UNREAL
ENGINE

Unreal Engine 4

张宝荣◎编著

学习总动员

游戏开发

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

游戏开发

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

游戏开发是Unreal Engine 4 的拿手好戏，本书全面介绍游戏开发的全部内容。对游戏概念设计、游戏框架、游戏元素、游戏工具应用、游戏制作具体操作一网打尽。通过讲解几十个具体操作层面的教程，使读者全面掌握游戏制作的技能技巧。

配套资源中提供书中案例的工程源文件并精选十个精彩游戏制作视频教程，历时3小时以上，手把手一步步教会读者掌握Unreal Engine 4 游戏开发实战操作。使读者一步跨入游戏制作高端领域。

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. 游戏开发/张宝荣编著. —北京：中国铁道出版社有限公司，2019.7
ISBN 978-7-113-25780-4

Ⅰ.①U… Ⅱ.①张… Ⅲ.①虚拟现实—程序设计②游戏程序—程序设计 Ⅳ.①TP391.98

中国版本图书馆CIP数据核字(2019)第091690号

书　　名：Unreal Engine 4 学习总动员——游戏开发
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：MX 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：17.5　　字数：424 千
书　　号：ISBN 978-7-113-25780-4
定　　价：99.00 元

配套资源下载地址：

http://www.m.crphdm.com/2019/0515/14087.shtml

前言

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：

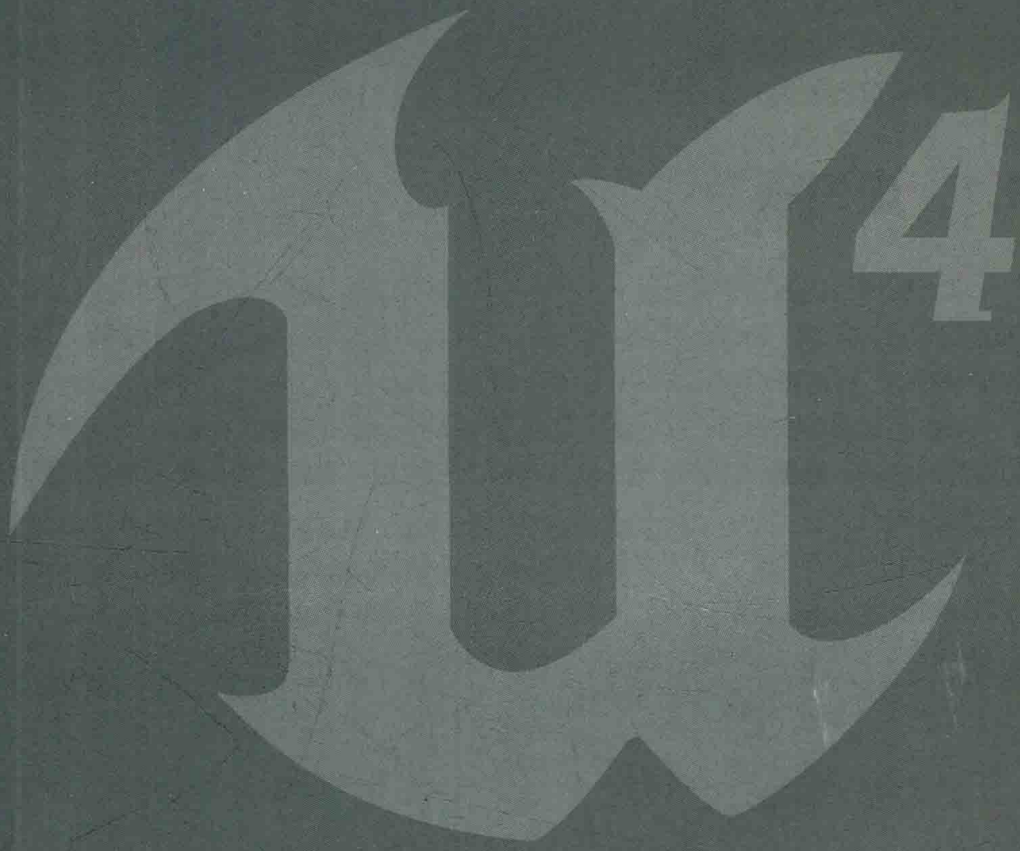
http://www.m.crphdm.com/2019/0515/14087.shtml

技术支持QQ群：596664789

配套资源下载地址：

http://www.m.crphdm.com/2019/0515/14087.shtml

目　录



第1章

基础游戏概念

本章内容将会让你了解关键的 UE4 术语。如需添加新类型的游戏对象，一般来说你需要创建一个新类。一个类指的就是你的新对象的模板或规则集合，这样你就可以创建尽可能多的拷贝，每个拷贝都包含了你可以在模板中设置的属性和行为。

1.1 虚幻项目和游戏

项目的所有内容均包含在项目目录中。可创建任意数量的项目，但每个均为自含式。使用 UE4 的项目浏览器（Project Browser）创建新项目，要设置必要的项目框架，如目录结构和可在编辑器中打开的虚幻项目文件（[ProjectName].uproject）。

项目所包含的资源作为 .uasset 文件存储在 Content 文件夹中。这些资源包括材质、静态和骨骼网格体、蓝图、声音提示，以及纹理。它们是可重复使用的参考物质和模板，可被项目中的对象调用。

项目中还包括关卡。关卡通常被称作地图，作为 .umap 文件存储在 Content 文件夹中。在虚幻编辑器中，每次可针对一个关卡进行操作，关卡将显示在视口中，如图 1-1 所示。

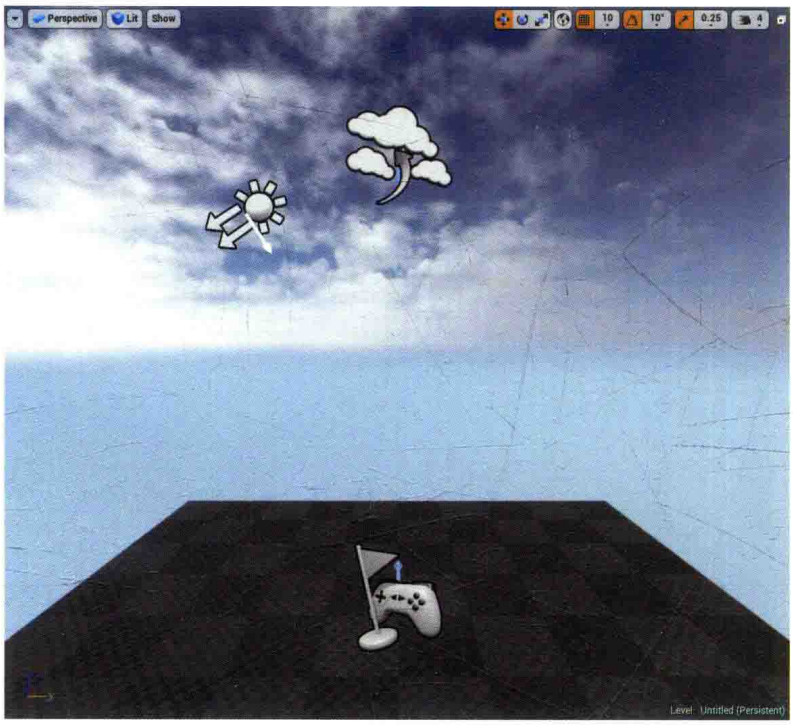


图 1-1

从最基础的层面而言，Actor 是一个游戏实体，（通常）包含一个或多个组件，可被放置在关卡中或在游戏进程中被生成，支持多人游戏中的网络复制。在上图的关卡中，地面平台、关卡中央的块，以及图片下方的 PlayerStart 都是 Actor。所有的 Actor 均由 AActor 类（可生成游戏对象的基类）延展而来。

在某种意义上，Actor 可被视为包含特殊类型对象（称作组件）的容器。例如，一个 CameraActor 包含一个摄像机组件（CameraComponent），如图 1-2 所示。

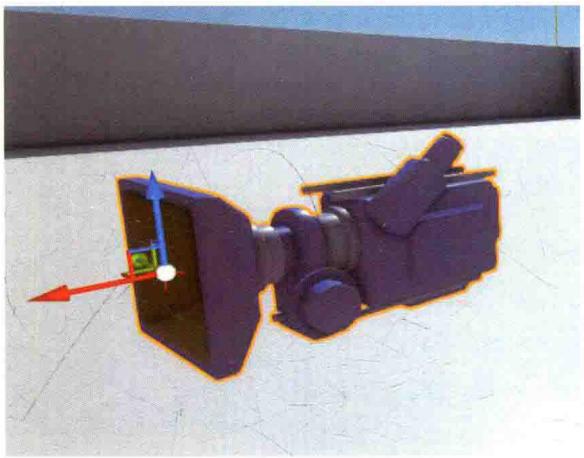


图 1-2

而摄像机组件包含摄像机的全部功能（如视场）。这意味着摄像机组件可包含在其他 Actor 中（如角色），为这些对象赋予相同的摄像机功能，如图 1-3 所示。

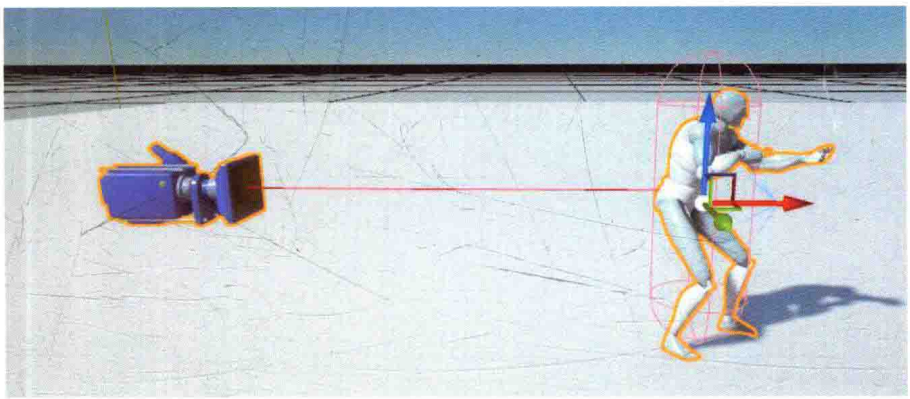


图 1-3

不同类型的组件可用于控制 Actor 移动的方式、Actor 被渲染的方式，以及 Actor 功能的诸多其他部分。所有对象，包括组件，皆由 UObject 类（游戏对象的基类）延展而来。这意味着它们无法被实例到世界场景中，必须从属于 Actor。

每个 Actor 或对象皆为一个类的单独实例。类设置 Actor 或对象的模板。它定义可针对该 Actor 或对象进行设置的变量，还定义可在该 Actor 或对象中进行执行的函数。可使用 C++ 代码新建类，或对象和 Actor 的类型。蓝图类主要用于创建设置新 Actor 的类，但也可通过蓝图类扩展一些对象。也可新建一个 C++ 类，然后制作一个派生自此 C++ 类的蓝图类，将以上两者结合起来。如需了解创建类并制作新类型 Actor 和对象的更多内容，请查阅类创建基础页面。

1.2　创建类的基础知识

这些示例展示了如何仅使用蓝图、仅使用 C++ 及同时使用二者来创建一个新类。目标是分别使用这三种流程来创建一个具有同样属性和行为的新 LightSwitch 类，然后把每个新

类的实例添加到关卡中，这样就有三个全新的 LightSwitch Actor 了，如图 1-4 所示。

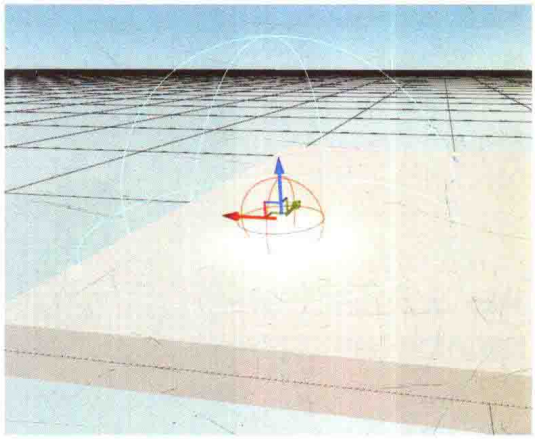


图 1-4

LightSwitch 类直接基于 Actor 类，因为它们的主要需要就是可被放置在关卡中。它们包含了一个 PointLightComponent 和 SphereComponent 来作为根组件，SphereComponent 是 PointLight Component 的子项。每个 LightSwitch 类都还包含一个名称为 DesiredIntensity 的变量，用于设置 PointLightComponent 的亮度。最后，这些类的默认行为是：当玩家或离开 SphereComponent 时，切换 PointLightComponent 的可见性，如图 1-5 所示为仅使用蓝图时的情形，图 1-6 所示为仅使用 C++ 时的情形，图 1-7 所示为同时使用 C++ 和蓝图时的情形。

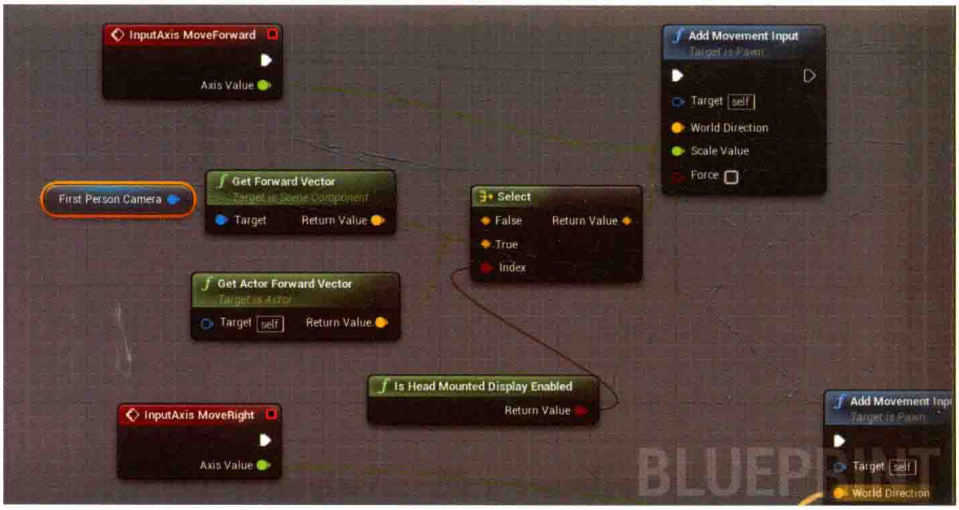


图 1-5

仅使用 C++：

```
// Copyright 1998-2018 Epic Games, Inc. All Rights Reserved.

#pragma once

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

UCLASS()
class HOWTO_AUTOCAMERA_API ACameraDirector : public AActor
{
    GENERATED_BODY()

public:
    // Sets default values for this actor's properties
    ACameraDirector();

    // Called when the game starts or when spawned
    virtual void BeginPlay() override;

    // Called every frame
    virtual void Tick( float DeltaSeconds ) override;

    UPROPERTY(EditAnywhere)
    AActor* CameraOne;

    UPROPERTY(EditAnywhere)
    AActor* CameraTwo;
```

图 1-6

```
// Copyright 1998-2018 Epic Games, Inc. All Rights Reserved.

#pragma once

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

UCLASS()
class HOWTO_AU
{
    GENERATED_B

public:
    // Sets de
    ACameraDire

    // Called w
    virtual vo

    // Called e
    virtual vo

    UPROPERTY(E
    AActor* Cam

    UPROPERTY(E
    AActor* Cam
```

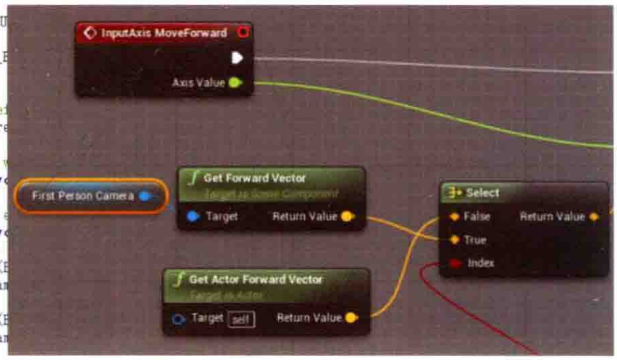


图 1-7

1.2.1　仅使用蓝图

蓝图类允许你使用蓝图 – 可视化脚本系统来设置新类。在你创建一个新的蓝图类后，你可以使用可视化脚本系统来添加组件、创建函数及其他游戏或设计行为，并设置类变量的默认值。仅使用蓝图创建的 LightSwitch 类的名称是 LightSwitch_BPOnly，以下进行解释。

类设置：

LightSwitch_BPOnly 类是在 Content Browser（内容浏览器）中创建的，使用选中的 Actor 作为父类。

组件可以通过 Blueprint Editor（蓝图编辑器）的组件模式静态地添加到蓝图中，或者在图表模式中通过可视化脚本动态地将组件添加到蓝图中。因为 LightSwitch 类总是具有 PointLightComponent 和 the SphereComponent，所以在这个示例中，是通过蓝图编辑器在组件窗口中静态地添加的组件。

使用 Components（组件）选卡添加 PointLightComponent，命名为 PointLight1，将其作为根组件；添加 SphereComponent，命名为 Sphere1，并将其附加到 PointLightComponent 上。在蓝图编辑器中添加到类上的组件具有淡蓝色图标，从父类继承而来的组件具有深蓝色图标，如图 1-8 所示。

在 Blueprint Editor（蓝图编辑器）中，你可以在我的蓝图选卡中添加新建的变量、函数，以及宏。你还可以访问蓝图类中包含的所有图表。在这些图表中，各种节点连接到一起，来创建由类变量、游戏事件，甚至 Actor 的周边环境驱动的设计及游戏功能。

通过使用 My Blueprint（我的蓝图）选卡，把浮点变量 DesiredIntensity 添加到了 LightSwitch_BPOnly 类中。My Blueprint（我的蓝图）选卡还显示了在 Components Mode（组件模式）中添加的组件，所以如果需要可以在图表中进行访问。要想阅读关于使用 My Blueprint（我的蓝图）选卡添加变量的更多信息，请阅读变量文档，或者阅读我的蓝图文档来获得通用的使用信息，如图 1-9 所示。

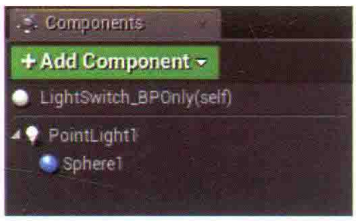


图 1-8

图 1-9

有两个图表用于设置 LightSwitch_BPOnly 类的行为。首先是构造脚本函数图表，它包含了构建脚本的函数入口节点。当把一个 Actor 添加到关卡中时或当一个现有 Actor 在关卡中移动时会执行该事件。在 LightSwitch_BPOnly 类中，Construction Script 事件连接到了 Set Brightness 节点上，以便当在关卡中添加或移动 Actor 时或者 Desired Brightness 发生改变时，将 Point Light 1(PointLightComponent) 的亮度设置为 DesiredIntensity 的值，如图 1-10 所示。

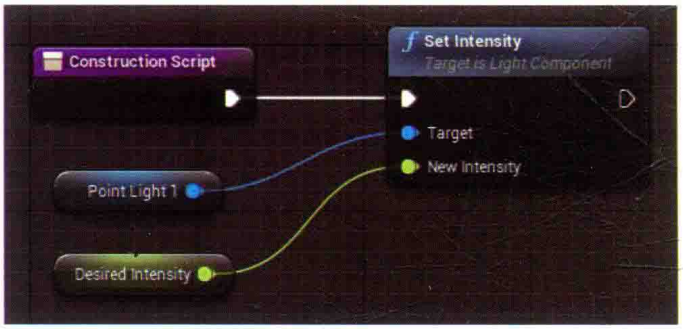


图 1-10

LightSwitch_BPOnly 类中设置的另一个图表是事件图表。EventGraph（事件图表）的执行是由事件启动的，比如 OnComponentBeginOverlap 和 OnComponentEndOverlap 事件。当关卡中的其他 Actor 和 SphereComponent 重叠或者离开 SphereComponent 时，会执行这些事件。这两个事件都连接到了 Toggle Visibility 节点上，所以当这些事件执行时会切换 PointLightComponent 的可见性，如图 1-11 所示。

如果你无法从右击菜单中找到 Toggle Visibility（切换可见度）选项，请取消勾选情境关联或在 Find a Node（搜寻节点）菜单中尝试搜索。你还可以通过直接从点光源变量节点拉根线然后搜索 Toggle Visibility（切换可见度）来获得相同的结果。

如果你无法找到 Sphere1 的 OnComponentEndOverlap，请确认你选择了球体变量，并使用事件 > 添加事件来添加事件，或者从右击菜单中选择添加事件 > 碰撞，查看 OnComponentBeginOverlap/OnComponentEndOverlap。

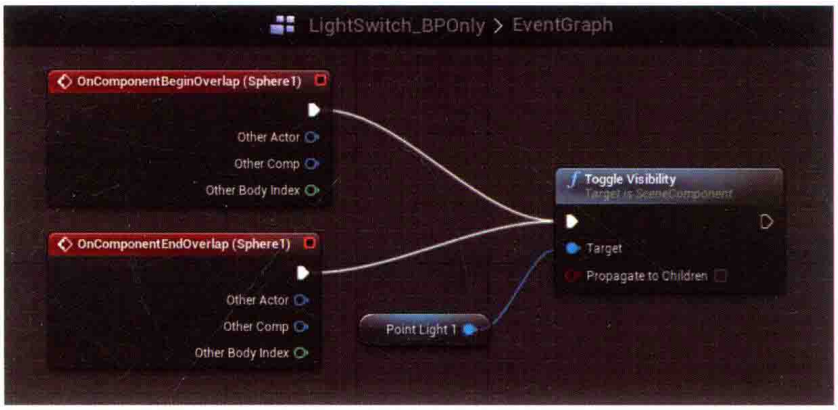


图 1-11

在变量的设置中，DesiredIntensity 变量设置为 Editable（可编辑的），所以在蓝图编辑器的类默认值中它是可见的，并且可以进行编辑。这意味着对于类的每个实例，这个变量是可以变化的，所以每个 Actor 可以有其自己的 DesiredIntensity，如图 1-12 所示。

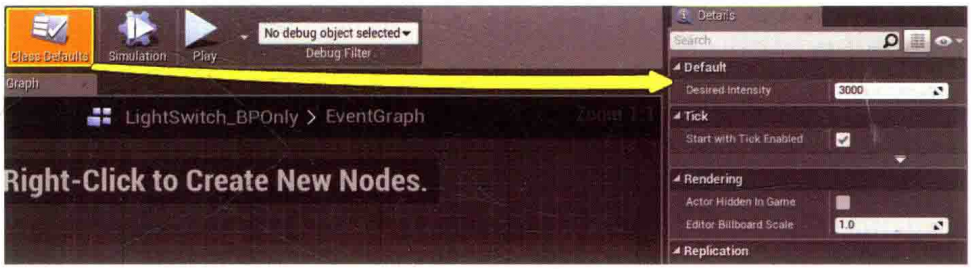


图 1-12

蓝图类可以使用其他蓝图类来进行扩展，通过以下两种方式实现：使用 Class Viewer（类别查看器）中的类附近的下拉列表按钮来创建一个新蓝图，或者通过右击该蓝图并选择 Create New Blueprint Based on This（基于此蓝图创建一个新蓝图）。

蓝图类 LightSwitchBPOnly 位于内容浏览器中，你可以从那里将其拖动到关卡内。它同时位于类别查看器中。

1.2.2　C++ 和蓝图

1. 概述

蓝图可以继承 C++ 类，从而使得程序员可以在代码中创建新的游戏类，而关卡设计人员可以使用蓝图来继承该类并对其进行修改。有很多种修饰符可以改变 C++ 类和蓝图系统间交互方式，其中某些修饰符会在本示例中突出介绍。

2. 类设置

在类设置的第一部分中，使用 C++ 类向导创建一个名称为 LightSwitchBoth 的类。

LightSwitchBoth 类中的大部分代码设置都和仅使用 C++ 的 LightSwitch 示例类似。尽管你可以让一个蓝图继承 LightSwitchCodeOnly 类，但蓝图图表并不能访问该类中创建的组件、属性及函数。该示例将使用 UPROPERTY() 和 UFUNCTION() 修饰符，这两个修饰符使得 LightSwitchBoth 作为继承它的蓝图的模板。

你会发现首先参考仅使用 C++ 的 LightSwitch 示例是有用的，你可以阅读下文了解如何设置头文件及源文件来创建 LightSwitchComponent、SphereComponent、DesiredBrightness 属性及 OnOverlap 函数。

这个头文件是从仅使用 C++ 的 LightSwitch 示例改编而来，添加了以下功能：

- PointLightComponent 和 SphereComponent 是 BlueprintReadOnly（仅蓝图可读的），并且将显示在我的蓝图选卡中的 Switch Components（切换组件）类目中。
- OnOverlap 现在是一个 BlueprintNativeEvent，将显示在我的蓝图选卡中的 Switch Functions（切换函数）类目中。
- DesiredBrightness 是 BlueprintReadWrite（蓝图可读写的），将显示在我的蓝图选卡中的 Switch Properties（切换属性）类目中。
- DesiredBrightness 现在是 EditAnywhere（随处可编辑的），而不是 VisibleAnywhere(随处可见的)。

UCLASS() 宏有个 Blueprintable 修饰符。在这个示例中，该修饰符不是必须的，因为 LightSwitchBoth 直接继承 Actor，而 Actor 是 Blueprintable（可蓝图化的），所以 LightSwitchBoth 会继承该修饰符。

加上 UPROPERTY() 和 UFUNCTION() 宏中的额外修饰符，LightSwitchBoth 类的头文件如下所示：

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

#pragma once

#include "LightSwitchBoth.generated.h"

/**
    *
    */
UCLASS()
class ALightSwitchBoth : public AActor
{
    GENERATED_UCLASS_BODY()

    /** point light component */
     UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category="Switch
Components")
    TSubobjectPtr<UPointLightComponent> PointLight1;

    /** sphere component */
     UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category="Switch
Components")
    TSubobjectPtr<USphereComponent> Sphere1;

    /** called when something overlaps the sphere component */
    UFUNCTION(BlueprintNativeEvent, Category="Switch Functions")
     void OnOverlap(class AActor* OtherActor, class UPrimitiveComponent*
```

```
OtherComp, int32 OtherBodyIndex);

    /** the desired brightness for the light */
     UPROPERTY(EditAnywhere, BlueprintReadWrite, Category="Switch
Properties")
    float DesiredBrightness;
};
```

在 LightSwitchBoth 的源文件中，构造器仍然是一样的。但是，需要对 OnOverlap 函数做一点修改。这个函数现在是一个 BlueprintNativeEvent。这意味着在继承这个类的蓝图中，可以放置一个覆盖 OnOverlap 的事件，当正常调用该函数时会执行此事件。如果该事件不存在，那么则是执行那个函数的 C++ 实现。要想使这个设置正常工作，该 C++ 函数需要重命名为 OnOverlap_Implementation。稍后在本示例中将介绍这个蓝图设置。对 OnOverlap 函数进行了修改后，LightSwitchBoth 的源文件如下所示：

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

#include "BasicClasses.h"
#include "LightSwitchBoth.h"

ALightSwitchBoth::ALightSwitchBoth(const FObjectInitializer&
ObjectInitializer)
    : Super(ObjectInitializer)
{

    DesiredBrightness = 15.0f;

     PointLight1 = ObjectInitializer.CreateDefaultSubobject<UPointLightCo
mponent>(this, "PointLight1");
    PointLight1->Brightness = DesiredBrightness;
    PointLight1->bVisible = true;
    RootComponent = PointLight1;
    Components.Add(PointLight1);

     Sphere1 = ObjectInitializer.CreateDefaultSubobject<USphereComponent>
(this, TEXT("Sphere1"));
    Sphere1->InitSphereRadius(250.0f);
     Sphere1->OnComponentBeginOverlap.AddDynamic(this, &ALightSwitchBot
h::OnOverlap);          // set up a notification for when this component
overlaps something
     Sphere1->OnComponentEndOverlap.AddDynamic(this, &ALightSwitchBoth::O
nOverlap);       // set up a notification for when this component overlaps
something
    Sphere1->AttachParent = RootComponent;
    Components.Add(Sphere1);
}

void ALightSwitchBoth::OnOverlap_Implementation(AActor* OtherActor,
UPrimitiveComponent* OtherComp, int32 OtherBodyIndex)
```

```
{
    if ( OtherActor && (OtherActor != this) && OtherComp )
    {
        PointLight1->ToggleVisibility();
    }
}
```

当创建类时，新的 UCLASS()、UFUNCTION() 和 / 或 UPROPERTY() 宏意味着该代码必须在 Visual Studio 或 Xcode 中进行编译，然后使用虚幻编辑器重新加载它们。关闭虚幻编辑器，在 Visual Studio 或 Xcode 中编译该项目，然后打开编辑器并重新加载该项目，以确保正确地重新加载该游戏模块。同时，需要注意的一点是，要确保 Build Configuration（版本配置）和你打开该项目使用的虚幻编辑器可执行文件的版本一致。请在编译游戏项目文档中阅读关于编译配置及编译项目的更多信息。

当重新打开虚幻编辑器并重新打开你的项目后，便可以创建一个新的类蓝图了。在本示例中，选择 LightSwitchBoth 作为该蓝图的父类，蓝图名称为 LightSwitchBoth_BP，如图 1-13 所示。

在 C++ 代码中添加的 PointLightComponent 和 SphereComponent 也会显示在 Blueprint Editor（蓝图编辑器）的组件模式中的 Components（组件）选卡内。它们的图标是深蓝色的，表示它们是从父类 LightSwitchBoth 类继承而来的原生组件。而刚刚添加到 LightSwitchBoth_BP 蓝图中的新组件的图标是浅蓝色的，如图 1-14 所示。

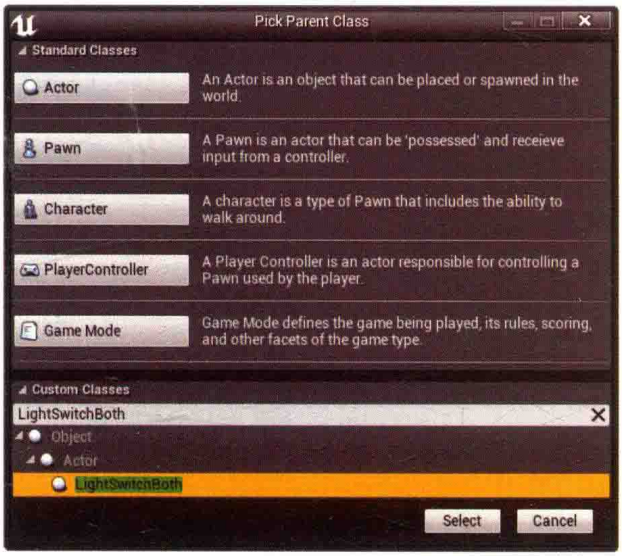


图 1-13

图 1-14

Blueprint Editor（蓝图编辑器）的图表模式是蓝图编辑的核心。在 Graph Mode（图表）模式中，你可以在我的蓝图选卡中添加新变量、函数及宏。你也可以访问该类蓝图中包含的所有图表。在这些图表中，各种节点连接到一起，来创建由类属性、游戏事件，甚至 Actor 的周边环境驱动的设计及游戏功能。

在 Graph Mode（图标模式）中，My Blueprint（我的蓝图）选卡显示了在 C++ 中添加到 LightSwitchBoth 类中的 PointLightComponent 和 SphereComponent。这是因为 Blueprint