ОГЛАВЛЕНИЕ

ОБ АВТОРЕ

Кенни Ламмерс работает в индустрии игр уже 13 лет. Он работал на такие компании, как Microsoft, Activision и Surreal Software. В данный момент он работает с двумя компаниями. В первой – Creative TD – он занимается консультированием по Unity3D и созданием материалов для таких компаний, как IGT, Microsoft, Janus Research и Allegorithmic. Второй компанией – Ozone Interactive – он руководит совместно со своим бизнес-партнёром Ноа Каарбо (Noah Kaarbo). Ozone Interactive специализируется на создании интерактивных приложений и высококачественных дизайнов с использованием Unity3D для таких компаний как Amazon, E-line Media, Microsoft и Sucker Punch games. Во время его работы в индустрии игр он создавал персонажей в Zbrush и Maya, писал шейдеры и постэффекты, а так же разрабатывал игры полностью в Unity3D, используя C#. В данный момент он работает над несколькими играми и разрабатывает набор инструментов для облегчения процесса создания игр.

Я бы хотел высказать благодарность всем её заслуживающим, но тогда на это ушла бы целая глава. Перво-наперво я определённо хочу поблагодарить мою маму за то, что она всегда говорила мне, что моя работа должна вести к моей мечте, и за то, что она всегда была мне поддержкой. Я бы хотел поблагодарить моего партнёра по бизнесу Ноа Каарбо за то, что был мне другом и поддержкой на протяжении написания этой книги. Я хочу поблагодарить всех, с кем мне довелось работать, но, что более важно, – я хочу поблагодарить тех, кто побуждал меня всегда совершенствовать мои навыки и тем самым открыл для меня мир этой индустрии. Эти люди – Бэн Каммерано (Ben Cammerano) из MGS, Пол Амер (Paul Amer) из MGS, Филлипо Костанцо (Fillipo Costanzo) из 5D Institute, Алессандро Тенто (Alessandro Tento) из Lakshya, Джеймс Роджерс (James Rogers) из MGS и Тони Гарсия (Tony Garcia) из Unity Technologies. Я бы не добился того, чего добился, без кого-либо из этих людей, и они заслуживают моего самого сильного почтения!

О РЕЦЕНЗЕНТАХ

Винсент Лим (Vincent Lim) закончил The One Academy по специальности цифровая анимация и разработка игр. Сразу же после завершения учёбы он присоединился к Big Ant Studio, где он преобрёл массу бесценного опыта в игровой индустрии. Проведя несколько лет в компании, Винсент многому научился, начиная от низкополигонального моделирования и заканчивая укладкой текстур для создания ландшафта, а также – немного программированию и написанию MEL-скриптов. Работая над разнообразными задачами в Big Ant Studio, Винсент накопил знания об игровых движках, о тонкостях работы разных шейдеров и о пайплайне разработки игр. В компании он занимался оптимизацией пайплайна разработки, создавая с помощью MEL-скриптов инструменты для художников, которые упрощали те или иные задачи на пути 3D-модели, – от заготовки в 3D-редакторе до модели в игре. Получив достаточный базис знаний в Big Ant Studio, Винсент продолжил изучение игровых механик и движков. Это привело его к Unity, навыки работы с которым он продолжает совершенствовать и по сей день.

Кристиан 'XeviaN' Менегхини (Christian 'XeviaN' Meneghini) в молодости был обладателем и поклонником Sinclair ZX Spectrum. Он начал своё знакомство с миром игровой индустрии со спрайтов, жёстко закодированных с помощью Бейсика и Ассемблера. По прошествии времени ему довелось работать с такими замечательными технологиями, как C64, культовым Amiga и всей линейкой процессоров PC, при этом работая с видеокартами от Hercules и CGA от первых 3D-ускорителей до современных. При его специализации на программировании графики и оптимизации производительности, ему очень нравилось заниматься рендерингом и принимать участие в демосцене. Кристиан в своё свободное время сочиняет музыку.

После нескольких лет работы по ночам вместе с друзьями и коллегами, изучения технической документации, написания движков и работы на другие компании Кристиан в 2011 году совместно со своими друзьями Марко Ди Тимотео (Marco Di Timoteo) и Лукой Марчетти

(Luca Marchetti) основал небольшую студию, которую они назвали STUDIO EVIL. Их первым продуктом стала игра Syder Arcade – ретро-шутер с 3D-графикой, рассчитанный на PC и MAC, а позднее – портированный под iOS, Android и OUYA.

Я бы хотел высказать слова благодарности моим итальянским друзьям разработчикам игр за их приверженность делу развития игровой индустрии в нашей стране.

ПРЕДИСЛОВИЕ

Мы приветствуем вас на страницах книги «Шейдеры и эффекты в Unity». Эта книга поможет вам освоиться с созданием шейдеров и постэффектов в Unity3D. Вы начнёте ваше путешествие по страницам этой книги с самого начала, с создания наиболее базовых шейдеров и получения представления о структуре шейдерного кода. Эти базовые знания пригодятся вам в последующих главах, в которых вы будете создавать шейдеры, имитирующие людскую кожу, шейдеры, обрабатывающие динамические отражения, а также, создавать постэффекты, такие как эффект ночного видения.

К концу каждой главы, у вас сформируется новый набор навыков, с помощью которых, вы сможете повысить качество ваших шейдеров, а также, сделать более эффективным процесс их написания. Эти главы были сформированные таким образом, что вы можете перейти к любой секции и сразу овладеть любым требуемым навыком, развив его от уровня новичка до эксперта. А, если процесс написания шейдеров для вас в новинку, вы можете читать главы последовательно, одну за другой, постепенно приобретая необходимые знания. В любом случае, вы познакомитесь с приёмами, которые используются в большинстве современных игр.

После того, как вы закончите работу с этой книгой, у вас будет набор шейдеров, который вы сможете использовать в ваших играх, сделанных с помощью Unity3D, а также, у вас появится понимание того, как и что нужно будет добавлять к ним, чтобы получать новые эффекты и удовлетворять запросам производительности. Так что, давайте, перейдём к делу.

Что рассматривается в этой книге

Глава 1 «Диффузный шейдинг» содержит азы написания шейдеров, в этой главе объясняется структура шейдеров в Unity3D. Далее в этой главе данные знания применяются для создания диффузной модели

освещения по умолчанию, а также приводятся тонкости и приёмы, используемые в игровой индустрии для создания собственных моделей освещения.

Глава 2 «Создание эффектов с помощью текстур» описывает использование текстур для создания различных эффектов. В этой главе вы узнаете, как можно анимировать спрайт-листы с помощью шейдера, а также, как использовать различные каналы текстуры для повышения эффективности шейдеров. К концу этой главы, ваши навыки использования текстур будут достаточны для создания ваших собственных эффектов.

Глава 3 «Пусть ваши игры засияют отражённым светом» рассказывает всё, что вам может потребоваться, о создании наиболее распространённых типов бликового отражения – Blinn и Phong. Вы узнаете, как можно применить эти шейдерные эффекты для создания маскированного отражённого освещения, металлического отражения, а так же, узнаете о создании анизотропного отражённого освещения. К концу этой главы, вы будете иметь достаточно знаний, для того, чтобы реализовывать свои собственные specular-эффекты.

Глава 4 «Добавим отражения в ваш мир» рассказывает о применении одного из наиболее распространённых эффектов в современных играх, с помощью которого вы сможете учитывать в ваших шейдерах приёмы создания отражений. Эта глава научит вас всему – начиная с азов организации отражения в шейдерах Unity3D и заканчивая созданием вашей собственной системы динамического отражения с помощью C#.

Глава 5 «Модели освещения» рассказывает о более сложных шейдерах. Вы узнаете, как создавать свои собственные модели освещения, использование которых позволит вам сымитировать произвольный тип поверхности. Каждый рецепт демонстрирует применение различных техник для достижения различных задач, каждая из которых приведёт к развитию ваших навыков написания шейдеров. К концу этой главы, вы создадите ваш собственный шейдер кожи, узнаете об освещении Lit Sphere, и напишете свой шейдер автомобильной краски.

Глава 6 «Прозрачность» продемонстрирует то, что на некотором этапе производства игр вам неизбежно потребуется использовать прозрачность. Практически любая игра, так или иначе, задействует прозрачность для таких объектов, как элементы пользовательского интерфейса, опавшая листва, объекты-трафареты, и т. д. В этой главе вы узнаете, как работать с прозрачностью в Unity3D, и как можно уладить трудные моменты, возникающие при этом.

Глава 7 «Волшебные возможности вершин» рассказывает о том, как можно получить доступ к информации, хранящейся в вершинах 3D сетки объекта. Вы научитесь работать с этими данными использовать их в шейдере, для создания таких эффектов, как блендинг текстур и анимация.

Глава 8 «Настройка шейдеров для мобильных приложений» посвящена способам оптимизации шейдеров в Unity, с помощью встроенных типов и макросов. Так как, эта задача становится особенно важной при работе шейдеров для мобильных платформ.

Глава 9 «Делаем наш шейдерный мир модульным с помощью CgIncludes» продемонстрирует вам, почему нужно повторно использовать уже написанный код, для того, чтобы улучшить ваш навык написания шейдеров. Эта глава покажет, как вы можете создать ваши собственные файлы CgInclude, для хранения и повторного использования повторяющихся блоков кода.

Глава 10 «Создание экранных эффектов в Unity с помощью рендер-текстур» начинается с рассмотрения того, как в современных играх используются экранные эффекты (так же, называемый постэффектами) для изменения итогового отрендеренного изображения игры. Вы узнаете, как вы можете создать ваши собственные экранные эффекты, а также, как можно осуществлять корректировку цвета и наложения текстур для создания различных визуальных эффектов в вашей игре.

Глава 11 «Гейм-плей и экранные эффекты» углубляет ваши знания об экранных эффектах, и показывает, как вы можете создать эффекты, усиливающие атмосферность моментов в вашей игре. Вы научитесь создавать эффекты старого фильма и ночного видения.

Что потребуется при чтении этой книги

Для того чтобы вы смогли выполнить рецепты, приводимые в этой книге, вам понадобится следующее необходимое и опциональное программное обеспечение:

- Unity3D (для глав 10 и 11 вам потребуется Unity3D Pro);
- 3D-приложение, такое как Maya, Max или Blender (опционально);
- приложение для работы с 2D-графикой, такое как Photoshop или Gimp (опционально).

Для кого эта книга

Эта книга предназначена для программистов, работающих с Unity3D, для новичков и продвинутых. Лучше всего, чтобы у вас уже был опыт работы с C# или JavaScript, и вы бы уже владели базовыми навыками работы с Unity. Мы советуем вам взглянуть на руководство для новичков в Unity 3.x по разработке игр от Packt Publishing (http://www.packtpub.com/unity-3-x-game-development-by-example-beginners-guide/book), для того, чтобы получить достаточные базовые навыки по работе с азами Unity3D.

Условные соглашения

В этой книге вы встретите несколько стилей оформления текста, которыми форматируется разная по смыслу информация. Сейчас мы приведём примеры этих стилей и объясним их смысл.

Ключевые слова кода будут приводиться в тексте следующим образом: «введите следующий код в блок `Properties` вашего шейдера».

Блоки кода будут оформляться таким образом:

```
void surf (Input IN, inout SurfaceOutput o)
{
     float4 c;
     c =  pow((_EmissiveColor + _AmbientColor), _MySliderValue);

     o.Albedo = c.rgb;
     o.Alpha = c.a;
}
```

Новые термины и **ключевые слова** будут приводится полужирным шрифтом. Слова, которые вы увидите на экране, в меню или, например, в диалоговых окнах, будут приводиться по тексту следующим образом: «таким образом создаётся палитра кубмапы на закладке **Инспектора компонентов**, пользователь может перетаскивать кубмапу на шейдер».

Так будут оформляться предупреждения и важные примечания.

Так будут оформляться подсказки и приёмы.

Обратная связь с читателями

Мы всегда приветствуем обратную связь с нашими читателями. Сообщите нам, что вы думаете об этой книге – что вам в ней понравилось или же не понравилось. Обратная связь поможет нам преподносить именно тот материал, который вам нужен.

Чтобы связаться с нами, просто пошлите письмо по адресу feedback@packtpub.com, а в теме письма укажите название книги.

Если есть предметная область, в которой вы имеете обширный опыт, и вы заинтересованы в написании книги или в другом содействии, то посетите www.packtpub.com/authors.

Поддержка клиентов

Теперь, когда вы стали покупателем книги Packt, перед вами открываются следующие возможности.

Скачивание программного кода примеров

Вы можете скачать файлы программного кода примеров для всех книг Packt, которые вы заказали с вашего аккаунта на страничке http://www.packtpub.com. Если вы заказали эту книгу откуда-то ещё, то вы можете посетить страницу http://www.packtpub.com/support и зарегистрироваться, чтобы мы выслали вам эти файлы электронным письмом.

Скачивание цветных изображений этой книги

Также мы предоставим вам PDF-файл, содержащий цветные изображения скриншотов/диаграмм, использованных в данной книге. Цветные изображения помогут вам лучше понять содержимое глав. Вы можете скачать этот файл с http://www.packtpub.com/sites/default/files/downloads/5084OT_Images.pdf.

Ошибки и опечатки

Несмотря на то что мы предприняли всё возможное для обеспечения точности содержимого наших книг, ошибки иногда всё-таки встречаются. Если вы найдёте ошибку в какой-либо из наших книг – может быть, ошибку в тексте или в коде, – мы будем рады, если вы сообщите

нам об этом. Таким образом вы сможете облегчить жизнь другим читателям и помочь нам улучшить последующие издания книги. Если вы найдёте какую-либо ошибку, то, пожалуйста, сообщите о ней на странице http://www.packtpub.com/submit-errata – выберите там вашу книгу, кликните на ссылку отправки ошибки и введите её описание. Как только присланная вами ошибка будет подтверждена, она будет загружена на наш веб-сайт или добавлена к списку существующих ошибок в секции ошибок для выбранной книги. Найденные ошибки можно посмотреть, выбрав название книги на странице http://www.packtpub.com/support.

Нарушение авторских прав

В Интернете пиратство защищённого авторскими правами материала является насущной проблемой для всех форм материалов. Мы в Packt относимся к защите наших авторских прав и лицензий очень серьёзно. Если вы найдёте в Интернете в любой форме нелегальные копии наших работ, то, пожалуйста, незамедлительно сообщите нам ссылку или название веб-сайта, чтобы мы смогли принять меры.

Пожалуйста, свяжитесь с нами по адресу copyright@packtpub.com и укажите ссылку на предположительно пиратские материалы. Мы ценим вашу помощь в защите прав наших авторов и нашей возможности предоставлять вам ценную информацию.

Вопросы

Вы можете связаться с нами по адресу questions@packtpub.com, если у вас имеются вопросы касательно любого аспекта этой книги, и мы постараемся на них ответить.

ГЛАВА 1
Диффузный шейдинг

В этой главе будут рассмотрены некоторые наиболее распространённые приёмы, используемые сегодня в игровой индустрии при разработке шейдеров. Вы узнаете о том, как:

- создать простой поверхностный шейдер;
- добавить свойства поверхностному шейдеру;
- использовать свойства в поверхностном шейдере;
- сделать собственную модель диффузного освещения;
- написать модель освещения Half Lambert;
- использовать текстуру для контроля над диффузным шейдингом;
- имитировать эффект BRDF с помощью 2D-текстуры.

Введение

В основе любого хорошего шейдера всегда лежит модель освещения, а точнее, его диффузного (рассеивающего) компонента. Так что имеет смысл начинать написание шейдера именно с него.

Ранее в компьютерной графике диффузный шейдинг делали с помощью так называемой **неперепрограммируемой** (fixed function) **модели освещения**. Она предоставляла графическим программистам единственную модель освещения, которую они могли настраивать с помощью набора параметров и текстур. Сейчас же, с появлением шейдеров и языка Cg, мы получили больше возможностей контролировать освещение. Тем более в Unity с его поверхностными шейдерами.

Диффузный компонент шейдера описывает, как свет отражается от поверхности во всех направлениях. Возможно, вам покажется, что это описание очень похоже на принцип работы зеркала, но в действительности это не так. Зеркальная поверхность отражает изображение объектов окружающей среды, в то время как диффузное освещение рассеивает во все направления суммарный свет, испускаемый источ-

никами света, такими как, например, солнце. Отражения мы рассмотрим в последующих главах, а на данный момент, нам просто нужно знать, чем они отличаются от диффузного освещения.

Чтобы создать базовую модель диффузного освещения, нам нужно будет написать шейдер, в который будут передаваться цвет испускаемого излучения, цвет фонового освещения и суммарный свет от всех источников. Следующие рецепты покажут, как сделать законченную модель диффузного освещения, а также продемонстрируют некоторые известные приёмы работы с текстурами, которые пригодятся при создании более сложных моделей.

К концу этой главы вы научитесь создавать простые шейдеры, которые выполняют основные функции шейдинга. Вооружившись этими знаниями, вы сможете создать практически любой поверхностный шейдер.

Создаём простой поверхностный шейдер

В то время как мы продвигаемся дальше по рецептам этой книги, важно, чтобы вы знали, как настроить рабочую среду в Unity так, чтобы можно было работать эффективно и без каких-либо неудобств. Если вы уже знакомы с созданием шейдеров и настройкой материалов в Unity 4, то можете пропустить этот рецепт. Мы приводим его в этой книге для того, чтобы те, кто только начинает работу с поверхностным шейдингом в Unity 4, могли работать с остальными рецептами книги.

Подготовка

Для того чтобы начать работать с этим рецептом, вам потребуется запустить Unity 4 и создать новый проект. Вы также можете воспользоваться прилагаемым к книге проектом, просто добавляя в него свои собственные шейдеры по мере вашей работы с рецептами из книги. Разобравшись с проектом, вы будете готовы окунуться в прекрасный мир шейдинга!

Как это сделать...

Прежде чем мы займёмся нашим первым шейдером, давайте создадим небольшую сцену, с которой мы будем работать. Для этого, в ре-

дакторе Unity зайдите в меню **Game Object** (Игровой объект) | **Create Other** (Создать другое). Там вы можете создать плоскость (Plane), которая будет играть роль земли, парочку сфер (Sphere), к которым мы будем применять наш шейдер, и направленный источник света (Directional Light), чтобы осветить сцену. После того как мы создали сцену, мы можем перейти к следующим шагам написания шейдера:

1. В панели **Project** (Проект) редактора Unity нажмите правой кнопкой по папке **Assets** (Ресурсы) и выберите **Create** (Создать) | **Folder** (Папку).

Если вы используете проект Unity, предоставляемый с этой книгой рецептов, то вы можете перейти к шагу 4.

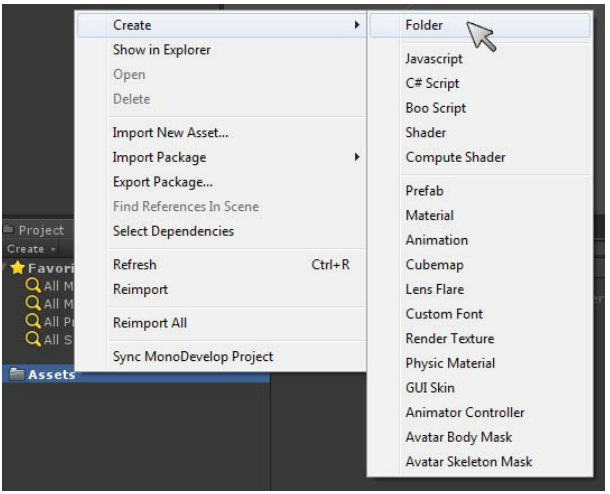

Рис. 1.1. Создание в проекте новой папки

2. Переименуйте папку, которую вы создали, в **Shaders**, нажав на неё правой кнопкой мышки и выбрав **Rename** (Переименовать) из выпадающего списка, либо выбрав папку и нажав **F2** на клавиатуре.
3. Создайте ещё одну папку и назовите её **Materials**.
4. Нажмите правой кнопкой мыши по папке **Shaders** и выберите **Create** (Создать) | **Shader** (Шейдер). После этого нажмите правой кнопкой мыши по папке **Material** и выберите **Create** (Создать) | **Material** (Материал).
5. Переименуйте и Shader, и Material в **BasicDiffuse**.

6. Запустите **BasicDiffuse** шейдер в **MonoDevelop** (редактор скриптов по умолчанию для Unity), сделав двойной щелчок мышкой по нему. Это автоматически запустит редактор и отобразит код шейдера.

Вы увидите, что в только что созданном шейдере уже есть какой-то код. Unity по умолчанию создаёт самый простой diffuse-шейдер с одной текстурой. Изменяя и дополняя этот код, мы научимся разрабатывать наши собственные шейдеры.

7. Теперь давайте переименуем наш шейдер и поместим его в отдельную папку. Первая строчка кода шейдера задаёт его имя и путь, которые будут отображаться в выпадающем списке в Unity при назначении шейдера материалу. Мы переименовали наш шейдер в `"CookbookShaders/BasicDiffuse"`, но вы можете переименовать его во что захотите и когда захотите. Так что сейчас можете за это не волноваться. Сохраните шейдер в MonoDevelop и вернитесь в редактор Unity. Unity автоматически скомпилирует шейдер, когда увидит, что файл был обновлён. На данном этапе ваш шейдер должен выглядеть так:

```
Shader "CookbookShaders/BasicDiffuse"
{
 Properties
 {
   _MainTex ("Base (RGB)", 2D) = "white" {}
 }
 SubShader
 {
    Tags { "RenderType"="Opaque" }
  LOD 200

  CGPROGRAM
  #pragma surface surf Lambert

  sampler2D _MainTex;

  struct Input
  {
   float2 uv_MainTex;
  };

  void surf (Input IN, inout SurfaceOutput o)
  {
   half4 c = tex2D (_MainTex, IN.uv_MainTex);
   o.Albedo = c.rgb;
```

```
    o.Alpha = c.a;
   }
   ENDCG
  }
  FallBack "Diffuse"
}
```

8. Выберите материал **BasicDiffuse**, который мы создали на шаге 4, и посмотрите на панель **Инспектора**. Из выпадающего списка **Shader** (Шейдер) выберите **CookbookShaders | BasicDiffuse** (путь к вашему шейдеру может отличаться, если вы решили использовать другое имя). Таким образом, мы назначим шейдер материалу и сможем теперь применить его к объектам на сцене.

*Чтобы назначить материал объекту, вы можете просто перетащить его из панели **Project** (Проект) на объект в сцене или на панель инспектора при выделенном объекте.*

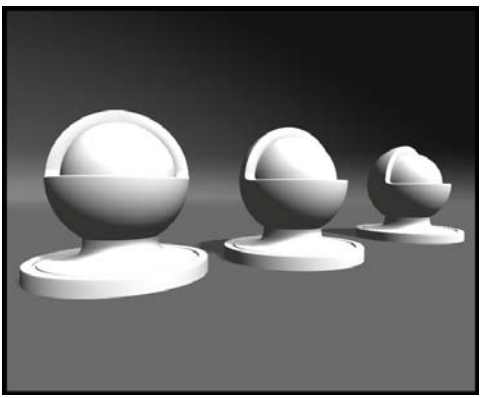

Рис. 1.2. Объекты, используемые в данном рецепте

И хотя пока что особо не на что смотреть, тем не менее мы настроили нашу среду для разработки шейдеров и теперь можем приступить к модификации шейдера под наши нужды.

Как это работает...

Как видите, в Unity настроить среду разработки шейдеров дело буквально нескольких кликов. В поверхностном шейдере незаметно для пользователя работает много компонентов. Unity сделала шейдер-

ный язык Cg более эффективным, генерируя за вас большую часть Cg-кода. Поверхностные шейдеры – это более компонентно ориентированный способ написания шейдеров. Такие задачи, как обработка текстурных координат и матриц преобразований, уже решены за вас, так что вам больше не придётся каждый шейдер начинать с нуля. Раньше же, начав писать новый шейдер, нам бы пришлось переписывать большие куски кода снова и снова. По мере того как вы будете набираться опыта при работе с поверхностными шейдерами, вы, естественно, захотите узнать больше о заложенных в основу функциях языка Cg и о том, как Unity обрабатывает за вас все низкоуровневые задачи, выполняемые **графическим процессором (GPU)**.

Итак, мы создали простой диффузный шейдер, который уже правильно взаимодействует с источниками света и тенями. И всё, что мы сделали, – поменяли одну строчку кода, переименовав наш шейдер.

Дополнительная информация

Если вы хотите узнать подробнее, какие встроенные функции доступны вам при написании поверхностных шейдеров, загляните в папку `Editor\Data\CGIncludes` в установленной версии Unity. В этой папке находятся три файла, на которые вам стоит обратить внимание: `UnityCG.cginc`, `Lighting.cginc` и `UnityShaderVariables.cginc`. На данном этапе наш шейдер использует все эти файлы.

Подробнее на файлах из папки `CGIncludes` мы остановимся в главе 9 «Делаем наш шейдерный мир модульным с помощью CgIncludes».

Добавление свойств поверхностному шейдеру

Свойства шейдера играют очень важную роль в разработке и использовании шейдеров. Для каждого из свойств Unity автоматически создаёт элементы интерфейса в панели **Инспектора**, с помощью которых можно легко настраивать шейдер непосредственно в редакторе. Откройте ваш шейдер в MonoDevelop и обратите внимание на блок кода с третьей по шестую строку. Этот блок называется **Блоком свойств**. На данный момент в нём присутствует лишь одно свойство – **_MainTex**. Если вы посмотрите на ваш материал, который использует этот шейдер, то заметите, что в панели **Инспектора** есть поле для настройки **текстуры**. Это поле было автоматически создано из его описания в блоке свойств.