ДМИТРИЙ ЕЛИСЕЕВ

bhv®

Разработка и продажа программ для iPhone и iPad

- ОСНОВНЫЕ СВЕДЕНИЯ об Apple iOS
- РАЗРАБОТКА ПРОСТЕЙШЕЙ ПРОГРАММЫ В СРЕДЕ Xcode
- РАБОТА С GUI, ЗВУКОМ, ТАБЛИЦАМИ, ТЕКСТОВЫМИ И ГРАФИЧЕСКИМИ ДАННЫМИ
- ЗАГРУЗКА ДАННЫХ ИЗ ИНТЕРНЕТА
- БИБЛИОТЕКА ИГРОВОГО ФИЗИЧЕСКОГО МОДЕЛИРОВАНИЯ Box2D
- ГРАФИЧЕСКАЯ БИБЛИОТЕКА ДЛЯ СОЗДАНИЯ ИГР Cocos2D
- ОСНОВЫ ЯЗЫКА ObjectiveC

PRO

ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

Дмитрий Елисеев

Разработка и продажа программ для iPhone и iPad

Санкт-Петербург

«БХВ-Петербург»

2011

УДК 681.3.06
ББК 32.973.26-018.2
Е59

Елисеев Д. В.

Е59 Разработка и продажа программ для iPhone и iPad. — СПб.: БХВ-Петербург, 2011. — 336 с.: ил. — (Профессиональное программирование)

ISBN 978-5-9775-0687-8

Рассмотрены вопросы создания программных приложений для мобильных устройств Apple iPhone и iPad. Изложены основные принципы функционирования программ в операционной системе Apple iOS, даны основы языка ObjectiveC. Рассмотрена работа с GUI, звуком, таблицами, текстовыми и графическими данными, приведены способы загрузки данных из Интернета.

Каждый раздел снабжен практическими примерами приложений. Описаны как разработка простейших программ в среде Xcode, так и создание сложных игровых программ с использованием профессиональных библиотек Box2D и Cocos2D. Особое внимание уделено практике размещения и продажи собственных программ в магазине Apple App Store.

Для программистов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Елена Кашлакова*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Наталия Першакова*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 22.07.11.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 27,09.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12.

ISBN 978-5-9775-0687-8

Оглавление

Введение

В настоящее время на рынке представлены сотни различных видов телефонов, смартфонов и карманных компьютеров. Но среди всего этого разнообразия есть одна компания, продукцию которой знают все, и как, конечно, догадались читатели, это фирма Apple. Это компания, поклонники которой разбивают палатки у магазина за неделю до начала продаж, чтобы стать первым обладателем новой модели устройства…

Конечно, основной успех Apple в настоящее время вызван линейкой телефонов iPhone, но даже без них имя Apple вписано в историю компьютерной техники: компания представила еще в 1977 г. персональный компьютер Apple-II и операционную систему Apple DOS (которая поддерживала длинные имена файлов, опередив на 15 лет MS Windows). Уже в 1984 г. Mac OS поддерживала оконный интерфейс и манипулятор "мышь", а в 1993 г. было выпущено устройство Apple MessagePad, представляющее собой карманный компьютер. Он имел процессор с тактовой частотой 20 МГц, 640 Кбайт памяти и экран с разрешением 240×336, управляемый стилусом. Пожалуй, MessagePad весьма опередил свое время, ведь аналогичное устройство от Palm появилось только через 3 года, а настоящий расцвет портативных устройств произошел, наверное, уже в XXI веке. Наконец, компания совершила переворот, представив смартфон iPhone, имеющий необычный (для 2007 г.) "бесknопочный" дизайн, экран, управляемый нажатием пальца, встроенный датчик ориентации устройства и прочие новшества, которые лишь потом стали спешно "клонироваться" в телефонах других производителей. И уже буквально во время написания этой книги Apple (в очередной раз) на шаг впереди конкурентов представила смартфон iPhone 4, не имеющий пока аналогов на рынке.

Но впрочем, эта книга не об истории, а о программировании. Поэтому стоит вернуться к настоящему. Сегодня смартфоны и планшетные компьютеры Apple пользуются заслуженной популярностью, поэтому изучение программирования для этой платформы может быть не только интересным, но и коммерчески привлекательным. *В главе 1*, от простого к сложному, описываются основные этапы создания программного обеспечения для устройств iPhone и iPad. Читатель найдет *в главах 2–5* информацию, необходимую для использования графического интерфейса операционной системы iOS, описание работы с файлами, звуком *в главах 6 и 7*, сетью Интернет. *В главе 8* рассматриваются практические примеры использования

библиотек Box2D и Cocos2D, применяющихся при создании игровых программ. Книга предоставляет информацию не только о программировании, *в главах 9–11* на практике описывается процесс продажи программ в магазине App Store, рассматриваются необходимые требования для выпуска программы в продажу, вплоть до открытия счета в банке.

Каждой теме посвящено отдельное приложение на сайте книги, что позволяет быстро найти нужную информацию или использовать фрагменты кода в качестве основы для новых проектов. Для компиляции примеров программ необходима среда разработки XCode 3.1, распространяемая бесплатно. Авторские примеры можно скачать по ссылке **http://www.bhv.ru/files/9785977506878.zip**.

Автор надеется, что книга позволит читателям подробнее изучить программирование для операционной системы iOS и, возможно, выпустить на рынок новые и интересные программы. Книга рассчитана на читателей, знакомых с основами программирования на языке C или C++.

Глава 1

Портативные устройства Apple

Смартфоны и портативные компьютеры Apple, имеющиеся на рынке

Компания Apple — одна из старейших компаний на компьютерном рынке, ее история начинается еще с 1976 г., когда Стив Джобс (Steve Jobs) и Стив Возняк (Steve Wozniak) создали один из первых персональных компьютеров. По одной из версий, название компании специально было подобрано так, чтобы быть в первых страницах телефонных справочников. С тех пор дела компании шли с переменным успехом: основными проблемами были закрытость архитектуры и использование процессоров Motorola, несовместимых с популярными IBM PC, и доля компьютеров Mac на рынке была не столь большой. Но в то же время она была достаточной и позволила компании развивать параллельные ниши, например плееры iPod и другие устройства.

Датой массового прихода Apple на рынок смартфонов можно считать 2007 г., когда первый iPhone был представлен на выставке MacWorld. Устройство объединило многие передовые тенденции, некоторые из которых были достаточно революционными: управление с помощью касания экрана пальцем, а не стилусом, поддержка технологии множественного касания Multi-touch, встроенный акселерометр и датчик ориентации с возможностью автоматического поворота интерфейса системы при изменении наклона устройства. И наконец, самое главное — уникальный дизайн, упрощенный до максимума: на устройстве только одна кнопка. В компании пришли к важному выводу: большинство пользователей покупают не "мегагерцы и мегабайты", им нужно красивое и эффектно выглядящее устройство. И расчет оправдался, смартфоны Apple не прошли незамеченными, и хотя одним такой концепт нравился, а другим — нет, равнодушных к новому устройству практически не осталось. По большому счету, это и есть основной фактор, определяющий популярность продукции у пользователей. В 2010 г. был выпущен новый смартфон iPhone 4, который опять-таки произвел фурор благодаря экрану сверхвысокого разрешения 960×640 и приятному дизайну. Ажиотаж был столь большим, что предварительные заказы размещались задолго до выхода.

С точки зрения программиста, все это приводит к простому выводу: разработка приложений для таких устройств не только интересна, но и перспективна и, возможно, прибыльна.

Все устройства Apple, работающие под управлением iOS, можно разделить на три категории:

- смартфоны Apple iPhone со встроенной операционной системой, дающей возможность пользователю загружать свои программы. Эти устройства имеют модули GSM и GPS, встроенную камеру и прочие атрибуты, свойственные современным телефонам;
- плееры iPod Touch — мало кто знает, что с точки зрения разработки программного обеспечения iPod Touch предоставляет практически те же возможности, что и iPhone, но при этом стоит вдвое дешевле;
- планшетные компьютеры Apple iPad с IPS-дисплеем диагональю 9,7 дюймов и разрешением 1024×768. iPad может выполнять также и приложения для iPhone, что позволяет успешно использовать его для отладки приложений, разрабатываемых как для iPhone, так и для iPad.

Итак, типы имеющихся устройств можно свести в табл. 1.1.

Таблица 1.1. *Устройства, работающие под управлением iOS*

Модель	Операционная система	Разрешение экрана	Процессор
iPhone 3G	2.0−4.2	480×320	412 МГц
iPhone 3GS	3.0−4.2	480×320	600 МГц
iPod Touch 3	3.0−4.2	480×320	600 МГц
iPhone 4	4.0−4.3	960×640	1 ГГц
iPod Touch 4	4.0−4.3	960×640	1 ГГц
iPad	3.2−4.3	1024×768	1 ГГц
iPad 2	4.3	1024×768	1 ГГц, два ядра

Приведенные данные тактовой частоты являются ориентировочными и взяты из сторонних источников, что, впрочем, для разработки прикладных программ (кроме игр) не так уж критично. Более важно то, что разрешение экрана новых смартфонов iPhone 4 ровно вдвое больше и обеспечивает совместимость со старыми версиями программ, ведь масштабирование в два раза может осуществляться системой быстро и без потери четкости.

Внешний вид iPhone 4 показан на рис. 1.1. В верхней части устройства имеются кнопка включения, разъем для наушников, на передней части можно видеть камеру для видеозвонков и кнопку **Home**, слева находятся кнопки регулировки громкости. В нижней части расположены динамик и разъем для подключения устройства к компьютеру. Через этот разъем iPhone подключается к компьютеру для зарядки и загрузки программного обеспечения (отладка программ также может проводиться на устройстве, а при его отсутствии — на симуляторе).

Рис. 1.1. Внешний вид Apple iPhone

Рис. 1.2. Внешний вид Apple iPad

Apple iPad выглядит практически так же, как увеличенный iPhone, это можно видеть на рис. 1.2. Операционная система обоих устройств одна и та же, что для разработчиков ПО является большим плюсом. Но, в отличие от iPhone, iPad не имеет камеры (она имеется только в новой модели iPad 2), в ряде моделей отсутствует GSM-модуль и приемник GPS. Процессор нового iPad 2 быстрее, чем в iPhone: очевидно, что более высокое разрешение экрана приводит к более высоким требованиям к памяти и быстродействию. На момент написания книги в продаже имеются модели с 16, 32 и 64 Гбайт флеш-памяти. Важно заметить, что память iPhone или iPad не может быть расширена, никакого слота для карт памяти там нет. Поэтому в отличие от программ, распространяемых на DVD, здесь пользователя вряд ли обрадует демонстрационный видеоролик HD-качества объемом в 100 Мбайт, вложенный в дистрибутив программы. С другой стороны, объема памяти предоставляемого устройствами обычно более чем достаточно для большинства прикладных программ.

Помимо "стандартных" устройств ввода (сенсорный экран, микрофон) iPhone и iPad имеют и специфические для этой платформы дополнительные средства — встроенные акселерометр и компас. С их помощью можно разнообразить управление программами (в основном конечно, игровыми, например, весьма популярны гонки, управление автомобилем в которых осуществляется с помощью наклона устройства).

Более-менее разобравшись с аппаратной частью, перейдем к программной: посмотрим, на каких элементах графического интерфейса построена операционная система Apple iOS, и как мы можем их использовать.

Основные принципы графического интерфейса Apple iOS

Несмотря на внешнее сходство с другими мобильными операционными системами, в Apple достаточно сильно переработан графический интерфейс. Основным для начинающего изучение iOS разработчика является документ "iOS Human Interface Guidelines", размещенный на сайте **http://developer.apple.com**, ознакомиться с которым следует каждому, кто собирается программировать или серьезно интересуется этой системой. В этом документе также содержится множество важных для разработки сведений, например размеры и форматы всех графических файлов, поставляемых с программой, без соблюдения которых программа не будет допущена к продаже в магазине App Store.

Наиболее важными для понимания являются несколько базовых принципов.

- Интерфейс iOS ориентирован на нажатие пальцем. Это обстоятельство накладывает ограничения на размер элементов, минимальный рекомендуемый Apple размер для комфортного нажатия составляет 44×44 пикселов для обычных экранов и 88×44 — для экранов высокого разрешения (960×640).
- Ориентация экрана может меняться в процессе работы программы. Два основных положения, портретное и ландшафтное, программа должна корректно масштабировать. К счастью для разработчиков, iOS предоставляет встроенные возможности, значительно облегчающие эту задачу.
- В один момент времени пользователь работает только с одной программой. В отличие от настольных компьютеров, в iOS нет многооконного интерфейса, и для небольших экранов это вполне оправданно. В то же время программы могут работать в фоне, начиная с версии 4.0, в iOS имеется многозадачность.
- Каждая программа имеет единственное "рабочее" окно. В iOS нет понятия многодокументного интерфейса (MDI, Multiple Document Interface), пользователь может работать лишь с одним окном в данный момент времени. Опять-таки, для небольших экранов это вполне логично и целесообразно.
- Интерфейс должен быть наглядным и интуитивно понятным. Для этого в iOS предусмотрены специальные средства: "переключатели", возможность изменения масштаба при помощи сжатия или растяжения объекта пальцами, перетаскивание объектов движением руки. Программные продукты в iOS приобретаются через сеть Интернет, обычно непосредственно с мобильного устройства, поэтому пользователь, скорее всего, не будет отдельно искать и загружать документацию. В идеале программа должна быть понятной пользователю без обращения к справке.

Вот некоторые рекомендации, которые дает Apple в документе "Human Interface Guidelines":

- Элементы управления должны выглядеть "нажимаемыми". Для встроенных компонентов iOS это уже реализовано, все кнопки и переключатели имеют контуры и градиенты, делающие их трехмерными.

- Структура программы должна быть понятной для навигации. Для этого iOS предоставляет встроенные средства, обеспечивающие удобное иерархическое представление, о них будет рассказано далее.
- Реакция программы на действия пользователя должна быть быстрой и понятной. Для этого iOS предоставляет встроенные средства, позволяющие сделать программу более красивой, такие как анимация или встроенные индикаторы прогресса. Эти "мелочи" позволяют пользоваться программой более комфортно.
- Зачастую пользователи мобильных устройств запускают приложения "на ходу", поэтому программа должна предоставлять доступ к содержимому просто и быстро. В этом плане логика работы пользователя за мобильным устройством отличается от действий за настольным компьютером.
- Желательно, чтобы программа сохраняла свое состояние при переключении. Новая версия iOS 4 поддерживает многозадачность, но есть еще немало пользователей предыдущих версий операционной системы. В них по нажатию кнопки **Home** программа закрывается, и это не должно приводить к потере важных для пользователя данных.
- В программе нет кнопки **Выход**. Это может показаться удивительным для тех, кто привык к Windows, тем не менее, программа в iOS не имеет кнопки закрытия. Для свертывания программы и появления главного экрана используется упомянутая кнопка **Home**.

Для иллюстрации этих принципов рассмотрим программу Mail, входящую в состав операционной системы Mac OS (рис. 1.3).

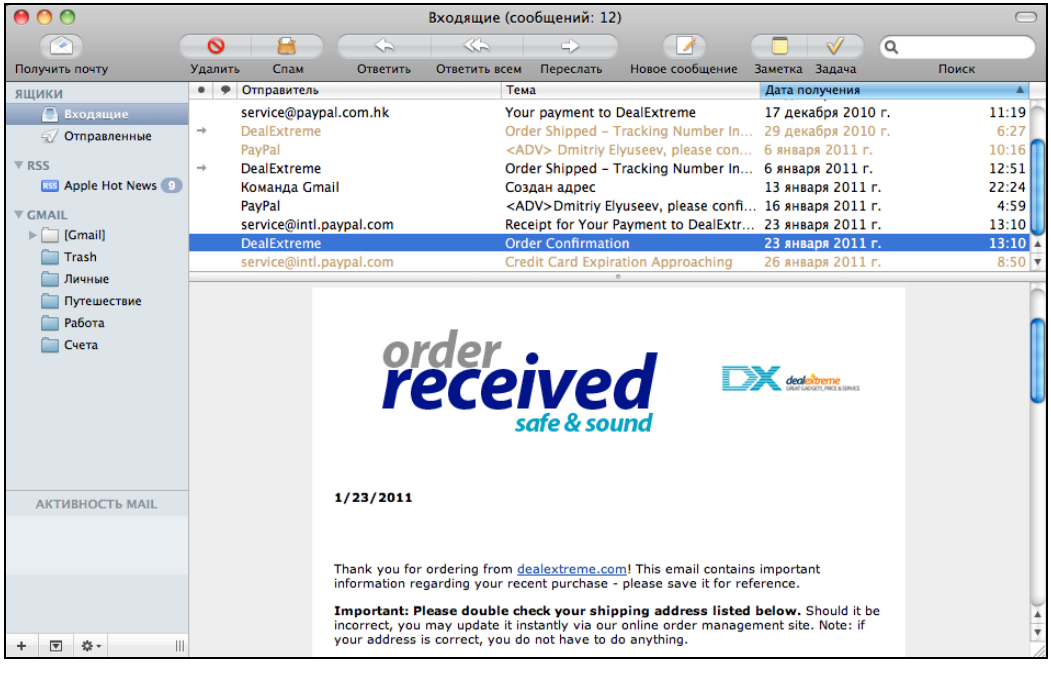

Рис. 1.3. Программа Mail для персонального компьютера

Здесь мы видим стандартное приложение для чтения электронной почты, однако в таком виде оно не поместится на экране мобильного телефона. Большое количество элементов управления, рассчитанных на использование "мыши", и несколько уровней вложенности интерфейса не могут быть с тем же удобством реализованы на экране меньшего размера. Для сравнения рассмотрим другую программу (окно программы настроек iPhone можно видеть на рис. 1.4).

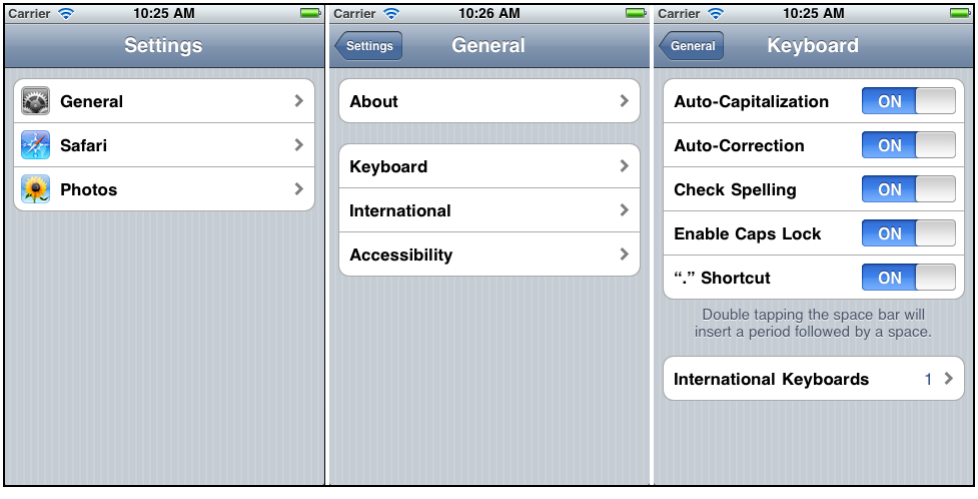

Рис. 1.4. Настройка параметров системы на iPhone

Вместо многооконного интерфейса с "параллельным" доступом (пользователь сразу может выбрать любую папку в почтовом ящике) был использован "последовательный", где пользователь открывает окна в четко заданной иерархии. Также можно видеть отсутствие кнопок маленького размера, ориентированных на "мышь", вместо них используются более крупные элементы управления.

На рис. 1.5 можно видеть ту же программу настроек, выполняющуюся на Apple iPad.

Рис. 1.5. Настройка параметров системы на iPad

Размер экранного пространства здесь позволил "сэкономить" один уровень вложенности и разместить на одном экране и список разделов, и параметры выбранного раздела. На рис. 1.3 можно заметить еще одну особенность — в iOS не используются древовидные списки. Действительно, сделать навигацию по дереву компонентов с помощью лишь нажатий пальцем было бы не очень удобно. Так что тем, кто привык пользоваться элементом `TreeCtrl` в Windows, придется привыкнуть к его отсутствию здесь. Раз уж речь зашла об элементах Windows, можно заметить, что в iOS также отсутствуют выпадающие списки `ComboBox`. Зато обычные линейные списки в iOS предоставляют гораздо больше возможностей.

Говоря об элементах управления, нельзя не упомянуть о "неэкранных" элементах iOS. Это, например, встряхивание устройства, обработчик которого пользователь может встроить в свою программу. Сложно придумать ему частое применение, в некоторых приложениях оно использовалось для обновления информации вместо кнопки **Refresh**. Впрочем, нельзя не заметить, что такое крупное устройство, как iPad, встряхивать не очень удобно, и подобным способом управления лучше не злоупотреблять.

Кроме встряхивания, пользователю доступны такие экзотические инструменты, как датчик положения, компас и акселерометр. Датчик положения может возвращать 6 значений: портретные и ландшафтные в двух ориентациях, а также два варианта горизонтального положения (экраном вверх и экраном вниз). Применение зависит от фантазии разработчика: например, если пользователь положил устройство на стол, можно отключить фоновый звук и приостановить игру. Подробнее об акселерометре будет рассказано далее, а пока вернемся к традиционным методам ввода и вывода информации. Рассмотрим основные элементы пользовательского интерфейса iOS более подробно.

Панель статуса (Status Bar)

В операционной системе iOS Status Bar находится в верхней части экрана. Панель статуса содержит информацию об устройстве и его окружении: уровень сигнала сети, заряд батареи, текущее время. Разработчик может указать в настройках программы, отображать панель статуса или работать в полноэкранном режиме. Но как рекомендует Apple, "нужно дважды подумать", прежде чем убрать Status Bar в своем приложении, ведь выводимая там информация является важной для пользователя (например, не видя уровня заряда батареи, пользователь может сильно разрядить свой смартфон и остаться без возможности совершить важный звонок).

Панель статуса можно видеть на рис. 1.4.

Панель навигации (Navigation Bar)

Панель навигации также можно видеть на рис. 1.4. Она всегда располагается в верхней части экрана под панелью статуса и, согласно своему названию, обеспечивает навигацию между различными окнами программы. Согласно рекомендациям Apple, надпись на панели навигации должна четко и ясно показывать пользователю его "местоположение" в иерархии интерфейса программы, надписи

должны быть легко читаемыми, их не рекомендуется закрывать дополнительными кнопками или картинками (не нужно забывать, что места на мобильных экранах довольно мало).

Панель инструментов (Toolbar)

В устройствах с iOS Toolbar располагается снизу. Toolbar может использоваться для выполнения каких-либо действий над содержимым текущего окна или для навигации, например содержать кнопки **Save** и **Cancel**, однако его нельзя использовать для переключения видов или вкладок — для этого имеется отдельный компонент Tab Bar. И как уже упоминалось, кнопки должны быть не меньше 44×44 пикселов для удобного нажатия пальцем.

Для выполнения стандартных действий (поиск, создание, удаление, воспроизведение) рекомендуется использовать стандартные значки, предоставляемые Apple, их описание можно найти в разделе документации "Standard Buttons for Use in Toolbars and Navigation Bars".

Пример использования панели инструментов также можно видеть на рис. 1.4.

Панель вкладок (Tab Bar)

Панель Tab Bar имеет такое же назначение, что и в Windows, вид панели вкладок в iOS показан на рис. 1.6.

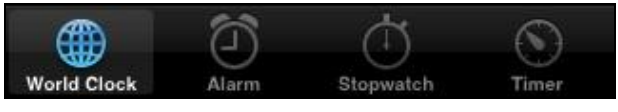

Рис. 1.6. Tab Bar в iOS

В отличие от Windows, панель вкладок всегда располагается в нижней части экрана. Согласно рекомендациям Apple, панель вкладок можно использовать только для переключения между разными окнами, относящимися к одному и тому же объекту. Нельзя использовать панель вкладок для выполнения каких-либо действий, для этого существует панель инструментов Toolbar, описанная ранее. Также не рекомендуется использовать более семи вкладок, это неудобно для восприятия (не говоря уже о том, что большое количество значков просто не поместится на экране iPhone).

Как и в панели инструментов, имеется большое количество системных значков, которые разработчик может использовать для стандартных функций, их можно найти в разделе документации "Standard Icons for Use in Tab Bars". В то же время Tab Bar (как, кстати, и Toolbar) позволяет использовать собственные значки, значки которых должны храниться в формате PNG с прозрачностью.

Всплывающее окно (Popover)

Вообще говоря, в iOS пользователю доступны только полноэкранные диалоговые окна, за исключением системных сообщений и индикаторов загрузки. Но для больших экранов iPad было сделано исключение: начиная с версии 3.2, появилась возможность вывода всплывающих окон, как показано на рис. 1.7.

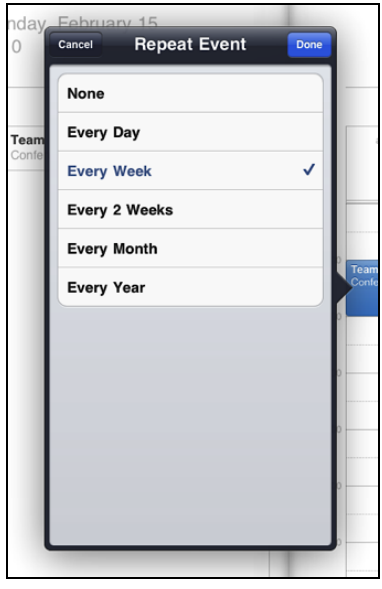

Рис. 1.7. Всплывающее окно в Apple iPad

В отличие от всплывающих окон в Windows, это окно имеет также стрелку, показывающую, откуда оно было открыто, что можно признать вполне удобным. В документации Apple всплывающие окна рекомендуют использовать для отображения более подробной информации в списках, дополнительных настроек и пр. Но при этом рекомендуется выводить окно так, чтобы оно не закрывало важной информации, особенно это касается вывода настроек (к примеру, довольно неудобно настраивать режим вывода изображения с помощью всплывающего окна, если окно настроек заслоняет то самое изображение, и результата не видно). Также согласно рекомендациям Apple, стрелка должна по возможности указывать непосредственно на элемент, к которому относится всплывающее окно (как показано на рис. 1.7).

Совмещенный вид (Split View)

Еще одним элементом управления, используемым исключительно на iPad, является Split View (см. рис. 1.5). С помощью Split View можно совместить два разнообразных элемента навигации, например список писем в левой панели и содержимое выбранного письма в правой, как это делается в популярных почтовых программах.

Таблица (Table View)

Все смартфоны имеют небольшой экран, поэтому основным способом отображения информации является список или таблица данных с возможностью прокрутки. Table View является, наверное, самым развитым из всех компонентов графического интерфейса, по возможностям заметно превосходя аналогичный элемент в Windows. Примеры табличного отображения показаны на рис. 1.8.

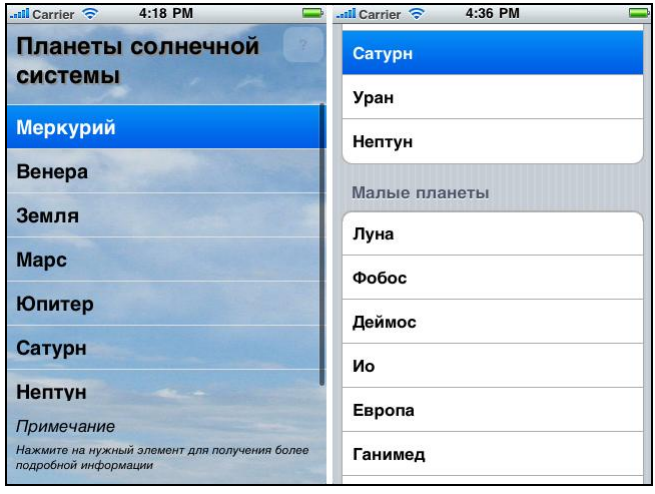

Рис. 1.8. Table View и различные режимы вывода

Как можно видеть, таблица может быть линейной, содержать справа колонку индексов, а также разбиваться на несколько сегментов. Помимо этого каждая ячейка в Table View может являться самостоятельным объектом со своим интерфейсом, что позволяет выводить сложно структурированные данные. Одним из примеров такого сложного табличного представления является показанный ранее на рис. 1.4 экран почтовой программы. Также Table View представляет встроенную поддержку анимации для скроллинга или удаления элементов.

Сообщение (Alert)

Вид окна сообщения показан на рис. 1.9.

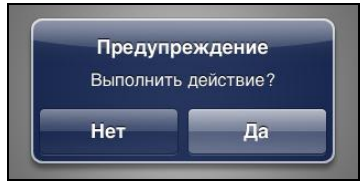

Рис. 1.9. Окно Alert в iOS

Интересно отметить прозрачную "подложку", сквозь которую видно содержимое окна нижнего уровня. Более того, в отличие от довольно-таки простой функции `MessageBox` в среде Windows, класс `UIAlertView` значительно более развит, он позволяет добавлять в сообщения не только несколько кнопок, но и другие элементы, например поле текстового ввода.

Окно действия (Action Sheet)

Это окно применяется для предоставления пользователю выбора из нескольких вариантов действий, что можно видеть на рис. 1.10.

Рис. 1.10. Окно Action Sheet в iOS

На iPhone это окно автоматически располагается в нижней части экрана, на iPad оно открывается во всплывающем окне, что можно видеть на рисунке.

Отображение HTML (Web View)

Для разработчика важно то, что в Web View возможна загрузка не только страниц из сети Интернет с помощью ввода адреса, но также локально сохраненные или программно формируемые HTML-страницы, что позволяет использовать этот компонент для сложно форматируемого вывода, формирования отчетов, таблиц и пр. Подробнее об этом будет рассказано *в главе 4*.

Остальные элементы интерфейса

Текстовое поле ввода (Text View)

Стандартное поле ввода, по "последней моде" современных интерфейсов, имеет закругленные края. Через редактор ресурсов пользователь может настроить размер, тип шрифта и выравнивание, но сделать это можно для всего поля в целом, одновременное использование нескольких шрифтов в одном компоненте невозможно. Как и в других системах, пользователь может настроить вид текстового поля и тип используемых данных, например разрешить вводить только цифры, или только цифры и знаки препинания.

Помимо простого текстового поля, есть и модификации данного компонента. Например, текстовое поле поиска (Search Bar), представляющее собой поле ввода

с размещенными в нем значком и кнопкой **старт**, поле поиска с настраиваемыми параметрами (Scope Bar). Имеется также поле ввода для одиночной строки описания вводимого значения (Text Field).

Индикатор активности (Activity Indicator)

Поскольку в iOS курсора нет, этот элемент является статическим. Разработчик может поместить его в нужной части окна и делать видимым по мере необходимости: ожидание отображается в виде анимированного вращающегося круга.

Поле ввода даты и времени (Date and Time Picker)

Вид поля ввода даты в iOS показан на рис. 1.11. Как можно видеть, предлагаемый операционной системой компонент занимает весьма большое экранное пространство, поэтому зачастую целесообразнее использовать обычные текстовые поля для ввода даты или времени в цифровом формате.

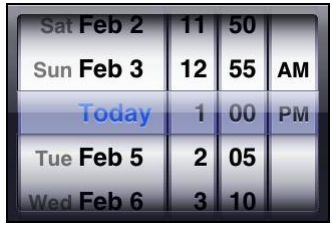

Рис. 1.11. Окно **Date and Time Picker** в iOS

В iOS, помимо Date and Time Picker, существует и компонент Picker, позволяющий выбирать произвольные величины из списка.

Выбор из нескольких вариантов (Segmented Control)

Вид этого элемента управления показан на рис. 1.12.

Рис. 1.12. Окно **Segmented Control** в iOS

Он предоставляет пользователю возможность выбора только одного из вариантов.

Переключатель (Switch)

Вид этого компонента интерфейса представлен на рис. 1.13.

Рис. 1.13. Элемент управления Switch в iOS

Остальные элементы управления, такие как индикатор прогресса (Progress View), компоненты для скроллинга, вывода графических изображений, здесь не рассмотрены, их можно найти в редакторе ресурсов, предоставляемых средой разработки XCode или в документации на сайте Apple. Также разработчик может самостоятельно создавать элементы интерфейса.

Возможности и ограничения платформы Apple iOS

Самое время задаться вопросом о возможностях и ограничениях операционной системы для разработчика. Ключевым является принцип "песочницы" (Sandbox), по аналогии с детской игровой площадкой:

- программа может делать что угодно в пределах своей "песочницы";
- программа не может "залезть" в песочницу другой программы.

Мы можем работать с файлами, графикой, данными из сети Интернет, создавать модель нейронной сети или распознавать речь. Но мы не можем сохранить созданный нами файл в папку другой программы или обратиться напрямую к системным файлам. Хорошо это или плохо? Это хорошо, в первую очередь, тем, что защищает систему от вирусов и просто некачественных программ. Отсутствие полноценного межпрограммного взаимодействия вряд ли можно считать критичным для iOS, ведь не надо забывать, что это все-таки смартфон, а не полноценный компьютер. И вряд ли кто-либо будет использовать на смартфоне сложные программные комплексы. Что касается возможностей iOS для разработчика, то они весьма обширны. Разработчик может использовать в своих программах следующие технологии:

- операции чтения и записи данных в файлы и каталоги (но только в пределах собственной папки, как упоминалось ранее);
- операции работы с сетью как на высоком уровне (загрузка Web-страниц), так и на низком (sockets);
- 2D-графика посредством встроенных функций операционной системы;
- 2D- или 3D-графика посредством OpenGL ES, включая даже поддержку шейдеров;
- низкоуровневый доступ к процедурам записи и воспроизведения звука, позволяющий делать обработку аудиоданных или их визуализацию;
- ассемблерные вставки процессора ARM для более высокого быстродействия наиболее ответственных фрагментов программы;
- набор функций для аппаратной поддержки математических вычислений (таких как работа с векторами, преобразования Фурье), доступные в iOS 4.0 (accelerate framework).

Но в упомянутой бочке меда есть ложка дегтя, даже, скорее, две:

- сложность работы со сторонними файлами создает определенные неудобства для разработчиков, например пользователь не может штатными способами загрузить в программу файлы по беспроводной сети. Встроенная программа синхронизации Apple iTunes позволяет осуществлять синхронизацию только по кабелю, что является атавизмом в эпоху домашних и рабочих сетей Wi-Fi. По-

этому некоторые программы имеют свой встроенный Web-сервер, позволяющий пользователям загружать файлы через браузер с любого компьютера локальной сети. Пока что использование HTTP является практически единственным способом беспроводной коммуникации, хотя есть надежда, что здравый смысл восторжествует, и в новых версиях iTunes беспроводная синхронизация все же появится;

- операционная система iOS не позволяет запускать программы, не подписанные специальным сертификатом разработчика. Для его получения требуется членство в iPhone Developer Program, стоимость которого составляет 100 долларов в год. Таким образом, даже написав собственную программу, владелец iPhone не сможет просто взять и загрузить ее в устройство. Впрочем, в сети Интернет описываются и бесплатные способы создания сертификата для личного использования. Все программы можно без ограничений запускать на эмуляторе, который не требует сертификатов и распространяется бесплатно.

Конечно, эти ограничения являются неприятными, но далеко не фатальными. То, что ограничений для фантазии практически нет, можно видеть на примере Parrot AR Drone, описание которого можно найти на сайте **http://ardrone.parrot.com**.

Это полноценная летающая радиоуправляемая модель квадрокоптера, в качестве пульта к которой используется iPhone или iPad. Обмен данными осуществляется с помощью Wi-Fi, передатчик которой установлен на борту модели. Благодаря технологии Multi-touch владелец смартфона может управлять аппаратом, а встроенная видеокамера передает на iPhone изображение прямо с модели. Помимо стандартной программы управления полетом имеется также комплект средств для разработчиков, позволяющий, например, устраивать воздушные бои.

После рассмотрения подобного устройства самое время перейти к рассмотрению способов взаимодействия iOS с внешним миром.

Интерфейсы сети и внешних устройств

Как и любое современное устройство, iPhone и iPad имеют основные интерфейсы.

- Все устройства с iOS (iPhone, iPad и iPod Touch) имеют встроенный модуль Wi-Fi, позволяющий подключиться к беспроводной сети. Находящееся в сети устройство имеет собственный IP-адрес, что используется многими прикладными программами. Например, популярная программа Air Video просмотра видео в различных форматах состоит из сервера, формирующего видеопоток, и клиента на iPhone или iPad.
- Все смартфоны iPhone и некоторые модели iPad (с префиксом 3G) имеют GSM-модем. Знать, какой тип беспроводного соединения используется в конкретный момент, может быть полезно для экономии трафика, Apple предоставляет специальные классы, позволяющие учитывать это в прикладных программах.
- Все устройства имеют Bluetooth, однако его поддержка на уровне iOS ограничена. В то же время iOS имеет встроенный протокол GameKit, позволяющий на-

прямую соединять два устройства для передачи данных между ними. В Apple рекомендуют использовать GameKit для обмена короткими посылками данных объемом не более 1000 байт. Этого вполне достаточно, например, для несложных игр.

Другие интерфейсы

Помимо стандартных средств ввода-вывода, Apple стала одной из первых компаний, предоставившей пользователям новый уровень взаимодействия с портативным устройством. Например, наклон устройства или его встряхивание теперь могут использоваться для выполнения разнообразных действий в программе. Пользователю и разработчику iOS предлагает немало возможностей, способных значительно расширить функциональность смартфона.

Камера

Последние модели iPhone 4 снабжены камерой, оснащенной фотодиодной вспышкой, также возможна запись видео. Интересным приложением, например, для iPhone является программа чтения и распознавания штрихкодов.

К сожалению, iPad первой модели и ряд моделей iPod Touch не имеют камеры, поэтому разработчик должен предусмотреть проверку наличия камеры перед использованием соответствующих функций.

Акселерометр

Немаловажным устройством в iOS является акселерометр. Датчик измеряет наклон корпуса устройства относительно земли, и разработчик может учитывать эти данные в программе игр вроде гонок, когда наклон устройства используется для поворота руля или лабиринта, где нужно закатить шарик.

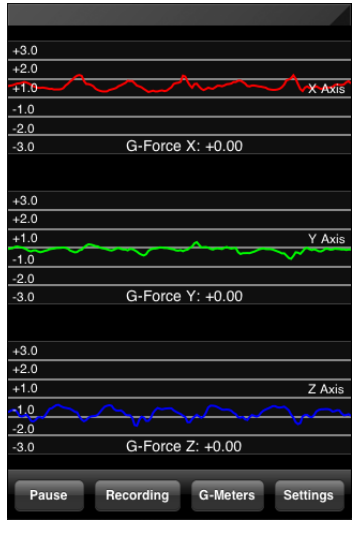

Рис. 1.14. Тестирование акселерометра при ходьбе с помощью AccelGraph

Частным случаем использования акселерометра являются изменение ориентации, которое автоматически отслеживается операционной системой, и встряхивание устройства. Для тестирования акселерометра есть удобная программа AccelGraph, которая распространяется бесплатно. Акселерометр можно использовать и достаточно необычным образом: например, с его помощью можно отслеживать даже сотрясение устройства во время ходьбы, что использовано в программе Шагомер.

Как показало тестирование в программе AccelGraph, ходьба со смартфоном действительно хорошо видна на графиках, что можно видеть на рис. 1.14.

Трехосевой гироскоп

Одним из нововведений в iPhone 4 является трехосевой гироскоп. Обычный акселерометр измеряет лишь углы наклона устройства относительно земли и поэтому не может реагировать на определенные виды движения, например на поворот игрока вокруг своей оси. Гироскоп в отличие от акселерометра, измеряет *угловое ускорение* относительно самого устройства, что дает еще одну степень свободы. Например, теперь можно поворачиваться в разные стороны вместе со смартфоном, и виртуальная камера будет поворачиваться в соответствующем направлении. Пример такой реализации можно посмотреть в игре "Eliminate Gun Range", сделанной специально для iPhone 4. Эта технология также может использоваться в проектах "дополненной реальности", позволяющих смартфону отображать объекты, "наложенные" на окружающую среду. Например, при наведении камеры смартфона на дом на экране устройства отобразится список компаний, находящихся в этом доме. Пока что эта технология еще развивается, в качестве одного из примеров реализации можно назвать бесплатную программу Layar, загрузить которую можно с сайта **http://www.layar.com**.

Компас

Еще одним датчиком для связи с внешним миром является компас. В отличие от акселерометра, ему сложно найти какое-либо реально полезное применение (за исключением программ навигации). К тому же, по личному опыту автора, работа компаса довольно неустойчива, стрелка часто колеблется и показывает не туда, в общем, обычный компас со своей задачей показывать направление на север справляется заметно лучше. Хотя для навигационных программ использование компаса для ориентации карты может быть весьма актуально.

GPS

Технология GPS давно уже стала привычной, тем не менее, без упоминания о ней этот раздел был бы неполным. Новые версии iPhone имеют встроенный блок GPS, а вот iPad имеет GPS-модуль лишь в 3G-версии.

В iOS встроен механизм определения координат даже на устройствах без модуля GPS — определение положения по IP-адресу с точностью до дома (естественно, при подключении к сети Интернет через домашнюю, а не сотовую сеть).

На этом мы закончим теоретическое введение и перейдем к практической части — изучению программирования для iOS.

Глава 2

Программирование для iPhone и iPad — первые шаги

Краткая информация об операционной системе Mac OS

Причем тут Mac OS, может подумать читатель, если книга описывает программирование для iPhone. На эту тему у автора для читателя есть две новости, одна плохая, одна хорошая. С какой начать? Плохая новость состоит в том, что среда разработки для iPhone работает в операционной системе Mac OS. Поэтому для освоения программирования iPhone придется изучить Mac OS, хотя бы на минимальном уровне, чтобы установить и запустить среду разработки XCode. Хорошая новость состоит в том, что благодаря программе эмуляции виртуальной машины VMWare установить Mac OS можно даже без покупки отдельного компьютера. В любом случае, познакомиться с этой системой придется, и это весьма интересно.

Mac OS была первой операционной системой, предоставляющей пользователям полноценный и дружественный графический интерфейс. Достаточно посмотреть на рис. 2.1: трудно поверить, что это было в 1984 году.

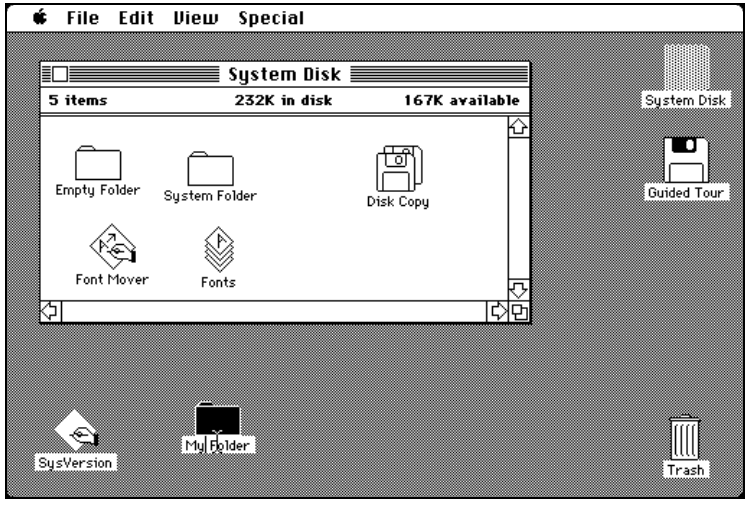

Рис. 2.1. Рабочий стол Mac OS, 1984 г., фото с сайта **http://ru.wikipedia.org**

Уже тогда в Mac OS была поддержка длинных имен файлов. В настоящее время трудно сказать, почему в итоге система Macintosh не стала популярной, да сейчас это уже и не столь актуально. Гораздо интереснее обратиться к системе Mac OS сегодняшнего дня. Здесь у пользователя есть три варианта действий.

- Установка Mac OS на виртуальной машине VMware. Самостоятельно настроить Mac OS для такой установки непросто, однако в сети Интернет можно найти уже готовый образ. Достаточно скачать и установить программу VMware Player, которая распространяется бесплатно — можно просто поискать в сети Интернет файл VMware-player-3.1.3-324285.exe, который и является инсталлятором этой программы. Размер образа Mac OS Snow Leopard.7z составляет около 10 Гбайт, для его работы требуется иметь компьютер с объемом памяти не менее 2 Гбайт и процессором не ниже, чем Core2Duo. В такой конфигурации скорость работы системы в виртуальной машине вполне достаточна для написания программ и подходит даже для разработки несложных игр.
- Установка Mac OS на обычный компьютер или ноутбук. Этот способ предпочтительнее первого по быстродействию, однако не так прост. В штатном варианте поставки система Mac OS привязана к BIOS компьютера производства Apple и на других системах просто не заработает. Существуют способы отключить эту привязку: в сети Интернет можно найти уже настроенные диски с системой, иронично называемой "Hakintosh". Выходом в этом плане может быть покупка компьютера такой же конфигурации, что и у оригинального Apple, в этом случае количество проблем минимально, а цена такого компьютера может быть в 1,5−2 раза меньше оригинального.
- Третий способ — просто пойти и купить компьютер или ноутбук Apple. Самым дешевым является Mac Mini. Этот настольный компьютер имеет небольшие размеры (вес около 1,5 кг) и достаточные для работы характеристики: процессор Core2Duo, 2 или 4 Гбайт памяти, видеокарта GeForce 320M. Ноутбуки MacBook стоят заметно дороже, да и клавиатура MacBook для пользователя персональных компьютеров несколько непривычна. Кстати, все компьютеры Macintosh позволяют устанавливать Windows в качестве второй операционной системы, так что купленный компьютер в любом случае пригодится, даже если не использовать Mac OS в дальнейшем.

Таким образом, существуют различные варианты действий на любой вкус и кошелек. Внешний вид операционной системы Mac OS показан на рис. 2.2.

Для тех, кто до этого использовал только Windows, полезно запомнить следующее:

- аналогом кнопки **Пуск** в Mac OS является яблоко в верхнем углу экрана;
- проводником в Mac OS является Finder;
- переключение раскладки между русским и английским языками выполняется комбинацией клавиш <Cmd>+<Пробел>;
- для работы с буфером обмена используются сочетания клавиш <Cmd>+<C> и <Cmd>+<V>;
- на клавиатуре нет отдельной клавиши <Delete>, ее заменяет сочетание клавиш <Fn>+<Backspace>;

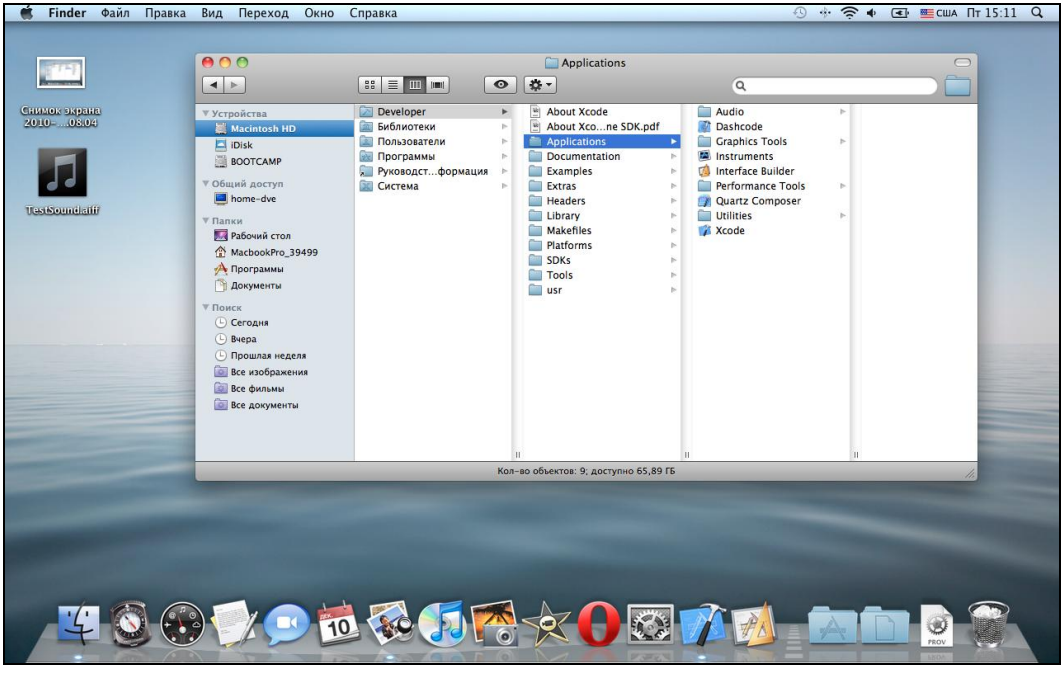

Рис. 2.2. Рабочий стол Mac OS версии Snow Leopard

- в качестве браузера по умолчанию используется Safari, хотя автору привычнее браузер Opera, который выглядит практически одинаково под Mac OS и Windows;
- для того чтобы сделать скриншот экрана, нужно нажать <Cmd>+<Shift>+<3>.

Да простят меня поклонники Mac за столь краткое изложение, но для первого знакомства с Mac OS перечисленного вполне достаточно. Интересующиеся более подробно устройством системы, могут обратиться к специальной литературе или online-документации.

Следующим необходимым шагом является установка среды разработки XCode. Ее установка бесплатна, достаточно регистрации на сайте **http://developer.apple.com**. Размер дистрибутива составляет около 2 Гбайт. Но перед тем как работать с XCode, настало время ознакомиться с основным языком программирования для iOS — Objective-C.

Основы языка Objective-C

Согласно сведениям из Википедии, язык Objective-C был создан в 1986 г. как объединение двух языков — C и Smalltalk. Язык Objective-C специально разрабатывался так, чтобы облегчить его изучение для C-программистов. И еще хорошая новость — язык Objective-C полностью совместим с языком C, т. е. все функции на языке C можно вставить в проект для iOS. Мы не будем вдаваться в теоретические особенности языка, а сразу перейдем к практике.

Объявление переменных

Как уже говорилось, мы можем создавать переменные как в C-стиле, так и в стиле Objective-C. Рассмотрим листинг 2.1.

Листинг 2.1. Объявление переменных

```
int i = 0;
float pi = 3.1415f;
int data1[4] = { 1, 2, 3, 4 }, data2[4] = {0};
BOOL res1 = TRUE;
Boolean res2 = YES, res3 = NO;
```

Логические переменные можно записывать как `Boolean` (ObjC-стиль), а можно как `BOOL` (WinAPI-стиль), кто к чему привык. Рассмотрим теперь более сложные случаи, которые пригодятся нам на практике (листинг 2.2).

Листинг 2.2. Объявление переменных (продолжение)

```
// прямоугольник
CGRect rect = CGRectMake(0, 0, 320, 480);
// точка
CGPoint point = CGPointMake(100, 100);
// строковые константы
NSString *str1 = @"This is a test string", *str2 = @"100";
// конвертация строки в число
int str2_val = [str2 intValue];
// конвертация числа в строку
NSString *pi = [NSString StringWithFormat:@"%f", pi];
```

Структуры `CGRect` и `CGPoint` используются при работе с пользовательским интерфейсом. Например, чтобы сдвинуть кнопку или изображение в новое место, нужно изменить параметр `center`, который описывается как `GPoint`, а чтобы задать размеры, нужно изменить параметр `frame`, который описывается как `CGRect`.

Сами структуры описываются в листинге 2.3.

Листинг 2.3. Файл CGGeometry.h

```
/* Points. */

struct CGPoint {
  CGFloat x;
  CGFloat y;
};
```

```
typedef struct CGPoint CGPoint;

/* Sizes. */

struct CGSize {
  CGFloat width;
  CGFloat height;
};
typedef struct CGSize CGSize;

/* Rectangles. */

struct CGRect {
  CGPoint origin;
  CGSize size;
};
typedef struct CGRect CGRect;
```

Объявления похожи на структуры `RECT` и `POINT`, используемые в WinAPI, за исключением того, что для координат используются числа типа `float`.

> ***Примечание***
>
> Кстати, для любого элемента интерфейса, даже для кнопки, можно задать матрицу преобразования. Более подробно об этом будет рассказано далее.

Со строками ситуация чуть сложнее. Если `CGRect` и `CGPoint` — обычные структуры, то `NSString` — это класс, содержащий свои функции и имеющий разные методы инициализации (как метод `intValue`). О функциях в Objective-C стоит рассказать более подробно.

Функции в Objective-C

Поскольку Objective-C совместим с языком C, код листинга 2.4 корректен.

Листинг 2.4. Функции в C-стиле

```
int sum(int a, int b)
{
      return a+b;
}
// …
int a =10, b = 20;
NSString *s_sum = [NSString StringWithFormat:@"Сумма: %d", sum(a.b)];
```

Функции в стиле Objective-C описываются в листинге 2.5.

Листинг 2.5. Описание функций

```
// сумма 2 чисел
- (int) summa: (int)a: (int)b
{
      return a+b;
}
// преобразование строки в число
- (int) StrToInt: (NSString*)str
{
      return [str intValue];
}
```

Вызов функции в Objective-C осуществляется с помощью квадратных скобок. Две строки следующего кода делают одно и то же:

```
int i1 = 10, i2 = 20;
int res1 = sum(i1, i2), res2 = [self summa: i1: i2];
```

Как нетрудно догадаться, `self` является аналогом this в C++, он используется для вызова функций текущего класса. При вызове функций какого-либо объекта в квадратных скобках сначала ставится его имя, что можно видеть в приведенном примере `[str intValue]`. Здесь вызывается метод `intValue`, принадлежащий объекту `str`, что было бы эквивалентно записи `str.intValue()` в C++. Строка `[NSString StringWithFormat…]` означает вызов статической функции, что было бы эквивалентно следующей записи в C++: `NSString::StringWithFormat(…)`. Синтаксис отличается, а суть фактически остается той же самой.

Раз уж мы коснулись классов, пора перейти к их рассмотрению более подробно. Тем более что ни одна из современных систем программирования не обходится без них.

Классы в Objective-C

Поскольку речь пойдет об объектно-ориентированном программировании, нужно создать какой-нибудь полезный объект. Создадим класс для сравнения скорости эмулятора и реального устройства с iOS — метод с большим количеством вычислений, в качестве результата будет возвращаться время его работы. Итак, приступим (листинг 2.6).

Листинг 2.6. Класс измерения скорости работы

Файл SpeedCheck.h:

```
#import <Foundation/Foundation.h>
@interface SpeedCheck : NSObject
{
       // Время выполнения
       float fResults;
```