Unity 实战

(第 2 版)

[美] 约瑟夫·霍金(Joseph Hocking) 著
蔡俊鸿 译

清华大学出版社
北 京

Joseph Hocking
Unity in Action，Second Edition
EISBN: 978-1-61729-496-9

图书在版编目(CIP)数据

Unity 实战：第 2 版 / (美)约瑟夫 • 霍金(Joseph Hocking) 著；蔡俊鸿 译. —北京：清华大学出版社，2019

书名原文：Unity in Action, Second Edition
ISBN 978-7-302-51895-2

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

中国版本图书馆 CIP 数据核字(2018)第 294422 号

责任编辑：王　军　于　平
封面设计：孔祥峰
版式设计：思创景点
责任校对：牛艳敏
责任印制：宋　林

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 装 者：三河市国英印务有限公司
经　　销：全国新华书店
开　　本：170mm×240mm　　**印　　张**：22　　**字　　数**：444 千字
版　　次：2019 年 1 月第 1 版　　**印　　次**：2019 年 1 月第 1 次印刷
定　　价：69.80 元

产品编号：077874-01

推 荐 序 一

关于艺术，但凡有点“私心”的人都想跟它沾上边。有人把它作为茶余饭后的谈资，有人作为业余消遣的娱乐，有人视其为谋生的工具……有人浅尝辄止，有人不求甚解，有人半途折返，有人情之所至……无论你是基于哪一种出发点，都希望自己能够仰高山之巅、充分实现自我。

进入移动游戏时代后，游戏作为第九艺术，不仅给众多玩家带来了电影级别的剧情体验以及美妙的音乐享受，亦带给玩家们现实生活中难以实操的情感体验与交互。游戏作为文化产品所带来的视听享受以及超越电影的交互优势，使得玩家群体以难以置信的速度在扩大，这是游戏最好的时代，但也是游戏开发者面临巨大挑战的时代。

随着玩家群体审美提升，游戏的复杂度、可玩性、渲染效果以及性能面临着巨大的挑战。Unity3D 作为用于移动游戏开发的现代化引擎，凭借其强大的跨平台开发能力，在当前的移动游戏时代中应运而生，具有强大的竞争力。凭借其出色的渲染能力，在游戏中为视觉效果贡献巨大力量，此外，由于其友好的可视化开发环境，又使之成为移动游戏开发的不二选择。

本书的上一版《Unity5 实战　使用 C#和 Unity 开发多平台游戏》已成为行业内认可度颇高的优秀专业书籍。身边很多技术开发者，在进入移动游戏开发前都通过这本书学习 Unity3D 开发的相关知识，他们均赞不绝口。我与译者在多个游戏项目中都有合作，译者本人就是一位非常资深的技术专家，也是使用 Unity3D 进行移动游戏开发的先驱。他对于游戏开发始终抱有巨大的热情，对于开发技术始终精益求精，对于艺术由始至终情之所至。我相信译者这样的技术态度，足以让《Unity 实战(第 2 版)》的译文更加精确无误且更易于理解与推广。现在，您手上阅读的这本书已是第 2 版，我相信本书不仅仅是第 1 版精彩的延续，更是在此基础上的超越。相信译者本人，即是相信自己，希望你在本书中能拾你所想，得你所思，如你所愿。

陈笑雨

——360 游戏艺术副总裁

推 荐 序 二

作为一本以“实战”为名的技术类专业书籍，本书的第 1 版《Unity5 实战 使用C#和 Unity 开发多平台游戏》此前在亚马逊上连续数月居于同类书籍销量之冠，结合身边诸多从业者阅读后的赞誉推荐，已经证明了自身的价值。这里的价值既反映出了原书作者的丰富经验和工程智慧，也体现了译者极强的专业素养。值此再版之际，应译者邀请，为此作序，深感荣幸。

在行业中深植多年，深知追求艺术性或专业性乃至两者合一的难能可贵。因为我们时常会在概念上把工程、设计和管理进行融合，但在实践上又把它们分离，并且将具体的研发方式归结于各种流派，将诸多的不可控因素归结于风格、习惯的区别或者环境使然。以各种限定的模式去适配知识的获取途径与种类，从而降低了知识应有的价值。本书很好地解决了我们作为游戏开发者为何而读书的问题。因为这种实战派的传授方式，相当于一次言传身教，使得我们很容易理解技术。

游戏作为特殊类型的软件，为逻辑性和设计性的结合体，兼具理性和感性的部分，在面向用户层体现得更加清晰。Unity3D 作为主流的商用引擎，功能强大，易用性好，并且具有较完善的工具链和较丰富的扩展，也让本书所传递的知识具有更广泛的受众群体。而随着引擎的升级换代，并不会影响本书的价值，这种经验的传承性，作者和译者在文字中所传递的钻研和分析，结合引擎发展所面临的问题以及解决思路，都能为读者在遇到本书范畴以外的问题时获得启发，此谓举一反三。

与译者相识数载，知其作为一名资深的行业技术专家，有过多年多产的研发履历，从未离开过游戏开发的一线。本书虽为他业余时间的译作，亦是一番呕心沥血，未敢一丝一毫怠慢。因此，诸君读此书，应有所得，或得他山之石，或得其中金玉。

庄　宏

——GameArk 副总裁

译 者 序

Unity 是当今最炙手可热的游戏开发工具之一，它是轻松创建诸如三维视频游戏、建筑可视化、实时三维动画的综合型游戏开发平台，是一个全面整合的专业游戏引擎。它可发布运行在 Windows、Mac、iPhone、Windows Phone 8 和 Android 平台上的游戏，也可以利用插件发布网页游戏。很多著名的游戏，如神庙逃亡、新仙剑、QQ 乐团等，都出自这个平台。

Unity 有以下几项优点：第一，Unity 的部署很简单，还自带一个 IDE：MonoDevelop，只要按下 install，之后的创建新项目、多平台打包等操作均可以在编辑器中直接完成。第二，Unity 形成了一个规模化的插件市场，在此基础上，Unity 具有相当多的中间件，可以大大加快独立开发者和公司的开发进度。第三，Unity 的社区是当前各种游戏开发社区中最活跃的，这点可以从“知乎”上的 Unity 3D 话题的关注人数看出。第四，C#作为脚本可以在编程效率和运行效率之间取得比较好的平衡，使用 C#，今后的微软一系列新技术也很有可能会和 Unity 搭配。

Unity 有大量有价值的学习资源，但这些资源比较零散，需要进行深度挖掘才能找到需要的内容。而本书把初学者需要了解的所有内容都放在一个地方，以清晰、富有逻辑的方式呈现出来，为初学者打开了游戏编程的大门。尤其是本书的 “实践”部分，读者很快就可以开始编写代码——不只是编写书中的示例代码，还编写自己的游戏代码，因为本书并不仅仅简单完成游戏示例所需的功能代码，还经常对代码进行重构，提升可扩展性和复用性。

本书分为三部分，第 I 部分介绍跨平台的游戏开发环境 Unity，演示在 3D 中编写移动示例的步骤，再将移动示例转变为第一人称射击游戏，讲解射线发射和基础 AI，最后导入和创建美术资源。第 II 部分学习如何在 Unity 中创建 2D 益智游戏，接着用平台游戏机制扩展 2D 游戏，介绍 Unity 中最新的 GUI 功能，展示如何在 3D 中创建第三人称移动游戏，阐述如何在游戏中实现交互设备和物品。第III部分讨论如何与互联网通信，如何编写音频功能，如何将不同章节的碎片整合到一个游戏中，最后构建最终应用，发布到多个平台。本书最后还提供了 4 个附录，分别介绍了场景导航、外部工具、Blender 和学习资源。

本书针对的读者群是对 Unity 很陌生的编程老手，以及游戏开发新手。

掌握了本书的内容后，可以关注《Unity 圣典》和《Unity 用户手册》，把在初学阶段忽略的内容进行选择性的补充学习。再进一步，可以关注 Unity 社区、Unity Answers、Unity Wiki 和“知乎”的 Unity 板块，此时，要对 Unity 的各种细节问题、优化、底层原理和新的技术方案进行深入思考和系统学习。

在此要感谢清华大学出版社的编辑，他们为本书的翻译投入了巨大的热情并付出了很多心血。没有他们的帮助和鼓励，本书不可能顺利付梓。

对于这本经典之作，译者本着“诚惶诚恐”的态度，在翻译过程中力求“信、达、雅”，但是鉴于译者水平有限，错误和失误在所难免，如有任何意见和建议，请不吝指正。本书主要章节由蔡俊鸿翻译，参与翻译的还有陈妍、何美英、陈宏波、熊晓磊、管兆昶、潘洪荣、曹汉鸣、高娟妮、王燕、谢李君、李珍珍、王璐、王华健、柳松洋、曹晓松、陈彬、洪妍、刘芸、邱培强、高维杰、张素英、颜灵佳、方峻、顾永湘、孔祥亮。

译　者

第一版的赞誉

“本书简明扼要，示例突出。作为一名新用户，我发现本书是一个无价之宝。”

—Dan Kacenjar Sr.，基石软件

“所有的障碍都消失了，我很快就把游戏从概念变成构建好的软件。”

—Philip Taffet, SOHOsoft LLC

“很快就能让游戏运转起来。”

—Sergio Arbeo, codecantor

“涵盖了有效使用 Unity 的所有关键元素。”

—Shiloh Morris，南内华达州水务局

“推荐给所有使用 Unity 开始游戏编程的人。”

—Alex Lucas，独立承包商

“使用干净的代码教学并说明了如何修改代码，以获得更有趣的结果。”

—亚马逊读者

第一版序

我在 1982 年就开始进行游戏编程。那时候很困难，因为没有互联网。资源只有少数糟糕的书籍和杂志，里面的代码片段虽然吸引人，却很混乱，而且根本就没有游戏引擎！编写游戏代码是一场艰巨的战斗。

非常羡慕今天的读者可以阅读《Unity 5 实战 使用 C#和 Unity 开发多平台游戏》，Unity 引擎为许多人打开了游戏编程的大门。Unity 达到了一个很好的平衡：一方面，Unity 已经成为一个强大、专业的游戏引擎；另一方面，Unity 仍然是初学者负担得起的、易于接近的游戏引擎。

“易于接近”指的是通过恰当的引导可以很快上手。有一次，我参加了一个由魔术师运营的马戏团。魔术师对我非常友善，帮助我成为一名出色的表演者。魔术师说：“当你站在舞台上时，需要许下承诺：‘我不会浪费你们的时间’”。

我最喜欢本书的 “实践”部分。作者没有浪费读者的宝贵时间，读者很快就开始编写代码——不是编写无意义的代码，而是可以理解和构建的有趣代码，因为他知道，读者不只是想阅读本书，不只是想编写书中的示例——还想编写自己的游戏。

在本书的指导下，读者上手的速度远超自己的期望。请随着 Joseph 的步伐学习，在准备好之后，不要羞于抛弃他的学习路线，去规划自己的学习路线。跳到最感兴趣的部分——尝试实验，请大胆而勇敢地进行尝试！如果你迷失了方向，还可以返回到《Unity 实战(第 2 版)》中。

不必在此序中浪费时间——游戏在等着你开发！在日历上标记一下今天的日期，因为从今天开始要发生翻天覆地的变化。永远记住今天是你开始制作游戏的第一天。

Jesse Schell
Schell Games 的 CEO
The Art of Game Design 一书的作者

前　言

我从事游戏编写工作很长时间了，但最近才开始使用 Unity。当我开始开发游戏时，Unity 尚未出现，它的第 1 版在 2005 年发布。从一开始，它就承诺要作为游戏开发工具，但直到发布了几个版本，它也没有实现诺言。iOS 和 Android(统称为“移动”平台)等平台是后来才出现的，这些平台在很大程度上促成了 Unity 日益突出的地位。

最初，我将 Unity 视为一个有趣的开发工具，我关注它，但并不真正使用它。那段时间，我在为桌面计算机、网站编写游戏，为各种客户端开发项目。我使用过 Blitz3D 和 Flash 等工具，它们很适合编程，但有诸多限制。随着这些工具开始衰落，我就一直在寻找开发游戏的更好方式。

我从 Unity 3 开始体验，后来在 Synapse Games 进行开发时就完全转向了 Unity。最初是为 Synapse 开发网页游戏，最终全面转向移动游戏。那时我们进行游戏开发的整个生命周期，因为 Unity 允许从同一个代码库部署到网页和移动平台！

我一直认为分享知识很重要，讲授游戏开发课程也有好几年了。这么做的主要原因是很多导师和老师的表率作用。我在多所学校授课，一直想写一本关于游戏开发的书籍。

本书的许多方面都是我第一次学习 Unity 时所期望包含的教学内容。Unity 的众多优点之一是有大量有价值的学习资源，但这些资源比较零散(诸如脚本参考或独立的教程)，需要读者进行深度挖掘才能找到需要的内容。最好有一本书，把需要了解的所有内容都放在一个地方，以清晰、合乎逻辑的方式呈现出来，这就是本书的目标。本书针对的读者群是对 Unity 很陌生的编程老手，以及游戏开发新手。项目的选择反映了我快速连续地完成各种自由职业项目而获得技能和信心的经验。

学习使用 Unity 开发游戏是一场激动人心的冒险。对我来说，学习如何开发游戏意味着要忍受很多麻烦。但对读者而言，拥有了本书则意味着拥有了一份清晰简明的学习资源。

致　　谢

我要感谢 Manning 出版社给了我撰写本书的机会。与我共事的编辑，包括 Robin de Jongh、Dan Maharry，帮助我完成了这个任务，本书也因为他们的反馈更加出色。Candace West 担任第 2 版的主编。我真诚地感谢在开发和出版本书时与我一起共事的人。

我的写作受益于每一位审稿人的审查。感谢 Alex Lucas、Craig Hoffman、Dan Kacenjar、Joshua Frederick、Luca Campobasso、Mark Elston、Philip Taffet、Rene van den Berg、Sergio Arbeo Rodriguez、Shiloh Morris、Victor M. Perez、Christopher Haupt、Claudio Caseiro、David Torribia Inigo、Dean Tsaltas、Eric Williams、Nickie Buckner、Robin Dewson、Sergey Evsikov 和 Tanya Wilke。

特别感谢技术开发编辑 Scott Chaussee 和技术校对员 Christopher Haupt、Rene van den Berg 和 Shiloh Morris 对本书第 2 版进行了审核。我还要感谢 Jesse Schell 为我的书作序。

接下来，我要感谢给予我丰富 Unity 经验的相关人员。首先要感谢的是 Unity Technologies(制作 Unity 游戏引擎的公司)。我很感激 gamedev.stackexchange.com 社区。我几乎每天都会访问那个 QA 站点，向其他人学习并回答问题。促使我使用 Unity 的最大动力来自 Alex Reeve，我在 Synapse Games 的老板。同样，我从同事那里学到了一些技巧和技术，它们都展现在我编写的代码中。

最后，我要感谢我的妻子 Virginia，感谢她在我写这本书时给予我的支持。直到我开始写这本书，才真正明白这个项目在我的生活中占据了多大的位置，对周围的人产生了多大的影响。非常感谢她的爱和鼓励。

关 于 本 书

本书介绍如何使用 Unity 编写游戏。有经验的程序员可以把它当成 Unity 的入门书籍。本书的目标十分明确：带领有一些编程经验但没有 Unity 经验的读者使用 Unity 开发游戏。

讲授开发最好的方式是完成示例项目，学生通过制作示例来学习，这正是本书采用的方式。本书的各个主题展现为构建游戏示例的步骤，当浏览本书时，鼓励读者在 Unity 中构建这些游戏。每几章挑选不同的项目来讲解，而不是整本书只开发一个项目。其他有些书籍采用“一个完整项目”的方法讲解，不足之处是如果对前面的章节不感兴趣，就很难跳到中间的章节。

本书比大多数 Unity 书籍(特别是入门书籍)有更严格的编程内容。如果不知道如何编写计算机程序，最好先使用 Codecademy 之类的资源学习，在学会如何编写程序之后再回到本书。

不要担心具体的编程语言，本书大量使用了 C#，也可以使用其他语言的技能。本书的第 I 部分会花时间介绍新的概念，会小心谨慎、一步一步地在 Unity 中开发第一款游戏，但剩下的章节将更快速地推进，让读者了解多个游戏类型。本书最后会描述部署到各种平台(如 Web 和移动平台)，但本书的主旨不会提及最终的部署目标，因为 Unity 与平台无关。

至于游戏开发的其他方面，广泛覆盖的美术学科会稀释本书涵盖的 Unity 知识，加大 Unity 外部软件(例如，所使用的动画软件)的比重。关于美术任务的讨论将仅限于 Unity 或所有游戏开发者都应该知道的方面。

学习路线图

第 1 章　介绍跨平台的游戏开发环境——Unity。学习 Unity 中任何对象所基于的组件系统原理，介绍如何编写和运行基本脚本。

第 2 章　演示在 3D 中编写移动示例的步骤，涵盖鼠标和键盘输入等主题。全面解释 3D 位置和旋转的定义和管理。

第 3 章　将移动示例转变为第一人称射击游戏，讲解射线发射和基础 AI。射线发射(向场景发射一条线，并观察相交情况)是所有类型游戏中很有用的操作。

第 4 章　涵盖了美术资源的导入和创建。本章不关注代码，因为每个项目都需要(基本)模型和贴图。

第 5 章　学习如何在 Unity 中创建 2D 益智游戏。尽管 Unity 开始时仅包括 3D 图形，但现在也能很好地支持 2D 图形。

第 6 章　用平台游戏机制扩展了 2D 游戏。特别是，为玩家实现控件、物理和动画。

第 7 章　介绍 Unity 中最新的 GUI 功能。每个游戏都需要 UI，而最新版本的 Unity 为创建 UI 提供了一个改进的系统。

第 8 章　展示如何在 3D 中创建另一种移动游戏，此时从第三人称的视角看到场景。实现第三人称控制将展示一系列 3D 数学操作，学习如何使用带动画的角色。

第 9 章　浏览如何在游戏中实现交互设备和物品。玩家有很多方式操作这些设备，包括直接触摸它们，接触游戏中的触发器，或者是按下控制器的某个按钮。

第 10 章　涵盖了如何与互联网通信。学习如何使用标准互联网技术来发送和接收消息。例如 HTTP 请求，从服务器获取 XML 数据。

第 11 章　介绍如何编写音频功能。Unity 对短音效和长音轨提供了很好的支持，这两种类型的音频对于所有视频游戏都很重要。

第 12 章　将不同章节的碎片整合到一个游戏中。此外，还学习如何编写指向-单击的控件，以及如何保存玩家的进度。

第 13 章　构建最终应用，发布到多个平台，例如桌面、网页和移动，甚至 VR。Unity 与平台无关，允许为每个主流的游戏平台创建游戏。

本书最后还提供了 4 个附录，分别介绍场景导航、外部工具、Blender 和学习资源。

代码约定、要求和下载

本书的所有源代码，不管是代码清单或是片段，都使用等宽字体，以便与周围的文本区别开来。在大多数代码清单中，代码都通过注释指出关键概念，而编号有时用于在文本中提供关于代码的额外信息。代码是经过格式化的，通过合理地增加换行和缩进，以适应本书可用的页面空间。

唯一需要的软件是 Unity，本书使用的是 Unity 2017.1，它是编写本书时的最新版本。某些章节偶尔讨论其他软件，但那些仅作为可选的额外部分，而非核心的学习内容。

警告：

Unity 项目会记住它们在哪个版本的 Unity 中创建，如果尝试在不同版本的 Unity 中打开它们，会显示警告。如果打开本书下载的示例时看到警告，请单击 Continue 并忽略它。

本书的代码清单通常展示了在已有的代码文件中应该添加或修改的内容，除非是首次出现的代码文件，否则不要用后来的清单覆盖整个文件。尽管可以下载书中引用的完整示例项目，但最好输入代码清单中的内容，并观察所引用的示例。可以从 Manning 出版社的网站(www.manning.com/books/unity-in-action-second-edition)和 GitHub(https://github.com/ jhocking/uia-2e)下载示例。也可扫描封底二维码获取本书示例文件。

本 书 论 坛

购买本书， 就可以免费访问由 Manning 出版社运营的私人网页论坛，在该论坛上可以发表关于本书的评论，提问技术问题，并接受来自作者和其他用户的帮助。为了访问论坛，可以进入 https://forums.manning.com/forums/unity-in-action-second-edition，在 https://forums.manning.com/forums/about 中可以了解 Manning 论坛和行为准则。

Manning 对读者的承诺是提供一个场所，让读者之间以及读者与作者之间进行有意义的对话。不承诺作者具体会参与多少分享，作者对论坛的贡献是自愿的(而且是无偿的)。建议试着向作者提出一些具有挑战性的问题，以免令作者兴味索然！只要本书还在印刷，该论坛和之前讨论的档案文件都可以在出版商的网站上找到。

作 者 简 介

Joseph Hocking 是一名软件工程师，专门研究交互式媒体开发。他目前为 InContext Solutions 公司工作，在为 Synapse Games 公司工作期间撰写了本书的第 1 版《Unity5 实战　使用 C#和 Unity 开发多平台游戏》。他还在伊利诺伊大学芝加哥分校、芝加哥艺术学院和哥伦比亚大学芝加哥分校授课。可以访问他的网站 www.newarteest.com。

封面插图声明

本书封面上的插图标题是“Habit of the Master of Ceremonies of the Grand Signior”。Grand Signior 是土耳其帝国苏丹的另一个名称。插图取自 Thomas Jefferys 的 *A Collection of the Dresses of Different Nations, Ancient and Modern (4 volumes)*, 这些书在 1757—1772 年于伦敦出版。标题页表明了，这些是手工上色的铜版雕刻，使用阿拉伯树胶增加厚度。Thomas Jefferys(1719—1771)被称为“国王乔治三世的地理学家”。他是一位英国制图师，是当时顶尖的地图供应商，他为政府和其他官方机构雕刻和印刷地图，制作了大量的商业地图和地图集，尤其是北美地图集。作为一名地图绘制师，他的工作激起了人们对他所调查地区的当地服饰习俗的兴趣，这些服饰在这套四卷书集中得到了很好的展示。

在 18 世纪末，兴起了一股风潮，人们开始向往远方，并享受旅行的乐趣。像 Jeffery 画作这样的收藏品是很流行的，为旅行者和向往旅行、但是没能出发的人们介绍异域居民是什么样子。Jeffery 画作藏品的多样性生动描绘了两百多年前各个国度的独特性。从那以后，着装上就发生了变化，而当时各国家、各地区丰富的多样性也渐渐趋同。现在很难区分来自不同大陆的人们。如果从乐观的角度看，我们是把文化和视觉上的多样性作为代价，换来了更丰富的私人生活，或者变化更大、更有趣的知识和技术生活。

在如今这个计算机图书封面大同小异的时代，Manning 出版社以两个世纪前丰富多样的地域生活为基础设计图书封面，令 Jeffery 的画作重新焕发生机，颂扬计算机行业的革新性和首创精神。

目　录

第 I 部分　起步

第Ⅰ部分

起　　步

是时候迈出使用 Unity 的第一步了。如果你一点也不了解 Unity，那也没关系！本部分将首先解释 Unity 是什么，以及使用它编写游戏程序的一些基本原理。接着将讲解一个在 Unity 中开发简单游戏的指南。该指南将讲述一系列游戏开发技术及其大概的工作流程。

下面开始学习第 1 章！

第 1 章

初识 Unity

本章涵盖：

- 是什么使得 Unity 成为一个极佳选择
- 操作 Unity 编辑器
- 在 Unity 中编程
- 比较 C#和 JavaScript

如果你像我一样，曾经有很长一段时间梦想着开发一款视频游戏。但是从玩游戏到实际开发游戏是一个很大的跳跃。这些年出现了很多游戏开发工具，而我们准备讨论的正是这些工具中最现代、最强大的一个。Unity 是一个专业的游戏引擎，它用于创建针对不同平台的视频游戏。它不仅是一个被成千上万经验丰富的开发者使用的开发工具，也是当代游戏开发新手比较容易上手的现代开发工具。直到现在，游戏开发新手在制作游戏时，仍然面临很多巨大的障碍，但 Unity 的出现让这些技能变得更容易学习。

你正在阅读本书，可能是你对计算机技术比较好奇，并且使用其他工具开发过游戏或者构建过其他类型的软件，例如桌面应用或网站。创建一个视频游戏与编写其他类型的软件没有什么根本区别，但也有一定的区别。例如，视频游戏比大多数网站有更多的交互，而且会包含很多不同类型的代码，但制作所用的技术和方法很相似。如果你

已经克服了学习游戏开发道路上的第一道障碍，已经学习了编写软件程序的基本原理，那么下一步就是选择一些游戏开发工具并把编程知识转化到真正的游戏中去。Unity 是开发游戏工作环境的一个极佳选择。

关于术语的提示

这是一本关于 Unity 编程的书，因此它针对的主要是编程人员。尽管很多资源讨论游戏开发和 Unity 中的其他方面，但本书还是把编程放在了首位。

顺便提一下，“开发者”在游戏开发上下文中有着不同的含义：一般情况下，“开发者”和 Web 开发的编程人员是同义词。但在游戏开发中，“开发者”指的是做游戏开发工作的任何人，除了刚刚提及的编程人员，还有其他类型的游戏开发者，如艺术家和设计师。但本书将关注编程部分。

在开始介绍 Unity 前，请先到网站 www.unity3d.com 下载软件。本书使用的是 Unity 2017.1 版本，此版本是编写本书时的最新版本。这个 URL 是 Unity 最初专注于 3D 游戏遗留下来的；它对 3D 游戏的支持依然强大，也能很好地服务于 2D 游戏。本书介绍 2D 和 3D 游戏。实际上，即使在 3D 游戏中演示，许多主题(保存数据、播放音频等)都适用于这两种情况。同时，尽管还有一些付费版本，但基础版本还是完全免费的。本书的所有内容都基于免费版本，不需要 Unity 付费版本。这些版本之间的区别在于商业声明条款。

1.1　为什么 Unity 如此优秀

Unity 是一个专业级的高质量游戏引擎，它用于创建针对多种平台的视频游戏。这个答案相当直接地回答了“什么是 Unity？”这样的问题。然而，这个答案具体意味着什么，为什么 Unity 如此优秀？

1.1.1　Unity 的优势

游戏引擎都提供了充足的特性，这些特性在很多不同的游戏中都有用。因此通过使用引擎，只要加入自定义的美术资源并增加自己游戏玩法的代码，就可以轻松获得那些特性，从而实现一个游戏。Unity有物理模拟、法线贴图、屏幕空间环境光遮蔽(Screen Space Ambient Occlusion，SSAO)、动态阴影等特性。很多游戏引擎以有诸多特性而自豪，但 Unity 比起其他尖端的游戏开发工具有两个主要优势：提供了非常高效的可视化工作流和多维度的跨平台支持。

可视化工作流是相当独特的设计，它和其他大多数游戏开发环境不同。其他游戏开发工具通常混杂了必定引发争议的不相关部分、需要自己设置集成开发环境(Integrated Development Environment，IDE)的编程类库、构建链和其他陈旧的设计等，

而 Unity 中的开发工作流是通过精心设计的可视化编辑器定位的。这些编辑器用于布局游戏中的场景，将美术资源绑定在一起并对可交互对象进行编码。这些编辑器的美妙之处在于它允许快捷高效地构建专业、高质量的游戏。当需要在视频游戏中使用大量的新技术时，它将提供难以置信的高效工具。

注意　其他大多数带有集中式可视化编辑器的游戏开发工具通常被限制得很死，它们支持的脚本也不灵活，但 Unity 没有这个缺点。尽管 Unity 基本上是通过可视化编辑器创建所有的内容，但这个核心接口还包括了一系列链接到可运行于 Unity 游戏引擎上的自定义代码的项目。通过为项目提供核心接口这种方式很像为诸如 Visual Studio 或 Eclipse 的 IDE 在项目设置中链接类一样。有经验的编程人员应该考虑这个开发环境，不要误以为它只能通过鼠标单击，从而限制了 Unity 的编程能力。

可视化编辑器对于快速迭代以及在原型制作和测试游戏的周期中打磨游戏都非常有益。可以在编辑器中调整物体，甚至是在游戏运行时移动物体。另外，Unity 允许通过编写脚本来自定义编辑器，以在界面上增加一些新特性或菜单。

除了编辑器非凡的生产力优势，另一个主要长处在于 Unity 的工具集提供了高度跨平台的支持。不仅是在部署目标方面跨平台(能部署到 PC、Web、移动设备或游戏主机)，还包括开发工具跨平台(能在 Windows 或 Mac OS 上开发游戏)。这个平台潜力很大，因为 Unity 最开始作为 Mac 独享的软件，后来才移植到 Windows。Unity 的第一个版本在 2005 年发布，现在 Unity 已经更新到了第五个主要版本。最初，Unity 仅支持开发和部署在 Mac 上，但数月后，Unity 已经更新到也能工作在 Windows 上。后续版本的 Unity 中添加了更多部署平台，例如 2006 年添加了可跨平台的 Web 播放器，2008 年添加了对 iPhone 的支持，2010 年添加了支持 Android，甚至支持更多游戏主机，比如 Xbox 和 PlayStation。最近它们已经添加了到 WebGL 的部署，WebGL 是一个用于 Web 浏览器 3D 图形的新框架，甚至支持 VR 平台，如 Oculus Rift 和 Vive。一些游戏引擎也支持和 Unity 一样多的部署目标，但是没有一个能像 Unity 那样让部署到多平台的工作变得如此简单。

除了这些主要的优点外，第三条微妙的优点是 Unity 使用模块化组件系统构造游戏对象。在一个组件系统中，“组件”是一个混合搭配功能的包，对象由一系列组件构建，而不是由层级严格的类构建。换句话说，组件系统是和面向对象编程不同的方法(它更灵活)，游戏对象是通过组合的方式而不是继承的方式构建的。图 1-1 演示了继承和组件系统的对比。

在组件系统中，物体存在于一个扁平的层级结构中，不同的物体有不同的组件集合，而不像继承结构，不同物体处于树的完全不同的分支中。这种设计加快原型的开发，因为当物体改变时，可以快速混合搭配不同组件而不必重构继承链。

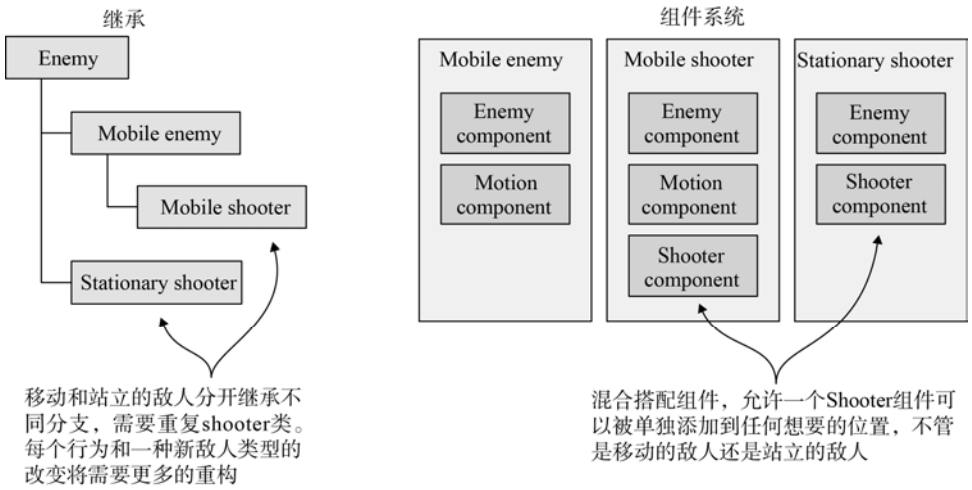

图 1-1　继承与组件系统

当没有组件系统时，可以编写代码实现自定义的组件系统，但是 Unity 已经有一个健壮的组件系统，这个系统甚至与可视化编辑器无缝地集成在一起。不仅能通过代码维护组件，还能使用可视化编辑器附加和移除组件。另外，可以通过组合构建对象，也可以在代码中选择使用继承，包括所有基于继承的最佳设计模式。

1.1.2　要意识到的缺点

Unity 有很多优点，这让它成为开发游戏的好选择，我们也极力推荐它，但如果不提它的缺点就有点失职了。实际上，混合使用可视化编辑器和复杂的代码，尽管和 Unity 的组件系统能高效地组合在一起，但它也不简单。在复杂场景中，你会搞不清楚场景中的哪个物体附加了指定组件。Unity 提供了搜索功能用于查找附加的脚本，但该搜索是很粗糙的；有时你还会遇到一些情况，为了找出脚本链接，需要手动检查场景中的物体。这些情况虽然不会经常出现，但是一旦出现，会令人沮丧。

第二个缺点会让有经验的编程人员吃惊且失望，即 Unity 不支持链接外部代码库。需要的库必须手动复制到每个使用它们的项目中，而不是引用一个中心共享的位置。对类库缺少中心位置，在多个项目中共享类库就会变得笨拙。这个缺点能通过巧妙使用版本控制系统来绕开，但 Unity 不支持外部代码库的使用。

注意　难以使用版本控制系统(例如 Subversion、Git 和 Mercurial)在过去是 Unity 的一个致命弱点，但现在 Unity 的一些版本已经可以使用它们。一些过时的资源提到，Unity 不能使用版本控制，但是一些新资源会说明，项目中的哪些文件和文件夹需要放在资源库里，哪些不用。为使用版本控制系统，请阅读 Unity 的文档(http://mng.bz/BbhD)或查看 GitHub(http://mng.bz/g7nl)维护的.gitignore 文件。

第三个缺点是必须使用预设(prefab)。预设是 Unity 特有的一个概念，将在第 3 章中解释，现在只需要知道预设是可视化定义交互对象的一种灵活方式。预设很强大，只属于 Unity(它和 Unity 的组件系统绑定在一起)，但预设的编辑流程却十分粗糙。由于预设极其有用，且作为 Unity 的主要部分进行工作，因此希望在未来的 Unity 版本中能改善对预设进行编辑的工作流。

1.1.3　使用 Unity 构建的游戏示例

前面已经介绍了 Unity 的优缺点，但仍然需要确信 Unity 中的开发工具是最好的选择。请访问 Unity 图片库 http://unity3d.com/showcase/gallery，看一看使用 Unity 开发的游戏和仿真程序。这个列表包含成百上千个游戏和仿真程序，并且在不断更新中。本节仅介绍一些类型和部署平台的游戏示例。

1. 桌面(Windows、Mac、Linux)

Unity 编辑器运行在同一个平台上，并通常部署到诸如 Windows 或 Mac 这样比较常见的目标平台上。下面给出一些不同类型的桌面游戏示例：

- Guns of Icarus Alliance(见图 1-2)，一个由 Muse Games 开发的第一人称射击游戏。

图 1-2　Guns of Icarus Alliance 游戏

- Gone Home(见图 1-3)，一个由 Fullbright Company 开发的冒险游戏。

图 1-3　Gone Home 游戏

2. 移动平台(iOS 和 Android)

Unity 能将游戏部署到移动平台上，比如 iOS(iPhone 和 iPad)和 Android(手机和平板电脑)。下面是一些不同类型的移动平台的游戏：

- Lara Croft GO(见图 1-4)，由 Square Enix 开发的 3D 拼图游戏。

图 1-4　Lara Croft GO 游戏

- INKS(见图 1-5)，由 State of Play 开发的 2D 拼图游戏。

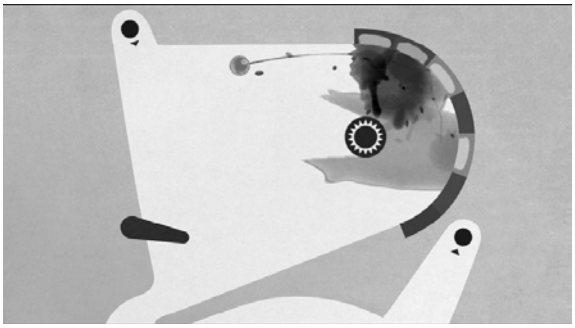

图 1-5　INKS 游戏

- Tyrant Unleashed(见图 1-6)，由 Synapse Games 开发的收集类卡牌游戏。

图 1-6　Tyrant Unleashed 游戏

3. 主机(PLAYSTATION，XBOX，SWITCH)

Unity 能将游戏部署到游戏主机，但开发者依然需要从 Sony、Microsoft 或 Nintendo 获得许可。因为有这种需求和 Unity 容易跨平台部署的特性，主机游戏通常在桌面计算机上也可以看到。以下是一些不同类型的主机游戏的示例：

- Yooka-Laylee(见图1-7)，由Playtonic Games开发的3D Platformer游戏。

图 1-7　Yooka-Laylee 游戏

- Shadow Tactics(见图 1-8)，由 Mimimi Productions 开发的隐形游戏。

图 1-8　Shadow Tactics 游戏

从这些示例中可以看到，Unity 的强大功能肯定可以应用到商业品质的游戏中。虽然 Unity 超越其他游戏开发工具的优势明显，但新手可能会误解编程在开发流程中的存在。Unity 通常表现得就像是一个简单的特性列表，它不需要编程，其实这是错误的观点，它并没有告诉人们制作一款商业游戏需要什么。让大家误以为可以通过单击，使用现有的组件，甚至不需要编程人员就能制作一个精良的原型。严格来讲，将一个有趣的原型变成可发布的精品游戏，是必须要编程的。

1.2　如何使用 Unity

上一节讨论了很多通过 Unity 的可视化编辑器提高生产率的问题，因此下面介绍 Unity 的界面以及如何操作它。如果你尚未下载 Unity，请从 www.unity3d.com 下载程序，并在计算机上安装(在安装程序中请确认选中了 Example Project)。在安装后，运行 Unity，开始浏览界面。

为了通过示例了解该界面，请打开所包含的示例项目。刚安装好的 Unity 会自动打开示例项目，也可以选择 File | Open Project 来手动打开它。示例项目安装在共享的用户目录中，例如，在 Windows 上是在 C:\Users\Public\Documents\UnityProjects\中；在 Mac OS 上是在 Users/Shared/Unity/中。还可能需要打开示例场景，双击 Car 场景文件(图 1-9 中高亮显示的文件，场景文件图标是 Unity 的立方体)，这个文件可以在编辑器底部的文件浏览器的 SampleScenes/Scenes/中找到。图 1-9 显示了该界面。

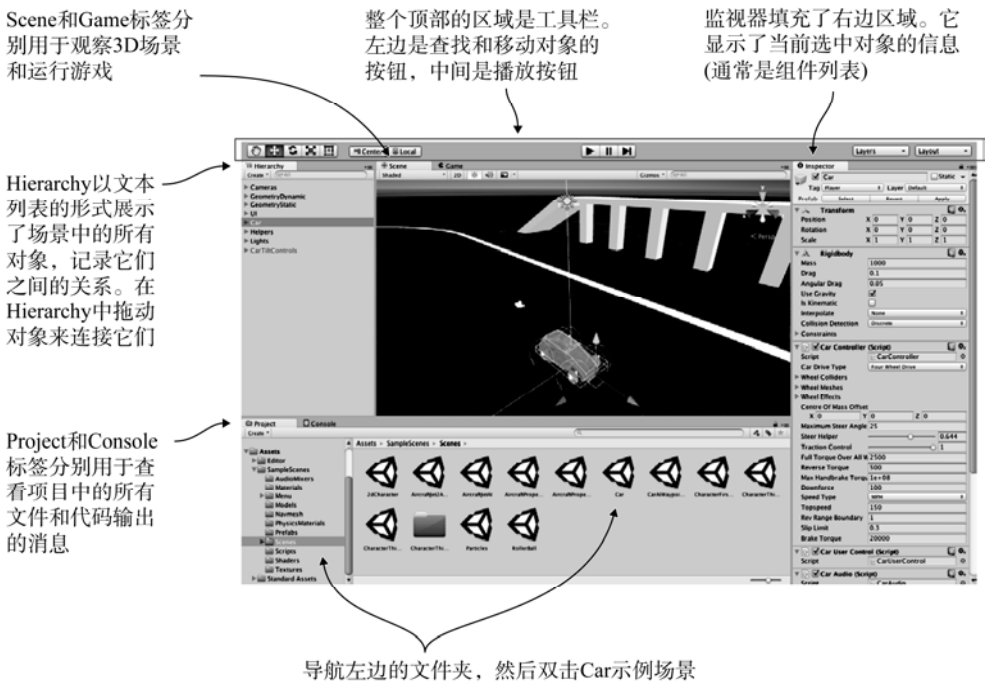

图 1-9　Unity 中的部分界面

Unity 中的界面分为不同的部分：Scene 标签、Game 标签、工具栏、Hierarchy 标签、Inspector 标签、Projector 标签和 Console 标签。每个部分都有不同的用途，它们在构建游戏的生命周期中都很重要：

- 可以在 Project 标签中浏览所有文件。
- 使用 Scene 标签，可以在浏览 3D 场景的同时将对象放置进去。

- 工具栏中的控件用于处理场景。
- 可以在 Hierarchy 标签中拖放对象的关系。
- Inspector 列出了所选对象的信息，包括链接的代码。
- 可以在 Game 视图中测试游戏，并在 Console 标签中查看错误输出。

这是 Unity 中的默认布局，所有不同的视图都有标签，而且可以移动它们或修改它们的尺寸，将其停靠在屏幕中不同的位置。后面将介绍如何自定义布局，不过现在默认布局是理解所有视图用途的最佳方式。

1.2.1　Scene 视图、Game 视图和工具栏

界面中最突出的部分是中间的 Scene 视图。这是观察游戏世界和移动对象的场所。网格对象也会出现在场景中(后面给出它的定义)。也能在场景中看到其他对象，它们由不同的图标和彩色线条表示：摄像机、灯光、声源、碰撞区域等。注意在这个视图看到的东西和运行游戏时看到的不一样。可以在 Scene 视图中到处浏览，而不会受到 Game 视图的约束。

定义　网格对象是 3D 空间中的可视对象。3D 中的可视对象是由很多连接的线和图形构成的，因此世界是由网格构成的。

Game 视图不是屏幕中独立的部分，而是另一个标签，它位于 Scene 视图的右侧(在视图左上角可以找到该标签)。在界面上的有些地方可能有多个这样的标签，如果单击一个不同的标签，视图会被新激活的标签替换。当游戏运行时，这个视图会变成 Game 视图。在每次运行游戏时不需要手动切换标签，因为当游戏启动时，视图会自动切换为 Game 视图。

提示　运行游戏时，可以切换回 Scene 视图，这样就能在运行的场景中检查对象。运行游戏时，这个功能非常适合于查看什么对象在执行什么操作，它是一个很有用的调试工具，而这正是大多数引擎所不具备的。

所谓运行游戏，就是简单地单击 Scene 视图上方的 Play 按钮。界面的顶部是工具栏，Play 按钮正位于中间。图 1-10 为了展示顶部的工具栏，把整个编辑器界面分开了，Scene/Game 标签位于工具栏的下面。

工具栏左边的按钮用于场景导航和变换对象——如何浏览场景和移动对象。建议花一些时间练习浏览场景和移动对象，因为它们是 Unity 可视化编辑器中最重要的两个操作(因为它们非常重要，所以后面有两小节专门介绍它们)。工具栏的右侧是布局和图层的下拉菜单。如前所述，Unity 界面的布局很灵活，Layouts 菜单允许在布局之间来回切换。Layers 菜单具有高级功能，现在可以先忽略它(后续章节中将介绍)。

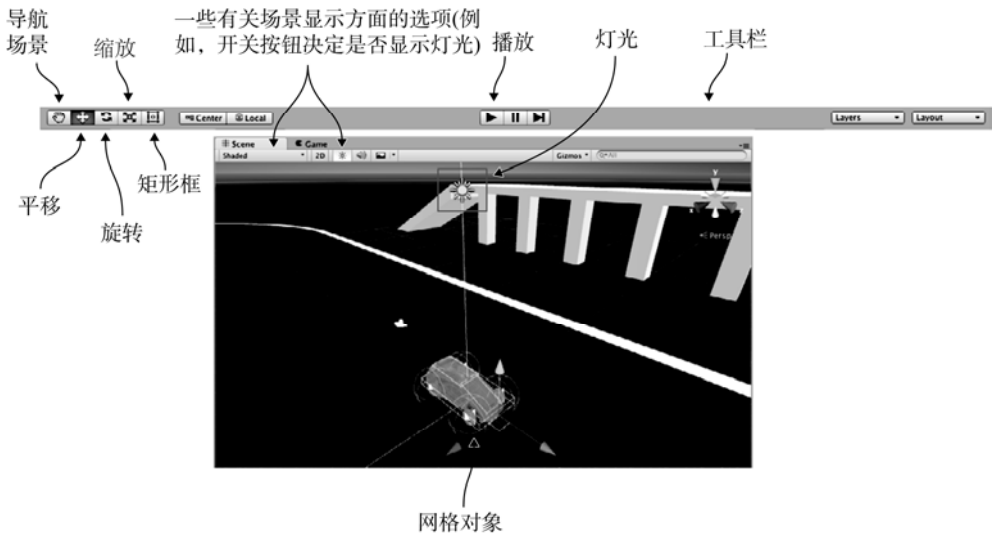

图 1-10　编辑器屏幕截图用于显示工具栏、场景和游戏

1.2.2　使用鼠标和键盘

场景导航主要使用鼠标，通过一些修饰键来修改当前鼠标的行为。这三个主要的导航菜单项是移动(Move)、盘旋(Orbit)和变焦(Zoom)。具体的鼠标移动操作在本书的附录 A 中给出。这三种不同的移动是指按住一些 Alt(在 Mac 上是 Option)和 Ctrl 的组合键时的单击和拖动操作。花几分钟在场景中移动，了解移动、盘旋和变焦的作用。

提示　虽然 Unity 能使用带一个或两个按钮的鼠标，但强烈建议使用带三个按钮的鼠标(带三个按钮的鼠标在 Mac OS X 中也工作得很好)。

通过三个主要的菜单项可以完成对象的变换，而三个场景导航操作也类似于三个变换操作：平移(Translate)、旋转(Rotate)和缩放(Scale)。图 1-11 演示了如何变换一个立方体。

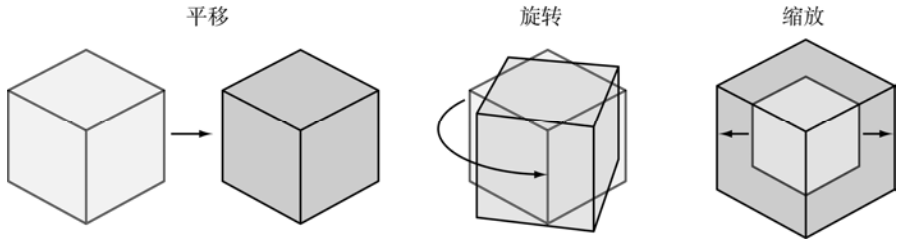

图 1-11　应用三种变换：平移、旋转和缩放(较浅的线图是对象在变换前的状态)

选中场景中的一个对象后，就能移动(数学上精确的技术术语是平移)、旋转或缩放它。在场景导航菜单项中，Move 是平移摄像机，Orbit 是旋转摄像机，Zoom 是缩

放摄像机。除了使用工具栏的按钮之外，按下键盘的 W、E 或 R 键还能切换这些功能。当激活一个变换操作时，会有一系列彩色箭头或圆圈围绕着场景中的对象，这是 Transform 的 gizmo，可以单击或拖动 gizmo 来进行变换。

在变换按钮旁边还有第四个工具，即 Rect 工具，它用于 2D 图形。该工具组合了移动、旋转和缩放功能。这些操作在 3D 中是分开的，但在 2D 中是组合在一起的，因为少了一个需要考虑的维度。Unity 有许多其他键盘快捷键来加速不同的任务。可以参考附录 A 学习它们。下面介绍界面部分剩下的内容！

1.2.3　Hierarchy 视图和 Inspector 面板

在屏幕边缘，左边是 Hierarchy 标签，右边是 Inspector 标签(见图 1-12)。Hierarchy 是一个列表视图，列出了场景中每个对象的名称，名字是否嵌套取决于它们在场景中链接的层次。基本上，选择对象的方式是通过名称，而不是在场景中逐个查找。Hierarchy 将对象连成一组，看起来它们就像文件夹，并且允许一起移动整个组。

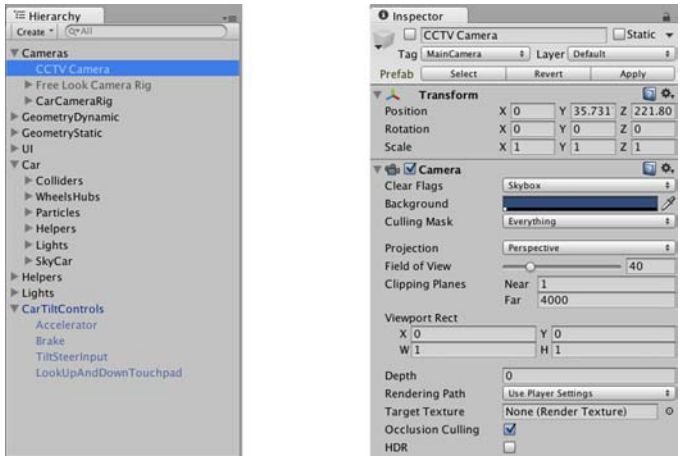

图 1-12　编辑器屏幕截图，用于展示 Hierarchy 标签和 Inspector 标签

Inspector 显示当前所选对象的信息。选择某个对象后，该对象的信息便会填充 Inspector。所显示的信息大部分是组件列表，甚至可以从对象上添加或移除组件。所有游戏对象至少有一个组件，即 Transform，所以在 Inspector 中至少可以看到位置和旋转的信息。很多时候，对象会在 Inspector 中列出很多组件，包括它关联的脚本。

1.2.4　Project 和 Console 标签

屏幕底部是 Project 和 Console 标签(见图 1-13)。与 Scene 和 Game 视图一样，这两个标签不是屏幕中两个独立的部分，可以通过标签在它们之间切换。Project 显示项目中所有的资源(美术、代码等)。具体而言，视图左边是项目中目录的列表；当选择

一个目录时，视图右边会显示该目录中独有的文件。Project 中列出的目录类似于 Hierarchy 中的列表视图，但 Hierarchy 只显示场景中的对象，Project 则显示不包含在特定场景中的文件(包括场景文件——当保存场景时，它会显示在 Project 中)。

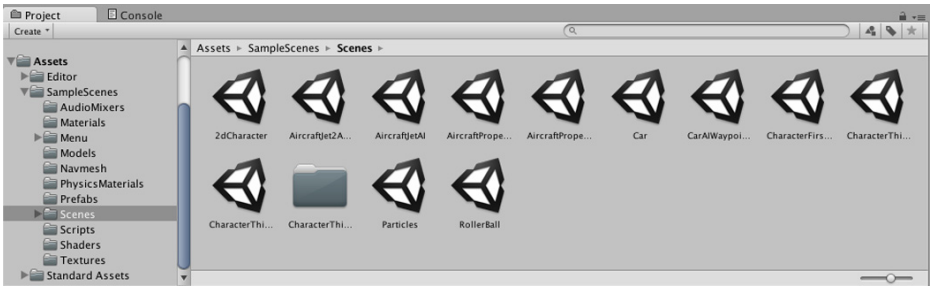

图 1-13　编辑器屏幕截图，用于展示 Project 和 Console 标签

提示　Project 视图镜像了磁盘上的 Assets 目录，但通常不应该直接从 Assets 文件夹中移动或删除文件。如果在 Project 视图中执行这些操作，Unity 会与那些文件夹保持同步。

代码将消息显示在 Console 中。这种消息是专门输出来调试程序的，但是当编写的脚本出现问题时，Unity 也会在 Console 中发出错误消息。

1.3　开始使用 Unity 编程

现在观察 Unity 中编程工作的流程。虽然可以在可视化编辑器中布局美术资源，但需要编写代码来控制它们并让游戏变得可以交互。Unity 支持一些编程语言，特别是 JavaScript 和 C#。它们各有优缺点，本书建议使用 C#。

为什么选择 C#而不是 JavaScript?

本书所有的代码都使用 C#编写，因为相对于 JavaScript，C#有很多的优点，缺点较少，特别是对于专业开发者而言。

C#的一个优点在于它是强类型语言，而 JavaScript 不是。现在，在有经验的编程人员之间存在关于动态类型更适合开发的争议，特别是 Web 开发，但针对特定的游戏平台(比如 iOS)，通常最好甚至必须使用静态类型。Unity 甚至增加了指令#pragma strict 来强制 JavaScript 使用静态类型。尽管技术上可以这么做，但它破坏了 JavaScript 工作的基本原理，而如果想这么做，更好的方式是使用原本就是强类型的语言。

这仅是 Unity 中的 JavaScript 不同于其他地方的 JavaScript 的一个示例。Unity 中的 JavaScript 肯定类似于 Web 浏览器的 JavaScript，但在每个上下文中语言的工作方式还是有很多区别的。很多开发者将 Unity 中的 JavaScript 称为 UnityScript，这个名

字表明了它们之间的相似性，又和 JavaScript 做了区分。这种“相似但不同”的状态给编程人员带来了一些问题，既要学习 Unity 外的 JavaScript 知识，又要在 Unity 中应用学到的编程知识。

同时，由于这些原因，Unity 正在删除 JavaScript/UnityScript 支持。如博客 http://mng.bz/B9au 所述，这种支持正逐渐被淘汰。

下面编写和运行一些代码，尝试第一个示例。启动 Unity，创建一个新项目。在 Unity 的开始窗口中选择 New，如果 Unity 已经在运行，就选择 File | New Project，打开 New Project 窗口。输入项目名称，保留默认 3D 设置(后续章节会介绍 2D)，然后选择一个位置，保存这个项目。Unity 项目只是一个目录，其中包含了不同的资源和设置文件，所以可以在计算机上的任何位置保存项目。单击 Create Project，之后 Unity 会暂时消失，此时它在建立项目目录。

警告 Unity 项目会记住它们是在哪个 Unity 版本中创建的，如果尝试在不同版本的 Unity 中打开它们，将会出现警告。有时这些警告无关紧要(例如，当打开本书下载的示例时，出现警告时忽略它即可)，但有时，需要在打开项目之前备份项目。

同样，打开下载的示例时，Unity 可能会发出以下警告：重新构建库，因为找不到资源数据库！这是指项目的 Library 文件夹。该文件夹包含 Unity 生成并在工作时使用的文件，但不需要分发这些文件。

当 Unity 重新出现时，会看到一个空的项目。下一步，将讨论如何在 Unity 中执行程序。

1.3.1　代码在 Unity 中运行：脚本组件

Unity 中所有代码的执行都从链接到场景对象的代码文件开始。前述的组件系统最终是由代码文件构成的；游戏对象是由组件集合构建而成的，而这些集合可以包含要执行的脚本。

注意 Unity 将代码文件当成脚本，使用“脚本”这个定义，这和运行在浏览器中的 JavaScript 大致一样：代码运行在 Unity 游戏引擎中，不像编译好的代码运行在它自己的可执行环境中。但不要混淆这些概念，因为很多人对这个词的定义是不同的；例如“脚本”通常也指短小、完整的实用程序。Unity 中的脚本更像是独立的 OOP 类，附加到场景对象上的脚本是对象的实例。

Unity 中的脚本是指组件——并非所有脚本，注意，只有从 MonoBehaviour 继承的脚本才指组件，MonoBehaviour 是脚本组件的基类。MonoBehaviour 定义一些看不

见的基础工作，使组件附加到游戏对象上，而继承它，会提供一系列自动运行的方法(见代码清单 1.1)，可以重写它们。这些方法包括 Start()，当对象变成激活状态时(通常是加载带有该对象的关卡)会调用它一次。还包括 Update()，它会在每帧调用。因此，把代码放到这些预定义的方法中时，代码就会运行。

定义 帧是游戏循环代码中的一个周期。几乎所有的视频游戏(不仅 Unity，包括大多数视频游戏)都是围绕一个核心游戏循环建立的。当游戏运行时，代码会在每个周期中执行。每个周期都包括了绘制屏幕，因此命名为帧(电影也由一系列的帧组成)。

代码清单 1.1　基本脚本组件的代码模板

```
using UnityEngine;                                  ← 包含用于 Unity 和 Mono 类的名称空间
using System.Collections;
using System.Collections.Generic;

public class HelloWorld : MonoBehaviour {           ← 用于继承的语法
    void Start() {
        // do something once                        ← 把运行一次的代码放在这里
    }

    void Update() {
        // do something every frame                 ← 把每帧运行的代码放在这里
    }
}
```

这是创建新 C#脚本时文件包含的内容：定义一个合法的 Unity 组件需要的最少模板代码。Unity 有一个脚本模板隐藏在应用内部，当创建新脚本时，它复制该模板，并重命名新类，使之匹配文件名(本示例中的名称为 HelloWorld.cs)。这个脚本还包含空的 Start()和 Update()方法，因为我们大都在这两个方法中调用自己的代码。

为了创建脚本，从 Create 菜单中选择 C# Script，Create 菜单在 Assets 菜单中(注意 Assets 和 GameObjects 都有 Create 的列表，但它们是不同的菜单)，或者在 Project 视图的右击菜单中。输入新脚本的名称，例如 HelloWorld。如本章后面所述(见图 1-15)，单击并拖动这个脚本文件到场景中的对象上。双击这个脚本，它就会在另一个程序中自动打开，进行编辑，如下所述。

1.3.2　使用 MonoDevelop，跨平台的 IDE

准确而言，编程并不是在 Unity 中进行的，代码是 Unity 指向的独立文件。脚本文件能在 Unity 内创建，但仍然需要使用文本编辑器或 IDE 在初始为空的文件中编写所有代码。Unity 绑定了 MonoDevelop，它是一个开源、跨平台、编写 C#的 IDE(见图 1-14)。可以访问 www.monodevelop.com 进一步学习这个软件，但这里使用的版本是和 Unity

绑定的，而不是从网站下载的版本，因为基础软件进行了一些修改，以更好地集成到 Unity 中。

图 1-14　MonoDevelop 中的界面

注意　MonoDevelop 把文件组织成组，称为解决方案。Unity 自动生成一个解决方案，它包含所有脚本文件，因此通常不必关心它们。

C#最开始是 Microsoft 的产品，能使用 Visual Studio 在 Unity 中编程吗？答案是可以。支持工具可以使用 Visual Studio 给 Unity 编程(特别是调试和断点能正常工作)。要查看此支持工具是否已经安装，请检查 Debug 菜单中的 Attach Unity Debugger 选项。如果没有安装，只需要运行 Visual Studio Installer，来修改安装，并查找 Unity 游戏开发模块。

本书主要使用 MonoDevelop，但如果已经使用了 Visual Studio 来编程，则可以继续使用它，学习本书的内容不会产生任何问题(该章之后不会再讨论 IDE)。但是将工作流限制在 Windows 上，会和使用 Unity 的最大优点背道而驰。虽然 C#最初是一个 Microsoft 产品，只能在 Windows 和.NET Framework 中工作，但是现在它已经成为开源标准的语言，而且有一个有意义的跨平台框架：Mono。Unity 使用 Mono 支撑它的编程，而使用 MonoDevelop 允许使整个项目的开发工作流保持跨平台。

警告　MonoDevelop 是与 Unity 2017.1 版本捆绑在一起的 IDE，但如 Unity 博客 http://mng .bz/9HR8 所述，这将在 Unity 2018.1 版本中改变。

时刻记住，尽管代码是使用 Visual Studio 或 MonoDevelop 编写的，但代码不在 Visual Studio 或 MonoDevelop 中运行。IDE 只是一个强大的文本编辑器，只有在 Unity

中单击 Play 按钮，代码才会运行。

1.3.3　打印到控制台：Hello World！

至此，项目中已经有了一个空脚本，但场景中还需要一个附加这个脚本的对象。图 1-1 描述了组件系统的工作原理；脚本是一个组件，因此需要设置为对象上的一个组件。

选择 GameObject | Create Empty，在 Hierarchy 列表中将出现一个空的 GameObject。现在从 Project 视图将脚本拖到 Hierarchy 视图，并放在那个空的 GameObject 上。如图 1-15 所示，Unity 将高亮验证可以放置脚本的地方，在 GameObject 上释放它，使脚本附加到对象上。为验证脚本是否已附加到对象上，选择该对象并查看 Inspector 视图。其中会列出两个组件：一个是 Transform 组件，它是基础位置/旋转/缩放组件，所有对象都包含该组件，而且不能移除它；另一个组件就是脚本。

图 1-15　将脚本链接到 GameObject 上

注意　将对象从一个地方拖到其他对象上并释放的操作是很常见的。Unity 中很多不同的链接都是通过将对象拖到其他对象之上来创建的，不只是将脚本关联到对象上。

脚本链接到对象后，其结果将如图 1-16 所示，脚本像组件那样显示在 Inspector 中。播放场景时，脚本就会执行，不过现在还不会发生任何事情，因为还没编写任何代码。下面接着介绍下一步！

打开 MonoDevelop 中的脚本，回到代码清单 1.1。当学习新的编程语言环境时，最经典的做法是输出文本“Hello World!”，将这行文本添加到 Start()方法中，如代码清单 1.2 所示。

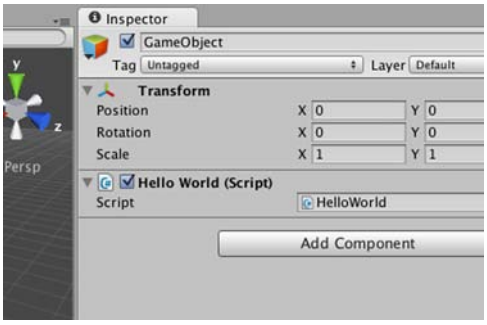

图 1-16　在 Inspector 中显示链接的脚本

代码清单 1.2　添加一个控制台消息

```
...
void Start() {
    Debug.Log("Hello World!");      ←— 在此添加了日志命令
}
...
```

Debug.Log()命令将一个消息输出到 Unity 的 Console 视图中。与此同时，由于 Debug.Log 这一行代码出现在 Start()方法中，Start()方法会在对象激活时调用一次。换句话说，Start()方法会在单击编辑器中的 Play 时调用一次。一旦将日志命令添加到脚本中(请确认保存了脚本)，请单击 Unity 中的 Play 按钮并切换到 Console 视图，就会显示消息“Hello World!”。恭喜，第一个 Unity 脚本已经完成了！后续章节中的代码会更复杂，但这是重要的第一步。

“Hello World!”简明步骤

下面重申和总结一下前几页的步骤:

(1) 创建新项目。

(2) 创建新的 C#脚本。

(3) 创建空的 GameObject。

(4) 将脚本拖动到对象上。

(5) 给脚本添加日志命令。

(6) 单击 Play 按钮。

现在可以保存场景，这将创建一个带 Unity 图标的.unity 文件。场景文件是当前游戏中加载的任何东西的快照，因此可以在以后重新载入这个场景。保存这个场景没有什么价值，因为它太简单了(只是一个单一的空 GameObject)，但如果不保存这个场景，当退出 Unity 再回到这个项目时，就会发现它又变成空的。

脚本中的错误

为了查看 Unity 如何显示错误，故意在 HelloWorld 脚本中添加了一个拼写错误。例如，如果输入额外的圆括号，这个错误消息会出现在 Console 中，并带有红色的错误图标。如图 1-17 所示。

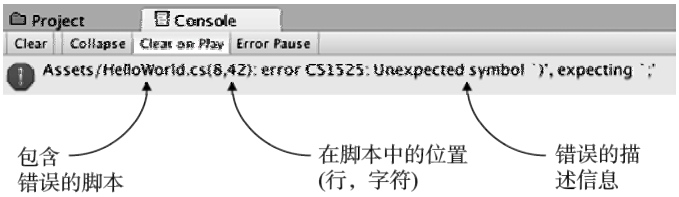

图 1-17　错误消息

1.4　小结

- Unity 是一个多平台的开发工具。
- Unity 的可视化编辑器包括可以协同工作的几部分。
- 脚本是作为组件附加到对象上的。
- 代码是使用 MonoDevelop 编写的内部脚本。