



[美]Alan Thorn 著 刘君 译

Unity 脚本设计

Mastering Unity Scripting



清华大学出版社

Unity 脚本设计

[美] Alan Thorn 著

刘君译



清华大学出版社

北京

内 容 简 介

本书详细阐述了与 Unity 脚本设计相关的基本解决方案, 主要包括 Unity 中的 C#语言, 调试机制, 单例模式、静态模式、GameObject 以及场景世界, 事件驱动程序设计, 基于高级动画的 Mecanim 系统、相机、渲染和场景, 与 Mono 协同工作, 人工智能, 与纹理、模型和 2D 元素协同工作, 资源控制等内容。此外, 本书还提供了相应的示例、代码, 以帮助读者进一步理解相关方案的实现过程。

本书适合作为高等院校计算机及相关专业的教材和教学参考书, 也可作为相关开发人员的自学教材和参考手册。

Copyright © Packt Publishing 2015. First published in the English language under the title *Mastering Unity Scripting*.

Simplified Chinese-language edition © 2016 by Tsinghua University Press. All rights reserved.

本书中文简体字版由 Packt Publishing 授权清华大学出版社独家出版。未经出版者书面许可, 不得以任何方式复制或抄袭本书内容。

北京市版权局著作权合同登记号 图字: 01-2016-5197

本书封面贴有清华大学出版社防伪标签, 无标签者不得销售。

版权所有, 侵权必究。侵权举报电话: 010-62782989 13701121933

图书在版编目 (CIP) 数据

Unity 脚本设计/ (美) 艾伦·索恩 (Alan Thorn) 著; 刘君译. —北京: 清华大学出版社, 2016
书名原文: *Mastering Unity Scripting*
ISBN 978-7-302-45398-7

I. ①U… II. ①艾… ②刘… III. ①游戏程序—程序设计 IV. ①TP317.6

中国版本图书馆 CIP 数据核字 (2016) 第 260884 号

责任编辑: 贾小红

封面设计: 刘超

版式设计: 李会彤

责任校对: 赵丽杰

责任印制: 何 芊

出版发行: 清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址: 北京清华大学学研大厦 A 座 邮 编: 100084

社 总 机: 010-62770175 邮 购: 010-62786544

投稿与读者服务: 010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈: 010-62772015, zhiliang@tup.tsinghua.edu.cn

印 刷 者: 三河市君旺印务有限公司

装 订 者: 三河市新茂装订有限公司

经 销: 全国新华书店

开 本: 185mm×230mm

印 张: 19.5

字 数: 403 千字

版 次: 2016 年 12 月第 1 版

印 次: 2016 年 12 月第 1 次印刷

印 数: 1~3000

定 价: 69.00 元

译者序

Unity 是近几年非常流行的一款 3D 游戏开发引擎（特别是移动平台），其特点是跨平台能力强，支持 PC、Mac、Linux、iOS、Android、网页等几乎所有平台，移植便捷，3D 图形性能出众，为众多游戏开发者所喜爱。在手机平台中，Unity 几乎成为 3D 游戏开发的标准工具。本书则在此基础上讨论较为高级的开发技术和解决方案。

本书介绍了游戏中的某些高级话题和技巧，并通过 C# 实现 Unity 游戏的脚本程序。书中涵盖了实例特效，并通过 C# 语言讲解专业级的脚本实例，例如，理解 C# 语言的核心组件，包括类基础、接口、单体以及静态对象；实现 NPC 的人工智能行为；与事件驱动程序协同工作，进而优化代码；深入讨论调试和诊断技术；为后处理效果定制渲染功能等内容。

除此之外，读者还可以创建具有人工智能特征的游戏角色，针对后处理效果定义相机渲染，并通过组件结构实现场景管理。为了进一步增加程序的稳定性，本书还深入讨论了 .NET 类，进而处理数据集，例如 CSV 文件，以及高级查询操作的运行机制。本书可有效地提升开发人员的程序设计能力，并帮助程序员熟练地运用其他游戏开发工具箱。

在本书的翻译过程中，除刘君之外，郭志杰、白永丽、赵洪玉、米玥、潘冰玉、李强、皮雄飞、史云龙、王巍、孙年果、程聪、朱利平、王晓晓、解宝香、李保金、王梅、林芮、刘鹤等也参与了本书的翻译工作，在此一并表示感谢。

限于译者的水平，译文中难免有错误和不妥之处，恳请广大读者批评指正。

译者

前 言

针对 Unity 中基于 C#语言的游戏脚本设计，本书简明、扼要地阐述了这一主题。当今市场上充斥着大量 Unity 的入门书籍和教程，但鲜少以专业、结构化的方式探讨这一相对高级的话题。本书假设读者已经熟悉了 Unity 的基本内容，例如数据资源的导入、关卡设计、光照贴图以及 C#或 JavaScript 语言中的脚本机制，通过大量的实例讲述脚本机制与复杂任务间的应用方式，其中包含了调试方法、人工智能、定制渲染、编辑器扩展动画和运动行为等。本书的主要目的并非是叙述抽象的原理，以及理论级的操作提示。相反，本书展示了理论与真实实例之间的实践方法，以帮助读者强化编程技术，进而构建优化的游戏作品。这里，也建议读者采用概括和抽象相结合的思维方式逐章阅读本书内容。具体而言，可将每章内容视为特定的实例，以及通用原理的具体描述。最后，读者可脱离本书的应用环境，并在实际应用中满足具体的需求条件。简而言之，读者不应拘泥于特定示例以及书中提供的学习用例，而是将相关知识运用于自己的开发项目中。

本书内容

第 1 章简述了 C#语言的基本内容，以及 Unity 中的脚本机制。尽管本章并未系统、完整地对此加以介绍，却可帮助读者快速浏览相关知识，或者对其进行简要的回顾。如果读者已对脚本机制的基本内容有所了解（例如类、继承机制、属性以及多态），则可略过本章的学习。

第 2 章深入讨论调试方法。查找并修复错误是编写切实、高效代码不可或缺的技术之一，这也使得调试操作成为一种重要的技能。本章不仅考察其中涉及的基础内容，还将分析 MonoDevelop 基于界面的调试功能，以及如何构建有效的错误日志系统。

第 3 章介绍游戏对象的访问机制、修改方法以及管理操作。特别地，当构建全局持久性对象时，还将使用到单例模式，以及其他诸如搜索、表单、排序以及对象排列等技术。Unity 中的脚本机制依赖于游戏统一场景或坐标系空间内的对象操控行为，进而实现令人可信的效果。

作为游戏体系结构优化的重要途径，第 4 章考察事件驱动编程。对于源自更新和频繁事件造成的重载荷任务，将其转换为事件驱动系统可节省大量的处理时间，进而执行其他任务。

第 5 章将深入探讨相机的工作方式，包括其体系结构的考察方式以及如何定制渲染结果。除此之外，本章还将考察视锥体测试、剔除操作、视线、正交投影、景深和层以及后处理效果等内容。

第 6 章将介绍庞大的 Mono 库及其相关类，包括目录、表、栈以及其他特性和概念，例如字符串、正则表达式以及 Linq。通过本章的阅读，可实现与大量数据间的快速、高效的协同工作。

第 7 章将在独立项目中运用前述知识，并对人工智能进行考察，同时将创建一个敌方角色，进而实现各种行为，其中包括漫游、追逐、巡逻、攻击、逃脱以及搜索能量棒。在该角色的创建过程中，将涉及视线问题、近似检测以及寻路问题。

第 8 章主要讨论 Unity 编辑器，其中涵盖了诸多特性。针对定制编辑器，本章将考察编辑器类的定义方式，进而实现不同的操作行为以便于更好地工作。另外，本章还将创建自定义的面板属性，甚至对于多语言间的无缝切换，还将构建一个全功能定位系统。

第 9 章与 2D 元素有关，例如精灵对象、纹理以及 GUI 元素。即使在 3D 游戏中，2D 元素也扮演了重要的角色，本章将对此制订有效的解决方案。

第 10 章涵盖了一些综合性的内容，并对某些难以分类的提示与技巧（包括有用的概念和应用）加以讨论。相关内容对于问题的整体考察十分重要，例如良好的编码实践、如何编写清晰的代码、数据序列化、资源和版本控制一体化操作等。

背景知识

本书内容主要关注 Unity，这也意味着，读者需要获取一份 Unity 副本，其中包含了本书所需的全部内容，例如代码编辑器。读者可访问 <http://unity3d.com/> 下载 Unity。作为独立的应用软件，Unity 支持两种授权许可，即免费版和高级版。其中，免费版本将对某些特性的访问予以限制，但依然提供了强大的特性集。总体而言，本书大多数章节和示例均兼容于 Unity 免费版。因此，免费版可视为读者的最佳选择。相应地，某些章节和示例则需要使用高级版本。

适用读者

本书适用于学生、教师，以及熟悉 Unity 脚本基础内容的专业人员。无论是初级读者抑或是高级开发人员，本书均提供了极具价值的信息，以帮助读者提升游戏开发技能。

本书约定

本书涵盖了多种文本风格，进而对不同类型的信息加以区分。下列内容展示了对应示例及其具体含义。

文本中的代码、数据库表名称、文件名称、文件名、文件扩展名、路径名、伪 URL、用户输入以及推特用户名采用如下方式表示：

“当构建完毕后，新的脚本文件位于包含 .cs 文件扩展名的 Project 文件夹中”。

代码块则通过下列方式设置：

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

当某些代码行希望引起读者注意时，将会采用黑体表示，如下所示：

```
//We should hide this object if its Y position is above 100 units
bool ShouldHideObject = (transform.position.y > 100) ? true: false;

//Update object visibility
gameObject.SetActive(!ShouldHideObject);
```

 图标表示较为重要的概念，而  图标则表示提示或相关操作技巧。

读者反馈和客户支持

欢迎读者对本书的建议或意见予以反馈，以便进一步了解读者的阅读喜好。反馈意见对于我们来说十分重要，以便改进我们日后的工作。

对此，读者可向 feedback@packtpub.com 发送邮件，并以书名作为邮件标题。

若读者针对某项技术具有专家级的见解，抑或计划撰写书籍或完善某部著作的出版工作，则可阅读 www.packtpub.com/authors 中的 **author guide** 一栏。

对于本书的读者，我们将对每一名用户提供竭诚的服务。

资源下载

读者可访问 <http://www.packtpub.com> 下载本书中的示例代码文件；或者访问 <http://www.packtpub.com/support>，经注册后可直接通过邮件方式获取相关文件。

另外，我们还以 PDF 文件方式提供了本书中截图/图表的彩色图像，以帮助读者进一步理解输出结果中的变化，读者可访问 https://www.packtpub.com/sites/default/files/downloads/0655OT_ColoredImages.pdf 下载该 PDF 文件。

勘误表

尽管我们在最大程度上做到尽善尽美，但错误依然在所难免。如果读者发现谬误之处，无论是文字错误抑或是代码错误，还望不吝赐教。这对于其他读者以及本书的再版工作，将具有十分重要的意义。对此，读者可访问 <http://www.packtpub.com/submit-errata>，选取对应书籍，单击 **Errata Submission Form** 超链接，并输入相关问题的详细内容。经确认后，填写的内容将被提交至网站，或添加至现有勘误表中（位于该书籍的 **Errata** 部分）。

另外，读者还可访问 <http://www.packtpub.com/books/content/support> 查看之前的勘误表。在搜索框中输入书名后，所需信息将显示于 **Errata** 项中。

版权须知

一直以来，互联网上的版权问题从未间断，Packt 出版社对此类问题异常重视。若读者在互联网上发现本书任意形式的副本，请告知网络地址或网站名称，我们将对此予以处理。

关于盗版问题，读者可发送邮件至 copyright@packtpub.com。

对于读者的爱护，我们表示衷心的感谢，并于日后向读者呈现更为精彩的作品。

问题解答

若读者对本书有任何疑问，均可发送邮件至 questions@packtpub.com，我们将竭诚为您服务。

目 录

第 1 章 Unity 中的 C#语言	1
1.1 为何选择 C#语言	1
1.2 创建脚本文件	2
1.3 脚本的实例化操作	4
1.4 变量	6
1.5 条件语句	7
1.5.1 if 语句	8
1.5.2 switch 语句	10
1.6 数组	13
1.7 循环	16
1.7.1 foreach 循环	16
1.7.2 for 循环	17
1.7.3 while 循环	18
1.7.4 无限循环	20
1.8 函数	20
1.9 事件	23
1.10 类和面向对象程序设计	24
1.11 类和继承机制	26
1.12 类和多态	28
1.13 C#属性	32
1.14 注释	34
1.15 变量的可见性	37
1.16 ?操作符	38
1.17 SendMessage 和 BroadcastMessage	38
1.18 本章小结	40
第 2 章 调试机制	41
2.1 编译错误和控制台	41

2.2	利用 Debug.Log 进行调制——定制消息	44
2.3	覆写 ToString 方法	46
2.4	可视化调试	50
2.5	错误日志	52
2.6	编辑器调试	56
2.7	使用分析工具	59
2.8	基于 MonoDevelop 的调试	62
2.9	Watch 窗口	66
2.10	恢复执行程序 and 步进操作	70
2.11	调用栈	71
2.12	Immediate 窗口	73
2.13	设置条件断点	74
2.14	跟踪点	76
2.15	本章小结	78
第 3 章	单例模式、静态模式、GameObject 以及场景世界	79
3.1	GameObject 对象	79
3.2	组件间的交互方式	81
3.2.1	GetComponent 函数	82
3.2.2	获取多个组件	83
3.2.3	组件和消息	84
3.3	GameObject 和场景世界	85
3.3.1	获取 GameObject	86
3.3.2	对象比较	88
3.3.3	获取最近对象	88
3.3.4	获取特定类型的对象	89
3.3.5	GameObject 之间的路径	90
3.3.6	访问对象的层次结构	92
3.4	场景、时间和更新操作	93
3.4.1	规则 1——帧的重要性	95
3.4.2	规则 2——相对于时间的运动	95
3.5	永久对象	96
3.6	理解单例模式和静态模式	98

3.7 本章小结	101
第 4 章 事件驱动程序设计	102
4.1 事件	102
4.2 事件管理	106
4.2.1 基于接口的事件管理	107
4.2.2 定义 EventManager	109
4.3 MonoDevelop 中的代码折叠——#region 和#endregion	114
4.3.1 使用 EventManager	115
4.3.2 基于委托机制的替代方案	116
4.3.3 MonoBehaviour 事件	121
4.3.4 鼠标事件	122
4.3.5 应用程序焦点和暂停	125
4.4 本章小结	127
第 5 章 相机、渲染和场景	128
5.1 相机 Gizmo	128
5.2 可见性	131
5.2.1 检测对象的可见性	132
5.2.2 关于对象可见性的其他问题	133
5.2.3 视锥体测试——渲染器	134
5.2.4 视锥体测试——点	135
5.2.5 视锥体测试——遮挡	136
5.2.6 相机前、后视觉	137
5.3 正交相机	138
5.4 相机渲染和后处理	142
5.5 相机震动	148
5.6 相机和动画	150
5.7 相机和曲线	152
5.8 本章小结	158
第 6 章 与 Mono 协同工作	159
6.1 表和集合	160
6.1.1 List 类	160

6.1.2	Dictionary 类	163
6.1.3	Stack 类	164
6.2	IEnumerable 和 IEnumerator 接口	166
6.3	字符串和正则表达式	172
6.3.1	null、空字符串和空格	172
6.3.2	字符串比较	173
6.3.3	字符串的格式化	174
6.3.4	字符串循环	175
6.3.5	创建字符串	176
6.3.6	搜索字符串	176
6.3.7	正则表达式	176
6.4	无穷参数	178
6.5	语言集成查询	178
6.6	Linq 和正则表达式	181
6.7	与文本数据资源协同工作	182
6.8	从本地文件中加载文本数据	184
6.8.1	从 INI 文件中加载文本数据	185
6.8.2	从 CVS 文件中加载文本数据	187
6.8.3	从 Web 中加载文本数据	187
6.9	本章小结	188
第 7 章	人工智能	189
7.1	游戏中的人工智能	189
7.2	开始项目	191
7.3	烘焙导航网格	192
7.4	NPC 主体对象	195
7.5	Mecanim 中的有限状态机	198
7.6	C#语言中的有限状态机	202
7.7	构建 Idle 状态	204
7.8	构建 Patrol 状态	207
7.9	构建 Chase 状态	211
7.10	构建 Attack 状态	213
7.11	构建 Seek-Health (或逃跑) 状态	214
7.12	本章小结	217

第 8 章 定制 Unity 编辑器	219
8.1 批量重命名	219
8.2 C#属性和反射	224
8.3 颜色混合	227
8.4 显示属性	232
8.5 本地化	238
8.6 本章小结	246
第 9 章 与纹理、模型和 2D 元素协同工作	247
9.1 天空盒	247
9.2 过程式网格	252
9.3 UV 动画——纹理滚动	259
9.4 纹理绘制	261
9.4.1 创建纹理混合着色器	262
9.4.2 创建纹理绘制脚本	265
9.4.3 设置纹理绘制	272
9.5 本章小结	275
第 10 章 资源控制和其他	276
10.1 Git——资源控制	276
10.1.1 下载	277
10.1.2 构建 Unity 项目	278
10.1.3 基于源控制配置 Unity	279
10.1.4 构建 Git 存储库	280
10.1.5 忽略文件	281
10.1.6 创建首次提交	282
10.1.7 修改文件	284
10.1.8 从存储库中获取文件	285
10.1.9 浏览存储库	288
10.2 资源文件夹和外部文件	289
10.3 AssetBundles 和外部文件	290
10.4 持久数据和游戏保存	294
10.5 本章小结	298

第 1 章 Unity 中的 C#语言

本书阐述 Unity 的脚本设计，因而读者需要了解 Unity 游戏开发环境下的 C#语言。在进一步阅读之前，读者有必要明晰相关概念，进而可在理论上掌握脚本设计这一高级内容，此类内容多具有衔接性和实践性特征。关于衔接性，任何一种程序设计语言均会强调语法及其编程规则，这也是一种语言的正式内容之一，其中涉及变量、循环以及函数。随着程序员经验的不断增加，其关注点逐渐从语言本身转向对实际问题的处理，即由语言自身内容转向特定环境下的语言应用。因此，本书并非是一本 C#语法书籍。

在结束本章的学习后，相信读者已经掌握了 C#语言的基本内容，后续章节将运用 C#语言处理相关案例以及实际问题，这也是本书的特点之一，并覆盖了 C#语言的全部功能项，以使读者更好地理解相关操作结果。无论经验如何，这里建议读者逐章阅读，对于期望解决复杂问题的 C#语言新手而言尤其如此。对于经验丰富的开发人员，本书则可强化其现有的知识，并在学习过程中提供新的建议和理念。本章将采用循序渐进的方式，从头开始阐述 C#语言的基础内容。另外，如果读者熟悉另一门语言的编程知识，且尚未接触过 C#语言，现在则是学习该语言的良好时机。

1.1 为何选择 C#语言

当提及 Unity 脚本设计时，面临的一个问题则是选取哪一种语言，Unity 对此提供了解决方案。相应地，官方选取方案则是 C#和 JavaScript 语言。然而，考虑到基于 Unity 的特定应用，JavaScript 应称作 JavaScript 或是 UnityScript 尚存争论，但其中原因并非是本书讨论的重点。当前问题是项目所选取的设计语言。作为一种方案，可在项目中选择两种语言，同时在其中分别编写脚本文件，并对这两种语言进行混合。当然，这在技术上是可行的，Unity 对此并未加以限制，但这会导致混淆以及编译冲突，就像尝试同时以英里和千米为单位计算距离。

因此，这里建议采用一种语言，并在项目中作为主语言加以使用。本书则选用了 C#语言，其原因在于：首先 C#语言并非优于其他语言，根据个人观点，此处并不存在绝对意义上的优劣性，每种语言均包含各自的优点和应用场合；同时，所有 Unity 语言均可用于游戏制作。这里选择 C#语言的主要因素在于其应用的广泛性，以及对 Unity 的支持。

针对 Unity, C#语言可最大限度地与开发人员现有的知识体系结构相结合。大多数 Unity 教程均采用 C#语言编写,同时也常见于其他应用开发领域中。C#语言的历史可追溯至.NET 框架,后者也可用于 Unity 中(称作 Mono)。另外,C#语言也借鉴了 C++语言的内容。在游戏开发中,C++则是一类主要的开发语言。通过学习 C#程序设计语言,读者可向当今游戏界的 Unity 程序开发人员看齐。因此,本书选用了 C#语言,进而扩大其应用范围,在现有教程以及资源的基础上,最大限度地发挥读者的知识水平。

1.2 创建脚本文件

当定义游戏的逻辑或行为时,用户需要编辑相应的脚本文件。Unity 中的脚本机制始于新文件的创建,即添加至项目中的标准文本文件。该文件定义了一个程序,并列出了 Unity 需要理解的全部指令。如前所述,对应指令可通过 C#、JavaScript 或 Boo 语言编写,而本书则选用了 C#语言。在 Unity 中,存在多种方式可创建脚本文件。

其中,一种方法是从应用菜单中选择 Assets | Create | C# Script 命令,如图 1-1 所示。

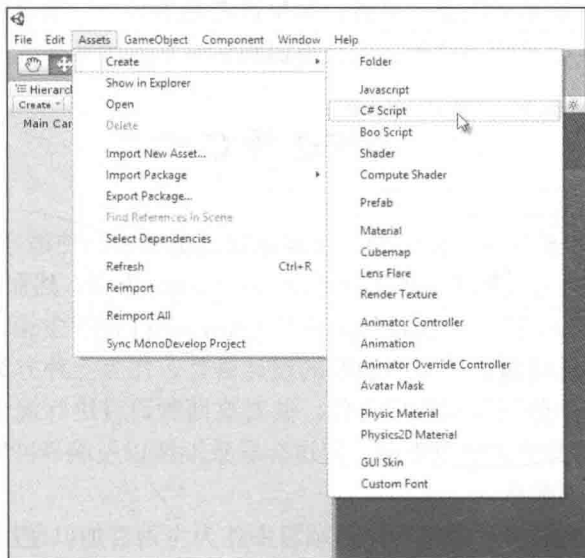


图 1-1

另一种方法则是右击 Project 面板中的空白区域,并在快捷菜单中选择 Create1 的 C# Script 命令,如图 1-2 所示。这将在当前开启的文件夹中创建数据资源。

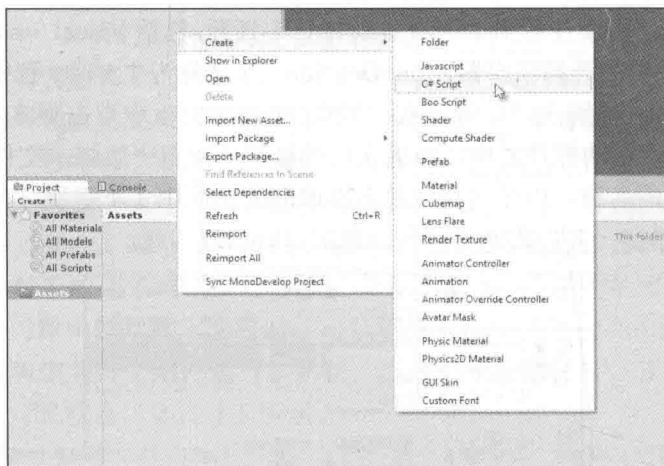


图 1-2

当创建完毕后，新的脚本文件将位于 **Project** 文件夹内，且包含 **.cs** 扩展名（表示 C# 文件）。该文件名十分重要，并对脚本文件的有效性产生重要影响——Unity 使用该文件名确定创建于该文件内的 C# 类的名称。本章稍后将对类加以深入讨论。简而言之，用户应确保文件包含唯一且具有实际意义的名称。

关于唯一性，其含义是指项目中的其他文件名不应与此相同，无论该文件是否位于不同的文件夹内。也就是说，全部脚本文件在项目中应具有唯一的名称。另外，文件名应具有实际意义，并表达脚本行将执行的任务。进一步讲，C# 语言中存在多种有效规则可对文件名和类名予以限定。关于此类规则的正式定义，读者可访问 <http://msdn.microsoft.com/en-us/library/aa664670%28VS.71%29.aspx>。简单地讲，文件名应始于字母或下划线字符（不允许采用数字作为首字符）；同时，文件名不应包含空格。对应示例如图 1-3 所示。

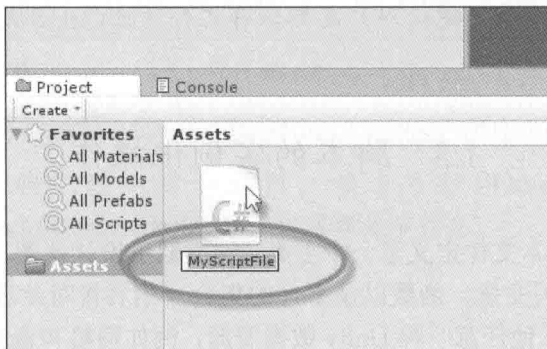


图 1-3

Unity 脚本文件可在任意文本编辑器或 IDE 中打开,包括 Visual Studio 和 Notepad++,但 Unity 提供了免费的开源编辑器 MonoDevelop。该软件为主 Unity 包中的部分内容,并包含于安装过程中,因而无须单独下载。当在 Project 面板中双击脚本文件时,Unity 将在 MonoDevelop 内自动打开文件。如果在后续操作中决定更改脚本文件名,则还需要在文件内修改 C#类的名称,以使其与文件名准确匹配,如图 1-4 所示。否则,这将生成无效代码以及编译错误,并在脚本文件与对象绑定时出现问题。

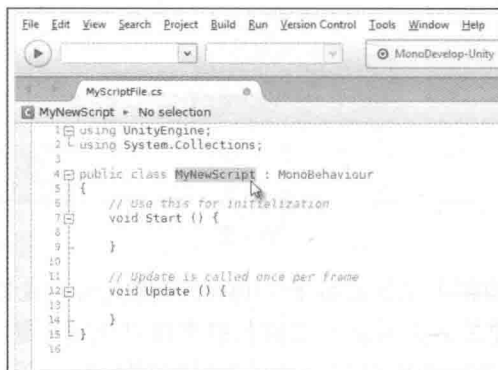


图 1-4



当在 Unity 中编译代码时,需要在 MonoDevelop 中保存脚本文件,即选择应用菜单中 File 命令中的 Save 选项(或者按 Ctrl+S 快捷键),并于随后返回至 Unity Editor 中。当 Unity 窗口再次处于焦点状态时,Unity 可自动检测到文件中的变化,进而对代码进行编译。如果存在错误,则游戏将无法正常运行,对应的错误信息将显示于 Console 窗口中。若编译成功,单击 Editor 工具栏上的 Play 按钮即可。需要注意的是,如果在修改代码后未保存文件,Unity 将使用之前的编译版本运行程序。针对这一原因以及备份要求,建议用户定期保存文件(按 Ctrl+S 快捷键,将结果保存至 MonoDevelop 中)。

1.3 脚本的实例化操作

Unity 中的各个脚本文件定义了一个主类,这类似于设计蓝图,并可对其进行实例化操作。该类可视为相关变量、函数以及事件的集合(稍后将对此进行分析)。默认状态下,脚本文件类似于其他任意一种 Unity 数据资源,例如网格和音频文件。特别地,脚本文件通常处于静止状态,且不执行任何操作,直至添加至某一特定的场景中(作为组件