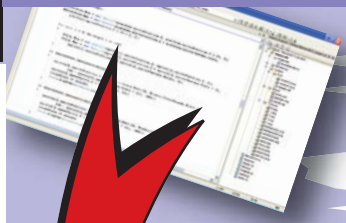
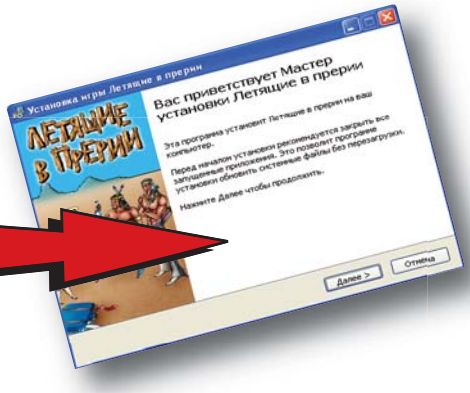
Горнаков С. Г.

Разработка игр под Windows

ДМК ПРЕСС ИЗДАТЕЛЬСТВО



в XNA Game Studio Express



3 CD к одной книге!

- *лицензионный **Visual C# Express!***
- *лицензионный **XNA Game Studio Express!***
- *примеры из книги+материалы конференций!*

УДК 004.4
ББК 32.973.26-018.2
****

Горнаков С. Г.
*** Программирование компьютерных игр под Windows в XNA Game Studio Express. – М.: ДМК Пресс, 2008. – 384 с.: ил.

ISBN ***

Это серия книг настольной библиотеки начинающего программиста игр. На данный момент серия состоит из двух книг и поможет программистам изучить технику разработки игр для системы Windows и Xbox 360. В этом издании с помощью студии XNA Game Studio Express рассматривается полный цикл создания компьютерных игр для операционной системы Windows. Изучая эту книгу, вы освоите основы работ с инструментариями Visual C# Express и XNA Game Studio Express. Научитесь работать с двухмерной и трехмерной графикой, анимацией, познакомитесь с техникой создания игровых классов и формированием механизма игровых состояний. Овладеете секретами создания интерактивных заставок и меню, работой с устройствами ввода и звуком. Итогом книги станет создание двухмерной и трехмерной игры с формированием инсталляционного пакета. На базе полученных знаний вы сможет создавать свои собственные компьютерные игры и продавать или распространять их бесплатно через Интернет. В дополнение на диске имеется потрясающая подборка материала по технике разработки компьютерных игр для операционной системы Windows!

УДК 004.4
ББК 32.973.26-018.2

ISBN ***

Содержание

Часть 1
Введение в программирование игр

Глава 1
В качестве вступительного слова

Важно понимать, что процесс создания игры – это очень трудоемкое дело, отнимающее огромное количество времени из жизни тех людей, которые делают эту работу. Самому, в одиночку, создать полноценную игру фактически нереально, и тем более при всех нынешних технологиях и требованиях к графической картинке. Никогда не пытайтесь в своем первом проекте делать игры плана Gears of War или Need for Speed!

Нельзя откусить больше, чем можно прожевать, не говоря уже о том, что все откушенное нужно проглотить, да еще и не поперхнуться. Создание таких больших и мощных игр – это удел игровых студий с огромным штатом программистов, художников, дизайнеров, аниматоров, модельеров... Если вы в одиночку или небольшим коллективом решите создавать нечто подобное, то скорее всего ваш проект растянется на долгие, долгие годы работы, а за это время все ваши графические инновации станут просто неактуальными. Эта аксиома проверена временем, а ее несоблюдение сгубило уже множество коллективов и программистов-одиночек.

Тогда возникает вопрос: а что тогда можно программировать? Чаще всего программисты-одиночки или коллективы единомышленников создают небольшие, так называемые казуальные игры, или игры, распространяемые через Интернет, стоимость которых составляет не более пары десятков долларов. Набив руку на этом поприще и небольших проектах, эти самые коллективы либо перерастают в более мощные игровые студии, либо сотрудники фирмы находят себе хорошую работу в стане сильных компаний.

В свою очередь, казуальные игры – это огромная индустрия развлечений с большим рынком сбыта по всему миру. Для начала рекомендую вам посетить сайт Европейской конференции казуальных игр (Casuality Europe: East 2007). На этом сайте вы найдете массу ссылок на компании, которые занимаются распространением казуальных игр и которые с большим удовольствием помогут продвинуть на рынок предлагаемую вами игру. Например, всем известная компания Alawar (адрес в Интернете www.alawar.ru) занимается тем, что помогает разработчикам продвигать свой товар на рынке (рис. 1.1).

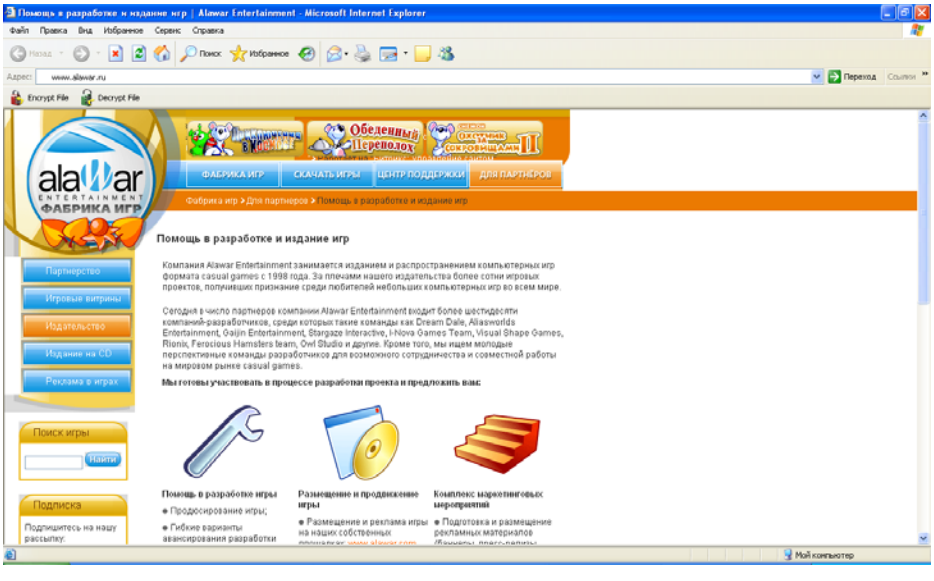


Рис. 1.1. Сайт компании Alawar

В контексте этой книги мы постараемся создать нечто подобное и похожее на казуальную игру. Цель книги заключается в том, чтобы научить вас создавать двухмерные и трехмерные игры для операционной системы Windows. Самый лучший способ научиться программировать игры – это сделать игру своими собственными руками, поэтому на протяжении всей этой книги мы будем создавать две небольшие, пусть и простенькие, но зато свои игры.

В дальнейшем на базе полученных знаний, применив все свое желание и нереализованные идеи, вы вольны сделать нечто большее и лучшее, чем описано в этой книге (я на это искренне надеюсь). Свою задачу в этом плане я вижу в том, чтобы дать вам необходимый запас знаний для старта и толчок в реализации своих идей, все остальное зависит исключительно от вас...

1.1. Этап проектирования игры

Это очень важный и главный этап работы над игрой. На этой стадии необходимо решить массу различных организационных вопросов и выработать общую концепцию игры, создать документацию к игре, проработать дизайн уровней, персонажей, спроектировать игровую модель, сформировать набор игровых классов, разделить проектные задания среди персонала и многое, многое другое. По этой теме написано немало книг, но у нас есть одна небольшая проблема. Дело в том, что вы не умеете программировать игры и до сих пор не сделали ни одной игры. Вы не имеете представления о том, как устроен каркас игровых классов, что нужно сделать для того, чтобы загрузить в игру графику, и т. д. Поэтому для книги была придумана своя модель реализации игр.

Вместо большого этапа проектирования игры мы с вами в каждой последующей главе, строка за строкой, будем писать исходный код игры. Наш каждый следующий проект будет модернизировать предыдущий. Например, в двухмерной игре мы сначала создадим пустой проект, затем выберем разрешение для экрана. Загрузим в игру простое графическое изображение, нарисуем его на экране. Создадим анимацию, увеличим на экране количество изображений, добавим фон, рассмотрим работу с клавиатурой и мышью. Изучим механизм игровых столкновений, добавим в игру меню и заставки – и в итоге создадим своими руками простую двухмерную игру, а весь процесс работы над игрой станет для вас простым учебным пособием, изучив которое вы в дальнейшем сможете создавать свои проекты.

Что касается серьезного подхода в этапе проектирования игры, а также составления необходимой документации к игре, то могу вам порекомендовать интересные материалы на эту тему. В первую очередь это, конечно, книги по этой тематике, которые можно найти в магазинах, но, пожалуй, самое главное, что можно посоветовать по этой теме, – это документация от ведущих отечественных компаний разработчиков игр.

Не поленитесь и зайдите на сайт компании 1С в раздел **Разработчикам**. На этой странице сайта для свободного скачивания доступен потрясающий набор документации, рассказывающий о том, как правильно можно создать дизайн-документ для игры, с чего лучше начать и как лучше всего представить свою игру издателю и т. д. Это просто потрясающая подборка материала, которая проверена жизнью. Аналогичная документация имеется на сайтах компании Бука и Alawar. Это лучшее, что можно порекомендовать вам в этом плане.

1.2. Двухмерная игра

Теперь несколько поясняющих слов об идее двухмерной игры. Основной задачей игры является разностороннее изучение способов работы с двухмерной графикой. В этом контексте базовой идеей для игры послужили игры, где пользователь должен ловить некоторые предметы, падающие с неба. За каждый такой пойман-

ный объект пользователю начисляются очки. Дополнительно с неба падают объекты, которые ловить не нужно, за столкновение с такими объектами очки соответственно отнимаются. По достижении определенного порога очков пользователь переходит на следующий уровень.

Это простой показательный пример, с помощью которого мы сможем проследить за сменой игровых состояний, переходом на новые уровни игры, движением объектов на экране, анимацией, столкновением, работой с клавиатурой и т. д. Главное – разобраться с техникой программирования игр и внутренним механизмом работы платформы XNA. Все остальное, в том числе графика, – дело вторичное.

Оформление и антураж игры – также дело личных и коммерческих предпочтений. Предлагаемая идея игры, чтобы вам было не скучно, «одета» в обвертку индейских прерий. Соответственно игра носит название «Летящие в прерии»... Предположительно все действия в игре разворачиваются в прериях, над которыми терпит крушение самолет, а люди и предметы с самолета падают с неба. У вас как у игрока имеется специальная платформа (с матрасом...), с помощью которой вы должны спасти как можно больше падающих людей. Платформа перемещается исключительно в нижней части экрана с помощью клавиш с командами **Влево** и **Вправо**, и вам нужно подставить ее под падающего человека. Спасли человека – получите очки, набрали потолок очков для текущего уровня – переходите на следующий уровень и т. д.

1.3. Трехмерная игра

В трехмерной игре, так же как и в двухмерной, главной задачей является возможность всестороннего рассмотрения методов работы с 3D-моделями и графикой. По большому счету, работать с трехмерной графикой на порядок сложнее, чем с двухмерной, поэтому уровень изучаемой игры невысок, но этого уровня вполне достаточно, чтобы понять, как работать с 3D-графикой и в каком направлении необходимо двигаться дальше.

В качестве примера в книге была сделана игра «Футбольный стрелок». Суть игрового процесса этой игры состоит в следующем. Сверху на футбольное поле падают три мячика. Задача игрока заключается в том, чтобы стрелять по мячам (курсор мыши – это мушка), подбивая мячи тем самым вверх, не давая в конечном счете одному из мячей коснуться футбольного поля. Как только один из мячей касается футбольного поля, наступает окончание текущего уровня, и игроку предлагается переиграть этот уровень заново.

Для того чтобы перейти на новый уровень, необходимо подбить каждый из мячей по двадцать раз. Самое интересное заключается в том, что даже если игрок подбил один из мячей 20 раз, то мяч все равно будет падать на землю, но его приоритетность в подбитии значительно уменьшается. Как только все три мяча будут подбиты по двадцать раз, то считается, что текущий уровень пройден. На новом уровне скорость падения мячей увеличивается. Подробнее о способах реализации

игры и используемой графике вы узнаете из третьей части книги, но изучать книгу необходимо строго в хронологическом порядке.

1.4. Исходные коды проектов

Я по-прежнему остаюсь сторонником книг, в которых дается полный исходный код рассматриваемого примера. Не всегда удобно, а порой и невозможно иметь под рукой компьютер для просмотра всего примера в целом. Поэтому в книге в каждой главе приводится полный листинг изучаемого в данный момент класса, но поскольку в каждой последующей главе мы модернизируем код, то часть кода методов может не изменяться. В связи с этим в некоторых главах код этих методов может быть вырезан за ненадобностью. Плюс все нововведения в коде выделяются жирным шрифтом, чтобы вам было проще следить за изменениями.

На этом небольшая вступительная часть этой главы подошла к концу, пора переходить от слов к делу!

Глава 2

Платформа XNA

Разработка игр для компьютерных систем, консольных приставок и мобильных устройств – занятие трудоемкое. Если говорить только о специфике создания игр для любой из перечисленных платформ, то можно отметить, что подход в создании игр у каждой платформы будет свой (по крайней мере, так было до недавнего времени). Более того, огромное количество устройств может иметь различное аппаратное обеспечение. Если взять, к примеру, несколько компьютеров, то все они с большой долей вероятности будут компоноваться различными комплектующими. И как быть в этом случае бедному программисту, которому нужно попытаться создать игру для всех этих разных устройств?

Ответ прост: необходимы общие стандарты и спецификации. И они есть, за что им большой и огромный респект от всех нас. Конечно, не все в этом мире идеально и гармонично, те же стандарты и спецификации бывают разными, но! Для компьютерных систем есть свой стандарт – это DirectX и теперь XNA Framework, для мобильных устройств Java 2 ME, Symbian OS & C++ и Windows Mobile & C++ и C#, для Xbox 360 тоже появился свой стандарт в виде платформы XNA. В целом все постепенно унифицируется и сводится к одному стандарту, по крайней мере, в семействе продуктов Microsoft. Кстати, я больше чем уверен, что вы не очень сильно любите Microsoft и дядюшку Билла, но при этом все равно пользуетесь Windows на своем компьютере. Но это не главное, главное другое.

Представьте, если бы у нас сейчас было 10 разных и мощных операционных систем от разных компаний. Думаю, что все эти компании ни за что и никогда не договорились бы об общем стандарте в разработке игр для своих операционных систем. И тогда у нас на рынке сейчас имелся бы не DirectX 10, а десять разных DirectX'ов (или как бы они там назывались)... Со стороны пользователя конкуренция – это просто замечательно, а вот со стороны программиста куча разных стандартов в одной сфере – это ой как тяжело.

2.1. И был сначала DirectX

Осознав факт необходимости целостной спецификации, корпорация Microsoft много лет назад попыталась создать стандартный набор игровых библиотек для операционной системы Windows. В 1995 году на свет появилась первая версия DirectX 1.0 (изначально, правда, была еще одна ранняя версия с названием WinG, но сейчас это не столь важно). Эта версия игровой библиотеки (API) была построена на базе библиотеки Reality Lab компании RenderMorfics, которую Microsoft прикупила по этому случаю.

Первый блин, как всегда, комом, поэтому и у DirectX не все сложилось сразу. Выходили новые версии библиотек, которые сменяли одна другую, а главное – постоянно изменялось содержимое самих библиотек. И только в 1999 году после выхода DirectX 7.1 (2D) и в 2001 году после выхода DirectX 8.1 (с полноценной 3D-поддержкой и первой версией шейдеров) появилась относительная стабильность.

Выход новой версии DirectX 9 в 2002 году привносит ряд изменений в API, но это все плановая модернизация, а не коренное изменение всей целостности библиотеки. В период с 2004 по 2007 год DirectX 9 подвергался мощной перестройке, причем обновления происходили (особенно в 2006 году) буквально через каждые два-три месяца. И уже в 2007 году появилась новая библиотека DirectX 10.

Все изменения в DirectX в основном связаны с перестройкой работы с 3D-графикой и, главное, с некоторыми упрощениями в подходе создания конечного продукта. Нельзя сказать, что перестройка DirectX прошла безболезненно, в том числе и для нас с вами. Вы просто не представляете, какое количество писем я получил за это время! Люди, купившие мою книгу «DirectX 9. Уроки программирования на C++» (дата выхода – 2004 год), абсолютно не понимали, почему новый DirectX SDK не хочет работать с примерами, рассматриваемыми в книге! Но со временем все утряслось, и сейчас на рынке появились стабильные и мощные инструменты, направленные на создание хороших игр.

2.1.1. Уровни абстракции

Схема взаимодействия DirectX и компьютерной системы на уровне абстракции может «вылиться» примерно в следующий рисунок (рис. 2.1). Как видно из этой схемы, любая игра базируется на классах DirectX, которые, в свою очередь, взаимодействуют с операционной системой или специальными сервисами, включающими в себя драйвера устройства и другие API.

Такой подход в реализации многоуровневых прослоек позволяет программисту не обращать внимания на аппаратную часть устройства и спокойно работать с программным кодом. В итоге для программиста абсолютно не важно (в идеале), какое аппаратное обеспечение имеет это устройство, он работает непосредственно с кодом игры и библиотечными классами DirectX.

Останавливаться подробно на устройстве и работе библиотеки DirectX в этой главе и всей книге мы не будем, поскольку к нашей теме DirectX не имеет прямого отношения и служит просто звеном одной цепочки, о котором обязательно нужно было упомянуть.

2.2. Новые реалии

Время вносит в жизнь свои коррективы. В какой-то момент стали очень бурно развиваться мобильные устройства, а также у Microsoft появился определенный интерес к рынку консольных приставок. Этот интерес материализовался кон-

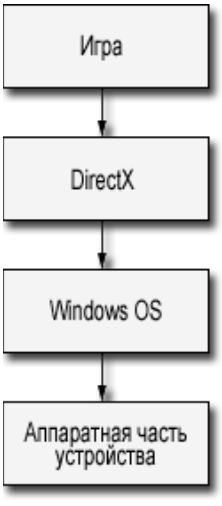


Рис. 2.1. Схема взаимодействия DirectX и аппаратной части устройства

солью Xbox, а затем появилась и вторая версия приставки с несколько видоизмененным названием Xbox 360. После этого корпорация Microsoft имела на руках три разные платформы – компьютерную Windows, мобильную Windows Mobile и консольную приставку. Для всех трех платформ игровая спецификация была абсолютно разной.

Разработчики, создавая игру для PC, могли перенести ее на Xbox, но для этого приходилось тратить очень много времени. Порой проекты для обеих платформ делали две разные команды, а точнее одна команда создавала, а другая адаптировала. С этим нужно было что-то делать.

Создание общей спецификации для всех платформ семейства Microsoft стало большой головной болью для программистов, работающих в корпорации. Чтобы создать такую спецификацию, необходимо было создать общую инструментальную базу, один общий язык программирования, а также общую библиотеку классов. Но задача была решена.

Сейчас в качестве инструментальной базы выступает инструментарий Visual C# Express или Visual Studio. Общим языком программирования стал язык C#, который безусловно проще и лучше своих прародителей C/C++/Java. Общей библиотекой классов стала платформа XNA Framework, которая включает огромный набор системных инструментов для решения большинства серьезных задач. В итоге мы имеем одну спецификацию, меньше головных болей и безболезненное портирование игр с одной платформы на другую. Правда, перед XNA Framework у Microsoft был еще один промежуточный этап понимания того, как должна быть устроена подобная библиотека, и этот этап был ознаменован выходом Managed DirectX.

2.3. Managed DirectX

Основная проблема переноса DirectX на все семейство продуктов Microsoft связана с тем, что библиотека DirectX построена на независимой спецификации модели составных компонентов (COM – Component Object Model), которая, в свою очередь, очень сильно привязана к Windows и языку программирования C++. Поэтому перенос DirectX на все продукты был просто невозможен. Необходимо было выдумать и реализовать нечто новое, простое, да так чтобы еще и работало на всех платформах сразу и одинаково.

Вот так и родилась библиотека *Managed DirectX* для языка C# и .NET-платформы, которая по своей сути представляла системную надстройку классов над стандартным DirectX. В конечном счете на свет появились всего две версии Mana-

ged DirectX. В данный момент эта технология пребывает в летаргическом сне и, по всей видимости, там и будет пребывать еще долгое время, поскольку от развития этой платформы на неопределенное время пришлось отказаться.

В основном проблема Managed DirectX и всего проекта в частности заключалась в том, что Managed DirectX – это всего лишь надстройка над DirectX. В момент создания и развития Managed DirectX никто не принимал во внимание консоль Xbox 360, а также возможность портирования полноценных трехмерных игр на мобильные устройства под управлением Windows Mobile. Когда встал вопрос о том, чтобы абстрагироваться от привязанности к DirectX и Windows, выяснилось, что с Managed DirectX сделать это будет очень-очень трудно. Вот тогда и начались разработки новой платформы, которая впоследствии получила название XNA Framework.

На момент выхода книги работать с мобильными устройствами на базе Windows Mobile 5.0 можно в основном с языком программирования C++. В Visual Studio 2005 есть встроенные инструменты для работы с C# и .NET Compact Framework 1.0. В новой операционной системе Windows Mobile 6.0 используется уже вторая версия платформы .NET Compact Framework 2.0, которая значительно мощнее и лучше.

2.4. Платформа XNA Framework

Платформа *XNA Framework* – это большой набор системных библиотек, построенных на базе .NET-библиотек и направленных на улучшение и упрощение создания игр для всех продуктов семейства Microsoft. Создавая игру для PC на базе платформы XNA Framework, вы можете быть уверены, что эта игра будет работать на приставке Xbox 360 и в скором времени на новой системе Windows Mobile. Исключения в этом случае могут составлять некоторые системные классы, которые необходимы только для работы с той или иной платформой.

Например, приставка Xbox 360 не имеет мыши, поэтому в программах для консоли нельзя использовать методы и классы, направленные на работу с мышью. В свою очередь, для компьютерных игр можно задавать различные разрешения экрана, тогда как Xbox 360 использует в качестве монитора телевизор, который имеет свою специфику вывода изображения на экран. Эти и другие исключения составляют мышиную долю (примерно 3–5%) от всей библиотеки XNA Framework. При правильном подходе в использовании библиотечных классов можно написать программу, которая на 100% будет работать как на консоли, так и на компьютере и, надеемся, на Windows Mobile тоже.

Еще одной важной особенностью платформы XNA Framework является значительное упрощение в подходе реализации игр для всех платформ. Прежде чем начать писать именно исходный код игр с использованием DirectX SDK, необходимо создать большую кучу различных объектов, создать окно, настроить видеоадаптер и т. д. Применяя платформу XNA Framework, вы все перечисленное и

даже больше создадите за пару строк исходного кода, при этом эти строки сформирует за вас сам компилятор и XNA Framework! Работать стало проще, быстрее, и главное – нет борьбы и драк с DirectX... Роль DirectX в этом деле – это низкоуровневая прослойка API, к которой вы не касаетесь никоим образом. Все манипуляции производятся только через библиотеку XNA Framework.

2.4.1. Уровни абстракции XNA

Так же как и в случае с DirectX, мы можем графически представить взаимодействие приложений через XNA Framework с аппаратной частью устройства. Посмотрите на рис. 2.2, где представлены уровни абстракции платформы XNA Framework.

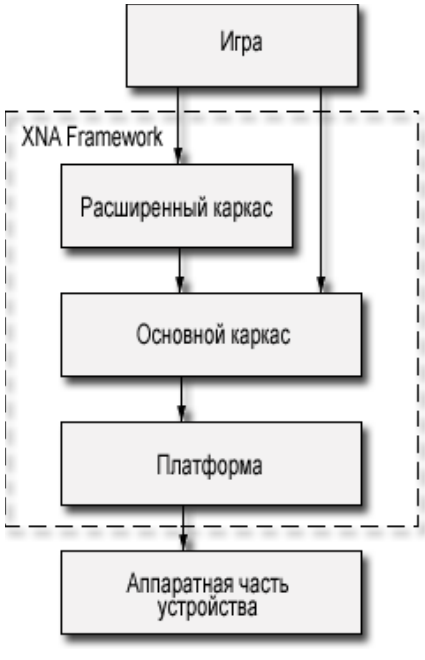


Рис. 2.2. Взаимодействия XNA и аппаратной части устройства

Как видно из рис. 2.2, схема работы XNA Framework напоминает схему работы с DirectX, но в основе библиотеки XNA лежат другие компоненты и механизмы взаимодействия с аппаратной частью устройства. На диаграмме также видно, что вся платформа XNA Framework состоит из трех уровней абстракции, через которые программа обращается к системе.

- **Игра** – это самый высокий уровень абстракции, включающий в себя исходный код игры, различные игровые данные и компоненты. Это все то, что вы пишите и создаете лично сами.
- **Расширенный каркас (Extended Framework)** – расширенный каркас системных высокоуровневых классов содержит простые механизмы для работы с загрузкой моделей, текстур и других графических элементов. В состав этого каркаса входят **Application Model** и **Content Pipeline**. Со временем эта часть библиотеки может пополняться новыми компонентами, что позволит еще больше упростить создание игр. Методы, классы, структуры этого уровня абстракции можно смело использовать в своих программах.
- **Основной каркас (Core Framework)** – основной каркас библиотеки заключает в себе ядро платформы XNA Framework и обеспечивает базовые механизмы работы всей библиотеки в целом. Этот уровень абстракции содержит несколько основных классов, таких как **Graphics**, **Math**, **Input, Audio** и **Storage**. Все классы упрощают работу соответственно с графикой, звуком, устройствами ввода информации, математическими операциями и работой с данными для записи или чтения их с файловой системы. Классы этого уровня вам также будут доступны в полном объеме. Со временем эта часть библиотеки может пополняться новыми компонентами.
- **Платформа (Platform)** – это самый нижний и независимый уровень абстракции или набор классов, осуществляющий обращение непосредственно к аппаратной части устройства. Доступ к компонентам этого уровня у программиста есть, но если их использовать, то это значительно сузит портирование созданного приложения на другие платформы. Более того, не рекомендуется использовать прямое обращение исходного кода игры к прослойке этих компонентов, и по большому счету ваша игра даже не должна подозревать об этом уровне абстракции. В состав этой прослойки входят DirectX, XACT, XInput и XContent.

Подводя промежуточный итог обзора платформы XNA Framework, можно констатировать, что механизмы работы как DirectX, так XNA Framework с аппаратной частью устройства почти одинаковы, но внутренние особенности обеих библиотек значительно отличаются друг от друга, и сейчас мы поговорим об этом более подробно.

2.4.2. Application Model

В расширенном каркасе высокоуровневых классов и, в частности, в **Application Model** (Модель приложения) реализована общая структура работы модели приложения. Этот набор компонентов позволяет автоматизировать процесс создания игры и создает за вас механизм реализации оконного или полноэкранного приложения, организует необходимый в игре таймер, обрабатывает различные сообщения, управляет процессом рисования сцены на экране монитора (рендиринг), загрузкой ресурсов игры и т. д.

Все перечисленные элементы преподносятся вам в виде готового шаблона игры с небольшим количеством строк исходного кода, в которые вам необходимо вставлять свой код игры. То есть за вас создается полнофункциональный каркас игрового приложения, готовый к «употреблению». Например, давайте посмотрим на исходный код каркаса игры, созданный шаблоном XNA Game Studio Express и Visual C# Express, представленный в *листинге 2.1*.

```
//
//=============================================================================
/// <summary>
/// Листинг 2.1
/// Исходный код к книге:
/// "Программирование компьютерных игр под Windows в XNA Game Studio Express"
/// Автор книги: Горнаков С. Г.
/// Глава 2
/// Класс: Game1
/// Простой шаблонный проект
/// <summary>
//
//=============================================================================
#region Using Statements
using System;
using System.Collections.Generic;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Storage;
#endregion

namespace WindowsGame1
{
/// <summary>
/// Это шаблон вашей игры
/// </summary>
public class Game1 : Microsoft.Xna.Framework.Game
{
   GraphicsDeviceManager graphics;
   ContentManager content;

   public Game1()
   {
        graphics = new GraphicsDeviceManager(this);
        content = new ContentManager(Services);
   }

   /// <summary>
   /// Этот метод предназначен для инициализации различных игровых данных
   /// Здесь можно задавать различные первичные состояния игровым компонентам
   /// Но здесь нельзя загружать графику, этот метод предназначен только для
   /// инициализации первичных игровых состояний
```

```
/// </summary>
protected override void Initialize()
{
    // Добавьте здесь код инициализации данных
    base.Initialize();
}

/// <summary>
/// Этот метод предназначен для загрузки графической составляющей в игру
/// </summary>
protected override void LoadGraphicsContent(bool loadAllContent)
{
    if(loadAllContent)
    {
        // Здесь происходит загрузка компонентов в игру
    }

}

/// <summary>
/// Метод для освобождения захваченных системой ресурсов
/// Этот метод автоматически выгружает загруженные компоненты
/// при выходе или закрытии игры
/// </summary>
protected override void UnloadGraphicsContent(bool unloadAllContent)
{
    if (unloadAllContent == true)
    {
        content.Unload();
    }
}

/// <summary>
/// Здесь происходит обновление состояния игровых объектов, логики, звука,
/// получение событий с устройств ввода и так далее
/// Метод реализует простой таймер
/// </summary>
protected override void Update(GameTime gameTime)
{
// Обработка событий, получаемых с джойстика
if(GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
    this.Exit();

    // Добавьте свой код для обновления состояния игровых объектов

    // Это таймер
    base.Update(gameTime);
}

/// <summary>
/// Рендиринг сцены или вывод на экран графики
/// </summary>
protected override void Draw(GameTime gameTime)
{
```

```
        graphics.GraphicsDevice.Clear(Color.CornflowerBlue);

        // Добавьте здесь свой код

        base.Draw(gameTime);
    }
    }
}
```

Класс `Game1` описан в файле с расширением *.cs (`Game1.cs`). Расширение в данном случае указывает на то, что используется язык программирования C#. Весь исходный код создан шаблоном, а перевод комментариев на русский язык осуществлен в вольной форме.

Класс `Game1.cs` создал за нас полное игровое оконное приложение с методами для загрузки графической составляющей игры, инициализации данных, систему вывода сцены на экран, систему обновления состояния игры (таймер) и даже обработку событий, получаемых с джойстика. Неплохо, не правда ли? Более подробно с классом `Game1` мы начнем знакомиться в следующих главах, когда приступим к работе над своей игрой.

2.4.3. Компонент Content Pipeline

Одной из важнейших составляющих любой программы является графический и игровой контент, который включает в себя различные графические изображения (текстуры, спрайты...), звуковые файлы, модели и материалы. Если вспомнить методику загрузки графики в DirectX, то станет понятно, почему в XNA уделено этому большое внимание.

В DirectX, чтобы загрузить модель, приходилось писать десяток-другой строк исходного кода! Более того, загружать можно было только определенные графические файлы, которые DirectX понимал. Для остальных моделей и изображений приходилось писать свои экспортеры или искать инструменты для конвертирования контента в необходимый формат данных.

Используя XNA Framework и **Content Pipeline** (Конвейер контента), который входит в пространство имен `Microsoft.Xna.Framework.Content`, вам не нужно беспокоиться о том, в каком формате представлено изображение или модель. Вы просто пишете пару строк исходного кода, и ваш графический контент будет успешно загружен в игру внутренними средствами конвейера. В *главе 6* мы подробно остановимся на этом вопросе.

Конвейер контента понимает следующие форматы.

- **Трехмерные модели** – формат X и формат FBX. Формат X – это всем известный X-файл, или мэш. Формат FBX – относительно новый формат в игровой индустрии, продвигаемый компанией Autodesk, и поскольку эта же компания является создателем всем известного инструментария 3DMAX Studio, то сомнений в дальнейшем распространении этого формата нет. На сайте компании Autodesk можно бесплатно скачать плагин и конвертер для

преобразования сделанной модели в 3DMAX Studio до версии 8 в формат FBX. В 3DMAX Studio 9 все перечисленные средства встроены по умолчанию.

- **Графические изображения** – поддержаны наиболее распространенные графические форматы DDS, BMP, JPG, PNG и TGA.
- **Материал** – имеется поддержка формата FX, или формата шейдеров.
- **Аудиоформат** – осуществлена поддержка работы с проектами, созданными при помощи XACT. Подробно о самом формате и загрузке в игру звуковых данных вы узнаете в *главе 13*.

Как видите, список поддерживаемых форматов достаточно большой и внушительный, но главное – это простота загрузки, например текстуры или модели в игру. Посмотрите на пример двух строк исходного кода, загружающих в программу графическое изображение и модель.

```
private Texture2D mySprite;
private Model myBoll;
…
mySprite = content.Load<Texture2D>("sprite")
myBoll = content.Load<Model>("boll");
```

В последних двух строках этого блока кода мы загрузили в программу из корневого каталога проекта графическое изображение и модель (для дополнительной или вложенной папки используется следующая запись «`Имя_папки\\boll`»). Вот и все, теперь вы можете использовать загруженные элементы для вывода на экран монитора (о том, как это сделать, в следующих главах).

Если представить себе абстрактно внутренний принцип работы **Content Pipeline** и то, как происходит загрузка контента в игру и его преобразование, то можно перейти к следующей схеме (рис. 2.3). Как видно из этого рисунка, весь игровой контент проходит через конвейер импорта и преобразования данных, после чего на выходе в проекте получаются файлы со специфическим расширением XNB и XACT для звуковых данных. На этапе запуска игры в игровом процессе принимают участие именно преобразованные файлы с расширением XNB и XACT. Все преобразования контента происходят внутренне, без вашего участия и на этапе компиляции проекта в Visual C#.

2.4.4. Компонент Graphics

Графическая часть библиотеки `Microsoft.Xna.Framework.Graphics` основана на принципе работы библиотеки DirectX. Подход в реализации программ при помощи графического конвейера XNA Framework полностью исключает работу с фиксированным конвейером, присущим всем программам с использованием DirectX. Как вы помните, DirectX до последней девятой и десятой версии имел два конвейера: фиксированный и программируемый. Второй конвейер характерен тем, что все работы с вершинами объектов, пикселями текстур, материалом,