Ян Клифтон

Проектирование пользовательского интерфейса в ANDROID



ДМК ПРЕСС
издательство

УДК 004.451.9 Android:004.5
ББК 32.972.11
К49

Клифтон Я.
К49 Проектирование пользовательского интерфейса в Android/ пер. с англ. А. Н. Киселева. 2-е изд. – М.: ДМК Пресс, 2017. – 452 с.: ил.

ISBN 978-5-97060-449-6

С развитием операционной системы Android и увеличением ее конкурентоспособности разработчики признали исключительную важность хорошего оформления приложений. В виде положений материального дизайна компания Google представила самые радикальные визуальные изменения, которые когда-либо производились, и подняла значимость эффективного дизайна на еще более высокий уровень. В данной книге автор демонстрирует, как создавать исключительно удобные и внешне очень привлекательные интерфейсы. Книга научит вас создавать приложения, использующие предыдущий опыт пользователя и никогда не испытывающие его терпения! Приложения, успешно осваиваемые новыми пользователями с первой попытки!

Издание предназначено всем разработчикам Android-приложений, как начинающим, так и более опытным.

УДК 004.451.9 Android:004.5
ББК 32.972.11

Authorized translation from the English language edition, entitled ANDROID USER INTERFACE DESIGN: IMPLEMENTING MATERIAL DESIGN FOR DEVELOPERS, 2nd Edition; ISBN 0134191404; by CLIFTON, IAN G.; published by Pearson Education Inc., publishing as Addison-Wesley Professional. Copyright © 2016 by Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education Inc. RUSSIAN Language edition published by DMK PUBLISHERS. Copyright © 2017.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-0-134-19140-9 (анг.)
ISBN 978-5-97060-449-6 (рус.)

© 2016 Pearson Education, Inc.
© Оформление, издание, перевод, ДМК Пресс, 2017

Содержание

Предисловие 14

Благодарности 15

Об авторе 16

Введение 17

Часть I. ОСНОВЫ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ 19

Глава 1. Пользовательский интерфейс Android и материальный дизайн 20

Краткая история развития дизайна Android 20
Материальный дизайн 23
- Основные понятия 23
- Взаимодействия и анимация 26
- Типографика 27
- Единицы измерения и выравнивание 28

Веб-сайт о дизайне Android 28
Основные принципы 29
- Делайте что-то одно, но делайте это хорошо 29
- Будьте вежливы 31
- Образы, образы, образы 32
- Простое, но мощное 33
- Единообразие с платформой 35
- Покорите пользователя 36

Стандартные компоненты 36
- Системные панели 36
- Уведомления 37
- Панель приложения 38
- Вкладки и навигационное меню 40
- Плавающая кнопка 40

Поддержка разных устройств 41
Как избежать болезненных ошибок 42
- Кнопка «Меню» 42
- Долгое нажатие 42
- Пиктограммы уведомлений 43
- Стили из других платформ 43

В заключение 44

Глава 2. Виджеты – строительные блоки пользовательского интерфейса 45
Что такое виджет 45
Идентификаторы виджетов 47
Размеры виджетов 48
Отображение текста 52
TextView 53
EditText 56
Button 56
Отображение изображений 57
Фон 57
ImageView 58
ImageButton 59
Виджеты для ввода информации пользователем 59
Другие примечательные виджеты 63
Прием и обработка событий 66
OnClickListener 66
OnLongClickListener 66
OnTouchListener 66
Другие обработчики событий 67
В заключение 67

Глава 3. Создание пользовательских интерфейсов с применением фрагментов и групп компонентов 68
ViewGroup и типовые реализации 68
FrameLayout 69
LinearLayout 69
RelativeLayout 72
AdapterView 74
ListView 74
GridView 76
Spinner 76
Gallery 77
Adapter 77
Интерфейсы для **AdapterView** 78
ViewPager 78
Toolbar 79
Другие примечательные реализации ViewGroup 79
Инкапсуляция логики отображения с использованием фрагментов 81
Жизненный цикл фрагментов 82
Передача данных фрагменту 83
Взаимодействие с Activity 85
Транзакции с фрагментами 87

Обсуждение проблем ... 88
Библиотека поддержки ... 89
Библиотека CardView ... 91
Библиотека Design ... 91
Библиотека GridLayout ... 92
Библиотека Leanback ... 92
Библиотека MediaRouter ... 92
Библиотека Palette ... 92
Библиотека RecyclerView ... 93
Библиотека Support Annotations ... 93
В заключение ... 94

Глава 4. Добавление графики и ресурсов в приложения ... 95

Введение в ресурсы ОС Android ... 95
Квалификаторы ресурсов ... 96
Разрешение ... 101
Поддерживаемые файлы изображений ... 102
Растровые изображения ... 103
Векторные изображения ... 104
Изображения в формате Nine-Patch ... 104
Графические элементы в XML ... 106
Список слоев ... 107
Список состояний ... 108
Список уровней ... 111
TransitionDrawable ... 112
InsetDrawable ... 112
ClipDrawable ... 112
ScaleDrawable ... 113
ShapeDrawable ... 113
VectorDrawable ... 115
AnimatedVectorDrawable ... 118
RippleDrawable ... 119
Другие ресурсы ... 120
Строки ... 121
Массивы ... 122
Цвета ... 123
Размеры ... 124
Анимации ... 124
Числовые идентификаторы ... 124
Меню ... 125
В заключение ... 126

Часть II. ПОЛНЫЙ ПРОЦЕСС ПРОЕКТИРОВАНИЯ И РАЗРАБОТКИ....127

Глава 5. Начало создания нового приложения 128
Методы проектирования 128
Типичные методы 128
Проектирование с ориентацией на пользователя 129
Определение целей....132
Цели пользователей....132
Группы пользователей....133
Цели продукта....134
Поддержка устройств и конфигураций 135
Высокоуровневая диаграмма....136
Эскизы 139
Начните с навигации 142
Информационное наполнение 145
Эскиз страницы с описанием....147
Поддержка нескольких устройств 149
Соглашения об именовании....150
Заготовки для ресурсов 151
В заключение....152

Глава 6. Макетирование и разработка основы приложения 153
Организация экранов и фрагментов 153
Создание первого прототипа 154
Вкладки 156
Навигационное меню 158
Представление инструментов 161
Фрагменты для вкладок 166
Сведения об инструменте 180
Оценка первого прототипа 184
Работа с пользователями 184
Свободное исследование....185
Специализированные цели 185
Динамические цели....186
Отзывы пользователей 186
Следующие шаги....188
В заключение....189

Глава 7. Визуальный дизайн 190
Эскизы и графический дизайн....190
Инструменты....191
Стили....192

Скевоморфизм ... 192
Минимализм ... 192
Плоский дизайн ... 193
Материальный дизайн ... 193
Освещение ... 194
Цвета ... 195
Теория ... 195
Выбор цветов ... 198
Приложение выбора столярных инструментов ... 202
Текст ... 202
Контраст текста ... 203
Размер текста, стили оформления и заглавные буквы ... 204
Интервалы в тексте ... 204
Тени, отбрасываемые текстом ... 205
Нестандартные шрифты ... 205
Доступный словарь ... 207
Другие вопросы ... 207
Разный объем текста ... 207
Размеры и доступность изображений ... 208
Прозрачность и правило 3× ... 208
Стандартные пиктограммы ... 209
Навигация и визуализация переходов ... 209
Обработка ошибочных ситуаций ... 210
Пошаговое оформление ... 210
В заключение ... 215

Глава 8. Реализация дизайна ... 216
Совместная работа с дизайнером ... 216
Нарезка графических ресурсов ... 217
Простые нарезки ... 218
Изображения в формате Nine-Patch ... 219
Создание альтернативных ресурсов для других размеров ... 223
Темы и стили ... 224
Деление дизайн-макета на виджеты ... 225
Разработка приложения выбора столярных инструментов ... 227
Главный экран ... 227
Список инструментов ... 236
Описание отдельных инструментов ... 249
Тестирование на разных типах устройств ... 253
В заключение ... 254

Глава 9. Украшение анимацией ... 255
Назначение анимации ... 255
Анимация виджетов ... 256

Анимация свойств ... 257
Управление анимацией свойств ... 258
Обработчики ... 258
Типы вычисляемых значений ... 259
Интерполяция во времени ... 262
Ключевые кадры ... 264
Класс ViewPropertyAnimator ... 265
Анимация вывода сообщений об ошибках в формах ... 265
Анимация пиктограмм ... 269
Анимация векторных пиктограмм ... 269
Анимация растровых пиктограмм ... 274
Простые переходы ... 275
Переходы между сценами ... 276
Переходы между экранами ... 279
Переходы в виде расширяющегося круга ... 283
В заключение ... 286

Часть III. ДОПОЛНИТЕЛЬНЫЕ ВОПРОСЫ СОЗДАНИЯ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ ... 287

Глава 10. Продвинутые приемы ... 288

Выявление задержек ... 288
Использование Systrace для определения причин задержек ... 290
Оптимизация изображений ... 297
Сжатие изображений ... 297
Использование изображений подходящих размеров ... 299
Кэширование изображений ... 303
Дополнительные методы увеличения производительности ... 309
Управление сборкой мусора ... 309
Шаблон «Держатель виджетов» ... 310
Устранение избыточного рисования ... 312
Обозреватель иерархии ... 316
Поиск пропавших виджетов ... 318
Устранение ненужных виджетов ... 319
Экспорт в формат PSD ... 319
Нестандартные шрифты ... 320
Комплексные виджеты TextView ... 323
Имеющиеся промежутки ... 323
Использование промежутков для отображения текста со сложным оформлением ... 325
Класс RecyclerView ... 326
Диспетчер размещения ... 327
Адаптер ... 327

Аниматор элементов ... 328
Декорирование элементов ... 328
В заключение ... 328

Глава 11. Холсты и дополнительные приемы отображения графики ... 330
Создание собственных графических элементов ... 330
Основная идея ... 330
Другие важные методы ... 331
Класс Paint ... 332
Класс Canvas ... 332
Работа с текстом ... 333
Простой текстовый элемент ... 333
Улучшенный текстовый элемент ... 335
Работа с изображениями ... 338
Цветовые фильтры ... 340
Простой цветовой фильтр ... 340
Цветовая матрица ... 341
Наложение изображений с помощью режимов PorterDuff ... 343
Шейдеры ... 353
Закругление краев изображений ... 353
Эффект градиентного растворения ... 356
В заключение ... 359

Глава 12. Разработка собственных виджетов ... 360
Основные идеи ... 360
Измерение размеров ... 361
Размещение ... 361
Рисование ... 362
Сохранение и восстановление состояния ... 363
Создание собственного виджета ... 363
Измерение ... 365
Размещение дочерних виджетов ... 366
Создание изображения ... 366
Сохранение и восстановление состояния ... 368
В заключение ... 375

Глава 13. Обработка ввода и прокрутки ... 376
Сенсорный ввод ... 376
Другие формы ввода ... 378
Создание собственного виджета ... 378
Создание первоначальных файлов виджета ... 379
Измерение ... 384

Рисование ... 387
Подготовка к обработке событий сенсорного ввода ... 392
Обработка событий сенсорного ввода ... 396
Прочие вопросы ... 412
В заключение ... 412

Приложение A. Ресурсы Google Play ... 413
Описание приложения ... 413
Список изменений ... 414
Пиктограмма приложения ... 415
Простой способ создания пиктограмм ... 416
Создание пиктограмм вручную ... 417
Скриншоты ... 421
Основной баннер ... 423
Рекламный баннер ... 425
Видео (YouTube) ... 426
Продвижение приложения ... 426
Amazon Appstore ... 427

Приложение B. Справочник по типичным задачам ... 428
Сокрытие экранной клавиатуры ... 428
Включение полноэкранного режима ... 429
Предотвращение выключения экрана ... 430
Определение физических размеров экрана устройства ... 430
Определение размеров экрана в пикселях ... 431
Определение разрешения экрана ... 431
Проверка подключения к сети ... 432
Проверка, является ли текущий поток выполнения потоком пользовательского интерфейса ... 433
Нестандартные атрибуты виджетов ... 434

Предметный указатель ... 441

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.

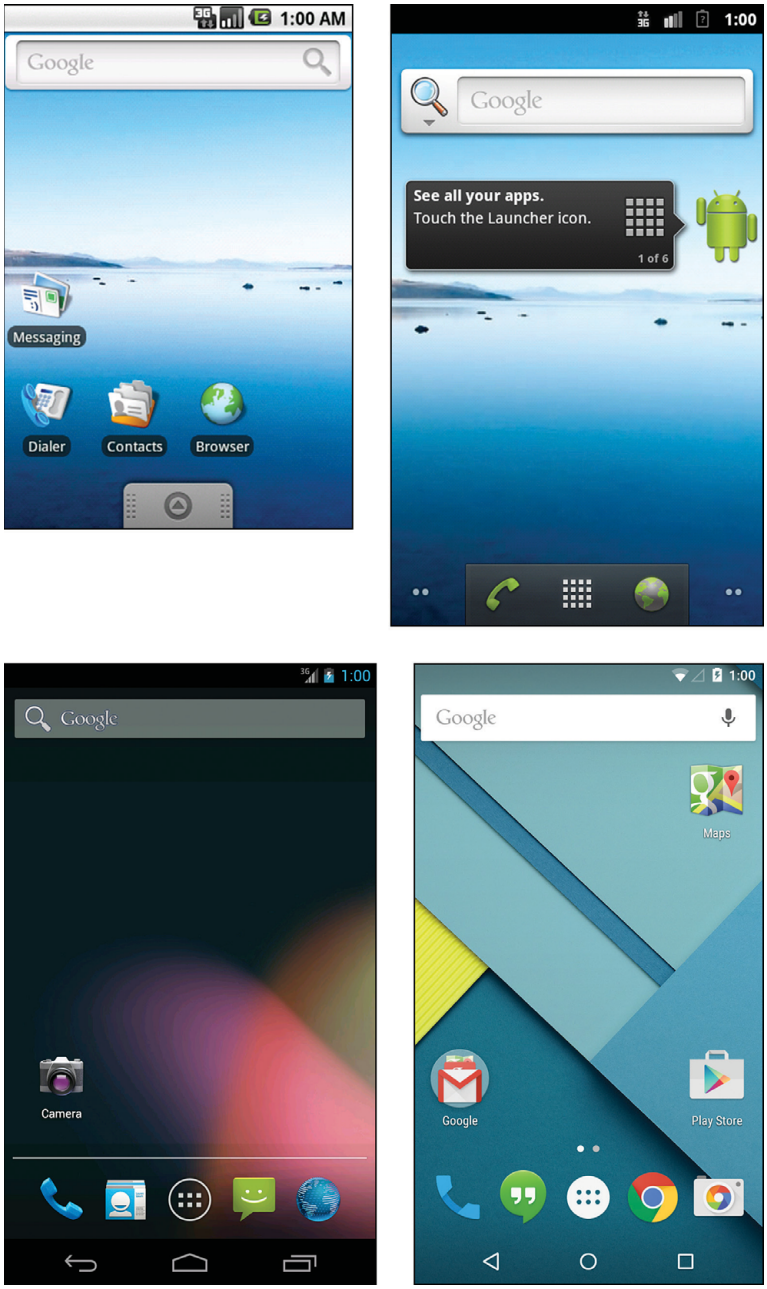


Рис. 1.1 ❖ Домашний экран в версиях Android 1.6 (вверху слева), 2.3 (вверху справа), 4.2 (внизу слева) и 5.0 (внизу справа)

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.

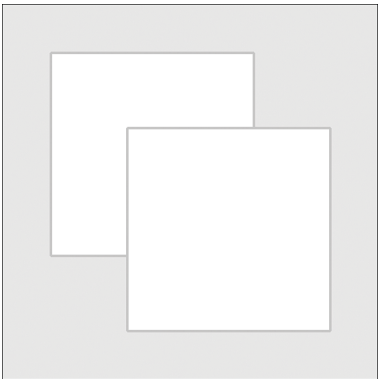


Рис. 1.3 ❖ Эффект заслонения одних объектов другими создает ощущение, что один объект находится перед другим, но это не позволяет судить, насколько один объект ближе к вам, чем другой



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

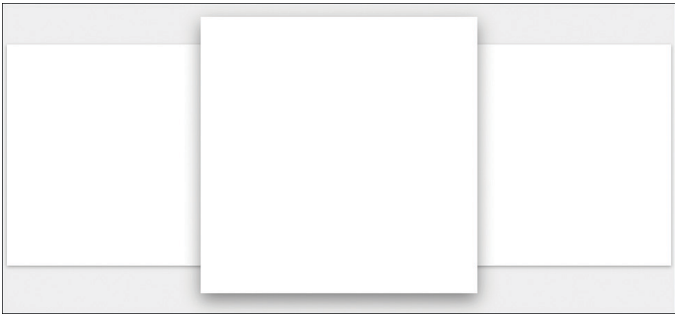


Рис. 1.5 ❖ Объединение всех перечисленных принципов создает более полное ощущение, что объект в середине находится ближе к поверхности экрана, даже если соседние объекты чуть-чуть «приподнять»

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».

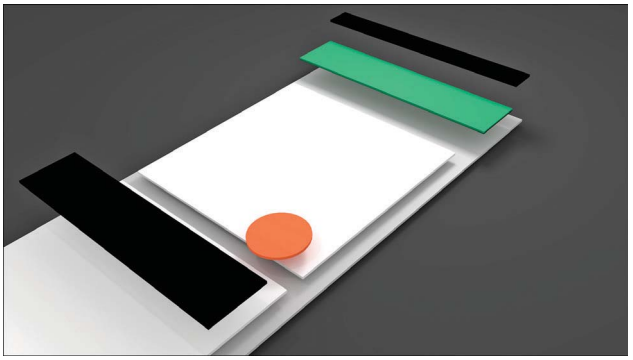


Рис. 1.6 ❖ Трехмерное изображение, созданное с применением принципов материального дизайна

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.

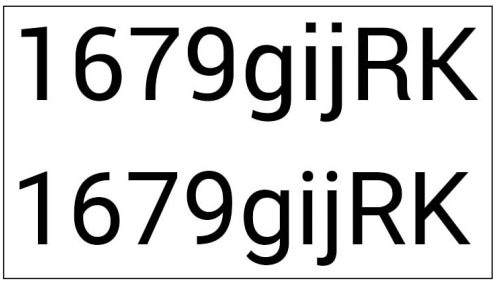


Рис. 1.7 ❖ Некоторые наиболее заметные отличия между исходным шрифтом Roboto (вверху) и его улучшенной версией (внизу)

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.

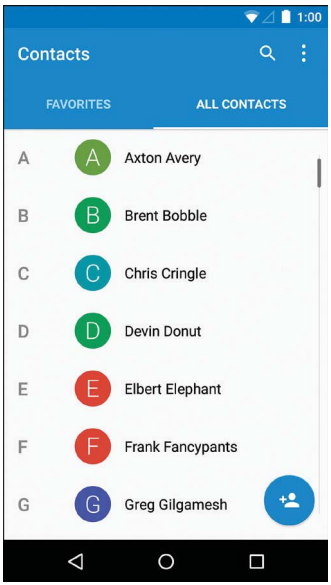


Рис. 1.8 ❖ Приложение Contacts (Контакты) – отличный пример приложения с единственной и ясной целью

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.

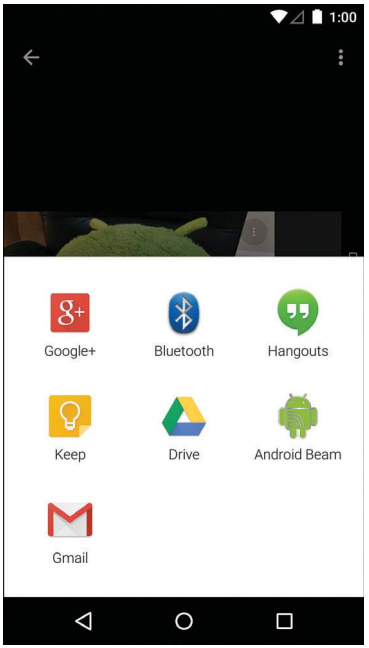


Рис. 1.9 ❖ Пользовательский интерфейс по умолчанию, позволяющий выбрать приложение

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.

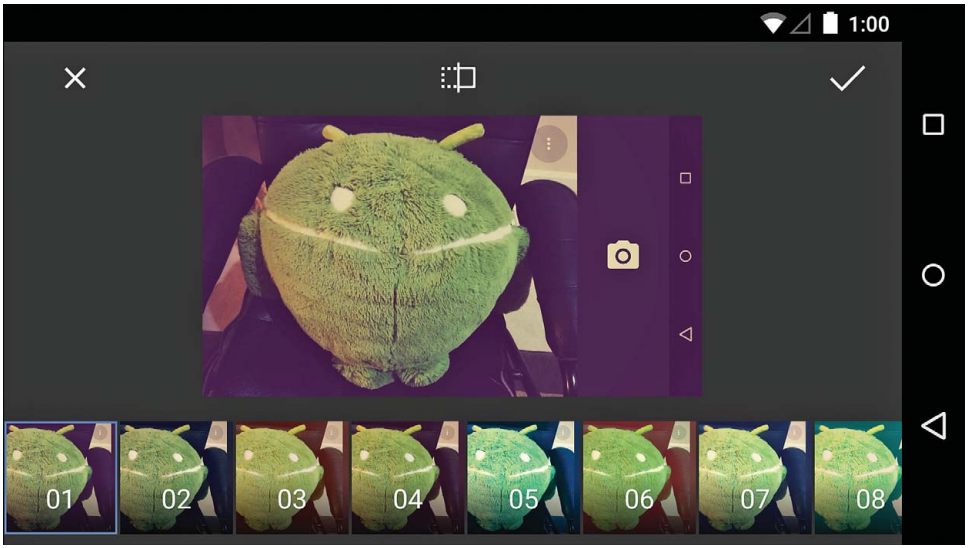


Рис. 1.10 ❖ Каждая функция обработки изображений обозначена в нижнем ряду миниатюрой, иллюстрирующей ее действие

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.