Дмитрий Осипов

bhv®

Delphi

Программирование для Windows, OS X, iOS и Android

- СОВРЕМЕННЫЕ ВОЗМОЖНОСТИ ЯЗЫКА ПРОГРАММИРОВАНИЯ Delphi XE5/XE6
- ПЛАТФОРМА ПРИЛОЖЕНИЙ FM
- РАЗРАБОТКА НАТИВНЫХ ПРИЛОЖЕНИЙ ДЛЯ Windows И OS X
- ПОЛНОФУНКЦИОНАЛЬНЫЕ ПРИЛОЖЕНИЯ ДЛЯ Android И iOS
- РАБОТА С СЕНСОРНЫМИ ЭКРАНАМИ
- ТЕХНОЛОГИЯ ЖИВОГО СВЯЗЫВАНИЯ LiveBindings
- ПОДДЕРЖКА БАЗ ДАННЫХ InterBase ToGo И IBLite
- СОВРЕМЕННАЯ 2D- И 3D-ГРАФИКА

PRO

ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

+ *Материалы на www.bhv.ru*

Дмитрий Осипов

Delphi

Программирование для Windows, OS X, iOS и Android

Санкт-Петербург
«БХВ-Петербург»
2014

УДК 004.4'2
ББК 32.973.26-018.1
О-74

Осипов Д. Л.

О-74 Delphi. Программирование для Windows, OS X, iOS и Android. — СПб.: БХВ-Петербург, 2014. — 464 с.: ил. — (Профессиональное программирование)

ISBN 978-5-9775-3289-1

Книга посвящена одному из самых совершенных языков программирования Delphi XE5/XE6. В ней подробно рассматривается новейшая кроссплатформенная библиотека FM, позволяющая создавать полнофункциональное программное обеспечение для операционных систем Windows и OS X, а также для смартфонов и планшетных компьютеров, работающих под управлением Android и iOS. Проекты примеров из книги размещены на сайте издательства.

Для программистов

УДК 004.4'2
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Екатерина Капалыгина*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Марины Дамбиевой*

Подписано в печать 31.01.14.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 37,41.
Тираж 1000 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-5-9775-3289-1

© Осипов Д. Л., 2014
© Оформление, издательство "БХВ-Петербург", 2014

Оглавление

Введение **13**

Глава 1. Подготовка к работе **15**
Выбор типа приложения 17
Выбор целевой платформы для проекта 18
Выпуск приложения для OS X 19
Выпуск приложения для iOS Mobile 22
Выпуск приложения для Android 23
Что делать, когда код зависит от платформы? 27

Глава 2. Забываем VCL? **30**
Опорный класс VCL — *TObject* 30
Управление жизненным циклом объекта 32
Механизм учета ссылок в мобильных проектах 34
Информирование о классе 35
Класс *TPersistent* 37
Основа компонента *TComponent* 38
Владение объектами 39

Глава 3. Классы-шаблоны **41**
Обобщенный тип данных в полях записей 41
Обобщения в процедурах и функциях 43
Обобщенные типы данных в шаблонах классов 44
Наследование шаблона класса 46
Перегрузка методов с параметром обобщенного типа 47
Шаблон массива, класс *TArray<>* 47
Шаблон списка объектов, класс *TObjectList<>* 49
Шаблон словаря *TDictionary<>* 53

Глава 4. Базовые классы FireMonkey **56**
Опорный класс *TFmxObject* 56
Управление дочерними объектами 57
Сопоставление дополнительных данных 60
Поддержка LiveBindings 61

Поддержка анимации....61
Поддержка сенсорного ввода....61
Взаимодействие с командами....61
2D-элементы управления, класс *TControl*....62
Размещение 2D-элемента управления....63
Выравнивание объекта....64
Группировка объектов и компонент *TLayout*....65
Масштабирование и вращение объекта....66
Видимость и прозрачность элемента управления....68
Грани, фаски и визуальные эффекты....68
Состояние элемента управления....70
Обработка событий....70
Простейшие события — щелчок....70
Клавиатурные события....73
События мыши....74
События получения и потери фокуса ввода....77
Событие изменения размера....77
События перетаскивания drag and drop....79
Особенности прорисовки элемента управления....82
Стилевое оформление, класс *TStyledControl*....83
3D-элементы управления, класс *TControl3D*....83
Размеры объекта....84
Повороты объекта....84
3D-события мыши....85

Глава 5. Приложение FireMonkey....88
Приложение *TApplication*....88
Значок приложения....89
Название приложения....89
Расположение исполняемого файла приложения....91
События приложения....91
Контроль активности пользователя....93
Характеристики дисплея, класс *TFormFactor*....94
Формы HD и 3D....95
Описание формы в fmx-файле....96
Общие черты форм....98
Создание, отображение и уничтожение форм....98
Состояние формы....102
Жизненный цикл формы....103
Доступ к элементу управления по его координатам....106
Совмещение форм для разных мобильных устройств в одном приложении....106
Качество графического вывода....107
Форма HD *FMX.Forms.TForm*....108
Трехмерная форма *FMX.Forms3D.TForm3D*....108
Пример 3D-проекта....110
Совместное применение 2D- и 3D-компонентов....113
Стили оформления формы, компонент *TStyleBook*....114
Подключение ресурсов и изображений....115

Глава 6. Меню приложения ... 118
Элемент меню *TMenuItem* ... 120
Элемент меню в виде флажка ... 121
Группировка элементов меню ... 122
Доступ к дочерним элементам меню ... 124
Главное меню *TMainMenu* ... 124
Планка меню *TMenuBar* ... 124
Контекстное меню *TPopupMenu* ... 124

Глава 7. Командный интерфейс ... 126
Команда *TAction* ... 127
Связь с элементом управления ... 129
Выполнение команды ... 130
Установка команды в актуальное состояние ... 130
Связь команды с контейнером ... 131
Предопределенные команды ... 131
Список команд *TActionList* ... 133

Глава 8. Управление папками и файлами ... 135
Работа с дисками ... 135
Сбор сведений о каталогах и файлах ... 136
Проверка существования файла и каталога ... 137
Расположение системных каталогов ... 137
Создание, удаление, копирование и перемещение ... 138
Запись в файл и чтение из файла ... 139
Атрибуты файла и каталога ... 140
Дата и время создания файла и каталога ... 141

Глава 9. Компоненты для работы с текстом ... 142
Класс *TTextControl* ... 143
Метка *TLabel* ... 144
Интерфейс *IVirtualKeyboardControl* ... 146
Основа строк ввода, класс *TCustomEdit* ... 146
Ограничения на ввод ... 148
Выделение части текста ... 148
Взаимодействие с буфером обмена ... 150
Управляющие символы ... 150
Особенности оформления ... 151
Строки ввода *TEdit* и *TClearingEdit* ... 152
Многострочный редактор *TMemo* ... 153
Позиция каретки ... 154
Редактирование текста ... 154
Быстрое перемещение по тексту ... 155
Ввод чисел *TNumberBox*, *TSpinBox* и *TComboTrackBar* ... 156

Глава 10. Компоненты-списки ... 158
Базовый элемент списка *TListBoxItem* ... 159
Список выбора *TListBox* ... 161
Редактирование элементов ... 162
Доступ к выделенному элементу списка ... 164

Доступ к произвольному элементу списка 165
Выбор нескольких элементов 165
Представление элементов в виде кнопки выбора 166
Перестановка элементов 166
Сортировка элементов 166
Текстовый поиск, элемент *TSearchBox* 167
Особенности оформления списка 168
Основные события списка 169
Нередактируемый комбинированный список *TComboBox* 169
Редактируемый комбинированный список *TComboEdit* 172
Компонент выбора значения *TPopupBox* 174

Глава 11. Иерархическая структура 175
Узел дерева *TTreeViewItem* 176
Управление дочерними узлами 176
Положение узла в дереве 178
Состояние узла 179
Дерево *TTreeView* 179
Выделение узла 179
Доступ к узлу 180
Управление составом узлов 180
Узел в роли флажка 183
Свертывание и развертывание узлов 183
Упорядочивание узлов дерева 184

Глава 12. Сетки 185
Колонки сетки 185
Сетка *TGrid* 187
Сетка *TStringGrid* 188
Пример обслуживания текстовых данных 188

Глава 13. Окна сообщений и диалоги 192
Окна сообщений 192
Окна выбора действия 194
Окна ввода данных 196
Компоненты-диалоги 197
Открытие и сохранение файлов *TOpenDialog* и *TSaveDialog* 197
Параметры страницы *TPageSetupDialog* 202
Настройка печати *TPrinterSetupDialog* 203
Отправка задания на печать *TPrintDialog* 203

Глава 14. Дата и время 206
Дата и время *TDateTime* 206
Интервал времени *TTimeSpan* 207
Отсчет времени, таймер *TTimer* 208
Календари *TCalendar* и *TCalendarEdit* 209

Глава 15. Управление цветом 212
Представление цвета ARGB 212
Стандартные цветовые комбинации 214

Компоненты цветовой модели ARGB....215
Компоненты цветовой модели HSL....216
Компоненты *TColorPicker* и *TColorQuad*....216
Цветовые полосы *THueTrackBar*, *TAlphaTrackBar* и *TBWTrackBar*....217
Градиентная заливка *TGradientEdit*....219

Глава 16. Двухмерная графика....222
Управление холстом....223
Кисть *TBrush*....224
Внешний вид линий....226
Шрифт *TFont*....227
Заливка замкнутых областей....228
Вывод простейших фигур....229
Траектория *TPathData*....230
Вывод текста....232
Отображение рисунков....233
Отсечение....234
Сохранение и восстановление состояния холста....234
Работа с растровой графикой, класс *TBitmap*....235
Загрузка и сохранение изображения....235
Кодирование и декодирование графических форматов....236
Получение миниатюры изображения....236
Свойства изображения....237
Простые манипуляции графическим образом....237
Редактирование битового образа....238
Управление графической производительностью....239

Глава 17. Графические эффекты....240
Применение эффекта к файлам изображений....242
Применение нескольких эффектов к файлам изображений....243
Простейшие корректирующие эффекты....246
Заливка цветом *TFillEffect* и *TFillRGBEffect*....246
Яркость и контрастность *TContrastEffect*....246
Регулировка оттенка цвета *THueAdjustEffect*....247
Ясная *TBloomEffect* и пасмурная *TGloomEffect* погода....247
Прозрачность *TColorKeyAlphaEffect*....247
Эффекты размытия и искажения....248
Размытие....248
Искажения....249
Вертикальные полосы *TBandsEffect*....249
Водоворот *TSwirlEffect* и *TBandedSwirlEffect*....250
Увеличительное стекло *TMagnifyEffect* и *TSmoothMagnifyEffect*....251
Стягивание области *TPinchEffect*....252
Рябь на воде *TRippleEffect*....253
Волны *TWaveEffect*....254
Горизонтальная деформация краев текстуры *TWrapEffect*....254
Аддитивные эффекты....254
Отражение *TReflectionEffect*....254
Эффекты свечения *TGlowEffect* и *TInnerGlowEffect*....255

Тень *TShadowEffect* 255
Эффект тиснения *TEmbossEffect* 255
Набросок на бумаге *TPaperSketchEffect* 256
Карандашный набросок *TPencilStrokeEffect* 256
Пикселизация *TPixelateEffect* 257
Старая фотография *TSepiaEffect* 257
Управление резкостью *TSharpenEffect* 258
Глубина цвета *TToonEffect* 258
Геометрические эффекты 258
Аффинные преобразования *TAffineTransformEffect* 258
Обрезка *TCropEffect* 258
Перспектива *TPerspectiveTransformEffect* 259
Эффект плитки *TTilerEffect* 260
Наложение изображений *TNormalBlendEffect* 260
Эффекты трансляции 261

Глава 18. Анимация 263
Простой пример анимации 263
Общие черты компонентов-аниматоров, класс *TAnimation* 265
Индивидуальные особенности компонентов-аниматоров 267
Цветовая анимация, компонент *TColorAnimation* 268
Градиентная анимация, компонент *TGradientAnimation* 268
Анимированная картинка, компонент *TBitmapAnimation* 268
Анимированный ряд, компонент *TBitmapListAnimation* 268
Анимация числовых свойств, компонент *TFloatAnimation* 269
Анимация прямоугольной области, компонент *TRectAnimation* 269
Анимация траектории, компонент *TPathAnimation* 269

Глава 19. Мультимедиа 271
Воспроизведение мультимедиа 271
Менеджер кодеков *TMediaCodecManager* 271
Проигрыватель *TMediaPlayer* и компонент *TMediaPlayerControl* 273
Захват аудио- и видеопотока 275
Менеджер устройств *TCaptureDeviceManager* 275
Захват потоков мультимедиа 276
Аудиозахват *TAudioCaptureDevice* 277
Видеозахват *TVideoCaptureDevice* 277
Камера *TCameraComponent* 280

Глава 20. Сенсорный ввод 281
Описание жеста 281
Реакция на сенсорный ввод 283
Интерактивные жесты 285
Пример обработки стандартных жестов 286

Глава 21. InterBase ToGo 287
Соединение с БД *TSQLConnection* 288
Управление соединением 288
Регистрация пользователя 290

Управление подчиненными наборами данных....291
Управление транзакциями....291
Выполнение SQL-инструкций....292
Информирование о БД....293
Набор данных *TSQLDataSet*....293
Хранимая процедура *TSQLStoredProc*....296
Запрос *TSQLQuery*....299
Выпуск приложения....301

Глава 22. LiveBindings....303
Визуальный дизайнер....305
LiveBindings в проектах баз данных....308
Binding Expressions — связь с помощью выражений....310
Класс *TBindExpression*....313
Выражение LiveBindings....315
Класс *TBindings*....317
Lists — связь между списками....319
Класс *TBindList*....322

Глава 23. Многопоточные приложения....323
Поток *TThread*....323
Метод ожидания....328
Управление приоритетом потока....329
Синхронный и асинхронный вызовы внешнего метода....330
Пример многопоточного приложения....330
Синхронизация потоков в Windows....332
Синхронизация событием *TEvent*....333
Критическая секция *TCriticalSection*....336
Мьютекс *TMutex*....337
Семафор *TSemaphore*....338

Глава 24. Мультиязычные проекты....341
Компонент языковой поддержки *TLang*....341
Режим автоматического перевода....345
Перевод меню....345

Глава 25. Мобильная платформа....347
Интернет-браузер *TWebBrowser*....348
Привязка к местности....350
Датчик местоположения *TLocationSensor*....350
Прямое и обратное преобразования координат *TGeocoder*....352
Датчик ориентирования *TOrientationSensor*....356
Менеджер датчиков *TSensorManager*....358
Увеличительное стекло *TMagnifierGlass*....360
Подсистема уведомлений....361
Пример вывода текстового уведомления в назначенное время....364
Вызов приложения из окна уведомления....365
Пример размещения числа на значке приложения....366
Звонок по телефону....367

Глава 26. Законы трехмерного мира 369
Система координат 369
Единица измерения 371
Точка 372
Вектор 372
Объект 373
Фрейм 374
Проекция 375

Глава 27. Проектируем 3D-сцены 378
Построение сцены 378
Источник света, класс *TLight* 380
Камера, класс *TCamera* 381
Объект-заместитель, класс *TProxyObject* 383
Макет, класс *TDummy* 384

Глава 28. Геометрическое описание фигур и mesh-объекты 385
Произвольный объект, классы *TMesh* и *TMeshData* 385
Проектируем треугольник 388
Проектируем тетраэдр 389
Проектируем четырехугольник 391
Управление нормалями вершин 393
3D-модель, класс *TModel3D* 394
Импорт модели во время выполнения программы 397

Глава 29. Материал объекта 398
Заливка цветом, компонент *TColorMaterialSource* 399
Текстурирование 399
Источник текстуры *TTextureMaterialSource* 400
Управление координатами текстуры в *TMesh* 400
Отраженный свет и компонент *TLightMaterialSource* 402
Дополнительная настройка текстур и класс *TTexture* 404

Глава 30. 3D-контекст *TContext3D* 406
Управление графической сессией 407
Графические примитивы класса *TContextHelper* 408
Графические примитивы класса *TContext3D* 411
Освещение 413
Матрицы и матричные преобразования 414
Текстуры 416
Шейдеры 416

Приложение 1. Вектор *TVector3D* 419
Длина вектора 420
Нормализация вектора 421
Проверка равенства двух векторов 421
Сложение и вычитание векторов 421
Расстояние между двумя векторами 422
Масштабирование вектора 423

Векторное произведение 423
Скалярное произведение 424
Поворот вектора 426
Отражение вектора 426

Приложение 2. Матрица преобразований *TMatrix3D* **428**
Нулевая и единичная матрицы 429
Матрица переноса 429
Матрицы вращения 430
Матрица масштабирования 431
Умножение матриц 431
Дополнительные матричные операции 433

Приложение 3. Модуль *System.IOUtils* **434**

Приложение 4. Датчики **442**

Приложение 5. Описание электронного архива **445**

Список литературы **446**

Предметный указатель **447**

Введение

Совсем недавно среда проектирования Embarcadero RAD Studio совершила очередной эволюционный скачок — в составе языков Delphi и C++Builder появилась принципиально новая возможность разработки кроссплатформенных приложений. Современные версии Delphi позволяют создавать не только приложения для Win32 и Win64, но и полноценные программные продукты, которые предназначены для работы под управлением операционных систем, разработанных компанией Apple (OS X 10.7 Lion, OS X 10.8 Mountain Lion, iOS начиная с версии 5.1) и компанией Google (речь об Android с диапазоном версий от 1.5 до 4.3)! В основу кроссплатформы положена во всех отношениях уникальная библиотека FireMonkey. Компания Embarcadero рекомендует называть "библиотеку FireMonkey" "платформой FM", но большинству читателей привычней использовать первый вариант, поэтому мы будем придерживаться в книге именно этого словосочетания.

Книга, которую вы держите в руках, в большей степени рассчитана на подготовленного программиста, имеющего представление о языке и возможностях так называемых классических версий Delphi. Такого читателя в первую очередь интересует ответ на единственный вопрос: "Чего такого нового появилось в языке Delphi XE5, чтобы я заинтересовался им?" Постараюсь во введении к книге ответить на этот вопрос как можно более кратко и формализовано, разбив ответ на пункты.

- Библиотека VCL ни в коем случае не умерла и по-прежнему поддерживается компанией Embarcadero, но по темпам своего развития (важнейшему показателю для претендующего на успех программного продукта) она замедлилась. Вряд ли раскрою вам секрет, утверждая, что в IT-индустрии остановка равносильна смерти, а VCL движется все медленнее... Поэтому на смену VCL неотвратимо приходит инновационная по своей сути библиотека FireMonkey.
- FireMonkey — это по-настоящему кроссплатформенная библиотека, которая на данный момент поддерживает Win32, Win64, OS X, iOS и Android. Очень важно, что вы можете использовать один и тот же код для компиляции проекта как под Windows, так и под OS X и Android!
- FireMonkey обладает непревзойденными графическими возможностями и позволяет создавать приложения, опирающиеся в первую очередь на DirectX, OpenGL и GDI+ (напомню, что проекты VCL изначально ориентированы на устаревший GDI).

- В продолжение темы с графикой отмечу, что FireMonkey позволяет создавать не только классические двухмерные приложения, но и способные впечатлить самого притязательного пользователя приложения с трехмерной графикой.
- Еще одно замечание касается того, что все визуальные элементы управления FireMonkey в буквальном смысле нарисованы (представляют собой битовые образы), и это позволяет применять к ним как по отдельности, так и в целом самые нетривиальные визуальные эффекты.

❒ Еще со времен Borland язык Delphi славился своим высокопроизводительным компилятором. Сегодня в Delphi XE5 их шесть (Win32, Win64, OS X, эмулятор iOS для x86 и iOS ARM, Android). Важно отметить тот факт, что компиляторы iOS ARM и Android модульные. Модуль фронтального (front-end) компилятора ARM переводит исходный код программы на языке Delphi в промежуточный байт-код. Конечный модуль (back-end) компилятора представляет собой низкоуровневую виртуальную машину (Low Level Virtual Machine, LLVM), преобразующую промежуточный байт-код в машинный код целевой платформы. Компиляторы LLVM широко используются в компаниях Apple, Google и Adobe, что подсказывает направление дальнейшего развития Delphi.

❒ Язык Delphi всегда отличался своими возможностями по разработке приложений для БД. Библиотека FireMonkey вобрала все лучшее по работе с БД и поддерживает все распространенные коммерческие и бесплатные системы управления данными. В этой книге мы поговорим об одной из новых разработок Embarcadero — настольной БД InterBase ToGo.

Одним словом, за библиотекой FireMonkey большое будущее, в какой-то степени это пригодный для всех случаев жизни швейцарский нож, который должен быть всегда под рукой.

ГЛАВА 1

Подготовка к работе

Для того чтобы программист, впервые столкнувшийся с платформой FireMonkey (FMX), сразу вошел в курс дела, предложу обобщенную схему библиотеки (рис. 1.1). Платформа FireMonkey включает в себя многочисленный набор классов и сервисных интерфейсов, написанных на языке Delphi, в их числе элементы управления для 2D- и 3D-приложений, высококачественная графическая подсистема, поддержка сенсорного ввода и многое другое. Замечу, что рис. 1.1 существенно упрощает представление о FireMonkey, на самом деле это гораздо более сложная и многоуровневая система, но для первого знакомства его вполне достаточно.

> ***Замечание***
>
> Стоит знать, что у истоков платформы FireMonkey стоит российский программист Евгений Крюков.

Появление принципиально новой программной платформы, получившей несколько необычное название FireMonkey, существенно расширило возможности языка Delphi. Теперь, помимо классических приложений VCL, у программиста есть возможность разрабатывать несколько типов кроссплатформенных проектов:

1. FireMonkey Desktop Application для Windows и OS X:
   - приложения FireMonkey HD Application;
   - приложения FireMonkey 3D Application.
2. FireMonkey Mobile Application для iOS.
3. FireMonkey Mobile Application для Android.

Приложения FireMonkey HD Application позволяют создавать программные продукты с высококачественным двухмерным графическим интерфейсом для операционных систем Windows и OS X. Проекты HD в первую очередь окажутся востребованы в качестве бизнес-приложений (прикладное программное обеспечение и клиентские приложения баз данных). С некоторой степенью допущения возможности FireMonkey HD Application можно сравнить с традиционными проектами VCL. Хотя о знаке равенства между этими платформами речи идти не может. Взвешивая

достоинства и недостатки двух знаковых библиотек, на чашу весов, оценивающую преимущества FireMonkey, стоит положить две внушительные "гири" с названиями "кроссплатформа" и "качественная высокопроизводительная графика". В пользу горячо любимой программистами Delphi и C++Builder платформы VCL я бы отнес такие достоинства, как исключительная проработанность библиотеки, огромное число компонентов и безусловная поддержка Win API, COM, ADO и т. д. (что позволяет писать весьма эффективный код под Microsoft Windows).

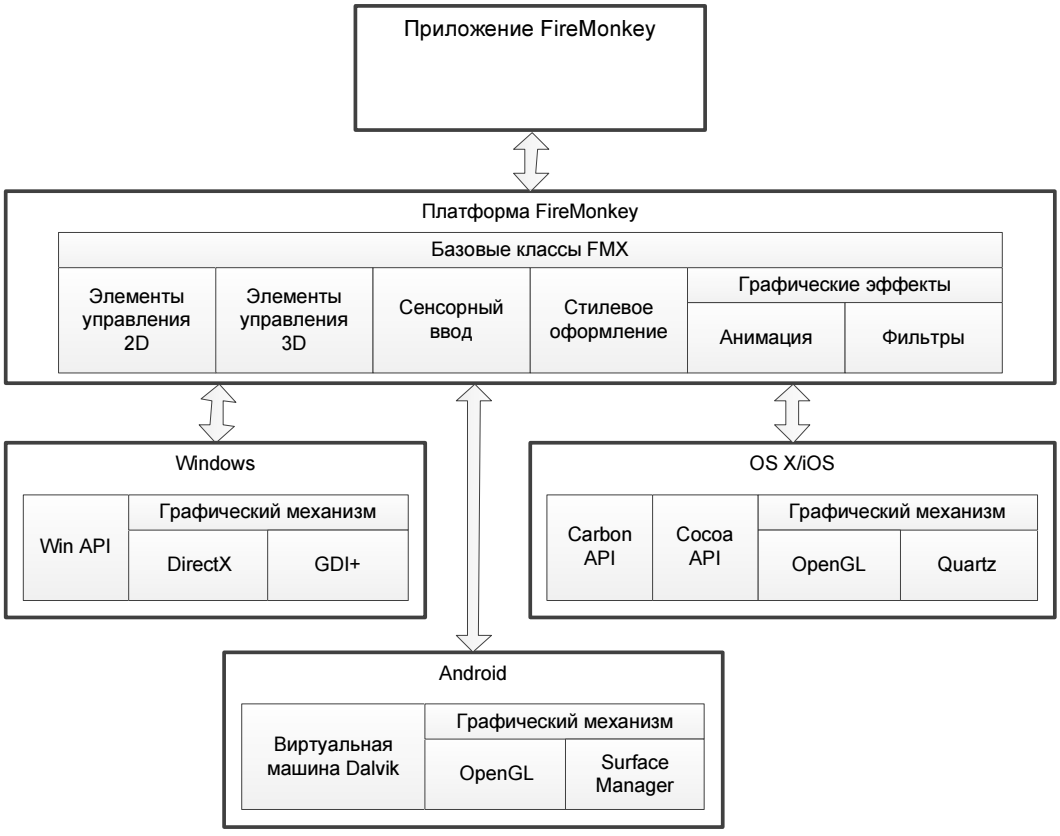

Рис. 1.1. Платформа FireMonkey

Замечание

Для разработчиков, ориентированных на новейшую операционную систему Windows 8, платформа FireMonkey предоставляет возможность создавать приложения с интерфейсом Metropolis — аналогом интерфейса Windows 8.

Приложение FireMonkey 3D Application — это еще один шаг, сделанный Embarcadero навстречу востребованному сегодня мультимедийному направлению. Наличие аббревиатуры "3D" подсказывает, что на этот раз программист получает право создавать уникальный трехмерный пользовательский интерфейс. Заметим, что сама по себе идея 3D не нова, она широко применяется в общеизвестных системах

DirectX и OpenGL. Но то, что сделали в Embarcadero, без стеснения можно называть инновацией. На момент написания этих строк ни одна из коммерчески успешных систем разработки ПО не была способна создавать полноценные кроссплатформенные бизнес-приложения 3D путем простого переноса компонентов на форму! Насколько это стало удобно, вы поймете сразу, если у вас есть хотя бы небольшой опыт разработки интерфейсной части программных продуктов с помощью инструментария DirectX или OpenGL...

> ***Замечание***
>
> Подчеркнем, что Embarcadero не позиционирует FireMonkey как "движок" для разработки игровых приложений. Однако пока речь идет о первых шагах платформы, поэтому нельзя исключить, что в последующих версиях библиотеки она позволит работать и в этом сегменте ПО.

Приложения FireMonkey Mobile Application предназначены для работы под управлением операционных систем iOS и Android. Таким образом, благодаря Delphi XE5 вы приобретаете уникальную возможность писать программное обеспечение для iPad и iPhone корпорации Apple и для многочисленных устройств, использующих операционную систему Android компании Google!

Выбор типа приложения

Для создания приложения на базе библиотеки FireMonkey следует обратиться к элементу меню **File | New | Other** и в появившемся диалоговом окне **New Items** следует выделить необходимый значок с названием типа приложения (рис. 1.2).

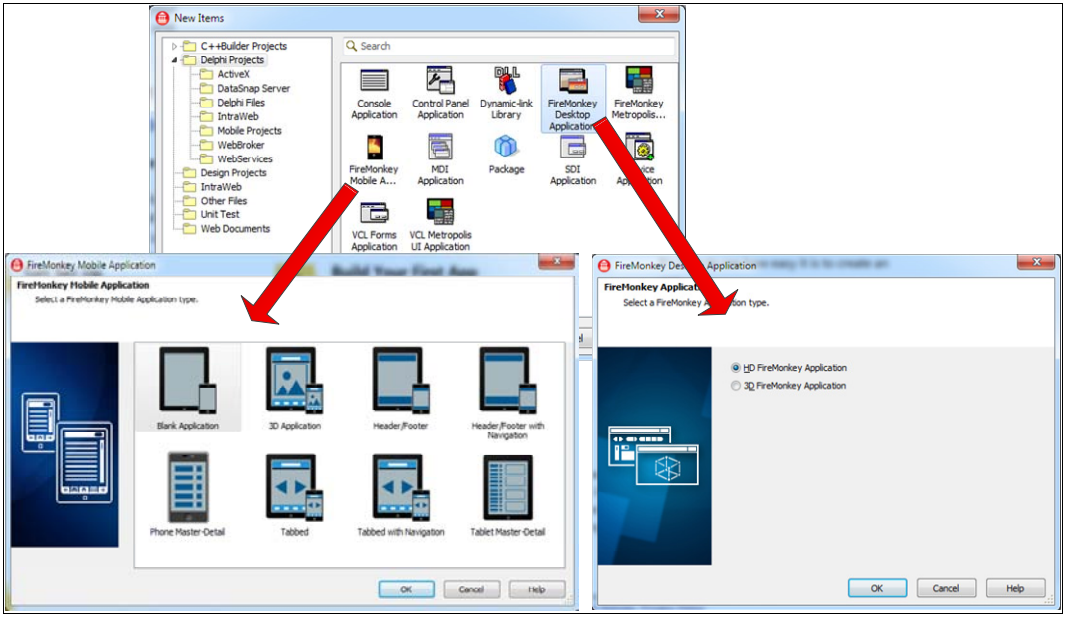

Рис. 1.2. Диалоговые окна выбора типа проекта и создания проекта

Обратите внимание (см. рис. 1.2), что в составе шаблонов имеется заготовка проекта FireMonkey Metropolis UI Application. Это разновидность проектов HD с модным сегодня "плиточным" интерфейсом Metropolis, применяемым в Windows 8.

Выбор целевой платформы для проекта

Раз основная заслуга FireMonkey заключается в поддержке не только Windows, но и OS X, то изучение приложения FireMonkey начнем с определения целевой платформы для реализации приложения.

Создайте новый проект. Для этого воспользуйтесь пунктом меню **File | New | FireMonkey Desktop Application**.

После появления на свет нового проекта обратитесь к окну менеджера проекта (рис. 1.3). В дереве менеджера проекта найдите узел **Target Platforms** и, воспользовавшись услугами контекстного меню узла, добавьте интересующую вас платформу (32-разрядная Windows, 64-разрядная Windows или OS X). В результате у узла **Target Platforms** появится дочерний элемент с названием вновь добавленной платформы.

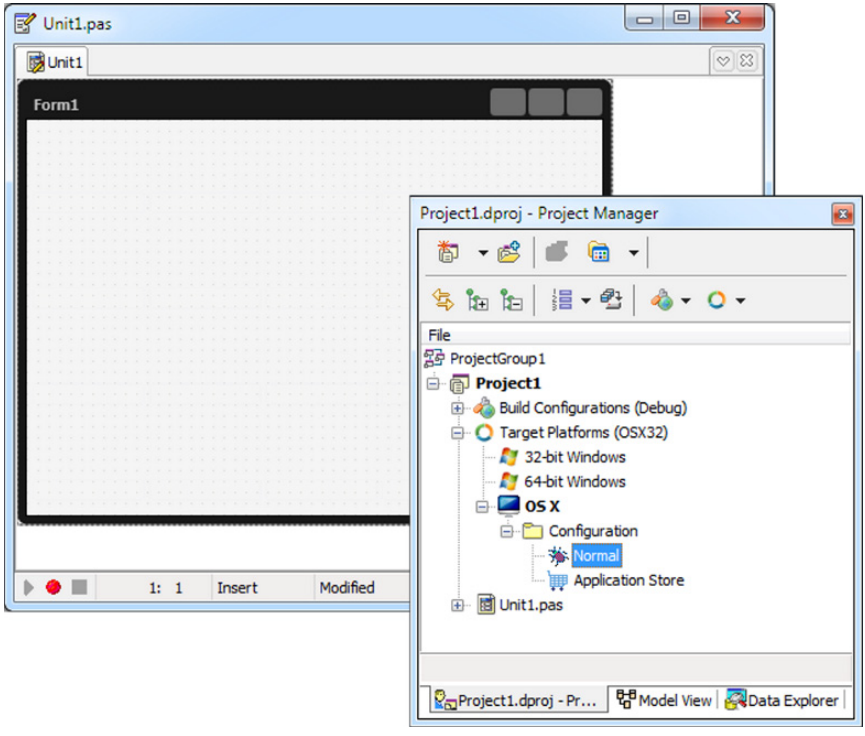

Рис. 1.3. Выбор целевой платформы для проекта FireMonkey

Выпуск приложения для OS X

Выпуск приложения для 32- и 64-разрядной Windows не вызывает никаких затруднений. Программист указывает предпочтительную платформу и просто нажимает клавишу <F9>. Если же вы планируете создать релиз для OS X, то придется еще немного потрудиться.

1. По возможности полностью отладьте приложение под управлением Windows (это возможно, если вы не используете функции API OS X). Это исключит многие проблемы при создании исполняемого бинарного кода под OS X и в целом ускорит работу над выпуском релиза.
2. Соедините в сети два компьютера. На первом должна быть установлена операционная система Windows и развернуто ваше программное обеспечение Delphi. Второй компьютер должен работать под управлением операционной системы OS X.
3. Обращаемся к компьютеру с FireMonkey. Найдите в каталоге C:\Program Files\Embarcadero\RAD Studio*n.n*\PAServer (или C:\Program Files (x86)\Embarcadero\RAD Studio*n.n*\PAServer, если вы работаете в 64-разрядной Windows) файл RADPAServerXE5.pkg (в версиях XE2 и XE3 это был архив setup_paserver.zip, в XE4 — RADPAServerXE4.pkg). Можете перенести файл на MAC с помощью флешки, а лучше всего предоставьте к данному каталогу сетевой доступ, так чтобы папку с файлом смогла увидеть станция Mac (рис. 1.4).

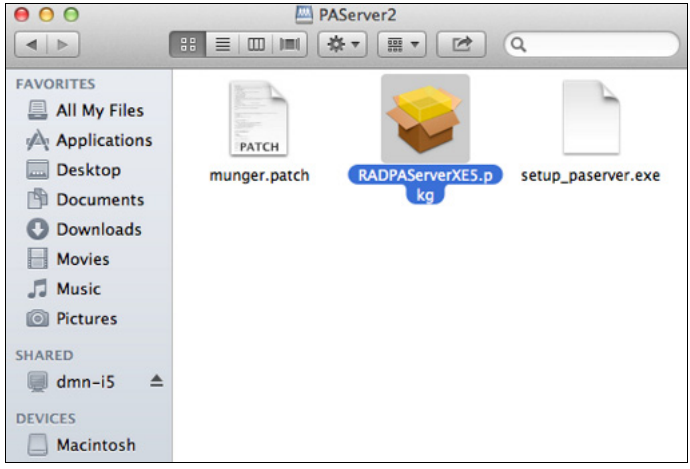

Рис. 1.4. Сетевой доступ к файлам папки PAServer со станции Mac

4. Разверните программное обеспечение на рабочей станции с OS X, в результате в папке с приложениями вы увидите файл RAD PAServer XE5.app. Это приложение (в официальной документации именуемое **Platform Assistant**) возьмет на себя обязанности по компиляции приложения FireMonkey.
5. Запустите в терминальном окне компьютера с OS X установленное программное обеспечение (рис. 1.5).

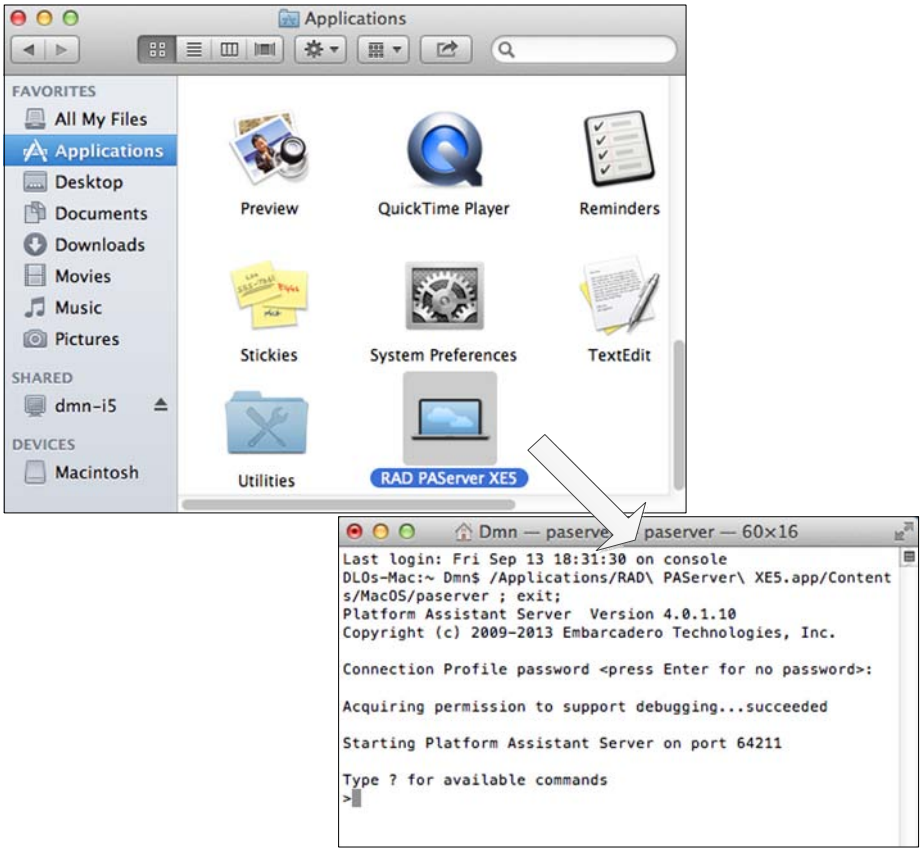

Рис. 1.5. Запуск RAD PAServer XE5 в терминальном окне

6. Возвращаемся к компьютеру с Windows. В менеджере проектов Delphi создайте целевую платформу (**Target Platforms**) OS X и, дважды щелкнув по узлу, сделайте ее активной (см. рис. 1.3).
7. Нажмите "священную" для программиста Delphi клавишу <F9>. И если с активной платформой OS X это делается впервые, то перед вами появится окно мастера создания профиля (рис. 1.6). Придумайте для профиля имя, введите IP-адрес станции Mac, при желании поменяйте номер порта и (если вы верите в теорию заговора) придумайте пароль. Нажав кнопку **Test Connection**, протестируйте соединение и завершите работу помощника, нажав кнопку **Finish**. Настроив профиль, вновь "давим" <F9> — в ответ вы увидите свое первое приложение для Mac (рис. 1.7). Откомпилированное приложение для платформы OS X вы обнаружите на станции Mac в папке /Users/*имя_пользователя*/RADPAServer/scratch-/*имя_профайла*.

Внимание!

Компания Embarcadero постоянно улучшает свое программное обеспечение, это утверждение также относится и к PAServer. Поэтому после любого обновления Delphi обязательно переустановите и PAServer.

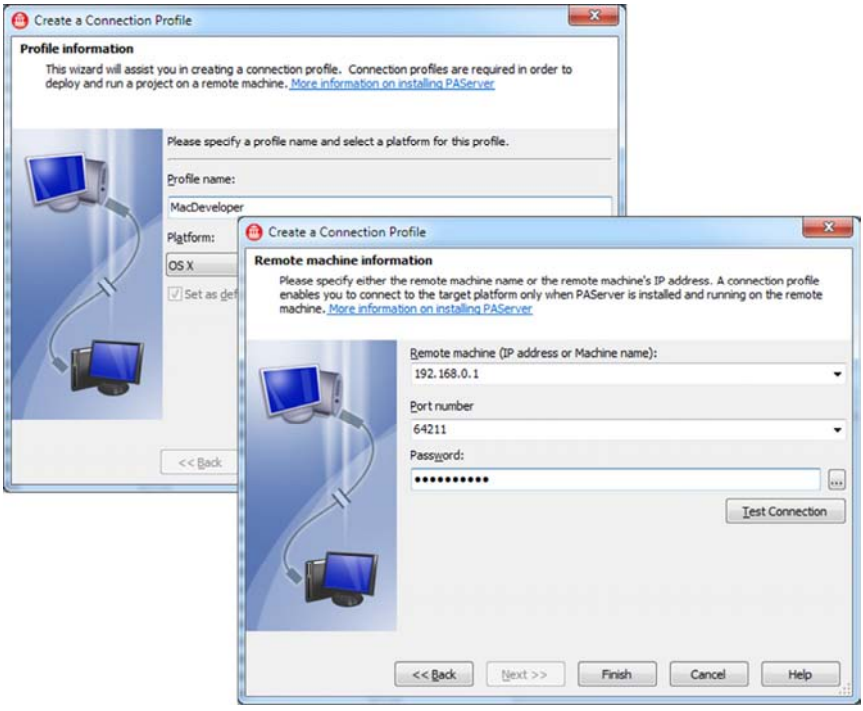

Рис. 1.6. Мастер создания удаленного профиля

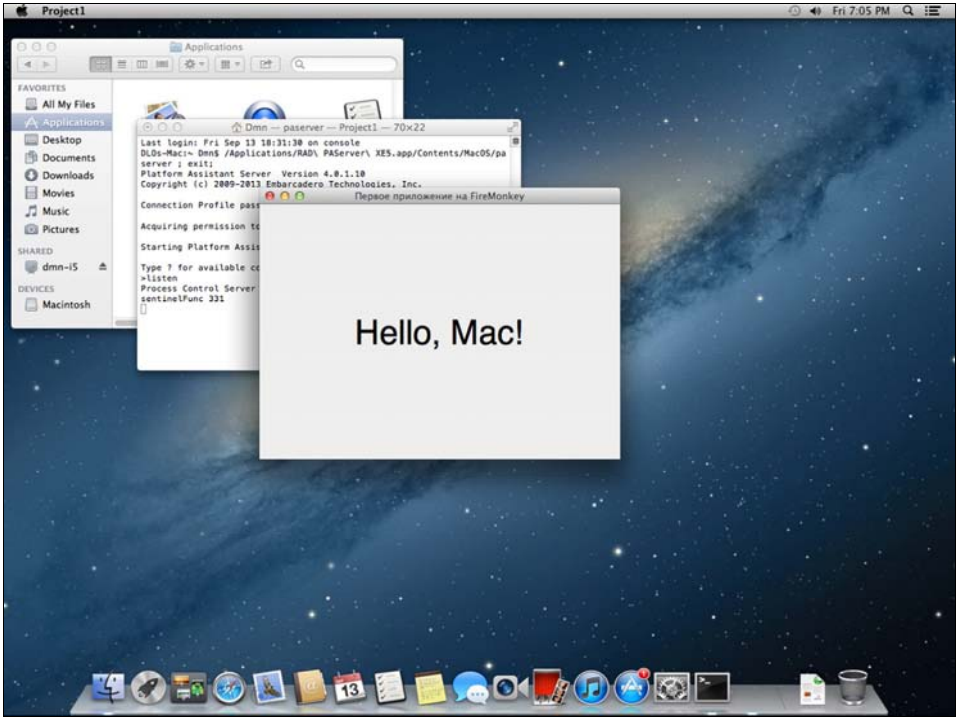

Рис. 1.7. Приложение FireMonkey для платформы OS X

Замечание

Если в вашем распоряжении нет компьютера Mac, но необходимо осуществить тестирование приложения, то можно воспользоваться услугами облачного сервиса macincloud, который расположен по адресу **http://www.macincloud.com/**.

Выпуск приложения для iOS Mobile

Для выпуска приложения, предназначенного для работы под управлением мобильной платформы iOS, нам вновь потребуется рабочая станция Mac с развернутой на ней операционной системой OS X 10.7 Lion или OS X 10.8 Mountain Lion. Мобильное устройство (с операционной системой iOS 5.1 и выше) стоит подключить к компьютеру Mac через USB-порт, для того чтобы ускорить процесс тестирования приложения.

Если вы намерены испытать свои силы в разработке приложений для мобильной платформы iOS Mobile, то в самом начале пути для настройки компьютеров рекомендую воспользоваться услугами специализированного помощника. Для его вызова достаточно, обратившись к пункту меню **File | New | FireMonkey Mobile Application**, создать свое первое мобильное приложение. После того как вы выберите шаблон приложения и укажете папку, в которую следует сохранить файлы проекта, среда разработки выведет на экран окно **iOS Mobile Help Wizard**.

С этого момента всеми остальными вашими действиями станет руководить помощник. Во-первых, он попросит вас установить на станции Mac среду разработки Xcode, позволяющую разрабатывать приложения для Mac, iPhone и iPad. На момент написания этих строк программное обеспечение Xcode 5 можно было свободно скачать со страницы **https://developer.apple.com/xcode/** компании Apple.

Во-вторых, вам придется развернуть Xcode Command Line Tools. Для этого следует запустить установленное на предыдущем этапе программное обеспечение Xcode, выбрать пункт меню **Preferences** и щелкнуть по расположенной на главной панели команде **Downloads**. В окне загрузок выбираем вкладку **Components** и щелкаем по кнопке **Install**.

Внимание!

Для разработки приложений для Mac вы должны обладать учетной записью Apple Developer.

В-третьих, вам следует развернуть на станции Mac программу-ассистент **Platform Assistant**, которую вы обнаружите на жестком диске компьютера с Embarcadero RAD Studio XE5. Интересующий нас файл RADPAServerXE5.pkg расположен в папке C:\Program Files\Embarcadero\RAD Studio*n.n*\PAServer\. Завершив инсталляцию программы-ассистента, найдите в папке Applications приложение RAD PAServer XE5.app и запустите его на выполнение.

В-четвертых, вам предстоит соединить компьютер с Embarcadero RAD Studio XE5 и станцию Mac и сконфигурировать профиль соединения (connection profile). Для вызова мастера конфигурации обратитесь к IDE Delphi и выберите пункт меню **Tools |**

Options. В появившемся окне настройки опций выберите узел **Environment Options | Connection Profile Manager**. Щелчок по кнопке **Add** вызовет на экран уже знакомый нам помощник (см. рис. 1.3).

Выпуск приложения для Android

Процесс выпуска приложения для операционной системы Android (в сравнении с работой с iOS) имеет ряд существенных отличий, ведь на этот раз нам предстоит работать с операционной системой, разработанной не в компании Mac, а в Google.

Если вы планируете написать приложение для конкретного мобильного устройства, то целесообразно установить на ваш персональный компьютер USB-драйвер этого смартфона или планшетного компьютера. Все необходимые драйверы вы обнаружите в Интернете, в первую очередь рекомендую ресурсы:

http://developer.android.com/sdk/win-usb.html

https://developer.amazon.com/sdk/fire/connect-adb.html

http://developer.android.com/tools/extras/oem-usb.html

Если же вы нацелены на разработку ПО для разнотипных устройств, то запустите **Android SDK Manager** (кнопка **Пуск | Все программы | Embarcadero RAD Studio | Android Tools**) и отметьте "галочкой" в окне менеджера строку **Google USB Driver** (рис. 1.8). Щелчок по кнопке **Install packages** автоматически отправит запрос на соответствующий сайт, с которого будет осуществлено бесплатное скачивание драйверов.

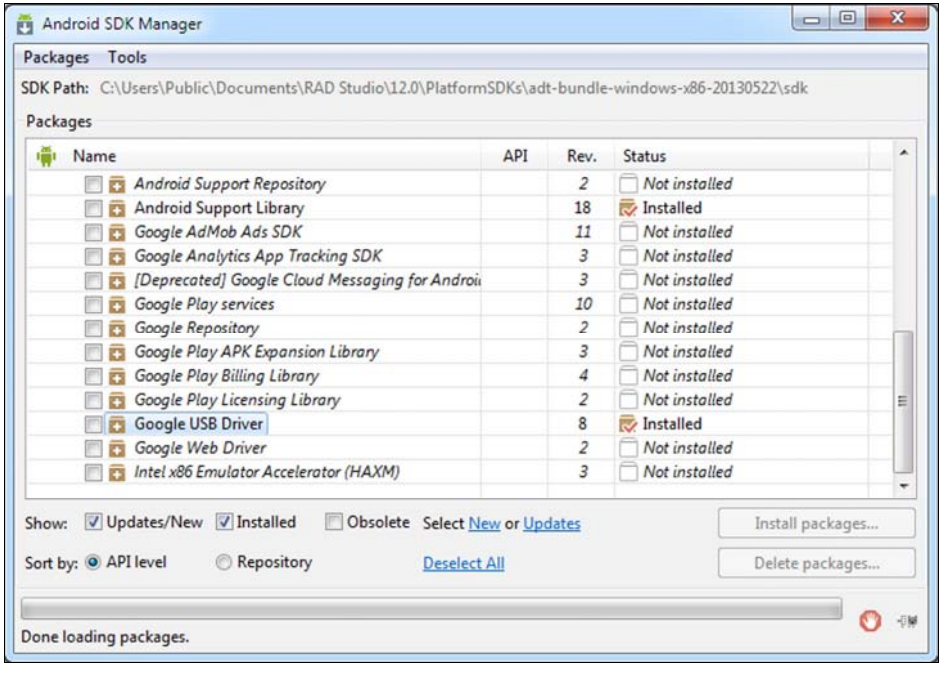

Рис. 1.8. Установка USB-драйверов для устройств Android

Хотя существует возможность непосредственной отладки приложения на реальном устройстве (подключаемом к компьютеру через интерфейс USB), лучшим решением станет развертывание на компьютере эмулятора Android — говоря языком программиста, Android Virtual Device (AVD). Для лучшей формализации процесса развертывания эмулятора вновь разобью наши действия по пунктам.

1. Установите на компьютер образ системы Android (Android system image). Для этого:
   - нажмите кнопку **Пуск** Windows и выберите пункт **Все программы | Embarcadero RAD Studio XE5 | Android Tools**;
   - в появившемся на экране окне **Android SDK Manager** отметьте "галочкой" элемент **ARM EABI v7a System Image** в интересующей вас версии ОС Android и нажмите кнопку **Install *nn* package**. Подтвердите свой выбор в очередном окне **Choose Packages to Install** (рис. 1.9) и дождитесь установки с сайта Google виртуальной машины, SDK и других бесплатных модулей.

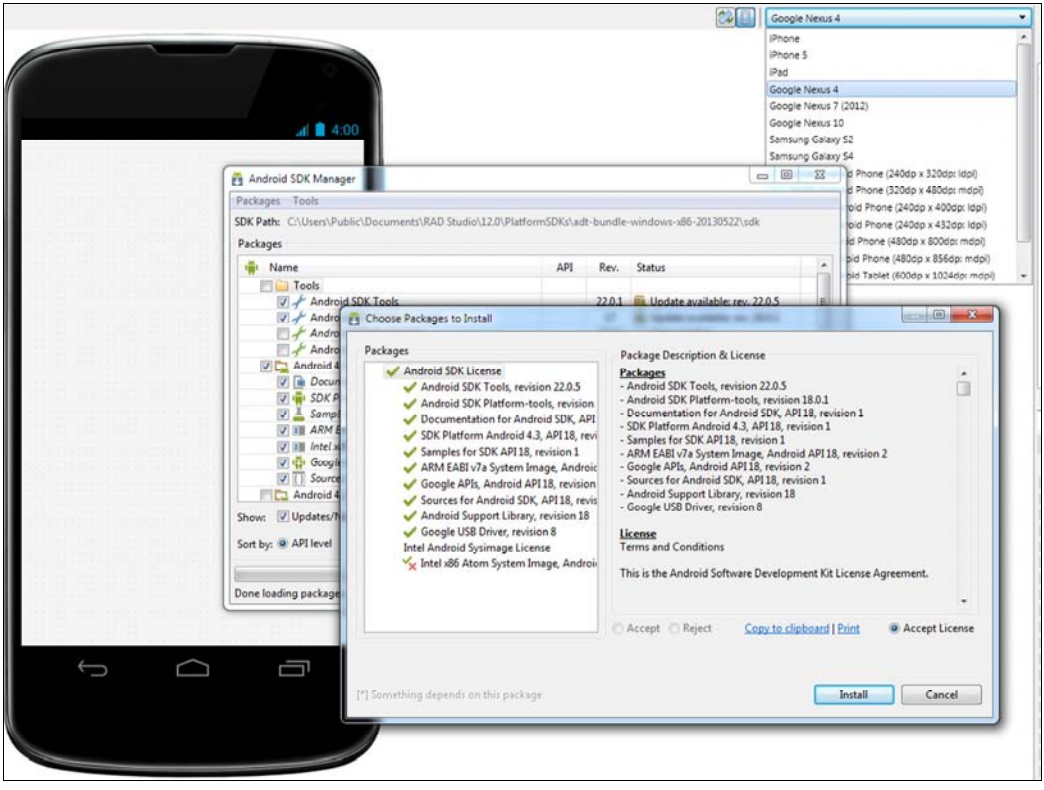

Рис. 1.9. Установка виртуальной машины для приложений Android

2. Завершив инсталляцию необходимых библиотек, приступим к созданию виртуального устройства. Для этого:
   - при посредничестве кнопки **Пуск** Windows вновь доберитесь до элемента меню **Android Tools**;

- в меню появившегося на экране Android SDK Manager найдите элемент **Tools | Manage AVDs**;
- нажав кнопку **New** в окне **Android Virtual Device Manager**, приступаем к определению свойств виртуального устройства (рис. 1.10). В обязательном порядке конкретизируйте тип устройства (раскрывающийся список **Device**) и целевую платформу (раскрывающийся список **Target**). Рекомендую установить флажок **Use Host GPU**, это заставит эмулятор при работе с OpenGL использовать функционал вашего графического процессора.

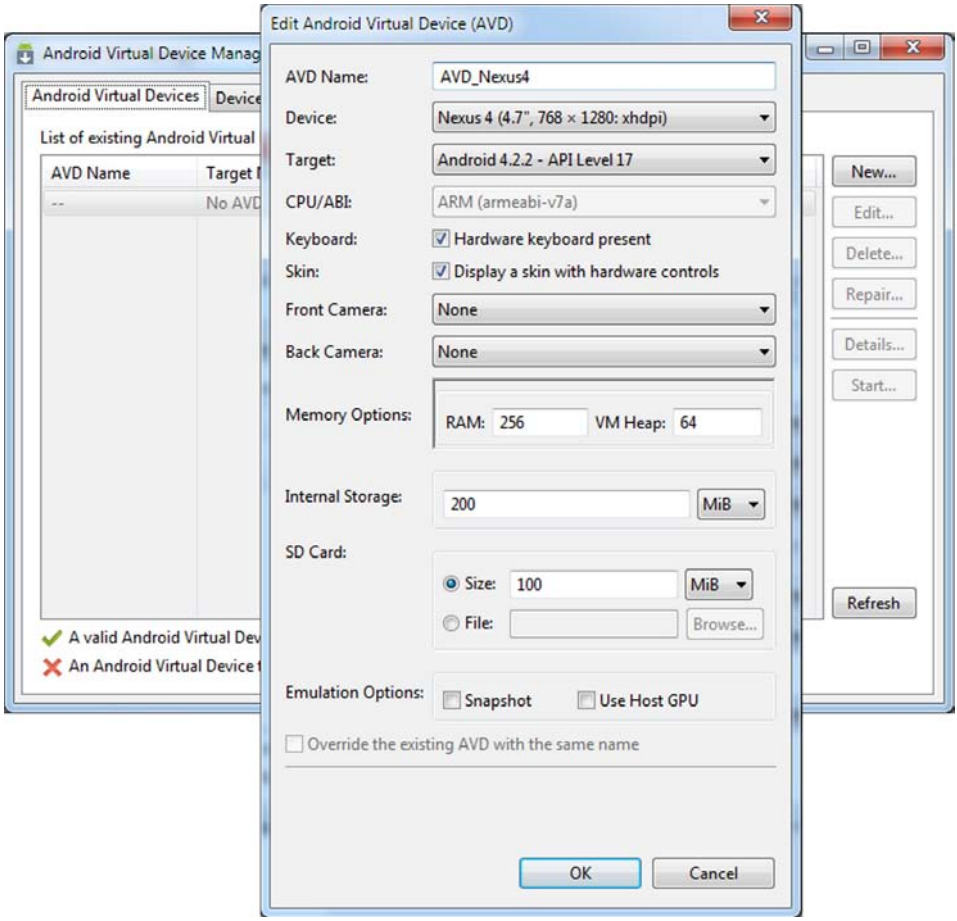

Рис. 1.10. Создание виртуального устройства

3. Проверим эмулятор устройства. Для этого найдите его в перечне сконфигурированных нами устройств и нажмите кнопку **Start** (рис. 1.11).

Замечание

Запуск эмулятора устройства на базе платформы Android — достаточно трудоемкое мероприятие и может занимать несколько минут!

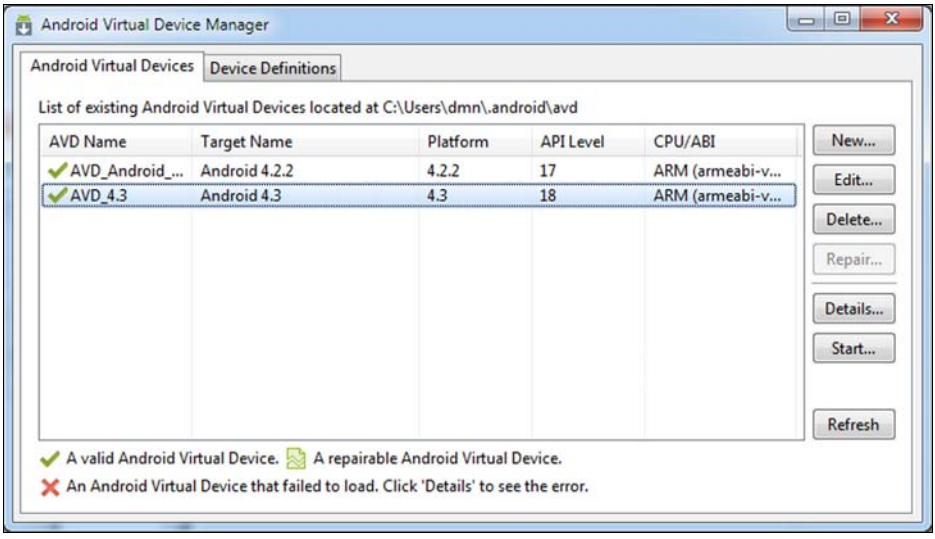

Рис. 1.11. Запуск виртуального устройства

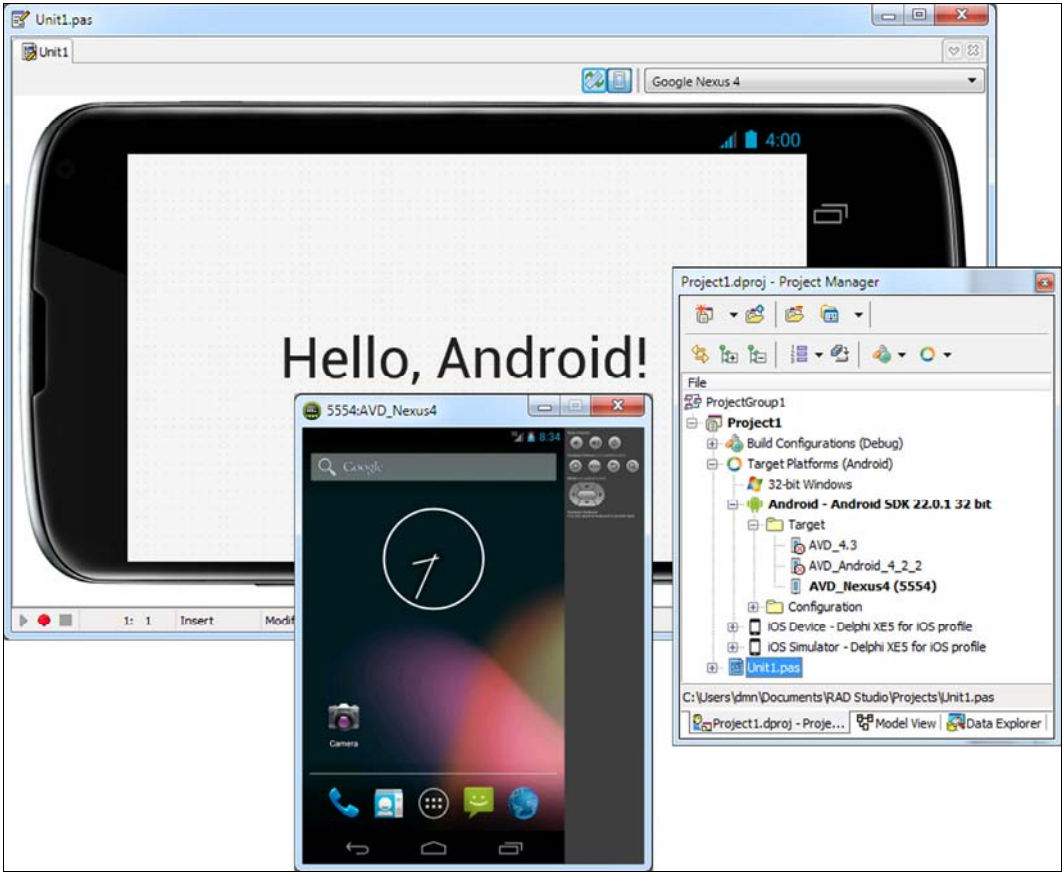

Рис. 1.12. Выбор эмулятора устройства для проверки приложения для Android

4. Подготовив к работе эмулятор Android, мы можем приступать к программированию. Для этого создайте новое приложение для мобильной платформы и в обязательном порядке в дереве менеджера проектов выберите узел с целевым эмулятором (рис. 1.12). Осталось нажать клавишу <F9>...

Что делать, когда код зависит от платформы?

Несмотря на то, что библиотека FireMonkey предлагает нам универсальный инструмент, позволяющий создавать приложения, предназначенные для работы на разных программных платформах, надо понимать, что ни одна из библиотек не в состоянии полноценно заменить родной API целевой операционной системы. Поэтому уже в данной главе позволю дать вам совет — если в вашем проекте потребуется воспользоваться функциями API, принадлежащими исключительно Windows, OS X или iOS, то перед обращением к ним необходимо явным образом указать компилятору об их использовании так, как предложено в листинге 1.1.

Листинг 1.1. Директивы компилятора выбора платформы

```
procedure TForm1.FormShow(Sender: TObject);
begin
{$IFDEF MSWINDOWS}
   //код для операционной системы Windows
   Label1.Text:='Hello, Windows!';
{$ELSE} {$IFDEF MACOS}
   //код для операционной системы OS X
   Label1.Text:='Hello, Mac!';
{$ELSE} {$IFDEF IOS}
    //код для мобильной платформы iOS
    Label1.Text:='Hello, iPhone (iPad)!';
{$ELSE} {$IFDEF ANDROID}
    //код для мобильной платформы Android
    Label1.Text:='Hello, Android!';
 {$ENDIF}
 {$ENDIF}
 {$ENDIF}
 {$ENDIF}
end;
```

Во время компиляции вашего приложения Delphi разберется с подключением к проекту необходимых библиотек и создаст бинарный код для требуемой операционной системы.

Еще одна задача, которую, возможно, придется решать разработчику кроссплатформенного приложения, связана с определением базовых характеристик операционной системы, под управлением которой запускается его творение. Если речь идет о настольном приложении, предназначенном для работы под Windows или OS X, то

наиболее универсальным помощником станет интеллектуальная запись `TOSVersion`, объявленная в программном модуле `System.SysUtils`. С помощью этой записи мы в два счета выясним тип ОС и процессора (листинг 1.2).

Листинг 1.2. Определение ОС и процессора в настольном приложении

```
procedure TForm1.FormShow(Sender: TObject);
var s:string;
begin
//------------------------- OS -------------------------
case TOSVersion.Platform of
  pfWindows : s:='Windows';
  pfMacOS   : s:='OS X';
  pfiOS     : s:='iOS';
  pfAndroid : s:='Android';
  pfWinRT   : s:='Win RT';
  pfLinux   : s:='Linux';
end;
Label1.Text:='Операционная система '+s;
Label2.Text:=TOSVersion.ToString;

//--------------------- CPU -----------------------------
case TOSVersion.Architecture of
  TOSVersion.TArchitecture.arIntelX86 : s:='IntelX86';
  TOSVersion.TArchitecture.arIntelX64 : s:='IntelX64';
  TOSVersion.TArchitecture.arARM32 :    s:='ARM32';
else s:='Неопределена';
end;
Label3.Text:='Архитектура процессора '+s;
Label4.Text:=Format('Ядер %d',[ TThread.ProcessorCount]);
end;
```

Если речь идет о мобильной платформе iOS, то для получения сведений об устройстве и его ОС можно воспользоваться услугами API операционной системы, подключив к проекту модуль `iOSapi.UIKit` (листинг 1.3).

Листинг 1.3. Операционная система и устройство в мобильном приложении

```
uses  iOSapi.UIKit;
{$R *.fmx}

procedure TForm1.FormShow(Sender: TObject);
var  MobileDevice : UIDevice;
     s:string;
begin
MobileDevice := TUIDevice.Wrap(TUIDevice.OCClass.currentDevice);
```

```
 case MobileDevice.userInterfaceIdiom of
    UIUserInterfaceIdiomPhone: s:='iPhone';
    UIUserInterfaceIdiomPad :  s:='iPad';
    else s:='Other';
 end;
Label1.Text:='Устройство '+s;
Label2.Text := Format('ОС: %s %s', [MobileDevice.systemName.UTF8String,
                                    MobileDevice.systemVersion.UTF8String]);
end;
```