Содержание

Об авторах

Крис Адамсон – независимый писатель, редактор и разработчик. Проживает в Гранд Рапидс, штат Мичиган. В настоящее время интересуется разработкой для iOS и Mac, является соавтором книги *iOS SDK Development* (Pragmatic Programmers, 2012). Также является автором книги *QuickTime for Java: A Developer's Notebook* (O'Reilly Media, 2005) и соавтором *Swing Hacks* (O'Reilly Media, 2005). Ранее работал редактором на сайтах *java.net* и *ONJava.com*. Консультационную и издательскую деятельность осуществляет через свою компанию Subsequently and Furthermore, Inc., специализирующуюся на разработке пользовательских интерфейсов и цифрового мультимедийного материала для приложений на платформах Mac и iOS. Ведет блог по разработке ПО для работы с мультимедиа (www.subfurther.com/blog). Ранее был автором текстов и помощником продюсера на кабельном телевизионном новостном канале *CNN Headline News*. С годами стал счастливым обладателем 11 с половиной Mac'ов.

Кэвин Авила (известный также под ником dogbert) – удачная смесь углерода, кислорода, водорода и азота с добавлением небольшого количества разных примесей. Попутно имеет за плечами более 15 лет разработки для Mac и для iPhone – с момента его появления. Кэвин засветился в каждом уголке рынка аудио – от инженера в Apple до настройки профессиональных студий звукозаписи. В настоящее время пишет код по заказу различных клиентов, посиживая в нижнем белье у себя дома и попивая кофе.

Предисловие

Задумайтесь о своем ремесле. О тех, кто на протяжении минувших столетий разделял аналогичный опыт. О художниках, доводивших себя до исступления, пытаясь найти внутри себя источник, который позволил бы создать что-то значимое. О промышленниках, веривших, что стоят у порога новой эры, которую сами и заложили.

Подумайте о древних поборниках магии, стремящихся овладеть силами, скрытыми в тайных знаниях. А затем вспомните, как однажды, овладев секретами уличного фокусника – например, потоком управления и структурами данных, – вы наконец получили доступ к библиотекам API. Представьте себе глаза ученика мага, широко распахнувшиеся и заблестевшие от зрелища неисчерпаемых возможностей, открывшихся в комнате, полной пахнущих плесенью книг.

Это один из ключевых моментов в карьере любого программиста, его можно сравнить с восхождением на вершину холма, с которой только открывается вид на гору вдалеке. В этот момент с пугающей ясностью осознаешь, что программирование – это путешествие длиною в жизнь, на каждом этапе которого предстоит учиться. Многие незадавшиеся волшебники просто стремглав выбежали бы из комнаты, оставив этот страшный мир ради более спокойного времяпрепровождения.

Но раз вы читаете эти страницы, то принадлежите к немногим избранным, у которых имеется генетическая предрасположенность к решению трудных задач. Именно они пересекают порог и входят в открывшийся мир, без остатка посвящая себя изучению новых заклинаний и исследованию новых видов магии.

Ибо что такое программирование, как не магия? Произнесение тайных слов для управления могучими силами, которые вы едва понимаете, призывание духов ушедших волшебников, чтобы облегчить свое бремя, упрощение последовательностей действий и – в качестве награды – способность вызывать восторг и удивление непосвященных.

Раздумывая над очередным томом, в котором описываются новые виды магии, вы начинаете комбинировать их с тем, что уже знаете,

и так рождаются новые заклинания, открывающие новые возможности. Все познанное ранее мостит дорогу к новым знаниям, дорога разветвляется снова и снова, пока в один прекрасный день не упирается в тупик.

Во многих древних текстах упоминается темное искусство программирования звука, но нигде не дается ясных указаний. И какой бы важной ни была технология Core Audio, вы не найдете ни посвященных ей книг, ни свитков с заклинаниями, ни даже простого примера кода, чтобы попрактиковаться. Программировать звук желательно в разных случаях, но коль скоро это черная магия, то сведения о ней скудны и непонятны.

Крис Адамсон нашел человека, практикующего звуковые искусства. Имя этого седеющего мудреца, сведущего в C, – Кэвин Авила. Применяя попеременно подкуп и честное расспрашивание, Крис сумел набиться к нему в подмастерья. Кроличья нора, в которую он проник, не имеет конца, но, распознавая темные повороты и закоулки лишь по слуху, он выучил новый язык для описания звука, а вместе с ним и новый взгляд на мир.

Спустя вечность, он – сам уже седеющий мудрец – снова сидел в библиотеке, размышляя об отсутствующих томах, и в этот момент осознал свое предназначение – пролить свет на темное искусство Core Audio. Само определение мастерства предполагает, что мы должны обучать тому, чему научились сами. Именно так устроено круговращение от подмастерья к мастеру. Именно этот механизм стремит поколения вперед, так что каждое проходит дальше по пути к той неуловимой маячащей впереди точке, которую мы называем будущим.

Как и любой обряд посвящения, задача оказалась титанически трудной, потребовала совершенно нового набора навыков и иного способа мышления. Мы должны вывернуть наизнанку и себя, и свои знания, чтобы найти не только истину, но и скрытую за ней красоту, которая оставляет отклик в душах на протяжении многих веков.

Все для того, чтобы в какой-то неведомый момент в будущем там, где когда-то был тупик и незаполненный промежуток между Core Animation и Core Data, некий юный ученик смог обрести мудрость и учительское наставление. Чтобы он смог объединить новое знание с тем, что ему уже известно, и подготовиться к встрече с собственными тупиками и собственным темным искусством.

Этот момент, любезный читатель, настал. Юный ученик – это ты, а колдовская книга, которую ты держишь в руках, содержит всю

мудрость и более чем достаточно заклинаний, чтобы ты мог перейти на следующую ступень постижения магии. Эта книга – твой ключ к овладению невыразимой силой, силой звука и природы, силой Core Audio.

Что, слишком много слов для какой-то книги о программировании звука? Не усмехайтесь скептически, я, пожалуй, расхвалил ее даже недостаточно. Звук – невероятно мощная штука по силе воздействия на мозг человека, которое мы понимаем очень плохо. Подумайте о влиянии музыки на свою жизнь. А теперь учтите, что вся на свете музыка составляет жалкие 10% от того, что можно сказать о звуке.

Возможности программирования звука далеко не ограничиваются тем, что можно воспринять на слух. Раньше для считывания информации с кредитной карты был необходим дорогостоящий прибор. Теперь для этого достаточно иметь дешевый пластиковый электронный ключ, вставляемый в гнездо для наушников в iPhone. Чтобы творить чудеса со звуком, не нужна музыка.

Первый образчик кода для Core Audio вы встретите уже в главе 1 этой книги «Общие сведения о Core Audio», когда только будете знакомиться с тем, что такое технология Core Audio и когда имеет (а когда не имеет) смысл прибегать к ее мощи.

Core Audio, как и все темные искусства, корнями своими уходит в свойства природы. Глава 2 «Кое-что о звуке» повествует о звуке не как о природном явлении, а как о науке. Вы узнаете о языке и технике, позволяющих переводить колебания молекул воздуха в математические формулы, понятные компьютеру, и наоборот.

Вы также узнаете о терминологии, применяемой в работах, посвященных звуку, и, в частности, о том, что же на самом деле означают технические термины, которые вам доводилось слышать, а то и использовать на протяжении многих лет: шаг дискретизации, частота кадров, буфер и сжатие. Эта тематика будет продолжена в главе 3 «Обработка звука с помощью Core Audio», где будет сорван покров тайны со звуковых форматов и рассказано о канонических форматах, используемых на внутреннем уровне Core Audio.

Познакомившись с основами Core Audio, вы, несомненно, захотите применить новые умения, освоив такие дешевые трюки, как запись (глава 5 «Запись») и воспроизведение (глава 6 «Воспроизведение»), с помощью высокоуровневой архитектуры Audio Queue.

Разумеется, слово «высокоуровневый» может показаться неудачным, особенно если у вас за плечами опыт работы с таким объектно-

ориентированным каркасом, как Cocoa. Расстаться с комфортным уютом Objective-C и вступить в хладные чертоги чистого C – испытание не для слабых духом, но, немного освоившись, вы увидите, что и каркас программирования на C может очень напоминать Cocoa – просто старые друзья типа пар ключ–значение предстают в новых одеяниях.

Разобравшись с очередями Audio Queue, вы уже – почти – готовы претендовать на звание магистра звуковых форматов. Однако сначала надлежит завершить квест и научиться преобразовывать из одного формата в другой, а заодно понять место канонических форматов в общей картине.

А теперь пора проститься с высокоуровневыми подпорками, надеть гидрокостюм и погрузиться в пучины Core Audio, где скрываются модульные аудиоблоки (Audio Units), с помощью которых и вершится волшебство. Прочитав главу 7 «Аудиоблоки: генераторы, эффекты и рендеринг» и главу 8 «Аудиоблоки: ввод и микширование», вы либо станете программистом звука, либо порвете с этим делом, потому что именно здесь рассказывается, как создавать сквозные решения, для чего не существует «легких путей».

Если раньше речь шла просто о воспроизведении, то теперь пора заняться пространством. В главе 9 «Позиционный звук» вы научитесь изменять звучание путем размещения источников звука в пространстве с помощью технологии трехмерного звука OpenAL.

Своими корнями технология Core Audio уходит в Mac, но развивалась вместе с Apple. В главе 10 «Core Audio в iOS» речь пойдет только о системе iOS и о новых задачах и изменениях, которые потребовались в пришедшем на смену ПК мире сверхпортативных устройств со сверхэффективным программным обеспечением.

Мобильные устройства – не единственный способ вывести звук за пределы компьютера. В главе 11 «Core MIDI» вы научитесь подключать компьютер к музыкальным инструментам и другому оборудованию, применяя включенную в Core Audio реализацию стандарта Musical Instrument Digital Interface (цифровой интерфейс музыкальных инструментов) – Core MIDI.

На этом квест походит к концу, но путешествие только начинается. В главе 12 «Кода» вы заглянете в будущее, уже вооруженные знаниями, необходимыми для понимания таких ранее неприступных вещей, как цифровая обработка сигналов и дискретизация.

Если вы хотите стать истинным магистром темных искусств, то перед вами долгая дорога. Нет смысла подслащивать истину: эта

дорога будет трудной. Но не падайте духом – вы в надежных руках. Авторы говорят простым языком и приводят достаточно примеров кода, чтобы изгнать демонов и показать логические основы технологии, которые помогут освоить ее идеи и понятия.

Core Audio – самая мощная система программирования звука из когда-либо созданных человеком, но ее возможности до сих пор оставались недостижимыми для большинства разработчиков приложений, а хранились под замком в мозгах таких корифеев, как Кэвин. Крис сделал то, что до него никому не удавалось и вряд ли кто сумеет повторить: объяснить Core Audio понятными словами.

Эта книга потребовала нескольких лет титанической работы и внимания лучшего в отрасли технического редактора – легендарного Чака Топорека, а также его талантливых коллег в издательстве Pearson. Люди, в чьи трепещущие от долгого ожидания руки попадет этот колдовской том, станут авторами новой волны потрясающих аудиоприложений.

Представьте только, какие новые возможности заинтересовать человека открываются благодаря магии звука. Это невероятное будущее предстоит открыть вам. Мы стоим на заре новой эры волшебства в компьютерах, и волшебник – вы. Овладение всеми каркасами подсистемы Core Audio изменит ваши представления о мире.

Майк Ли, Амстердам

Введение

Mac – великолепный мультимедийный компьютер, а iPhone – лучший в мире iPod, но за счет чего это достигается? Каким образом некоторые из первых приложений для iOS превратили iPhone в виртуальный музыкальный инструмент, тогда как разработчики на других мобильных платформах счастливы уже тем, что «натянули новую шкуру» на простой MP3-плеер? Почему именно Mac выбирают многие профессионалы в области цифрового мультимедиа, и благодаря каким секретам удается создавать такие приложения, как Bias Peak, Logic и Soundtrack Pro?

Подсистема Core Audio – вот ответ.

Core Audio – это низкоуровневый API, предоставленный Apple для работы с цифровым звуком в системах Mac OS X и iOS. В ней имеются средства для одновременной обработки нескольких потоков многоканального цифрового звука и интерфейсы к аудиооборудованию для ввода (микрофоны) и вывода (динамики и наушники). Core Audio позволяет писать приложения, которые напрямую работают с несжатыми звуковыми данными, полученными от микрофона, применять к ним эффекты, микшировать с другими звуковыми потоками и либо выводить результат в динамики, либо преобразовывать его в сжатый формат, который затем можно сохранить в виде файла или отправить по сети. Для тех, кто не занимается разработкой полнофункциональных приложений, Core Audio позволяет написать нестандартный эффект и обернуть его подключаемым модулем, который называется аудиоблоком (*audio unit*). Это даст другим пользователям возможность добавить ваш эффект в свое приложение на основе Core Audio.

Apple впервые включила Core Audio в операционную систему Mac OS X 10.0, где она в конечном итоге заменила подсистему SoundManager, входившую в состав классической Mac OS. Поскольку API подсистемы Core Audio основан на C, его можно использовать как в приложениях Cocoa, написанных на Objective-C, так и в приложениях Carbon на C++. Можно даже вообще обойтись без каркасов и

обращаться к Core Audio из командной утилиты, написанной на чистом C с соблюдением спецификации POSIX (на самом деле большинство примеров в этой книге именно так и устроены). Будучи написана на C, подсистема Core Audio демонстрирует чрезвычайно высокую производительность, что критически важно для обработки сотен и даже тысяч звуковых отсчетов в секунду.

В основе Core Audio лежит идея потоков звуковых данных, то есть непрерывных последовательностей данных, представляющих звуковой сигнал. Поскольку звук изменяется во времени, изменяются и описывающие его данные. В Core Audio взаимодействие со звуком осуществляется главным образом за счет работы с потоками: получение потоков из файлов или от устройств ввода, их микширование, преобразование в различные форматы, отправка устройствам вывода и т. д. Для этого ваша программа обращается к Core Audio или – наоборот – Core Audio выполняет обратный вызов вашей программы всякий раз, как появляются новые данные для обработки. Эта метафора отличается от тех, что вы могли встречать в других мультимедийных API. Простые мультимедийные плееры типа тега HTML5 `<audio>` или элемента iOS `AVAudioPlayer` рассматривают источник звука (например, файл или URL) как черный ящик: его можно воспроизвести, поставить на паузу или остановить, быть может, даже перемотать вперед или назад, но ни заглянуть внутрь данных, ни что-нибудь сделать с ними возможности нет. А прелесть и мощь Core Audio в том и заключаются, что она позволяет *манипулировать* данными.

Ах, если бы это еще и было просто...

Core Audio имеет заслуженную репутацию одной из самых трудных подсистем Mac OS X и iPhone. А объясняется это тем, что для оперирования на таком уровне требуется решать массу необычных задач: работать с потоками отсчетов в естественном формате, работать с моделями программирования Core Audio, характеризующимися высокой степенью асинхронности, и делать все достаточно быстро, чтобы не заставлять Core Audio ждать, когда ей необходимо послать данные динамикам или наушникам. Не облегчает жизни и тот факт, что в версии iPhone OS 2.0 – первой, которая поддерживала сторонние приложения, – Core Audio была *единственной* мультимедийной подсистемой; даже если разработчик хотел всего лишь воспроизвести файл, ему все равно приходилось опускаться на уровень потоков и обрабатывать отсчеты вручную – на C. Если вы хотите и готовы работать на таком уровне, все замечательно, но

программисты, нуждавшиеся в более простой высокоуровневой абстракции, открыто и *громко* возмущались.

В Core Audio никто специально не встраивал «жестокость» или «тупость». Она сложна, потому что такова предметная область. На наш взгляд, веб-приложение для сохранения покупок в базе данных – тривиальная задача по сравнению с моделированием звуковых волн потоком отсчетов, применением к ним эффектов за счет математических манипуляций и доставкой результатов оборудованию несколько сотен или тысяч раз в секунду, да еще так, чтобы пользователь не заметил никакой задержки. Производить трудные операции по-настоящему быстро – вообще сложно, и надеемся, к концу этой книги вы сумеете оценить, сколь много в этом смысле делает за вас Core Audio.

И думается, к этому моменту вы уже сумеете написать кое-что интересненькое самостоятельно.

На кого рассчитана эта книга

Никакая книга не может быть полезной для всех, поэтому будет лучше сразу поставить условия: эта книга задаст вам жару. Но еще Ницше говорил: «То, что не убивает нас, делает нас сильней». Освоив изложенный материал, вы сами сможете задавать жару, решая серьезные задачи.

Кому имеет смысл читать эту книгу

Основная аудитория – опытные программисты, знакомые с Mac или iOS, но еще не приступавшие к исследованию Core Audio. Знание языка C предполагается, однако предварительных знаний о цифровом звуке не требуется, об этом мы расскажем в главе 2. Мы считаем, что интерес к книге по программированию звука возник у вас не на пустом месте, а после работы с достаточным количеством мультимедийных приложений, так что вы представляете, что они вообще умеют делать: получение звука из внешнего источника, наложение эффектов в реальном масштабе времени, воспроизведение MP3-файлов, виртуальные музыкальные инструменты, веб-радио, передача голосу по протоколу IP (VoIP) и т. д. Если сама мысль обо всех этих вещах не щекочет ваше программистское нутро, что ж – где-то на соседнем стеллаже наверняка найдется отличная книжка по Ruby on Rails.

Кому не имеет смысла читать эту книгу

Как однажды заметил Майк Ли – самопровозглашенный «крутейший программист в мире», «Core Audio – настоящее темное искусство». Вам придется разгребать низкоуровневые API, и если вы не любите марать руки черной работой, то начинать с этой книги, пожалуй, не стоит (однако помните о ее существовании, чтобы вернуться, когда наберетесь достаточно опыта).

Вы должны знать Xcode, C и Objective-C и не испытывать затруднений при чтении и интерпретации файлов-заголовков. Никогда заранее не знаешь, когда придется нырнуть глубже, так что эти знания и навыки определенно помогут при чтении книги.

Что необходимо знать

Предполагается, что вы свободно владеете языком C, в частности знаете, что такое указатели и функция `malloc()`, и понимаете, какие опасности обычно сопровождают низкоуровневое управление памятью. Если вы никогда не работали с C или каким-нибудь C-подобным языком (C++, Java или C#), остановитесь, почитайте сначала какую-нибудь книгу по C, а потом возвращайтесь.

Предполагается также, что вы знакомы со средой программирования Xcode и языком Objective-C. Здесь вы не найдете инструкций по созданию проекта или отладке в Xcode; но существует достаточно книг вводного уровня для новичков и программистов, знакомых с другими платформами и средами. Раз уж вы заинтересовались подсистемой Core Audio и низкоуровневыми C API, мы вправе предполагать, что основами вы давно овладели.

Поскольку речь в книге идет о подсистеме Core Audio в Mac и iOS, мы считаем, что у вас имеется учетная запись разработчика в Apple; если нет, заведите ее (в наши дни они продаются по дешевке!). Зайдите по адресу developer.apple.com/mac или developer.apple.com/ios и зарегистрируйтесь – за 198 долларов вы получите доступ ко всем необходимым обновлениям Mac OS X и iOS, а заодно к Xcode, документации разработчика Apple, примерам кода и даже видеозаписям из центра WWDC.

Поиск в документации

Любой программист, работающий с Core Audio, постоянно заглядывает в онлайновую документацию в поисках имен функций, спис-

ков параметров и разного рода семантической информации (например, что может быть передано в параметре или что должна делать функция). Все это есть на сайте Apple, но у онлайновой документации Apple есть дурная привычка менять место прописки, поэтому было бы трудно включить в книгу URL-адреса, рассчитывая, что они останутся действительными через полгода после выхода в свет.

Вместо этого мы рекомендуем подружиться с обозревателем документации, встроенным в Xcode (если вы еще этого не сделали). В Xcode 4.2 для доступа к нему нужно выбрать из меню пункт **Help ⇒ Documentation and API Reference** – при этом окно организатора открывается на вкладке Documentation. При первом заходе вы увидите экран Quick Start, на котором представлены ссылки на ресурсы, посвященные знакомству с Xcode. В правой панели имеются кнопки для обзора, поиска и управления закладками. Кнопка **Browse** позволяет выбрать активные наборы документов верхнего уровня. На рис. 0.1 показана домашняя страница библиотеки Mac OS X 10.7 Core Library.

Рис. 0.1. Домашняя страница библиотеки Mac OS X 10.7 Core Library в программе просмотра документации, встроенной в Xcode

Левая колонка позволяет организовать документацию по типу, теме, а затем уровню архитектуры Mac OS X или iOS. Прокрутив страницу вниз до уровня Media Layer, вы обнаружите Core Audio, Core Audio Kit и Audio Toolbox – инструментарий, с помощью ко-

торого Core Audio раскрывает большую часть своей функциональности приложениям. Щелкнув по любому пункту, вы увидите список справочных руководств, технических бюллетеней Q&A (вопросы и ответы) и примеров кода. Например, можно щелкнуть по **Audio Toolbox Framework Reference** (Справочное руководство по каркасу Audio Toolbox), а затем воспользоваться кнопкой **Bookmarks** (Закладки) на панели инструментов, чтобы поставить закладку, которая позволит быстро вернуться на эту страницу впоследствии.

На самом деле мы пользуемся обзором редко. Вторая кнопка на панели инструментов в левой панели дает доступ к интерфейсу поиска, с которым мы чаще всего и работаем. Введя искомое слово, вы получите список удовлетворяющих запросу элементов API (методов, функций, типов, структур и т. д.), а также вхождений слова в иные документы, скажем, примеры кода или руководства по программированию. Любой такой документ можно открыть непосредственно в программе просмотра документации. Выше мы уже упоминали термин *audio unit* (аудиоблок); на рис. 0.2 показано, что произойдет при попытке найти слово «AudioUnit» в документации. Как видите, оно встречается в именах функций, typedef'ах, директивах #define и других элементах API, а также в руководствах по программированию, документах Q&A и примерах кода.

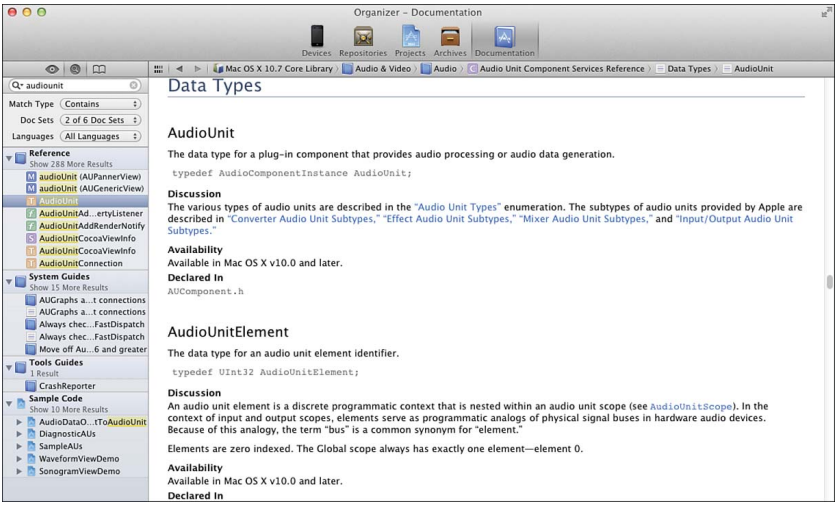

Рис. 0.2. Поиск по слову «AudioUnit» в программе просмотра документации

Можно также искать по слову прямо из исходного кода; двойной щелчок мышью с нажатой клавишей **Option** по слову в тексте программы приводит к появлению всплывающего окна с краткой выдержкой из документации (рис. 0.3). Полная документация будет показана при щелчке по значку с изображением книги в верхней части этого окна. Там же есть значок с изображением буквы **.h**, который открывает определение термина. Обе функции доступны также через контекстное меню, появляющееся при Control-щелчке (или щелчке правой кнопкой мыши) по термину в тексте; им соответствуют пункты **Find Text in Documentation** (Искать текст в документации) и **Jump to Definition** (Перейти к определению).

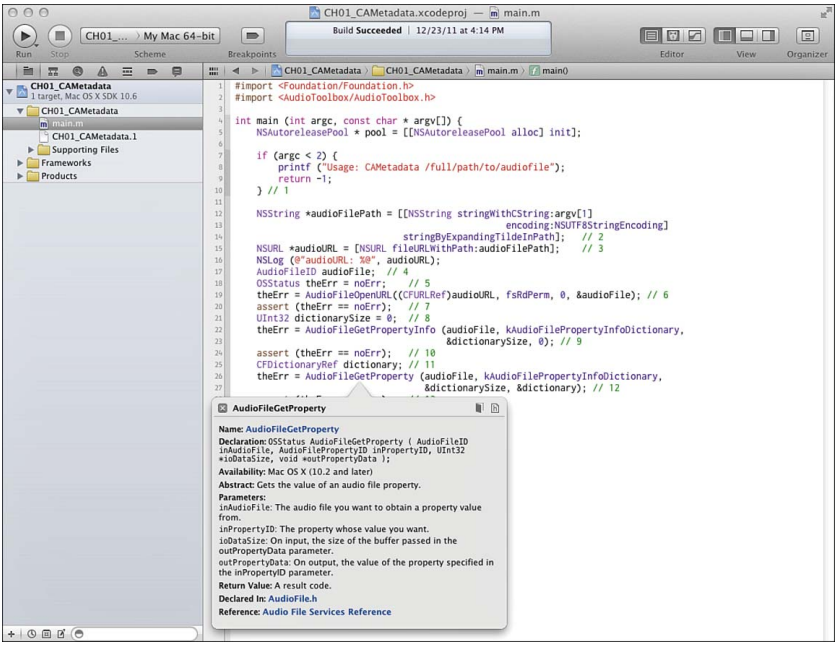

Рис. 0.3. Поиск в документации непосредственно из редактора Xcode

Далее в этой книге мы будем считать, что вы умеете искать информацию с помощью этого интерфейса. Например, вводя в рассмотрение новую функцию, мы полагаем, что вы сами наберете ее имя в поле поиска и поинтересуетесь, что говорит на ее счет официальная документация. Если параметры или возвращаемое значение

функции имеют нестандартные типы, то имена этих типов обычно являются гиперссылками, по которым можно перейти к связанным разделам документации.

Как организована эта книга

Прежде чем отправиться в путешествие, поговорим о том, что стоит на кону. Нас ожидает трудная дорога, усеянная ловушками, но вы должны помнить о щедрой награде, ожидающей в конце пути.

Мы начнем с общего обзора функций Core Audio. Мы кратко опишем и представим примеры ввода и вывода звуковых данных, «транскодирования» форматов, звуковых эффектов, воспроизведения и записи звука и применения интерфейса MIDI.

Затем мы дадим обзор самого API. Мы опишем его процедурную, управляемую свойствами природу, а потом быстро пробежимся по архитектурным модулям, начав с высокоуровневых (Audio Queue, OpenAL, Extended Audio File) и далее перейдя к модулям среднего и нижнего уровней (Audio Units, Audio File, Audio Converter). И завершим этот беглый обзор такими смежными темами, как Core MIDI, OpenAL и особенности работы Core Audio в системе iOS.

Часть I. Принципы работы Core Audio

В этой части закладывается фундамент, на котором покоится здание всей книги. Уже сам объем материала, с которым необходимо познакомиться, прежде чем приступать к написанию кода, указывает на сложность предмета. Понимать, в чем состоит проблема оцифровки звука и какое решение предлагает Core Audio, абсолютно необходимо для чтения последующих разделов и примеров кода.

- **Глава 1. Общие сведения о Core Audio**
  Мы начинаем путешествие с описания Core Audio как подсистемы Mac или iOS: где находятся файлы, как включить ее в проект, куда обращаться за помощью.
  Далее мы даем обзор самого API. Мы опишем его процедурную, управляемую свойствами природу, а затем быстро пробежимся по архитектурным модулям, начав с высокоуровневых (Audio Queue, OpenAL, Extended Audio File) и далее перейдя к модулям среднего (Audio Units, Audio File, Audio Converter) и нижнего уровней (IOKit/IOAudio, HAL и – в случае iOS – Remote I/O). После этого мы разработаем простое приложение, в котором Core Audio используется для извлечения ме-

таданных из звукового файла, – чтобы вы почувствовали, как пишется код с применением Core Audio.

- **Глава 2. Кое-что о звуке**
  Основная задача цифровой обработки звука заключается в том, чтобы представить форму колебаний аналогового звукового сигнала в цифровом компьютере. Мы поговорим о том, как с помощью процесса дискретизации превратить воспринимаемый на слух звук в последовательность нулей и единиц. Мы расскажем, что такое скорость цифрового потока и какие существуют компромиссы между качеством, производительностью и размером файла. Чтобы попрактиковаться в процедуре дискретизации, вы самостоятельно запишете звуковые сигналы непосредственно в файл, отсчет за отсчетом, и увидите (точнее, *услышите*), как по-разному человеческое ухо воспринимает колебания различной формы. Сплошной поток битов необходимо далее разбить на кадры и пакеты. Мы поговорим о различии между постоянной и переменной скоростью цифрового потока и частотой кадров.
- **Глава 3. Обработка звука с помощью Core Audio**
  Разобравшись с понятиями обработки звука на естественном языке, мы должны научиться выражать их на C. Здесь речь пойдет о деталях реализации – о том, как в Core Audio представляются звуковые потоки и какие функции имеются для работы с этими потоками. Мы поговорим о форматах файлов и о форматах потоков и поясним, в чем различия между ними. Затем мы напишем программу, которая исследует, какие комбинации звуковых форматов и форматов файлов поддерживает Core Audio.
  Вслед за описанием форматов мы перейдем к модели обработки в Core Audio и посмотрим, как функциональность Core Audio инкапсулируется в виде аудиоблоков, как эти блоки можно комбинировать разными интересными способами и как для перемещения звуковых данных по системе применяется модель вытягивания.

Часть II. Простые операции со звуком

Эта часть начинается с практического применения API и знаний, полученных в предыдущих главах. Сначала мы обсудим поток данных между файлами и аудиосистемами, для чего запишем, а затем

воспроизведем звуковой файл. Далее мы поговорим об API транскодирования, то есть преобразования данных из одного формата в другой, и объясним стоящую за этим важную функцию.

- **Глава 4. Запись**
  Почему мы поставили запись раньше воспроизведения? Потому что запись проще, а, кроме того, мы попутно сгенерируем файлы с отсчетами звука, которые потом сможем воспроизвести. В этой главе мы познакомимся с высокоуровневым API для записи данных в файл и чтения из файла и исследуем API аудиоочередей (Audio Queue API), позволяющий использовать эти данные. Мы разработаем законченную программу, которая получает звуковые данные от подразумеваемого по умолчанию устройства ввода и записывает их в файл. По ходу дела мы разберемся с некоторыми хитрыми аспектами сжатых форматов, в частности научимся работать с зависящими от форматов сигнатурами и при любом формате вычислять размер буфера, необходимого для промежуточного хранения данных.
- **Глава 5. Воспроизведение**
  С точки зрения программирования, запись и воспроизведение – две стороны одной медали. При воспроизведении данные перемещаются из файла в последовательность буферов; при записи – из последовательности буферов в файл.
  В этой главе мы приведем пример полной программы, которая читает звуковые данные из файла и воспроизводит их на устройстве вывода по умолчанию. Попутно будут рассмотрены способы работы с форматами с переменной скоростью цифрового потока. Кроме того, мы поговорим о некоторых интересных вещах, которые можно делать с помощью свойств и параметров аудиоочереди, и обсудим, как наличие большого количества буферов влияет на задержку.
- **Глава 6. Преобразование**
  Для привыкших к объектно-ориентированному программированию даже высокоуровневый API подсистемы Core Audio кажется весьма низкоуровневым. В этой главе мы продемонстрируем скрытую в глубине сложность и с головой погрузимся в технические детали современных аудиокодеков и проблемы преобразования их в канонический формат. В качестве примера мы разработаем программу, которая напрямую использует службу аудиоконвертера (Audio Converter Services) для преобразования сжатого файла в формат без сжатия. А затем

мы упростим программу за счет использования расширенной службы звуковых файлов (Extended Audio File Services), которая объединяет ввод-вывод и преобразование данных в один шаг.

Часть III. Более сложные операции со звуком

Теперь, когда вы знаете, как перемещать данные туда и обратно, пора заняться чем-то более интересным. Для начала мы научимся накладывать на звуковые данные различные эффекты, затем поговорим о трехмерном позиционном звуке, а закончим производительностью и низкоуровневой архитектурой.

- **Глава 7. Аудиоблоки: генераторы, эффекты и рендеринг**
  Core Audio предоставляет элегантную архитектуру подключаемых модулей для цифровой обработки сигналов, эти модули называются аудиоблоками. Однако аудиоблок – это самый нижний из обычно используемых уровней Core Audio API, и при его программировании возникают совершенно новые проблемы. В этой главе мы покажем, как аудиоблоки используются для воспроизведения звука, полученного из файла и от синтезатора речи в Mac OS X, с наложением эффектов, при этом все действия будут координироваться с помощью AUGraph API. Мы также увидим, как подать на вход аудиоблока программно сгенерированные звуковые данные.
- **Глава 8. Аудиоблоки: ввод и микширование**
  Чтобы помочь вам расширить опыт работы с API аудиоблоков, мы покажем, как воспользоваться блоком ввода-вывода для захвата звуковых данных, а затем с помощью хитроумных манипуляций с потоками поместить захваченные данные в граф аудиоблоков наряду с эффектами и другими источниками. Для этого нам понадобится мощный блок многоканального микширования.
- **Глава 9. Позиционный звук**
  До сих пор речь шла о звуке как таковом, но восприятие звука человеком заставляет по-новому взглянуть на проблему. В этой главе обсуждается OpenAL – API трехмерного позиционного звука, который позволяет ассоциировать источники звука с точками в трехмерном пространстве. Мы начнем с зацикленного проигрывания, но к концу главы вы сможете вос-

производить звуковые потоки произвольной длины из трехмерных источников.

Часть IV. Дополнительные темы

Итак, мы уже рассмотрели большую часть Core Audio, но кое-что еще осталось. В этой части обсуждаются разные разности, для которых не нашлось другого места. Начнем с главы, посвященной исключительно iOS, затем поговорим об обработке MIDI-данных и напоследок – о расширении Core Audio.

- **Глава 10. Core Audio в системе iOS**
  Концептуально разница между обработкой звука в iPhone и в Macintosh невелика, но дьявол, как известно, кроется в деталях. В этой главе речь пойдет как раз о различиях, с особым упором на пределы и исключения, обусловленные ограниченностью аппаратных ресурсов.
  Мы также обсудим API аудиосеансов (Audio Session API), совершенно необходимый для того, чтобы приложение корректно работало в среде iPhone, для которой характерно вытеснение задач из памяти, а также множественность источников и приемников.
- **Глава 11. Core MIDI**
  Музыканты обожают стандарт MIDI, на котором держатся подключение музыкальных инструментов и обработка цифровых музыкальных данных. В этой главе мы рассмотрим, как интерфейс Core MIDI обрабатывает музыкальные события, поступающие к устройству с системой Mac или iOS от инструментов, подключенных физически или по беспроводной сети. Вы увидите, как MIDI-данные можно подать на вход аудиоблока, превратив тем самым сведения о ноте и времени звучания в звук.
- **Глава 12. Кода**
  В заключительной главе мы подведем итог тому, что рассмотрели и что осталось нерассмотренным. Мы также упомянем о самых последних и самых ярких возможностях работы со звуком, появившихся в версиях Mac OS X 10.7 (Lion) и iOS 5.

В этой книге мы не использовали все без исключения функции, имеющиеся в Core Audio, и даже на концептуальном уровне не описали все, что умеет эта система, но постарались глубоко рассмотреть наиболее важные идеи и приемы, которые чаще всего находят при-

менение на практике. Проработав всю книгу, вы будете понимать, как работает Core Audio, и, надеемся, сумеете самостоятельно исследовать оставшуюся за скобками функциональность, необходимую, тем не менее, вашей программе.

О примерах кода

Весь исходный код, приведенный в этой книге, имеется на вкладке Downloads на странице каталога по адресу:

www.informit.com/title/9780321636843

В состав скачиваемого архива входит файл README, а также папки с проектами для каждой главы.

В этой книге много примеров кода, и код с использованием Core Audio современному программисту может показаться довольно пространным. Чтобы не увеличивать без нужды размер кода, большая часть программ написана в виде командных утилит для OS X, а не полноценных приложений Cocoa для Mac OS X с графическим интерфейсом. Запускать их можно прямо из Xcode или из командной строки (в окне программы Terminal или xterm). Примеры для iOS в главе 10 и далее – настоящие приложения iOS (ведь у iPhone нет командной строки), но мы не уделяли пользовательскому интерфейсу внимания больше, чем абсолютно необходимо.

Core Audio на платформах Mac OS X и iOS отличается мало, поэтому разработчики для iOS могут использовать продемонстрированные идеи в приложениях для iPhone, iPod Touch и iPad. В большинстве случаев код будет работать точно так же; различия между Mac и iOS, будь то в самом API или в особенностях его работы на различных платформах, мы отмечали особо. Те части Core Audio, которые уникальны для iOS, рассматриваются в главе 10.

В этой книге мы работали с SDK, входящим в состав Xcode 4.2; он включает отдельные SDK для Mac OS X 10.7 (Lion) и iOS 5. Чтобы писать код с применением Core Audio для Mac, больше ничего не нужно: все библиотеки, файлы-заголовки и документация поставляются вместе с Xcode[1]. Наши примеры поддерживают версии Mac

---

[1] Раньше Core Audio комплектовалась как отдельный загружаемый файл, что смущало разработчиков, которые видели отдельный пункт в списке на странице Apple Development Kits. Этот файл необходим лишь в случае, когда вы собираетесь что-то разрабатывать для версии Tiger (Mac OS X 10.4.x).

OS X 10.6 и 10.7 и iOS 4 и 5. Из-за ряда изменений и исключения устаревших API некоторые примеры для Mac не будут работать в версии 10.5 (Leopard) и более ранних, хотя во многих случаях для восстановления работоспособности достаточно всего лишь изменить константу (например, константа `kAudioFileReadPermission`, появившаяся в версии 10.6, заменила константу `fsRdPerm` с той же семантикой в предыдущих версиях Mac OS X).