этой главе мы узнали, как оживить домашний экран Android с помощью виджетов и интерактивных обоев. Android Market полон примеров и того и другого, созданного такими же людьми, как вы. Я буду рад встретить вас там, поэтому читайте главу 14 «Публикация на Android Market» , чтобы найти инструкции и содействие.

Обратите внимание: если ваше приложение — это виджет, интерактивные обои или другое расширение домашнего экрана, не разрешайте устанавливать его с SD-карты (см. раздел 13.6 «Установка на SD-карту» для получения более подробных сведений).

Когда Android вышел впервые, была лишь одна его версия и одна модель телефона, о которых следовало беспокоиться. Сейчас мы имеем дело с десятками устройств различных форм и размеров, на которых исполняется как минимум четыре различные версии Android. Итак, вас зовут не Элли, вы больше не в Канзасе, но следующая глава поможет спастись от летающих обезьян.

13 Написав однажды, протестируй везде

Сегодня Android можно найти на огромном количестве мобильных телефонов, планшетных компьютеров и на многих других устройствах. Это и хорошо, и плохо. Это хорошо для потребителей, так как они могут выбирать между Android-устройствами различных форм, размеров и цен. Но для разработчика попытка поддержки всех этих вариантов может стать настоящим проклятием.

Усложняет дело то, что из-за быстрой разработки новых платформ в системе остаются «хвосты» от поддержки предыдущих версий устройств Android. Следующая таблица показывает все выпущенные версии Android[1]:

Версия	Кодовое название[2]	API	Дата выпуска	Комментарии
1.0	BASE	1	Октябрь, 2008	Не используется
1.1	BASE_1_1	2	Февраль, 2009	Не используется
1.5	CUPCAKE	3	Май, 2009	Виджеты
1.6	DONUT	4	Сентябрь, 2009	Дисплеи высокой и низкой плотности
2.0	ECLAIR	5	Ноябрь, 2009	Не используется
2.0.1	ECLAIR_0_1	6	Декабрь, 2009	Мульти-тач, не используется
2.1	ECLAIR_MR1	7	Январь, 2010	Интерактивные обои
2.2	FROYO	8	Май, 2010	Установка с SD-карты
2.3	GINGERBREAD	9	Декабрь, 2010	Игровые возможности

[1] Воспользуйтесь http://d.android.com/resources/dashboard/platform-versions.html для того, чтобы найти обновленную таблицу, показывающую процент устройств, работающих под управлением каждой из версий Android.

[2] Коды версий Android и уровни API заданы в классе Build.VERSION_CODES.

В этой главе рассказывается о том, как поддерживать множество версий Android и разрешений экрана в одной программе. Первый шаг — это тестирование.

13.1. Джентльмены, запустите ваши эмуляторы

В большинстве примеров в этой книге я говорил о целевой платформе для ваших приложений Android 2.2 (также известной как FroYo). Однако с этим советом есть одна небольшая проблема: ваши программы могут не запуститься на телефонах, на которых установлена одна из более старых версий Android.

Единственный подходящий способ понять, будет ли программа работать — протестировать ее. И не спешите покупать по телефону на каждую версию Android, лучший способ протестировать вашу программу на совместимость с различными версиями Android и с разными разрешениями экрана — испытать ее с помощью эмулятора.

Для этого создайте несколько виртуальных устройств с различными версиями и скинами. *Скин* задает ширину, высоту и плотность пикселей эмулируемого устройства. В дополнение к AVD em22, которое вы создали в подразделе 1.3, «Создание AVD», я советую создать следующие виртуальные устройства для тестирования:

Имя	Платформа	Скин	Устройство-прототип
em15	1.5	HVGA (320×480)	HTC G1, Eris
em16	1.6	HVGA (320×480)	HTC Hero
em16-qvga	1.6	QVGA (200×320)	HTC Tattoo
em21-854	2.1	WVGA854 (480×854)	Motorola Droid (Sholes)
em22-800	2.2	WVGA800 (480×800)	HTC Nexus One
em22-1024	2.2	Пользовательское (1024×600)	Notion Ink Adam

Используйте AVD em22 для разработки, а затем тестируйте вашу программу на других AVD, прежде чем выпустите приложение. И не забывайте пробовать его в портретном и ландшафтном режимах экрана. В эмуляторе нажмите Crtl+F11 или используйте клавиши 7 или 9 на цифровой клавиатуре (при выключенном режиме NumLock) для переключения между портретным и ландшафтным режимами.

Если у вас не очень мощный компьютер, вам не удастся запустить все эти AVD одновременно. Практика показывает, что лучше всего запускать по одному экземпляру. Попробуйте это сейчас, запустив эмулятор em16 (Android 1.6). Подождите, пока эмулятор доберется до домашнего экрана, и отключите блокировку экрана, если она появится. Теперь давайте посмотрим на возможные проблемы.

13.2. Компоновка для множества версий Android

Для начала попробуем запустить программу «Hello, Android» из раздела 1.2 «Создание первой программы» в эмуляторе для версии платформы 1.6. Из меню, выберите Run ▸ Run Configurations. Найдите конфигурацию для HelloAndroid и выберите ее, или создайте новую конфигурацию приложения Android, если она не существует.

Щелкните по вкладке Target и установите параметр Deployment Target Selection Mode в Manual, теперь нажмите кнопку Run.

Eclipse покажет диалоговое окно, которое задаст вопрос о выборе устройства Android. Если вы заметили красный крестик около имени эмулятора — не обращайте на него внимания. Выберите устройство, которое называется *em16*, затем щелкните OK. В окне Console появится следующее сообщение об ошибке:

```
ERROR: Application requires API version 8. Device API version
is 4 (Android 1.6).
Launch canceled!
```

(Ошибка: приложение требует API версии 8. Версия API устройства — 4 (Android 1.6). Запуск отменен).

ОБРАТИТЕ ВНИМАНИЕ!

При появлении ошибки с текстом «Application does not specify an API level requirement» (В приложении не задан требуемый уровень API) укажите минимальную версию SDK (мы рассмотрим это ниже). Если подобная ошибка не появилась и приложение запустилось в em16-эмуляторе, это означает, что вы уже применили к нему изменения, описываемые в этом разделе. И, наконец, если Eclipse запустит новое окно эмулятора, это означает, что ваши установки выбора целевой платформы для развертывания переключены в режим Automatic. Измените их на Manual и попробуйте снова.

Ошибка возникает, так как мы задали версию 2.2 как версию целевой платформы Android, когда создавали проект. Несмотря на то что мы сделали некоторые настройки, программа может не запуститься на более старых версиях. Такие устройства даже не отобразят ее в списке при обращении на Android Market. Но я знаю, что эта программа будет отлично работать на любых версиях Android. Итак, как нам сообщить об этом Android?

Просто: в манифесте Android установите целевую версию вашего SDK на одно число и минимальную версию SDK — на другое. Это позволит ориентироваться в первую очередь на новую версию Android, но работать и с телефонами, имеющими более старые версии платформы.

Для того чтобы это сделать, отредактируйте файл AndroidManifest.xml и измените строку, которая содержит следующий текст:

```
<uses-sdk android:minSdkVersion="8" />
```

на такую строку:

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

Если у вас этой строки нет, добавьте ее перед тегом </manifest>.

Эта строка сообщит Android о том, что ваша программа скомпилирована для Android 2.2 (уровень API 8), но будет нормально работать и на Android 1.5 (уровень

API 3). Сохраните файл и попробуйте запустить проект снова. Он должен работать в этот раз без ошибок. При получении предупреждения следующего вида: «Manifest min SDK version (3) is lover than project target API level (8)» (Минимальная версия SDK (3), заявленная в манифесте, меньше, чем уровень целевого API (8)) просто проигнорируйте его.

К несчастью, для некоторых программ простое сообщение, что они поддерживают версию 3, не приводит к действительной поддержке этой версии. Мы увидим пример этого далее.

13.3. Разворачивание программы на различных API Android

Иногда вам нужно использовать API, которые имеются не во всех версиях Android. Например, в примере обработки сенсорного ввода (глава 11 «Мульти-тач») мы использовали некоторые новые методы класса MotionEvent, которых не было до Android 2.0. При попытке запустить этот пример на эмуляторе em16 вы получите ошибку, как на рис. 13.1.

Рис. 13.1. Пример Touch вызывал ошибку на Android 1.6

Откройте окно LogCat в Eclipse (Window ▸ Show View ▸ Other ▸ Android Log (Окно ▸ Показать окно ▸ Другое ▸ Журнал Android) и слегка прокрутите его содержимое назад, вы найдете более подробное сообщение об ошибке, которое будет выглядеть примерно так:

```
Could not find method android.view.MotionEvent.getPointerCount,
referenced from method org.example.touch.Touch.dumpEvent
VFY: unable to resolve virtual method 11:
Landroid/view/MotionEvent;.getPointerCount ()I
Verifier rejected class Lorg/example/touch/Touch;
Class init failed in newInstance call (Lorg/example/touch/Touch;)
Uncaught handler: thread main exiting due to uncaught exception
java.lang.VerifyError: org.example.touch.Touch
at java.lang.Class.newInstanceImpl(Native Method)
at java.lang.Class.newInstance(Class.java:1472)
...
```

Важная часть здесь — исключение VerifyError. Во время выполнения Android выдает исключение VerifyError, когда он использует классы, использующие методы, не существующие в другом классе. В этом случае класс Touch ссылается на метод getPointerCount() в классе MotionEvent. Класс MotionEvent существует в версии 1.5, но метод getPointerCount() не был представлен до версии 2.0. Обратите внимание на то, что он не жалуется на константы, которые мы используем, вроде ACTION_POINTER_DOWN, так как они встроены в нашу программу компилятором во время сборки программы.

Было бы неплохо, если бы мы могли изменить определение класса MotionEvent для того, чтобы добавить недостающий метод, но, в отличие от JavaScript или Ruby, Java этого не поддерживает. Мы можем сделать что-то подобное, используя одну из трех методик:

- *Subclassing*: мы можем создать новый класс, который расширяет возможности MotionEvent и добавляет новый метод. К сожалению, MotionEvent — это завершенный класс (final), что в Java означает, что он не может быть расширен, то есть этот метод здесь не заработает.
- *Reflection*: используя утилиты из пакета java.lang.reflect, мы можем написать код для тестирования экземпляра нового метода. Если он существует, мы вызываем его, если не существует, мы делаем что-то еще, например возвращаем 1. Это будет работать, но рефлексия в Java медленна и сложна в программировании.
- *Delegation and factory*: мы можем создать два класса, один — использующий старую версию Android, и один — для более новых версий. Первый будет содержать макеты новых методов, в то время как второй будет делегировать все вызовы к настоящим новым методам. Затем мы выберем, какой класс создавать, используя статический фабричный метод (метод, который вызывается вместо ключевого слова *new* для создания класса). Эта техника довольно проста и может быть использована для поддержки большинства различий в API, поэтому здесь мы воспользуемся ею.

Одна из наших целей заключается в том, чтобы минимизировать изменения в исходном классе Touch. Мы начнем с замены всех ссылок на MotionEvent на новый класс WrapMotionEvent, как здесь:

Touchv2/src/org/example/touch/Touch.java

```
@Override
public boolean onTouch(View v, MotionEvent rawEvent) {
    WrapMotionEvent event = WrapMotionEvent.wrap(rawEvent);
    // ...
}
private void dumpEvent(WrapMotionEvent event) {
    // ...
}
private float spacing(WrapMotionEvent event) {
    // ...
}
private void midPoint(PointF point, WrapMotionEvent event) {
    // ...
}
```

В методе onTouch() мы принимаем необработанные данные MotionEvent, переданные Android, и конвертируем их во WrapMotionEvent, вызывая статический фабричный метод WrapMotionEvent.wrap(). Все остальное, до конца класса Touch, осталось нетронутым.

Сейчас давайте создадим класс WrapMotionEvent. Делегирование весьма распространено в Java, поэтому Eclipse обеспечивает команду для того, чтобы упростить этот процесс. Щелкните на пакете org.example.touch в окне Package Explorer, и, затем выберите команду File ▸ New ▸ Class. Введите имя нового класса (WrapMotionEvent) и нажмите Return. Теперь добавьте в класс поле для события, которое мы хотим заключить в оболочку. Код должен выглядеть так:

Touchv2/src/org/example/touch/WrapMotionEvent.java

```
package org.example.touch;
import android.view.MotionEvent;
public class WrapMotionEvent {
    protected MotionEvent event;
}
```

В редакторе Java щелкните на переменной события и затем выберите команду меню Source ▸ Generate Delegate Menhods. Eclipse предлагает большой список возможных методов, но нам не нужны они все, поэтому снимите с них выделение и выберите только те, которые действительно используются в программе: getAction(), getPointerCount(), getPointerId(int), getX(), getX(int), getY() и getY(int). После завершения диалоговое окно должно выглядеть так, как показано на рис. 13.2. Нажмите OK для создания кода.

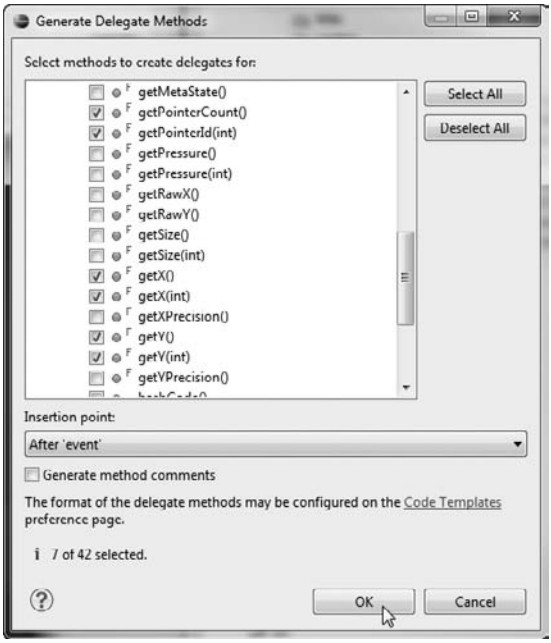

Рис. 13.2. Позвольте Eclipse создать для вас методы-делегаты

Прежде чем делать что-нибудь еще, сохраните файл WrapMotionEvent.java и сделайте копию, назвав ее EclairMotionEvent.java. Мы вернемся к ней совсем скоро.

При текущих установках WrapMotionEvent вызывает несколько методов, которые не существуют в более старых версиях Android, поэтому нам нужно их заменить. Наведите курсор мыши на вызов каждого из методов, и вы увидите, с какими из них возникли проблемы.

Новые методы сообщат «Since: API Level 5» (С уровня API 5), старые — «Since: API Level 1» (С уровня API 1). Другой способ узнать это — временно изменить целевую платформу на Android 1.6, перекомпилировать проект и посмотреть, где возникнут ошибки.

Вот новый код, который будет работать под Android 1.6:

Touchv2/src/org/example/touch/WrapMotionEvent.java

```
package org.example.touch;
import android.view.MotionEvent;
public class WrapMotionEvent {
    protected MotionEvent event;
    public int getAction() {
        return event.getAction();
    }
    public float getX() {
        return event.getX();
    }
    public float getX(int pointerIndex) {
        verifyPointerIndex(pointerIndex);
        return getX();
    }
    public float getY() {
        return event.getY();
    }
    public float getY(int pointerIndex) {
        verifyPointerIndex(pointerIndex);
        return getY();
    }
    public int getPointerCount() {
        return 1;
    }
    public int getPointerId(int pointerIndex) {
        verifyPointerIndex(pointerIndex);
        return 0;
    }
    private void verifyPointerIndex(int pointerIndex) {
        if (pointerIndex > 0) {
            throw new IllegalArgumentException(
                "Invalid pointer index for Donut/Cupcake" );
        }
    }
}
```

В этой версии getPointerCount() всегда возвращает 1, показывая, что произошло нажатие лишь одним пальцем. Это гарантирует, что getX(int) и getY(int) никогда не будут вызваны с индексом указателя большим, чем 0, но просто на всякий случай я добавил метод verifyPointerIndex() для проверки на эту ошибку.

Далее нам нужно добавить метод wrap() и конструктор для класса WrapMotionEvent:

Touchv2/src/org/example/touch/WrapMotionEvent.java

```
protected WrapMotionEvent(MotionEvent event) {
    this.event = event;
}
static public WrapMotionEvent wrap(MotionEvent event) {
    try {
        return new EclairMotionEvent(event);
    } catch (VerifyError e) {
        return new WrapMotionEvent(event);
    }
}
```

Метод wrap() — это наш статический фабричный метод. Для начала он пытается создать экземпляр класса EclairMotionEvent. Это не получится под Android 1.5 и 1.6, так как EclairMotionEvent использует новые методы, представленные в Android 2.0 (Eclair). Если он не можете создать класс EclairMotionEvent, тогда, вместо этого, он создает экземпляр класса WrapMotionEvent[1].

Сейчас пришло время поработать над классом EclairMotionEvent, который мы сохранили ранее. Мы завершим работу над ним, сделав его расширяющим WrapMotionEvent и добавив конструктор. Мы так же изымем из него метод getAction() и версии без параметров методов getX() и getY(), так как они существуют во всех версиях Android и уже реализованы во WrapMotionEvent. Вот полное определение класса:

Touchv2/src/org/example/touch/EclairMotionEvent.java

```
package org.example.touch;
import android.view.MotionEvent;
public class EclairMotionEvent extends WrapMotionEvent {
    protected EclairMotionEvent(MotionEvent event) {
        super(event);
    }
    public float getX(int pointerIndex) {
        return event.getX(pointerIndex);
    }
    public float getY(int pointerIndex) {
        return event.getY(pointerIndex);
    }
```

[1] Блюстители чистоты шаблонов дизайна, возможно, высмеют этот код, сказав, что мне следует создать отдельные классы для хранения фабричных методов и общий интерфейс для классов WrapMotionEvent и EclairMotionEvent. Но так проще, и это работает, что важнее в моей книге.

```
    public int getPointerCount() {
        return event.getPointerCount();
    }
    public int getPointerId(int pointerIndex) {
        return event.getPointerId(pointerIndex);
    }
}
```

Если вы попытаетесь запустить программу сейчас, она будет работать на эмуляторе 1.6 так же, как и на новых версиях (2.0 и выше). На старых версиях, конечно, будет утеряна некоторая функциональность. Мульти-тач не поддерживается в версии 1.6, поэтому вам не удастся использовать жест изменения масштаба двумя пальцами для сжимания или растягивания изображения. Но вы сможете перемещать изображения, используя жест перетаскивания.

В реальной программе вы можете внедрить какой-нибудь альтернативный способ изменения размера изображения, когда изменение масштаба двумя пальцами недоступно. Например, добавьте кнопки для увеличения и уменьшения масштаба.

Просто для развлечения запустите *em15* (эмулятор версии 1.5) и выполните на нем программу. Она выглядит нормально, но попытайтесь сделать жест перетаскивания — ничего не происходит! Мы нашли ошибку в Android 1.5.

13.4. Парад ошибок

Оказывается, Android 1.5 (Cupkake) имеет ошибку в классе ImageView. Ошибка не дает работать методу setImageMatrix(), когда ImageView находится в «матричном» режиме — своего рода ирония судьбы, если об этом подумать.

К несчастью, нет полного списка ошибок, поэтому, решив, что это ошибка в Android (в противоположность ошибке в нашем коде или неправильного понимания того, как что-то работает), предпримем небольшое расследование. Рассмотрим этапы, через которые в таких ситуациях прохожу я:

1. Мое первое подозрение пало на код, который мы только что добавили: классы WrapMotionEvent и EclairMotionEvent. Я поместил в них несколько команд записи в журнал для проверки правильности обработки событий и правильности создания матрицы трансформации. Здесь проблем не возникло, поэтому я начал подозревать, что что-то не так с классом ImageView или с матричными операциями.
2. Далее я проверил замечания к выпуску[1] каждой из версий Android, чтобы увидеть, были ли где-нибудь серьезные изменения, упомянутые как влияющие на класс ImageView. Серьезные изменения — это обновления, которые могут привести работающий код к прекращению работы. Я не нашел ничего, что выглядело бы относящимся к этой проблеме.

[1] http://d.android.com/sdk/RELEASENOTES.html

3. Далее я поискал в базе данных[1] ошибок Android по ключевым словам *ImageView* и *matrix*. К несчастью, общедоступная база данных ошибок неполна, так как Google поддерживает свой собственный закрытый список ошибок, в итоге я ничего не нашел.
4. Затем я посмотрел все форумы[2] разработчиков Android для того, чтобы увидеть, сталкивался ли кто-нибудь еще с подобной проблемой. Отдельные группы имеют собственные формы поиска, но самый быстрый и точный способ поиска по форумам — просто использовать поисковую машину Google и искать по всему Интернету. Используя те же самые ключевые слова (*ImageView* и *Matrix*), я нашел сообщения от 2009 года, в которых обсуждали то же самое. К несчастью, здесь не было последующих или связанных сообщений, однако я знал, что я на правильном пути.
5. Наконец, я перешел к исходному коду[3]. За несколькими исключениями, весь код Android доступен в открытом онлайновом хранилище. У меня была подсказка о том, где расположен исходный код класса ImageView в дереве исходного кода, так как одни из предыдущих поисков дали мне имя пути. Это был проект platform/frameworks/base.git, находящийся в папке core/java/widget. Я открыл историю, используя веб-интерфейс хранилища[4]. И вот оно, изменение, сделанное 30 июля 2009 года с комментарием: «Исправлена ошибка в ImageView: матрица рисования не обновляется, когда вызывается метод setImageMatrix». Исправление этой ошибки было сделано между версиями Android Cupcake и Donut, что объясняет, почему проблема наблюдается в 1.5, но не в 1.6.
6. Почти во всех случаях можно найти решение без обращения к чтению исходного кода Android. Но приятно знать, что он доступен, если он вам понадобится. Изучив код и испробовав несколько вариантов, я смог обойти ошибку.

   Добавьте эти строки в конец метода onCreate() в классе Touch:

Touchv2/src/org/example/touch/Touch.java

```
@Override
public void onCreate(Bundle savedInstanceState) {
    // ...
    // Обход ошибки в Cupcake
    matrix.setTranslate(1f, 1f);
    view.setImageMatrix(matrix);
}
```

7. Запуская программу на эмуляторе, используя различные версии Android, убедитесь, что решили проблему в Android 1.5 и не испортили ничего в более поздних версиях. Как только ваша программа заработает на разных версиях Android, проверьте ее работоспособность с различными разрешениями экрана.

---

[1] http://b.android.com

[2] http://d.android.com/resources/community-groups.html

[3] http://source.android.com

[4] https://android.git.kernel.org/?p=platform/frameworks/base.git;a=history;f=core/java/android/widget/ImageView.java

13.5. Все экраны, большие и маленькие

Поддержка различных размеров, разрешений и плотности пикселей экранов важна, так как это позволяет вашему приложению отлично выглядеть на как можно большем количестве Android-устройств. В отличие от iPhone, который имеет лишь один стандартный экран (ну хорошо, три, если участь iPad и iPhone 4), устройства под управлением Android имеют самые разные экраны.

Android пытается настроить размер пользовательского интерфейса ваших программ для того, чтобы он наилучшим образом подошел для конкретного устройства, но он не всегда делает это достаточно хорошо. Единственный способ узнать, так ли это, как вы себе представляли, — тестирование. Используйте скины эмулятора, рекомендованные в разделе 13.1 «Джентльмены, запустите ваши эмуляторы», и это придаст вам уверенности, что ваша программа будет работать на устройствах с наиболее распространенными размерами экрана. Если вы хотите точно настроить макет или изображение для конкретной конфигурации, используйте суффиксы в именах папки ресурсов.

Например, поместите изображения для экранов высокой плотности в папку res/drawable-hdpi, для экранов средней плотности — в res/drawable-mdpi и для экранов низкой плотности — в папку res/drawable-ldpi. Во всех примерах это сделано для значков программ, которые будут отображены на домашнем экране. Изображения, которые не зависят от плотности экрана (масштаб которых не нужно менять), располагаются в папке res/drawable-nodpi.

Вот список подходящих квалификаторов имен папок, в порядке приоритета[1]:

Квалификатор	Значения
MCC и MNC	Мобильный код страны и, необязательно, мобильный код сети. Я не рекомендую использовать его
Язык и регион	Двухбуквенный код языка и дополнительный двухбуквенный код региона (с предшествующим *r* в нижнем регистре). Например: fr, en-rUS, fr-rFR, es-rES
Размер экрана	small, normal, large
Широкие/высокие экраны	long, notlong
Ориентация экрана	port, land, square
Экранная плотность пикселей	ldpi, mdpi, hdpi, nodpi
Тип сенсорного экрана	notouch, stylus, finger
Доступна ли клавиатура?	keysexposed, keyshidden, keyssoft

[1] Обратитесь к http://d.android.com/guide/topics/resources/providing-resources.html#BestMatch за полным объяснением того, как Android находит подходящую папку.

Квалификатор	Значения
Тип клавиатуры	nokeys, qwerty, 12key
Доступны ли навигационные клавиши?	navexposed, navhidden
Тип навигационных клавиш	nonav, dpad, trackball, wheel
Разрешения экрана	320×240, 640×480 и так далее (не рекомендовано Google, но разработчики все равно это используют)
Версия SDK	Уровень API, поддерживаемый устройством (с предшествующим *v* в нижнем регистре). Например: v3, v8

Для использования более чем одного квалификатора просто соедините их друг с другом с помощью дефиса (–), который нужно поставить между ними. Например, папка res/drawable-fr-land-ldpi может содержать изображения, предназначенные для экрана с низкой плотностью пикселей в ландшафтном режиме на французском языке.

Обратите внимание: до версии 2.1 Android содержал ошибки в обработке этих квалификаторов. Для того чтобы получить совет о том, что делать с этими особенностями, посмотрите презентацию[1] Джастина Маттсона (Justin Mattson) Google I/O.

13.6. Установка на SD-карту

Начиная с версии Android 2.2 в программу включена возможность установки вашего приложение на SD-карту вместо ограниченной внутренней памяти телефона.

Для того чтобы это сделать, добавьте атрибут android:installLocation= в тег <manifest> вашего файла AndroidManifest.xml, как здесь:

```
<manifest ... android:installLocation="auto" >
```

Подходящие значения — *auto* и *preferExternal*. Я рекомендую использовать *auto*, что позволит системе самостоятельно решить, куда следует направить приложение. Задание *preferExternal* запрашивает у системы установку приложения на SD-карту, но не гарантирует этого. В любом случае, пользователь может перемещать ваше приложение между внутренней и внешней памятью с помощью установок приложения.

Этот атрибут игнорируется в более старых версиях Android. Если вы опустите его, Android всегда поместит вашу программу во внутреннее хранилище.

Итак, почему SD-карта не является местом для установки по умолчанию? Оказывается, на практике она может быть отключена, поэтому существует множество ситуаций, когда установка на внешнюю карту — это не лучшая идея. При подклю-

[1] http://code.google.com/events/io/2010/sessions/casting-wide-net-android-devices.html

чении телефона по USB-кабелю к компьютеру с целью зарядить его или перенести файлы с устройства на устройство любое работающее приложение, установленное на внешнем носителе, будет остановлено. Это является проблемой для виджетов домашнего экрана, которые просто исчезают и никогда не появляются снова.

Поэтому Google рекомендует[1] не позволять устанавливать на внешние носители программы, которые используют любую из следующих возможностей:

- управление учетными записями;
- таймеры;
- управление устройством;
- подпрограммы способов ввода данных;
- интерактивные папки;
- интерактивные обои;
- сервисы;
- адаптеры синхронизации;
- виджеты.

Так как Android 2.2 становится все более распространенным, пользователи закономерно ожидают, что все можно будет устанавливать на SD-карту. Если вы установите опцию, запрещающую это, будьте готовы объяснить вашим пользователям — почему.

13.7. Вперед>>

Поддержка множества версий Android, запускающихся на различных устройствах с различными размерами экрана, не проста. В этой главе мы рассмотрели наиболее распространенные проблемы и решения для того, чтобы вы начали работать в этом направлении. Если вы хотите большего, я рекомендую почитать замечательный практический документ «Supporting Multiple Screens» (Поддержка различных экранов) на веб-сайте Android[2].

Вы серьезно поработали над тем, чтобы довести ваше приложение до этого момента. Сейчас наступило приятное время: позволить другим пользователям воспользоваться им. В следующей главе мы узнаем, как опубликовать ваше приложение на Android Market.

---

[1] http://d.android.com/guide/appendix/install-location.html
[2] http://d.android.com/guide/practices/screens_support.html

Публикация на Android Market 14

До настоящего момента вы лишь создавали программы для запуска на эмуляторе или для загрузки на Android-телефон. Вы готовы к следующему шагу? Публикуясь на Android Market[1], вы делаете ваше приложение доступным миллионам других пользователей Android. Даная глава расскажет, как это сделать.

14.1. Подготовка

Первый шаг публикации приложения на Market — конечно, написание программы. Просмотрите дополнительные материалы, чтобы получить рекомендации о том, как это сделать. Но недостаточно просто написать код. Ваша программа должна быть высококачественной, лишенной ошибок (да, правда) и совместимой с максимальным количеством устройств. Вот несколько советов:

- Протестируйте программу как минимум на одном реальном устройстве, прежде чем ее увидит кто-то еще. Если вы забудете все остальные советы, помните хотя бы этот.
- Сохраняйте вашу программу простой и уберите из нее все лишнее. Пусть лучше ваша программа качественно делает что-то одно, чем плохо выполняет множество функций.
- Подберите подходящее имя Java-пакета, такое как *com.yourcompany.progname*, которое вы будете использовать долгое время. Android использует имя пакета, определенное в AndroidManifest.xml, как главный идентификатор вашей программы. Две программы не могут иметь то же самое имя пакета, и, загрузив однажды на Market программу с выбранным именем, вы не сможете изменить его без полного удаления ее оттуда, кроме того, вам потребуется попросить всех пользователей удалить ее и только после этого опубликовать новую программу.
- Подберите понятные значения для android:versionCode= и android:versionName= в файле[2] AndroidManifest.xml. Примите во внимание будущие обновления и оставьте для них место в схеме именования.

[1] http://market.android.com
[2] http://d.android.com/guide/publishing/versioning.html

- Следуйте лучшим приемам разработки[1] для Android, таким как разработка с учетом целей производительности, быстрого времени отклика и прозрачности для пользователя.
- Следуйте указаниям по разработке пользовательского интерфейса[2], таким, как дизайн значков, меню и подходящее использование клавиши Back.

Совместимость со множеством устройств — это одна из сложнейших задач, встающих перед программистом под Android. Одна из проблем, с которыми вы столкнетесь, — это то, что пользователи имеют различные версии платформы, установленные на их телефонах (см. главу 13 «Написав однажды, протестируй везде» для получения рекомендаций).

Хотя первые впечатления и совместимость важны, соблюдайте баланс между настройкой и улучшением вашего приложения, чтобы сделать его превосходным и с приемлемыми сроками реализации. Как только вы решите, что приложение готово, следующий шаг — подписывание приложения.

14.2. Подписывание

Android требует, чтобы все приложения были упакованы в *.apk-файл* и подписаны цифровым сертификатом, прежде чем он сможет запустить их. Это не только справедливо для эмулятора и для вашего персонального тестового устройства, но и просто необходимо для программ, которые вы хотите публиковать на Android Market.

«Но подождите, — возразите вы. — Я ничего не упаковывал и не подписывал до сегодняшнего дня». На самом деле вы все это делали. Android SDK скрыто компилировал и подписывал все, используя сертификат, который Google создал, используя известный ему псевдоним и пароль. Так как пароль известен, вас никогда о нем не спрашивали, и, возможно, вы ничего не знали о его существовании. Однако отладочный сертификат нельзя использовать для приложения на Android Market, поэтому сейчас пришло время закончить тренировки и создать ваш собственный сертификат.

Существует два способа сделать это: вручную, используя стандартные Java-команды[3] keytool и jarsigner, или автоматически, с помощью Eclipse. Я собираюсь показать способ с использованием Eclipse.

Щелкните правой кнопкой мыши по проекту в окне Package Explorer и выберите Android Tools ▸ Export Signed Application Package. Мастер проведет вас через процесс подписывания вашего приложения, включающий в себя создание новой ключевой строки и личного ключа, если у вас еще его нет. Используйте один и тот же ключ для всех версий приложения и принимайте соответствующие меры для предотвращения попадания ключа в чужие руки.

---

[1] http://d.android.com/guide/practices/design
[2] http://d.android.com/guide/practices/ui_guidelines
[3] http://d.android.com/guide/publishing/app-signing.html

Обратите внимание: при использовании Google Maps API (см. подраздел 8.3, «Встраивание MapView») вам понадобится новый ключ к Maps API от Google, так как он привязан к вашему цифровому сертификату. Экспортируйте программу один раз, получите новый ключ Maps API, используя онлайновые инструкции[1], измените XML-файл макета, используя новый ключ, и затем экспортируйте программу снова.

По окончании процесса вы получите *.apk-файл*, готовый для публикации.

14.3. Публикация

Android Market — это сервис, поддерживаемый Google, который предназначен для распространения программ. Для того чтобы начать публиковаться, вы должны подписаться как зарегистрированный разработчик на веб-сайте для публикации (также известном, как Developer Console)[2]. Здесь предусмотрен небольшой регистрационный сбор.

В качестве дополнительного шага, если вы захотите продавать свою программу, подпишите ее с помощью процессора платежей. Веб-сайт для публикаций проинструктирует вас, как это сделать. Что касается подписывания, то на данный момент поддерживается лишь платежная система Google Checkout, но в дальнейшем может быть добавлена поддержка и других процессоров, например для PayPal.

Итак, ваш проект готов к выгрузке на сайт. Щелкните по ссылке Upload Application и заполните форму. Вот три совета:

- Выключите параметр Copy Protection. Защита от копирования Android полностью небезопасна и не выполняет никаких полезных функций, кроме надоедания пользователям. Если вы беспокоитесь о пиратах, используйте сервис Android Market Licensing (Лицензирование Android Market)[3].
- Только если у вас нет причины не делать этого, установите параметр Location в All Current and Future Countries. Новые страны добавляются в систему постоянно, и благодаря этому ваше приложение будет доступно большой аудитории.
- Не указывайте ваш номер телефона в разделе Contact Information. Все пользователи Android Market смогут видеть этот номер, и они будут набирать его, когда столкнутся с проблемами. Конечно, если у вас есть выделенный номер и сотрудник телефонной поддержки, не обращайте внимания на этот совет.

Просто для примера здесь показано, как я заполнил форму для приложения Re-Translate Pro (обновленная версия примера Translate из раздела 7.4 «Использование веб-сервисов»), которое я опубликовал.

```
Application .apk file: (select Browse and Upload)
Language: English (en_US)
Title (en_US): Re-Translate Pro
```

[1] http://code.google.com/android/add-ons/google-apis/mapkey.html
[2] http://market.android.com/publish
[3] http://d.android.com/guide/publishing/licensing.html

```
Description (en_US):
Re-Translate translates a phrase from one language to another and
then back again so you can make sure you're saying what you meant.
Try the Lite version to see if you like it first.
Features:
- Translates instantly as you type
- Long press for copy/paste
- Directly send SMS/Email
Application Type: Applications
Category: Tools
Price: USD $1.99
Copy Protection Off
Locations All Current and Future Countries with Payment
Website: http://www.zdnet.com/blog/burnette
Email: ed.burnette@gmail.com
Phone: (blank)
```

Щелкните на кнопке Publish, и ваше приложение немедленно появится на Android Market для всех подходящих устройств. Это именно так, здесь нет процесса проверки, нет периода ожидания (разве что несколько секунд для обновления серверов загрузки) и нет ограничений на то, что вы можете делать в своей программе. Ну, почти. Вам все еще необходимо следовать Android Content Guidelines[1]. В противном случае ваше приложение могут удалить из Android Market. Кроме прочего, указания говорят о том, что контент не должен быть нелегальным, непристойным, пропагандировать ненависть или насилие или быть неподходящим для пользователей моложе 18 лет. К тому же он не должен нарушать права операторов. Случаи удаления программ из-за жалоб пользователей или операторов редки, но до тех пор, пока вы поступаете разумно, не должно быть чего-то, о чем следует беспокоиться.

Следующий раздел посвящен обновлениям уже опубликованного приложения.

14.4. Обновление

Предположим, ваше приложение пробыло какое-то время на Android Market, и вы хотите внести в него изменения. Самый простой вид изменений заключается в модификации метаданных приложения, то есть названия, описания, цены и другой информации, которую вы вносили в предыдущем разделе. Для изменения этой информации, не касающейся кода программы, просто выберите программу из списка в Developer Console, внесите изменения и нажмите Save.

У вас есть новая версия программы? Нет проблем, вы можете выгрузить ее с этой же страницы. Прежде чем сделать это, найдите время для проверки того, изменили ли вы два номера версии в файле AndroidManifest.xml. Увеличивайте android:versionCode= на единицу каждый раз, выгружая новую версию программы (например, с 1 на 2), и увеличивайте номер версии, который может просмотреть

[1] http://www.android.com/market/terms/developer-content-policy.html

пользователь в android:versionName=, на подходящее число (например, с 1.0.0 на 1.0.1 для небольшого исправления ошибок). Если номер версии введен верно и проект перекомпилирован и переподписан, выберите Uload Upgrade; затем щелкните на Browse, найдите новый *.apk-файл* и щелкните на Upload для отправки его на сервер.

Не обращая внимания на изменения, нажмите кнопку Publish, и эти изменения станут видимы для пользователей Android Market. Если вы вместо этого нажмете Save, изменения будут сохранены в виде черновика до тех пор, пока вы не примете их окончательно и не нажмете Publish.

Я бы посоветовал делать частые обновления, каждые две недели или около того. Это нужно для того, чтобы:

- сделать пользователя счастливым, так как он находится в твердой уверенности, что вы поддерживаете его и прислушиваетесь к его предложениям;
- удерживать ваше приложение в верхней части списка Recently Updated на Android Market. Это один из способов позволить новым пользователям обнаружить вашу программу и дать ей достойный шанс.

14.5. Заключительные положения

Вот несколько заключительных советов по работе с Android Market, которые нелегко мне достались, когда я сам публиковал там программы:

- Вы можете сделать платное приложение бесплатным, но не можете сделать бесплатное платным. Если вы вдруг захотите иметь и бесплатный и платный вариант программы, тогда создайте их заранее и одновременно. Никогда не изымайте ничего, имеющегося в бесплатной версии, иначе вы рискуете столкнуться с волной протеста.
- В текущей версии Android Market вы не можете купить собственное платное приложение. Надеюсь, это будет исправлено в будущей версии.
- Читайте все комментарии, оставляемые пользователями, но не стесняйтесь сообщать о спаме при поступлении особо грубых или вульгарных сообщений. Сохраняйте зону комментариев чистой для полезных отзывов — как положительных, так и отрицательных.
- Не теряйте мужества. Люди могут быть жестокими, особенно когда дело касается анонимных комментариев. Толстая шкура и чувство юмора — неоценимые деловые инструменты.

V

Приложения

- **Приложение А. Сравнение языка API Java и Android**
- **Приложение Б. Библиография**

Приложение A

Сравнение языка API Java и Android

В основном Android-программы написаны на языке Java, в том виде, в котором он используется в библиотеках API Java 5 Standard Edition (SE). Я сказал «в основном», так как есть некоторые отличия. Это приложение рассматривает отличия между обычным Java и тем, что мы можем найти в Android. Если вы — опытный Java-разработчик для других платформ, обратите внимание на приложение, чтобы увидеть, какие приемы следует «забыть».

A.1. Подмножество языка

Android использует стандартный Java-компилятор для компиляции исходного кода в обычный байт-код и трансляции этого байт-кода в инструкции виртуальной машины Dalvik. Поэтому язык Java поддерживается целиком, а не только его подмножество. Сравните это с Google Web Toolkit, который имеет свой собственный транслятор Java в JavaScript. При использовании обычного компилятора и байт-кодов исходный код для библиотек, использованных в ваших приложениях, не нужен.

Уровень языка

Android поддерживает код, совместимый с Java Standard Edition 5 или более ранними версиями. Формат классов Java 6 и 7 — это неподдерживаемые возможности, которые могут быть добавлены в будущих выпусках.

Встроенные типы данных

Поддерживаются все встроенные типы данных Java, в том числе byte, char, short, int, long, float, double, Object, String и массивы. Однако на некоторых дешевых устройствах вычисления с плавающей точкой лишь эмулируются. Это означает, что они выполняются программным, а не аппаратным способом, что делает их гораздо медленнее,

чем целочисленные вычисления. Хотя для эпизодического использования это нормально подходит, избегайте использовать типы данных float или double в коде, критичном к производительности, до тех пор, пока вашим алгоритмам действительно не понадобятся вычисления с плавающей точкой или вы не будете уверены, что ваша программа будет выполняться на устройстве высшего класса.

Многопоточность и синхронизация

Множество потоков поддерживается на основе разделения времени: каждому потоку дается несколько миллисекунд на выполнение, после чего производится переключение контекста, чтобы дать ход другому процессу. Хотя Android поддерживает неограниченное количество потоков, обычно не следует использовать больше двух. Один поток выделен для главного пользовательского интерфейса (если у вас есть такой), другой используется для долгосрочных операций, таких как вычисления или обмен данными по сети.

Dalvik VM реализует ключевое слово synchronized и методы библиотек, связанные с синхронизацией, такие как Object.wait(), Object.notify() и Object.notifyAll(). Она также поддерживает пакет java.util.concurrent для более сложных алгоритмов. Используйте их, как и в любых Java-программах, чтобы несколько потоков не мешали друг другу.

Рефлексия

Хотя платформа Android поддерживает *рефлексию* (reflection) Java, примите за правило не использовать ее. Причина проста и заключается в производительности: рефлексия работает медленно. Вместо этого рассмотрите альтернативы, такие как инструменты времени компиляции и препроцессоры.

Финализация

Dalvik VM поддерживает финализацию объектов при сборке мусора так же, как обычная виртуальная машина Java. Однако большинство экспертов Java советуют не полагаться на финализаторы, так как сложно предвидеть, когда они запустятся и запустятся ли вообще. Вместо финализаторов используйте явные методы close() или terminate(). Android рассчитан на аппаратное обеспечение, ограниченное в ресурсах, поэтому важно освобождать все ресурсы, как только надобность в них пропадает.

А.2. Подмножество стандартных библиотек

Android поддерживает сравнительно большое подмножество библиотек Java Standard Edition 5.0. Некоторые из них не поддерживаются, так как просто не нужны

(такие, как печать), другие опущены, так как доступны лучшие API, специфичные для Android (такие, как пользовательские интерфейсы).

Поддерживаемые

Следующие стандартные пакеты поддерживаются в Android. Обратитесь к документации по Java 2 Platform Standard Edition 5.0 API для получения информации о том, как ими пользоваться:

- java.awt.font: несколько констант для Unicode и шрифтов;
- java.beans: несколько классов и интерфейсов для внесения изменений в JavaBeans;
- java.io: операции файлового и потокового ввода-вывода;
- java.lang (за исключением java.lang.management): поддержка языка и исключений;
- java.math: большие числа, округления, точность;
- java.net: сетевой ввод-вывод, URL, сокеты;
- java.nio: операции файлового и канального ввода-вывода;
- java.security: авторизация, сертификаты, открытые ключи;
- java.sql: интерфейс баз данных;
- java.text: форматирование, естественный язык, упорядочение;
- java.util (включая java.util.concurrent): списки, карты, наборы, массивы, коллекции;
- javax.crypto: шифрование, открытые ключи;
- javax.microedition.khronos: OpenGL-графика (из Java Micro Edition);
- javax.net: сокеты, SSL;
- javax.security (за исключением javax.security.auth.kerberos, javax.security.auth.spi, и javax.security.sasl);
- javax.sql (за исключением javax.sql.rowset): дополнительные интерфейсы баз данных;
- javax.xml.parsers: XML-парсинг;
- org.w3c.dom (но не подпакеты): DOM-узлы и элементы;
- org.xml.sax: простое API для XML.

Обратите внимание на то, что хотя обычные API Java для SQL-баз данных (JDBC) включены, вы не можете их использовать для доступа к локальной базе данных SQLite. Используйте вместо них API android.database (см. главу 9 «Работа с SQL»).

Неподдерживаемые

Эти пакеты обычно — часть Java 2 Platform Standard Edition — не поддерживаются Android:

- java.applet;
- java.awt;
- java.lang.management;
- java.rmi;
- javax.accessibility;
- javax.activity;
- javax.imageio;
- javax.management;
- javax.naming;
- javax.print;
- javax.rmi;
- javax.security.auth.kerberos;
- javax.security.auth.spi;
- javax.security.sasl;
- javax.sound;
- javax.swing;
- javax.transaction;
- javax.xml (за исключением javax.xml.parsers);
- org.ietf;
- org.omg;
- org.w3c.dom. (подпакеты).

Пакеты других производителей

В дополнение к стандартным библиотекам, перечисленным ранее, Android SDK поставляется с некоторым количеством библиотек сторонних производителей для удобства разработчиков:

- org.apache.http: HTTP-аутентификация, куки, методы и протокол;
- org.json: JavaScript Object Notation;
- org.xml.sax: XML-парсинг;
- org.xmlpull.v1: XML-парсинг.

Приложение Б

Библиография

[Bur05] Ed Burnette. *Eclipse IDE Pocket Guide*. O'Reilly & Associates, Inc, Sebastopol, CA, 2005.

[Gen06] Jonathan Gennick. *SQL Pocket Guide*. O'Reilly Media, Inc., Sebastopol, CA, second edition, 2006.

[Goe06] Brian Goetz. *Java Concurrency in Practice*. Addison-Wesley, Reading, MA, 2006.

[Owe06] Mike Owens. *The Definitive Guide to SQLite*. Apress, Berkeley, CA, 2006.