Содержание

Об авторе

Алан Торн (Alan Thorn), разработчик игр, независимый программист и писатель, с более чем 13-летним опытом работы, живущий в Лондоне. В 2010 году основал компанию Wax Lyrical Games и является создателем игры «Baron Wittard: Nemesis of Ragnarok», удостоенной многочисленных наград. Автор 10 видеокурсов и 11 книг по разработке игр, в том числе «Unity 4 Fundamentals: Get Started at Making Games with Unity» (Focal Press), «UDK Game Development» и «Pro Unity Game Development with C#» (Apress). Кроме того, как приглашенный лектор читает курс «Game Design & Development Masters Program» в Национальной школе кино и телевидения.

Участвовал как независимый разработчик в более чем 500 проектах по созданию игр, симуляторов, игровых киосков, «серьезных» игр, программ дополненной реальности для игровых студий, музеев и тематических парков по всему миру. В настоящее время работает над приключенческой игрой «Mega Bad Code» для настольных компьютеров и мобильных устройств. Алан обожает графику. Увлекается философией, йогой и пешими загородными прогулками. Его адрес электронной почты: directx_user_interfaces@hotmail.com.

О технических рецензентах

Дилан Агис (Dylan Agis), программист и дизайнер игр, в настоящее время участвует в нескольких сторонних проектах, как независимый разработчик, и одновременно развивает несколько собственных. Имеет большой опыт работы на C++ и C#, а также в Unity, и любит решать проблемы.

Я хотел бы поблагодарить издательство Packt Publishing за предоставленную возможность ознакомиться с книгой и автора за интересное чтение.

Джон П. Доран (John P. Doran), дизайнер игр, созданием которых занимается более 10 лет. Участвовал в разработке разнообразных игр, и в одиночку, и в командах, численностью до 70 человек, в рамках в учебных, и профессиональных проектов.

Одно время работал в компании LucasArts над созданием игры «Star Wars: 1313» как дизайнер-стажер, где был единственным начинающим дизайнером в команде опытных специалистов. Также был ведущим преподавателем на курсах DigiPen®-Ubisoft® Campus Game Programming Program, где обучал студентов-выпускников программированию игр по интенсивной программе.

В настоящее время Джон занимает пост технического дизайнера в отделе исследований и разработки, в институте DigiPen. Кроме того, он преподает и консультирует студентов по нескольким предметам, читает лекции по разработке игр, в том числе на C++, в Unreal, Flash, Unity и др.

Был техническим рецензентом девяти книг по разработке игр и является автором книг «Unity Game Development Blueprints», «Getting Started with UDK», «UDK Game Development [Video]» и «Mastering UDK Game Development HOTSHOT». Все они вышли в издательстве Packt Publishing. Также является соавтором книги «UDK iOS Game Development Beginner's Guide» (Packt Publishing).

Алессандро Моки (Alessandro Mochi) играет в видеоигры со времен появления «Amstrad» и «NES» на всех возможных устройствах: компьютере, консоли и мобильном телефоне. Большие и маленькие видеоигры – его любовь и страсть. Ролевые игры (RPG), стратегии, динамические игры-платфомеры... ничто не ускользнуло от него.

Профессионально занимаясь программированием, имея диплом с отличием в области управления проектами, свободно владея испанским, итальянским и английским языками, он получил глубокое знание многих программ. Всегда готов встретить новые вызовы.

В настоящее время внештатный дизайнер и программист, помогает молодым разработчикам воплотить идеи в реальность. Хотя он часто путешествует по всему миру, его по-прежнему легко найти через его портфолио на www.amochi-portfolio.com.

Райан Уоткинс (Ryan Watkins) любит веселиться. Его можно найти на LinkedIn: www.linkedin.com/in/ryanswatkins.

Предисловие

Книга «Искусство создания сценариев в Unity» – это сжатое и специализированное исследование некоторых продвинутых, нетрадиционных и эффективных методов разработки игровых сценариев на C# в Unity. Это делает книгу очень актуальной, потому что, несмотря на большое число книг «для начинающих» и учебных пособий по Unity, очень немногие из них описывают приемы профессиональной разработки в ясной и структурированной форме. Автор книги предполагает, что вы уже знакомы с основами Unity, такими как импорт ресурсов, проектирование уровней, карты освещения и основы разработки сценариев на C# или JavaScript. Книга сразу начинается с рассмотрения примеров творческого использования сценариев для решения сложных задач, таких как отладка, искусственный интеллект, нестандартное отображение, расширение редактора, анимация и движение, и многое другое. Главная цель заключается не в демонстрации абстрактных принципов и теоретических основ, а в том, чтобы на реальных примерах показать, как применить теорию на практике, что поможет вам в полную силу задействовать свои знания в области программирования для создания качественных игр, которые не просто работают, но работают оптимально. Чтобы получить максимальную отдачу от этой книги, читайте ее главы по порядку, от начала до конца, и используйте навыки обобщенного и абстрактного мышления. То есть, рассматривайте каждую главу, как конкретный пример и демонстрацию более общих принципов, сохраняющихся во времени и пространстве; их можно выделить из конкретного контекста в данной книге и повторно использовать в иных ситуациях, где бы они не потребовались. Проще говоря, рассматривайте приведенные здесь сведения вне связи с конкретными примерами и выбранной мной тематикой, а как весьма актуальные знания для ваших собственных проектов. Итак, давайте начнем.

О чем рассказывается в этой книге

Глава 1 «*Основы C# в Unity*» кратко напоминает основы написания сценариев для Unity. Она не является полным и исчерпывающим руководством по основам. Скорее, это курс повторения для тех, кто ранее уже изучал основы, но, возможно, не писал сценарии некоторое время и был бы рад освежить память перед тем, как начать работу в следующих главах. Если вы знакомы с основами сценариев (такими как классы, наследование, свойства и полиморфизм), можете пропустить эту главу.

Глава 2 «*Отладка*» глубоко исследует процесс отладки. Надежность и эффективность кода нередко зависит от возможности успешно находить и исправлять ошибки при их появлении. Это делает отладку очень важным умением. В этой главе мы не только рассмотрим основы, но и опишем отладку в интерфейсе MonoDevelop, а также установим полезную систему регистрации ошибок.

Глава 3 «*Синглтоны, статические члены, игровые объекты и миры*» исследует широкий спектр возможностей для доступа, изменения и управления игровыми объектами. В частности, мы познакомимся с шаблоном проектирования «Одиночка» (синглтон) для создания глобальных объектов, а также со многими приемами поиска, перечисления, сортировки и размещения объектов. Сценарии в Unity манипулируют объектами в едином игровом мире (пространстве координат), обеспечивая реалистичность игр.

Глава 4 «*Событийное программирование*» рассматривает событийное программирование как важный подход к перестройке архитектуры игры с целью оптимизации. Переложив тяжелую нагрузку с часто возникающих событий и событий обновления на комплекс других событий, можно высвободить много ценного времени для решения других задач.

Глава 5 «*Камеры и отображение сцены*» глубоко исследует работу камер, подробно описывает их архитектуру и настройку отображения сцены. Мы изучим проверку попадания в область видимости, исследуем вопросы отбраковки, познакомимся с такими понятиями, как прямая видимость, ортогональная проекция, глубина, слои, эффекты постобработки и прочее.

Глава 6 «*Работа с фреймворком Mono*» исследует обширную библиотеку Mono и некоторые из ее наиболее практичных классов, от словарей, списков и стеков до таких функций и понятий, таких как строки, регулярные выражения и фреймворк запросов Linq. К концу

этой главы вы овладеете приемами быстрой и эффективной обработки больших объемов данных.

Глава 7 «*Искусственный интеллект*» содержит пример практического применения почти всего описанного ранее в одном проекте для создания искусственного интеллекта, а точнее – умного врага, способного ходить, преследовать, патрулировать, нападать, убегать и искать аптечки для восстановления здоровья. В процессе создания этого персонажа мы рассмотрим вопросы прямой видимости, обнаружения и прокладки маршрутов.

Глава 8 «*Настройка редактора Unity*» описывает редактор Unity, функциональность которого способна удовлетворить самые разные потребности, но иногда бывает нужно нечто большее. В этой главе рассказывается, как создавать классы для настройки самого редактора, чтобы сделать работу в нем удобнее и эффективнее. Мы создадим собственный инспектор свойств и полнофункциональную систему локализации для разработки многоязычных игр.

Глава 9 «*Работа с текстурами, моделями и двухмерной графикой*» содержит описание функций для работы с двухмерными графическими элементами, такими как спрайты, текстуры и элементы пользовательского интерфейса. Двухмерные элементы играют важную роль даже в трехмерных играх, и здесь мы рассмотрим ряд задач работы с двухмерной графикой и эффективные способы их решения.

Глава 10 «*Управление исходными текстами и другие подсказки*» завершает книгу. Она содержит множество советов и рекомендаций, которые не вписываются в какую-либо конкретную категорию, но в целом очень важны. Мы рассмотрим полезные навыки программирования, советы по поддержанию чистоты кода, сериализацию данных, применение систем управления версиями исходных кодов и многое другое.

Что потребуется для работы с книгой

Эта книга, как следует из ее названия, описывает работу с платформой Unity, а это значит, что понадобится только копия Unity. Unity поставляется со всем необходимым для работы с книгой, в том числе со встроенным редактором кода. Дистрибутив Unity можно загрузить с сайта http://unity3d.com/. Приложение Unity поддерживает две основные лицензии, бесплатную и профессиональную. Бесплатная лицензия ограничивает доступ к некоторым функциям, но сохраняет доступность обширного набора основных функций. В целом боль-

шинство глав и примеров в этой книге соответствуют бесплатной версии, то есть, для опробования примеров можно пользоваться бесплатной версией. Тем не менее, некоторые главы и примеры требуют наличия профессиональной версии.

Кому адресована эта книга

Эта книга адресована студентам, преподавателям и специалистам, знакомым с основами Unity и с приемами создания сценариев. Неважно, как давно вы знакомы с Unity, эта книга найдет, что предложить вам, чтобы помочь усовершенствовать приемы разработки игр.

Соглашения

В этой книге используется несколько разных стилей оформления текста для выделения разных видов информации. Ниже приведены примеры этих стилей с объяснением их назначения.

Программный код в тексте, имена таблиц баз данных, имена папок, имена файлов, расширения файлов, адреса страниц в Интернете, пользовательский ввод и ссылки в Twitter будут выглядеть так: «После создания новый файл сценария будет сохранен в папке `Project` с расширением `.cs`».

Блоки программного кода оформляются так:

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class MyNewScript : MonoBehaviour
05 {
```

Когда нам потребуется привлечь ваше внимание к определенному фрагменту в блоке программного кода, мы будем выделять его жирным шрифтом:

```
// Объект следует скрыть, если его координата Y
// получила значение выше 100
bool ShouldHideObject = (transform.position.y > 100) ? true : false;

// Изменить видимость объекта
gameObject.SetActive(!ShouldHideObject);
```

Новые термины и **важные определения** будут выделяться в обычном тексте жирным. Текст, отображаемый на экране, например в меню или в диалогах, будет оформляться так: «Выберите в меню приложения пункт **Assets ⇒ Create ⇒ C# Script**».


Так будут оформляться предупреждения и важные примечания.


Так будут оформляться советы и рекомендации.

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или может быть не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв прямо на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставить комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com, при этом напишите название книги в теме письма.

Если есть тема, в которой вы квалифицированы, и вы заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Загрузка исходного кода примеров

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com или www.дмк.рф в разделе «Читателям – Файлы к книгам».

Загрузка цветных иллюстраций к книге

Вы также можете скачать файл в формате PDF с цветными иллюстрациями и диаграммами к этой книге. Цветные изображения помогут вам лучше понять содержание книги. Загрузить этот файл можно по адресу https://www.packtpub.com/sites/default/files/downloads/0655OT_ColoredImages.pdf.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы удостовериться в качестве наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в тексте или в коде – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от расстройств и поможете нам улучшить последующие версии этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу dmkpress@gmail.com, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в Интернете по-прежнему остается насущной проблемой. Издательства ДМК Пресс и Packt очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в Интернете с незаконно выполненной копией любой нашей книги, пожалуйста, сообщите нам адрес копии или веб-сайта, чтобы мы могли принять меры.

Пожалуйста, свяжитесь с нами по адресу электронной почты dmkpress@gmail.com со ссылкой на подозрительные материалы.

Мы высоко ценим любую помощь по защите наших авторов, и помогающую нам предоставлять вам качественные материалы.

Вопросы

Вы можете присылать любые вопросы, касающиеся данной книги, по адресу dmkpress@gmail.com или questions@packtpub.com. Мы постараемся разрешить возникшие проблемы.

Глава 1

Основы C# в Unity

Эта книга посвящена освоению приемов создания сценариев для Unity, в частности игровых сценариев на языке C#. Перед тем как двигаться дальше, необходимо дать определение понятия освоения приемов создания сценариев. Под освоением подразумевается, что эта книга поможет вам совершить переход от теоретических знаний к более свободному, практическому и продвинутому овладению навыками разработки сценариев. Здесь ключевым является слово «свободное». С самого начала изучения любого языка программирования, в центре внимания неизменно оказывается его синтаксис, правила и законы, то есть формальная часть языка, включающая такие понятия, как переменные, циклы и функции. Однако, по мере накопления опыта, внимание программиста смещается от самого языка к творческим способам его применения для решения насущных задач; от задач, ориентированных на сам язык, к вопросам контекстно-зависимого применения. Следовательно, большая часть этой книги будет посвящена вовсе не формальному синтаксису языка C#.

В следующих главах я буду считать, что вы уже знакомы с основами языка C#. Поэтому далее речь пойдет о конкретных применениях и реальных примерах использования C#. Однако сначала в этой главе основное внимание будет уделено именно основам C#. И это не случайно. Эта глава кратко охватит все основные понятия C#, необходимые для продуктивной работы с последующими главами. Я настоятельно рекомендую прочесть ее от начала до конца, независимо от вашего опыта. Она адресована, прежде всего, читателям, имеющим поверхностное знакомство с C#, но стремящимся углубить свои знания. Однако, она также может помочь опытным разработчикам закрепить имеющиеся знания и, возможно, приобрести новые, свежие идеи. В этой главе я кратко опишу основы C# с нуля, шаг за шагом. Я буду излагать так, как будто вы уже знакомы с основами программирования, может быть на другом языке, но никогда не сталкивались с C#. Итак, начнем.

Почему C#?

Когда дело доходит до сценариев для Unity, перед началом работы над новой игрой всегда возникает вопрос, какой язык выбрать, потому что Unity предлагает выбор. Официально на выбор предлагается три варианта: Boo, C# и JavaScript. В настоящее время не утихают дебаты о том, как правильнее называть JavaScript – «JavaScript» или «UnityScript», – из-за ряда специфичных изменений, внесенных в язык для Unity. Но не это должно нас сейчас волновать. Вопрос в том, какой язык выбрать для проекта. Кроме того, может показаться, что у нас есть еще один вариант – можно выбрать оба языка и писать одни файлы сценария на одном языке, а другие – на другом, фактически смешав языки. Технически это возможно. Unity не запрещает так поступить. Тем не менее это плохо, потому что подобная практика, как правило, приводит к путанице, а также к конфликтам при компиляции, это все равно, что пытаться рассчитать расстояние в милях и километрах одновременно.

Рекомендуемый подход состоит в том, чтобы выбрать один язык и использовать его повсюду в проекте в качестве главного языка. Это упростит работу, но это также означает, что придется выбрать один язык, а от других отказаться. В этой книге выбран язык C#. Почему? Во-первых, не потому, что язык C# лучше других. На мой взгляд, нет абсолютно «лучшего» или абсолютно «худшего» языка программирования. Каждый язык имеет свои достоинства и недостатки, и все языки одинаково хорошо подходят для создания игр в Unity. Основная причина в том, что C# является, пожалуй, наиболее широко используемым и поддерживаемым языком в Unity, и позволяет большинству разработчиков применить уже имеющиеся знания. Большинство учебников по Unity ориентированы на C#, потому что он часто применяется для разработки приложений в других областях. Язык C# исторически привязан к платформе .NET, которая используется в Unity (под именем Mono), к тому же C# напоминает C++, который очень популярен у разработчиков игр. Кроме того, изучив язык C#, вы обнаружите, что ваши знания и умения работать с Unity востребованы в современной игровой индустрии. Таким образом, я выбрал C#, чтобы обеспечить этой книге более широкую аудиторию и позволить вам дополнительно использовать обширный набор уже существующих учебников и литературы. Этот выбор позволит найти применение знаний, полученных при чтении данной книги.

Создание файлов сценариев

Чтобы определить логику игры или поведение ее персонажей, потребуется написать сценарии. Разработка сценариев в Unity начинается с создания нового файла сценария – обычного текстового файла, добавляемого в проект. Этот файл содержит программные инструкции, каждая из которых является командой для Unity. Как уже упоминалось, программный код может быть написан на одном из языков: C#, JavaScript или Boo. В этой книге будет применяться язык C#. В Unity есть несколько способов создания файлов сценария.

Один из них состоит в выборе пункта **Assets** ⇒ **Create** ⇒ **C# Script** (Ресурсы ⇒ Создать ⇒ Сценарий C#) в меню приложения, как показано на рис. 1.1.

Другой способ – щелкнуть правой кнопкой мыши на пустом пространстве в любом месте панели **Project** (Проект) и выбрать в контекстном меню пункт **Create** ⇒ **C# Script** (Создать ⇒ Сценарий C#),

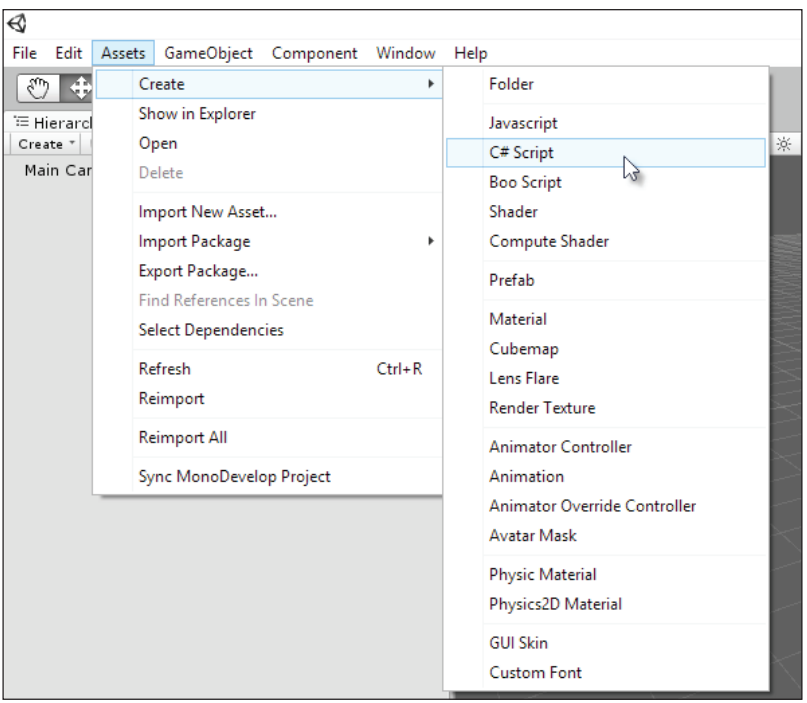

Рис. 1.1. Создание файла сценария с помощью меню приложения

как показано на рис. 1.2. При этом файл сценария будет создан в открытой в данный момент папке.

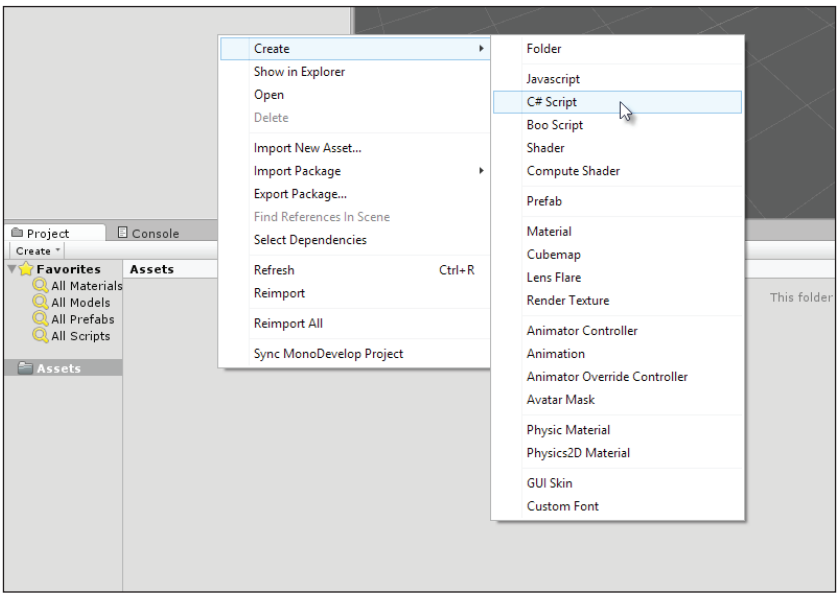

Рис. 1.2. Создание файла сценария с помощью контекстного меню панели **Project**

После этого в папке `Project` будет создан новый файл с расширением `.cs` (сокращенно от C Sharp). Имя файла особенно важно, и его изменение будет иметь серьезные последствия, потому что Unity использует имена файлов для определения имен классов C# в этих файлах. Классы будут рассмотрены более подробно далее в этой главе. Проще говоря, выбирайте для своих файлов уникальные и значимые имена.

Под уникальным именем имеется в виду, что ни какой другой файл в проекте не должен иметь то же имя, независимо от того, в какой папке он находится. Все файлы сценария должны иметь уникальное имя в рамках проекта. Имя должно быть осмысленным и явно выражать назначение сценария. Кроме того, существуют правила, определяющие допустимость имен файлов, а также имен классов в C#. Формальное определение этих правил можно найти по адресу http://msdn.microsoft.com/en-us/library/aa664670%28VS.71%29.aspx. Проще

говоря, имя файла может начинаться только с буквы или символа подчеркивания (цифры для первого символа не подходят), и имя не должно включать пробелов, их рекомендуется заменять символами подчеркивания (_), как показано на рис. 1.3.

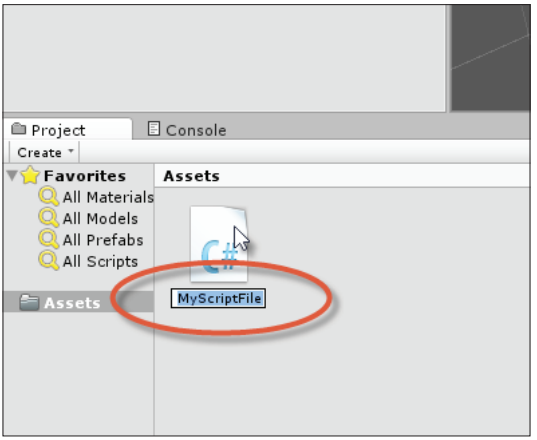

Рис. 1.3. Имена файлов должны быть уникальными и соответствовать принятым в C# соглашениям об именах классов

Файлы сценариев для Unity можно открывать и просматривать в любом текстовом редакторе или интегрированной среде разработки (IDE), в том числе в Visual Studio или Notepad ++, но в состав Unity входит бесплатный редактор исходного кода **MonoDevelop**. Эта программа является частью основного пакета Unity и входит в установочный дистрибутив, но не может быть загружена отдельно. Если дважды щелкнуть на файле сценария в панели **Project** (Проект), файл автоматически откроется в редакторе MonoDevelop. Если потом вы решите переименовать файл сценария, вам также придется переименовать класс C# в файле, чтобы его имя в точности соответствовало новому имени файла, как показано на рис. 1.4. В противном случае будут возникать ошибки во время компиляции и проблемы при подключении файла сценария к объектам.

Компиляция кода. Чтобы скомпилировать код в Unity, достаточно сохранить файл сценария в MonoDevelop, выбрав пункт меню **File** ⇒ **Save** (Файл ⇒ Сохранить) (или нажав **Ctrl**+**S** на клавиатуре), и вернуться в главное окно редактора Unity. При повторном получении фокуса ввода, Unity автоматиче-