HZ BOOKS 华章IT
[PACKT] PUBLISHING
游戏开发与设计技术丛书

Unity 着色器和屏幕特效开发秘笈

（原书第2版）

[英] 艾伦·朱科尼（Alan Zucconi）
[美] 肯尼斯·拉默斯（Kenneth Lammers） 著 占红来 译

Unity 5.x Shaders and Effects Cookbook

- 深度挖掘着色器和屏幕特效背后的秘密，创造更加令人惊叹的Unity项目
- 既循序渐进讲解Unity着色器的各种知识，详细介绍各类着色器的创建，又通过丰富实例，深入解析实现后期特效的各种实用技术和方法

机械工业出版社
China Machine Press

游戏开发与设计
—技术丛书—

Unity
着色器和屏幕特效开发秘笈

（原书第2版）

[英] 艾伦·朱科尼（Alan Zucconi）
[美] 肯尼斯·拉默斯（Kenneth Lammers） 著 占红来 译

Unity 5.x Shaders and Effects Cookbook

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Unity 着色器和屏幕特效开发秘笈（原书第 2 版）/（英）艾伦·朱科尼（Alan Zucconi）等著；占红来译．—北京：机械工业出版社，2017.5
（游戏开发与设计技术丛书）
书名原文：Unity 5.x Shaders and Effects Cookbook

ISBN 978-7-111-56442-3

I. U… II. ①艾… ②占… III. 游戏程序－程序设计 IV. TP311.5

中国版本图书馆 CIP 数据核字（2017）第 085370 号

本书版权登记号：图字：01-2016-4516

Unity 着色器和屏幕特效开发秘笈（原书第 2 版）

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：唐晓琳　　责任校对：殷　虹
印　　刷：三河市宏图印务有限公司　　版　　次：2017 年 5 月第 1 版第 1 次印刷
开　　本：186mm×240mm　1/16　　印　　张：11.75
书　　号：ISBN 978-7-111-56442-3　　定　　价：49.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzit@hzbook.com

The Translator's Words 译者序

我从2015年开始接触Unity开发，在此之前接触过很多游戏，也间或见过一些游戏开始时的Unity跳转页面。2014年暴雪发布了一款卡牌游戏《炉石传说》，该游戏画风精美，趣味十足，也深深地吸引了包括我在内的千万玩家。更让我吃惊的是，在该游戏的介绍页面中提到其开发平台使用的正是Unity3D。自此之后我便开始了Unity的学习之旅。

Unity是一个专业的游戏引擎，功能十分强大。但是对于没有接触过任何游戏开发或者没有接触过代码的新手而言，要学会从头到尾地开发一个游戏还是颇有难度的。本书并不涉及Unity游戏开发的所有方面，而是重点关注游戏开发中颇为晦涩难懂的着色器。

着色器是游戏表现超强拟实效果的基石。和大千世界中各种各样的材料对应的是，游戏里面也需要有各种各样的着色器及其组合来对其进行模拟。比如透明的玻璃效果，栩栩如生的动物皮毛效果等。如果读者恰好对这些内容感兴趣，或者需要实现一些类似的需求，那么本书应该很适合你。受篇幅所限，有一些不是本书主要内容的部分作者也提供了很多资料和文档的链接，非常有助于你对某些领域加强理解和认识。

就翻译本身而言，虽然我们会尽力细心地审查和校对，但仍难免有所疏忽。如果在阅读过程中发现一些瑕疵或者有争议的地方，十分欢迎读者联系译者进行核对和改进，以免对大家造成困扰。我的联系方式是SonlyF5020@gmail.com。

在承接本书翻译之时，恰逢我的儿子刚刚出生。当时除了工作的忙碌之外，还得花一部分业余时间来完成翻译，在此对我爱人周彩萍和儿子詹小贤道一声感谢，是你们一直以来的理解和支持才让本书得以完成。

占红来

2017年3月北京

前　言 Preface

本书介绍 Unity 5 中着色器的创建和后期特效开发。你可以从零开始学习创建最基本的着色器，掌握着色器代码是如何组织的。开始的基础知识可以有效地“武装”你，让你在后续如体积爆炸、毛皮着色等章节中游刃有余。本版是专门为 Unity 5 量身定制的，可以通过使用一些基于物理基础的渲染和全局照明来让你的场景栩栩如生。

在每一章的结尾，你都会获得一些新的技巧，比如改善着色器质量或者提升着色器代码编写效率等。这些章节都是量身定制的，所以如果你之前已经有了一些经验，完全可以直接跳到你感兴趣的章节来专门学习。对于新手来讲，可以逐章阅读来构筑整个知识体系。不管使用何种方式，你都会学到制作现代游戏视觉的技术。

在读完本书之后，你手上会有一堆已经做好的着色器，可以在你的 Unity3D 游戏中使用这些着色器，除此之外你会理解如何创建新的着色器，如何完成新的特效以及性能优化等。废话不多说，让我们开始吧！

本书主要内容

第 1 章　会将你引入 Unity 4 和 Unity 5 的着色器编码世界。

第 2 章　介绍表面着色器中的一些非常常用的技术，包括如何给你的模型使用纹理和法线映射。

第 3 章　深度解析着色器是如何给光照行为建模的。本章会教你如何创建自定义光照模型来模拟一些特殊效果，比如卡通着色。

第 4 章　会告诉你基于物理基础的渲染是 Unity 5 中使用的一种模拟现实的基础技术，会教你如何最大限度地使用好基于物理基础的渲染，如何使用透明度、反射型表面和全局照明等。

第 5 章　会教你如何使用着色器来修改物体的几何结构。本章会引入顶点编辑器，使用它可以制作体积爆炸、雪花等生动的特效。

第 6 章　解释如何使用抓取功能来制作一些半透明材料形成的变形效果。

第 7 章　会帮助你对着色器进行一些优化，以保证游戏在各种不同设备上都能正常运转。

第 8 章　展示如何创建特效和其他一些除了 Unity 几乎不可能实现的视觉效果。

第 9 章　会告诉你如何通过后期特效来提升游戏的可玩性，比如夜视效果。

第 10 章　介绍本书中的很多高级技巧，比如毛皮着色和热度图渲染等。

阅读前的准备工作

下面列出的是使用本书时所必需和可选的一些软件：

- Unity 5 (必需)
- 一个 3D 应用程序，比如 Maya、Max 或者 Blender (可选)
- 一个 2D 图像编辑软件，比如 Photoshop 或者 Gimp (可选)

本书的读者对象

如果你想用 Unity 5 来创建你的首个着色器，或者想通过一些专业的后期特效来将你的游戏提升到一个新的高度，这本书就很适合你，但是可能需要一些对于 Unity 的基础理解。

本书结构

在本书中，你会发现有几个频繁出现的标题（准备工作、操作步骤、工作原理、更多内容、参考），这几个标题一般是这样用的：

准备工作

这个部分告诉你预期要做出来的效果是什么，需要准备哪些软件和预先的设置。

操作步骤

这个部分包含了实现的具体步骤。

工作原理

这个部分一般是对操作步骤的详细解释。

更多内容

这一部分由一些相关的附加信息组成，以方便读者对整体内容有更全面的认识。

参考

这一部分会提供一些有用的链接和其他有用信息。

约定

注意以这种方式出现。

提示和技巧以这种方式出现。

下载示例代码

本书提供相关的一些示例代码文件下载，可以访问 http://www.packtpub.com/support 来注册，相关文件会用电子邮件直接发给你。

下载代码文件的步骤如下：

1. 通过电子邮件和密码在上述网站上登录或者注册。
2. 移动鼠标到网站顶部的 SUPPORT 标签处。
3. 点击 Code Downloads & Errata。
4. 输入书名，点击 Search 按钮。
5. 选择你在查找的书籍，下载相关代码文件。
6. 从下拉菜单中选择你的购买渠道。
7. 点击 Code Download。

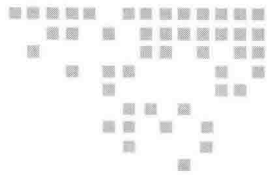
文件下载之后，请使用如下解压软件进行解压：

Windows 系统请使用 WinRAR / 7-Zip，Mac 系统请使用 Zippeg / iZip / unRarX，Linux 系统请使用 7-Zip / PeaZip。

下载本书的彩图

我们还提供本书中所用到的截图、图片的彩图 PDF 文件，这些彩图可以让你更好地理解输出的细微差别。你可以从 https://www.packtpub.com/sites/default/files/downloads/Unity5xShadersAndEffectsCookbook_SecondEdition_Graphics.pdf 处下载。

Contents 目　录



第 1 章 Chapter 1

创建你的第一个着色器

本章会讨论一些游戏开发着色流程中广泛使用的漫反射技术。在这一章中，你会学到如下内容：

- 创建基本的标准着色器
- 从 Unity 4 向 Unity 5 迁移
- 给着色器添加属性
- 在表面着色器中使用属性

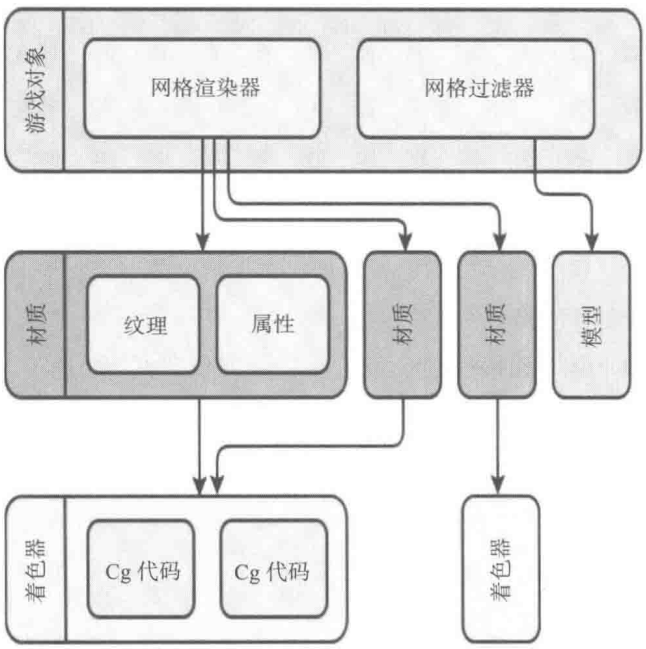
1.1 引言

首先让我们想象一个均匀涂白的立方体。这个立方体的各个面上的颜色都是一样的，但是随着光照方向的不同，以及观察者视角的不同，各个面上呈现出来的影像都是不同的。在 3D 图像技术中，这种级别的拟实效果是通过着色器完成的。着色器是一种特殊的程序，主要用来模拟光照效果。一个木质立方体和一个金属立方体可以共用一个同样的 3D 模型，然后使用着色器就可以让它们看起来大不相同。这一章会循序渐进地介绍 Unity 中的着色器代码。如果你之前没有怎么接触过着色器，经过这一章你就会理解着色器是什么、它们是如何工作的以及怎么对着色器进行自定义了。

在本章的结尾，你应该已经知道了如何创建一个有些基本功能的基础着色器。有了这些知识之后，你就基本上能随心所欲地创建表面着色器了。

1.2 创建基本的标准着色器

每一个 Unity 开发人员都应该熟悉模块 (component) 的概念。游戏中的所有物体都会包含一系列模块，这些模块会影响它的外观和行为。一般而言脚本 (script) 会决定物体的行为，而渲染器 (renderer) 则决定了它在屏幕上呈现出来的外观。Unity 有多种渲染器，根据我们想要显示的物体类型的不同，会使用不同的渲染器。每个 3D 模型一般都会有一个名为 MeshRenderer 的渲染器。一个物体可以只有一个渲染器，但是一个渲染器可以有多种材质 (material)。每一种材质就是一个着色器呈现出来的外观，因此着色器也就是 3D 图像食物链的最后一环。这些模块之间的关系可以从下图看出：



理解了这些模块之间的不同十分有助于我们理解着色器是如何工作的。

1.2.1 准备工作

在准备学习这一部分时，你需要运行 Unity 5 并且创建一个新的项目 (project)。在本书配套的代码里面也包含这样一个初始化好的 Unity 项目，你可以使用这个项目作为基础，随着后续章节的深入学习来自定义你的着色器。准备好这两样东西之后，实时着色的精彩世界已经为你敞开了。

1.2.2　操作步骤

在开始做着色器之前，可以先创建一个小的场景作为基础。创建场景的步骤是在 Unity 编辑器中选择 GameObject | Create Empty。在这个场景中，可以创建一个简单的地平面，再添加几个球体来供我们的着色器使用，再添加一个平行光来照亮这个场景。创建好场景之后，可以按照如下步骤编写着色器：

1. 在 Project 标签页中，右键单击 Assets 文件夹，然后选择 Create | Folder。

如果你使用的是本书配套的项目，可以直接跳到第 4 步。

2. 将你创建的文件夹重命名为 Shaders。重命名方式是右键单击文件夹，然后从弹出的菜单中选择 Rename。或者选中文件夹后按快捷键 F2（这也是 Windows 系统下默认的重命名快捷键）。

3. 创建另外一个名为 Materials 的文件夹。

4. 右键单击 Shaders 文件夹，选择 Create | Shader。然后右键单击 Materials 文件夹，选择 Create | Material。

5. 将新创建的着色器和材质都重命名为 StandardDiffuse。

6. 在 MonoDevelop（Unity 默认的脚本编辑器）中双击打开 StandardDiffuse 着色器。Unity 会自动打开该编辑器并且显示对应的着色器代码。

你会看到 Unity 已经为我们的着色器添加了一些基础代码。默认情况下，这个基础漫反射着色器会接受一个纹理。你可以在这些基础代码的基础上，快速自定义自己的着色器。

7. 现在我们需要声明着色器所在的自定义位置。着色器中的第一行代码就是我们指定给着色器的自定义路径，只有这样 Unity 才会知道这里有一个着色器，在给材质指定着色器的时候，该着色器才会出现在下拉菜单中。我们已经将路径重命名为“CookbookShaders/StandardDiffuse”，但是你完全可以按照自己的喜好给它换个名字。现在不用担心它有任何依赖。在 MonoDevelop 中保存着色器，然后返回 Unity 编辑器。Unity 在识别到着色器文件发生改动时，会自动编译着色器相关代码。确保你的着色器代码是这样的：

```
Shader "CookbookShaders/StandardDiffuse" {
  Properties {
    _Color ("Color", Color) = (1,1,1,1)
    _MainTex ("Albedo (RGB)", 2D) = "white" {}
    _Glossiness ("Smoothness", Range(0,1)) = 0.5
    _Metallic ("Metallic", Range(0,1)) = 0.0
  }
```

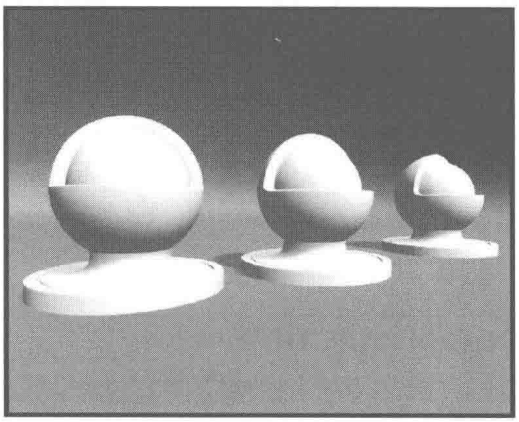
```
SubShader {
  Tags { "RenderType"="Opaque" }
  LOD 200

  CGPROGRAM
  // Physically based Standard lighting model, and enable
    shadows on all light types
  #pragma surface surf Standard fullforwardshadows

  // Use shader model 3.0 target, to get nicer looking
    lighting
  #pragma target 3.0

  sampler2D _MainTex;

  struct Input {
    float2 uv_MainTex;
  };

  half _Glossiness;
  half _Metallic;
  fixed4 _Color;

  void surf (Input IN, inout SurfaceOutputStandard o) {
    // Albedo comes from a texture tinted by color
    fixed4 c = tex2D (_MainTex, IN.uv_MainTex) * _Color;
    o.Albedo = c.rgb;
    // Metallic and smoothness come from slider variables
    o.Metallic = _Metallic;
    o.Smoothness = _Glossiness;
    o.Alpha = c.a;
  }
  ENDCG
}
FallBack "Diffuse"
}
```

8. 技术上来讲，这是一个基于物理基础渲染（physically-based rendering）的表面着色器。在 Unity 5 中已经将物理基础渲染作为标准。顾名思义，这种着色器会通过模拟光照到物体上来获得真实感。如果你使用的是之前版本的 Unity（比如 Unity 4），代码会大不相同。在引入物理基础渲染之前，Unity 4 并没有多少精细技术。关于不同类型着色器的介绍会在本书后面章节中详细展开。

9. 在创建好着色器之后，我们需要将其关联到一种材质上。选择第 4 步中创建的名为 StandardDiffuse 的材质，从 Shader 下拉菜单中查看 Inspector 标签页，选择 CookbookShaders | StandardDiffuse（如果你使用的路径与本书不同的话，看到的着色器路径也会有所不同）。这个操作就会将之前的着色器指定给该材质，接下来你就可以将这个材质指定给某个物体了。

将材质指定给物体的步骤是：从 Project 标签页中将材质拖曳到场景中的物体上。或者将材质拖曳到物体的 Inspector 标签页中也可以。

做完上述步骤之后的例子应该看起来和下面差不多：



现在还没什么好看的，但是我们的着色器开发环境已经准备好了，可以按照需要来修改着色器了。

1.2.3　工作原理

Unity 有一些帮助你准备着色器环境的指令，让你事半功倍。只需要简单地单击几下就已经准备好了。其实在表面着色器的背后，有很多元素在协同工作。Unity 使用的是 Cg 着色器语言，但是针对该语言做了很多优化和提升，以帮助你高效地编写着色器代码。表面着色器语言更多的是一种基于组件的编码方式。诸如处理纹理坐标和转换矩阵的事情，Unity 都已经为你做好了，所以你不用完完全全从头开始。放在以前，我们在创建着色器的时候要反反复复地编写大量重复代码。随着对表面着色器的了解逐步深入，你自然会对 Unity 如何使用这些 Cg 语言相关的底层函数来完成底层图像处理器（GPU）任务感兴趣的。

Unity 中所有文件都是从它们所在的文件夹中独立引用的。我们可以在 Unity 编辑器中移动着色器和材质文件的位置，它们之间的链接是不会被破坏的。但是不能把文件移出 Unity 编辑器，一旦移出了 Unity 就没办法跟踪到这些文件及其引用的更新了。

在简单地修改了一下着色器路径名之后，我们已经得到了一个可以在 Unity 环境中正常工作的基础漫反射着色器了。我们只改了一行代码，所有的光照和阴影都已经做好了。

1.2.4 参考

Unity 5 中内建着色器的源代码是隐藏的，你不能从编辑器中像打开自己的着色器那样打开这些内建着色器。

如果你想知道在哪里可以找到 Unity 中大量的内建 Cg 函数，可以到 Unity 的安装目录，然后进入 Unity45\Editor\Data\CGIncludes。在这个目录下，可以找到这些 Unity 中用到的着色器源码。这些源码本身也是随着时间变化的，如果你想找其他版本的着色器代码，可以去 UNITY DOWNLOAD ARCHIVE（https://unity3d.com/get-unity/download/archive）查找，方法是选中正确的版本之后，从下拉菜单中选择 Build in shaders，如下图所示。这里应该有三个文件：UnityCG.cginc、Lighting.cginc 和 UnityShaderVariables.cginc。当前这个着色器会用到所有三个文件。

在第 10 章中，我们会深入探讨如何使用这些 CgInclude 来将着色器代码模块化。

1.3 从Unity 4向Unity 5迁移

不可否认，电子游戏中的图像技术在过去的 10 年中发生了翻天覆地的变化。每一个包含前沿技术的新游戏的面世，带给我们的都是无与伦比的实时超现实体验。同样，在 Unity 中着色器及其相关技术也是日新月异，这种不断的更新换代很多时候也是大家产生迷惑的根源之一。在 Unity 5 之前，主要有两种着色器被采用：漫反射和高光着色器。顾名思义，这两种着色器分别用在无光和高光材料上。如果你已经用了 Unity 5，可以跳过这一部分，这部分内容主要解释如何使用 Unity 5 重现这些之前的特效。

1.3.1 准备工作

开始这一部分之前，我们需要准备一个 Unity 4 的环境。在 Unity 4 中同样有一些预先提供好的内建着色器。如果你是开始开发一个新游戏，毫无疑问你会使用最新版本的 Unity，但是如果是个遗留项目，很有可能之前使用的就是一些低版本的 Unity。在做版本迁移的时候一定要格外小心。在 Unity 引擎中，很多东西都发生了变化。而且就算是某些内建着色器能正常工作，脚本也可能变了。如果你准备迁移整个工作空间，首要的是要做好备份。很重要的一点是：要记住仅仅保存资源和场景文件是远远不够的，Unity 中大量的配置文件保存在其元数据中。做迁移备份时最安全的方式是将包含项目的整个文件夹拷贝一份。最好的方式是从资源管理器（Windows）或者 Finder（Mac）中将整个文件夹物理拷贝一份。

1.3.2 操作步骤

如果你想将之前做好的着色器迁移到 Unity 5，有两种方式：自动升级项目或统一切换到标准着色器。

自动升级

这是最简单的一种办法。Unity 可以导入之前版本的项目并对其进行升级。你会发现，升级转换完之后，就不能用 Unity 4 了，哪怕你的资源文件一个都没有改动。原因是 Unity 的元数据已经做过转换了。自动升级的步骤是打开 Unity 5，单击 OPEN OTHER 来选择老版本项目所在的文件夹。Unity 会询问你是否需要进行转换。单击 Upgrade 按钮开始转换。Unity 会重新导入你的所有资源文件，重新编译脚本。如果你的项目很大的话，这个过程可能持续数小时。转换完成之后，你就会发现一些改变。例如材质的 Inspector 栏中可能会从 Bumped Diffuse 变成了 Legacy Shader/Bumped Diffuse。

即使 Unity 4 中的诸如漫反射、镜面反射等着色器都已经废弃了，Untiy 5 还是保持了良好的向下兼容性。你可以在材质的下拉菜单中的 Legacy Shaders 目录下找到这些遗留着色器。

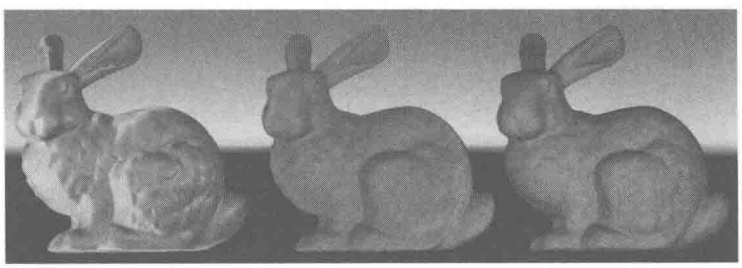
使用标准着色器

如果不想继续使用遗留的着色器，可以使用 Unity 5 中新的标准着色器来取而代之。在替换之前，需要记住一点：这两者是基于不同的光照模型来构建的，因此肯定会影响材质本身的外观。Unity 4 中有超过 80 种不同的内建着色器，总共分为 6 个大类（普通、透明、透明剪切、自照明和反射类等）。在 Unity 5 中，全都被标准着色器取代。不幸的是，并没有什么魔法可以将你的着色器直接转换成标准着色器，但是你可以通过下面这个表格来理

解如何配置 Unity 5 中的标准着色器来模拟 Unity 4 中那些遗留的着色器：

着色器	Unity 4	Unity 4（遗留）	Unity 5
漫反射着色器	漫反射朗伯	遗留着色器 / 漫反射朗伯	标准着色器 基于物理基础的渲染：金属工作流
高光着色器	高光 Blinn-Phong	遗留着色器 / 高光 Blinn-Phong	标准着色器（高光设置） 基于物理基础的渲染：高光工作流
透明着色器	透明顶点	遗留着色器 / 透明顶点	标准着色器 渲染模式：透明
	透明剪切顶点	遗留着色器 / 透明剪切顶点	标准着色器 渲染模式：剪切

可以使用 Inspector 中的 Shader 下拉菜单来让老的材质使用这些着色器，只需要选择合适的标准着色器即可。如果你的老着色器使用了纹理、颜色和法线映射，这些属性会自动应用在新的标准着色器中。你可能还是需要配置一些标准着色器的参数来让它跟之前的光照模型保持一致。下图展示的是一个斯坦福兔子在旧的漫反射着色器（右图）、转换成标准着色器（左图）和光滑度（Smoothness）调为 0 的标准着色器（中图）中的显示效果：



迁移自定义着色器

如果你在 Unity 4 中写过自定义着色器，大部分情况下在 Unity 5 中可以直接使用这些代码。除此之外，Unity 5 中对着色器的工作方式做了些微调，因此可能会引起一些错误或者矛盾。最重要的一点修改是光的亮度。Unity 5 中的光亮度是 Unity 4 中的两倍。所有老的着色器在重写的时候都考虑到了这一点。如果你升级一些内建着色器，或者切换到标准着色器，不会发现任何问题。但是如果你重写过自己的光照模型，需要注意新环境下光亮度不再需要乘以 2 倍了。下面这段代码可以确保这一点：

```
// Unity 4
c.rgb = s.Albedo * _LightColor0.rgb * (diff * atten * 2);
// Unity 5
c.rgb = s.Albedo * _LightColor0.rgb * (diff * atten);
```

如果你没写过着色器也不用害怕。光照模型相关内容会在第 3 章中详细解释。