第 5 章

动 画 系 统

Unity 的动画系统基于动画剪辑(Animation Clip)的概念,动画剪辑包含特定对象应如何随时间改变其位置、旋转或其他相关属性的相关信息。Animator Controller 作为动画状态机,负责集中管理由动画剪辑对应的所有状态,负责跟踪当前应该播放哪个剪辑及动画应该何时改变或混合在一起。通过本章的学习,读者将能够熟练掌握 Animator 的使用方法和技巧,为创建生动形象的游戏动画打下坚实的基础。

5.1 动画剪辑

Unity 的动画系统基于关键帧动画的概念,多个关键帧构成了动画的连续变化,这些关键帧动画信息存储于动画剪辑资产中。

通过编辑动画剪辑,开发者可以创建、修改和管理其中的关键帧,从而实现角色、物体或其他游戏元素的动画效果。

动画剪辑在 Animation 窗口中进行编辑,该窗口通过菜单 Window/Animation/Animation 或者快捷键 Ctrl+6 打开,如图 5-1 所示。

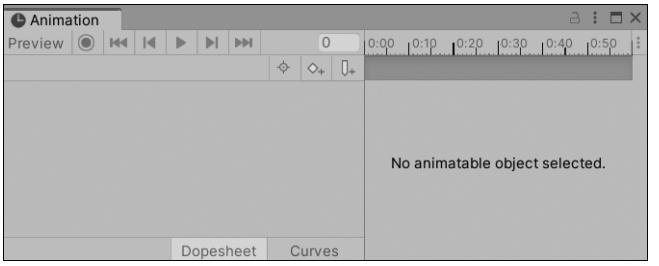

图 5-1 Animation 编辑窗口

开发者可以在时间轴上设置关键帧,然后系统会根据这些关键帧来自动计算中间帧,从而创建出平滑的动画效果。

关键帧可以通过手动创建,也可以通过进入录制模式录制关键帧,本节将介绍这两种关键帧创作方式。

5.1.1 录制关键帧

观察 Animation 窗口，可以看到左上角 Preview 按钮的右侧有一个红色的按钮，单击该按钮即可进入录制模式，如果在录制模式下对场景中的对象进行改动，系统则会自动在时间轴中的当前位置生成关键帧，记录修改的属性。

例如，在示例场景中创建一个 Cube 物体，为其创建一个新的 Animation Clip 资产，进入录制模式后，在 0:00 时间点将其坐标 x 值修改为 5，在 1:00 时间点将其坐标 x 值修改为 10，可以看到，系统自动在时间轴中的对应位置生成了关键帧，如图 5-2 所示。

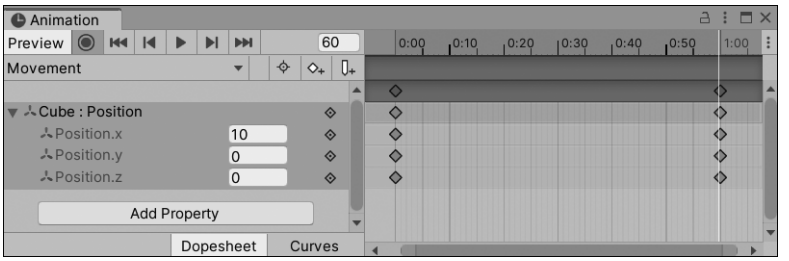

图 5-2 录制关键帧

录制完成后，再次单击录制按钮退出录制模式，单击“播放”按钮，即可进入预览状态，从而可以预览动画。

5.1.2 创建和编辑关键帧

在非录制模式下，如果要在指定的时间点记录对象属性，则需要先手动添加关键帧再进行记录，关键帧可以通过单击 Add keyframe 按钮进行添加，也可以在时间轴中右击并选择 Add Key 选项进行添加，如图 5-3 所示。

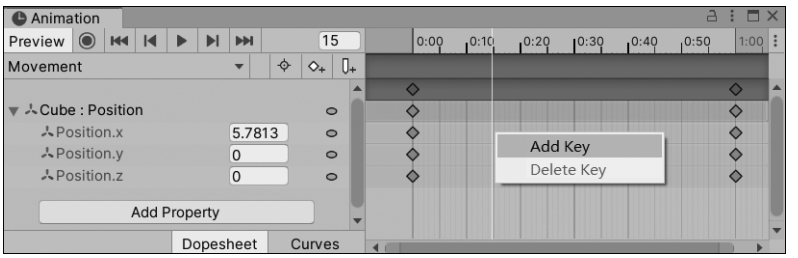

图 5-3 添加关键帧

添加关键帧之后可以在左侧的属性列表中编辑对应属性的值，也可以单击窗口下方的 Curves 按钮，进入曲线编辑窗口，通过编辑曲线的方式编辑属性值，如图 5-4 所示。

5.1.3 外部导入的动画资产

动画剪辑可以是在 Unity 中创建的资产，也可以是由外部导入的 fbx 文件包含的资产。

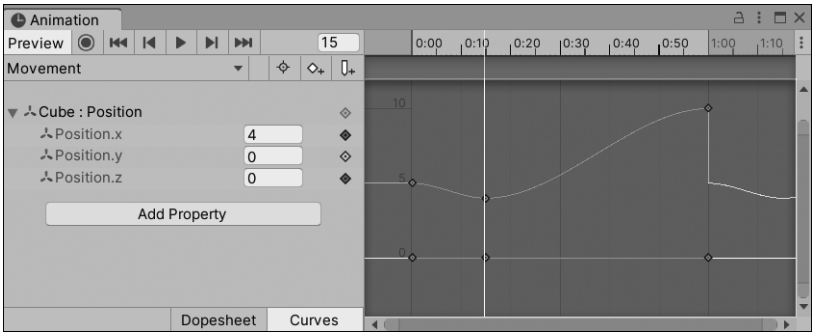

图 5-4　曲线编辑窗口

外部导入的动画剪辑可能包括美术工作人员在 3ds Max、Maya 或 Blender 中创建的动画，也可能是来自第三方库的动画集，如图 5-5 所示。

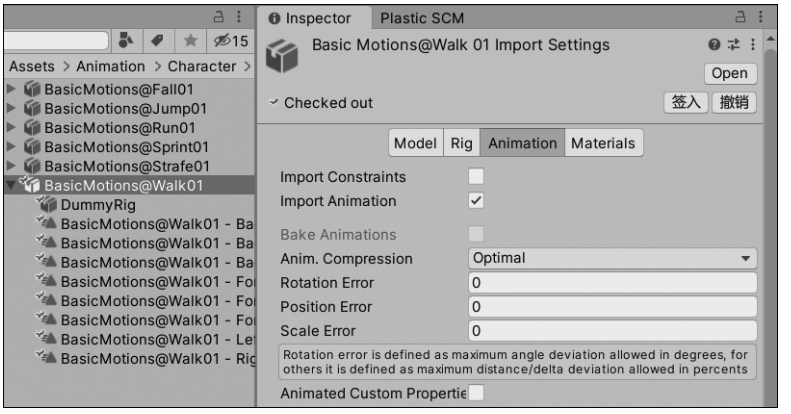

图 5-5　外部导入的动画资产

Import Animation 选项默认为勾选状态，表示导入外部动画，如果取消勾选，则动画资产将不会被导入。

打开 Rig 选项卡，可以看到动画类型有 None、Legacy、Generic 和 Humanoid 共 4 种类型，如图 5-6 所示。Legacy 是旧版本中使用的动画类型，已经不推荐使用。Generic 表示通用动画，Humanoid 表示人形动画，它们的核心区别在于，Humanoid 类型的动画可以通过骨骼重定向复用不同的人物角色动画，并且支持 IK 功能，IK 功能将在后续的章节中进行介绍。

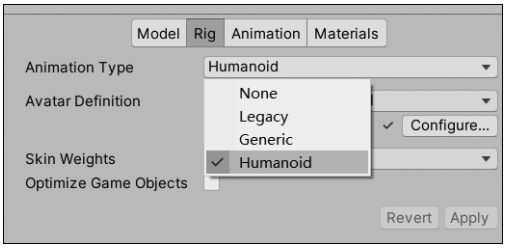

图 5-6　动画类型

在 Animation 选项卡中可以对动画剪辑进行裁剪，如图 5-7 所示，Start 表示起始帧，End 表示结束帧。假设将 Start 设置为 0，将 End 设置为 15，表示截取 0～15 帧的动画。

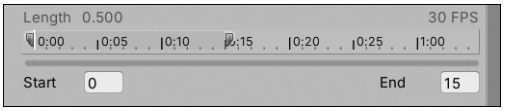

图 5-7 动画裁剪

5.2 动画状态机

动画状态机(Animator Controller)资产通过菜单 Assets/Create/Animator Controller 创建,双击该类型资产或者通过菜单 Window/Animation/Animator 可以打开 Animator 窗口,在该窗口中可以对动画状态机进行编辑。

5.2.1 Animator 窗口

Animator 窗口分为两个主要部分,分别是左侧的 Layers 与 Parameters 编辑区域和右侧的状态编辑区域。

动画参数在 Parameters 视图中进行编辑,包含 Float、Int、Bool 和 Trigger 共 4 种参数类型,如图 5-8 所示。在脚本中访问这些参数并向其赋值,即可实现状态间的切换,4 种类型参数对应的脚本中设置参数值的方法分别是 Animator 类中的 SetFloat()、SetInteger()、SetBool()及 SetTrigger()方法,代码如下:

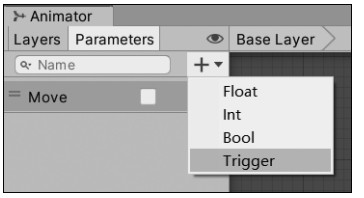

图 5-8 参数类型

```
public void SetFloat (string name, float value);
public void SetFloat (string name, float value, float dampTime, float deltaTime);
public void SetFloat (int id, float value);
public void SetFloat (int id, float value, float dampTime, float deltaTime);
public void SetInteger (string name, int value);
public void SetInteger (int id, int value);
public void SetBool (string name, bool value);
public void SetBool (int id, bool value);
public void SetTrigger (string name);
public void SetTrigger (int id);
```

可以看到,除了可以通过参数名称设置参数值外,还可以通过参数 id 设置参数值,参数 id 通过 Animator 类中的 StringToHash()方法获取,代码如下:

```
public static int StringToHash(string name);
```

在 Layers 视图中可以创建不同的动画层,如图 5-9 所示,使用动画层可以管理身体的不同部位。例如当前有行走和射击两个动画,通过将行走动画放到下身层,将射击动画放到

上身层，可以实现边行走边射击的动画融合效果。

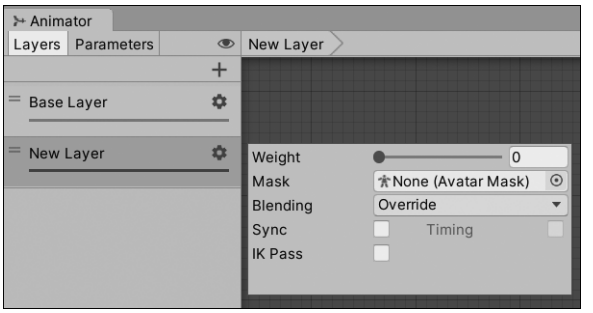

图 5-9　动画层

单击动画层右侧的齿轮按钮打开该层的设置弹窗，Weight 用于设置该动画层起作用的权重值，Avatar Mask 用于设定身体起作用的部位，该类型资产通过菜单 Assets/Create/Avatar Mask 进行创建。Blending 用于设置如何应用该动画层，包含 Override 和 Additive 两种类型，这决定了该层动画是替换上层动画还是叠加到上层动画。Sync 用于设置是否同步其他层的状态机，开启后，该层的状态机与要同步的层的状态机具有相同的结构，但是实际使用的动画剪辑可以不同。IK Pass 用于设置是否启用 IK。

5.2.2　动画状态

动画状态是动画状态机的基本组成部分，通过右击空白区域，选择 Create State/Empty 进行创建，如图 5-10 所示，也可以通过将动画剪辑拖入状态编辑区域进行创建。

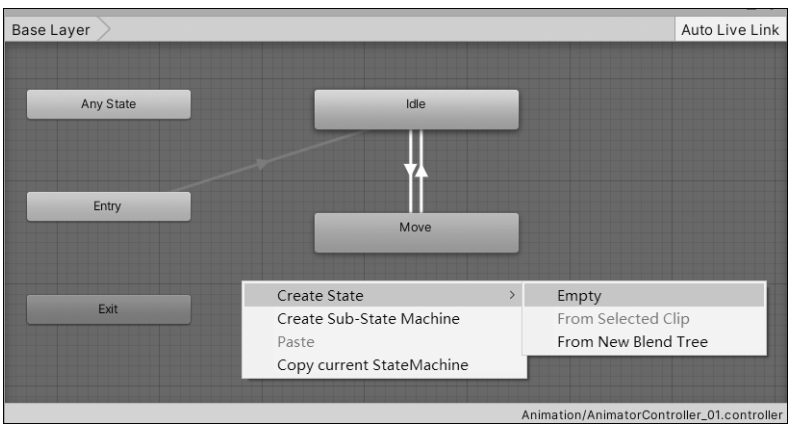

图 5-10　Animator 窗口

选中某个动画状态，在 Inspector 窗口中查看状态的属性，如图 5-11 所示，属性详解见表 5-1。

表 5-1 动画状态属性

属　性	详　解
Motion	该动画状态所引用的动画剪辑
Speed	该动画状态的默认播放速度，可以通过参数的形式进行修改
Motion Time	用于播放该动画状态的动作的时间
Mirror	是否启用镜像动画（仅用于人形动画），例如为一个向左行走的动画开启镜像，得到的是向右行走的动画
Cycle Offset	时间偏移量，当动画被设置为循环播放时，该属性将决定从动画的哪部分开始播放，例如当时间偏移量为 0 时表示从头开始播放，当时间偏移量为 0.5 时表示动画从其一半的位置开始播放
Foot IK	是否启用脚部 IK（仅用于人形动画）
Write Defaults	用于控制 Animator 是否将未动画化的属性写回其默认值
Transitions	源自该动画状态的过渡列表

当角色的行为十分复杂时，状态机将非常庞大，在这种情况下通常会使用子状态机对状态进行管理。子状态机通过在状态编辑区域的空白区域右击并选择 Create Sub-State Machine 进行创建，如图 5-12 所示。双击子状态机即可进入子状态机的编辑窗口，窗口顶部会显示当前正在编辑的子状态机。

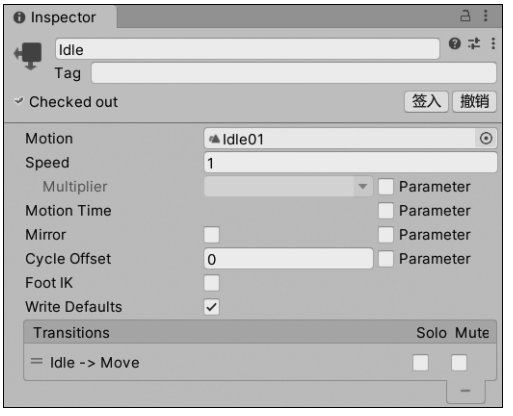

图 5-11 动画状态

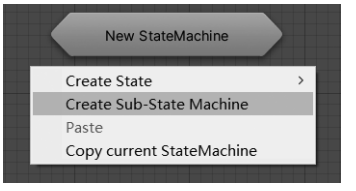

图 5-12 Create Sub-State Machine

5.2.3 动画过渡

状态之间的带有箭头的连线是过渡线，表示状态过渡，单击过渡线，在 Inspector 窗口中编辑状态过渡的属性，如图 5-13 所示，这些属性决定了状态机从一种状态过渡到另一种状态的混合时长，以及应该在什么条件下触发过渡，例如从 Idle 状态过渡到 Move 状态的过渡条件是当 Move 参数变为 true 时。

动画过渡的属性详解见表 5-2。

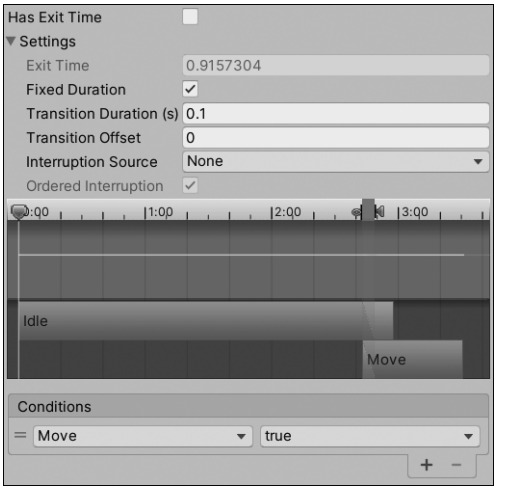

图 5-13　动画过渡

表 5-2　动画过渡属性

属　　性	详　　解
Has Exit Time	当动画过渡不依赖过渡条件时启用此选项，表示经过指定的标准化时间后进行过渡
Exit Time	当勾选 Has Exit Time 时，此值表示进行过渡的标准化时间
Fixed Duration	当值为 true 时表示过渡时间以秒为单位，当值为 false 时表示过渡时间以源状态的标准化时间进行表示
Transition Duration	过渡持续时间，当 Fixed Duration 为 true 时此值表示秒数，当 Fixed Duration 为 false 时此值表示标准化时间
Transition Offset	过渡到的目标状态的起始播放时间偏移值
Interruption Souce	控制过渡中断的类型
Ordered Interruption	确定当前过渡是否可在不考虑顺序的情况下被其他过渡中断，默认值为 true

Interruption Souce 的类型及详解见表 5-3。

表 5-3　Interruption Souce 详解

类　　型	详　　解
None	不添加任何过渡
Current State	将当前状态的过渡排队
Next State	对下一状态的过渡进行排队
Current State then Next State	将当前状态的过渡排队，然后将下一状态的过渡排队
Next State then Current State	将下一状态的过渡排队，然后将当前状态的过渡排队

如果将 Interruption Souce 设为 None，则在状态过渡期间不会因任何其他状态过渡的触发而中断，但是如果将其设为 Current State，则在状态过渡期间可能会因为源状态上的一些状态过渡触发而中断，为了更深入地理解，在示例状态机中添加 Jump、Talk 两个新状态，如图 5-14 所示，选中 Idle 状态，过渡优先级设置如图 5-15 所示。

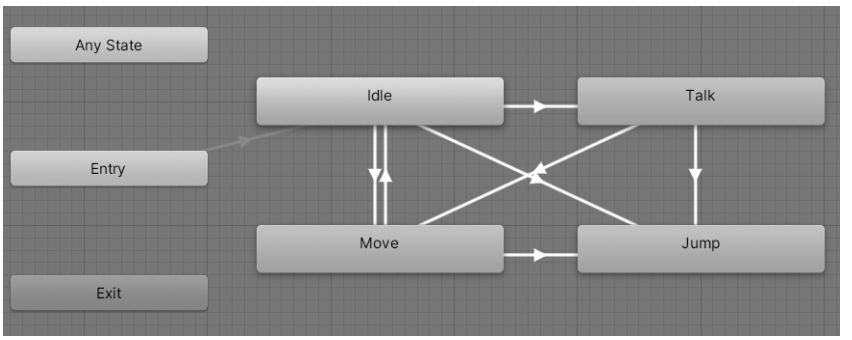

图 5-14　状态机

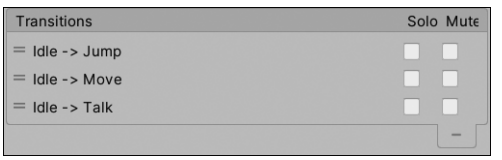

图 5-15　过渡优先级

在 Idle 至 Move 的状态过渡中，将 Interruption Souce 设为 Current State，那么在该状态过渡期间，如果 Idle 至 Talk 的状态过渡被触发，则 Idle 至 Move 的状态过渡将保持不间断，但是，如果 Idle 至 Jump 的状态过渡被触发，则 Idle 至 Move 的状态过渡将立即中断，开始向 Jump 状态过渡。

当然，以上是在 Ordered Interruption 属性为 true 的前提下，如果 Ordered Interruption 属性为 false，则 Idle 至 Talk、Idle 至 Jump 均会中断 Idle 至 Movc 的状态过渡，如果它们在同一帧中触发，则状态会向 Jump 过渡，因为 Idle 至 Jump 具有更高的优先级。

如果将 Interruption Souce 设置为 Next State，则 Idle 至 Talk、Idle 至 Jump 不会中断 Idle 至 Move 的状态过渡，但是当 Move 至 Jump 的状态过渡被触发时，Idle 将立即开始向 Jump 状态过渡。

Current State then Next State、Next State then Current State 的类型则是为了更好地控制，在这种情况下，状态机将先分析前一种状态的过渡再分析后一种状态的过渡。例如当设置为 Current State then Next State 时，在 Idle 至 Move 的状态过渡期间，如果 Idle 至 Talk、Idle 至 Jump、Move 至 Jump 的状态过渡同时被触发，则状态机会开始向 Jump 状态过渡。

5.2.4　混合树

混合树用于实现多个动作之间的平滑混合，每个动作对最终效果的影响由混合参数决

定。混合树通过在空白区域右击并选择 Create State/From New Blend Tree 进行创建。

例如，当前有 Idle、Walk、Run 共 3 个动画，分别是角色的静止动画、行走动画和奔跑动画，为了取得从静止到走跑或者从走跑到静止的良好的混合效果，将它们放在一个混合树中，通过一个 float 类型的参数进行控制，如图 5-16 所示。

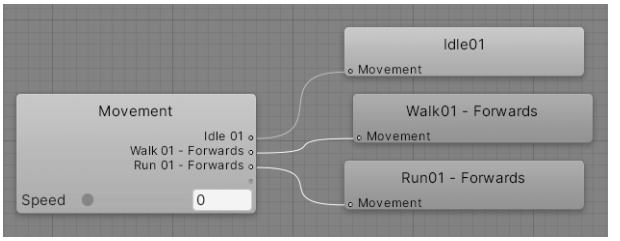

图 5-16　混合树

选中混合树，可以在检视窗口为各个动画设置阈值，如图 5-17 所示，当参数为 0 时，角色将处于静止状态，当参数从 0 逐渐过渡到 2 时，角色也将从静止动画逐渐过渡到行走动画，当参数逐渐过渡到 3.75 时，角色动画也将逐渐过渡到奔跑动画。

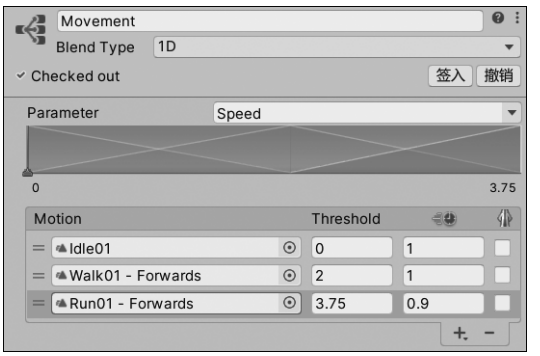

图 5-17　混合树阈值

以上是 1D 混合类型的混合树，假设角色有不同方向的移动动画，例如向前走、向左前方走、向右前方走等，移动方向需要通过两个参数进行控制，如图 5-18 所示，此时便需要 2D 混合类型的混合树。

2D 混合类型有多种，包括 2D Simple Directional、2D Freefoom Directional 和 2D Freeform Cartesian，它们具有不同的用途。

2D Simple Directional 适用于运动表示不同的方向，但是在同一方向上不会有多个运动的情况。假设在一个方向上有多个运动，例如向前移动有向前走和向前跑，这种情况适合使用 2D Freeform Directional 混合类型。2D Freeform Cartesian 适用于运动不表示不同方向的情况，两个参数表示不同的概念，例如速度和角速度。

Pos X 和 Pos Y 的取值可以通过 Compute Positions 下拉菜单选择不同的方式进行计算，如图 5-19 所示。Velocity XZ 表示根据运动的 Velocity X 和 Velocity Z 分别设置 Pos X

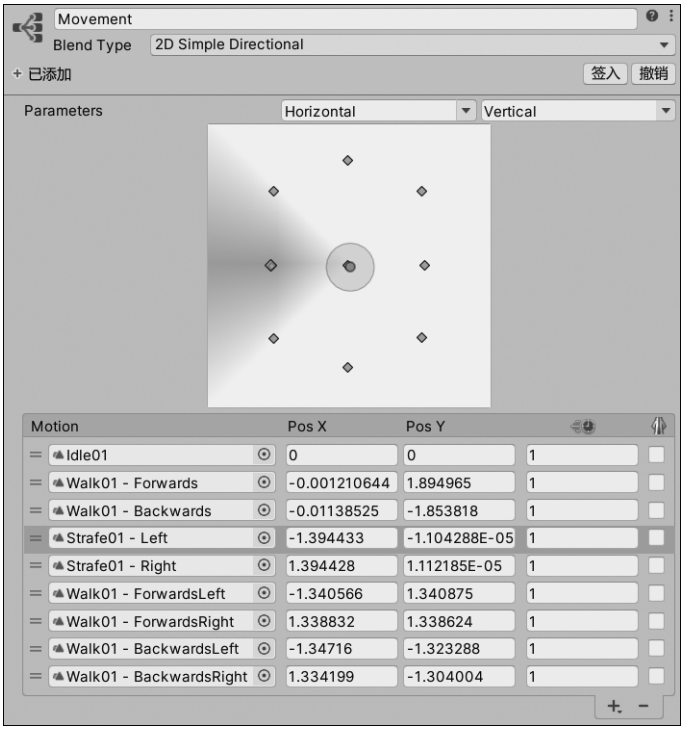

图 5-18　2D Simple Directional

和 Pos Y，Speed And Angular Speed 表示根据运动的角速度（弧度/秒）和速度分别设置 Pos X 和 Pos Y，而 X Position From 表示仅根据其中一项设置 Pos X，PosY 保持不变，Y Position From 同理。

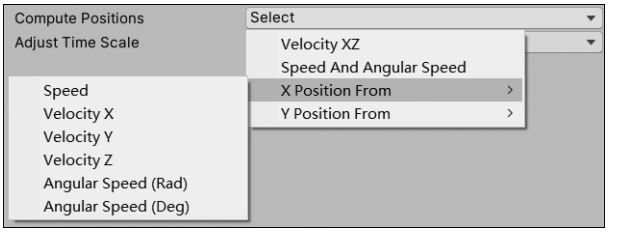

图 5-19　Compute Positions

5.3　动画事件

动画事件通常用于在动画播放的特定时间点触发某些操作，例如播放声音、切换动画状态、调用函数等。动画事件的实现方式有多种，下面介绍两种动画事件的实现途径。