第1章 Bolt的安装与配置

Unity 程序设计主要以 C# 作为编程语言，但是对于非计算机专业人士而言，学习一门语言并将其用于游戏开发并不容易。

Bolt 是一款专门为 Unity 而设计的功能强大的可视化编程插件，只要会画流程图，就能编写游戏。Bolt 包含的流图和状态图能帮助用户轻松地实现创意。在播放模式中，可通过实时编辑功能修改图形的任何部分，快速建立原型并进行测试。同时，它还具有预测调试和分析定位问题的功能，通过分析图预测数值并显示图中未使用路径。如果发生错误，则会在图中高亮错误源。Bolt 可以将复杂程序中的名称自动转换为便于阅读的格式，这对于用户来说更容易理解。Bolt 是程序员实现创意的一大利器，受到用户的热烈欢迎。Bolt 最新版本要求 Unity 为 2017.1 或更高版本，并且支持.NET 4.6。

1.1 获取并安装 Bolt

Bolt 安装视频讲解

可以通过如下方法获取 Bolt 插件：在 Unity 编辑器的菜单栏中选择 Window→General→Asset Store 命令，打开 Unity 的资产商店，在搜索界面中输入关键字 Bolt，如图 1-1 所示。

Unity 安装包视频讲解

找到对应的 Bolt 插件页面并显示相关信息，可以下载并导入 Bolt 到当前的 Unity 项目中，如图 1-2 所示。

识别右方二维码下载 Unity 软件，将其导入自己的项目中，这时系统会弹出 Import Unity Package 对话框，单击 Import 按钮，导入 Bolt 插件，如图 1-3 所示。

Unity 安装视频讲解

导入完成后，Bolt 自带的安装向导将自动打开。如果没有出现安装向导，可以在 Unity 编辑器的菜单栏中选择 Tools→Bolt→Setup Wizard 命令，系统将会弹出 Bolt Setup Wizard 对话框，如图 1-4 所示。

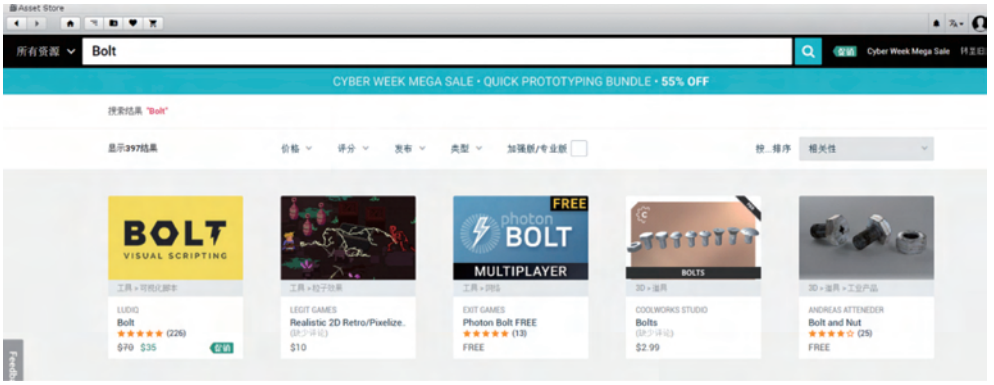

图 1-1　搜索 Bolt 的界面

图 1-2　Bolt 插件页面

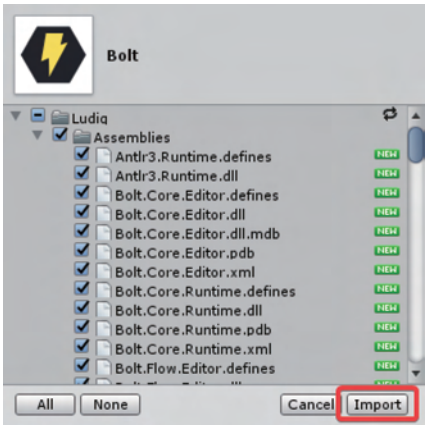

图 1-3　Import Unity Package 对话框

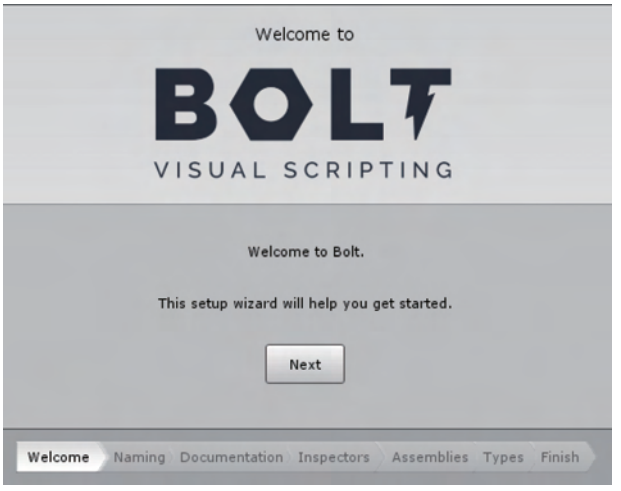

图 1-4　Bolt Setup Wizard 对话框

Bolt 支持两种函数和变量的命名方案：一种是和英语表达一致的命名方式；另一种是符合程序员习惯的命名方式。单击 Next 按钮后，系统会显示 Naming Scheme 对话框，如图 1-5 所示。参照屏幕上的说明，依据个人喜好来配置 Bolt 的命名方案。如果用户不是有丰富经验的程序员，强烈建议选择人类命名(Human Naming)方式。如果想用 Bolt 来学习 C＃语言，则建议选择程序员命名(Programmer Naming)方式。

在文档配置和生成步骤中，可以在 Generate Documentation 对话框内单击 Generate Documentation 按钮，从而让 Bolt 完成文档的生成工作，如图 1-6 所示。

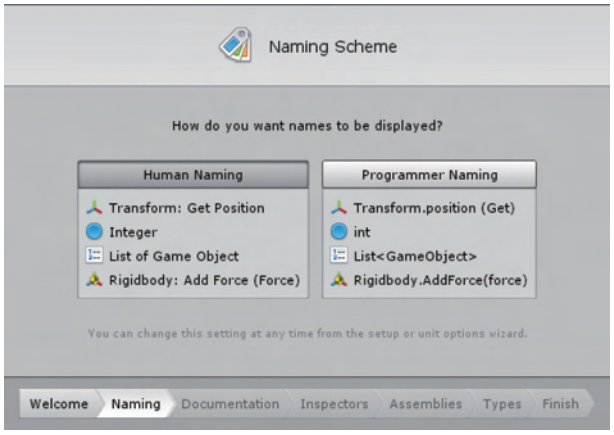

图 1-5 Naming Scheme 对话框

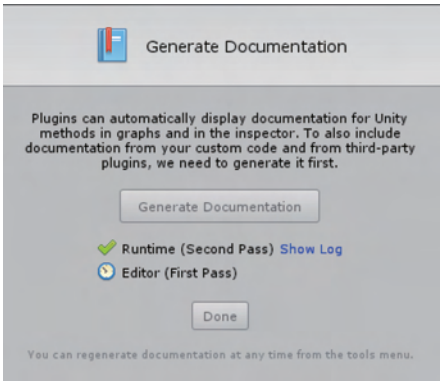

图 1-6 Generate Documentation 对话框

Bolt 将尝试使用 XML 文档来编译文档。如果文档生成失败，不要惊慌。该步骤对于 Bolt 的运行完全是可选的，可以在安装向导中跳过这一步。如果想要了解关于失败的更多详细信息，只需单击 Show Log 按钮。通常，失败往往是由于用户的系统缺少 MSBuild 组件所导致的。

Bolt 的初始化向导接着会显示 Generate Custom Inspectors 对话框，在 Bolt 初始化的时候，单击 Generate Inspectors 按钮生成自定义的检查器，如图 1-7 所示。

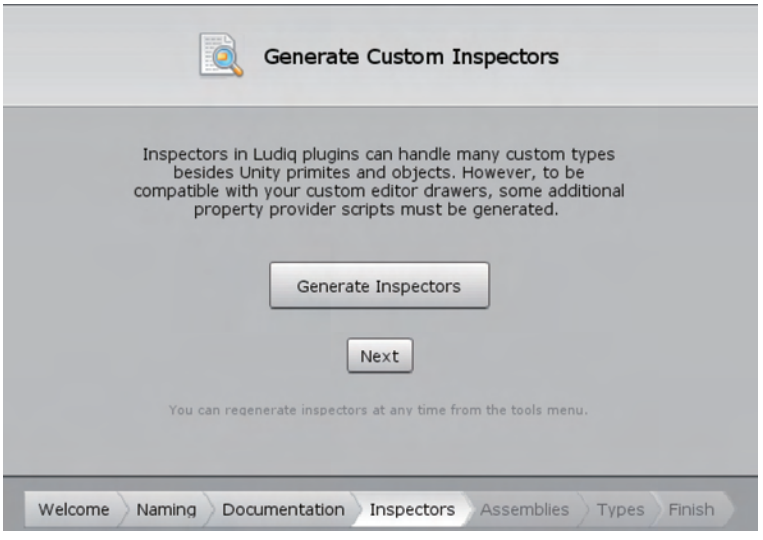

图 1-7 Generate Custom Inspectors 对话框

单击 Next 按钮后，系统会弹出 Assembly Options 对话框，如图 1-8 所示。在此对话框中，如果想在 Bolt 中使用第三方插件，并且这些插件是作为动态链接库形式分发的，那么就需要在 Assembly Options 对话框中添加。

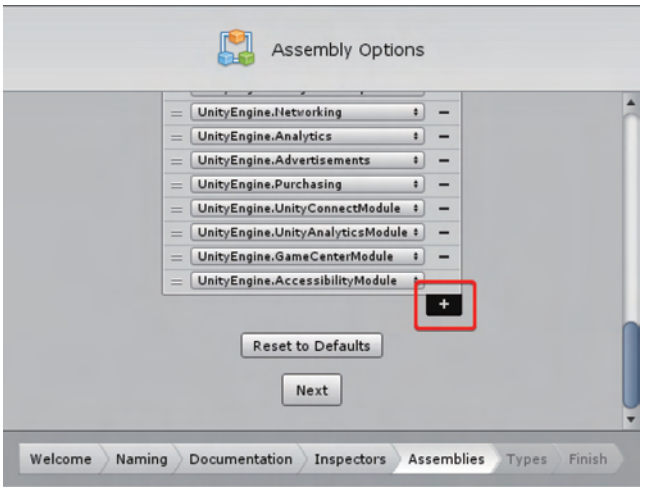

图 1-8　Assembly Options 对话框

系统会显示 Type Options 对话框，如图 1-9 所示。在该对话框中，可以添加任何想在 Bolt 中使用的自定义类型（类或结构）。如果添加的类型继承自 Unity 的对象类（例如 Mono 行为、可编写脚本的对象等），那么它将自动包含被继承的类。

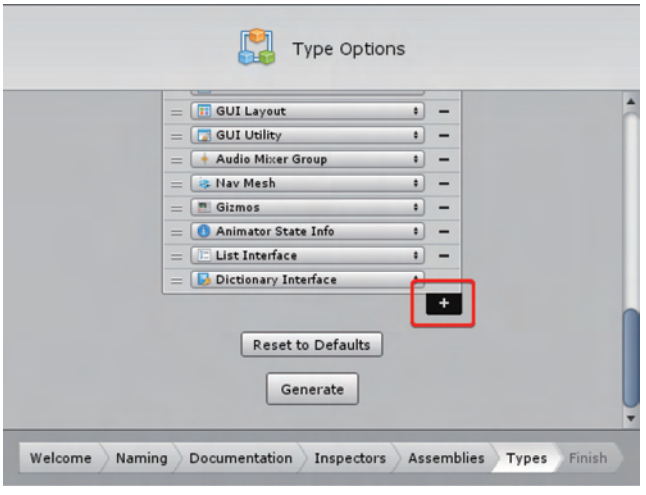

图 1-9　Type Options 对话框

单击 Generate 按钮，系统会显示 Building unit database 的进度对话框，让 Bolt 生成单元数据库，如图 1-10 所示。

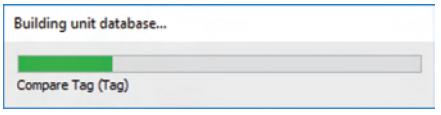

图 1-10　Building unit database 进度对话框

可以通过在 Unity 编辑器的菜单栏中选择 Tools→Bolt→Unit Options Wizard 命令来添加更多的程序集和类型。完成后，系统会显示 Bolt 插

件的 Setup Complete 对话框，单击 Close 按钮，就完成了 Bolt 插件的导入操作，如图 1-11 所示。

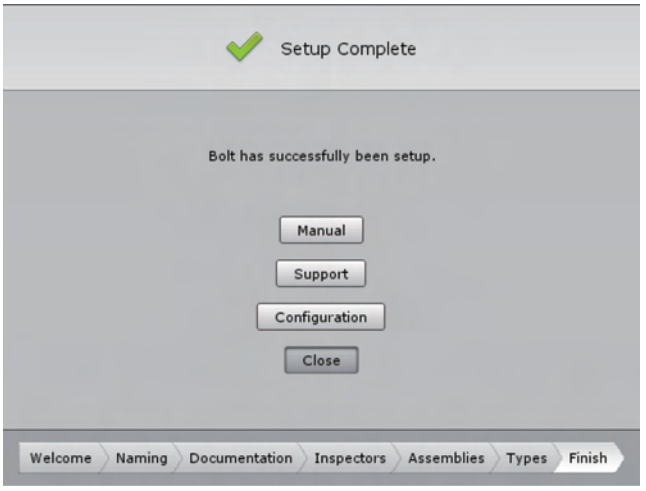

图 1-11 Setup Complete 对话框

1.2 配置 Bolt

Bolt 有许多配置选项，主要分为以下两大类。

(1) Ludiq：用于 Ludiq 框架的一般设置，包括图形设置。

(2) Bolt：用于与流图和状态图相关的设置。

在 Unity 编辑器的菜单栏中选择 Edit→Preferences 命令打开 Unity 编辑器首选项窗口，选择 Ludiq 页面并找到对 Bolt 配置的设置，然后在右侧栏顶部中选择任何一个对应的面板，如图 1-12 所示。

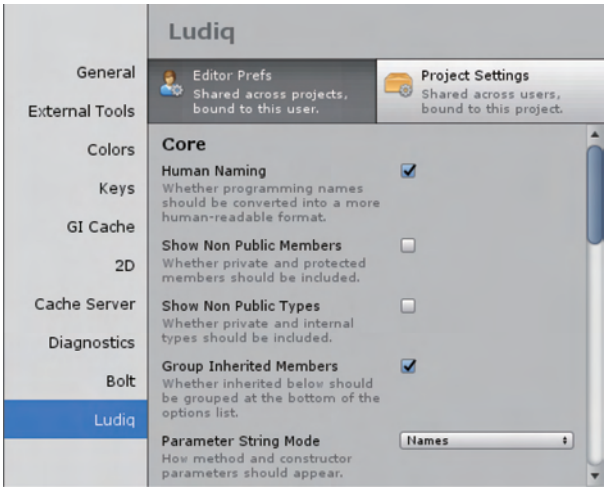

图 1-12 Ludiq 页面

每个面板含有两个子面板：编辑器偏好和项目设置。编辑器偏好通常是编辑器接口选项。它们被保存在每台计算机上，不会与团队共享。所有与 Bolt 一起的项目都共享同一个编辑器偏好。相反，项目设置会与团队共享版本控制。但是，它们属于每个项目，如果在其他项目中使用 Bolt，则不会共享。

每个选项都有一个提示，说明它的作用，所以可以根据自己的喜好来探索和配置 Bolt。如果不确定某个选项的功能，则可以保留它的默认值。

注意：有些选项需要重新启动。如果发现配置更改没有应用，可尝试重新启动 Unity 编辑器。

1.2.1 视图

Bolt 有 3 个主视图，可以在 Unity 编辑器的菜单栏中选择 Window 命令找到它们。

(1) Graph：主图编辑器。

(2) Graph Inspector：节点和其他图元素的检查器。

(3) Variables：定义和编辑变量的视图。

决定放置视图的布局取决于用户自己的喜好。本书给出了 Bolt 建议视图布局，建议为图视图和场景视图分配同样大小的空间，还建议将图检查器添加为 Unity 检查器旁边的另一个选项卡，如图 1-13 所示。

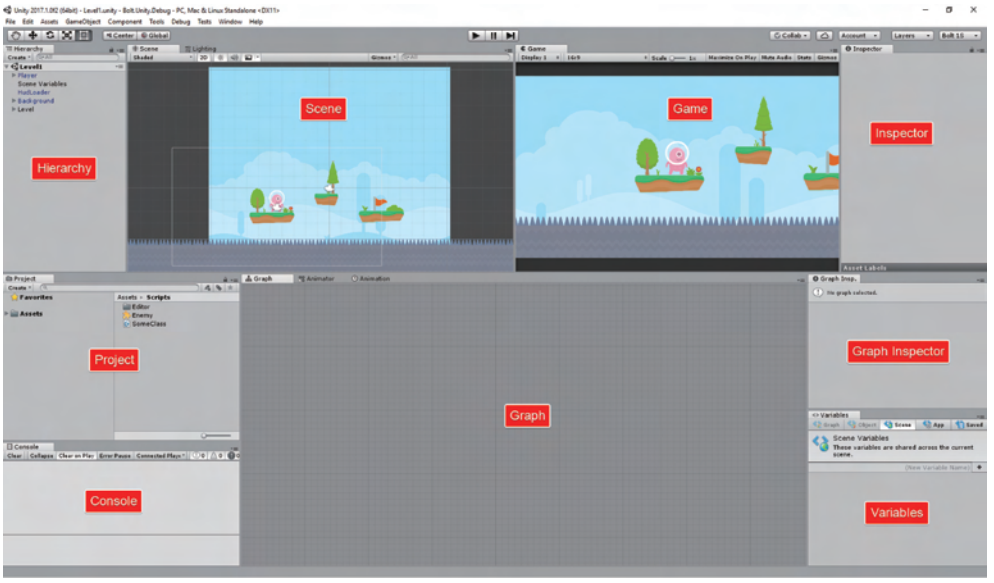

图 1-13 Bolt 建议视图布局

在 Bolt 最大化时视图布局中，可以最大化图窗口到整个屏幕，如图 1-14 所示。这将为图检查器和变量选项卡启用侧栏。最大化使复杂图的可视化变得更加容易。有如下 3 种最大化视图的方法：

(1) 当鼠标在图视图上方时，按 Shift＋Space 键。

(2) 双击图视图背景。

(3) 单击工具栏中的 Maximize 按钮。

侧边栏布局按钮(Sidebar Layout Button)允许重新排序和移动每个面板周围的最大化的视图。

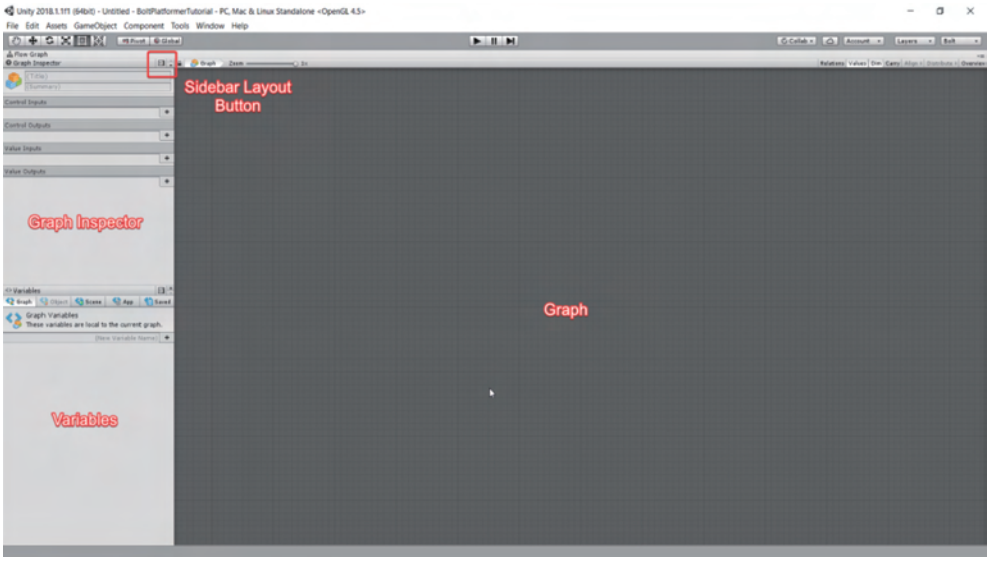

图 1-14 Bolt 最大化时视图布局

提示：如果有双显示器，可以把 Bolt 放在它的第二个显示器上。这为图的编辑提供了很大的空间，并保持了统一布局的完整性。窗口将自动检测到它有足够的空间来显示侧栏。

1.2.2 图视图

Bolt 中的图视图是专门用来编辑流图和状态图的。在打开相应的图之后，在 Bolt 图视图上的工具栏布局中的右上区域出现含有文字的工具栏，如图 1-15 所示。

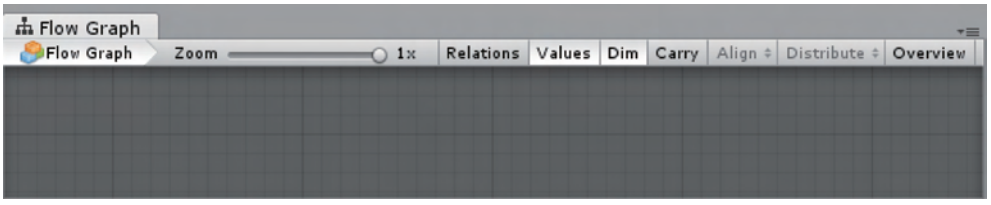

图 1-15 Bolt 图视图上的工具栏布局

在 Bolt 图视图上的工具栏布局的左上角部分，可以看到当前选中图的面包屑路径。该工具栏的中间部分的 Zoom 滑块允许缩放和获得鸟瞰图。当开启 Relations 选项时，将显示单元的内部连接。当开启 Values 选项时，Bolt 将尝试预测值并在图连接中显示它们(仅在流图中)。当开启 Carry 选项时，当前选择的子节点将被拖动。如果想在不手动选择每个节点的情况下重新组织图的很大一部分，这是非常有用的。Overview 按钮将对图进行平移和缩放，以显示窗口区域内的所有元素。一旦在图中至少选择了两个元素，系统就会弹出 Bolt

单元的 Distribute Operation 对话框，该对话框允许执行常见的自动布局操作，如图 1-16 所示。

在图视图中，有两个控制方案（Control Scheme）决定如何平移和缩放。可以在 Unity 菜单栏中选择 Tools→Ludiq→Editor Prefs→Graphs 命令配置控制方案。默认情况下，Control Scheme 的控制方案被设置为 Unreal，如图 1-17 所示。如果用户有触控板的话，则可以试着把设置改成 Unity，这将使得通过手指的操作可以更容易地浏览图。

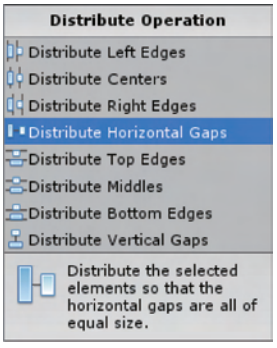

图 1-16 Distribute Operation 对话框

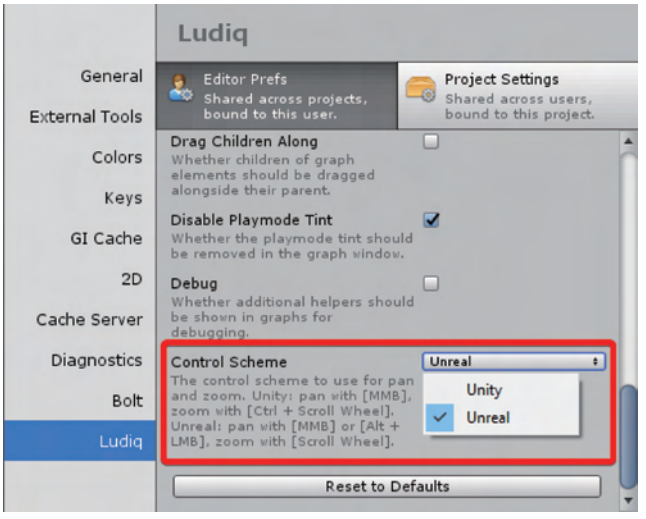

图 1-17 Control Scheme 选择

两个控制方案对应着不同的快捷键配置，如表 1-1 所示。图检查器显示关于当前选中的图元素的信息和设置。

表 1-1 各种快捷键配置

行　　为	Unity 控制方案	Unreal 控制方案
平移	鼠标中键	Alt＋鼠标左键
缩放	Ctrl/⌘＋鼠标滚轮	鼠标滚轮
选择	拖动	
全选	Ctrl/⌘＋A	
快捷菜单	右击 Ctrl＋单击（Mac 平台） Ctrl/⌘＋E	
创建组	Ctrl/⌘＋拖动	
复制	Ctrl/⌘＋C	
粘贴	Ctrl/⌘＋V	
剪切	Ctrl/⌘＋X	
克隆	Ctrl/⌘＋D	
删除	Del	
最大化	Shift＋空格键 双击	

1.3 更新和备份

Bolt 很好地支持了版本的更新和对旧版本用户代码的备份。Bolt 插件更新流程如下：

(1) 备份用户的项目；

(2) 下载并导入新版本；

(3) 通过更新向导更新 Bolt。

当 Bolt 发布更新时，Bolt 的开发者会尽最大努力确保不会引入向后不兼容的更改，这些更改可能会损坏用户的图数据。然而，可能会发生意外的错误，所以应该在下载新版本之前备份项目。更新之前备份项目是用户的责任，Bolt 开发者为用户准备了备份操作。在 Unity 编辑器的菜单栏中选择 Tools→Ludiq→Backup Project 命令，系统会弹出 Backup 对话框，单击 Create Backup 按钮开启备份过程，如图 1-18 所示。

Assets 文件夹将被压缩和保存到与 Assets 文件夹同级的 Backups 文件夹中。可以在 Unity 编辑器的菜单栏中选择 Tools→Ludiq→Restore Backup 命令快速恢复备份文件。在更新之前备份用户的项目文件是绝对必要的，但这并不意味着不能在其他时候备份它。如果不使用版本控制，经常备份是一个很好的做法。本书建议使用版本控制系统(VCS)，如 Unity 协作、Git 或 Subversion 等版本控制服务，而不是这个基本的备份实用程序。

一旦下载并导入了一个新版本的 Bolt，系统会自动打开 Bolt Update Wizard 对话框，如图 1-19 所示。如果没有出现此对话框，可以在 Unity 编辑器的菜单栏中选择 Tools→Bolt→Update Wizard 命令手动打开它。

图 1-18 Backup 对话框

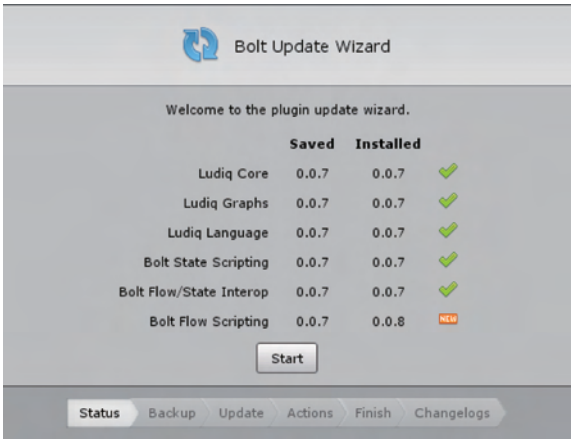

图 1-19 Bolt Update Wizard 对话框

在本例中，Bolt 插件已更新为 0.0.8 版本，Bolt 的版本更新速度比较快，本书编写的时候，版本已经为 1.4.6f3。安装向导的第一步将要求创建备份。如果之前已经创建了备份，可以安全地跳过它。但是，建议在导入新版本之前进行备份，而不是在安装向导中。然后，更新向导将运行自动迁移，并通知是否需要进一步的操作。完整的变更日志可以在向导窗口的末尾找到。

第2章

Bolt的基本概念

2.1 类型

在 Unity 脚本中，一切都是对象（Object）。数字、文本、矢量和 Unity 组件等都是对象。每个对象都有一个类型（Type）用于区分它们代表什么以及它们能做什么。在 Bolt 中，大多数类型都用图标表示。在 Unity 和 Bolt 中有数百种数据类型，不需要熟记每一种类型，一般只需熟悉最常见的类型就能满足大部分游戏开发任务的需要。本书列出了 Bolt 中常见的类型，如表 2-1 所示。

表 2-1 Bolt 中常见的类型

图标	类型	描　述
	数值型	淡蓝色的小圆圈，用于表示带或不带小数点的整数或者浮点数，如 300、0.5 或 13.25
	布尔型	淡紫色的小圆圈，用于表示布尔值（真或假），通常用于逻辑或切换
	字符串型	橙色的小圆圈，用于表示字符串，通常是一段文字，如名字或信息
	字符型	淡黄色的小圆圈，用于表示字符串中的单个字符，通常是字母或数字。很少使用
	枚举型	粉色的小圆圈，用于表示枚举。在下拉列表中的选项就是常见的有限枚举类型。例如，在 Unity 中，“力模式”的枚举可以是“力”“冲量”“加速度”或者“速度变化”

续表

图标	类型	描　　述
  	矢量型	矢量表示一组浮点坐标，例如位置或方向。 单位矢量有以下3个。 • 二维矢量：二维的X和Y坐标； • 三维矢量：有X、Y、Z坐标； • 四维矢量：有X、Y、Z和W坐标，很少用到
	游戏对象	Unity场景中的基本实体。每个游戏对象都有一个名称、位置和旋转的变换信息以及作为可选项的组件列表
	列表	元素的有序集合。元素可以是任何类型，但大多数情况下，列表中的所有元素必须是相同类型的。可以根据列表中的每个元素的从零开始的索引来检索和进行赋值
	字典	一个集合，其中元素具有映射到其值的唯一键。例如，可以通过名称（字符串键）获得年龄（整数值），可以根据键检索和访问每个元素
	对象	绿色的小圆圈，它是一种特殊类型，一般指一个对象。例如，一个节点需要一个对象作为输入时，在输入端口就能看到这样的图标，通常意味着它不关心对象的类型

当需要选择一个类型时，系统将会显示Bolt支持的常用数据类型的选择菜单，如图2-1所示。

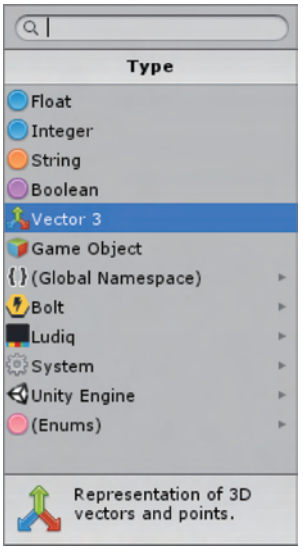

图2-1　Bolt支持的常用数据类型的选择菜单

最常见的类型在列表的顶部可见，枚举出现在列表的底部。所有其他类型都出现在其对应的命名空间(Namespace)分组中，这种分组就像类型的文件夹。所有 Unity 组件和类型都可以在 Unity Engine 命名空间下找到。

2.2 变量

变量是存放数据的容器。每个变量都有名称、类型和值。变量内部的值在运行时可以更改，这就是它们被称为变量的原因。Bolt 中变量的作用域有 6 种，如表 2-2 所示。

表 2-2 Bolt 中变量的作用域

图标	类型	描　述
	流级变量	与局部变量等价
	图级变量	流图实例的局部变量。它的作用域最小，不能在图外访问或修改
	对象级变量	属于游戏对象，可以被游戏对象上的所有图形共享
	场景级变量	可以在当前场景中共享
	应用程序级变量	即使场景发生变化，应用程序级变量也会持续存在。一旦应用程序退出，它将被重置
	存储级变量	即使在应用程序退出之后，保存的变量也会持续存在。它们可以作为一个简单但功能强大的保存系统使用。它们被保存在 Unity 的 player prefs 中，这意味着它们不能直接引用 Unity 对象，如游戏对象和组件

变量视图可以通过选择 Unity 编辑器菜单栏中的 Window→Variables 命令打开，如图 2-2 所示。Graph 选项卡仅在选择流图时启用，而 Object 选项卡仅在选择游戏对象时启用。

在变量视图中添加一个变量的步骤如下：

(1) 选择要添加的变量类型相对应的选项卡。

(2) 在 (New Variable Name) 中输入新变量的名称。

(3) 单击 + 按钮。

(4) 选择 Type (Null) 。

(5) (可选)更改其默认值。

在存储级(Saved)选项卡下有两个子选项卡：初始(Initial) 和保存(Saved)，如图 2-3 所示。在初始选项卡中，定义的变量含有为新游戏创建的初始值。在保存选项卡中，可以看到当前计算机的已保存变量的状态。可以通过手动编辑或者删除来重置这些变量。

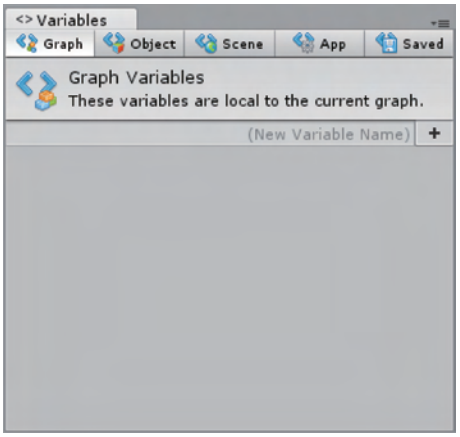

图 2-2 变量视图

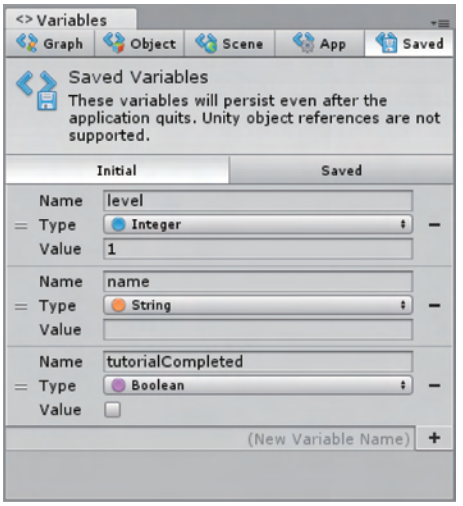

图 2-3 存储级别选项卡下的两个子选项卡

一旦掌握了每种变量的工作方式，就可以通过删除变量视图中的头信息来节省一些屏幕空间。在 Unity 编辑器的菜单栏中选择 Tools→Bolt→Configuration 命令来取消显示变量帮助(Show Variables Help)。

甚至可以在编辑模式中不事先声明变量，通过图在游戏模式中动态创建变量。在图运行的时候，设置不存在的变量的值，会自动创建一个动态变量。例如，自动创建一个名为 gold 的存储级的整数变量，其值为 250(即使之前没有定义它)，如图 2-4 所示。

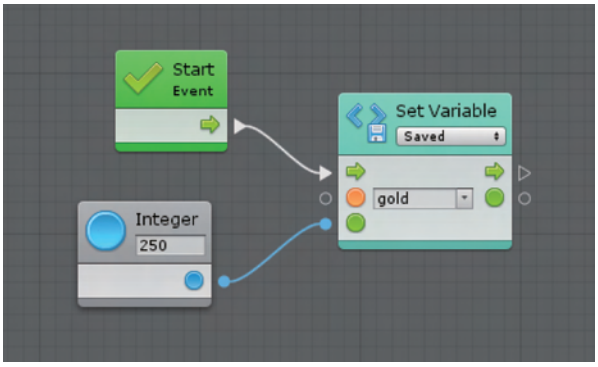

图 2-4 自动创建一个名为 gold 的存储级的整数变量

有关变量单元的更多信息，请参阅单元引用部分。

注意：Bolt 中的所有对象变量都是公共的，可以被其他对象访问。

第3章 Bolt的图

图是逻辑的视觉表示，是 Bolt 的核心。Bolt 有两种类型的图：流图和状态图。流图(Flow Graph)是按特定顺序连接多个功能单元和数值，如图 3-1 所示。执行的顺序就是所说的流程。如果以前使用过虚幻引擎，可能会发现它们类似于蓝图可视化脚本语言。

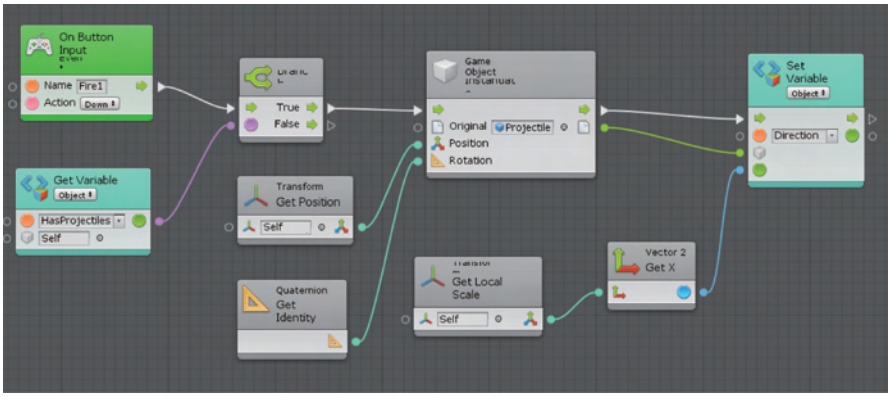

图 3-1　流图

状态图(State Graphs)中包含了不同的状态以及它们之间的转换机制，如图 3-2 所示。每个状态作为一个小的流图。如果以前使用过 PlayMaker 或其他有限状态机系统，那么应该对状态图不那么陌生。

总之，基本上可以通过组合运用这两种图创建任何想要的游戏。

3.1　图的使用

流图是被经常使用的，允许系统在每一帧或事件(如碰撞)发生时执行指定的操作。流图可以访问所有程序逻辑，如分支、循环、数学计算等。流图是面对“当某种情况发生时，应该以什么顺序做什么”之类问题的最好解决方案。

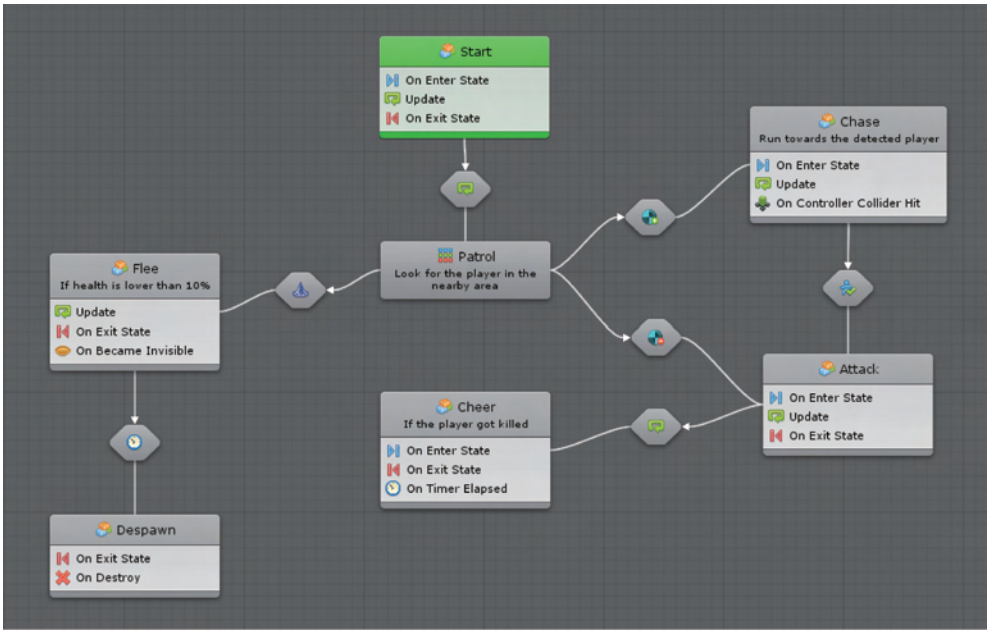

图 3-2 状态图

可以使用状态图创建更“高级”的逻辑，如 AI 行为或者任何需要状态概念的东西；又如有巡逻、追逐和攻击状态的敌人，或有锁定、解锁和打开状态的门。状态图是针对“对象目前的行为表现什么时候应该改变”之类问题的最好解决方案。

可以将这两种图组合起来。例如，状态图中的每个状态节点实际上是一个流图。以下是这两种图形都共享的一些基本概念。

3.2 机器

机器(Machine)是添加到游戏对象上的组件，用于在游戏模式中执行图逻辑。流图对应的机器称为流机器，状态图对应的机器叫作状态机，它们都属于可以添加的 Bolt 类型组件，如图 3-3 所示。

流机器和状态机这两个选项在检查器中都有相同的选项。以添加流机器组件作为示例，如图 3-4 所示。

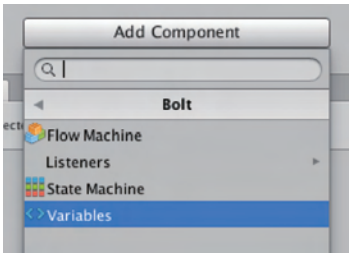

图 3-3 可以添加的 Bolt 类型组件

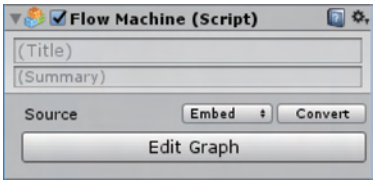

图 3-4 添加流机器组件作为示例

每个图都提供了标题和摘要选项，可以填写相关内容以帮助日后识别它们。它们对功能没有影响，但对资源的有序组织有帮助。单击 Edit Graph 按钮在图视图中打开机器的图。Bolt 有两种类型的源(Source)：嵌入(Embed)或宏(Macro)，如表 3-1 所示。

表 3-1　嵌入和宏的比较

	嵌　入	宏
关系	图嵌入到机器本身中	图是机器引用的宏观资产
可用性	不能为其他机器重用图，但它可以在预制件之间共享	可以在多个机器上重用相同的宏，即使它们不在同一个预制件上
性能	如果删除机器组件，则图将被删除。如果将源切换到宏，则图也会被删除	如果删除机器组件，宏资产仍然存在。如果将源切换到嵌入，图将不会被删除
场景参考	可以在图中引用当前场景中的游戏对象，只要它没有保存为预制件	该图不能引用当前场景中的游戏对象，因为它不“属于”任何场景
预制件	如果在编辑器中实例化预制件，则不应使用该机器	机器可以安全地用于所有的预制件

宏(Macro)是一个可重用的图，可以由多个不同的机器引用。如果将机器的源选项切换到 Macro，则必须告诉机器应该使用哪个宏。

可在编辑器的项目视图的文件夹中右击，弹出快捷菜单后，选择 Create→Bolt→Flow Macro 命令来创建宏，如图 3-5 所示。建议将宏放在顶级的 Macros 文件夹中，但是如何组织完全由用户决定，对功能没有影响。

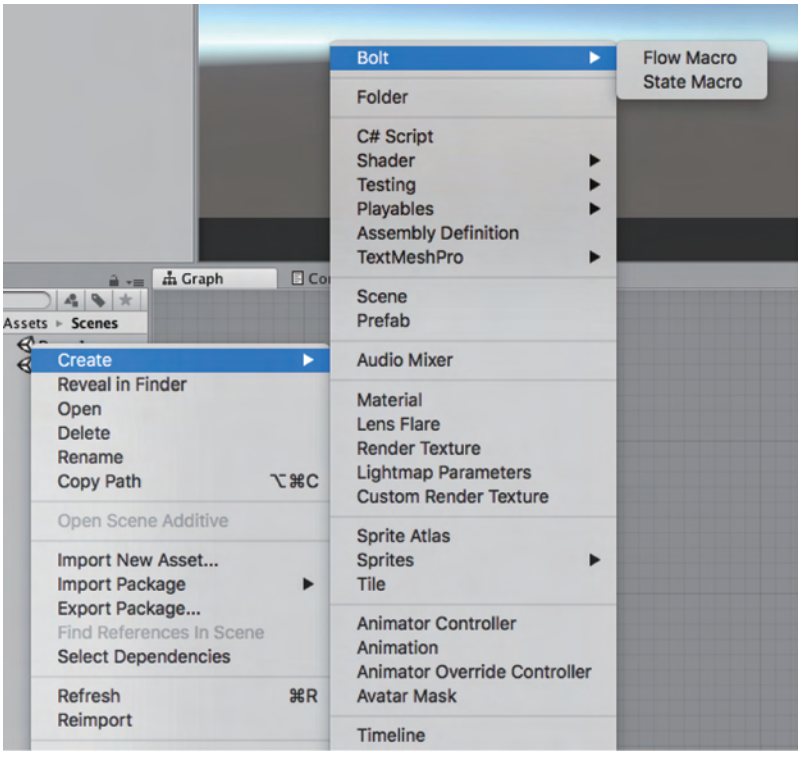

图 3-5　创建 Bolt 宏的快捷菜单

然后，可以简单地拖动新创建的图或使用 Unity 对象选择器将其分配给机器，如图 3-6 所示。

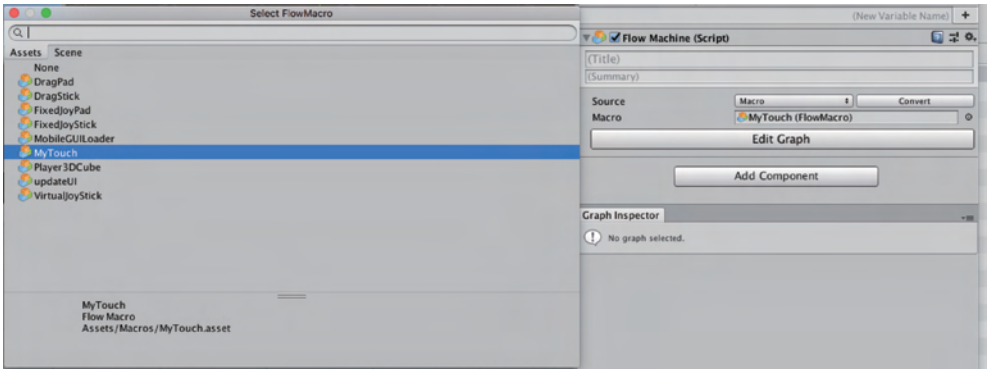

图 3-6　分配宏给机器

当在宏中更改图时，该更改将应用于所有具有该宏的对象。不需要为每个对象的实例都设置唯一的宏，或者复制、粘贴相应的更改。

3.3　源类型的选择

从表 3-1 中看到，宏通常优于嵌入，因为它是可重用的，不绑定到对象，并且可以安全地使用预制件。有一个非常简单的经验法则来决定使用哪种类型的源。大多数情况下，优先使用宏。如果图将在一个或多个对象或场景中重用，那么宏的加载速度更快，维护起来也更容易。对于只在当前场景中使用一次的图形，可以使用嵌入。这将允许使用场景引用，这对 GUI 非常有用。可能发现选择宏还是嵌入图会有点令人困惑，别担心，Bolt 允许随时随地从一种源转换到另一种源。

3.3.1　从宏到嵌入

例如，如果正在使用一个共享状态宏用于 AI 行为，但是随后意识到这个敌人有特殊的行为，可将宏转换为嵌入图，以便独立于其他对象修改它。

要做到这一点，只需单击 Convert 按钮，系统会弹出一个对话框警告提示这个转换将永久覆盖当前的嵌入图，所以在继续之前，应该确保没有问题。

3.3.2　从嵌入到宏

例如，可能会开始在敌人的游戏对象上做一个嵌入图，但是希望同样的逻辑应用到对其友好的非玩家角色上。因此，需要将嵌入图转换为宏以实现重用。

因此，只需单击 Convert 按钮，为新的宏选择一个路径和文件名，Bolt 将把嵌入图形中的所有项复制到宏中（除了在宏中不支持的场景引用），然后机器将自动切换到宏模式并引用该新图。

系统将弹出一个对话框警告提示这个操作会永久地删除当前的嵌入图，所以在继续之前，应该确保没有问题。

3.4 组

组(Group)是组织图中单元的简单框。要创建一个组,在按住 Ctrl 键(Mac 下为⌘)时拖动选定图上的一个区域。可以给每组起一个标题(Title),即使缩小了也能看得清。可以通过双击组的标题来选择组中的所有项。当使用 Unity 控制方案时,可以通过按住 Alt 键并拖动组的标题部分移动组而不移动动它的内容。可以使用图检查器为组提供注释和自定义颜色,如图 3-7 所示。注释仅在图检查器中可见。

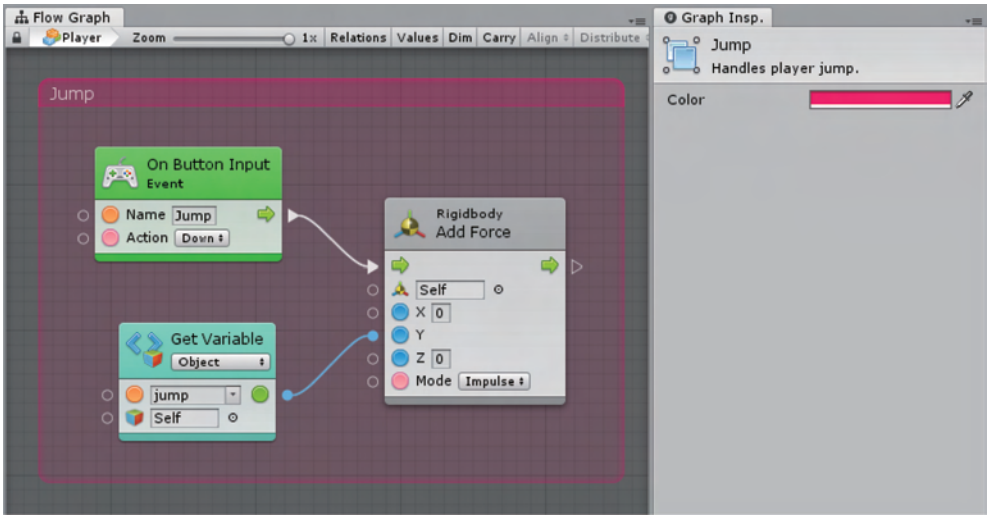

图 3-7 使用图检查器为组提供注释和自定义颜色

第4章
单元和端口

Bolt 流图中最基本的组件就是单元，单元用于执行特定的操作。一般而言，在创建了一个流机器以后，系统默认应该有一个带有 Start 和 Update 事件单元的流图，如图 4-1 所示。

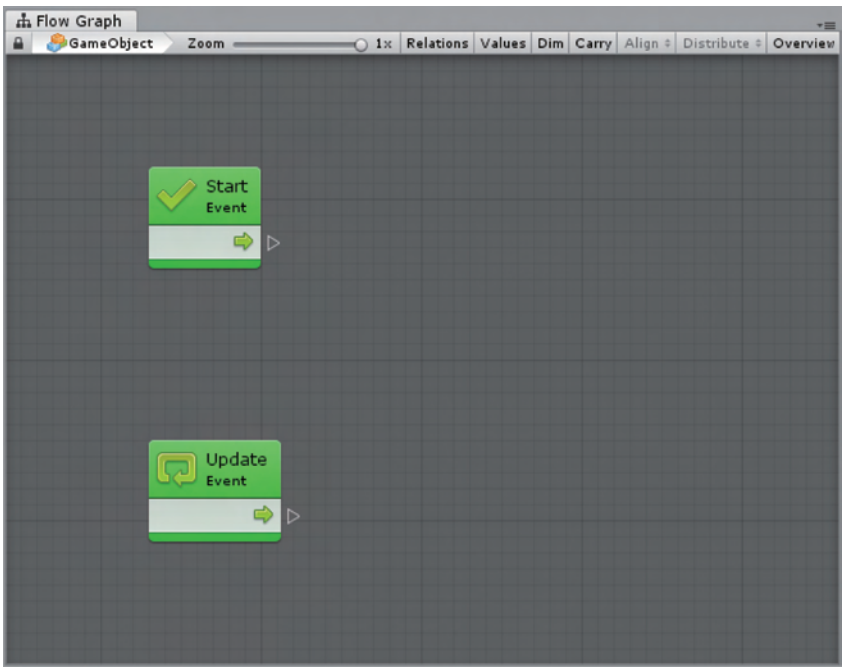

图 4-1　带有 Start 和 Update 事件单元的流图

4.1　单元

单元(Unit)是 Bolt 中最基本的计算节点，它们有时也被叫作节点或操作。它们在流图中表示为具有输入和输出端口的节点。单元可以做很多事情，例如

监听事件、获取变量的值、调用组件和游戏对象的方法等。单元使用连接来指示它们应该按什么顺序被调用，并互相传递值。

4.1.1 创建单元

默认情况下，Bolt 中有超过 23 000 个可用单元，包括了完整的 Unity 脚本 API，以及自定义脚本或第三方插件的所有方法和类。Bolt 中还有一些用于数学、逻辑、变量、循环、分支、事件和协同程序的附加实用程序单元。这些单元在一个简单的、可搜索的模糊查找器中组织得很好，要显示模糊查找器，只需右击空网格中的任何位置，然后可以浏览类别或在顶部字段中搜索以快速找到一个单元，如图 4-2 所示。

Bolt 通过变暗新建的单元来警告新单元的值永远不会被使用，这是一项非常有用的预测性调试特性，可以单击 Graph 视图中工具栏上的 Dim 按钮来切换此特征的开启。在创建某个单元之前，模糊查找器会给出一个预览文档，如图 4-3 所示。例如，对于 Add 单元，可以从模糊查找器中直接知道它的功能和端口。

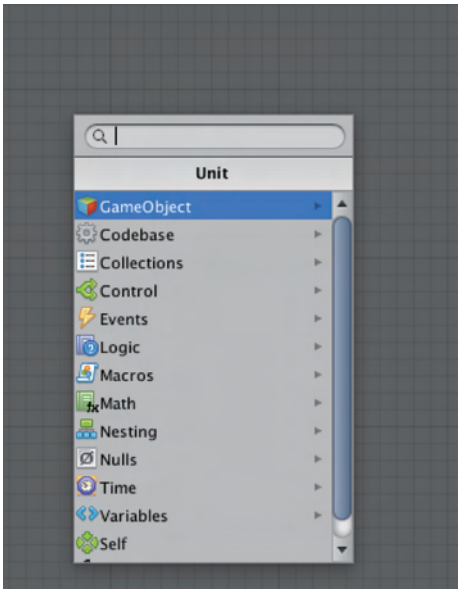

图 4-2 模糊查找器

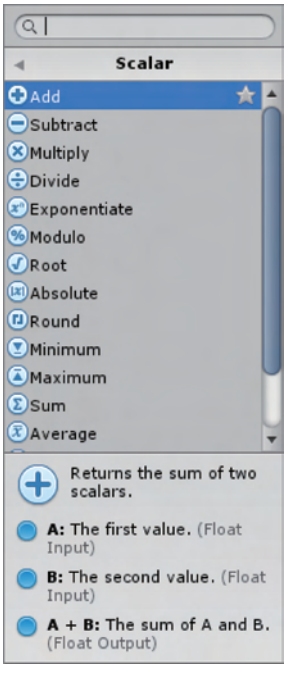

图 4-3 Add 单元在模糊查找器中的预览文档

4.1.2 重载

有些单元有多种变体，称为重载(Overloads)。例如，Bolt 有 4 个用于相加(Add)的单元，一个用于标量，其他的用于二维矢量、三维矢量和四维矢量，如图 4-4 所示。在这种情况下，可以使用它们的应用类型来区分它们。