Содержание

Предисловие

Развитие Android идет с невероятной скоростью, и не отстать от этого развития – сложная задача для любого разработчика. В гонке за последними нововведениями и изменениями в API легко упустить из виду архитектурные изменения в Android. Когда компания Google анонсировала принципы материального дизайна (Material Design), даже проектировщики, долгое время обходившие Android стороной, начали обращать на него внимание.

Разработчики для Android как никто другой должны знать и понимать основные аспекты проектирования интерфейсов и идеи материального дизайна; однако, не имея многолетнего опыта проектирования, может быть сложно разобраться во всем этом. Данная книга проведет вас через процесс проектирования, от абстрактной идеи и набросков на бумаге до воплощения анимационных эффектов, применения RenderScript и создания собственных виджетов. Идея состоит в том, чтобы коснуться каждого базового аспекта и охватить его достаточно полно, чтобы вы могли общаться с дизайнерами интерфейсов на профессиональном уровне или даже создавать интерфейсы самостоятельно.

Дизайнеры решают множество задач, но наиболее важными являются удобство использования и внешняя привлекательность. Вы должны так спроектировать интерфейс своего приложения, чтобы можно было запустить его и начать пользоваться им безо всяких усилий, потому что пользователи мобильных устройств более нетерпеливы, чем пользователи любых других платформ. Пользователи должны точно знать, что смогут взаимодействовать с приложением и будут иметь возможность делать это в спешке. Как следствие вы должны помнить, что соглашения для платформы разрабатывались специального для того, чтобы пользоваться преимуществами усвоенного поведения.

Коль скоро вы взяли эту книгу в руки, я, вероятно, не должен долго объяснять, как важен дизайн пользовательских интерфейсов. Вы и так знаете это. И у вас уже есть желание приступить к созданию удобных и привлекательных приложений.

Эта книга послужит учебным пособием по проектированию и реализации пользовательских интерфейсов, а также удобным справочником, к которому вы будете возвращаться снова и снова. Вы узнаете, как общаться с дизайнерами и разработчиками на их языке, чтобы способствовать улучшению приложений. Вы сможете создавать приложения с визуально привлекательным интерфейсом, легко поддающимся изменениям даже в самые последние минуты перед выпуском.

В конце концов, и разработчики, и дизайнеры желают, чтобы их приложения пользовались популярностью, и я буду рад рассказать вам, как этого добиться.

– Ян Клифтон (Ian G. Clifton)

Благодарности

Может показаться, что второе издание книги дается проще ее автору, но представьте, что вам требуется переписать 90% текста, потому что технологии и направления проектирования ушли далеко вперед, и вы поймете, что без помощи других вам придется очень трудно. Ответственный редактор Лаура Левин (Laura Lewin) еще раз помогла мне не выбиться из графика, даже когда я решил реорганизовать книгу и чересчур углубился в это занятие. Оливия Бесижио (Olivia Basegio), помощник редактора, постоянно следила за изменениями и попутно публиковала фрагменты рукописи, чтобы заинтересованные читатели могли знакомиться с книгой по мере ее развития. Сонглин Кью (Songlin Qiu) снова сыграла роль редактора-консультанта и взяла на себя тяжкий труд привести к удобочитаемому виду главы, написанные ночью. Я также бескрайне признателен моим техническим рецензентам: Адаму Портеру (Adam Porter), Кэмерону Банга (Cameron Banga) и Джошуа Джеймисону (Joshua Jamison), чьи отзывы помогли существенно улучшить эту книгу.

Об авторе

Ян Клифтон (Ian G. Clifton) – профессиональный разработчик приложений для Android, проектировщик пользовательских интерфейсов и автор книг. Работал со многими разработчиками и дизайнерами и руководил подразделением Android, создавшим такие известные приложения, как Saga, CNET News, CBS News и др.

Любовь к технологиям, искусству и своей работе вела Яна разными путями. Помимо приложений для Android, он также занимается разработкой настольных и веб-приложений. В свое время он служил в военно-воздушных силах США специалистом по спутниковым широкополосным системам телеметрии и тогда же создал немало художественных произведений ручкой, углем, кистью и камерой.

Вы можете последовать за Яном в Twitter (http://twitter.com/IanGClifton) и познакомиться с его мыслями о разработке мобильных приложений в блоге http://blog.iangclifton.com. Он также публикует видеоматериалы в серии «The Essentials of Android Application Development LiveLessons, 2nd Edition», доступные по адресу: http://goo.gl/4jr2j0.

Введение

Кому адресована эта книга

Эта книга адресована в первую очередь разработчикам для Android, желающим лучше понять, как создавать пользовательские интерфейсы (ПИ) в Android. Чтобы сосредоточить все внимание на наиболее важных темах проектирования ПИ для Android, в этой книге предполагается, что вы уже имеете элементарные представления об Android. То есть если вы еще не написали своего первого приложения «Hello, World» для Android или не подготовили компьютера для разработки, вы должны сделать это, прежде чем продолжить чтение (в этом вам поможет сайт для Android-разработчиков: http://developer.android.com/training/basics/firstapp/index.html).

Большинство разработчиков не имеет или имеет сильно ограниченный опыт дизайна пользовательских интерфейсов, поэтому в этой книге не делается никаких предположений о ваших знаниях в этой области. Всякий раз, когда мы будем сталкиваться с решением, важным с точки зрения дизайна, таким как выбор цвета для того или иного элемента, мы детально опишем, что лежит в основе такого решения, чтобы вы могли осознанно подходить к выработке своих решений.

Организация книги

Эта книга разбита на несколько частей. В первой части «Основы пользовательских интерфейсов» дается обзор пользовательского интерфейса Android и основных направлений проектирования, прежде чем погрузиться в изучение классов, используемых для создания интерфейсов. Она также охватывает приемы использования графики и ресурсов. Во второй части «Полный процесс проектирования и разработки» описываются этапы разработки приложений, от основных идей и целей, через макеты и прототипы, до разработки законченных приложений, включая эффективные схемы размещения элементов, анимационные эффекты и многое другое. В третьей части «Дополнительные вопросы создания пользовательских интерфейсов» исследуются намного более сложные темы, включая диагностику причин низкой производительности пользовательских интерфейсов с помощью Systrace и создание нестандартных виджетов, поддерживающих рисование графических изображений, прокрутку и сохранение информации о состоянии.

В этой книге имеются два приложения. В первом описываются ресурсы Google Play (и охватываются отличия от Amazon Appstore, которые необходимо знать, чтобы подготовить приложение к размещению в обоих электронных магазинах) и приемы создания пиктограмм, представляющих приложения. Второе охватывает общие задачи, связанные с пользовательскими интерфейсами, не укладывающиеся ни в одну из глав в книге (такие как пользовательские атрибуты виджетов).

На протяжении всей книги основной акцент делается на простоте и ясности реализации. Не нужно бояться сложных тем, таких как создание матриц для трехмерных

преобразований в OpenGL; мы не будем затрагивать их, вместо этого основное внимание уделим созданию гладких анимационных эффектов, добавлению поддержки режимов PorterDuff в виджеты и эффективной обработке событий сенсорного экрана. Мы максимально упростим используемые математические приемы. Кроме того, многочисленные иллюстрации сделают сложные примеры более ясными и практичными.

Как читать эту книгу

Эта книга начинается с обширного обзора перед погружением в специальные и сложные темы. Она задумана так, чтобы читать ее по порядку, но вместе с тем она с успехом может использоваться как справочник. Даже если вы опытный разработчик, читайте главы по порядку, потому что в них охватываются самые разные темы; но при желании вы можете сразу переходить к темам, которые представляют для вас наибольший интерес. Например, если вы действительно хотите сосредоточиться на создании собственных виджетов, вы можете сразу перейти к главе 12 «Разработка собственных виджетов».

Веб-сайт книги

Исходный код примеров из книги можно найти по адресу: https://github.com/IanGClifton/auid2, а также на сайте издательства: http://www.informit.com/store/android-user-interface-design-implementing-material-9780134191409. По этим адресам вы можете создать полную копию репозитория, загрузить ZIP-файл со всеми примерами или просматривать отдельные файлы.

Типографские соглашения в книге

В этой книге используются типографские соглашения, обычные для книг по программированию. Элементы программного кода, такие как имена классов или ключевые слова, оформляются `моноширинным шрифтом`.

Иногда слишком длинные строки с программным кодом, не умещающиеся по ширине страницы, содержат знак стрелки (➥), показывающий, что строка продолжается ниже.

Иногда вам будут встречаться врезки с полезной информацией, неуместной в основном тексте.

Так оформляются короткие примечания с дополнительной информацией по обсуждаемой теме, которая может вам пригодиться.

Так выглядят советы по обсуждаемой теме.

Потенциальные проблемы, связанные с безопасностью или вероятностью потери данных. Предупреждения, оформленные так, призваны привлечь ваше внимание к потенциальным проблемам, с которыми вы можете столкнуться.

Часть I

ОСНОВЫ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ

Глава 1

Пользовательский интерфейс Android и материальный дизайн

Считаю полезным дать обзор пользовательского интерфейса (ПИ) ОС Android, поэтому мы начнем с него. Здесь вы узнаете краткую историю дизайна пользовательского интерфейса в Android и увидите, как развивалась идея материального дизайна, прежде чем оформилась в виде нескольких принципов проектирования. Вы познакомитесь также с некоторыми высокоуровневыми элементами дизайна Android и изменениями, произошедшими в дизайне Android с ростом популярности мобильной операционной системы.

Краткая история развития дизайна Android

Операционная система Android начиналась как чисто технический проект, в рамках которого был проделан удивительно большой объем работы, чтобы превратить ее в платформу, на которой могли бы работать самые разные устройства, имеющие свои технические особенности. Эта основа позволила ОС Android обслуживать многие типы устройств ввода (трекболы, джойстики, выдвижные клавиатуры, сенсорные экраны и т. д.). Она также позволила ОС Android сфокусироваться на масштабируемом дизайне, гораздо более близко напоминающем «резиновую» веб-верстку, чем типичный современный мобильный дизайн. Программный код, составляющий основу системы, мог даже выполняться на устройствах, не имеющих графического процессора (Graphics Processing Unit, GPU). К сожалению, все это означало также, что первоначальный дизайн для Android был чересчур скупым и ориентированным на самые слабые устройства, поэтому разного рода украшения и анимация были редким явлением. Цвета были мягкими и часто плохо согласующимися между собой, а визуальная организация опиралась на разработки прошлых лет и медленно двигалась вперед.

В 2010 году компания Google наняла Матиаса Дуарте (Matias Duarte, более известного своей работой над WebOS) на должность старшего директора по пользо-

вательскому интерфейсу Android, что ясно показало, насколько серьезно относятся в Google к пользовательскому интерфейсу и визуальному дизайну. Бета-версия Android была выпущена задолго до этого, в 2007 году, поэтому перед Матиасом с коллегами был непочатый край работы. Как перейти от очень функционального, но визуально скудного пользовательского интерфейса к интерфейсу, усиливающему функциональность продуманным и качественным дизайном?

Примерно год спустя вышел первый планшетный компьютер на ОС Android версии «Honeycomb» (Android 3.0). Эти планшетные компьютеры дали компании Google настоящую возможность экспериментировать с пользовательским интерфейсом, потому что прежде не существовало версии Android для планшетных компьютеров и, соответственно, у пользователей не было серьезных требований. С радикально новой темой «Holo» эти планшетные компьютеры стали существенным шагом, уводящим от предыдущих стилей оформления Android.

В конце 2011 года компания Google представила версию Android 4.0 «Ice Cream Sandwich», которая показала, насколько удалось усовершенствовать оформление «Honeycomb» для планшетных компьютеров, скрыть некоторый «технический привкус» и улучшить восприятие пользователями. Было устранено деление планшет/телефон, и платформа получилась более единообразной. Изменился даже шрифт по умолчанию – вместо шрифтов Droid стали использоваться шрифты Roboto, имеющие более привлекательный внешний вид. Не важно, являетесь вы поклонником рубленых шрифтов без засечек или нет, вы не сможете не оценить то внимание к деталям (о чем говорит смена шрифта), которое было проявлено.

В 2012 году на конференции Google I/O была представлена обновленная версия Android 4.1 «Jelly Bean». Основной целью этого выпуска было увеличение удобства использования – «Jelly Bean» придала Android намного более привлекательный дизайн. Технология «Project Butter» еще больше улучшила адаптивность пользовательского интерфейса и привнесла в графическую подсистему такие усовершенствования, как тройная буферизация. Обновились даже компоненты Android, к которым разработчики не прикасались годами, такие как всплывающие уведомления. Спустя всего несколько месяцев вышла версия Android 4.2. В ней появился многопользовательский режим, а также поддержка технологий «Daydream» (по сути – хранители экрана, оформленные в виде приложений и обладающие интерактивными функциями), «Photo Sphere» (обеспечивает создание фотопанорам, охватывающих все 360 градусов) и беспроводного зеркалирования. Последовавшая за ней версия Android 4.3 добавила еще больше возможностей, включая поддержку OpenGL ES 3.0. И версия Android 4.4 завершила линейку 4.x, добавив существенные усовершенствования, позволяющие системе Android работать на устройствах с небольшим объемом ОЗУ, до 512 Мбайт.

Затем разработчики внесли радикальные изменения, вылившиеся в создание версии Android 5.0, «Lollipop», в которой внедрили технологию материального дизайна. Посмотрите на рис. 1.1, как менялся дизайн домашнего экрана между версиями Android 1.6 и Android 5.0.

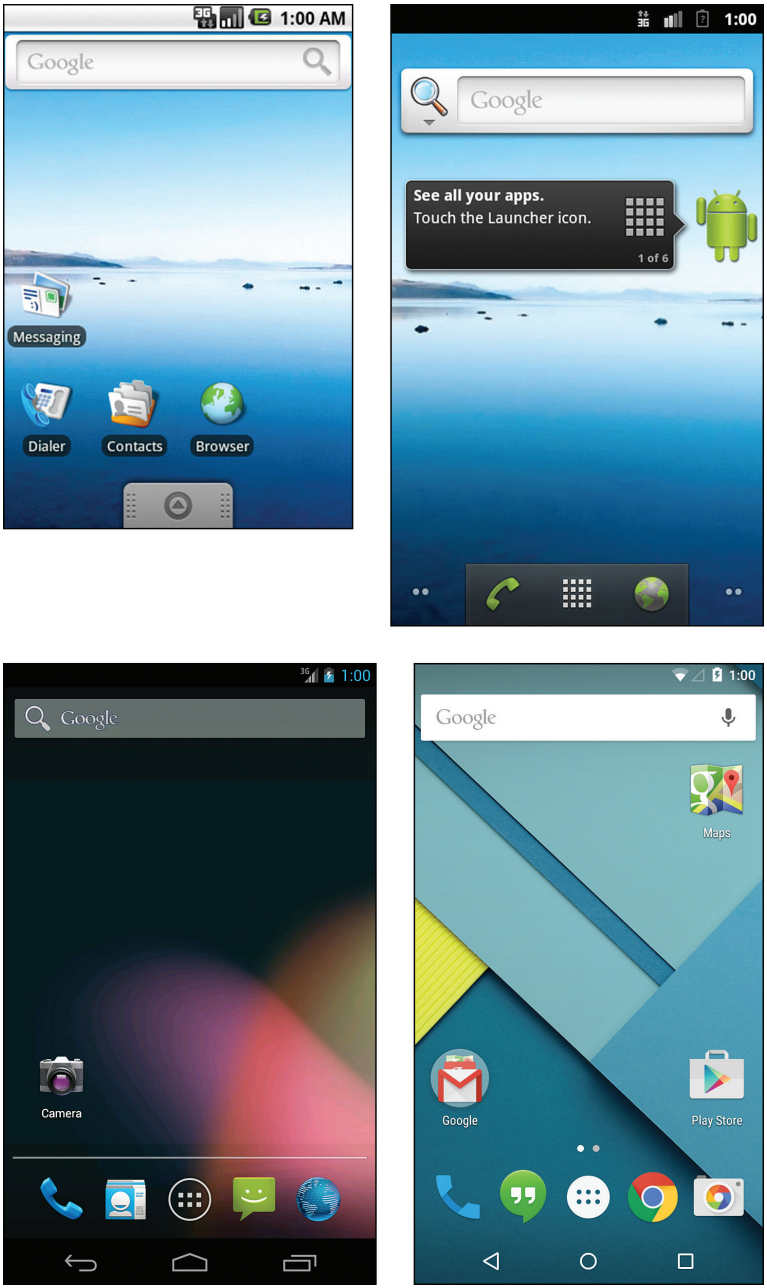

Рис. 1.1 ❖ Домашний экран в версиях Android 1.6 (вверху слева), 2.3 (вверху справа), 4.2 (внизу слева) и 5.0 (внизу справа)

Материальный дизайн

Несмотря на тесную связь между Android 5.0 и материальным дизайном, материальный дизайн (Material Design) – это не только язык визуальных образов для Android 5.0 и выше, но также набор принципов проектирования, применимых к любым типам устройств и произвольным версиям программного обеспечения. То есть принципы материального дизайна должны применяться также к старым версиям Android и даже к веб-приложениям. Google так описывает эту концепцию:

> Метафора материального мира – это теория, обобщающая рационализацию пространства и систему движений. Материальный дизайн основывается на ощущениях, возникающих при работе с бумагой и чернилами, но более развит технологически и открыт для воображения и магии.

Если вы рассуждаете как большинство далеких от профессии дизайнера, первой вашей реакцией наверняка будет вопрос: «Что?» Это так просто – отвергнуть подобные заявления, посчитав их обычным «дизайнерским трепом», и не оценить их смысл, но у дизайнеров имеются свои руководящие принципы, цель которых – повысить качество результатов их труда, так же как у обычных разработчиков (вспомните, например, принцип: «Не повторяйся»). В описании выше говорится, что материальный дизайн основан на ощущениях, возникающих при работе с бумагой и чернилами, но он не ограничивается только этими элементами физического мира.

Основные понятия

В мире материального дизайна под бумагой понимается прежде всего поверхность, на которой находится все остальное. Она может растягиваться и сжиматься, в отличие от листа бумаги в реальном мире. То есть лист бумаги может появиться в центре экрана, увеличиваясь из точки до размеров экрана и даже изменяя форму. Лист бумаги всегда появляется с некоторым эффектом (возникает из ничего, постепенно увеличиваясь в размерах или выдвигаясь из-за края экрана) – он никогда не появляется внезапно. Листы бумаги могут надвигаться друг на друга, образуя стопку, делиться пополам, объединяться. Листы бумаги не могут проходить сквозь друг друга, но один лист может скользить поверх другого.

Тот факт, что один лист бумаги может скользить поверх другого, означает, что материальный дизайн существует в трехмерном окружении. Третье измерение, ось Z, определяет, как близко к поверхности экрана находится объект, как накладываются тени, отбрасываемые объектами, и как объекты перекрывают друг друга. Материальный дизайн придает оси Z очень ограниченное значение, в том смысле, что не позволяет отображать сколько-нибудь существенную глубину (то есть не позволяет показать, насколько толстой получилась стопка из листов бумаги на экране). Листы бумаги всегда имеют толщину в один пиксель, не зависящий

от разрешения экрана (идея пикселей, не зависящих от разрешения, – Density-Independent Pixels, или dip, – рассматривается в главе 4 «Добавление графики и ресурсов в приложения», а пока рассматривайте dip как некую единицу измерения), то есть никогда не изгибаются и не скручиваются.

Устройства не имеют трехмерных экранов (пока), поэтому ощущение глубины создается традиционными приемами перспективного отображения, заслонения и затенения. Как показано на рис. 1.2, объект, находящийся ближе к поверхности, выглядит больше, чем тот же объект, удаленный от экрана. Объект, находящийся выше, заслоняет объекты, находящиеся под ним, как показано на рис. 1.3. Ближний объект отбрасывает тень, как показано на рис. 1.4. Объединение этих простых принципов позволяет получить более четкое ощущение глубины, как показано на рис. 1.5.

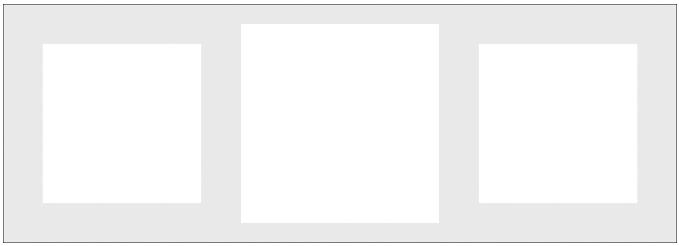

Рис. 1.2 ❖ Под перспективой понимается, что более близкие объекты выглядят больше, но одной перспективы недостаточно, так как не ясно, действительно ли один объект ближе других или он просто больше в размерах

Тени в материальном дизайне создаются светом от двух источников: основным и рассеянным. Представьте, что вы фотографируете кого-то своим телефоном, вспышка даст основной свет, а все остальное освещение даст рассеянный свет. Основной свет создает резкие тени в одном направлении. Рассеянный свет создает размытые тени во всех направлениях. Основной свет исходит из центра сверху над экраном и создает тени, отбрасываемые листами бумаги вниз; кроме того, листы бумаги, находящиеся в нижней части экрана, отбрасывают более длинные тени, потому что свет падает на них под более острым углом. Такого рода детали визуального отображения остаются почти не замеченными, но увеличивают целостность восприятия искусственной среды и реалистичность отбрасываемых теней. Чтобы лучше понять, что это означает, взгляните на рис. 1.6, демонстрирующий описанные идеи на трехмерном изображении.

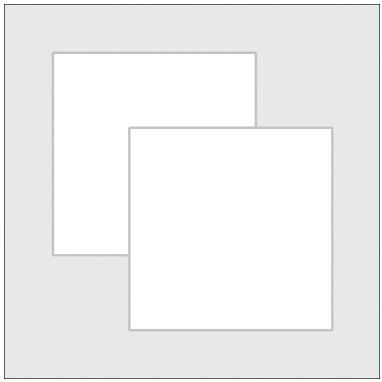

Рис. 1.3 ❖ Эффект заслонения одних объектов другими создает ощущение, что один объект находится перед другим, но это не позволяет судить, насколько один объект ближе к вам, чем другой

Рис. 1.4 ❖ Простая тень способна показать, какой объект находится ближе, но если размеры не изменяются (из-за перспективы), это может создавать ложные ощущения

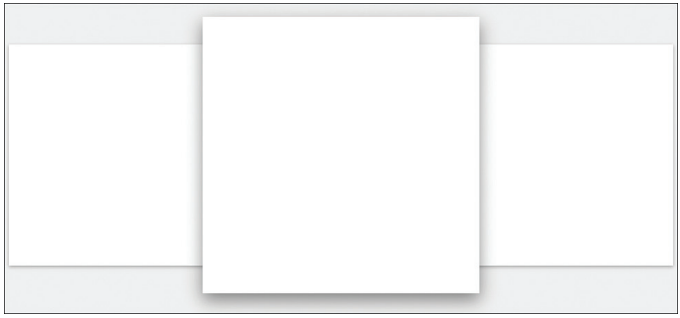

Рис. 1.5 ❖ Объединение всех перечисленных принципов создает более полное ощущение, что объект в середине находится ближе к поверхности экрана, даже если соседние объекты чуть-чуть «приподнять»

Помимо бумаги и чернил, в материальном дизайне существуют другие важные компоненты. Чернила определяют цвет текста, создаваемого на бумаге. Текст не имеет толщины. Он может перемещаться по листу бумаги (например, фотографию можно передвинуть от верхнего края листа к нижнему или увеличить в размерах), растягиваться и сжиматься, а также изменять цвет или форму. Текст может быть жирным. Многие приложения используют очень бедную, а порой просто черно-белую палитру цветов, тогда как материальный дизайн призывает использовать более обширные палитры. Подробнее о выборе цветов рассказывается в главе 7 «Визуальный дизайн».

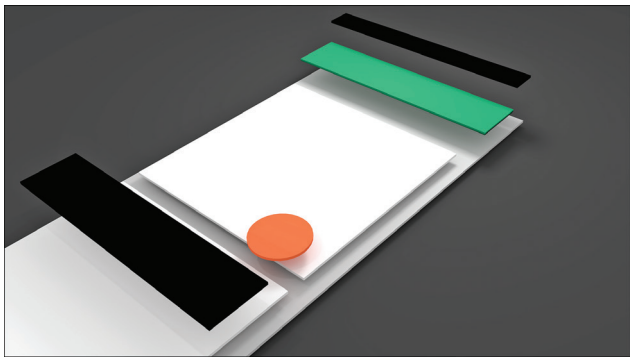

Рис. 1.6 ❖ Трехмерное изображение, созданное с применением принципов материального дизайна

Взаимодействия и анимация

Взаимодействия – один из важных аспектов дизайна приложений, который часто игнорируют. Что должно произойти, когда пользователь коснется этой кнопки? Разумеется, должен появиться новый экран, но как это должно выглядеть? Должна ли кнопка изменяться в размерах или изменять цвет? Должен ли новый экран выскользнуть из-за края или возникнуть из середины? Материальный дизайн уделяет взаимодействиям и анимации больше внимания, чем предшествовавшие ему руководства, что помогает придать приложениям более законченный вид.

События касаний в прошлом часто обозначались простым изменением цвета фона. Касание строки в списке вызывало немедленное изменение фона с белого на синий или оранжевый. Материальный дизайн рекомендует применять более естественные эффекты. Так, типичной реакцией на касание является появление ряби. Представьте, как камень падает в воду и появляются волны, разбегающиеся от точки падения; примерно так выглядит реакция интерактивного элемента с точки зрения материального дизайна. Кроме того, бумага тоже может реагировать. Например, лист бумаги мог бы устремиться вверх, как бы притягиваясь к пальцу. Это важное замечание, потому что оно противоречит традиционному дизайну, когда объекты, такие как кнопки, нажимаются вниз, имитируя поведение объектов физического мира (такой дизайн называется скевоморфным).

Анимационные эффекты должны быть плавными и при необходимости ускоряющимися или замедляющимися. Как автомобиль не начинает движение сразу с максимальной скоростью, когда вы нажимаете педаль газа, так и лист бумаги, ускользающий за границы экрана в ответ на ваше касание, не должен начинать движение сразу с максимальной скоростью. Каким бы маленьким ни был лист, он все же имеет некоторую массу. Изменение скорости воспроизведения анимации в зависимости от степени ее завершенности (от 0% до 100%) называют интерполяцией. Выбор закона изменения скорости (ускорения, замедления или какого-то другого) может показаться несущественной деталью, но это один из маленьких фрагментов мозаики, которые все вместе дают удивительные картины. Тем более что это легко реализуется (более подробно об организации анимационных эффектов рассказывается в главе 9 «Украшение анимацией»).

Еще одно важное замечание об анимации – ее назначение не в том, чтобы вызвать у пользователя восклицание: «Ух ты!» Она не должна отвлекать пользователя или просто придавать интересный вид обычным вещам. Ее цель – помочь пользователю ощутить факт перехода от того, что есть на экране, к тому, что вот-вот появится. Анимационные эффекты приковывают глаза пользователя к важным элементам и объясняют, что происходит изменение. Если все сделано правильно, такие переходы доставляют некоторое эстетическое наслаждение и способны даже подтолкнуть пользователя выполнять их снова и снова, только чтобы увидеть анимационный эффект.

Типографика

Одновременно с выходом версии Android 4.0 компания Google представила новое семейство шрифтов Roboto. Гарнитура Roboto была разработана специально для мобильных устройств, чтобы выглядеть максимально четко с разными разрешениями. В Android 5.0 шрифты Roboto были доработаны – в них были исправлены заметные оплошности, улучшены некоторые символы (такие как заглавная буква «R») и точки в знаках пунктуации и буквах (например, в буквах «i» и «j») сделаны круглыми, а не квадратными. Некоторые из этих изменений можно видеть на рис. 1.7.

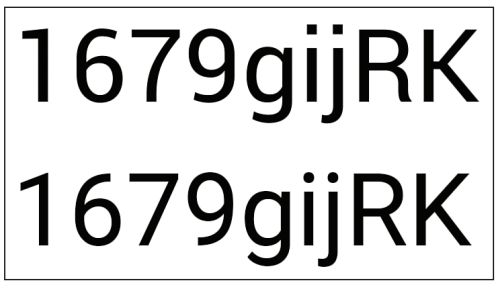

Рис. 1.7 ❖ Некоторые наиболее заметные отличия между исходным шрифтом Roboto (вверху) и его улучшенной версией (внизу)

Еще один шрифт, который используется в материальном дизайне и о существовании которого вы должны знать, – это Noto. В действительности этот шрифт используется по умолчанию в Chrome OS, а также в Android, когда применяются региональные настройки, не поддерживаемые Roboto. Шрифты Roboto поддерживают латинские, греческие и кириллические символы (широко используемые в Восточной Европе, а также в Северной и частично в Центральной Азии), для всех остальных алфавитов используется Noto. Более подробно типографика рассматривается в главе 7 «Визуальный дизайн».

Единицы измерения и выравнивание

В материальном дизайне используется сетка с размером 4 dp (четыре пикселя, не зависящих от разрешения). Это означает, что каждый элемент на экране выравнивается по этой сетке. Например, левая и правая стороны экрана обычно выравниваются по границе 16 dp, поэтому ярлыки и текст в списке должны начинаться с отступом 16 dp от левого края. Когда слева от элементов списка выводятся изображения (миниатюры или пиктограммы), материальный дизайн рекомендует выводить текст с отступом 72 dp от левого края экрана.

Эти правила размещения подробно описываются в главе 5 «Начало создания нового приложения». Кроме того, они используются во всех примерах в этой книге.

Веб-сайт о дизайне Android

Мы лишь вскользь коснулись материального дизайна, но это далеко не все. В этой книге вы узнаете еще массу подробностей о материальном дизайне, о факторах, влияющих на принимаемые решения, и как эти решения реализуются, но вы можете также заглянуть на сайт, посвященный вопросам дизайна в Android (http://developer.android.com/design/). На этом сайте вы найдете все подробности, касающиеся применения принципов материального дизайна к конкретным компонентам Android. Загляните также в спецификацию с описанием материального дизайна на сайте Google (http://www.google.com/design/spec/material-design/)[1], где вы найдете замечательные видеоролики, демонстрирующие некоторые аспекты, которые трудно объяснить одними словами, такие как анимация взаимодействий.

Не нужно стараться запомнить все, что вы увидите на этих сайтах, пока не начали проектировать свое приложение или разбираться в реализации приложения, написанного кем-то другим, но обязательно загляните на них, чтобы получить представление об имеющихся возможностях, чтобы потом вернуться туда, когда появятся конкретные вопросы. Имейте в виду, что на сайте разработчиков Android (в том числе и в разделе, посвященном дизайну) имеется огромное число страниц с описаниями и руководствами по различным темам. И это очень хорошо, потому что здесь вы найдете ответы почти на все свои вопросы, единственный недоста-

[1] Перевод можно найти по адресу: http://css-live.ru/articles/materialnyj-dizajn-vvedenie.html. – *Прим. перев.*

ток – эти страницы не всегда содержат самую актуальную информацию. Если вам встретится страница, которая, по вашему мнению, не согласуется с принципами материального дизайна, она почти наверняка содержит устаревшую информацию (такие страницы обычно можно опознать по скриншотам, демонстрирующим устаревшую версию Android, такую как 4.x, с синей панелью состояния из темы Holo), тогда лучше будет обратиться с возникшим вопросом к спецификации с описанием материального дизайна. Также следует отметить, что руководства по материальному дизайну регулярно обновляются, в них добавляются примеры и подробности, поясняющие принципы.

Основные принципы

Невозможно составить контрольный список, выполнив все пункты которого, можно было бы сказать, что ваше приложение полностью соответствует всем требованиям, но следование руководящим принципам поможет достичь желаемого результата. Начните с выяснения целей ваших пользователей и определите точно, что должно делать приложение. Возможно, вы удивитесь, узнав, что многие разработчики не имеют ясного представления о целях пользователей, приступая к созданию своих приложений, что, безусловно, отражается на их дизайне. О целях пользователей и целях продукта мы подробно поговорим в главе 5 «Начало создания нового приложения», но перед этим важно познакомиться с основными принципами.

Делайте что-то одно, но делайте это хорошо

Если у вас когда-нибудь появится желание написать посредственное приложение, лучший способ добиться этого – постараться заставить его делать все. Чем выше специализация приложения, тем проще убедиться, что оно решает поставленные перед ним задачи и хорошо справляется с ними. Приступая к работе над новым приложением, составьте список всего, что оно должно делать. Затем вычеркивайте из него наименее важные пункты, пока в списке не останется только то, без чего никак не обойтись. Позднее вы сможете добавить недостающие возможности, но вы не сможете вернуть ясность целей приложению, которое следует принципу «все в одном». Сужение фокуса также поможет гарантировать ясность ожидаемых возможностей.

Вы будете полностью готовы приступить к разработке, когда сможете ответить на вопрос «Зачем пользователям нужно мое приложение?» – без использования союзов «и» и «или» и добавочных предложений. Ниже приводятся два варианта ответов:

- **хороший:** «С помощью приложения пользователи будут создавать короткие заметки для себя»;
- **плохой:** «С помощью приложения пользователи будут создавать короткие заметки для себя, просматривать их и делиться ими с другими пользователями через Twitter».

Да, возможность просматривать заметки может быть важной частью приложения, но гораздо важнее возможность создавать заметки. Определив наиболее важный аспект, становится очевидно, что должна иметься возможность создавать новые заметки одним касанием в области начального экрана. Это может быть плавающая кнопка (Floating Action Button, FAB). Конечно, вы можете взяться за создание приложения, в котором важнейшим аспектом является организация заметок; в этом случае в центре вашего внимания будут такие функции, как обзор, сортировка и поиск.

Рассмотрим в качестве примера приложение Contacts (Контакты, см. рис. 1.8). Это простой список людей с картинками и различными данными. Из этого списка можно позвонить или послать электронное письмо, но все эти действия выполняются за пределами приложения.

Весьма заманчиво попытаться реализовать в приложении как можно больше функций, например редактирование изображений в клиенте электронной почты перед вложением в письмо, но вы должны начать с чего-то одного и, только добившись безупречного выполнения этой функции, переходить к чему-то другому. Если упомянутый клиент электронной почты плохо будет справляться со своими прямыми обязанностями – отправкой электронных писем, никто не захочет использовать его для редактирования изображений.

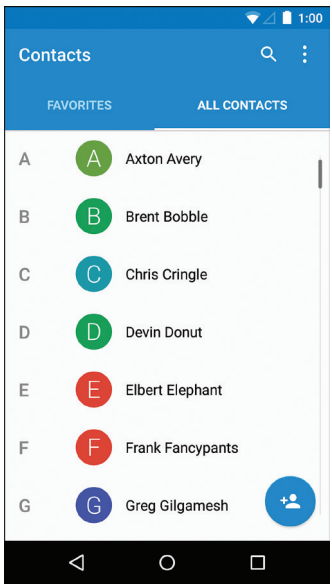

Рис. 1.8 ❖ Приложение Contacts (Контакты) – отличный пример приложения с единственной и ясной целью

Иногда приложение действительно должно преследовать несколько целей, потому что они слишком тесно переплетены, чтобы их можно было разделить, или

требования к приложению определяются не вами. В таких случаях следует разбить функции ясным и понятным для пользователя способом. Возьмем для примера приложение Gallery (Галерея). Оно позволяет пользователям просматривать свои фотографии и управлять ими. Оно не дает возможности рисовать картинки или управлять файлами в файловой системе, то есть имеет ясные и очевидные цели. В приложении имеется ссылка для быстрого перехода к приложению Camera (Камера), которое также имеет свой ярлык, позволяющий пользователю запускать это приложение из панели запуска. Концептуально приложение Camera (Камера) просто снимает фотографии и видеоролики. Каким бы сложным ни был код, если он делает свою работу хорошо, пользователю не придется думать о нем. Более того, пользователь вообще не должен знать, что приложения Camera (Камера) и Gallery (Галерея) являются составными частями одного и того же приложения Android (некоторые производители создают свои, отдельные приложения для выполнения этих задач, и Google также распространяет свои заменители). Если пользователь пожелает посмотреть фотографии, он запускает приложение Gallery (Галерея). Если он захочет сделать фотографию, он запускает приложение Camera (Камера).

Будьте вежливы

Только потому, что ваше приложение выполняет свою работу лучше всех, это не означает, что вы можете не задумываться об удобстве пользователя. Одна из лучших особенностей Android заключается в том, что приложения в действительности являются всего лишь компонентами полного пользовательского интерфейса. Ваше приложение должно обрабатывать «намерения» – экземпляры класса `Intent`, которые Android использует, чтобы понять, что пытается сделать пользователь, и отыскать подходящее приложение для этого. Ваше приложение обслуживает ссылки на данный конкретный сайт? Реализуйте в нем обработку ссылок на этот сайт. Позволяет редактировать изображения? Обрабатывайте все экземпляры `Intent` для работы с изображениями.

Не тратьте понапрасну времени, добавляя поддержку служб обмена информацией, таких как Twitter и Facebook. Иначе вам придется задумываться над поддержкой всех других служб, таких как Google Plus. Вашим пользователям нужна служба Google Plus? Почему вы должны заботиться об этом, почему бы просто не дать пользователям возможность выбрать приложение, которое они хотели бы использовать для общения? Если они используют Google Plus, они сами установят это приложение. Всего лишь парой строк кода вы можете вывести диалог (рис. 1.9) и обеспечить поддержку намного большего количества служб, чем можно запрограммировать вручную. Разрабатывая специализированный код для взаимодействия с конкретной службой, вы фактически ликвидируете возможность поддержки всех остальных служб, которые могут предпочитать ваши пользователи. Но и это еще не все. Поддержка конкретной службы требует дополнительных усилий на обновление соответствующих пакетов SDK, устранение ошибок и переделку в ответ на изменения в API.

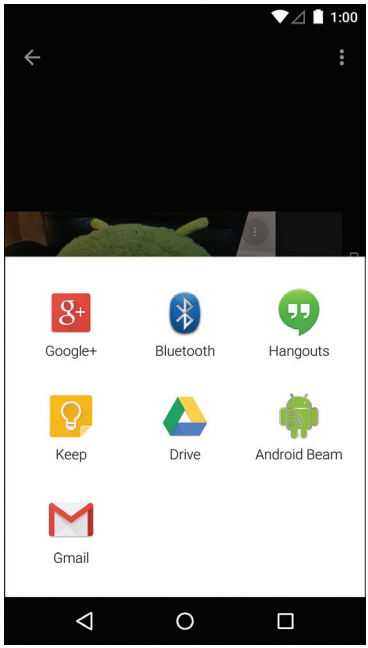

Рис. 1.9 ❖ Пользовательский интерфейс по умолчанию, позволяющий выбрать приложение

Время, затрачиваемое на реализацию взаимодействий со сторонними службами, – это время, которое можно было бы потратить на улучшение приложения. Почему вы должны тратить недели на реализацию инструментов общения, если эту задачу можно переложить на приложения, которые нравятся пользователям? Вне всяких сомнений, какого бы клиента Twitter не выбрал пользователь, разработчик потратил на него больше времени, чем вы можете выделить на реализацию одной из множества особенностей. В своих программах вы используете готовые Java-классы, так почему бы не использовать готовые приложения?

Это касается не только инструментов обмена информацией. Вы можете построить великолепное приложение-будильник, которое, с точки зрения пользователя, просто запускает другие приложения. Оно может выглядеть очень просто, но как только вы объедините с системой, позволяющей приложению запускать другое приложение для проигрывания музыки, отображения погоды или новостей, ваше простое и ясное приложение станет по-настоящему ценным.

Образы, образы, образы

Один из основных недостатков мобильных приложений состоит в том, что часто у вас имеется масса информации, которую требуется передать, но экран слишком маленький для этого. Хуже того, пользователь нередко бросает на приложение

лишь мимолетный взгляд. Это означает, что информация на экране должна восприниматься легко и содержать все необходимое. Используйте короткий текст в заголовках, подсказках и диалогах. Убедитесь, что надписи на кнопках ясно сообщают о выполняемых ими действиях, например «Сохранить файл» вместо «Ok», чтобы назначение кнопки распознавалось сразу и однозначно.

Используйте изображения, чтобы быстро донести смысл. Используйте непротиворечивые образы, чтобы не вводить пользователей в заблуждение. Всегда отображайте возможные состояния. Все, с чем может взаимодействовать пользователь, должно иметь как минимум нормальное состояние, нажатое состояние и состояние в фокусе ввода. Нажатое состояние отображается, когда пользователь активно касается видимого элемента, исключение этого состояния означает, что пользователь будет неуверен, с какими элементами можно взаимодействовать. Состояние в фокусе ввода отображается, когда элемент выбирается навигационными клавишами или другим способом, чтобы пользователь знал, какой элемент будет нажат. Это состояние может также отображаться в сенсорном режиме, чтобы, например, подсветить поле ввода и показать пользователю, куда будет вводиться текст. Без состояния фокуса ввода пользователь не сможет эффективно использовать альтернативные средства навигации.

Образное представление не ограничивается одними только изображениями. Пользователи приложений быстро учатся распознавать вновь и вновь встречающиеся надписи, например заголовки. Если какой-то из разделов приложения имеет заголовок «Похожие изображения», всегда оформленный одним и тем же шрифтом в одном и том же стиле, пользователи научатся распознавать этот текст по его форме, даже не читая.

Простое, но мощное

Люди субъективны. А люди, использующие приложение впервые, субъективны вдвойне, поэтому очень важно, чтобы ваше приложение было простым в использовании. Основные функции должны быть ясны и очевидны – они должны быть увязаны с легко распознаваемыми образами. Если, зайдя в приложение для создания заметок, вы увидите большой знак «плюс», вы сразу догадаетесь, что эта кнопка отвечает за создание новой заметки. Мощность такой кнопки заключается в том, что после нажатия она заполняет некоторые метаданные, чтобы пользователю не приходилось задумываться о них (по крайней мере, в данный момент), такие как дата создания заметки или местоположение пользователя в этот момент. После сохранения заметки приложение может просканировать ее в поисках важных слов или имен, присутствующих в списке контактов. Благодаря этому приложение сможет предоставить полезные функции, такие как поиск всех заметок за прошлый месяц, где упоминается имя Сюзанна, не требуя от пользователя рассматривать такую потребность во время создания заметки.

Если ваше приложение реализует фильтры для обработки фотографий, недостаточно просто использовать подписи «увеличить контраст» или «убрать крас-

ный канал». Добавьте миниатюру предварительного просмотра, чтобы пользователь мог видеть и понимать действие той или иной кнопки (см. рис. 1.10). Когда пользователь прокрутит список новостей до конца, желательно, чтобы приложение автоматически извлекло следующую порцию и вставило ее в конец списка. Простые особенности, как описанные выше, интуитивно понятны и дадут вашим пользователям ощущение мощности.

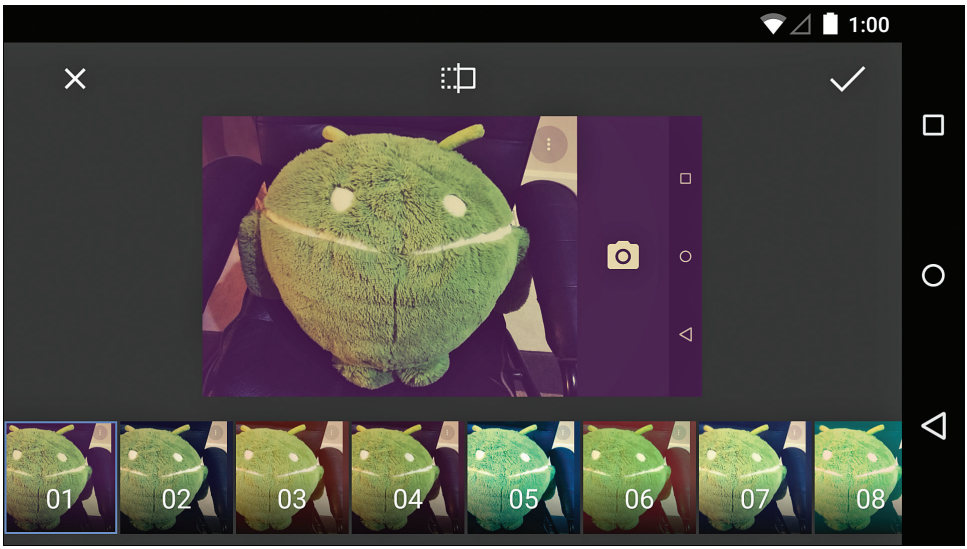

Рис. 1.10 ❖ Каждая функция обработки изображений обозначена в нижнем ряду миниатюрой, иллюстрирующей ее действие

И последнее: чтобы ваше приложение могло считаться простым и мощным, помните, что пользователь всегда прав, даже если он ошибается. Когда пользователь нажимает кнопку «удалить», в 99% случаев он делает это осознанно. Не нужно спрашивать его: «Вы действительно хотите сделать это?» Исходите из предположения, что пользователь действительно хотел выполнить данную операцию, но добавьте простую возможность ее отмены. Не затрудняйте доступ к функциям, пытаясь уберечь пользователей от ошибок. Наоборот, упростите доступ к функциям, в том числе и к функции отмены, чтобы пользователь мог без опаски заняться исследованием вашего приложения. Примером приложения, которое делает это очень хорошо, может служить Gmail. Удалив электронное письмо, пользователь имеет возможность восстановить его. Приложение не спрашивает, действительно ли он хочет удалить письмо, потому что такое поведение мешает появлению благоприятных впечатлений. Разумеется, если отменить какую-то операцию не представляется возможным (как, например, форматирование диска), тогда диалог подтверждения намерений действительно необходим.

Единообразие с платформой

Когда возникают какие-то сомнения, следуйте типичным ожиданиям пользователей, работающих с этой платформой. Даже если у вас нет сомнений, все равно следуйте типичным ожиданиям пользователей, работающих с этой платформой. Иными словами, приложение не должно делать что-то иначе, чем это делается во встроенных приложениях, если на то нет очень и очень веских причин. Заявление «Мы хотим, чтобы наше приложение выглядело и действовало одинаково и в iOS, и в Android» – не лучшее оправдание. Пользователи Android-версии вашего приложения каждый день пользуются устройствами на Android; они редко берут в руки (если вообще берут) устройства на iOS (или на другой мобильной платформе). Пользователи разных платформ имеют разные ожидания. Не используйте в Android-версии приложения стили или черты поведения, заимствованные из другой платформы, это вызовет у ваших пользователей замешательство и разочарование.

Другие платформы могут требовать наличия навигационных кнопок, например «Назад» или «Выйти»; такие кнопки отсутствуют в приложениях для Android. Ваше приложение должно правильно реагировать на стандартную Android-кнопку «Назад», а добавление в интерфейс еще одного элемента, который делает то же самое, будет только вводить пользователя в заблуждение. В Android не требуется явно завершать приложение, потому что пользователь может сделать это нажатием кнопки «назад» или «домой». Ваша кнопка будет просто копировать один из этих сценариев или, что еще хуже, действительно завершать приложение и тем самым замедлять его запуск в следующий раз. Это не только будет способствовать появлению неблагоприятных впечатлений у пользователя, но и повлечет дополнительный расход заряда аккумулятора из-за чтения ресурсов с диска, которые в ином случае могли бы оставаться в памяти.

Заимствование стилей оформления из других платформ не только создает визуальный диссонанс, но и вызывает ощущение дискомфорта. Например, в Android используется свое изображение для ярлыка «поделиться», которое существенно отличается от аналогичного изображения в других платформах, таких как iOS и Windows Phone. Пользователей почти наверняка смутит значок из другой платформы. Пользуйтесь тем фактом, что пользователи уже изучили другие приложения в Android, следующие духу и стилю этой платформы.

Многим разработчикам для Android приходится иметь дело с руководителями или другими ответственными лицами, настаивающими на копировании таких аспектов приложений для iOS, как вывод экрана загрузки. Боритесь с этим! Это не только приводит к напрасной трате времени, отпущенного на разработку, но и ухудшает впечатления пользователей и порождает у них нежелание пользоваться таким приложением. Когда приложение искусственно создает 2-секундную задержку, чтобы показать экран загрузки, это немногим лучше, чем вывод рекламы в полный экран, а если конкурирующее приложение запускается за доли секунды, как вы думаете, какое из них выберет пользователь? Хуже того, при реализации экрана загрузки очень легко допустить ошибку, из-за которой приложение выве-

дет информацию уже после того, как пользователь переключился в другое приложение, потому что ему надоело ждать загрузки приложения. Теперь он будет чувствовать, что приложение ведет себя чересчур агрессивно, пытаясь силой заставить пользоваться им, а это еще больше увеличит вероятность, что пользователь удалит такое приложение.

Покорите пользователя

Одна из замечательных особенностей Android – пользователи изначально имеют богатый выбор. Рабочий-строитель может предпочесть устройство в жестком корпусе с физической клавиатурой, нежели более мощное устройство, но в тонком корпусе. Обладатели больших рук имеют возможность приобрести устройство с 6-дюймовым экраном. Android здорово упрощает поддержку таких разных пользователей. Если на других устройствах сложно реализовать поддержку портретной и альбомной ориентации, это не означает, что вы должны отказаться от этого в Android.

Под покорением подразумевается такая реализация приложения, которая не только позволит комфортно работать с устройством, но и потворствует привычкам пользователя. А как им потворствовать – решать вам. В простейшем случае можно дать пользователю возможность настраивать личные предпочтения. Ваше приложение предназначено для чтения книг? Тогда имеет смысл предусмотреть настройку, принудительно устанавливающую определенную ориентацию экрана, чтобы пользователю было удобно читать, например, лежа. Если в ночное время приложение отображает белый текст на черном фоне, но пользователь постоянно переключается в режим отображения черного текста на белом, ваше приложение может запоминать такие предпочтения самостоятельно.

Стандартные компоненты

Как бы вы ни прорабатывали будущий дизайн своего приложения – тщательно, стараясь учесть все, что только можно; быстро, на скорую руку, надеясь подправить его потом; или просто обмакнув лапки своего домашнего питомца в чернила и поставив его на большой лист бумаги, надеясь на чудо, – важно знать, какие стандартные компоненты существуют в Android.

Системные панели

В Android имеются две системные панели: панель состояния и панель навигации. Панель состояния (показана на рис. 1.11) находится в верхней части экрана и отображает ярлыки уведомлений слева и стандартные пиктограммы состояния телефона справа, такие как состояние аккумулятора и уровень сигнала. Панель навигации (показана на рис. 1.12) находится в нижней части экрана и отображает кнопки «Назад», «Домой» и «Обзор задач», если устройство не имеет физических кнопок. Приложения, создававшиеся для старых версий Android, также выводят в навигационной панели кнопку меню. Планшетные устройства с Android 3.x

(Honeycomb) имеют комбинированную панель, отображающую элементы состояния и навигации, но в версии Android 4.2 пользовательский интерфейс был значительно переработан, чтобы максимально приблизить его к телефонному, с панелью состояния вверху и панелью навигации внизу.

Рис. 1.11 ❖ Стандартная панель состояния в Android (вверху) и панель состояния, окрашенная приложением (внизу)

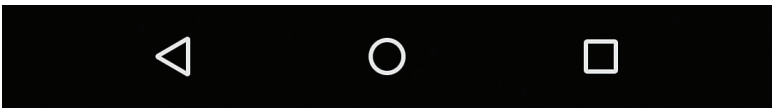

Рис. 1.12 ❖ Стандартная панель навигации в Android, обычно черного цвета, но может окрашиваться в другие цвета

Чаще всего они не требуют пристального внимания при проектировании. Приложение может скрывать панель навигации, но этого почти никогда не требуется. Системные панели допускается скрывать при проигрывании видео, а также когда они мешают благоприятному восприятию приложения. В «несерьезных» играх панель состояния обычно не скрывается (нет причин мешать пользователю видеть уведомления, пока он раскладывает пасьянс, ожидая какого-то события). Практически все приложения должны отображать панель состояния. Но как узнать, не является ли ваше приложение исключением? Попробуйте поработать с вашим приложением, когда панель состояния видна, и посмотрите, не противоречит ли это ожиданиям пользователей. Если нет, панель состояния должна отображаться. Примеры кода, демонстрирующие, как скрывать и отображать эти панели, вы найдете в приложении В «Справочник по типичным задачам».

Уведомления

С самого своего появления ОС Android создавалась как многозадачная система. Она не вынуждает пользователя замирать, наблюдая, как движется индикатор хода выполнения операции в приложении, которое загружает ресурсы, и не заставляет завершать приложение, чтобы попытаться обратиться к другой важной информации. Уведомления – замечательный механизм, который многие приложения почему-то не используют в своих интересах. Если ваше приложение выполняет какие-то операции в фоновом режиме, например синхронизирует список проигрывания, оно должно вывести уведомления по завершении операции. Если приложение имеет важную информацию для пользователя, например напоминания, настроенные самим пользователем, оно должно выводить их в виде уведомлений.

Версия Android 4.1 получила более совершенный механизм уведомлений, позволяющий сопровождать уведомления некоторыми действиями и отображать их с увеличенными размерами. Например, уведомление о получении электронного письма может отображать тему и отправителя, но при распахивании в нем может отображаться часть текста письма, а также кнопки для составления ответа или архивирования. В Android 5.0 дизайн уведомлений был усовершенствован еще больше: темную тему сменила светлая, появилась поддержка акцентирующих цветов и многое другое. Новый стиль уведомлений показан на рис. 1.13.

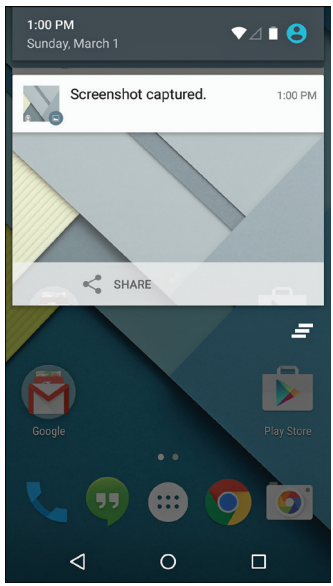

Рис. 1.13 ❖ Дизайн уведомлений в Android 5.0 был усовершенствован: теперь текст выводится темным шрифтом на светлом фоне

Дизайн уведомлений подробно рассматривается в главе 7 «Визуальный дизайн», а реализация (включая поддержку предыдущих версий Android) – в главе 8 «Реализация дизайна».

Панель приложения

Панель приложения – это панель инструментов, находящаяся в самом верху пользовательского интерфейса приложения, сразу под панелью состояния. Раньше она называлась панелью действий. Эта панель инструментов теперь так же доступна для реализации, как любой другой визуальный компонент. Такое усовершенствование фактически исправляет проблемы, характерные для старой панели действий (которая прежде была реализована как часть окна, что существенно ограничивало

возможности по ее оформлению), и привносит некоторые новые возможности по оформлению. Пример панели приложения показан на рис. 1.14.

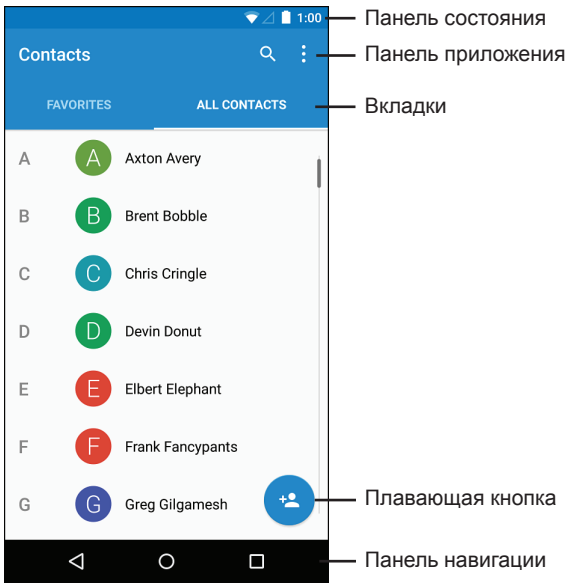

Рис. 1.14 ❖ Основные компоненты приложения для Android

Ярлык приложения теперь обычно не выводится в панели, но выводятся инструменты навигации слева (такие как кнопка «Вверх», позволяющая вернуться на один уровень выше в иерархии, или кнопка меню с изображением «бутерброда») и кнопки с контекстными операциями справа. Как и прежде, редко используемые операции и операции, не умещающиеся в панели приложения, помещаются в меню дополнительных действий с пиктограммой в виде трех точек, расположенных по вертикали. Вы можете сами включить логотип приложения в панель, но вообще делать это нежелательно. Вы можете также использовать собственный визуальный компонент в роли панели приложения, больше соответствующий вашим нуждам. В настоящее время стандартная панель приложения имеет высоту 56 dp на мобильных устройствах (панель действий имела высоту 48 dp), но ее можно сделать выше, если потребуется, и даже изменять высоту динамически. Другие листы бумаги (компоненты интерфейса) могут упираться в панель приложения, задвигаться под нее и даже налезать на нее. Вы можете также создать вторую панель инструментов под панелью приложения, которую обычно так и называют: «нижняя панель инструментов».

Панель приложения уместна для большинства приложений, но вы можете скрыть ее при необходимости (если, к примеру, приложение служит для просмотра видео или чтения книг). Вообще, в материальном дизайне принято задвигать

панель за верхний край, когда пользователь взаимодействует с основным содержимым (например, когда прокручивает страницу вниз), но возвращается на место, когда она может потребоваться (например, когда пользователь выполняет прокрутку вверх). Вы будете работать с классом `Toolbar` на протяжении всей книги. Он представляет конкретную реализацию, доступную в библиотеке поддержки, позволяющую легко и просто создать панель приложения, как любой другой визуальный компонент, и имеет встроенную поддержку для создания меню.

Вкладки и навигационное меню

Вкладки – распространенная форма навигации, появившаяся задолго до персональных компьютеров, и до сих пор они остаются весьма эффективным средством. В Android вкладки всегда отображаются сверху. Если на экране присутствует панель приложения, вкладки располагаются под ней (например, см. рис. 1.14). В отличие от большинства других средств навигации, вкладки интуитивно понятны и позволяют увидеть сразу все разделы приложения. Идеальной считается ситуация, когда две-три вкладки находятся в фиксированных позициях. Если вам необходимо больше вкладок, их можно сделать прокручиваемыми, но при этом вы должны предусмотреть альтернативное средство навигации, такое как навигационное меню.

Навигационное меню – это средство навигации, отображающее несколько разделов приложения и, когда разделы делятся на подразделы, разные уровни в разделах. Навигационное меню обычно открывается при нажатии на кнопку с изображением «бутерброда» (три горизонтальные полосы друг над другом) или мазком пальца справа налево от правого края экрана. Оно всегда должно отображаться в полную высоту экрана. То есть оно появляется под панелью состояния, перекрывая панель приложения. В прошлом навигационные меню выдвигались под панелью приложения (панелью действий, как она тогда называлась), но это было обусловлено ограничениями реализации (панель действий была частью рамки окна, из-за чего вывести визуальный компонент поверх ее было невозможно). В настоящее время любое навигационное меню должно выводиться в полный экран.

Более подробно о вкладках и навигационных меню рассказывается в главе 6 «Макетирование и разработка основы приложения».

Плавающая кнопка

«Плавающая кнопка» – весьма выразительный элемент материального дизайна. Это круглая кнопка, которая отображается акцентированным цветом, когда для данной страницы определено действие по умолчанию (например, см. рис. 1.14). Это позволяет привлечь внимание к действию по умолчанию и поместить его в наиболее удобную позицию (в смысле досягаемости для большого пальца). Хотя эта кнопка называется «плавающей», вы можете считать ее наиболее востребованной кнопкой, потому что она всегда отвечает за операцию, чаще других используемую на данной странице. Если представить, что пользователь рабо-

тает с приложением создания заметок и находится на странице с их списком, плавающая кнопка определенно должна отвечать за операцию создания новой заметки (и иметь изображение в виде большого значка «плюс» или включать пиктограмму заметки со значком «плюс»). Наличие плавающей кнопки необязательно, поэтому не помещайте ее на страницы, где в ней нет необходимости. Если пользователь просматривает архив со старыми заметками и не может добавить новую, нет смысла отображать кнопку на этой странице. Никогда не используйте плавающую кнопку для нетипичных операций, таких как переход к разделу с настройками.

Поддержка разных устройств

В разделе «Покорите пользователя» выше уже говорилось, что важность поддержки разных устройств невозможно переоценить. Если вы не используете NDK (если вы не знаете, что это такое, значит, вы точно его не используете и можете с облегчением выдохнуть), вам не составит большого труда гарантировать поддержку намного более широкого диапазона устройств, чем можно себе представить. Фактически организация поддержки различных устройств заключается в основном в реализации разных макетов экранов. Вам остается сгруппировать компоненты пользовательского интерфейса в так называемые фрагменты (подробнее об этом рассказывается в главе 3 «Компоновка пользовательских интерфейсов с применением фрагментов и групп компонентов») и скомпоновать их для каждого устройства в отдельности, чтобы создать самые благоприятные ощущения у пользователя. Фрагменты обычно включают всю логику, необходимую для загрузки данных, взаимодействий и т. д., поэтому они с легкостью могут использоваться без изменений на устройствах разных типов. Визуальные компоненты будут иметь свои идентификаторы, поэтому в своем коде вы сможете просто выполнять высокоуровневые операции, такие как: «Вывести в `TextView` с дополнительной информацией текст “Бекон”», – и не беспокоиться о том, где в действительности находится этот компонент `TextView`, вверху экрана, внизу или вообще невидим в данный момент.

Для тех, кому до крайности любопытно узнать, что такое NDK, могу сообщить, что это Native Development Kit – комплект инструментов для разработки приложений с компиляцией в машинный код платформы, – позволяющий писать программы для Android на C/C++. В основном этот комплект используется для реализации программ с интенсивными вычислениями, таких как игры и физические симуляторы. Дополнительную информацию об этом комплекте можно получить по адресу: http://developer.android.com/tools/sdk/ndk/index.html.

На протяжении всей книги мы будем обсуждать приемы поддержки разных устройств. А пока самое важное, чтобы вы осознали, что не следует забывать о существовании разных устройств. Приемы и методики, которые мы будем обсуждать, хорошо зарекомендовали себя и способны добавить в ваше приложение поддержку устройств, о которых вы даже не слышали. Например, включение под-

держки состояния в фокусе ввода для всех элементов пользовательского интерфейса позволит вам избежать зависимости от сенсорного экрана. То есть вашим приложением можно будет пользоваться на устройствах Android TV (правда, при этом придется предусмотреть отдельный набор компоновок интерфейса для этой платформы), а также на устройствах, где навигация осуществляется иными средствами – не с помощью сенсорного экрана.

Как избежать болезненных ошибок

Рекомендации по материальному дизайну, представленные выше, помогут сделать ваше приложение лучше, но существует несколько очень болезненных ошибок, которые склонны допускать многие дизайнеры и разработчики. Две из них пришли из старых стандартов Android, которые уже не действуют: кнопка «Меню» и контекстное меню. Две другие распространенные ошибки, которые могут вызвать отрицательное отношение к вашему дизайну: использование неправильных пиктограмм в уведомлениях и заимствование стилей из других платформ.

Кнопка «Меню»

В версиях Android, предшествовавших версии Android 3.x, предполагалось, что устройства имеют кнопку «Меню». К сожалению, это стало причиной нескольких проблем. Одна из них состоит в том, что пользователю неизвестно заранее, реагирует ли приложение на эту кнопку. Он должен нажать кнопку, чтобы увидеть, что из этого получится. Поскольку многие приложения не реагировали на эту кнопку, ее не использовали даже в приложениях, где она действительно обрабатывалась, что вызывало трудности доступа к некоторым функциям.

Кроме того, реакция на кнопку «Меню» предполагалась зависимой от контекста – содержимого текущего экрана (по сути, это была предтеча панели приложения и меню дополнительных действий), но многие разработчики и дизайнеры использовали кнопку «Меню» непоследовательно и даже злоупотребляли ею, возлагая на нее функции навигационного меню. К счастью, теперь надобность в использовании этой кнопки осталась в прошлом. Ваше приложение должно быть нацелено на новейшие версии Android, где кнопка «Меню» больше не поддерживается.

Долгое нажатие

Жест «долгое нажатие» (иногда его называют долгим касанием или долгим щелчком) использовался для вывода контекстного меню, подобно щелчку правой кнопки мыши в настольных приложениях. Такая интерпретация долгого нажатия ушла в прошлое, и теперь долгое нажатие следует интерпретировать как операцию выбора. Долгое нажатие на определенный элемент должно приводить к его выбору и появлению полосы контекстного меню. Выбор одного или нескольких элементов может приводить к появлению таких контекстных операций, как «удалить», «заархивировать» или «отметить». Если приложение

не предусматривает возможности выбора элементов, долгое нажатие никак не должно обрабатываться. В Android 5.0 по умолчанию долгое нажатие просто воспроизведет эффект медленно распространяющейся ряби без выбора элемента, благодаря чему пользователь сможет узнать, что его касание опознано, но интерпретируется как обычное касание, потому что реакция на долгое нажатие не предусмотрена.

Для элементов в панели инструментов долгое нажатие используется также для отображения заголовка. То есть можно выполнить долгое нажатие на кнопку с невразумительной пиктограммой и получить краткую информацию о ее действии. К счастью, Android автоматически обрабатывает этот случай, а от вас требуется только определить заголовки для своих кнопок и элементов меню (а это, как вы знаете, необходимо для удобства использования).

Пиктограммы уведомлений

Для пиктограмм уведомлений предусмотрены свои рекомендации, имеющие особенно важное значение, потому что уведомления появляются в панели состояния над приложением. Пиктограммы уведомлений должны состоять только из белых и прозрачных пикселей. Не красных, не зеленых, не полупрозрачных. Они должны быть либо белыми, либо прозрачными. Это объясняется тем, что Android использует пиксели пиктограмм уведомлений как маску, и к тому же такие пиктограммы появляются в ряду с другими подобными пиктограммами, созданными с соблюдением этого правила. Если ваша пиктограмма создана с нарушением правил, он будет вызывать визуальный диссонанс, а приложение создаст впечатление не предназначенного для Android.

Стили из других платформ

Выше, в разделе «Единообразие с платформой», уже говорилось об этом, но хотелось бы повторить еще раз: не заимствуйте стили оформления из других платформ. Одна из типичных ошибок в оформлении приложений для Android – заимствование оформления из приложений для iOS и его «перенос» в Android. Приложения для iOS следуют иным рекомендациям по оформлению и приемам взаимодействий, из-за чего приложения в стиле iOS выглядят в Android чужеродными. Такой подход сообщает пользователям, что лица, ответственные за приложение, не заботятся о пользователях Android и создали приложение просто потому, что у кого-то в планах присутствовал пункт «выпустить приложение для Android». Такое приложение не появится в Google Play, из-за чего пользователи могут перейти на использование конкурирующего приложения. Если вы получите прямое указание от руководителя или дизайнера реализовать приложение именно так, а не иначе, познакомьте их с руководством по материальному дизайну, сошлитесь на эту книгу и вообще попробуйте сообщить им все, что, на ваш взгляд, сможет повлиять на их мнение. Вы должны создать приложение, которым могли бы гордиться и которым другие захотели бы пользоваться, а не просто сырой аналог.

В заключение

Теперь, после прочтения этой главы, вы должны иметь достаточно полное представление о том, как должны выглядеть современные приложения для Android. Вы узнали о существовании стандартных компонентов (таких как панель приложения) и устаревших механизмах (таких как кнопка «Меню»), которых следует всячески избегать. В разделе «Основные понятия» вы познакомились с некоторыми основными идеями, которые вам предстоит воплощать в будущих приложениях. И, возможно, вы сочтете полезным вернуться к этой главе позднее, когда приступите к проектированию своего приложения.

Перед тем как перейти к главе 2, обязательно загляните на веб-сайт, посвященный вопросам дизайна в Android (http://developer.android.com/design/), и на веб-сайт с руководством по материальному дизайну (http://www.google.com/design/spec/material-design/)[1], если вы этого еще не сделали. Также может быть полезным выбрать несколько приложений, которые пользуются большой популярностью, и посмотреть, насколько они соответствуют всему тому, что вы узнали в этой главе. Исследование поведения и оформления популярных приложений поможет вам писать свои приложения, соответствующие ожиданиям пользователей и как минимум такие же популярные, как существующие. Будет просто замечательно, если вдобавок к этому вы подметите, в каких направлениях можно было бы улучшить приложения, которые прежде вы считали безупречными.

[1] Перевод можно найти по адресу: http://css-live.ru/articles/materialnyj-dizajn-vvedenie.html. – *Прим. перев.*