

# C#游戏开发快速入门教程

(内部资料 v2.0)



大学霸

[www.daxueba.net](http://www.daxueba.net)



# 前言

C#是微软发布的高级程序设计语言，这门语言和 C 语言一样，已经成为了大学计算机相关专业必修的一门语言。很多初学这门语言的同学，还以为 C#只能开发 Windows 桌面应用程序了，写写 ASP.NET 网站。本书将为读者纠正这种观念，使用 C#一样可以开发出精彩的游戏！

学过 C#的人，可能接触过 Visual Studio 2010 或者此软件的其它版本，因此也应该接受一种观念，那就是：即使是开发 Windows 桌面应用，也需要有集成开发环境来辅助 C#代码的编写。那么同理，使用 C#编写游戏也需要有集成开发环境的辅助才行！本书选择的开发环境是 Unity。

或许很多人对 Unity 并不了解，这里就做些简单的说明。Unity 是由 Unity Technologies 开发的一个多平台的综合型的游戏开发工具。使用它开发的游戏可以运行在所有的主流游戏平台上，如，Windows、Xbox、Mac、iPhone、Windows Phone 8、Android 等等。这款软件在国内业界的火热程度也不容忽视。

2014 年 4 月 13-14 日，在北京国家会议中心举办的 Unity 亚洲开发者大会上，据 Unity 大中华区总裁符国新的介绍，自 2012~2014 年，短短 2 年时间，业务量就增长了 10 倍；在全球超过 300 万的开发者中，有 1/4 在中国；超过 5000 家游戏公司和工作室在使用 Unity 开发；Unity 插件在全球有 4 亿次的下载量，仅中国就超过 1 亿次……

在大学已成为必修课的计算机语言，再加上最近火的一塌糊涂的游戏集成开发环境，没错！他们就是本书的全部。要开发游戏，C#需要学的有多精通才行？对 Unity 得多熟悉才行？得用多长时间才能做出一个能拿得出手的游戏？这些问题的答案在本书的各章节都有体现，读者只需慢慢体会就能明白，因为似乎仅仅用一个简单的否定词并不足以令读者信服。

## 1.学习所需的系统和软件

- ☐ 安装 Windows 10 操作系统
- ☐ 安装 Unity 5.4.1

## 2.学习建议

大家学习之前，可以到百度网盘 [xxxxxxxxxxxxxxxxxxxxxx](#) 获取相关的资料和软件。如果大家在学习过程遇到问题，也可以将问题发送到邮箱 [xxxxxxxxxxxxxx](#)。我们尽可能给大家解决。

# 目 录

第 1 章 使用 C#编写游戏前的准备 .....	1
1.1 Unity 概述 .....	1
1.1.1 Unity 简介 .....	1
1.1.2 Unity 的下载 .....	1
1.1.3 Unity 的安装 .....	2
1.2 游戏项目 .....	4
1.3 查看特定组件的参考手册 .....	5
1.4 创建并编辑 C#脚本 .....	7
2.1 构建游戏场景 .....	11
2.1.1 新建游戏场景 .....	11
2.1.2 添加游戏对象 .....	11
2.2 改变游戏对象的状态 .....	13
2.2.1 Main Camera 与 Game 视图 .....	13
2.2.2 Transform 组件 .....	14
2.2.3 直接操作游戏对象 .....	15
2.3 Scene 视图的快捷操作 .....	16
2.4 使用脚本 .....	19
2.4.1 示例效果展示 .....	19
2.4.2 脚本的构成 .....	20
2.4.3 将脚本赋予游戏对象的方法 .....	22
2.4.4 运行游戏 .....	23
2.5 小结 .....	23
第 3 章 游戏对象的属性——变量 .....	24
3.1 游戏对象的属性 .....	24
3.2 指代数据的名称 .....	25
3.2.1 变量命名规则 .....	25
3.2.2 变量命名约定 .....	25
3.2.3 变量名与属性名的些许不同 .....	26
3.3 属性名的可见性 .....	26
3.4 设定属性的数据类型 .....	27
3.4.1 数据类型 .....	27
3.4.2 属性数据类型的体现形式 .....	28
3.5 使用脚本变量 .....	29
3.5.1 声明变量 .....	29
3.5.2 变量的初始化 .....	30
3.5.3 变量的运算 .....	30

3.5.4	成员变量 .....	31
3.5.5	变量的作用域 .....	32
3.6	语句 .....	33
3.7	游戏示例 .....	34
3.8	小结 .....	37
第4章	游戏对象的行为逻辑——方法 .....	38
4.1	游戏对象的行为逻辑 .....	38
4.2	脚本中的方法 .....	38
4.3	使用脚本方法 .....	39
4.3.1	方法与变量 .....	40
4.3.2	定义方法 .....	40
4.3.3	调用方法 .....	41
4.3.4	方法使用示例 .....	41
4.4	Unity 内置的方法 .....	42
4.5	方法的参数 .....	44
4.5.1	参数的作用 .....	44
4.5.2	对于游戏的实际意义 .....	44
4.6	减少代码的重复书写 .....	46
4.7	游戏示例 .....	48
4.8	小结 .....	51
第5章	游戏执行路径的选择——判断 .....	52
5.1	游戏的执行路径 .....	52
5.2	判断玩家的选择 .....	52
5.2.1	最常用的判断语句——if .....	53
5.2.2	if 语句游戏示例 .....	54
5.2.3	其它判断语句 .....	57
5.3	循环遍历每个数据，并做出判断 .....	58
5.3.1	大量数据的存储与引用 .....	59
5.3.2	遍历大量的数据 .....	62
5.4	游戏示例 .....	66
5.4.1	游戏运行效果 .....	66
5.4.2	游戏实现步骤 .....	67
5.4.3	游戏脚本的编写 .....	67
5.5	小结 .....	73
第6章	游戏对象间的交流 .....	74
6.1	在游戏外部的体现 .....	74
6.2	在游戏内部的体现 .....	75
6.3	点运算符 .....	76
6.3.1	对象、组件与类 .....	76
6.4	在脚本中使用点运算符 .....	77
6.4.1	访问组件自己的属性和方法 .....	77
6.4.2	访问同一对象上的其它组件 .....	78

6.4.3	访问其它对象上的组件 .....	80
6.5	游戏示例 .....	81
6.6	小结 .....	85
第 7 章	游戏的中枢——状态管理机制 .....	86
7.1	概述 .....	86
7.2	工作流程 .....	87
7.2.1	游戏状态 .....	87
7.2.2	将游戏控制权授予游戏状态 .....	87
7.2.3	游戏状态的切换 .....	88
7.2.4	实时记录当前所处的游戏状态 .....	88
7.3	类的实例化 .....	89
7.4	引入接口 .....	92
7.4.1	接口概述 .....	92
7.4.2	示例演示 .....	93
7.5	小结 .....	95
第 8 章	状态管理机制使用示例 .....	96
8.1	添加游戏状态 .....	96
8.2	核心脚本对游戏状态的控制 .....	98
8.2.1	核心脚本遇到的问题 .....	98
8.2.2	示例演示 1 .....	99
8.2.3	示例演示 2 .....	100
8.3	核心脚本里的 OnGUI() .....	101
8.3.1	添加 OnGUI() .....	102
8.3.2	示例演示 1 .....	102
8.3.3	示例演示 2 .....	104
8.4	游戏场景的切换 .....	105
8.4.1	添加游戏场景 .....	105
8.4.2	引入的问题及解决方法 .....	107
8.4.3	示例演示 .....	108
8.5	示例代码 .....	111
8.5.1	核心脚本 .....	111
8.5.2	接口 IStateBase .....	112
8.5.3	游戏状态 .....	112
8.6	小结 .....	115
第 9 章	游戏示例——胶囊的净化战争 .....	116
9.0	游戏剧情梗概 .....	116
9.1	搭建游戏的框架 .....	116
9.1.1	游戏运行流程图 .....	117
9.1.2	核心脚本 .....	117
9.1.3	接口 .....	118
9.1.4	游戏状态 .....	119
9.1.5	框架运行效果 .....	124

9.2	存储游戏数据 .....	126
9.3	添加游戏启动画面 .....	128
9.4	添加“胶囊” Player 对象 .....	131
9.5	“胶囊”的设置 .....	133
9.5.1	添加脚本组件 .....	134
9.5.2	可以自转的胶囊 .....	134
9.5.3	胶囊的颜色选择 .....	139
9.5.4	胶囊生命值的设置 .....	143
9.5.5	切换到下一游戏状态 .....	144
9.6	胶囊的移动 .....	145
9.6.1	添加刚体组件 .....	146
9.6.2	负责移动效果的代码 .....	147
9.6.3	移动效果展示 .....	149
9.7	游戏视图的切换 .....	150
9.7.1	添加摄像机 .....	150
9.7.2	实现视图切换功能 .....	152
9.7.3	应用于游戏 .....	156
9.8	荒芜的星球及其原住民 .....	161
9.8.1	添加荒芜的地面 .....	161
9.8.2	原住民：益生菌和“埃博拉”病毒 .....	163
9.9	胶囊的进化之旅 .....	167
9.9.1	凶猛的埃博拉病毒 .....	167
9.9.2	病毒的净化过程 .....	170
9.9.3	触发净化的时机 .....	173
9.10	添加游戏分数与生命值 .....	176
9.10.1	游戏的规则 .....	176
9.10.2	分数和生命值的增减 .....	178
9.10.3	游戏各状态的切换 .....	179
9.10.4	游戏视图上的分数和生命值 .....	185
9.11	游戏流程图及源码概述 .....	186
9.11.1	游戏运行流程图 .....	187
9.11.2	核心脚本 .....	187
9.11.3	接口 .....	189
9.11.4	游戏状态 .....	189
9.11.5	其它功能脚本 .....	199
9.12	小结 .....	204

# 第 1 章 使用 C#编写游戏前的准备

使用 C#可以编写游戏？当然可以！但是除了 C#这门编程语言外，要编写游戏，还需要有一个平台。在本书中，我们选择的这个平台是 Unity。本章会首先为你简要介绍 Unity 这个游戏开发平台，然后就是最最重要的一一在其中创建 C#脚本的方法，毕竟 C#的代码是要写在这个脚本中的。

## 1.1 Unity 概述

在正式开始学习使用 C#开发游戏之前，本节就来简要说明下 Unity，及其下载和安装的方法。

### 1.1.1 Unity 简介

Unity 是一款跨平台的专业游戏引擎，使用它可以轻松的开发出各种 2D 和 3D 游戏，然后部署到各种游戏平台上。当然也包括所有主流游戏平台：Windows、iOS、Android、Xbox 360、PS3。

2012 年，就有权威机构统计过，国内 53.1%的人使用 Unity 进行游戏开发；有 80%的手机游戏是使用 Unity 开发的；而苹果应用商店中，有超过 1500 款游戏使用 Unity 开发。所以说，现在 Unity 的普及程度更是不可小觑。

对于使用 Unity 开发游戏的开发者，则无需过多考虑平台之间的差异，只需把精力集中到制作高质量的游戏即可，这真正做到“一次开发，到处部署”。

基于以上几个原因，本书强烈建议读者使用 Unity 作为使用 C#开发游戏的平台，而本书的所有内容也都是围绕着它们而展开的。

### 1.1.2 Unity 的下载

Unity 的安装包可以从它的官方网站下载到，网址为 <http://unity3d.com/cn/get-unity>。打开该网页，效果如图 1.1 所示。Unity 提供 PERSONAL（个人版）、Plus（加强版）、Pro（专业版）和 Enterprise（企业版）。其中，个人版是免费的，其他版本都是收费的。作为初学者，使用个人版即可。单击个人版下方的立即下载按钮，即可跳转到下载页面，如图 1.2 所示。



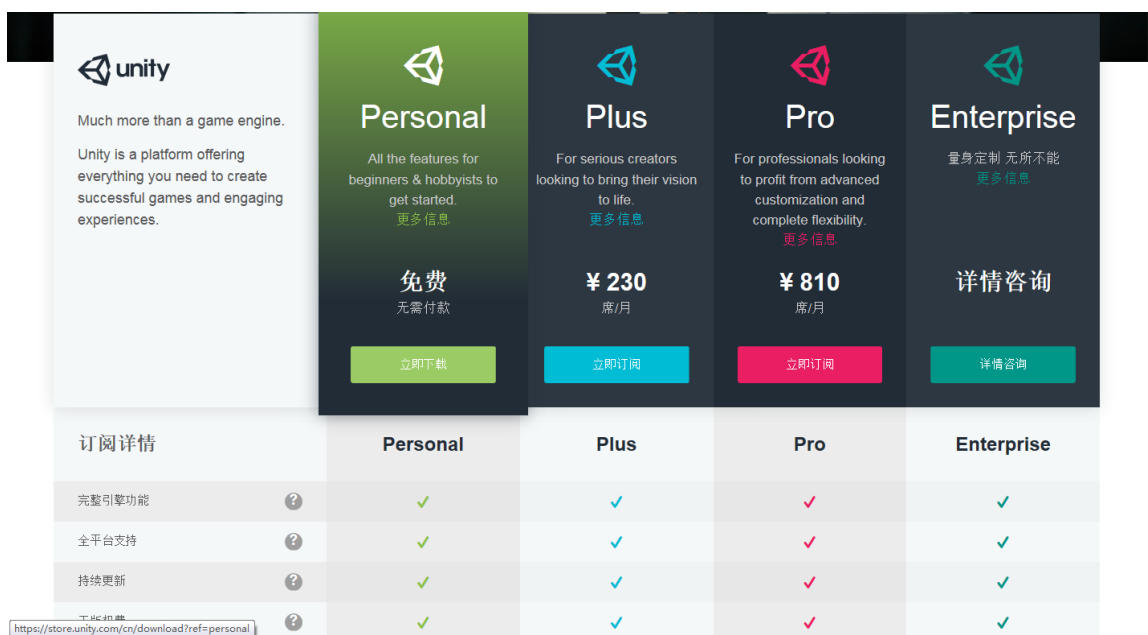


图 1.1 Unity5 版本页面



图 1.2 个人版下载页面

在该网页中, 显示 Unity 的最新版本信息, 如发布日期、版本号和大小。例如, 当前页面显示 2016 年 7 月 28 日, Unity 发布了最新版本 5.4.1, 默认平台是 Windows。由于 Unity 时时更新, 所以用户下载到的版本可能有所差异。小版本的变化, 不会影响到大家的学习。

为了方便用户下载, 这里提供的是一个 Unity 下载助手程序, 所以文件只有 636KB。用户可以单击页面中间的“选择 MAC OS X”蓝色文字选项, 选择苹果版本的 Unity 程序。单击“下载安装程序”按钮, 跳转到下载页面, 并在页面下方弹出下载提示栏, 如图 1.3 所示。

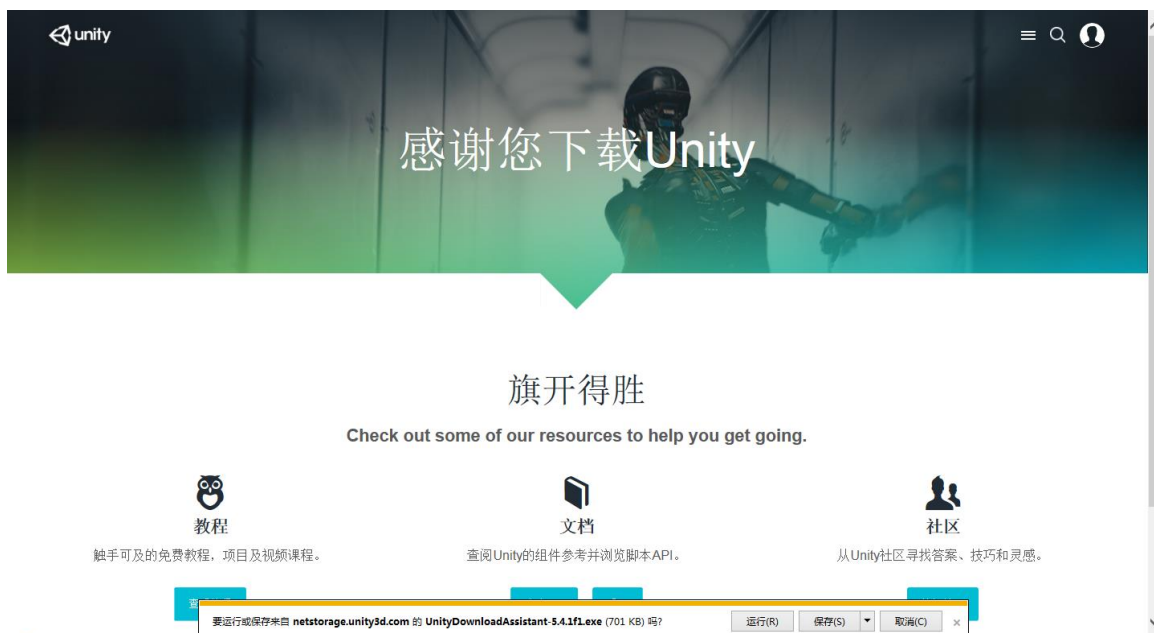


图 1.3 下载提示

单击“保存”按钮，选择保存位置，进行文件下载。下载完后，就可以得到一个文件名为 UnityDownloadAssistant-\*\*\*\*\*.exe 的文件（其中，\*\*\*\*\*表示具体的版本号）。

### 1.1.3 在线安装 Unity

双击下载到的 Unity 下载助手程序，就可以开始 Unity 的安装了。操作过程如下：

（1）双击 UnityDownloadAssistant-\*\*\*\*\*.exe 文件后，弹出 Unity 下载助手说明对话框，如图 1.4 所示。

（2）单击 Next 按钮，弹出许可协议对话框，如图 1.5 所示。

（3）单击 I Agree 按钮，同意 Unity 软件使用许可协议，并弹出组件选择对话框，如图 1.6 所示。

（4）该对话框中显示用户可以选择安装的组件，如 Unity 5.4.1f1（编辑器，即主程序）、Documentation（文件插件）、Standard Assets（标准资源）、Example Project（示例项目）、Microsoft Visual Studio Tools for Unity（Visual Studio 工具组件）。用户根据需要选择对应的组件，然后单击 Next 按钮，弹出下载和安装位置选择对话框，如图 1.7 所示。

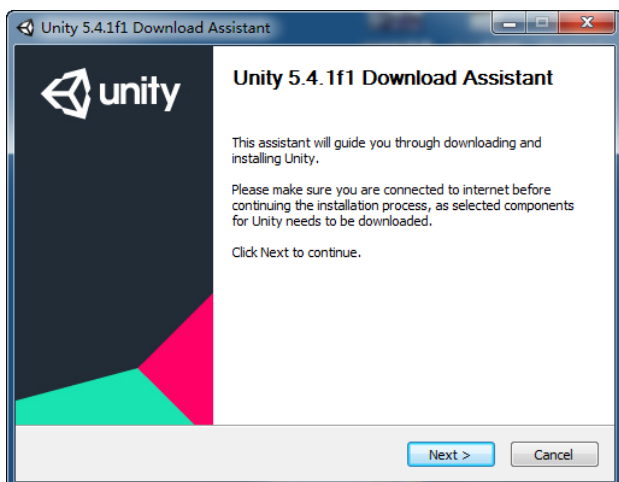


图 1.4 Unity 下载助手说明对话框

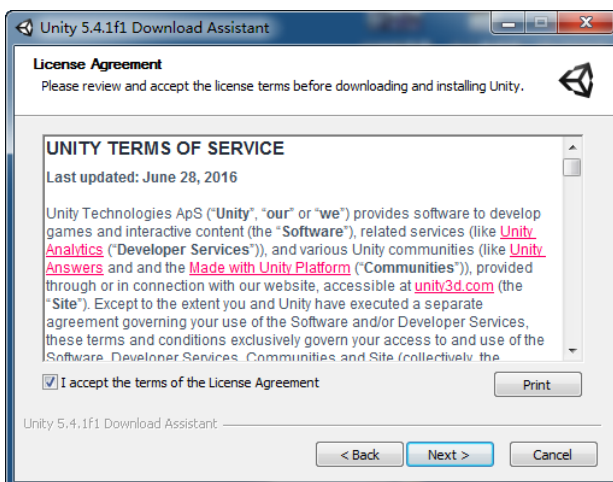


图 1.5 许可协议对话框

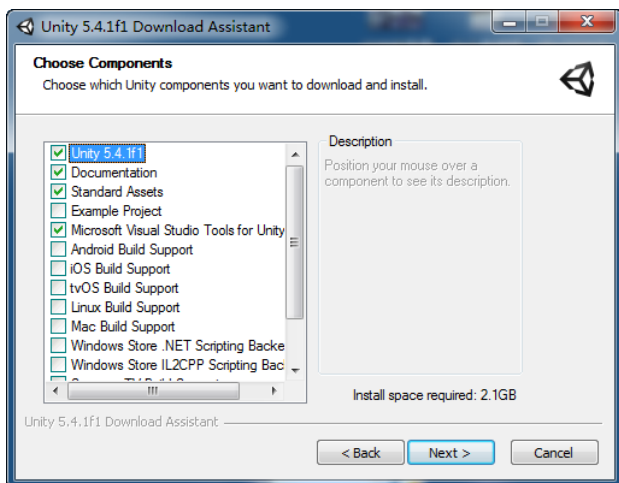


图 1.6 组件选择对话框

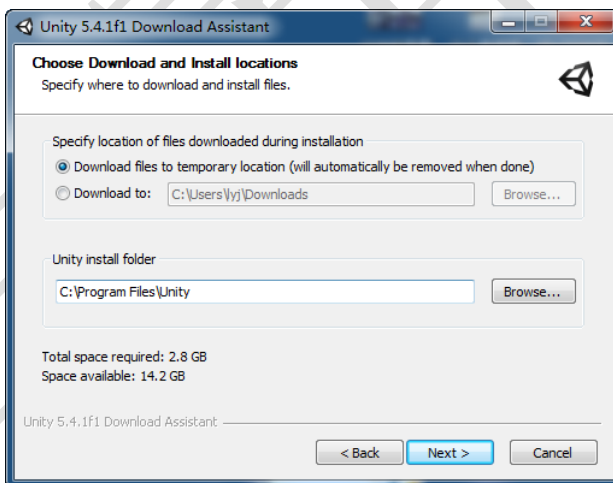


图 1.7 下载和安装位置选择对话框

(5) 在该对话框中，用户可以选择下载组件所保存的位置和安装位置。如果使用默认下载保存位置，当安装完成后，下载的组件文件都会被删掉。建议用户选择单击 **Download to** 单选按钮，并单击 **Browse** 按钮，选择组件的保存位置。设定完成后，单击 **Install** 按钮，弹出下载和安装对话框，如图 1.8 所示。

(6) 由于下载的文件较大，需要较长时间，安装完成后，弹出安装完成对话框，如图 1.9 所示。

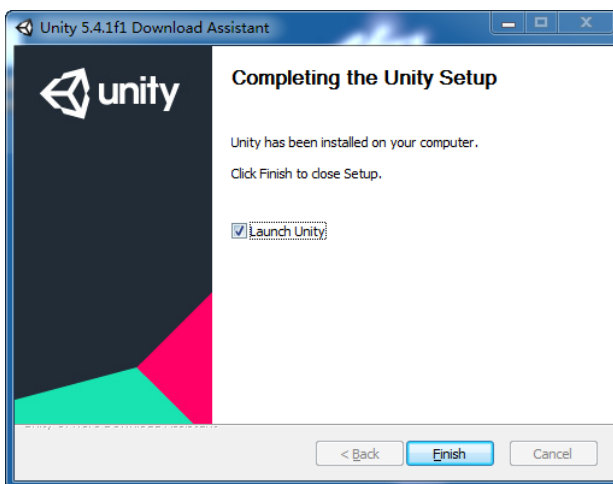
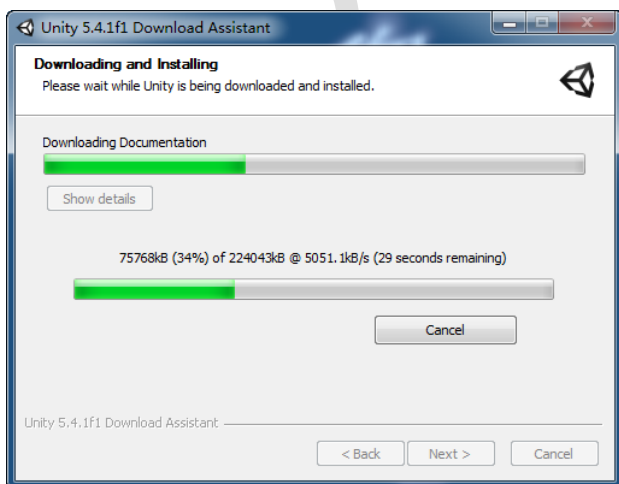


图 1.8 下载和安装对话框

图 1.9 安装完成对话框

(7) 单击 Finish 按钮, 完成 Unity 安装, 并打开 Unity 程序。

### 1.1.4 离线安装 Unity

由于网络差异, 用户连接 Unity 服务器速度可能不稳定, 会造成下载失败, 或者下载较慢。这时, 用户采用离线方式进行安装。在图 1.2 中, 用户在“其他下载”选项中, 选择下载的组件, 如图 1.10 所示。



图 1.10 其他下载组件

在这里, 用户可以直接选择下载 Unity 的各个组件, 如 Unity 编辑器、内置着色器、标准资源、示例项目。单击某一项, 浏览器就开始下载该程序。从 Unity 5.2 开始, 官方提供 Torrent 下载方式。用户可以下载种子文件后, 然后使用迅雷之类下载工具进行下载。

用户下载后, 就可以单独安装了。这里以 Unity 编辑器安装为例, 介绍安装方式。

(1) 双击下载的 UnitySetup\*\*\*\*.exe 文件 (其中, \*\*\*\*表示版本号), 弹出安装欢迎对话框, 如图 1.11 所示。

(2) 单击 Next 按钮, 弹出许可协议对话框, 如图 1.12 所示。

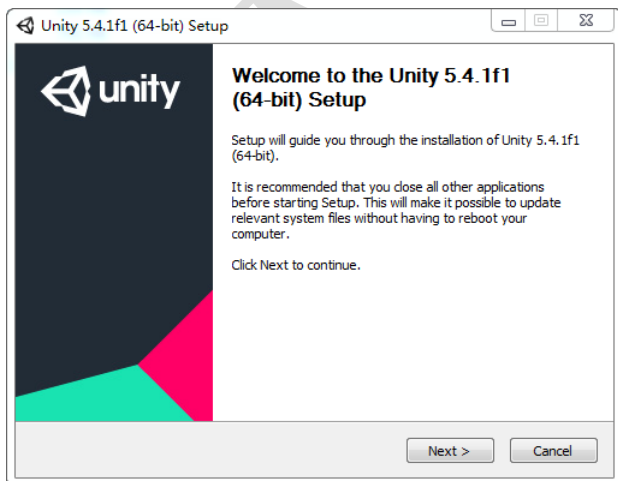


图 1.11 安装欢迎对话框

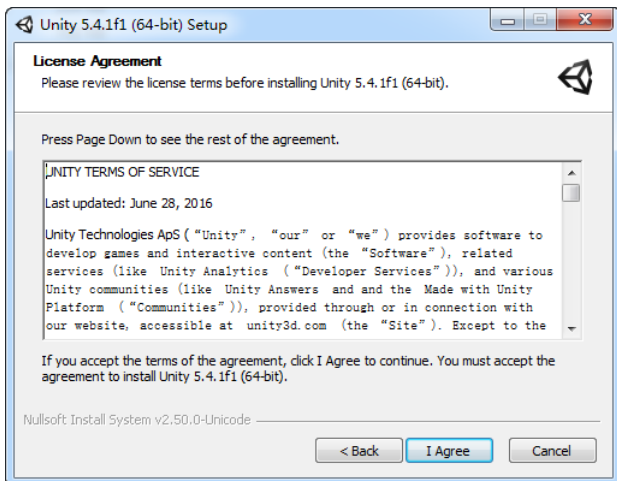


图 1.12 许可协议对话框

(3) 单击 I Agree 按钮, 弹出组件选择对话框, 如图 1.13 所示。

(4) 其中, MonoDevelop 可以用于 Unity 代码编写和调试。这里勾选该选项。单击 Next 按钮, 弹出安装位置选择对话框, 如图 1.14 所示。

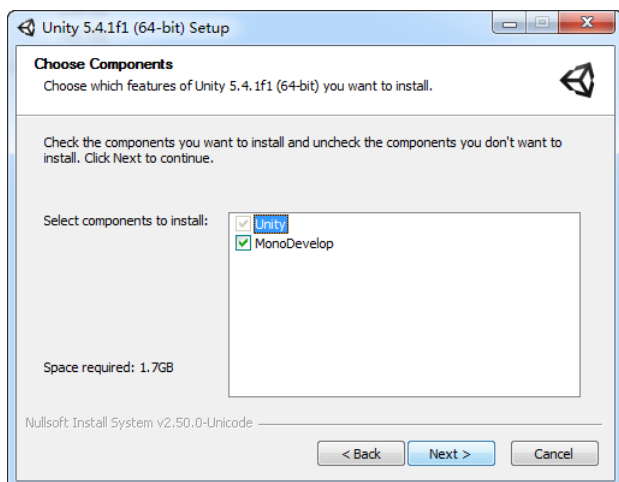


图 1.13 组件选择对话框

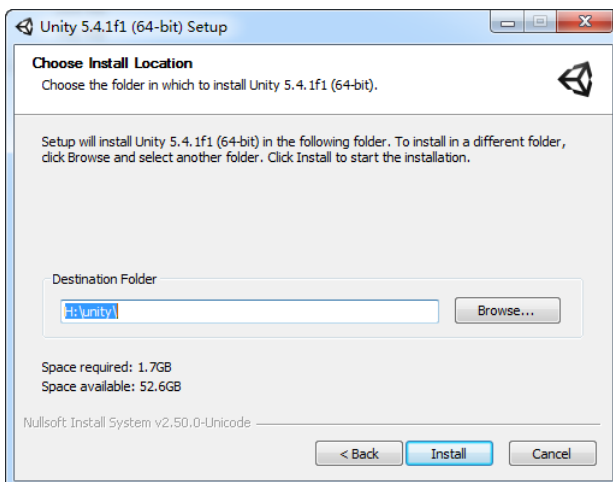


图 1.14 安装位置选择对话框

(5) 由于该组件要占用 5.6GB 的磁盘空间, 所以用户需要选择一个较大的分区来保存安装文件。设置后, 单击 Install 按钮, 弹出安装对话框, 如图 1.15 所示。

(6) 安装结束后, 弹出安装完成对话框, 如图 1.16 所示。

(7) 单击 Finish 按钮, 结束 Unity 安装过程, 并打开 Unity 软件。

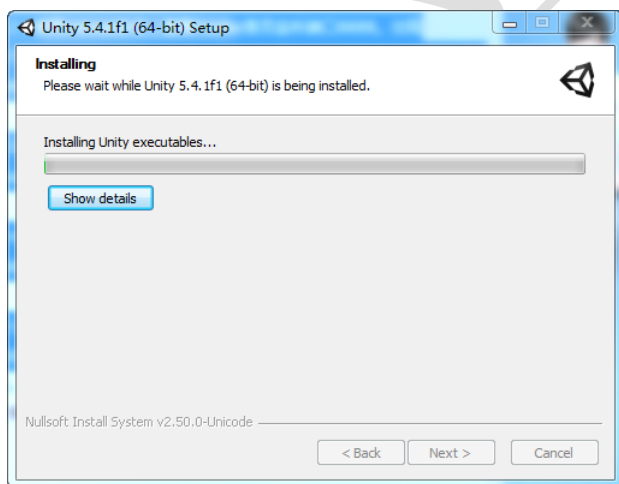


图 1.15 安装对话框

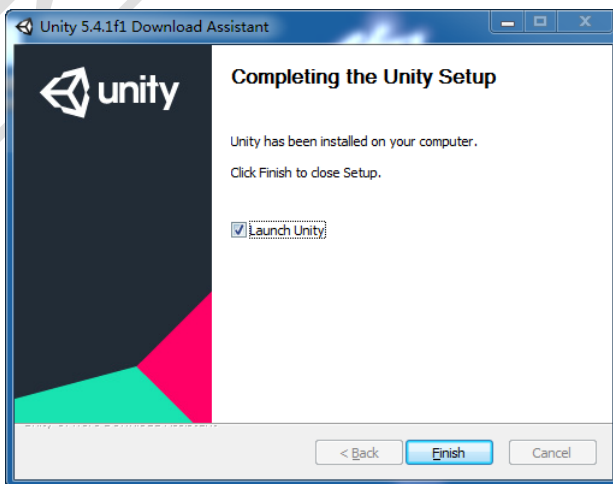


图 1.16 安装完成对话框

### 1.1.5 启动 Unity

安装完成, 用户就可以启动 Unity。从开始菜单中, 找到 Unity 命令, 单击运行, 弹出 Unity 欢迎界面, 如图 1.17 所示。在该界面中, 用户可以输入 Unity 帐号和密码, 然后单击 Sign In 按钮, 进入 Unity 界面。如果没有帐号, 用户可以到 Unity 官网的用户注册页面 <https://accounts.unity3d.com/sign-up>, 注册一个帐号。由于 Unity 会通过邮箱验证, 所以用户需要填写真实可用的邮箱。用户可以单击 Work offline 按钮, 采用不登录帐号的方式使用 Unity。这时, 用户就会进入创建项目对话框, 如图 1.18 所示。

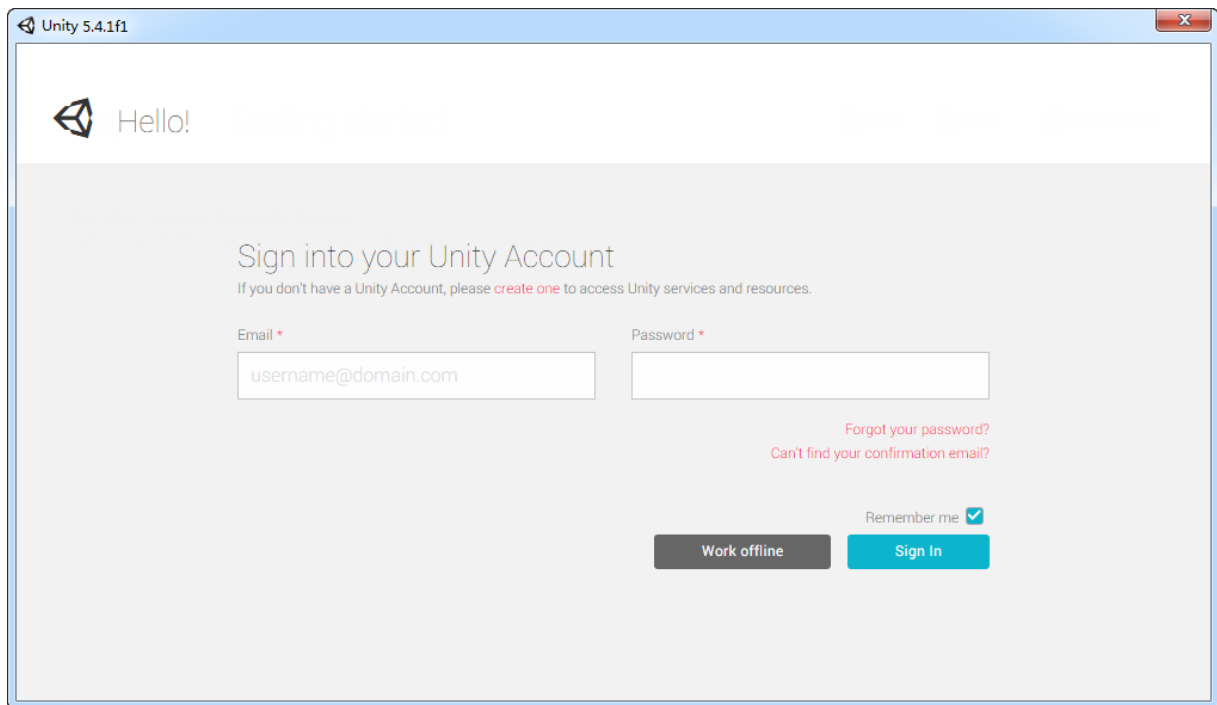


图 1.17 Unity 欢迎界面

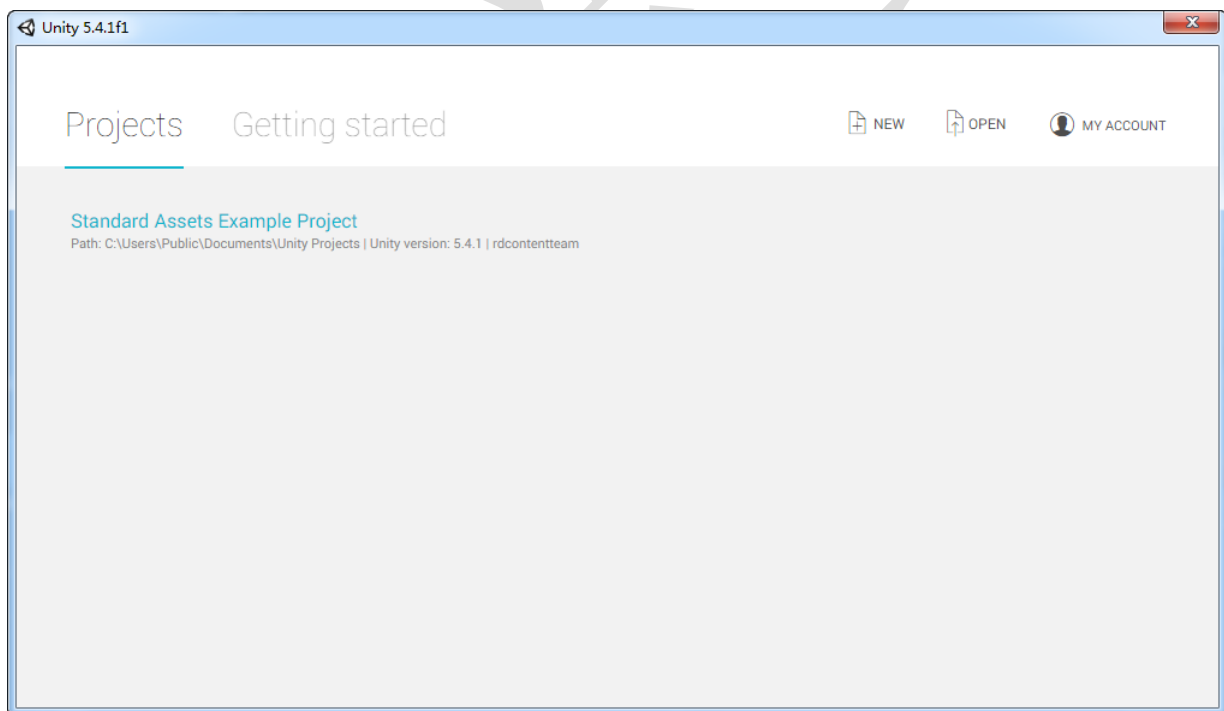


图 1.18 创建项目对话框

## 1.2 游戏项目

Unity 是一个基于项目的应用。这就意味着每开发一个新游戏，都要创建一个新项目。一个项目就

代表一个游戏，不管游戏是 2D 还是 3D 的。读者可以把项目当做容器，它包含了开发游戏时，自动生成还有引入的所有文件。创建项目的操作过程如下：

(1) 在图 1.18 的界面中，单击右上角的 NEW 按钮，弹出创建项目对话框，如图 1.19 所示。

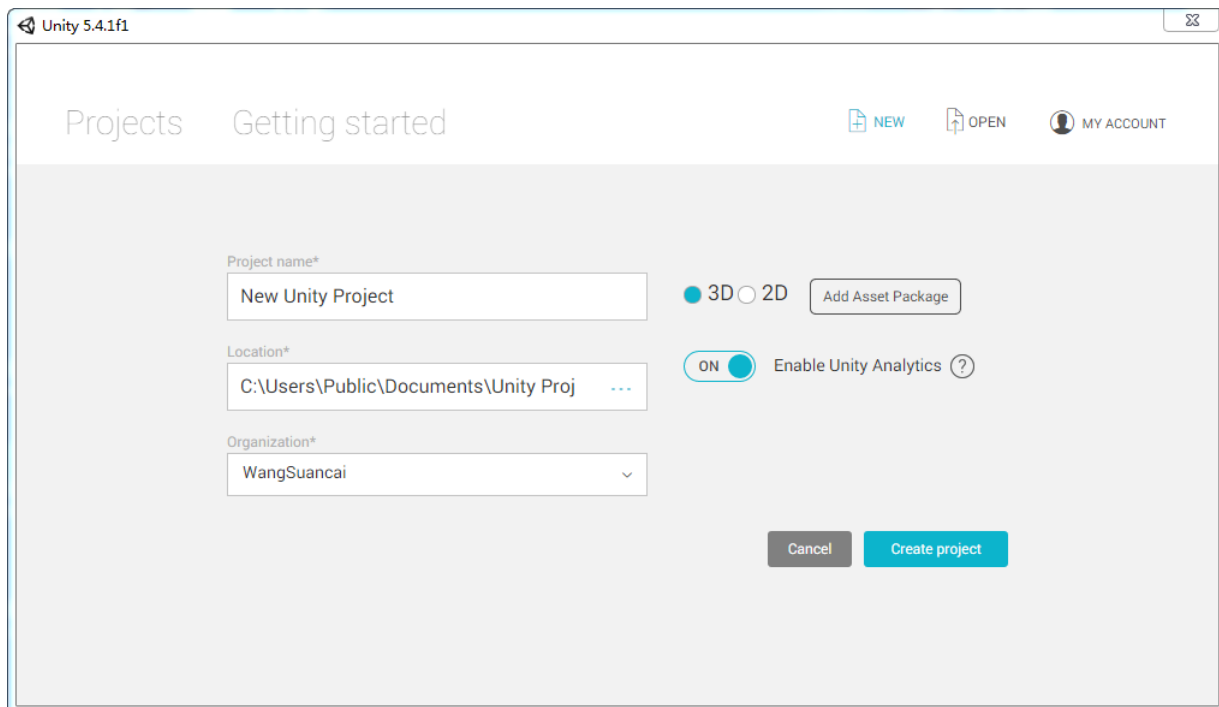


图 1.19 创建项目对话框

(2) 在 Project name 文本框中输入项目的名称；在 Location 文本框中设置项目的保存位置。单击 2D 或者 3D 链接，选择项目的类型。

(3) 单击 Create project 按钮，创建一个项目，并进入 Unity 编辑界面，如图 1.20 所示。

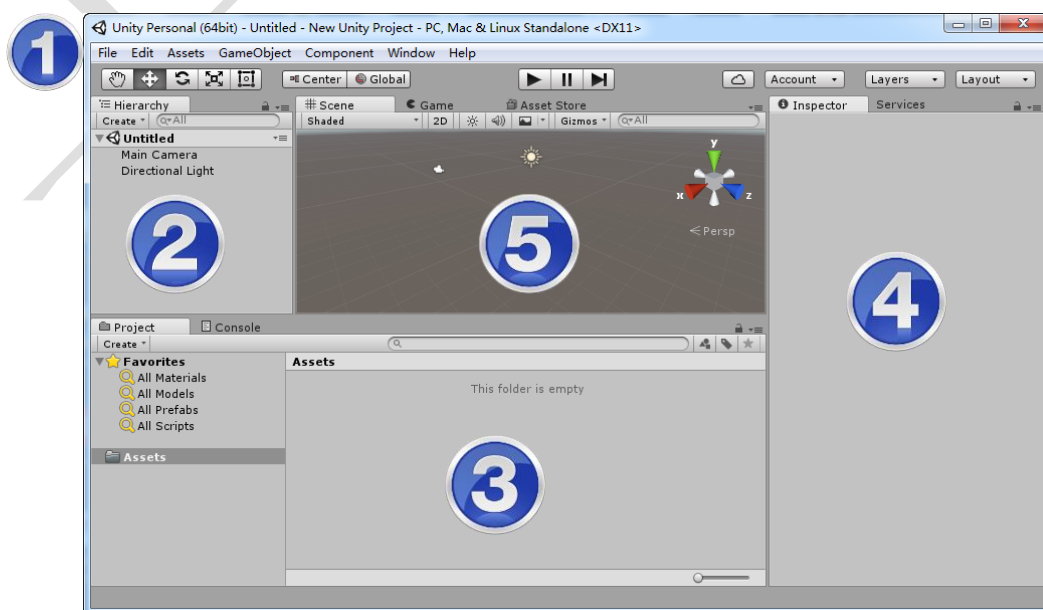


图 1.20 Unity 主界面



其中，标号 1 的部分是应用程序菜单；标号 2 的部分是 Hierarchy 视图；标号 3 的部分是 Project 视图；标号 4 的部分是 Inspector 视图；标号 5 的部分是 Scene 视图

同时，在指定的项目存放位置下，生成了 4 个子文件夹。它们分别为 Library、Assets、ProjectSettings 和 Temp，如图 1.21 所示。值得注意的是 Assets 文件夹，因为在 Unity 里使用 C#编写的脚本，都将被 Unity 存储到 Assets 文件夹中。

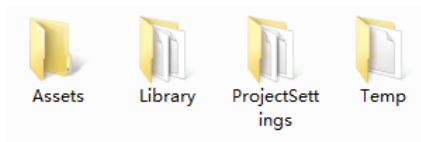


图 1.21 Unity 在项目文件夹里生成的 4 个子文件夹

### 1.3 查看特定组件的参考手册

在 Hierarchy 视图里的每一项都被称为“游戏对象”（GameObject）。而 Hierarchy 视图里会列出当前游戏场景中的所有“游戏对象”，如图 1.22 所示。此时，游戏场景包含两个游戏对象 Main Camera 和 Directional Light。

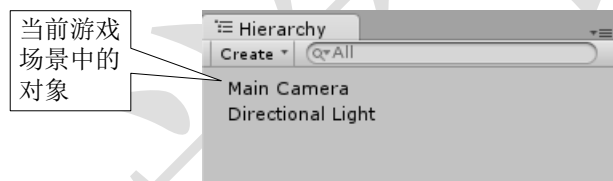


图 1.22 Hierarchy 视图里的 Main Camera 对象

选中 Hierarchy 视图里的 Main Camera 对象以后，就会在 Inspector 视图里看到 Main Camera 对象的属性。所有属性都在“组件”（Component）里。读者可以单击组件左侧的 ▢ 按钮，展开组件，即可看到此组件下的所有属性，如图 1.23 所示。

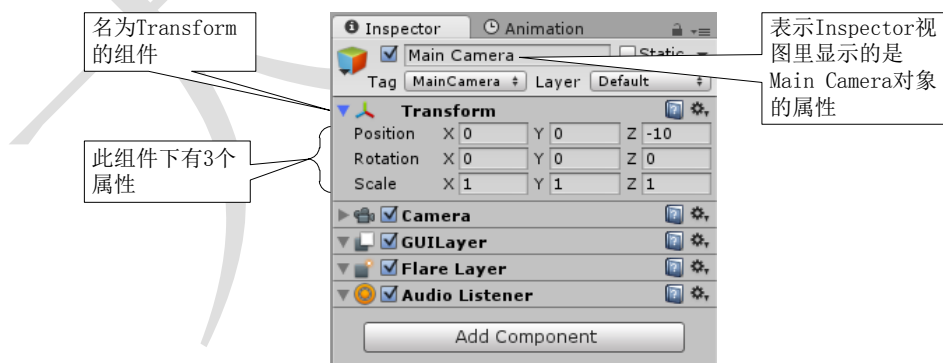


图 1.23 Inspector 视图里的组件，及其里面包含的属性

如果读者在 Inspector 视图里，看到不认识的组件及其属性时，可以单击此组件右侧的 ? 按钮，然后系统默认的浏览器就会被打开。显示的内容是记录在本地的帮助文档，且此文档已经自动定位了指定组件的帮助信息，如图 1.24 所示，单击 Transform 组件右侧的 ? 按钮。

注意：在打开 IE 浏览器后，IE 浏览器会提示“Internet Explorer 已限制此网页运行脚本或 ActiveX 控件”。用户需要单击右侧的“允许阻止的内容”按钮，才能显示完整的网页内容。





图 1.24 系统默认的浏览器，自动定位了 Transform 组件的帮助信息

对象上的每个组件，都可以在 C#脚本代码中被引用，关于这点会在本书后面的章节中介绍。在本节的学习中，读者只需要知道，如果需要在 C#脚本中使用代码来引用这个组件，该如何写代码，这时就需要用到脚本参考手册。本节前面我们查看了 Main Camera 对象上 Transform 组件的参考手册。在此手册 Transform 标题的下方，有个名为 SWITCH TO SCRIPTING 的按钮。单击此按钮后，帮助文档会自动定位到 Transform 脚本引用方法介绍的信息处，如图 1.25 所示。



图 1.25 系统默认的浏览器，自动定位了 Transform 脚本引用方法的帮助信息

这里，展现了特定组件脚本引用方法的帮助信息。这类信息通常有很多。用户并不需要一次将它们全看完，或者要求自己全部理解。那样的话，帮助文档就失去了意义。最好的办法是，只看需要用到的帮助信息！

## 1.4 创建并编辑 C#脚本

在 Unity 中，可以使用 3 种方式创建 C#脚本文件，且新创建的脚本文件会显示在 Project 视图下，如下：

- ☐ 在 Unity 中，单击 Assets|Create|C# Script 命令；
- ☐ 在 Project 视图里，单击 Create|C# Script 命令；
- ☐ 在 Project 视图里，单击鼠标右键，在弹出的快捷菜单中单击 Create|C# Script 命令；

使用上面介绍的 3 种方法中的一种，创建一个 C#脚本文件，重命名为 LearningScript，如图 1.26 所示。

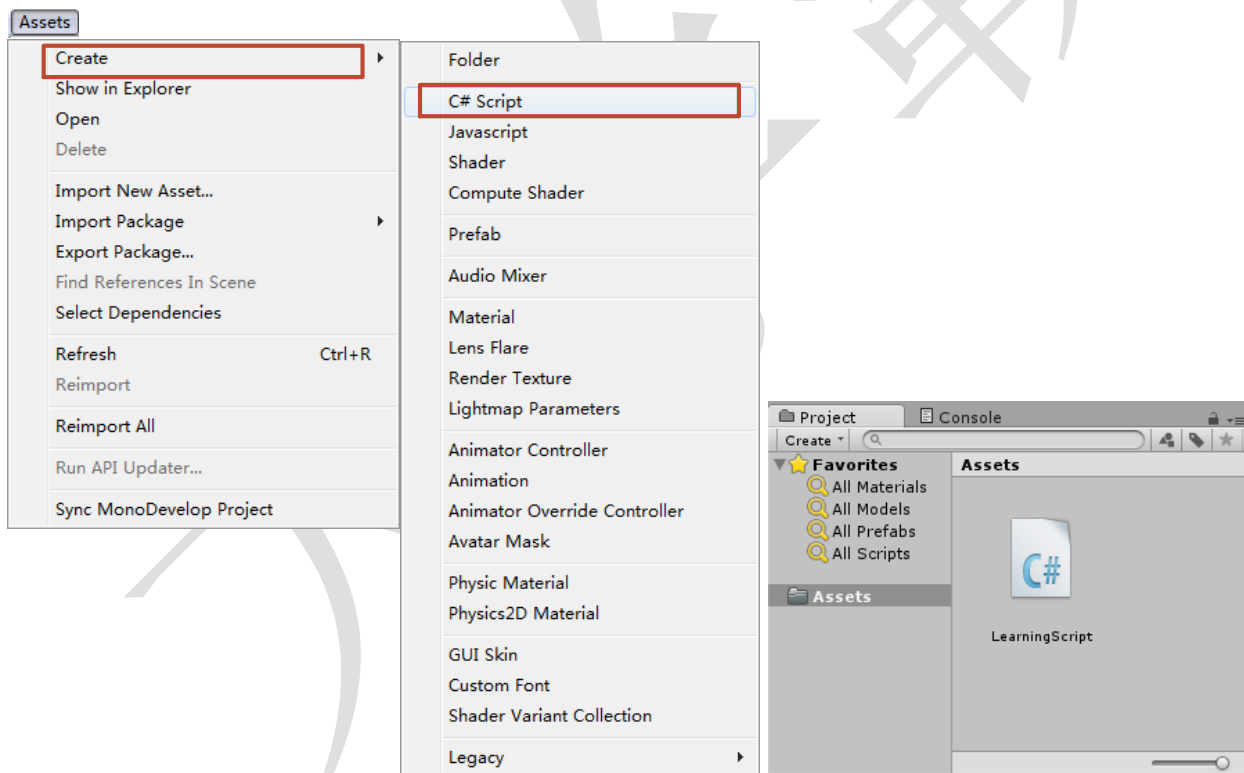


图 1.26 创建 C#脚本文件

当读者需要编辑 C#脚本里的代码时，可以使用鼠标双击脚本文件，然后 Unity 就会使用内置的脚本编辑和编译的软件 MonoDevelop，打开被鼠标双击的脚本。例如，鼠标双击 LearningScript 脚本文件后，出现的 MonoDevelop 软件视图，如图 1.27 所示。

打开 C#脚本文件以后，Unity 已经自动在脚本文件中添加了一些代码，如下所示：

```
01 using UnityEngine;
02 using System.Collections;
```

```
03  
04 public class LearningScript : MonoBehaviour  
05 {  
06     // Use this for initialization  
07     void Start ()  
08     {
```

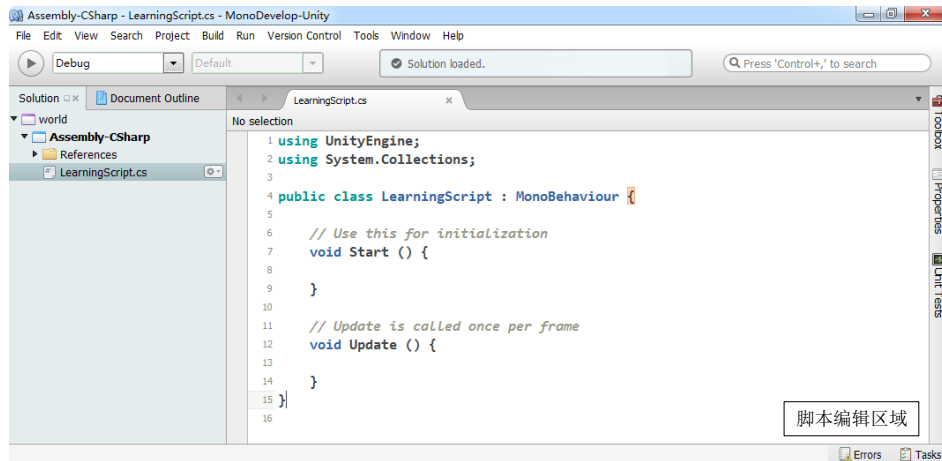


图 1.27 MonoDevelop 软件视图

```
09  
10     }  
11     // Update is called once per frame  
12     void Update ()  
13     {  
14  
15     }  
16 }
```

对于 Unity 自动添加的代码，此时读者只需要知道，脚本 04 行的 LearningScript 是与脚本文件名一致的。

注意：在 Unity 中，这两处的字符串必须一致，否则会在以后导致一些错误的出现。对于初学者而言，导致这两处字符串不一致的原因最可能是，新建了一个 C#脚本文件，并使用了默认的文件名 NewBehaviourScript。然后由于其它一些原因，读者在 Project 视图里修改了这个脚本文件的文件名，于是两处的字符串就不一致了，如图 1.28 所示。所以，当读者要修改脚本文件名的时候，一定要记得修改脚本代码中对应的字符串，反之亦然。



图 1.28 名称不一致

提示：Unity 和 MonoDevelop 是两个独立的软件。C#脚本文件将它们联系到了一起。也就是说，后者主要编辑 C#脚本文件，而前者主要使用 C#脚本文件。这就产生了一个“同步”的问题，MonoDevelop 中编辑的代码，必须保存，然后 Unity 才会知道脚本文件发生了改变。

脚本中的代码全部都是 Unity 添加的，即时运行也不会有任何效果。那么，至于在脚本中如何编写 C#代码，以及 C#代码在游戏中的效果和作用，会在下一章中介绍。而在进入下一章的学习前，用户需要多熟悉熟悉 Unity 开发环境。

## 第 2 章 构建第一个游戏

使用 Unity 创建游戏的过程分为三个阶段，分别为构建游戏场景，设置游戏对象状态，添加脚本。本章将会针对创建游戏的每一步过程做出简单说明，并在最后以一个十分简单的游戏收尾。相信大家在完成本章的学习后，就可以掌握构建游戏的方法，并了解其中涉及到的各种概念和专业术语。

### 2.1 构建游戏场景

如果已经计划好了要编写什么样的游戏，在打开 Unity 以后，要做的第一件事情就是构建游戏场景（Scene）。游戏场景就是玩家游戏时，在游戏视图中看到的一切，包括游戏背景、游戏角色、阳光等等。在 Unity 中，这些所有可视的元素被统称为游戏对象（Game Object）。本节就先来学习游戏场景的构建。

#### 2.1.1 新建游戏场景

类似于创建游戏就需要新建游戏项目一样，构建游戏场景也是需要新建游戏场景的。只不过，Unity 在开发者新建游戏项目的时候，已经默认创建了一个游戏场景，但是此游戏场景还没有被保存成具体的文件。保存过程如下：

（1）单击 File|Save Scene 命令，或者按下键盘上的快捷键 Ctrl+s，将弹出 Save Scene 对话框，如图 2.1 所示。

（2）在“文件名”文本框中输入文件名，单击“保存”按钮，就可以保存该场景。

保存后的场景，可以在 Project 视图的 Assets 面板看到，如图 1.2 所示。

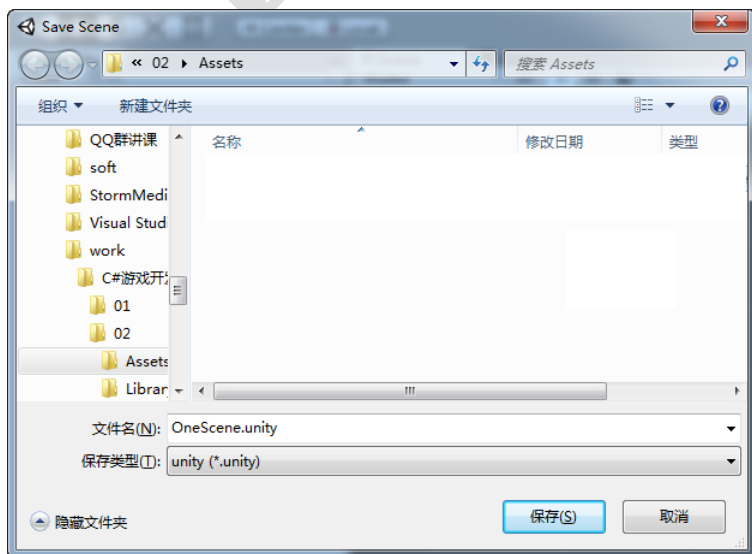


图 2.1 Sava Scene 对话框

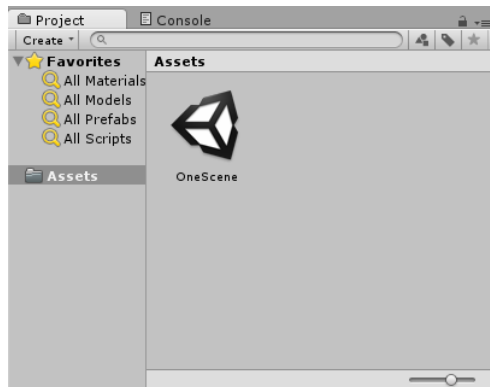


图 2.2 保存的场景文件

## 2.1.2 添加游戏对象

使用 Unity 新建的游戏场景都默认包含两个游戏对象 Main Camera 和 Directional Light，如图 2.3 所示。除了这两个游戏对象，用户还可以根据需要添加更多的游戏对象。

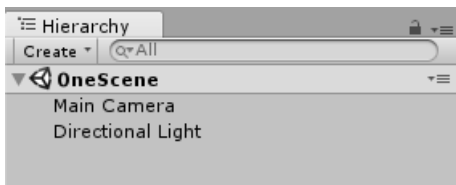


图 2.3 默认包含的游戏对象

在 Unity 中，为游戏场景添加游戏对象的方法有很多。下面讲解常见的两种方式。

- ❑ 在 Unity 中，单击 **GameObject** 命令，在弹出的命令菜单中选择要添加的游戏对象类型，如图 2.4 所示。
- ❑ 在 **Hierarchy** 视图里，单击左上角的 **Create** 按钮，从弹出的命令菜单中选择要添加的游戏对象类型，如图 2.4 所示。

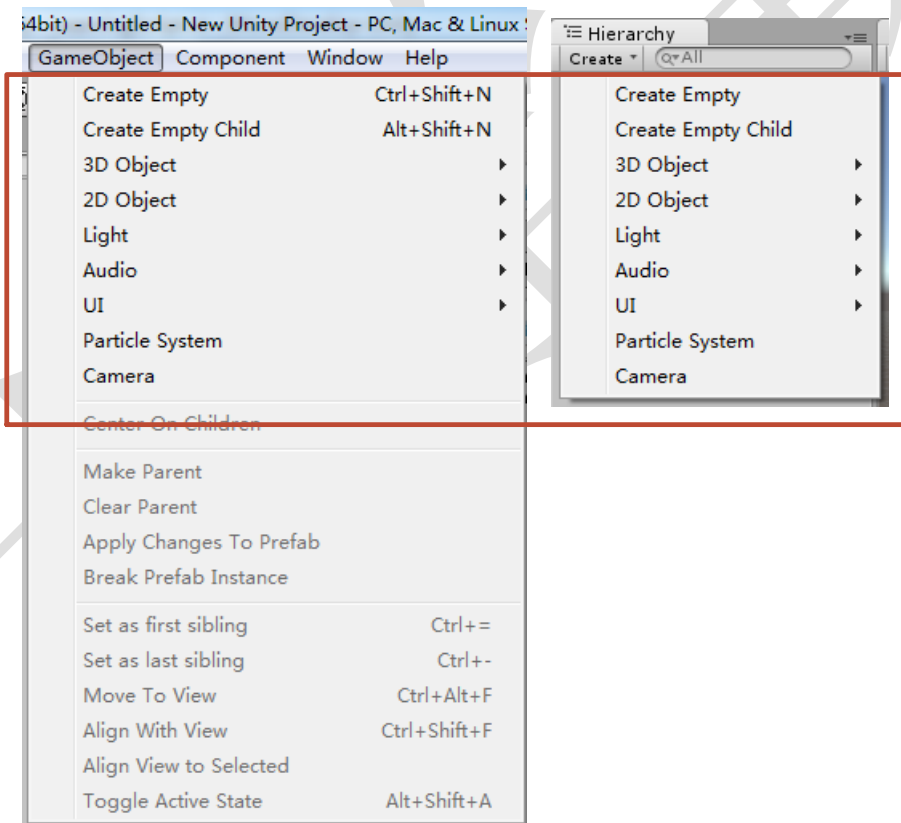


图 2.4 添加游戏对象

下面在游戏场景中添加一个立方体和平面。操作过程如下：

- (1) 单击 **GameObject|3D Object|Cube** 命令，向场景中添加一个立方体。
- (2) 单击 **GameObject|3D Object|Plane** 命令，向场景中添加一个平面。

操作结束后，游戏场景中总共包含 4 个游戏对象了，如图 2.5 所示。同时这些游戏对象都可以在

Hierarchy 视图中看到，如图 2.6 所示。

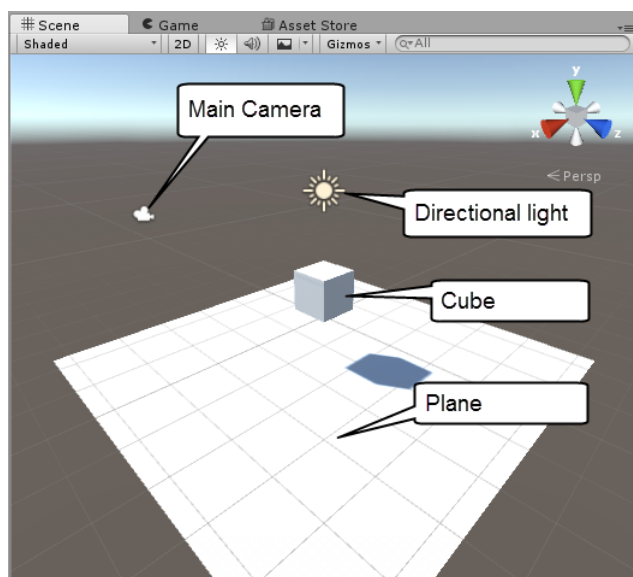


图 2.5 游戏场景



图 2.6 Hierarchy 视图

新添加的游戏对象都以默认类型名命名。当同类型的游戏对象，会造成对象名重复。这样，会不利于分辨游戏对象。这时，用户可以在 Hierarchy 视图中，右击游戏对象名，单击 Rename 命令，对游戏对象重新命名，操作如图 2.7 所示。

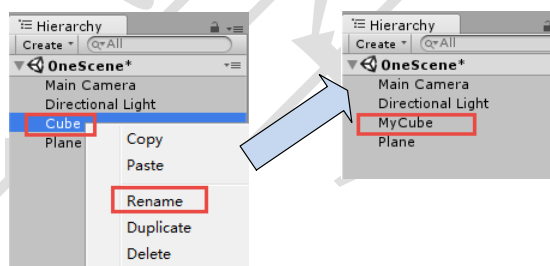


图 2.7 为游戏对象重命名

## 2.2 改变游戏对象状态

游戏对象状态是指游戏对象的位置、朝向和大小等因素。新创建的游戏对象都采用了 Unity 的默认设置。用户需要根据游戏开发实际的需要，改变这些游戏对象的状态。本节讲解游戏对象状态的三种修改方式。

### 2.2.1 使用 Main Camera 与 Game 视图

在前面的讲解中，我们都是通过 Scene 视图来查看游戏场景中的各个对象。但实际做出的游戏场景并不是 Scene 视图里看到的样子。用户需要单击 Scene 视图标签旁边的 Game 标签，切换到 Game 视图中。Game 视图展现才是最终的游戏场景效果。两个视图效果对比如图 2.8 所示。

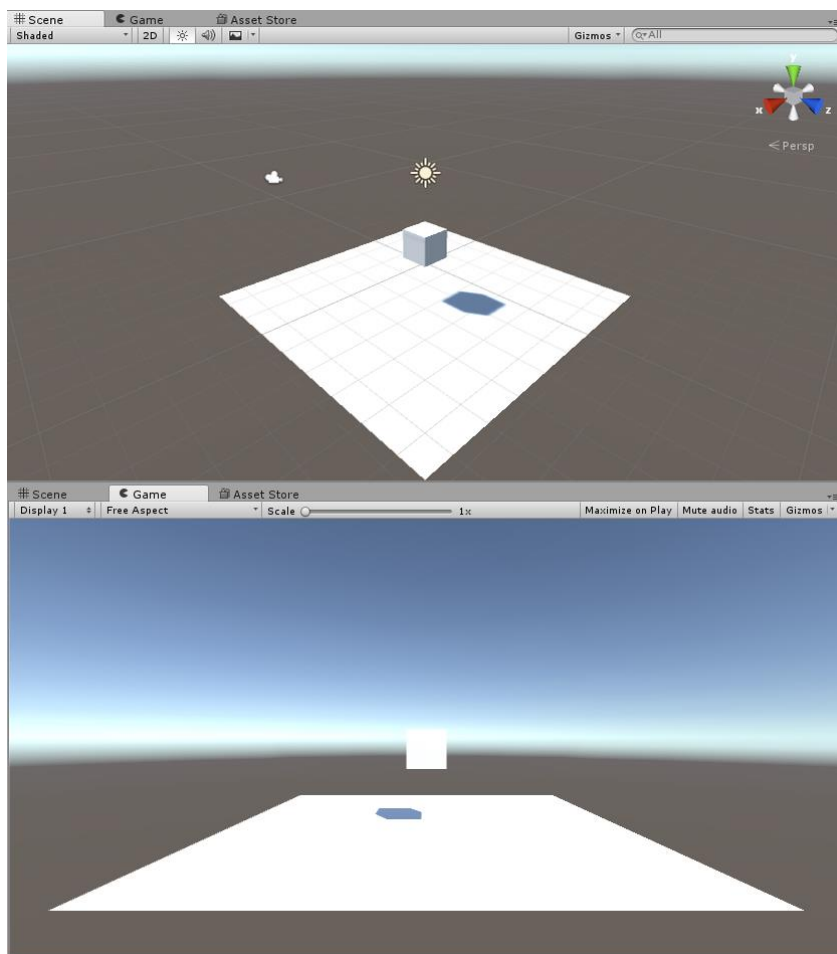


图 2.8 Scene、Game 视图效果对比

在 Game 视图中,用户是无法操作场景中的游戏对象。因为它只是作为效果展示视图而存在。在 Scene 视图中,开发者以旁观的角度来查看游戏场景和游戏对象。而 Game 视图则是以 Main Camera 的视角来查看游戏场景和游戏对象。

Main Camera 是场景默认创建的游戏对象。它的作用提供一个视角来展现游戏场景。在 Scene 视图中,用鼠标单击 Main Camera,选中该游戏对象,就会出现一个 Camera Preview 面板,如图 2.9 所示。该面板展现的就是 Game 视图的效果。当拖动游戏对象 Main Camera, Camera Preview 面板的场景就会发生改变。



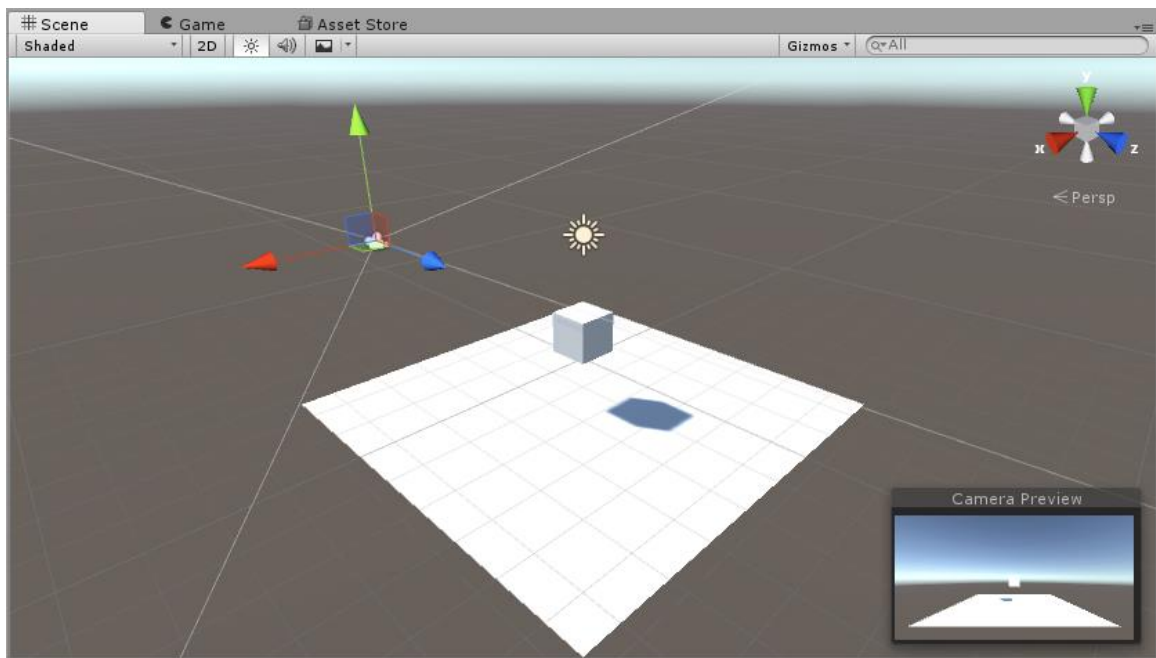


图 2.9 选中 Main Camera

通过拖动游戏对象 Main Camera 的方式，只是被动地改变游戏对象的状态。下面讲解如何直接修改游戏对象。

## 2.2.2 使用 Transform 组件

在 Unity 中，每个游戏对象都有很多属性，这些属性决定了对象位置、朝向、大小等状态。为了方便用户管理，将这些属性按照类别归并为组件。例如，游戏对象的位置、朝向、大小属性就被归并为 Transform 组件，如图 2.10 所示。

在 Transform 组件中，用户可以采用数值的方式修改游戏对象的属性。操作过程如下：

(1) 指定修改的游戏对象，指定的方式有两种：

- ☐ 直接使用鼠标点选 Scene 视图中要改变状态的游戏对象；
- ☐ 直接使用鼠标点选 Hierarchy 视图中要改变状态的游戏对象的名称；

被选中的游戏对象的所有可设置的属性显示 Inspector 视图中，如图 2.11 所示。

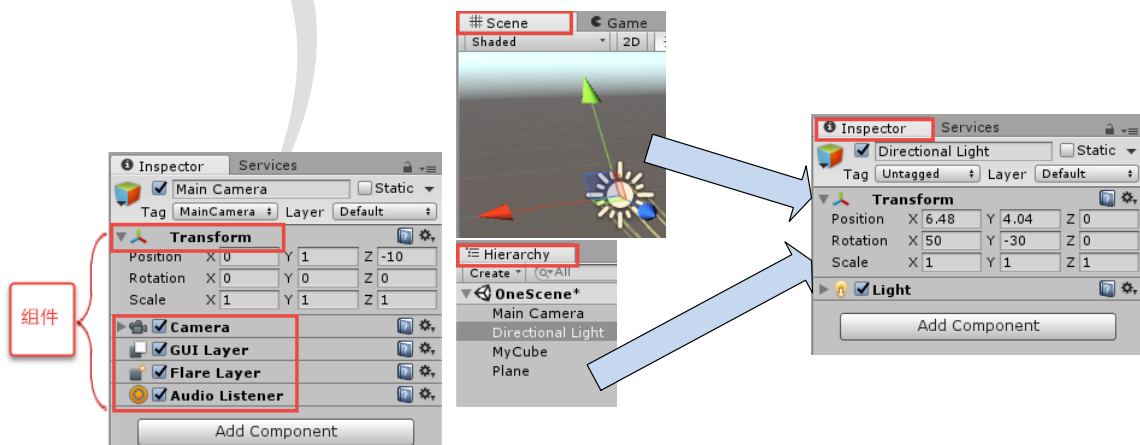


图 2.10 Transform 组件

图 2.11 Inspector 视图

(2) 在 Transform 组件下, Position 用于设置位置, Rotation 用于设置朝向, Scale 用于设置大小。对属性的修改会实时的反应在 Scene、Game 视图中。

### 2.2.3 直接操作游戏对象

在 Inspector 视图里, 设置属性而改变游戏场景中游戏对象状态的方式虽然很精确, 但比较抽象, 毕竟数字并不够直观。修改游戏对象的状态, 最直观、最简单的方法就是在 Scene 视图里, 直接使用鼠标操作游戏对象。

在 Unity 的左上角, 有个工具栏, 它显示了 5 个按钮, 如图 2.12 所示。前 4 个分别表示拖动 Scene 视图、改变游戏对象的位置、朝向和大小, 最后一个则专用于操作 2D 游戏对象的位置、朝向和大小。



图 2.12 Unity 左上角的 4 个工具栏按钮

要改变游戏对象的位置, 可以先使用鼠标选中工具栏的第二个按钮, 然后点选 Scene 视图中的指定对象。被点选的对象上会出现 3 个颜色的坐标轴, 每个坐标轴的交点处还有一个小矩形, 如图 2.13 所示。

- 使用鼠标点击坐标轴并拖动的话, 游戏对象就会朝着坐标轴的方向移动;
- 使用鼠标点击小方块并拖动的话, 游戏对象就会在组成小方块的两个坐标轴构成的平面上移动;

要改变游戏对象的朝向和大小, 可以分别选择第 3、4 个按钮, 然后选中指定的游戏对象, 最后操作辅助线即可, 如图 2.14 所示。

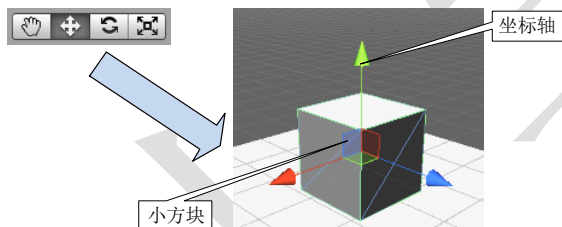


图 2.13 移动 Cube 对象

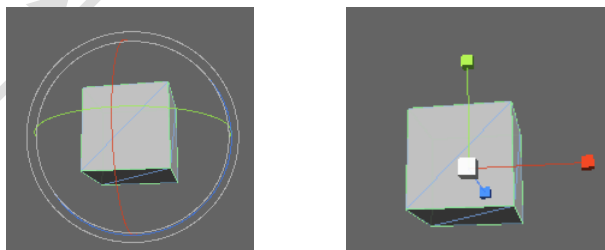


图 2.14 改变游戏对象的朝向和大小

本小节讲解了在 Scene 视图里, 改变 Cube 对象状态的方法, 这样的方法对于所有的对象都是适用的, 当然也包括当前游戏场景中的 Plane 对象, 如图 2.15 所示。

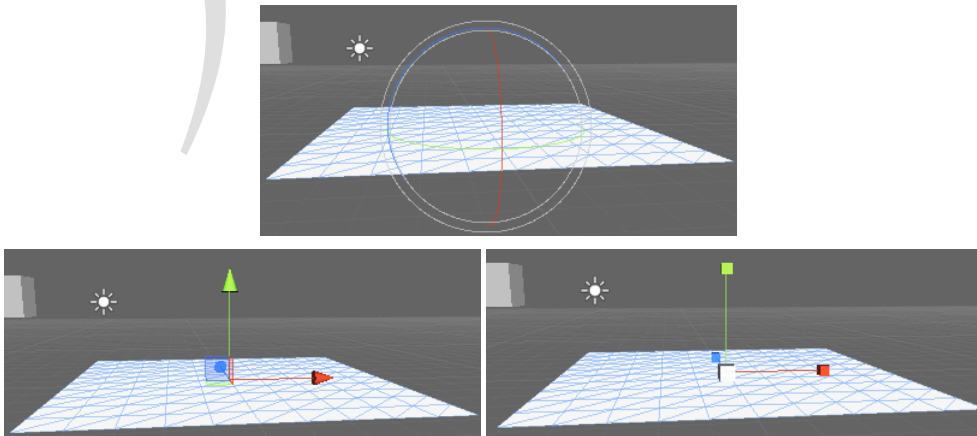


图 2.15 改变 Plane 对象的状态

## 2.3 Scene 视图的快捷操作

Scene 视图是开发者开发游戏时，操作最频繁的视图。因为一旦一个游戏对象被添加到游戏的场景中，就需要首先使用鼠标为这个游戏对象设置出合适的状态。而且开发者还需要多角度的观察游戏场景中的各游戏对象。基于以上的原因，Unity 提供了很多快捷操作，支持开发者对 Scene 视图所做的各种操作，常见的操作方式有：

- ❑ 直接按下键盘上的 Q、W、E、R 键，即可选中 Unity 左上角，工具栏上的 4 个按钮，且按钮与按键一一对应，省去了开发者使用鼠标点击的麻烦。
- ❑ 使用鼠标的滚轮，可以控制 Scene 视图的缩放，如图 2.16 所示，且向上滚动则放大，向下滚动则缩小。

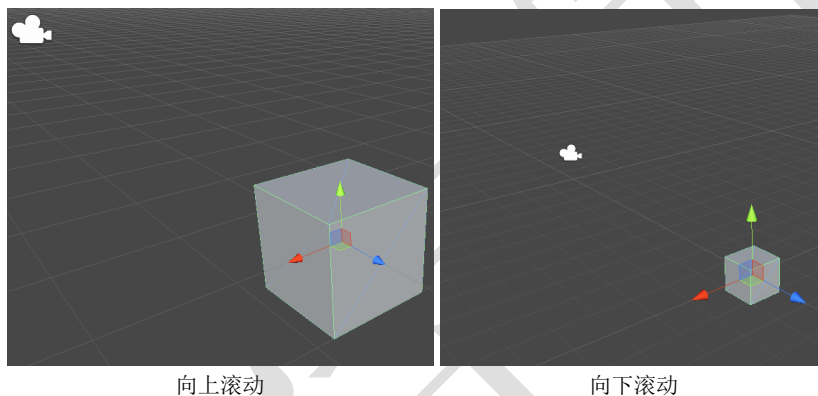


图 2.16 缩放 Scene 视图

- ❑ 使用鼠标双击 Hierarchy 视图上的游戏对象名，场景视图会移动，直到对应的对象处于场景视图的中间，如图 2.17 所示。当开发者在场景中找到对应的游戏对象时，可以使用这种方法快速找到。

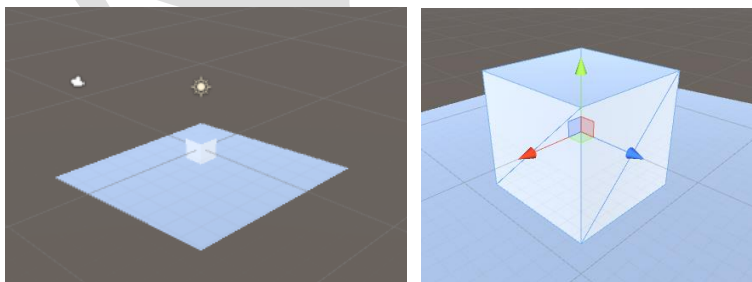




图 2.17 快速定位

- ❑ 在场景视图里，按下鼠标中间的按钮（或滚轮），鼠标变成了 ，然后移动鼠标，可以任意移动场景，如图 2.18 所示。其作用等同于工具栏上的第一个按钮。
- ❑ 在场景视图里，按下键盘上的 Alt 键，鼠标就会变成 ，再按下鼠标的左键拖动，可围绕指定

的游戏对象移动，如图 2.19 所示。

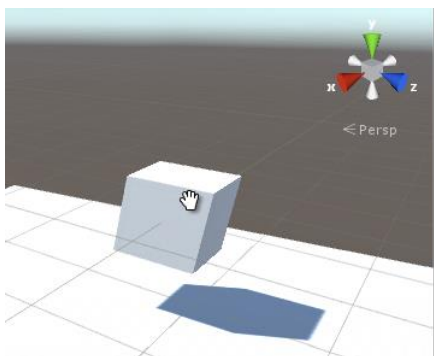


图 2.18 任意移动场景视图中的场景

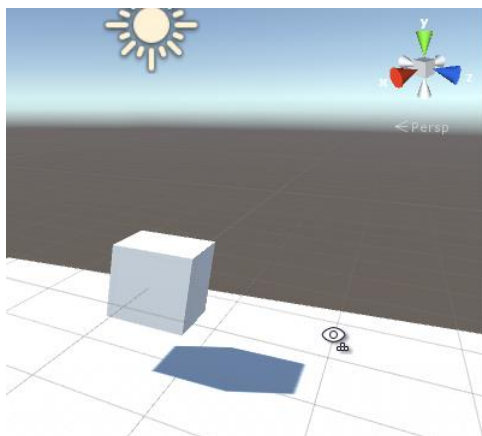



图 2.19 绕着场景视图中的对象移动

- ❑ 在场景视图里，按下鼠标的右键，鼠标就会变成 ，然后再按下键盘上的“W、S、A、D”键，就可以模拟第一人称在场景中向前、后、左、右移动，移动鼠标相当于转动人物的头部。如图 2.20 所示。
- ❑ 读者一定注意到，在 Scene 视图的右上角有个坐标轴的模型，在 Unity 中它被称为 Gizmo，如图 2.21 所示。使用它可以将 Scene 视图，迅速切换到预定义的观察视角，具体的操作是单击 Gizmo 的轴。如图 2.22，切换了三个视角来查看 Scene 视图中的游戏对象。

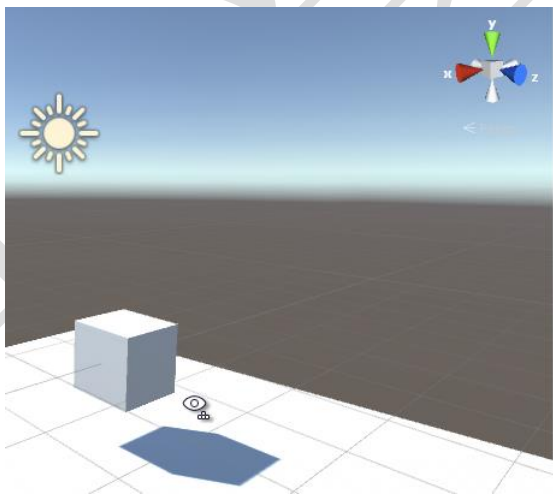


图 2.20 模拟第一人称，在场景中移动



图 2.21 场景视图中的 Gizmo 工具

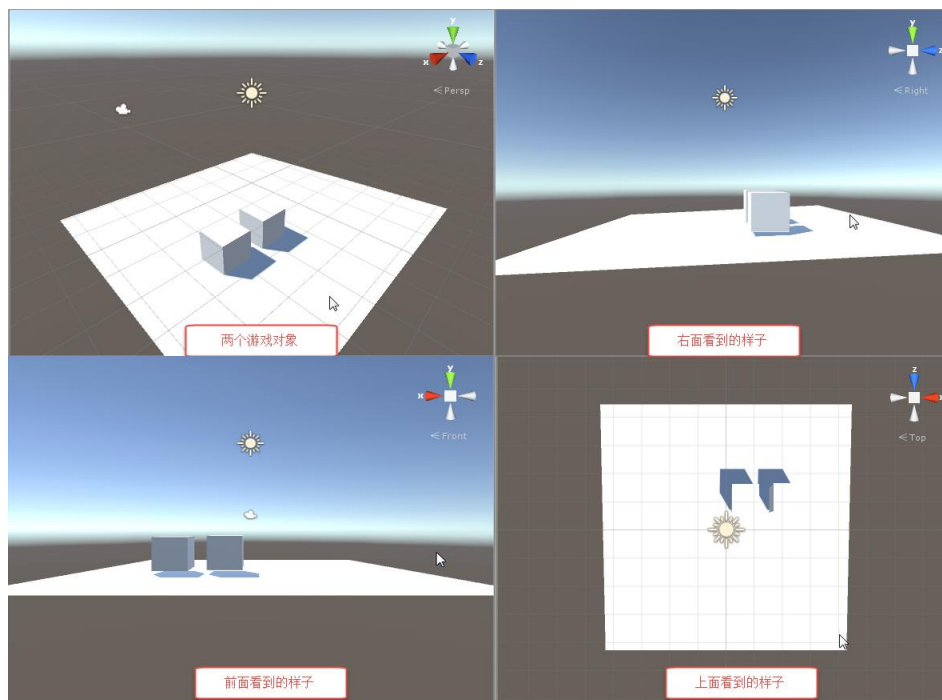


图 2.22 使用不同的视角查看视图中的游戏对象

## 2.4 使用脚本

使用本章前面介绍的方法，用户可以手动修改游戏场景中的游戏对象的状态。但是在游戏中，往往需要由游戏动态地修改游戏对象的状态。例如，玩家发出跑步指令，游戏中的对象就需要改变位置。这就需要使用脚本来控制游戏对象的状态。下面讲解使用脚本控制游戏对象的方式。

### 2.4.1 示例效果展示

下面是一个示例，它会让场景中的立方体边旋转边移动，而且在开始移动前还会改变自己的大小！是不是很神奇？下面是让立方体旋转、移动的脚本的全部代码：

```
01 using UnityEngine;
02 using System.Collections;
03
04 public class MyTransform : MonoBehaviour
05 {
06     public float scaleValue = 1.5f;
07     public float xPosition = 0.1f;
08     public int yRotation = 10;
09     // Use this for initialization
10     void Start ()
11     {
12         transform.localScale = new Vector3(1,scaleValue,1);
13     }
14     // Update is called once per frame
15     void Update ()
```

```
16    {  
17        transform.Translate ( new Vector3(xPosition,0,0),Space.World);  
18        transform.Rotate (Vector3.up * yRotation,Space.World);  
19    }  
20 }
```

这个脚本让立方体产生的运动效果如图 2.23 所示。

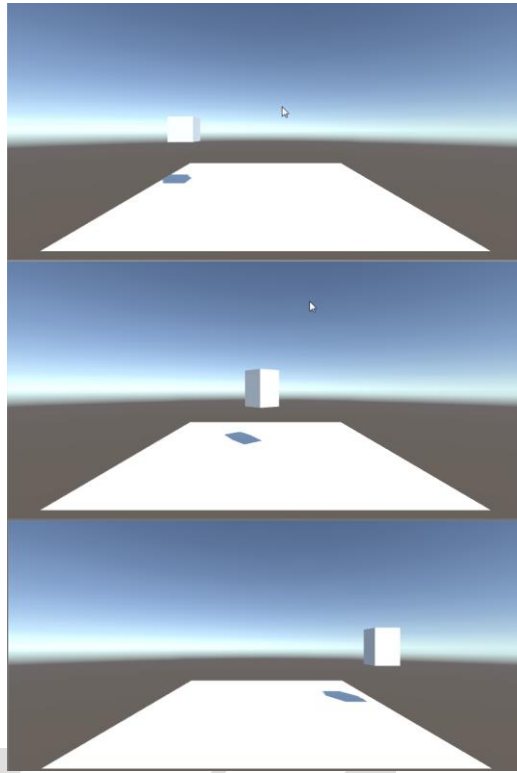


图 2.23 运行效果

为什么脚本可以改变立方体的状态？实际上，它改变立方体状态的方法与我们改变立方体状态的方法是一样的。它也是通过修改立方体上 Transform 组件的属性，来改变游戏对象的状态。

## 2.4.2 脚本的构成

所有脚本中的代码，实际上定义的是一个与脚本同名的类（class）。而类里又定义了变量（variable）和方法（method），如图 2.24 所示。

```

using UnityEngine;
using System.Collections;

public class MyTransform : MonoBehaviour
{
    public float scaleValue = 1.5f;
    public float xPosition = 0.1f;
    public int yRotation = 10;
    // Use this for initialization
    void Start ()
    {
        transform.localScale = new Vector3(1,scaleValue,1);
    }
    // Update is called once per frame
    void Update ()
    {
        transform.Translate ( new Vector3(xPosition,0,0),Space.World);
        transform.Rotate (Vector3.up * yRotation,Space.World);
    }
}

```

与脚本名同名的类名

变量

方法

图 2.24 脚本的各组成部分

提示：脚本代码中，以双斜杠开始的代码行是注释。这是为了说明代码的作用而添加的，对脚本本身不产生任何影响。

## 1.类

在前面讲解过，组件是将游戏对象的多个属性归并到一起。实际组件的本质就是类。组件与类只是不同环境下，对同一东西的不同的称呼而已。将脚本和游戏对象关联起来，最后以组件的形式成为了游戏对象的一部分，如图 2.25 所示。

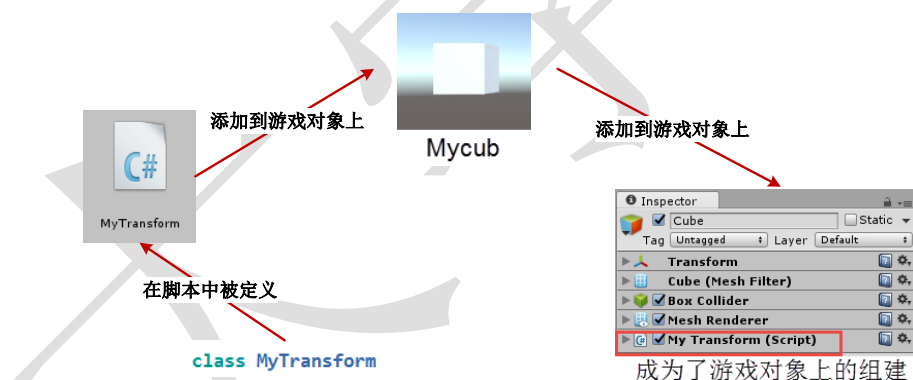


图 2.25 类、脚本、组件

## 2.变量

脚本中的变量，是用来存储数据的，读者可以把它看作是容器。其实，组件下的属性本质上也是变量，同样是同一事物在不同场景下的不同称呼而已，如图 2.26 所示。展开成为组件的脚本，就可以看到其下的属性名与变量名基本一致，只是字母的大小写可能有所不同。



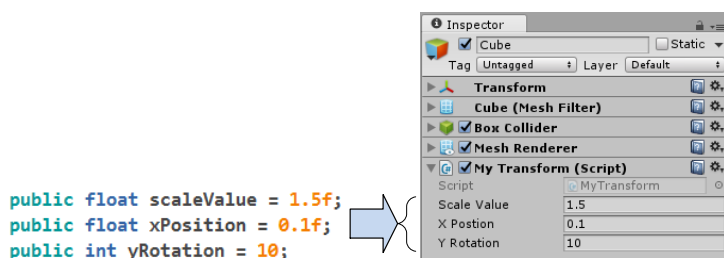


图 2.26 变量、属性

变量中存储的数据，成为了组件属性默认设置的值。

### 3.方法

脚本中的方法，用于完成特定的操作或者任务，例如，本节的示例中，让立方体变大、移动和旋转，就是脚本中由方法完成的任务，如图 2.27 所示。

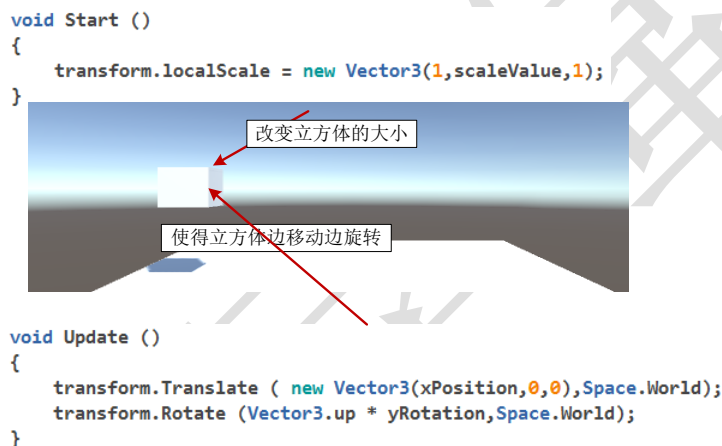


图 2.27 方法的作用

#### 2.4.3 将脚本赋予游戏对象的方法

在 Unity 中，将脚本和游戏对象关联的方法有多种，本小节介绍两种最常见的方法，如图 2.28 所示。

- ❑ 使用鼠标，直接将脚本从 Project 视图拖动到 Hierarchy 视图里指定的游戏对象上；
- ❑ 先选中 Hierarchy 视图里指定的游戏对象，Inspector 视图会显示此游戏对象的所有组件，使用鼠标，将脚本从 Project 视图拖动到 Inspector 视图即可；

最后通过查看 Inspector 视图里，是否出现了与脚本同名的组件，就可以验证脚本是否被成功的赋予了游戏对象。

对于本章的游戏示例，脚本 MyTransform 被赋予了 Cube 对象（名为 MyCube）。而在游戏运行的过程中，Cube 对象会发生状态的改变，则是因为脚本代码修改了 Cube 对象上 Transform 组件的属性，进而实现了改变游戏对象状态的效果。



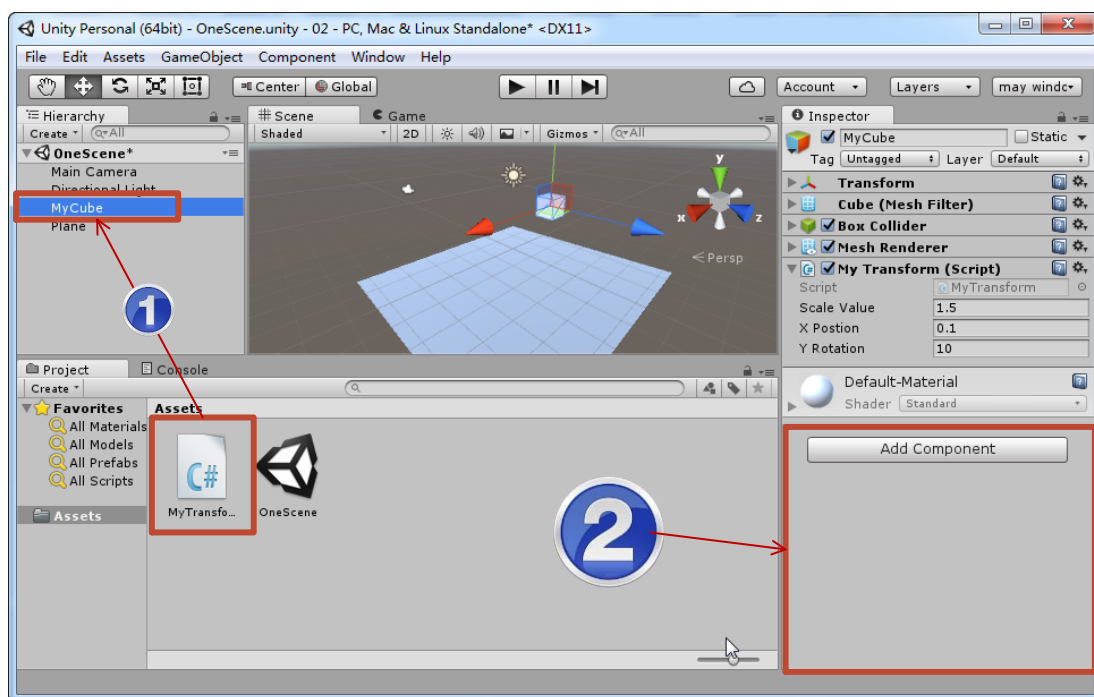


图 2.28 将脚本赋予游戏对象的两种方法

## 2.4.4 运行游戏

Unity 上方中间的部分，在工具栏的位置处有 3 个按钮，如图 2.29 所示，作用依次是运行游戏、中止游戏和单步运行游戏。读者可以单击对应的按钮，来决定运行游戏的方式。



图 2.29 控制游戏运行的 3 个工具栏按钮

## 2.5 小结

本章的内容可以分为三大部分。第一部分是构建游戏场景，读者可以从中了解到游戏场景的操作，以及添加游戏对象的方法；第二部分是改变游戏对象的状态，即游戏对象的位置、朝向和大小，借此了解了 Transform 组件；第三部分是使用脚本，脚本免去了读者手动改变游戏对象状态的麻烦，也使得项目有了游戏的样子，借此了解了脚本的各组成部分，即类、变量和方法，以及脚本各组成部分与游戏对象的关系。