并作为第一个参数传递给 SetInteger()。

SetInteger()的第二个参数是赋予 AnimationState 的实际值。因为 CharStates 枚举中的每个值都对应一个 int 值，所以在输入如下语句后：

```
CharStates.walkEast
```

实际上使用的是枚举中 walkEast 对应的值。在这个例子中，walkEast 对应 1，但是我们仍然需要通过在变量左侧写入(int)来显式地将其转换为 int 值。需要转换枚举的原因超出了本书的讨论范围，但与 C#语言的底层实现方式有关。

保存脚本并切换回 Unity 编辑器，这样就可以使用所有这些代码。选择指向 player-walk-south 的白色转换箭头，在 Conditions 区域单击加号，分别选择 AnimationState 和 Equals，然后输入值 2，以对应刚刚编写的脚本中的枚举值 2。

现在依次选择 player-walk-west、player-walk-north 以及 player-idle 状态的转换箭头。通过 Inspector 面板给它们中的每一个添加条件，并输入 CharStates 枚举中对应的值：

```
enum CharStates
{
    walkEast = 1,
    walkSouth = 2,
    walkWest = 3,
    walkNorth = 4,

    idleSouth = 5
}
```

当操作每个转换箭头时，记住取消选中 Has Exit Time、Fixed Duration、Can Transition to Self 复选框，并将 Transition Duration (%)设置为 0。

最后一件事，选择每个 player-walk 动画状态对象，将速度调整为 0.6，将每个 idle 动画调整为 0.25。这会使玩家动画看起来恰到好处。

现在你已经设置了游戏所需的大部分玩家动画。单击 Play 按钮，用箭头或 W、A、S、D 键让角色在屏幕上四处移动。

继续操作角色，迈开腿体验一下所有的行走动画。

---

**提示**　如果忘记了 C#中方法的确切参数，Visual Studio 将显示包含这些信息的有用弹出窗口(见图 3-43)。可以按 Enter 键自动完成方法调用。

---