UNREAL
ENGINE

Unreal Engine 4

张宝荣◎编著

学习总动员

C++编程

中国铁道出版社有限公司
CHINA RAILWAY PUBLISHING HOUSE CO., LTD.

Unreal Engine 4
学习总动员
C++编程

张宝荣◎编著

中国铁道出版社有限公司
CHINA RAILWAY PUBLISHING HOUSE CO., LTD.

内 容 简 介

由于Unreal Engine 4全开源代码，使具有C++程序编程经验的人员拥有了天然使用Unreal Engine 4的条件。但是Unreal Engine 4中的C++拥有自己的规则，还需要全面了解Unreal Engine 4中的代码规则。本书全面介绍了Unreal Engine 4中的C++编程技巧和规则。具体内容包括C++编程入门、Unreal Engine 4引擎架构、系统指南、资源处理等。本书通过专业的Unreal Engine 4 C++编程教程实战操作，详细展现了C++编程的方法和技巧。

配套资源中提供了Unreal Engine 4 C++编程视频教学，使读者快速详细了解Unreal Engine 4 C ++编程技巧。

本书适合Unreal Engine初级用户阅读学习，可作为游戏开发、虚拟现实开发相关行业从业人员的参考书，也可作为大中专院校和社会培训机构相关专业的教材。

图书在版编目（CIP）数据

Unreal Engine 4学习总动员. C++编程/张宝荣编著. —北京：中国铁道出版社有限公司，2019.7
ISBN 978-7-113-25779-8

Ⅰ.①U… Ⅱ.①张… Ⅲ.①虚拟现实—程序设计②C++语言—程序设计 Ⅳ. ①TP391.98②TP312.8

中国版本图书馆CIP数据核字(2019)第091688号

书　　名：Unreal Engine 4 学习总动员——C++ 编程
作　　者：张宝荣

责任编辑：于先军　　读者热线电话：010-63560056
责任印制：赵星辰　　封面设计：MXX DESIGN STUDIO

出版发行：中国铁道出版社有限公司（100054，北京市西城区右安门西街 8 号）
印　　刷：北京米开朗优威印刷有限公司
版　　次：2019 年 7 月第 1 版　2019 年 7 月第 1 次印刷
开　　本：787 mm×1 092 mm　1/16　印张：15.75　字数：380 千
书　　号：ISBN 978-7-113-25779-8
定　　价：99.00 元

配套资源下载地址：

http://www.m.crphdm.com/2019/0515/14086.shtml

放眼全球，纵观当今的时代，数字化、信息化、网络化是我们人类发展不可逆转的趋势。随着下一代互联网 IPV6 以及 5G 通信标准的逐步应用，人们的工作、生活、娱乐等领域都将发生革命性的变化。这其中以“虚拟现实”“人工智能”“大数据应用”“数据安全”等领域最为突出。在可以预见的未来，上述 4 个领域将引领世界科技发展潮流。

Epic Games 公司成立于 1991 年，公司总部位于美国北卡罗来纳州卡里镇，在美国、欧洲、日本、中国和韩国等国家和地区设有工作室。Epic Games 的作品包括《Unreal》（虚幻系列游戏）、《Gears of War》（战争机器）、《Infinity Blade》（无尽之剑）、《Paragon》（虚幻争霸）、《Fortnite》（堡垒之夜）、《SPYJiNX》（特工金克斯）、《BattleBreakers》（战争破坏者）、《Robo Recall》（机械重装），以及新的《Unreal Tournament》（虚幻竞技场）。1998 年随着《Unreal》（虚幻系列游戏）的推出，公司随即将开发这款游戏的工具也一并推出，供全球的游戏制作玩家免费使用，由此标志着 Unreal Engine 的正式诞生。

2014 年 Epic Games 公司推出了 Unreal Engine 4（虚幻引擎 4，简称 UE4，本书在不作特别说明时，都简称为 UE4）版本，并且将其源代码全部公开。UE4 进行了全新的渲染引擎升级，从而大大提升了渲染质量和速度。

UE4 是一套为使用实时技术的人士开发的完整开发工具。从企业应用和电影体验到高品质的 PC、主机、移动、VR 及 AR 游戏，UE4 都能为用户提供从启动项目到发行产品所需的一切，在同类产品中独树一帜。UE4 提供了强大的工具套件以及简易的工作流程，能够帮助开发者快速迭代概念并立即查看成品效果，且无须触碰一行代码。而完整公开的源代码则能让 UE4 社区的所有成员都能够自由修改和扩展引擎功能。

UE4 官方发布了许多的视频教程和在线帮助文档，以供用户学习和使用。另外，还公布

了大量的游戏制作项目工程，免费供全球用户使用。由于各种原因，国内目前关于 UE4 方面的学习资源极为稀少。鉴于此，十分有必要推出一套全面介绍 UE4 技术内容的丛书，以供国内用户学习和使用。

本系列图书全面介绍了 UE4 的全部内容。丛书共有 6 本，分为快速入门、材质渲染、蓝图应用、动画设计、游戏开发、C++ 编程。内容包含了 UE4 的全部模块内容。本套图书具有鲜明的特色，首先，整套图书以案例教程为核心，每本书有数十个案例教程。手把手教会你快速上手 UE4，使学习 UE4 变得极为容易，完全以实战操作为成书标准。其次，整套图书配有近 18 小时的语音视频教程，完全是精典案例实战操作式教学。最后，本套图书配有巨量的工程数据文件，以供读者非常方便地调用和查看。

由于编写这套图书工作量巨大，加之 UE4 更新较快，书中难免有不足和谬误之处，欢迎广大读者批评斧正。该套图书在开发过程中得到了 Epic Games 公司和许多业内人士的大力支持和帮助，在此特别表示感谢。

作　者

2019 年 6 月

配套资源下载地址：

http://www.m.crphdm.com/2019/0515/14086.shtml

技术支持QQ群：596664789

配套资源下载地址：
http://www.m.crphdm.com/2019/0515/14086.shtml

目 录

第 1 章

C++ 编程入门

1.1 编程入门指南

在本节我们会创建新的 UE4 项目，向其添加新的 C++ 类，然后编译项目并添加新类的实例到关卡中。在完成后，我们会看到以 C++ 来编程的 Actor 在屏幕上的移动。

1.1.1 必备的项目设置

项目设置的具体操作如下。

Step 1 从启动程序中打开 UE4，项目浏览器将出现。

Step 2 单击“New Project”（新建项目）选项卡，然后选择“C++”选项卡。在该处选择“Basic Code”（基础代码），这样我们可以获得全新的起始点确认设置“With Starter Content”（和入门内容共同启动）后，我们需要输入项目名称，就使用“QuickStart”吧。现在可以单击“Create Project”（创建项目）并开始了，如图 1-1 所示。

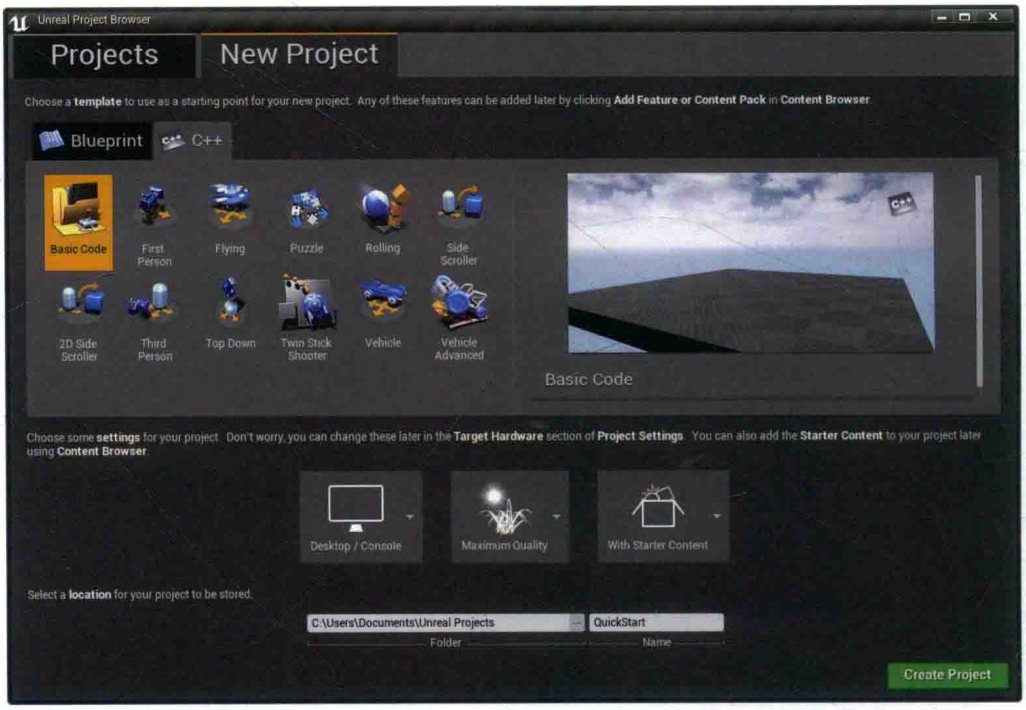

图 1-1

Unreal Editor（虚幻编辑器）现在会打开新项目，Visual Studio 也会打开，并载入项目刚创建的解决方案文件。

1.1.2 创建 C++ 类

创建 C++ 类的具体操作如下。

Step 1 在 Unreal Editor（虚幻编辑器）中，我们可以使用“Add Code to Project”（添加代码到项目）命令来创建新的 C++ 类，它位于“File”（文件）下拉菜单中，如图 1-2 所示。

Step 2　此时将打开“Choose Parent Class”（选择父类）菜单。由于 Actor 是能够存在于 UE4 层面的最为基础的类，我们将会把 Actor 类作为基类，如图 1-3 所示。

图 1-2

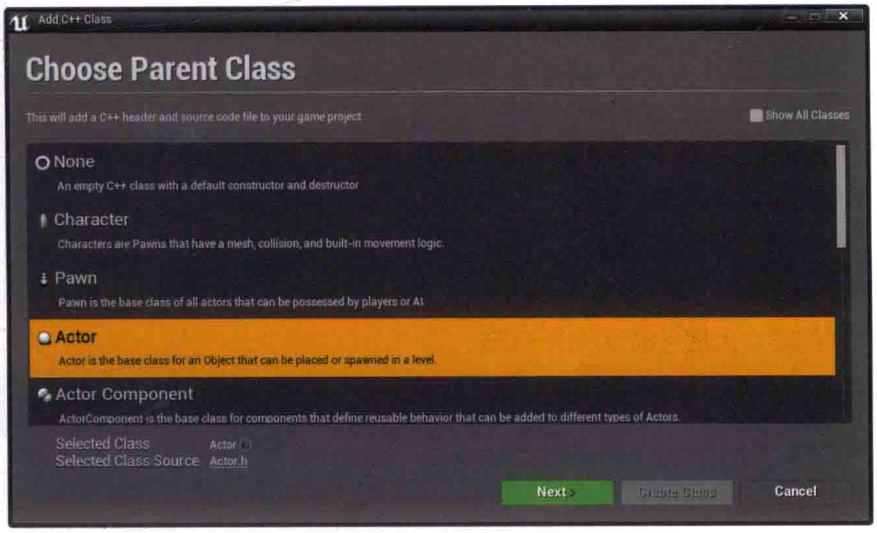

图 1-3

Step 3　此时将打开“Name Your New Actor”（命名你的新 Actor）菜单。对这个示例来说，让我们输入名称“FloatingActor”，然后单击“Create Class”（创建类），如图 1-4 所示。

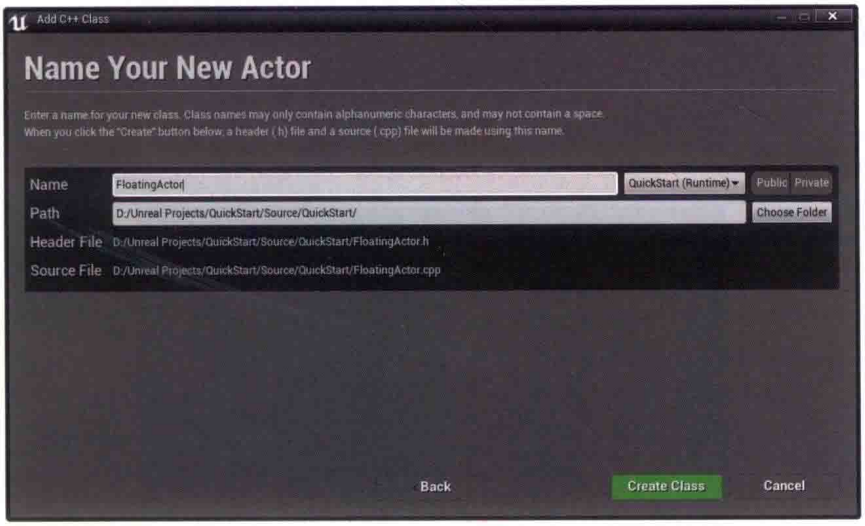

图 1-4

现在我们已经创建了一个 C++ 类，我们可以切换到“Visual Studio”来对它编程。“FloatingActor.cpp”将自动打开，UE4 会自动编译并使用我们的新类来重新载入代码。

1.1.3 写入并编译 C++ 代码

编译 C++ 代码的具体操作如下。

Step 1 在“Visual Studio”中，我们会使用“Solution Explorer”（解决方案浏览器）面板来搜寻新建的 C++ 文件。在我们的示例中，它们被命名为“FloatingActor.cpp 和 FloatingActor.h”，并且将被放置于“QuickStart”项目中，如图 1-5 所示。

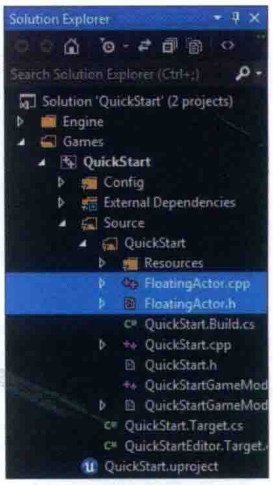

图 1-5

Step 2 在“FloatingActor.h”中，我们会在文件末尾处的终止大括号和分号之前添加以下代码：

```
float RunningTime;
```

Step 3 切换到“FloatingActor.cpp，”我们会在“AFloatingActor::Tick”底部的终止大括号前添加以下代码：

```
FVector NewLocation=GetActorLocation();
float DeltaHeight=(FMath::Sin(RunningTime+DeltaTime)-
FMath::Sin (RunningTime));
NewLocation.Z+=DeltaHeight*20.0f;          //把高度以20的系
数进行缩放
RunningTime+=DeltaTime;
SetActorLocation(NewLocation);
```

我们刚写的代码会导致 FloatingActors 平滑地上下跳动，使用我们创建的 RunningTime 变量来随时间追溯移动的轨迹。

Step 4 现在编码完成了，我们可以通过在“Solution Browser”（解决方案浏览器）中右击项目并选择“Build”（创建）命令，或通过单击 Unreal Editor（虚幻编辑器）的“Compile”（编译）按钮来进行编译。编译成功后，UE4 会自动载入我们的变更内容。在“Visual Studio”中，如图 1-6 所示。

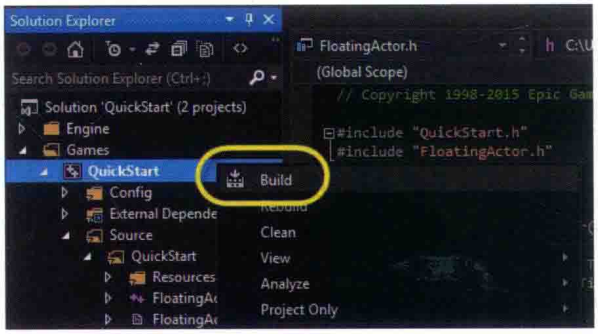

图 1-6

在虚幻编辑器中，如图 1-7 所示。

图 1-7

我们现在可以基于代码来在 UE4 中创建物体了！所有本示例中使用的代码都在下方，以供

你参考。

Finished Code

FloatingActor.h

```
// Copyright 1998-2017 Epic Games,Inc.All Rights Reserved.

#pragma once

#include "GameFramework/Actor.h"
#include "FloatingActor.generated.h"

UCLASS()
class QUICKSTART_API AFloatingActor : public AActor
{
    GENERATED_BODY()

public:
    // 设置此actor属性的默认值
    AFloatingActor();

    // 当游戏开始或生成时调用
    virtual void BeginPlay() override;

    // 在每一帧调用
    virtual void Tick( float DeltaSeconds ) override;

    float RunningTime;
};
```

FloatingActor.cpp

```
//Copyright 1998-2017 Epic Games,Inc.All Rights Reserved.

#include "QuickStart.h"
#include "FloatingActor.h"

// 设置默认值
AFloatingActor::AFloatingActor()
{
     //将此actor设置为在每一帧都调用Tick()。如果你不需要这项功能，你可以关闭它以改善
性能。
    PrimaryActorTick.bCanEverTick=true;

}

//当游戏开始或生成时调用
void AFloatingActor::BeginPlay()
{
    Super::BeginPlay();

}

//在每一帧调用
void AFloatingActor::Tick(float DeltaTime)
```

```
{
    Super::Tick(DeltaTime);

    FVector NewLocation=GetActorLocation();
      float  DeltaHeight=(FMath::Sin(RunningTime+DeltaTime)-
FMath::Sin(RunningTime));
    NewLocation.Z+=DeltaHeight*20.0f;   //把高度以20的系数进行缩放
    RunningTime+=DeltaTime;
    SetActorLocation(NewLocation);
```

1.1.4 测试代码

测试代码的具体操作如下。

Step 1 在虚幻编辑器中，找到“Content Browser”（内容浏览器），并展开名称为“C++类”的文件夹、在该文件夹中，有一个包含了 Actor 类的 FloatingActor 的“QuickStart”文件夹，如图 1-8 所示。

图 1-8

Step 2 我们可以直接把“FloatingActor”类拖动到“Level Editor”（关卡编辑器）窗口来创建世界中 FloatingActor 的实例。我们会在“Level Editor”（关卡编辑器）和“World Outliner”（世界大纲视图）中选择它，它在其中的名称为“FloatingActor1”。它的“Components”（组件）和其他属性可以在“Details”（详细信息）面板中看到，如图 1-9 所示。

Step 3 FloatingActor 应该在游戏中可见。在选择了它后，我们可以在“Details Panel”（详细信息面板）中单击“Add Component”（添加组件），然后选择“Cone”（椎体）从而赋予它简单的可视化表现，如图 1-10 所示。

Step 4 现在自定义的 Actor 已经完成了，让我们把它移动到明显的位置。我们可以用鼠标左键在世界中选择并拖动内容，或者我们也可以手动来移动它。如需手动移动，我们可以在“Level Editor”（关卡编辑器）或“World Outliner”（世界大纲视图）中选择它，然后使用“Details Panel”（详细信息面板）来选择“FloatingActor1”（实例）。我们现在可以直接编辑 FloatingActor1 的 Transform（变换）的 Location（位置）域了。让我们把“X”设置为“-200”，把“Z”设置为“200”。这样我们就可以在场景中的桌子上放置“FloatingActor1”了，如图 1-11 所示。

图 1-9

图 1-10

Step 5　按下“Play”（播放）按钮，然后观看椎体的上下浮动，如图 1-12 所示。

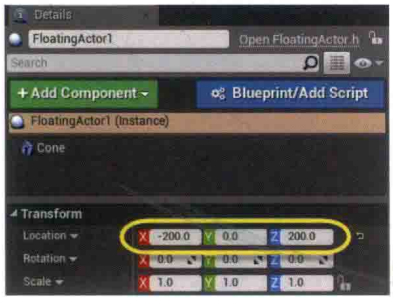

图 1-11

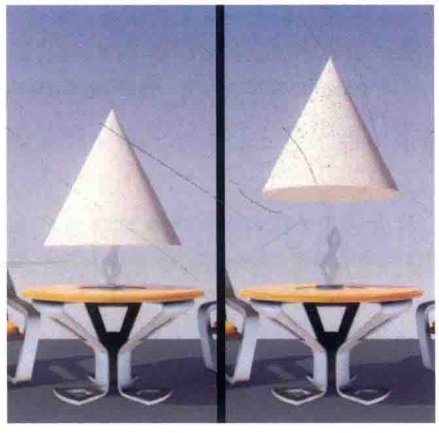

图 1-12

```
Finished Code
FloatingActor.h
//Copyright 1998-2017 Epic Games,Inc.All Rights Reserved.

#pragma once

#include "GameFramework/Actor.h"
#include "FloatingActor.generated.h"

UCLASS()
class QUICKSTART_API AFloatingActor : public AActor
{
    GENERATED_BODY()
```

```
public:
    //设置此actor属性的默认值
    AFloatingActor();

    //当游戏开始或生成时调用
    virtual void BeginPlay() override;

    //在每一帧调用
    virtual void Tick( float DeltaSeconds ) override;

    float RunningTime;
};
FloatingActor.cpp
//Copyright 1998-2017 Epic Games,Inc.All Rights Reserved.

#include "QuickStart.h"
#include "FloatingActor.h"

//设置默认值
AFloatingActor::AFloatingActor()
{
     //将此actor设置为在每一帧都调用Tick()。如果你不需要这项功能，你可以关闭它以改善
性能。
    PrimaryActorTick.bCanEverTick = true;

}

//当游戏开始或生成时调用
void AFloatingActor::BeginPlay()
{
    Super::BeginPlay();

}

//在每一帧调用
void AFloatingActor::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);

    FVector NewLocation=GetActorLocation();
      float  DeltaHeight=(FMath::Sin(RunningTime+DeltaTime)-
FMath::Sin(RunningTime));
    NewLocation.Z+=DeltaHeight*20.0f;  //把高度以20的系数进行缩放
    RunningTime+=DeltaTime;
    SetActorLocation(NewLocation);
}
```

1.1.5　发挥想象

通过使用你已经掌握的内容，你可以进行如下操作：

- 添加粒子系统组件到你的 FloatingActor 中。你的项目中已经包括了预编译的粒子系统。
- 你可以使用 UE4 的 UProperty 宏来显示一个变量，以用于你的 FloatingActor 的移动力的大小，而不是使用硬编码值。你可以查看变量、定时器和事件教程来获得关于此主题的帮助信息。
- 添加 X 和 / 或 Y 轴的周期性运动，并将“DeltaTime”值乘以 0.6 和 1.4 之间的值，这样你的 FloatingActor 看起来是自由悬浮的。对于能力加成道具来说，这个看起来很棒！

1.2　使用 Unreal Engine 中的 C++ 的导论

1.2.1　Unreal Engine 中的 C++ 妙不可言！

本节讲述如何在 UE4 中编写 C++ 代码。不必担心，UE4 中的 C++ 编程乐趣十足，上手完全不难！我们可以将 UE4 的 C++ 视为“辅助 C++”，因为诸多功能使 C++ 的使用变得十分简单。

阅读本节的前提是你需要熟悉 C++ 或其他编程语言。理解本节的前提是你已有 C++ 使用经验，但如你了解 C#、Java 或 JavaScript，也会发现其中的共通之处。

如你编程经验为零，我们也能助你一臂之力！阅读蓝图可视化脚本后即可上手。可通过蓝图脚本编写创建整个游戏！

可以在 UE4 中编写“纯旧式 C++ 代码”，但你通读本节并学习虚幻编程模型的基础后可达到更高的成就。我们将在随后进一步讨论。

1.2.2　C++ 和蓝图

UE4 提供两种方法创建游戏性元素：C++ 和蓝图可视化脚本。程序员可通过 C++ 添加基础游戏性系统。设计师即可在此系统上（或使用此系统）创建关卡或游戏的自定义游戏性。在这类情况下，C++ 程序员在他们最擅长的 IDE（通常为 Microsoft Visual Studio 或 Apple Xcode）中工作，而设计师则在虚幻编辑器的蓝图编辑器中工作。

两个系统均可使用游戏性 API 和框架类。这两个系统可单独使用，而结合使用形成相互补充后将展示真正的强大之处。那么这究竟意味着什么呢？这意味着：程序员在 C++ 中创建游戏性构建块，设计师利用这些块打造有趣游戏性时，引擎能发挥最佳工作效率。

如此说来，让我们一探究竟，了解 C++ 程序员为设计师创建构建块的典型工作流。在此例中，我们将创建一个类。此类稍后会由设计师或程序员通过蓝图进行延展。在此类中，我们将创建一些设计师可进行设置的属性，并且我们将从这些属性派生出新数值。结合我们提供的工具和 C++ 宏即可轻松完成整个过程的操作。

首先我们将使用虚幻编辑器中的类向导生成基础 C++ 类，以便蓝图稍后进行延展。下面

展示了向导的第一步：新建一个 Actor，如图 1-13 所示。

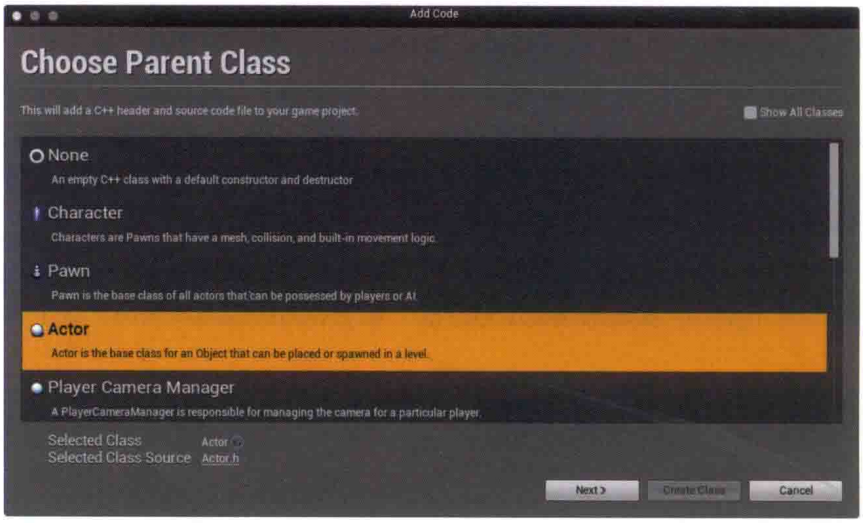

图 1-13

进程中的第二步是告知向导需要生成类的命名。下面的第二步中使用了默认命名，如图 1-14 所示。

图 1-14

选择创建类后，向导将生成文件并打开开发环境，以便开始编辑。这便是生成的类定义。

```
#include "GameFramework/Actor.h"
#include "MyActor.generated.h"

UCLASS()
class AMyActor : public AActor
{
    GENERATED_BODY()

public:
    //设置该actor属性的默认值
```