第 7 章　用 UE5 开发《俄罗斯方块》休闲游戏

通过前面章节的学习，终于完成了实际制作游戏之前所有的前置知识的学习。从本章开始，将进入实际的游戏制作实例中。本章将和大家一起制作一款非常流行的休闲游戏——《俄罗斯方块》。通过制作该游戏，把一个游戏从创意分析、玩法归纳制作到最终打磨完成的整个环节一一展示给大家，以此来说明在游戏开发的过程中，每个环节都负责什么内容。读完本章，读者除了可以掌握《俄罗斯方块》休闲游戏的开发之外，还能够理解游戏是如何从创意开始，一步一步开发制作的。理解这些基本技能，就可以把它们应用在其他的游戏开发当中。学会了分析游戏玩法，能够开阔制作游戏的思路，任何功能的实现都会游刃有余。

本章重点

- 如何分析提炼游戏玩法
- 如何拆分玩法为不同的功能实现
- 游戏的核心架构
- UE5 玩法的基础类
- 使用蓝图实现各种功能
- 打包输出为不同平台的可执行文件

7.1　游戏玩法分析

从零开始开发一款《俄罗斯方块》游戏，是本章的主要目的。之所以选择《俄罗斯方块》，首先是因为这款游戏流行度非常高，在分析游戏玩法或者实现的时候，方便读者快速地理解。其次，《俄罗斯方块》游戏的玩法比较简单，方便归纳总结游戏制作的方法。这些归纳出来的方法是所有游戏都通用的，在后续任何其他类型的游戏开发上都能够发挥价值。

作为一款经典的游戏，《俄罗斯方块》已经流行了 30 多年，这 30 多年中，无论游戏怎样创新变化，依然有大量的玩家在休闲时间会玩上一把。这款游戏能够流行这么长的时间，有一个重要原因是玩法相对来说比较简单，非常容易上手。游戏的基本玩法如图 7-1 所示。

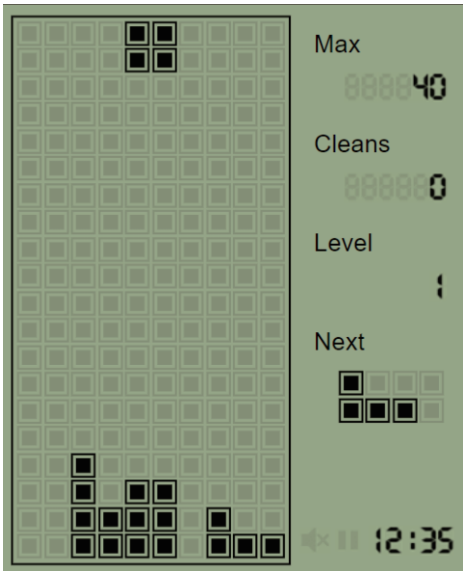

图 7-1　早期的《俄罗斯方块》

7.1.1　简单描述《俄罗斯方块》玩法

根据图 7-1 的演示，《俄罗斯方块》的基本玩法是这样的：有一个背景，方块只能在背景中左右移动或者旋转，方块不停下落，落到最下面后，生成新的方块。当背景网格的横向上有一行全部被填满时，此行消失，玩家得分。当新的方块达到背景网格的最上方时，游戏结束。

短短的几句话，就能描述《俄罗斯方块》的核心玩法。如果要制作该游戏，还需要在上面描述的基础上，进行更细化的规则总结。在开始制作游戏前，需要再细分一下上面描述的玩法。

7.1.2　详细总结《俄罗斯方块》玩法

（1）游戏运行在一个背景网格上，背景网格有宽和高。默认网格是空的，当玩家控制的方块停止运动后，会填充到背景网格上。

（2）同一时间内，只有一个方块在网格中缓慢地下降，当方块降到网格底部或者是降到现有的背景网格中的方块上的时候，方块停止移动，填充为网格上已有的方块。

（3）当方块停止移动后，检查背景网格上的每一行是否被方块填满。如果背景网格的某一行被方块填满，则被填满的行中的方块消失，当前行上面的行顺序往下降，玩家得分。

（4）当方块停止移动后，如果方块落在背景中的位置超出背景网格的最高点，代表新落下的方块超出了背景网格，游戏结束。

（5）当方块停止移动后，游戏没有结束，则继续生成下一个方块。

（6）玩家可以左右移动方块来控制方块在网格中的位置。

（7）玩家可以旋转方块来控制方块本身的旋转。

（8）玩家可以按下加速来加速方块下落的速度。

（9）每个方块由 4 个正方体组成，排列为不同的形状，有不同的颜色，但功能相同。

以上所有的逻辑和规则，构成了一款《俄罗斯方块》游戏。只要实现所有上面列出的逻辑，《俄罗斯方块》游戏也就差不多完成了。

> 注意：不要一开始就直接打开 UE5 开始编码。任何复杂的游戏都可以进行拆分。把一个复杂的游戏拆分成简单的规则后，只要对简单的规则进行实现即可。如果不能从逻辑上把一个游戏机制拆解为更简单的条件或者状态，那么在引擎中也是无法完成整个游戏的开发的。这和使用的引擎或者编程语言没有关系。而是对复杂问题简化分析的一种方法。所以，在开始制作游戏之前，通常需要先编写设计文档，良好的设计文档，保证了游戏开发的顺利进行。

7.2　创建项目

有了游戏的基础玩法的描述，这一节就可以开始创建项目了。

7.2.1　项目设置

打开 UE5 引擎，在弹出的项目管理器中单击“游戏”按钮，在右边单击“空模板”按钮，这个项目不需要任何其他模板中的功能（如图 7-2 所示）。

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 117　2024/4/3 16:06:10

图 7-2　新建游戏项目

在项目类型中单击“蓝图”选项，“目标平台”选择“桌面”选项，“质量预设”选择“最大”选项，“初学者内容包”和“光线追踪”都不要勾选（如图 7-3 所示）。

图 7-3　项目初始设置

接着选择项目保存的位置和项目名字。这里使用 Ch_07_Tetris 作为项目名。确认无误后，单击“创建”按钮。等待一段时间，UE5 创建项目完成后会自动打开。

7.2.2　创建2D空关卡

项目打开后，默认打开的 3D 关卡不符合 2D 游戏的制作需求，需要重新创建一个空关卡。

选择“文件”菜单，单击“新建关卡”按钮（如图 7-4 所示）。

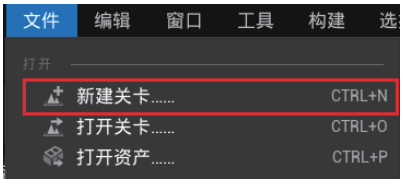

图 7-4　新建关卡

在弹出的对话框“新建关卡”里选择“空白关卡”（如图 7-5 所示）。

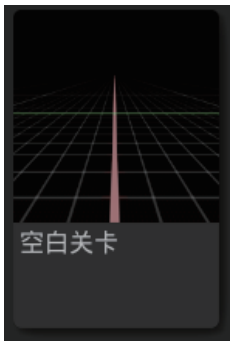

图 7-5　新建关卡模板

在内容浏览器中，右击“内容”目录，在弹出的快捷菜单中选择“新建文件夹”，创建一个新的文件夹，将新建的文件夹命名为 Maps（如图 7-6 所示）。

图 7-6　新建 Maps 文件夹

然后按快捷键 Ctrl+S，保存当前关卡。在弹出的“存储为 ...”对话框中，把当前的关卡保存在 Maps 目录下面并命名为 Game（如图 7-7 所示）。

图 7-7　关卡保存到内容浏览器中

7.2.3　创建关键游戏蓝图

创建完需要的关卡后，接着创建 UE5 游戏中最重要的几个玩法蓝图。这些蓝图

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 118　2024/4/3 16:06:10

在任何游戏中都存在，是基础架构性的蓝图。这里先创建它们，后面会逐一讲解。

1. 创建 GameInstance 类

在内容浏览器中，创建另外一个文件夹并命名为 Blueprints。双击打开 Blueprints 文件夹，右击文件夹内空白处，在弹出的快捷菜单中选择“蓝图类”，创建蓝图（如图 7-8 所示）。

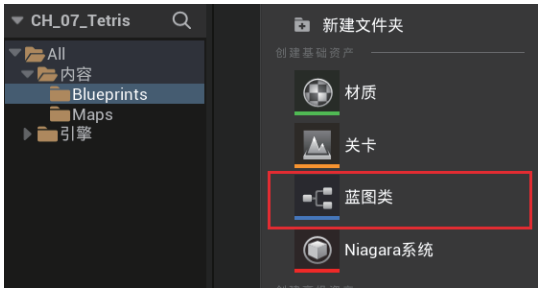

图 7-8　新建蓝图类

在弹出的“蓝图创建”对话框中，选择“所有类”，并在输入栏输入“game instance”，搜索 GameInstance（游戏实例）类（如图 7-9 所示）。

图 7-9　创建 GameInstance 子类

搜索到 GameInstance 类后，单击 GameInstance 选择这个类作为父类，后单击“选择”，将新创建的类命名为 BP_Tetris_GameInstance。

2. 创建游戏模式类

继续在 Blueprints 文件夹中右击，在弹出的快捷菜单中选择“蓝图类”创建蓝图，在新弹出的“创建蓝图”对话框中单击“游戏模式基础”，将新创建的蓝图的父类设置为“游戏模式基础”（如图 7-10 所示）。

图 7-10　父类设置为“游戏模式基础”

在内容浏览器中，将新创建的蓝图类命名为 BP_Tetris_GameMode（如图 7-11 所示）。

图 7-11　BP_Tetris_GameMode

3. 创建玩家 Pawn 类

Pawn 类是 UE5 中表示 NPC 或者玩家的类，大多数游戏都会有一个在游戏模式中指定默认要创建的 Pawn 存在。

接下来创建一个 Pawn 类作为《俄罗斯方块》的默认玩家角色。在内容浏览器中右击，在弹出的快捷菜单中选择“蓝图类”创建蓝图。在弹出的对话框中单击 Pawn，设置新创建的蓝图类的父类为 Pawn 类（如图 7-12 所示）。

图 7-12　创建 Pawn 蓝图类

在内容浏览器中，将新创建的类命名为 BP_Tetris_PlayerPawn（如图 7-13 所示）。

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 119　2024/4/3 16:06:11

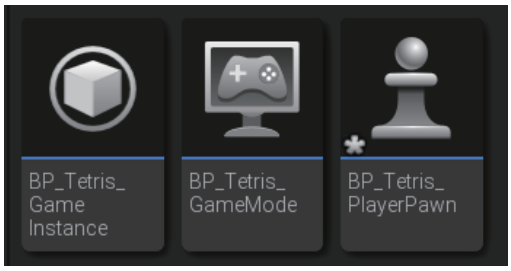

图 7-13　BP_Tetris_PlayerPawn

至此，所有框架相关的类就创建完成了。

7.2.4　设置UE5使用新创建的玩法类

创建完框架相关的玩法类，还要设置一下，才能让 UE5 引擎使用这些新创建的类。

1. 设置全局默认的游戏模式（游戏模式）

选择编辑菜单中的“项目设置”，打开项目设置面板。

在项目设置面板的左侧选择“地图与模式”类别，在右侧的“默认模式”类别下，将默认的游戏模式设置为新创建的 BP_Tetris_GameMode（如图 7-14 所示）。

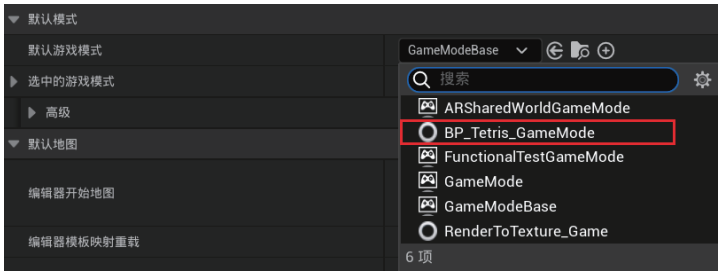

图 7-14　设置默认的游戏模式类为 BP_Tetris_GameMode

2. 设置默认玩家 Pawn

设置完默认的游戏模式后，单击“选中的游戏模式”前面的三角形按钮，打开细节选项。在“默认 pawn 类”中选择 BP_Tetris_PlayerPawn，这是设置游戏模式默认要生成哪个蓝图 Pawn 作为玩家（如图 7-15 所示）。

设置默认的 Pawn 类，也可以通过 BP_Tetris_GameMode 直接设置。在内容浏览器中，双击 BP_Tetris_GameMode，打开蓝图编辑器（如图 7-16 所示）。

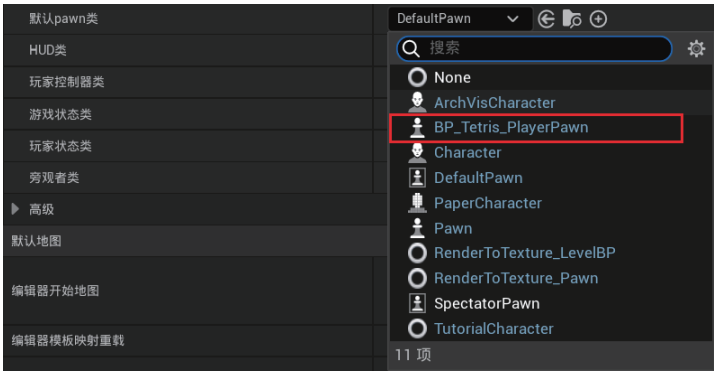

图 7-15　设置默认的玩家类

图 7-16　双击 BP_Tetris_GameMode

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 120　2024/4/3 16:06:11

在 BP_Tetris_GameMode 蓝图编辑器的右侧，找到“类”选项卡，在这里可以指定需要的“默认 pawn 类”（如图 7-17 所示）。

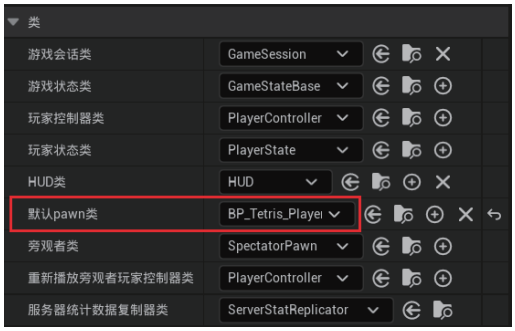

图 7-17　在“类”的默认面板中查看“默认 pawn 类”

3. 设置默认运行的关卡

在项目设置面板的“地图与模式”类别中还有一个重要的设置，就是“默认地图”的设置。在《俄罗斯方块》游戏中，自始至终只有一个 Game 地图。所以无论是“编辑器开始地图”，还是“游戏默认地图”，都选择 Game 地图。这样无论是打开 UE5 编辑器，还是最终打包好运行游戏，都会自动打开 Game 地图（如图 7-18 所示）。

图 7-18　设置默认关卡

4. 设置游戏使用的 GameInstance

在项目设置面板的“地图与模式”类别中，在页面最下面找到“游戏实例”类别，在“游戏实例类”中选择 BP_Tetris_GameInstance（如图 7-19 所示）。

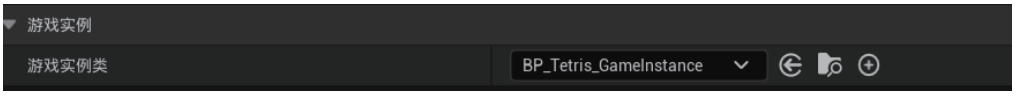

图 7-19　设置 Bp_Tetris_GameInstance

当所有类都设置完成后，单击工具栏上的播放按钮，运行当前关卡。在 UE5 编辑器的大纲面板中查看设置的类是否自动创建了（如图 7-20 所示）。

如果没有自动创建这些类，则可能是中间步骤出错。请从 7.2.3 节开始检查。

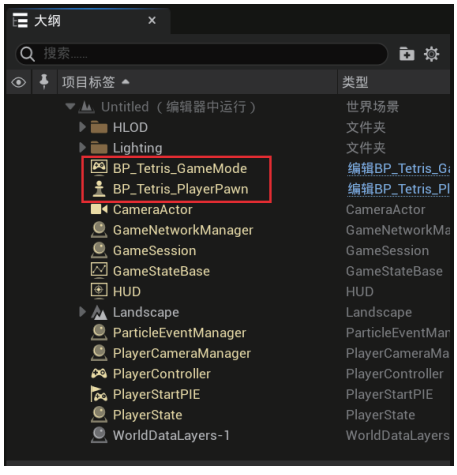

图 7-20　运行时自动创建的游戏类

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 121　2024/4/3 16:06:12

7.2.5 创建游戏相关类

根据 7.1 节的分析，《俄罗斯方块》项目除了上面引擎需要的玩法类之外，还有一些游戏相关的蓝图类需要创建。

首先创建背景网格类。这个类负责显示背景的网格、追踪已经放入网格中的小方块、计算游戏是否结束、创建新的方块等任务。这需要一个完整的蓝图类来处理这些逻辑。

在 Blueprints 文件夹中，右击空白处，在弹出的快捷菜单中选择“蓝图类”，父类选择 Actor，将其命名为 BP_Grid。

在背景网格中，会有一个方块一直往下运动。这个方块除了往下降落，还有左右移动、旋转等功能。它也需要一个单独的类来表示。

在 Blueprints 文件夹中，右击创建“蓝图类”，父类选择 Actor，将新创建的蓝图类命名为 BP_Tetris。

BP_Tetris 是由 4 个小方块组成的。当 4 个小方块落到网格上或者背景网格中现有的小方块上时，这 4 个小方块就添加到背景网格中，所以这些组成方块的小方块也适合用一个蓝图来表示。

在 Blueprints 文件夹中右击，在弹出的快捷菜单中选择“蓝图类”，父类选择 Actor，将新创建的蓝图类命名为 BP_Tile。

这三个蓝图类就是和《俄罗斯方块》游戏相关的游戏类。创建完成后如图 7-21 所示。

单击内容浏览器中的“保存所有”按钮，保存所有的修改。下一节导入游戏需要的外部资源。

图 7-21　创建蓝图类

7.2.6 导入资源

为了合理地导入资源，在“内容”目录中单独创建一个 Arts 目录，在 Arts 目录中分别创建 Sound 和 Textures 目录，用来存储声音和纹理（如图 7-22 所示）。

1. 导入纹理

打开本书自带的资源文件，找到 Texture 目录，这个目录是所需要的纹理文件。

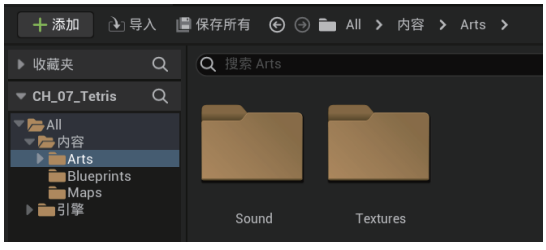

图 7-22　创建资源文件夹

在“内容浏览器”中打开 Arts → Textures 目录，然后将自带的 Texture 目录下的三个 TGA 文件拖放到内容浏览器中（如图 7-23 所示），UE5 会自动将三个纹理分别导入。

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 122　2024/4/3 16:06:12

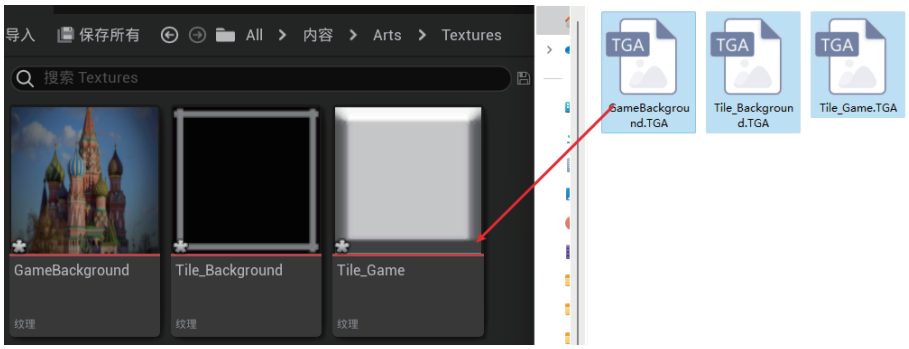

图 7-23　导入纹理

选择新导入的三张纹理，右击，在弹出的快捷菜单中选择“Sprite 操作”，然后选择“应用 Paper2D 纹理设置”，对这些纹理应用最适合 Paper2D 的设置（如图 7-24 所示）。

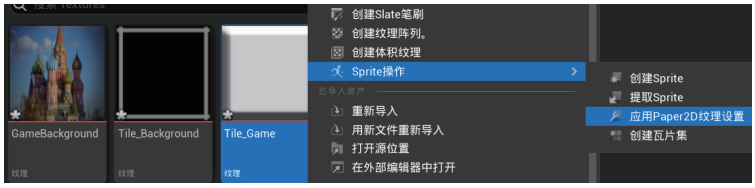

图 7-24　应用 Paper2D 纹理设置

再次在三个纹理选择的基础上右击，在弹出的快捷菜单中选择“Sprite 操作”，在子菜单中选择“创建 Sprite”。通过新导入的纹理创建 paper2D 的 Sprite（如图 7-25 所示）。

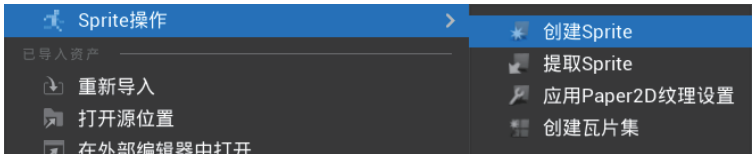

图 7-25　创建精灵

导入成功后，单击“保存所有”保存所有的修改。

2. 导入声音资源

打开本书自带的资源文件，在 Music 文件夹中有项目需要的音频资源。选择所有的音频文件，导入到“内容浏览器”→“内容”→ Sound 中（如图 7-26 所示）。

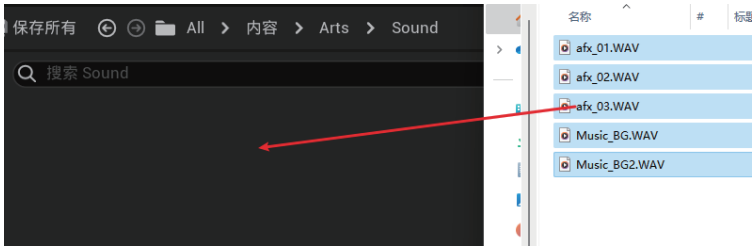

图 7-26　导入音频资产

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 123　2024/4/3 16:06:13

音频文件导入之后，可以把鼠标指针放置在音频文件的图标上，会出现播放按钮（如图 7-27 所示）。

图 7-27　出现播放按钮

单击播放按钮，可以对音频进行预览播放。单击“停止”按钮，停止播放音频。方便直接在 UE5 中查看音频，而不用打开外部的播放器。

确认音频播放正确后，单击“保存所有”，保存项目中所有的修改。

7.3　装饰场景

素材已经导入完成，下面将使用导入的素材对当前游戏关卡进行简单的装饰布局。

在内容浏览器中找到 Game Background_Sprite 背景 Sprite，直接把这个背景 Sprite 拖入场景中（如图 7-28 所示）。

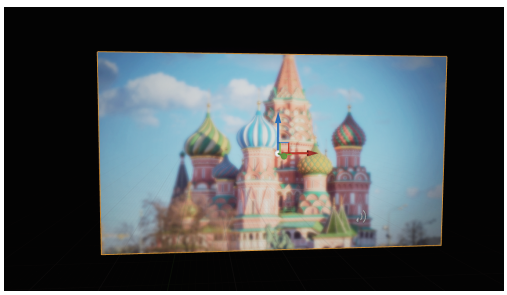

图 7-28　添加背景

在右侧的细节面板中把背景 Sprite 的“变换”面板里“位置”的值都设置为 0.0。把背景 Sprite 放在场景中默认的原点上。

7.3.1　设置后处理自动曝光

第 6 章中，已经介绍过自动曝光的处理，这里不再详细讨论，如果记不清细节，可以回到第 6 章查看。这里直接添加一个全局的后处理体，在细节面板中，找到“无限范围（未限定）”，让它处于选定状态（如图 7-29 所示）。

图 7-29　设置后处理体全局生效

将自动曝光的“最高亮度”和“最低亮度”都设置为 1.0（如图 7-30 所示）。

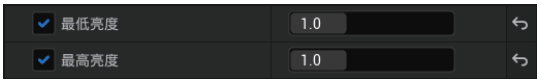

图 7-30　设置自动曝光参数

单击播放按钮播放当前关卡，确保编辑模式和运行模式下，看到的亮度是一样的。

7.3.2　设置色调映射器

选择后处理体，在细节面板中找到 Global 类别，勾选“对比度”，打开细节设置，将数值调整到 1.57；勾选“伽玛”，打开细节设置，将数值调整为 0.96；勾选“饱和度”，打开细节设置，将数值调整为 1.25。然后找到“电影”选项，勾选“趾部”并将数值修改为 0；勾选“肩部”并将数值修改为 1。这样就最大程度地避免了色调映射器对颜色造成的误差。具体细节可以参考第 6 章 6.3.3 节。

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 124　2024/4/3 16:06:13

7.3.3　其他后处理设置

设置完前面两项内容，当前的编辑器还有一些不符合 2D 游戏开发的设置，需要进一步调整。

打开设置面板，在左侧类别中选择“渲染”类别，找到“后处理”类别。这里是在没有后处理的情况下引擎使用的默认效果。把“泛光”“自动曝光”“环境光遮蔽”“动态模糊”全部取消勾选（如图 7-31 所示）。

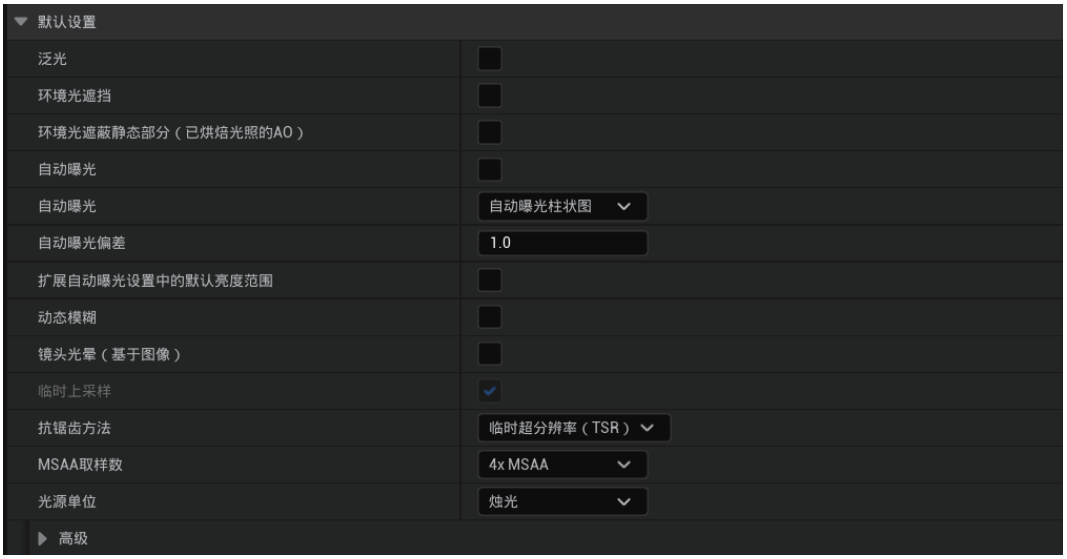

图 7-31　关闭渲染设置中的默认后处理效果

然后关闭项目设置，选择关卡中的后处理体，按照上面的设置再在后处理上设置一次。重启一下 UE5 编辑器，播放的效果就跟原先的设计图非常接近了。

> 注意：有时后处理明明设置好了，在播放时却不正确。这时只要重启一下 UE5 编辑器，问题基本就能解决。

7.3.4　添加背景音乐

在内容浏览器的 Sound 文件夹中，找到导入的背景音乐，这里选择 Music_BG，然后直接拖入场景中，UE5 会自动创建一个 AmbientSound 的角色来播放声音。再次单击播放运行游戏，音乐会自动播放（如图 7-32 所示）。

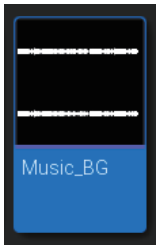

图 7-32　添加背景音乐

如果音量过大，可通过右侧细节面板音效里“音量乘数”来调节音量大小（如图 7-33 所示）。

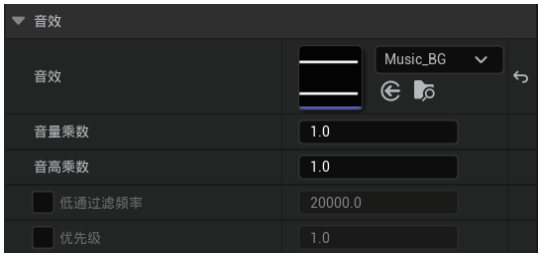

图 7-33　设置背景音乐音量

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 125　2024/4/3 16:06:13

7.4 设置摄影机

第 6 章中已经学习过如何创建 2D 摄影机了。这一节，要把 2D 摄影机放在 Pawn 中。UE5 是通过摄影机来观察场景的，如果当前的 Pawn 中没有摄影机，则会使用当前 Pawn 的某些设置，显示视口内容。

7.4.1 设置玩家使用的Pawn角色

在内容浏览器中找到 BP_Tetris_PlayerPawn，把它拖入视口中。在右侧细节面板“变换”中找到“位置”，将数值设置为 *X*=0.0，*Y*=400，*Z*=0.0。这保证新放入的 BP_Tetris_PlayerPawn 在背景前面一点的位置。

放好 BP_Tetris_PlayerPawn 后，单击工具栏中的播放按钮，观察右侧大纲视图，会看到两个 BP_Tetris_PlayerPawn，一个是刚刚放到场景中的，另一个是游戏模式配置在播放的时候自动生成的（如图 7-34 所示）。

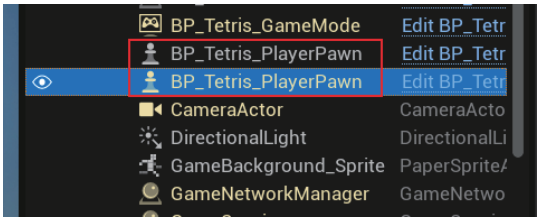

图 7-34　运行后关卡中有两个 BP_Tetris_PlayerPawn

选择放在场景中的 PlayerPawn，然后在右侧细节面板找到 Pawn 类别里边的“自动控制玩家”，把它设置为“玩家 0”，这样玩家进入游戏的时候就用这个 Pawn 作为自己的角色，游戏模式不再创建默认的 Pawn（如图 7-35 所示）。

图 7-35　设置关卡中的 BP_Tetris_PlayerPawn 为玩家 Pawn

7.4.2 添加玩家Pawn的摄影机组件

因为目前 BP_Tetris_PlayerPawn 中还没有摄影机组件，所以在播放当前关卡时，会把这个 Pawn 的方向作为摄影机的方向。

下面为 BP_Tetris_PalyerPawn 添加摄影机组件，精确地设置摄影机参数，让它更适合 2D 游戏的开发。

在内容浏览器中，双击 BP_Tetris_PlayerPawn，在蓝图编辑器中打开这个蓝图。

因为蓝图创建完成后并没有修改任何的逻辑，所以 UE5 会默认这个蓝图为数据蓝图，打开一种精简的数据编辑模式。单击“打开完整的蓝图编辑器”，在左上角的组件面板中单击“+ 添加”，输入 Camera，搜索并添加 Camera 组件。添加完成后，在组件面板中可以看到多了一个 Camera 组件（如图 7-36 所示）。

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 126　2024/4/3 16:06:14

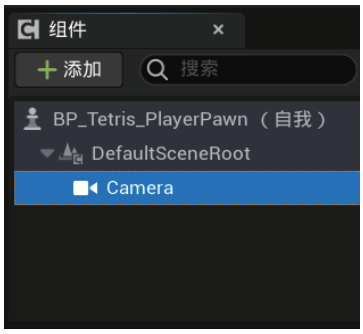

图 7-36　添加 Camera 组件

在蓝图编辑器面板的中间选择视口，这里和关卡视口一样，是以可视化的方式查看当前 Actor。摄影机模型显示代表添加摄影机组件成功（如图 7-37 所示）。

在蓝图编辑器的工具栏上单击“编译”“保存”（如图 7-38 所示），然后返回到主场景中。

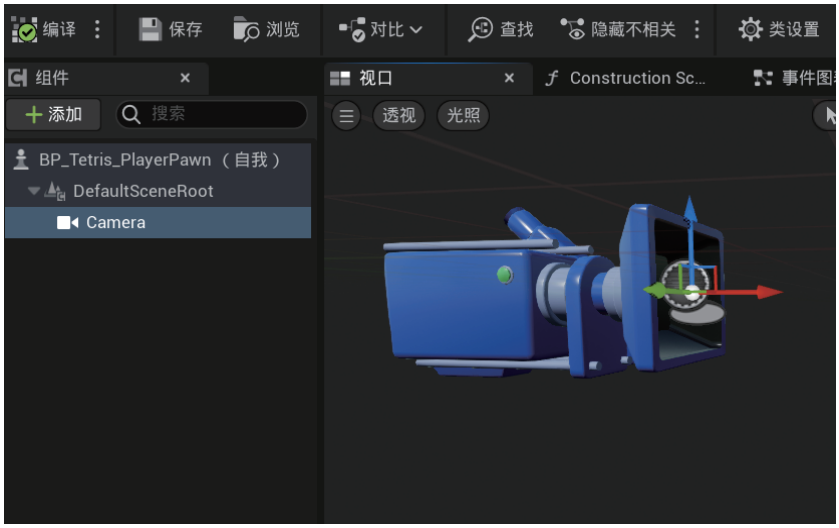

图 7-37　Camera 组件的摄影机模型显示

图 7-38　编译、保存蓝图

当选中的 Pawn 有摄影机的组件时，视口右下角就会出现当前摄影机的预览画面。

当前角色的方向是不正确的。选择关卡中的 BP_Tetris_PlayerPawn，然后旋转 90 度（如图 7-39 所示）。

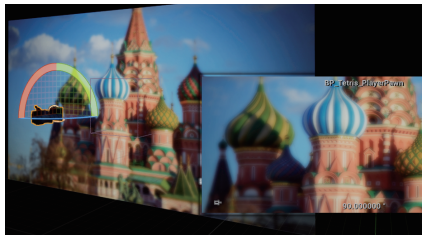

图 7-39　修改摄影机方向

7.4.3　设置摄影机组件参数

当前的摄影机是放在 BP_Tetris_PlayerPawn 角色中的一个组件。所以要设置摄影机，应该打开 BP_Tetris_PlayerPawn 的蓝图编辑器，选择 Camera 组件，在组件中进行设置。

选择摄影机组件，将右侧细节面板中的“摄像机设置”里面的投射模式改为“正交投影”。

然后设置“正交宽度”为 1280，调整后的摄影机，已经是能把整个背景图都正常显示的 2D 摄影机了（如图 7-40 所示）。

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 127　2024/4/3 16:06:14

图 7-40　设置正交摄影机宽度

单击蓝图编辑器上的“编译”“保存”。回到关卡编辑器，单击播放按钮。在播放的同时，左右拖动视口边框改变视口的大小，观察摄影机的行为是否正确（如图 7-41 所示）。

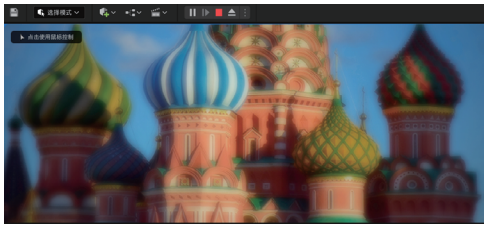

图 7-41　播放查看摄影机设置

7.5　创建背景网格

根据 7.1 节的分析，《俄罗斯方块》游戏需要一个 BP_Grid 的角色来管理很多的游戏逻辑。这一节就通过蓝图来添加 BP_Grid 的具体功能。

从本节开始将会使用大量的蓝图功能。读者放心，本书中所用到的蓝图功能，都是非常容易理解的。在遇到某个第一次出现的蓝图的功能或者用法的时候，会在下面的内容中详细讲解蓝图功能。读者可以先按照步骤完成具体的功能，然后在后面的蓝图讲解的部分学习蓝图的基础。

7.5.1　添加网格到关卡

之前已经提前创建了 BP_Grid 角色。在内容浏览器的 Blueprints 文件夹下，拖动 BP_Grid 到场景中，为了让网格在背景上显示，在右侧的细节面板设置中，把 BP_Grid 角色的位置设置为 *X*=0.0，*Y*=40.0，*Z*=0.0。

7.5.2　为背景网格添加的长度和宽度

双击 BP_Grid，单击“打开完整的蓝图编辑器”，进入 BP_Grid 的蓝图编辑器。要创建 Grid 背景的网格图片，首先要定义长和宽的变量，分别代表在宽度和高度上都有几个方块。

在蓝图编辑器的左下角，找到“我的蓝图”面板，找到变量类别，单击 + 按钮，添加变量，会创建一个默认的变量。将新创建的变量命名为 Width（宽），然后在右边把变量类型改成整数（如图 7-42 所示）。

图 7-42　设置变量名称和类型

接下来再创建另外一个变量，将其命名为 Height（高），变量类型会自动继承上一次的设置，类型是整数。在右侧的细节面板中，观察最下方的默认值（如图 7-43 所示）。

图 7-43　变量新加入后，不能设置默认值

这里提示要设置默认值，就必须先编译蓝图。单击工具栏上的“编译”按钮（如图 7-44 所示）。

图 7-44　“编译”按钮

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 128　2024/4/3 16:06:15

编译成功后，“编译”按钮会出现绿色的对号图标，代表编译成功。再到细节面板中，就可以为变量赋值了（如图 7-45 所示）。

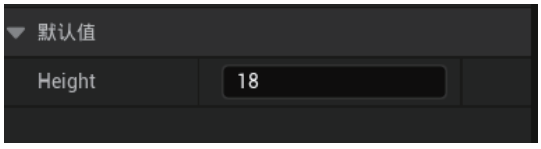

图 7-45　设置变量默认值

将 Width 设置为 10，Height 设置为 18，编译并保存蓝图。

1. 什么是变量

这一节，在蓝图编辑器中分别添加了两个变量：Width 和 Height。那什么是变量呢？

可以把变量理解为快递的盒子。快递盒子里可以放任意的物品，也可以把物品从快递盒子中取出来。

创建一个名称为 Width 的变量的意思是创建一个有收货地址栏的快递盒。名称 Width 可以理解为收货地址。在程序的其他部分，可以通过 Width 这个变量名称访问或者改变它里面保存的值。

2. 什么是变量类型

不同的快递盒能够放不同的快递。例如有一个冰箱，把它放在发送文件的信封中，肯定是放不下的。所以必须使用一个支持冰箱大小的快递盒来包装冰箱。

在创建变量的时候也是一样的。通常会给变量指定一个类型，来说明什么样类型的数据可以放到这个变量中。

默认创建的变量类型都是布尔类型。这种类型只有两个值：True 和 False。不是真就是假，这在进行条件判断的时候非常有用。

而 Width 和 Height 用来规定在背景网格的横向和纵向各有多少个方块。这样的数值用整数来表示是最合适的。所以前面把 Width 和 Height 的类型都设置为整数类型，整数类型的变量里面只能存储整数。

3. 什么是默认值

当一个变量创建完成后，还没有对变量赋值之前，去取变量的值，系统会返回一个默认值。这个默认值根据变量类型的不同而不同。如整数类型默认值就是 0，而布尔类型的变量默认值是 False。

可以在编译后对默认值进行设置，如把 Width 设置为 10。设置完默认值后，在第一次访问 Width 变量的值时，就能够得到默认的值，也就是 10。

默认值提供了一个初始化变量的机会。对创建的任何变量，都要仔细设计其默认值。

7.5.3　创建背景方块

下面根据设置的 Width 和 Height，创建背景网格中的背景方块。

1. 添加自定义事件

在组件面板中，单击“+ 添加”添加组件按钮，输入 PaperGroupedSprite 单击“添加”。这个组件是显示多个同样的 Sprite 的组件（如图 7-46 所示）。

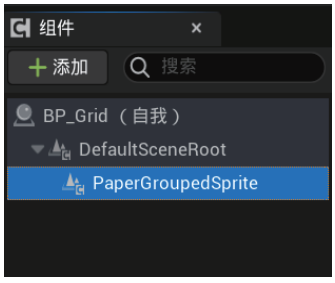

图 7-46　添加 PaperGroupedSprite 组件

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 129　2024/4/3 16:06:15

在蓝图编辑的中间部分，切换到“事件图表”标签。这里是编写蓝图逻辑的地方。在事件图表的空白处右击，输入“t.”，能快速找到“添加自定义事件”节点，单击该节点把“添加自定义事件”添加到事件图表中（如图 7-47 所示）。

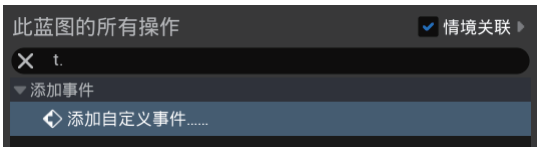

图 7-47　快速创建自定义事件

命名新加入的自定义节点为 CreateBackgroundGrid。到这里就成功地添加了一个自定义的事件（如图 7-48 所示）。

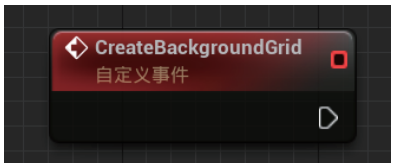

图 7-48　创建自定义的 CreateBackgroundGrid 事件

2. 循环添加背景方块

下面将遍历背景网格中的每一个方块，并创建对应的 Sprite。首先将 Height 拖动到面板中，在弹出的上下文菜单中单击“获取 Height”，然后按住鼠标左键拖动出连接线，释放鼠标左键，在弹出的上下文菜单中输入 For Loop，搜索到对应节点后单击创建一个 For Loop 循环，将 CreateBackgroundGrid 与 For Loop 循环相连（如图 7-49 所示）。

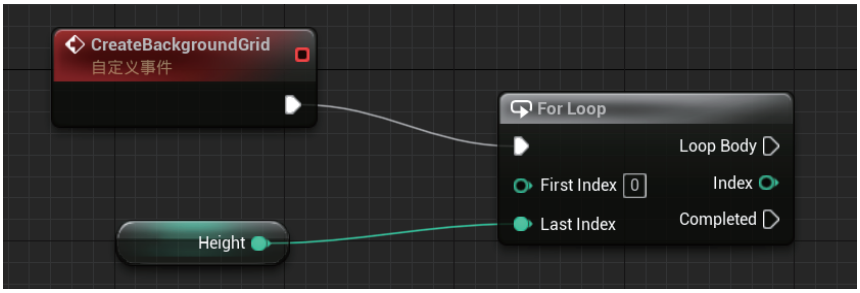

图 7-49　创建 For Loop 循环节点

因为 UE5 的 For Loop 默认是从 0 开始计数的，所以对于 For Loop 的 Last Index 索引，正确的数值应该是 Height−1，否则会多循环一次。在事件图标中拖动 Height 的数据线，在上下文中输入“-”，搜索“减”节点，这个节点对应数学上的减法操作。对 Height 的数值减 1，然后把减节点的输出连接到 For Loop 节点的 Last Index 上。这样在执行 For Loop 节点时，循环的 Index 依次是 0，1，2，3，4，5，6，7，8，9。正好从第 0 行开始，依次遍历 10 行（如图 7-50 所示）。

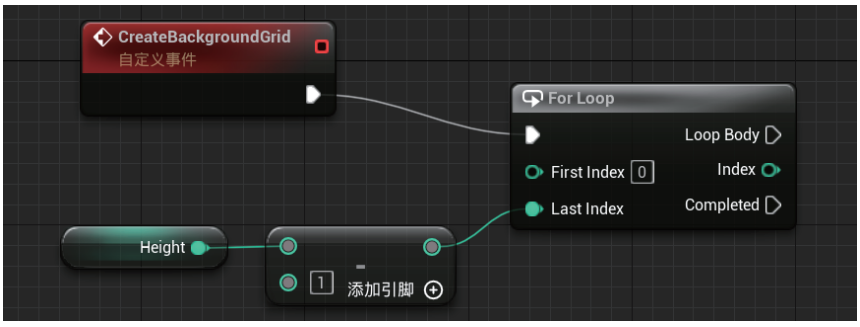

图 7-50　完整地遍历所有行

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 130　2024/4/3 16:06:15

在第一个循环执行时，针对每一行都需要再执行第二个循环，就是每一行上的列。和 Height 一样，需要对 Width 执行一遍循环（如图 7-51 所示）。

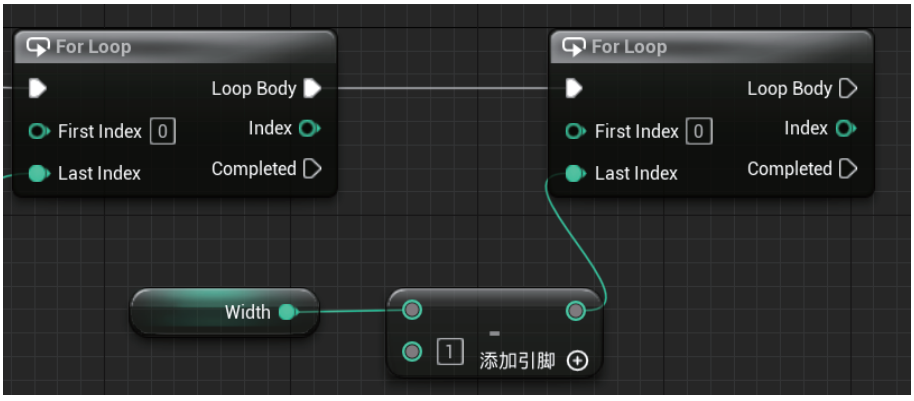

图 7-51　遍历每一行的每一列

接下来在组件面板中选择 Paper Grouped Sprite 组件，把它拖动到图表中，再拖动出数据线，从弹出的上下文菜单中搜索 Add Instance（添加实例），单击添加这个节点到事件图表中，连接第二个 For Loop 的 LoopBody（循环体）（如图 7-52 所示）。

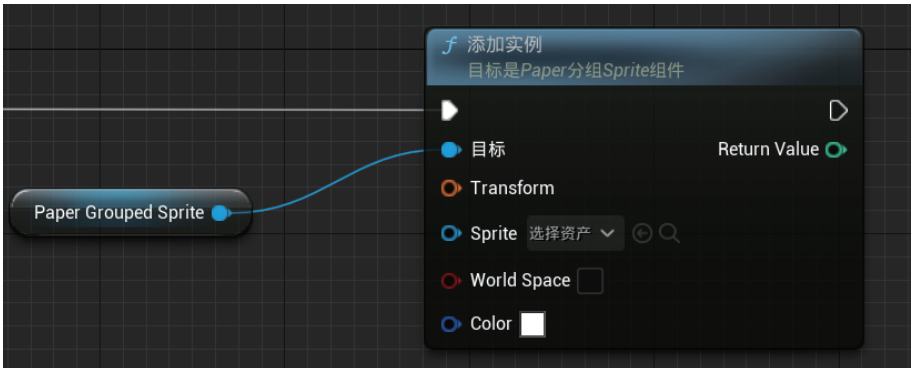

图 7-52　对每一行每一列都添加一个 Sprite 实例

图 7-52 的作用是，执行 Paper Grouped Sprite 上的 Add Instance（添加实例）事件。因为是在 for 循环中，这个事件执行的次数是 10×18 = 180 次。也就是每一个网格上的方块，都会执行这一次。刚好在这里添加方块的 Sprite。在 Add Instance 节点的 Sprite 设置中，选择背景方块精灵 Tile_Background_Sprite（如图 7-53 所示）。

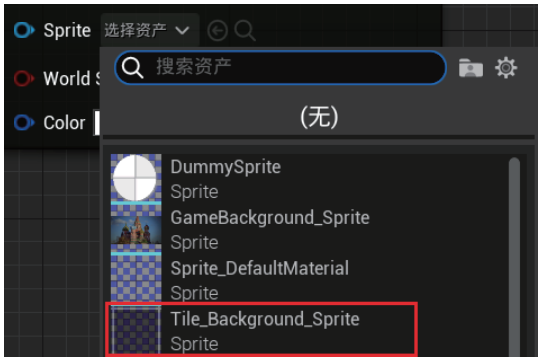

图 7-53　设置 Paper Grouped Sprite “组件”使用的 Sprite

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 131　2024/4/3 16:06:16

Add Instance 节点需要知道在什么位置创建 Sprite，所以需要设置 Transform 属性。右击 Add Instance 中 Transform 输入槽，在弹出的快捷菜单中选择“分割结构体引脚”，将 Transform 分离。一个 Transform 结构是由位置旋转缩放三个结构组成的。所以分离后可以对这三个部分分别赋值（如图 7-54 所示）。

图 7-54　分离 Transform 结构

首先设置一下 Add Instance 的 Scale。把 Transform Scale 设置为 0.25。这样每次添加的 Sprite 都是 25 像素，也是 25 个虚幻引擎单位大小（如图 7-55 所示）。

图 7-55　设置 Instance Sprite 的缩放

对每一个方块的位置也需要处理。每增加一行高度就增加 25× 行数；每增加一列就增加 25× 列数。

在 Location 上右击，再次选择“分割结构体引脚”，把 Location 分隔开。Location 是向量 3D 的变量，它由 *X*、*Y*、*Z* 三个浮点数组成（如图 7-56 所示）。

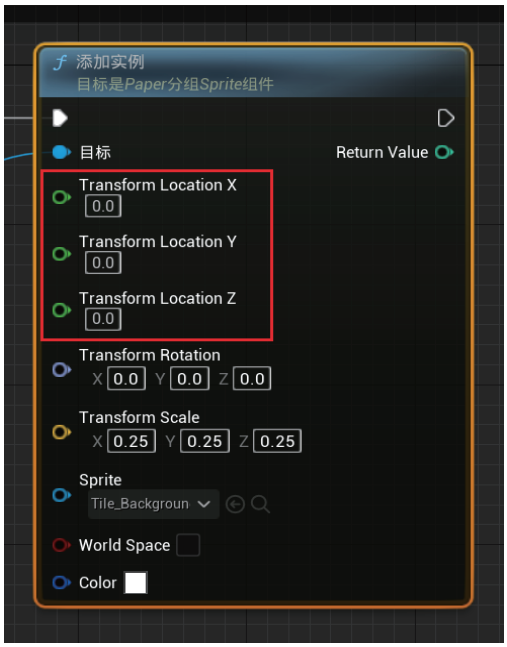

图 7-56　再次分离 Location 结构

对于 For Loop 来说，当前正在进行的循环的索引号，在 for 循环的 Index 数据上输出。例如第一行 index 输出 0，第 5 行 index 输出 4。按照刚才的分析，在 Width 上的每一个循环把 Index 的值乘以 25，然后赋值给 Add Instance 的 Location X。对于高度也是同样原理，把高度循环的 Index 乘以 25，赋值给 Location 的 *Z* 值（如图 7-57 所示）。

3. 调用自定义的添加方块事件

当前 BP_Grid 已经有了创建背景方块的功能，但是这个事件并没有被调用。如果要让 BP_Grid 创建背景方块，就要在合适的时机运行这个事件。

因为 BP_Grid 是放在关卡中的，在放进关卡之后就应该能看到背景网格上的方块，所以调用 Create Background Grid 事件的最合适的时机是在构造脚本函数中。在“我的蓝图”的函数类别中，找到“构造脚本”，双击打开（如图 7-58 所示）。

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 132　2024/4/3 16:06:16

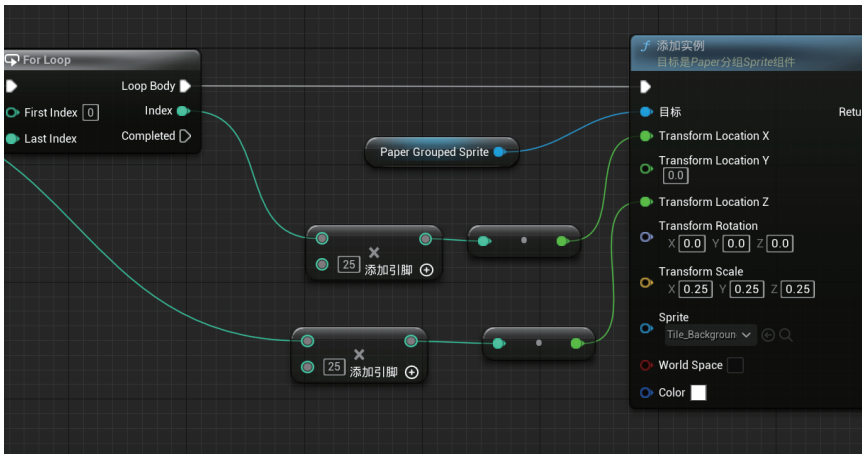

图 7-57　每一行的高度在 *Z* 轴上增加 25

图 7-58　双击构造脚本打开构造脚本函数编辑器

构造脚本是一个非常特别的函数，它在需要构建这个蓝图实例的时候调用。调用的时机包括关卡加载时、蓝图实例新添加到关卡中时、蓝图出现变动需要重新添加到关卡中时。

所以在构造脚本中调用 Create Background Grid 事件，会让 BP_Grid 在加入关卡中后，立即调用 Create Background Grid 事件，从而创建出背景方块。

在构造脚本的空白处右击，在弹出的上下文菜单中搜索 Create Background Grid，搜索节点后单击“添加”。然后连接 Construction Script 的输出流到 Create Background Grid 的输入流中（如图 7-59 所示）。

图 7-59　在构造函数中，调用 Create Background Grid 事件

重要说明：

你可能已经注意到了，蓝图图表编辑器中的自定义事件节点的名字，和定义时的名字有点不同。在蓝图图表编辑器中，为了可读性，UE5 默认对节点名称的显示进行了优化。主要的优化方式有：

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 133　2024/4/3 16:06:17

- 自动把下画线改为空格，如 BP_Tetris 会显示为 BP Tetris。
- 自动在单词首字母大写的函数名和变量名中间添加空格。如 CreateBackgroundGrid 显示为 Create Background Grid。
- 自动把小写的变量名的首字母改为大写，如 isJump 显示为 Is Jump 等。

这些优化措施让图表编辑器中的节点看起来更整洁，本书后面不再对这种情况单独说明，正文文字中将统一按照优化后的形式来写。由于这种优化显示问题，蓝图编辑器截图中的名称和原始名称会有稍微不一致的情况，所有后面出现这种情况的地方，请以定义的原始名称为基础进行理解。

再回到关卡中，可以看到 BP_Grid 已经生成好了背景的方块（如图 7-60 所示）。

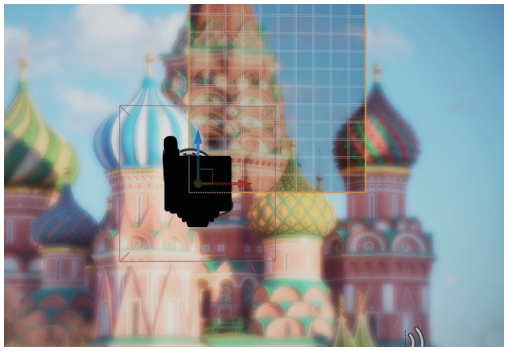

图 7-60　构造函数自动生成了背景网格

但是现在背景方块的位置是不对的。它从世界位置的 0×0×0 开始创建。我们需要背景网格的中心在世界坐标的 0×0×0 的位置。所以在创建背景方块之前，需要先根据宽度和高度，设置一下 BP_Grid 的位置。找到 Create Background Grid 事件，在第一个循环的前面右击，搜索添加“设置 Actor 位置”节点，该节点的作用是设置 BP_Grid 在关卡中的世界坐标位置。根据 Width 和 Height 的位置，首先除以 2，得到中间的方块的数量，然后乘以 −25，往左下方平移。*Y* 轴设置为 40，让背景网格比背景图片向前 40 个单位（如图 7-61 所示）。

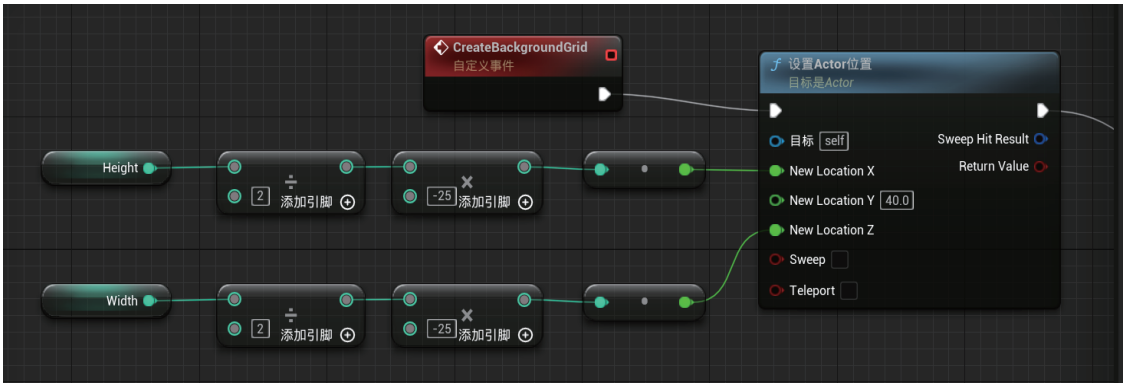

图 7-61　往左下方移动 BP_Grid

设置完成后，把后面循环的代码再次连接上，然后编译保存。回到关卡编辑器中，播放当前关卡，BP_Grip 就居中显示了（如图 7-62 所示）。

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 134　2024/4/3 16:06:17

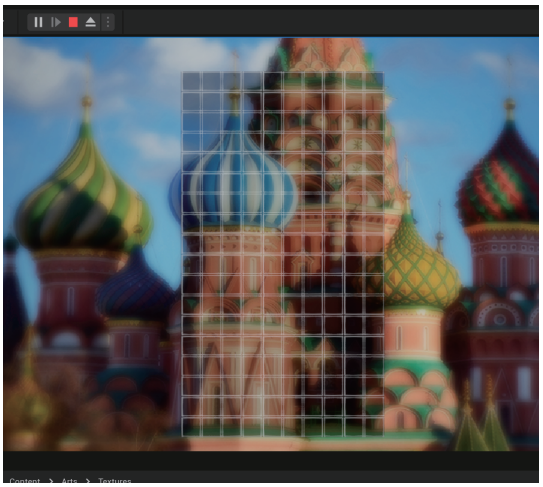

图 7-62　居中显示的 BP_Grid

7.6　BP_Grid 创建方块

接下来要从 BP_Grid 中创建出往下落的方块。双击 BP_Gird（网格），打开蓝图编辑器。在中间的事件图表的空白位置右击，创建一个新的自定义事件，命名为 CreateTetris（如图 7-63 所示），UE5 优化后会显示为 Creat Tetris。

图 7-63　创建 CreateTetris 自定义事件

在 Create Tetris 节点的后面右击，搜索 Spawn Actor，搜索添加 Spawn Actor from Class（从类生成 Actor）节点。将 Create Tetris 与 Spawn Actor from Class 连接起来，在 Spawn Actor from Class 的 Class 选择框中选择 BP_Tetris（如图 7-64 所示）。

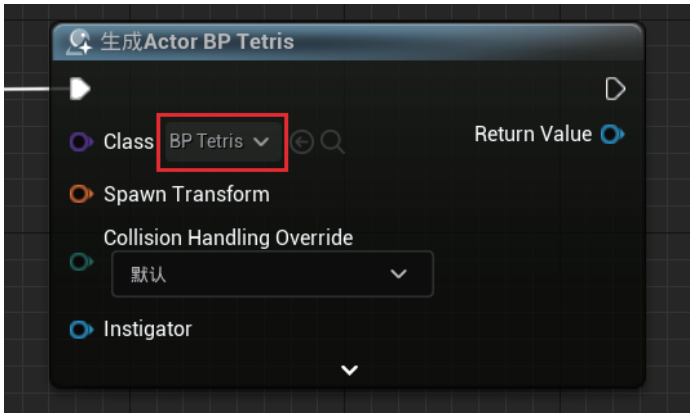

图 7-64　设置要 Spawn 的 Actor 的类为 BP_Tetris

设置要生成的蓝图类之后，还需要设置新生成角色的 Transform。右击 Spawn Transform Location 的“分割结构体引脚”将其分离开，方便分别设置位置。在 BP_Grid 的基础上，寻找 Width 在中间、Height 在比最高行还高一行的位置。BP_Grid 在开始的时候有一个位移，所以这里要把这个位移算进来（如图 7-65 所示）。

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 135　2024/4/3 16:06:18

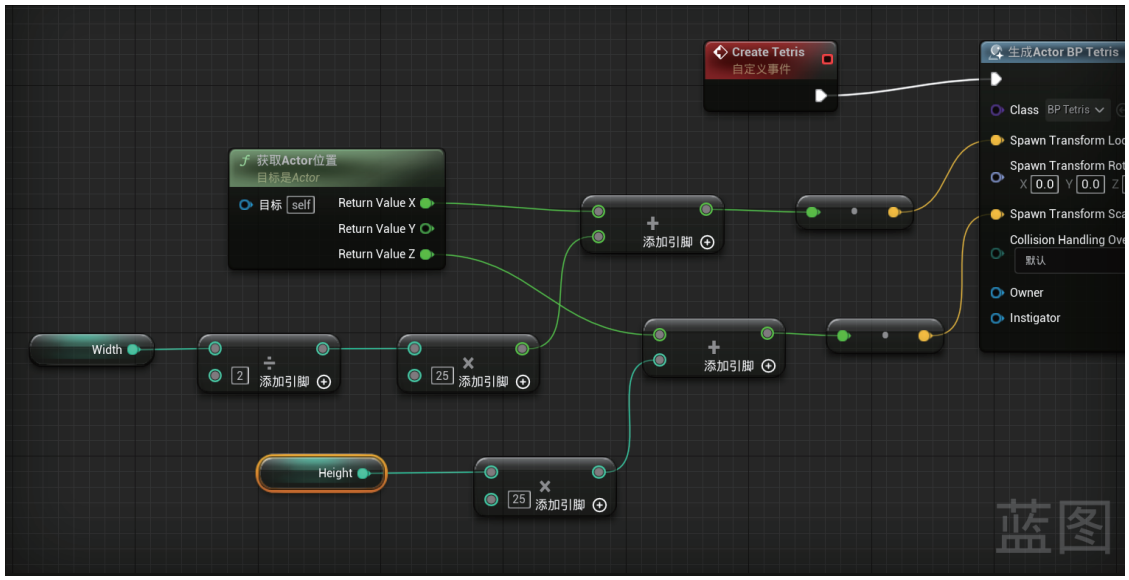

图 7-65　设置新生成 BP_Tetris 的位置

生成 BP_Tetris 的功能就完成了。要让这个事件起作用，还需要在合适的时机调用这个事件。这次是在游戏开始运行后生成一个 Tetris（Create Tetris），所以合适的位置是“事件开始运行”函数。“事件开始运行”函数在关卡加载完成后，开始绘制第一帧画面之前，调用一次（如图 7-66 所示）。

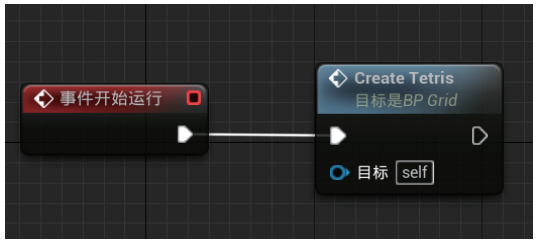

图7-66　在“事件开始运行”函数中，调用 Create Tetris

编译保存蓝图。回到关卡编辑器，单击运行。观察大纲视图，如果创建 BP_Tetris 的事件成功了，在大纲中能看到 BP_Grid 创建的 BP_Tetris 角色（如图 7-67 所示）。

图 7-67　在大纲面板中观察是否生成了 BP_Tetris

7.7　添加 BP_Tetris 自动下落功能

7.7.1　添加预览模型

因为 BP_Tetris 里还没有任何可视化的内容，所以当前还不能在游戏运行的时候看到它。一般出现这种需要的资源没有完成，但是当前的功能又依赖未完成资源的情况时，可以考虑添加使用可替换资源。

Unreal Engine 5与二维游戏设计 4校 正文第7-10章.indd 136　2024/4/3 16:06:19