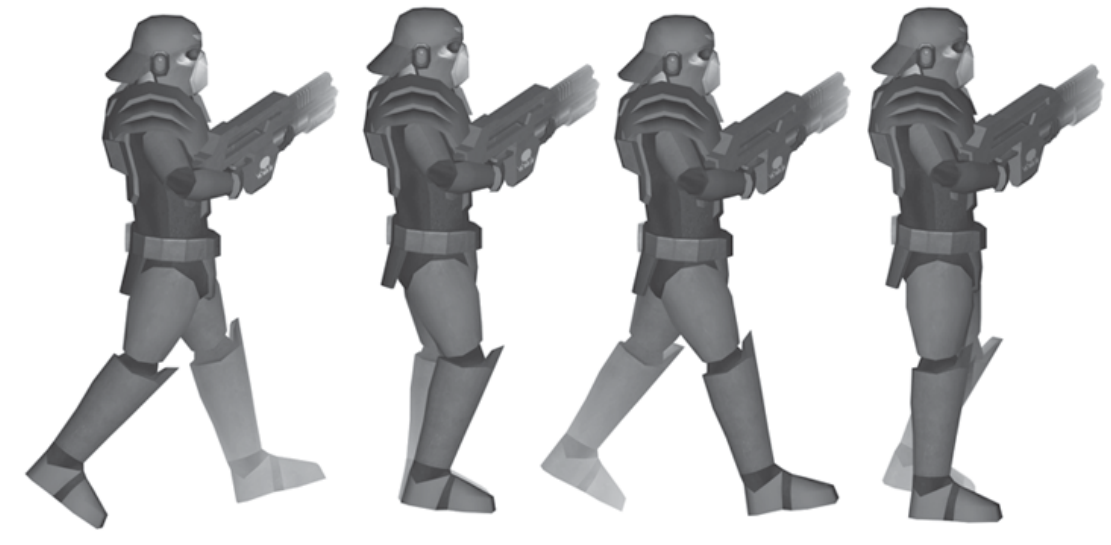
# 骨骼动画



前一章介绍了蒙皮网格的基础知识，以及如何从.x文件加载它们。除了增加的骨骼层次结构，这些网格仍然没有动画，因此没有比常规静态网格更有趣的东西。这将在本章中进行更改，您将学习如何加载动画数据并将其应用于蒙皮网格。本章包括以下内容：

关键帧动画基础知识

加载动画数据

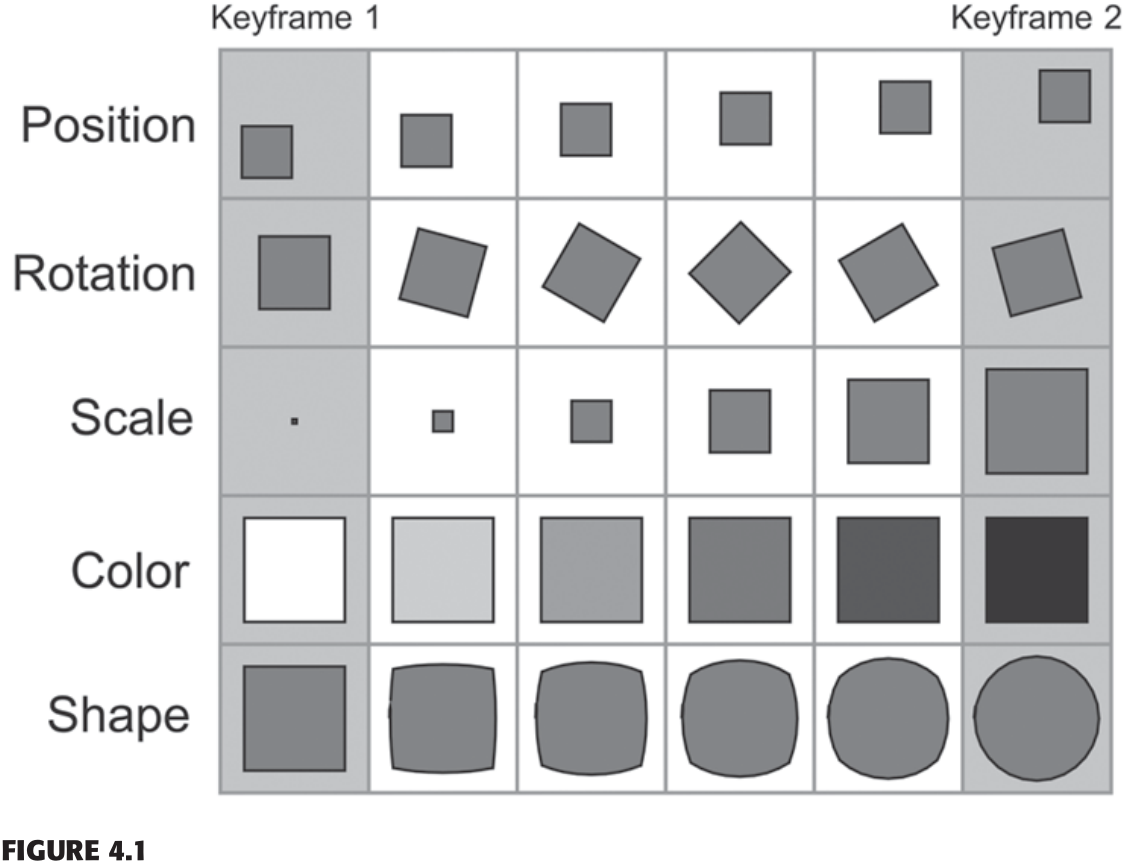
ID3DXAnimationController

让多个控制器影响同一个网格

正如您可能知道的那样，电影由几个快速运行的静止图像组成，因此产生了运动图像的错觉。这些静止图像称为帧。每个帧都有一个特定的时间位置，以及“世界”在这个特定时间步骤中的表现。

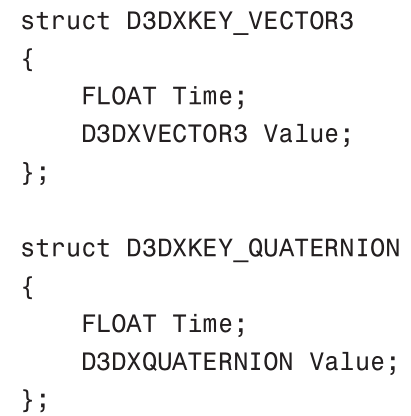
关键帧动画已存在很长时间了。事实上，它曾被用于最初的电视动画片中。它的工作方式是高级动画师会在两个不同的时间步骤绘制两个包含卡通人物姿势的图像（这些帧是所谓的关键帧）。然后，高级动画师将这些关键帧提供给初级动画师并让他填写其中的其余帧，这个过程也称为 “补间（来自“中间”）“。---这一段翻译有点懵。。。。

在许多情况下，关键帧由公司A中的一位艺术家绘制，然后其余的框架由公司B中的另一位艺术家绘制（其甚至可能位于完全不同的国家）。例如，辛普森的每一集都主要来自印度。使关键帧变得如此强大的原因在于它可以应用于几乎任何东西（见图4.1）。



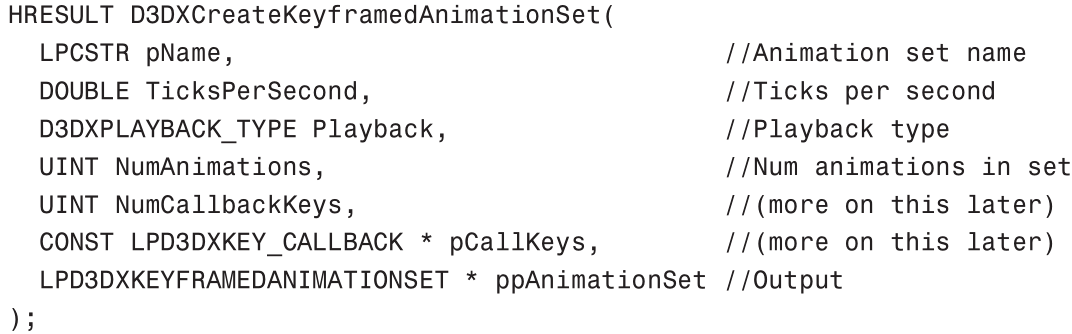
使用关键帧技术的几个例子。

在图4.1中，两个关键帧以灰色背景突出显示。现在花点时间尝试想象如果在这两个关键帧上同时应用所有变换，那么小方块会是什么样子。在计算机动画中，这种技术非常强大。即使时间步长在长度和规律性上变化（如在游戏中通常的帧速率），仍然可以使用该技术基于两个关键帧来计算当前帧。DirectX使用这两个结构来描述关键帧。D3DXKEY\_VECTOR3结构可以描述平移和缩放关键帧。另一方面，旋转由D3DXKEY\_QUATERNION结构描述，因为使用欧拉角可以导致万向节锁定。当一个物体沿一个轴旋转时，会发生万向节锁定，使其对齐x，y和z轴中的两个。一旦发生这种情况，就会失去一个自由度，而且无论对对象执行什么旋转操作，都无法逆转这种情况。四元数是一个比欧拉角更安全的选择（虽然有点难以理解）。无论如何，这是两个DirectX关键帧结构：

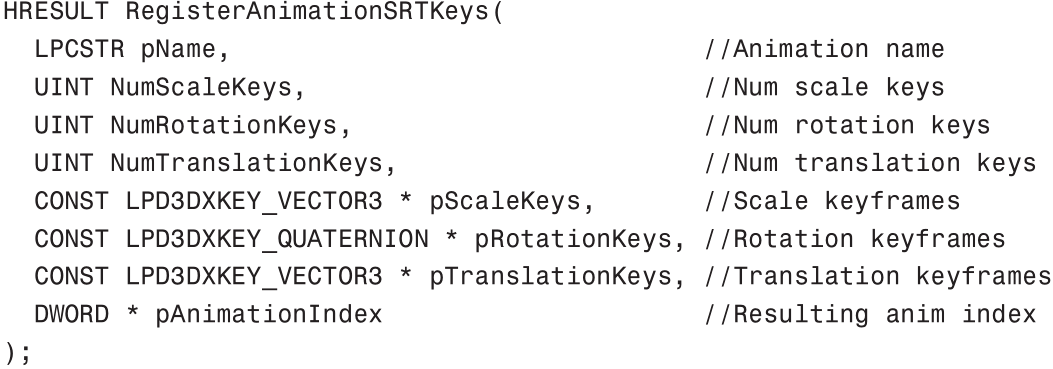


如您所见，它们都包含时间戳以及描述当时对象的平移，缩放或旋转的值。如果您现在还不熟悉四元数，不要担心;当你读到第6章时，我们会更深入地探讨这些问题。这些关键结构的时间是动画滴答（刻度）（。。。看怎么翻译了），而不是秒。动画使用的刻度数等于动画使用的时间分辨率。接下来，看看如何将这些关键帧组合成动画!

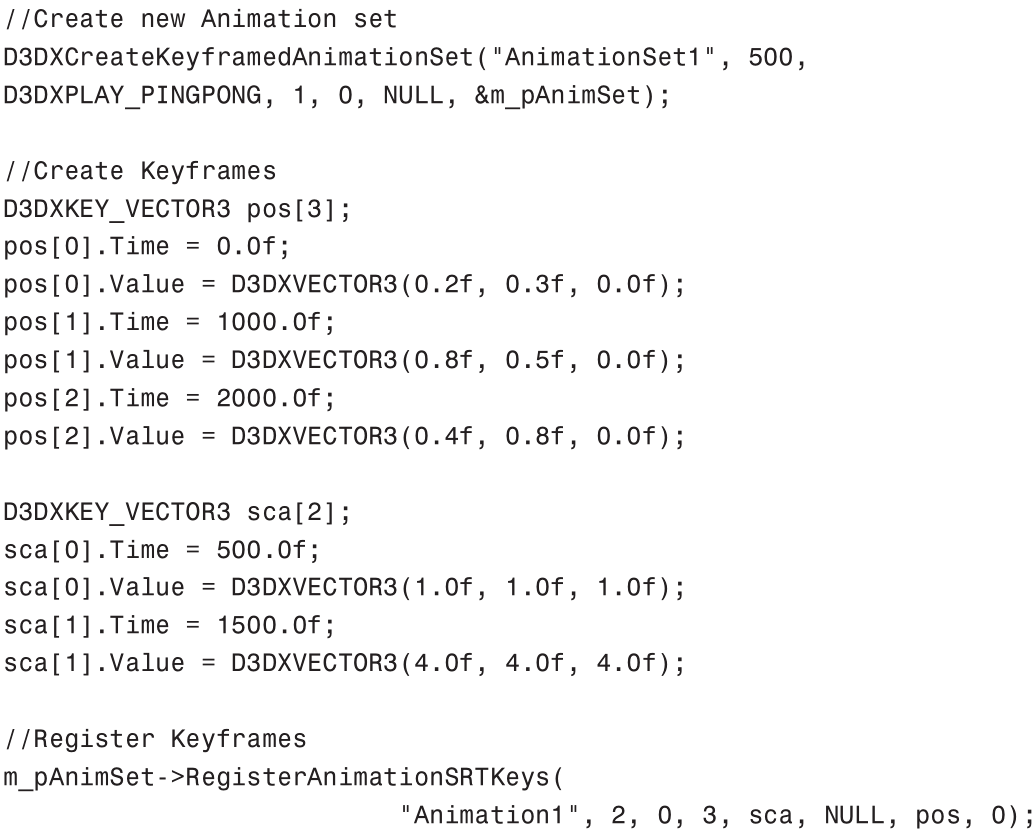
动画集只是动画的集合，其中动画是关键帧的集合。现在您已经了解了关键帧动画背后的理论，是时候转向实际方面了。在本节中，您将了解ID3DXKeyframedAnimationSet接口。该接口包含许多不同的功能，其中一些将在本章中使用。其他的将在以后使用动画回调等内容时使用。查看DirectX文档以获取完整的功能列表。要创建ID3DXKeyframedAnimationSet对象，请使用以下函数：



这里最有趣的参数是回放类型（playback type），它可以是以下之一：D3DXPLAY\_LOOP，D3DXPLAY\_ONCE或D3DXPLAY\_PINGPONG（乒乓选项将向前播放动画，然后向后播放，然后重新开始）。一旦创建了空动画集，您需要做的就是用一些新的关键帧填充它，您可以使用以下函数来完成：



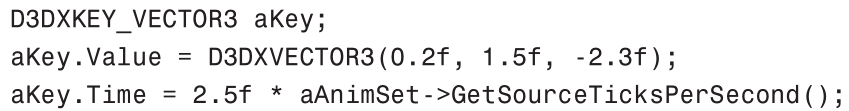
简单！ 创建了缩放，旋转和平移关键帧的数组（使用D3DXKEY\_VECTOR3和D3DXKEY\_QUATERNION结构）并使用此函数添加到动画集。在实际操作中，可以通过以下方式使用这些功能：



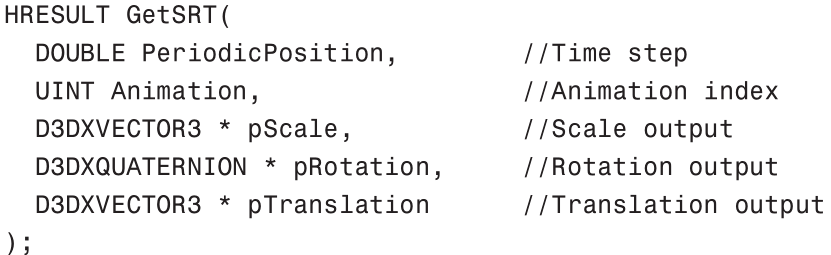
此代码使用位置和缩放元素创建具有乒乓播放的动画序列。要计算某个动画关键点的时间戳，您需要检索动画的每秒滴答（刻度）数量。为此，您可以使用ID3DXKeyframedAnimationSet接口中定义的以下函数：



这个函数可以像这样用来计算一个新的动画键的时间戳:



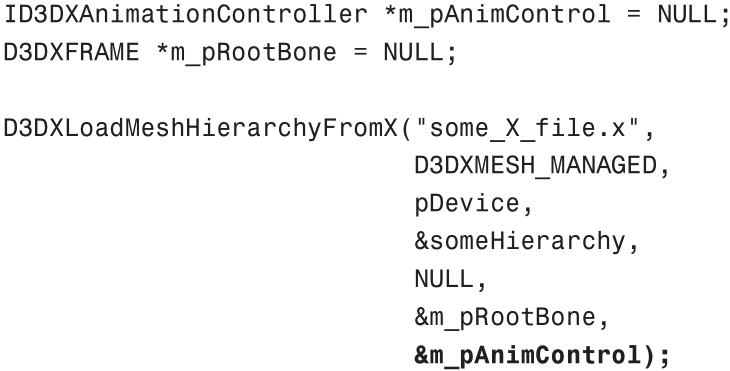
这段代码创建一个新的位置键，并将键的时间戳设置为2.5秒。很少需要像这样手动创建动画键，但是在下一章介绍动画回调事件时，关于如何创建动画键的知识将派上用场。无论如何，一旦创建了这样的动画，您需要一种方法来从动画中读取任何给定时间步长的位置、旋转和缩放数据。为此，您可以使用以下功能：



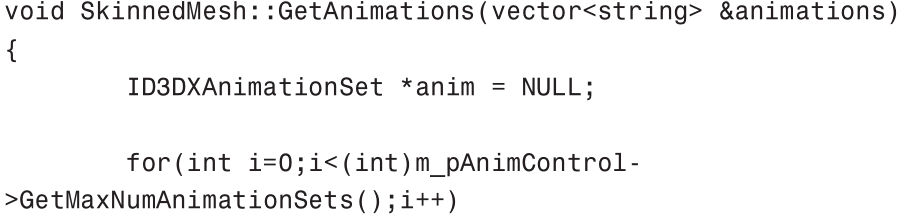
此函数将时间步长和动画索引作为输入（请记住，动画集可以包含多个动画）。 作为此函数的输出，您可以获得缩放，旋转和平移元素。

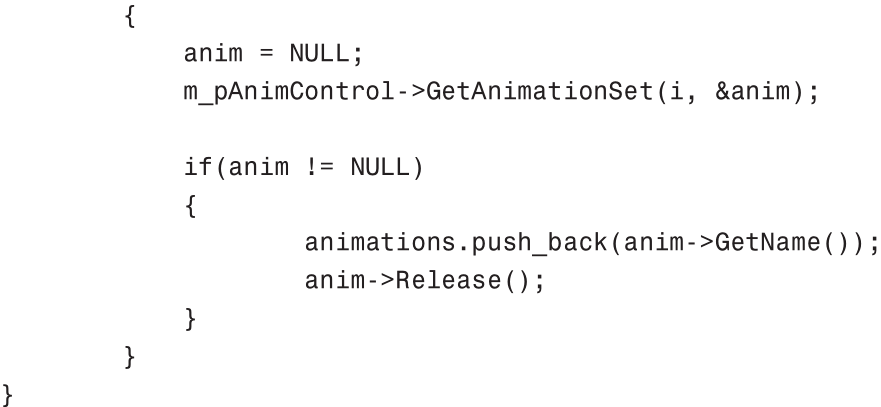
你已经知道如何创建一组不同的动画。为什么需要动画控制器接口? 好吧，ID3DXAnimationController接口控制着关键帧动画的所有方面。它涉及从设置活动动画到混合多个动画，动画回调等等的所有内容（更多内容见第5章）。在本章中，您将学习如何获得这个接口以及控制角色动画的基本方面所需的函数。

您已经接触过上一章中用于加载ID3DXAnimationController对象的函数。如果您还记得，D3DXLoadMeshHierarchyFromX（）函数用于从.x文件加载骨骼层次结构。此函数的输出参数之一是ID3DXAnimationController对象，其中包含与模型一起存储的所有动画数据。这样的数据加载如下：

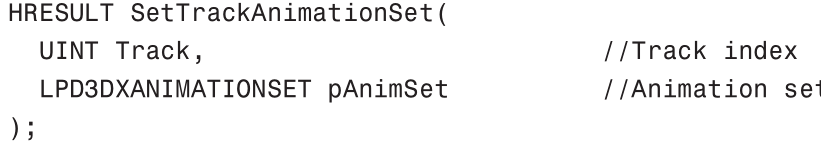


此代码将骨骼层次结构及其网格和存储加载到m\_pRootBone变量中。它还会在m\_pAnimControl变量中加载影响此骨骼层次结构的动画数据。现在通过此动画控制器，您可以为角色设置活动动画以及更新活动时间等。ID3DXAnimationController包含多个动画集（如上一节所述）。这些动画集与之前创建的动画集之间的区别在于它们直接连接到角色骨骼的变换矩阵。以下是如何获取存储在ID3DXAnimationController中的任何动画：

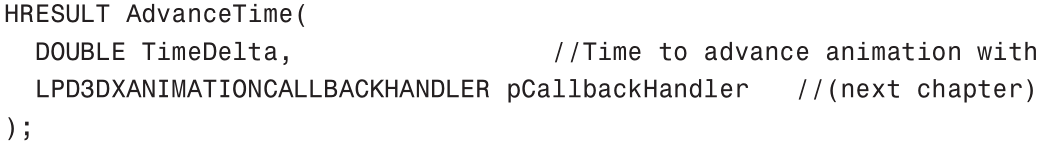


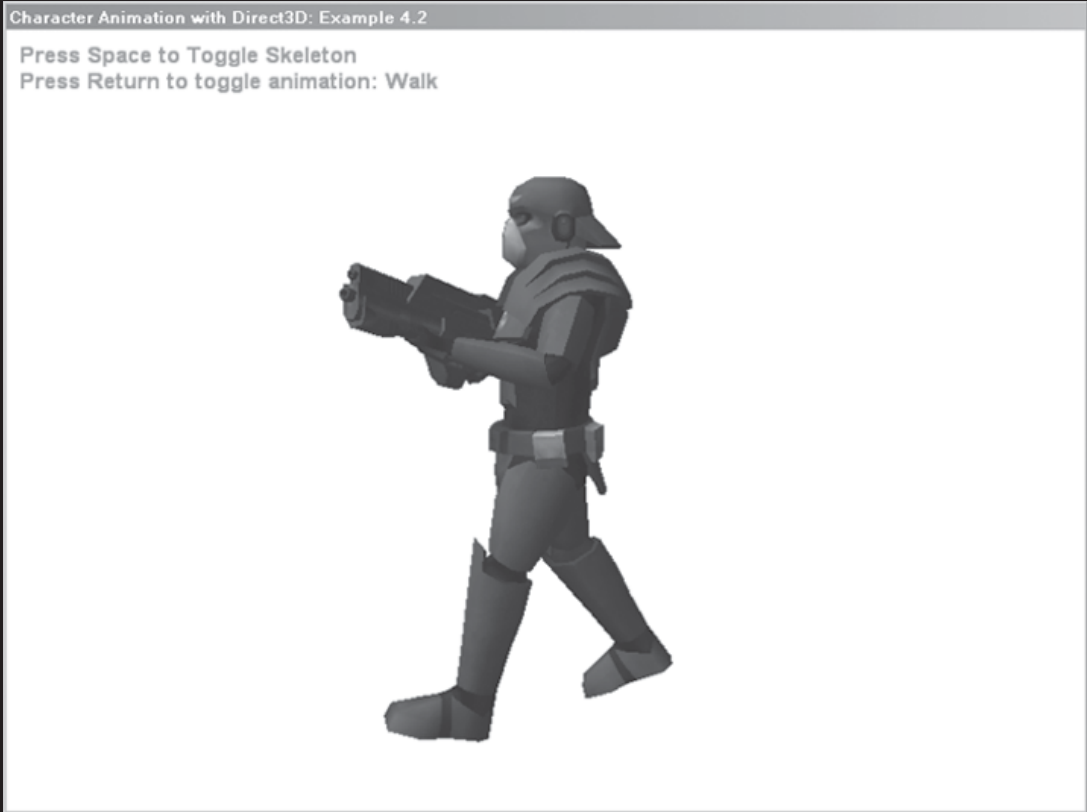


添加到SKINNEDMESH类的这个函数使用存储在角色的ID3DXAnimationController中的动画集的所有名称填充向量。首先，GetMaxNumAnimationSets（）函数用于查询动画集的数量，然后GetAnimationSet（）函数可用于获取实际的动画集。ID3DXAnimationController对象有多个轨道，每个轨道都是动画集的一个插槽。这意味着您可以同时拥有多个活动动画，甚至可以将它们混合在一起。动画控制器的轨道将在下一章中详细介绍。现在，假设您只有一个轨道带有一个活动动画。在这种情况下，您可以使用此函数为轨道设置活动动画：

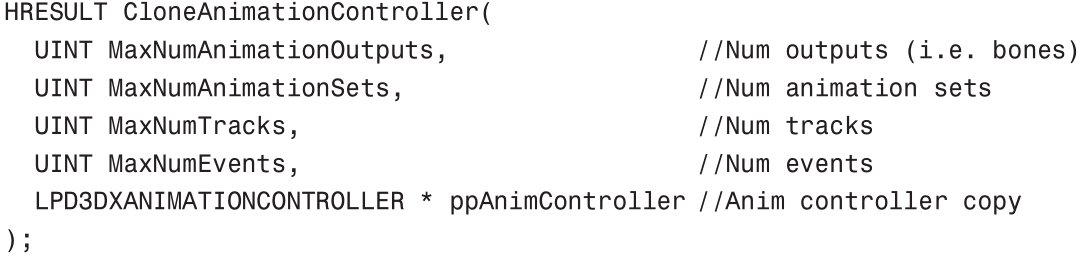


要设置要播放的动画，只需使用GetAnimationSet（）函数检索动画集，然后使用SetTrackAnimationSet（）激活它。一旦为一个（或多个）轨道设置动画集，就可以开始实际动画了。您可以使用以下功能更新/播放动画：





到现在为止还挺好。你有一个网格，一个动画控制器，总而言之，一个工作角色。如果你需要两个角色怎么办嗯...天真的方式是拥有两个网格和两个动画控制器。这没问题。但是，如果你需要一支军队呢？显然，为每个士兵实例设置一个网格将不是最明智的方法。问题的解决在于你可以使用以下函数克隆角色的动画控制器：



从动画控制器创建克隆后，它会保留独立的时间计数和活动的动画集等。这意味着，当您要创建角色的多个实例时，应仅复制动画控制器而不是整个骨骼结构。渲染相同蒙皮网格的多个实例时，请遵循以下大纲：1.为活动动画控制器调用AdvanceTime（）。

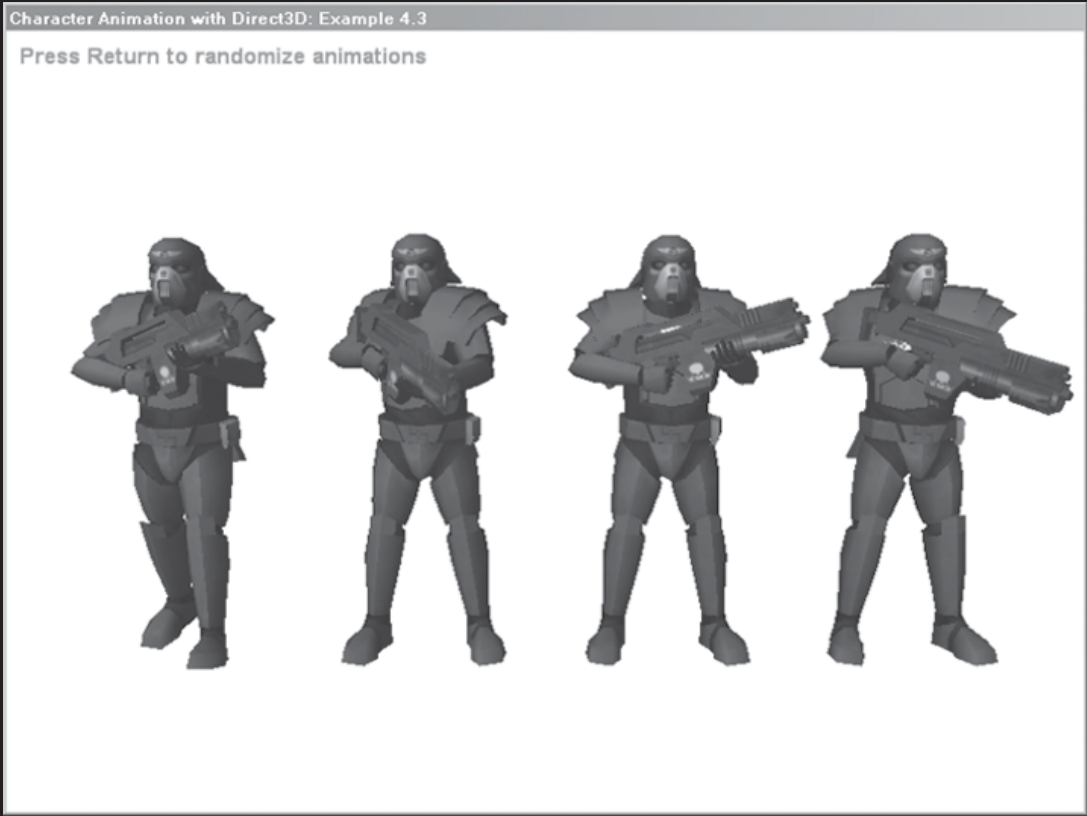
1.为活动的动画控制器调用AdvanceTime（）。

2.计算此角色实例的世界矩阵。

3.使用世界矩阵更新蒙皮网格的组合变换矩阵。

4.渲染蒙皮网格。

5.重复下一个角色实例。



本章从关键帧开始，编写了一个存储在ID3DXKeyframedAnimationSet（）接口中的动画集合，最后介绍了ID3DXAnimationController接口。现在，您应该对动画如何从头开始构建感到满意，即使在大多数情况下，您也可以将它们放在银幕（silver platter 我翻译成银幕）上。了解动画管道是如何工作的永远不会有坏处，特别是稍后涉及到动态动画等更高级的主题时。

在本章中，您第一次看到了ID3DXAnimationController接口，但这并不是您的最后一次。下一章将介绍使用这个接口可以做的一些更高级的事情。

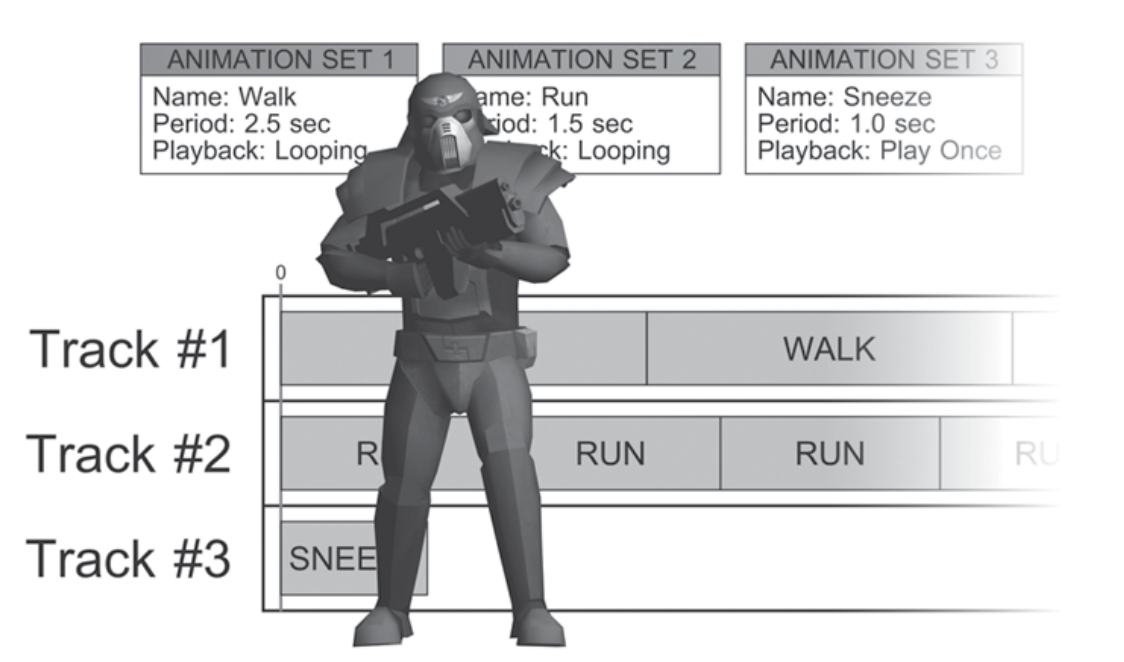
以下操作是有时间可以试一试的：

展开在Example 4.1中创建的Animation类。添加新关键帧，设置动画速度等。

将Animation类连接到网格。利用从动画集中获得的缩放，旋转和平移。

使用从Soldier检索到的ID3DXAnimationController。看看您是否可以在代码中创建新动画集并将其注册到控制器。(提示:RegisterAnimationSet()和RegisterAnimationOutput()函数应该很有用)。

# 高级的骨骼动画技术



在本章中，我将深入探讨一些更高级的骨骼动画技术。您将学习的第一件事是如何将多个动画混合在一起。例如，当您想要在不同动画/姿势之间平滑过渡时，这很有用。该技术还可用于创建全新的动画。这方面的一个例子可能是你有一个Run动画和一个Fire-Rifle动画。通过混合它们，您可以拥有Run and-Fire-Rifle动画，而无需在3D程序中手动设置动画。此外，在本章的最后，您将研究动作捕捉的主题。本章包括以下主题:

轨道结构

混合动画

压缩动画集

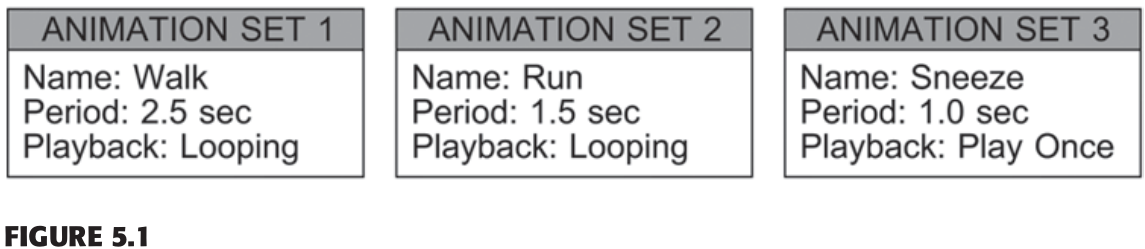
动画回调

动作捕捉

在逐渐淡入/淡出动画，将动画混合在一起等等之前，需要考虑一件事：动画控制器中的轨道。这在前一章中有过简要的介绍，但我没有深入讨论任何细节。您可能还记得在使用D3DXCreateAnimationController（）（前一章明明没有这个函数。。。。）函数创建新动画控制器时指定了轨道数。轨道还被用于使用动画控制器的SetTrackAnimationSet（）函数激活角色的某个动画。如上所述，动画控制器可以包含多个轨道。

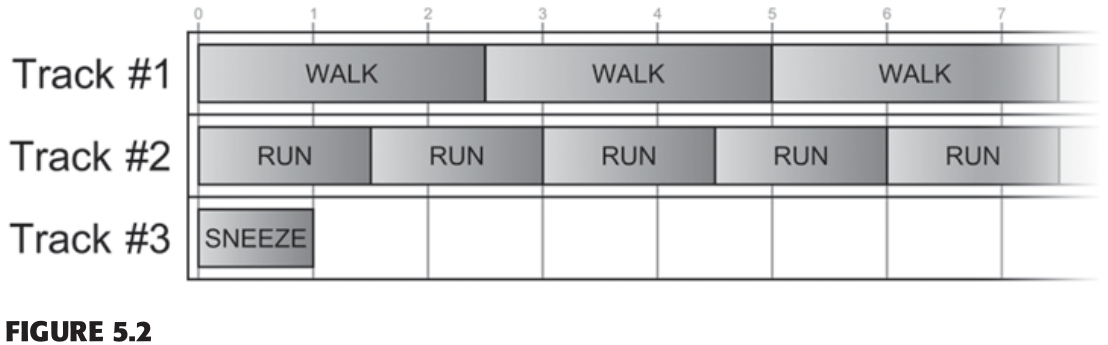
位置，重量和速度属性都很容易理解。轨道的优先级可以设置为D3DXPRIORITY\_LOW或D3DXPRIORITY\_HIGH。在添加低优先级轨道之前，首先将高优先级轨道混合在一起。当角色远离播放器/摄像机时，这也可用于关闭低优先级轨道。

为了更好地说明使用轨道混合动画的方式，请考虑以下示例。图5.1显示了三个动画集，每个动画集包含一个单独的动画，其中包含详细信息。



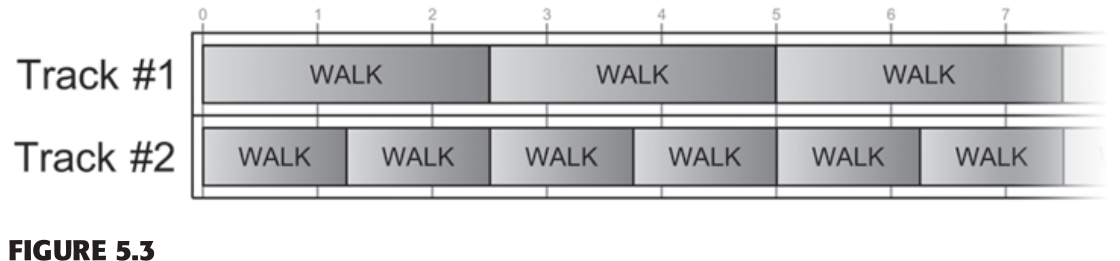
三个动画集示例。

图5.1显示了Walk，Run和Sneeze动画。Walk和Run动画都是循环动画，这意味着它们将永远持续，而Sneeze动画只发生一次然后停止。图5.2显示了如果将每个动画分配到单独的轨道上，会是什么样子。



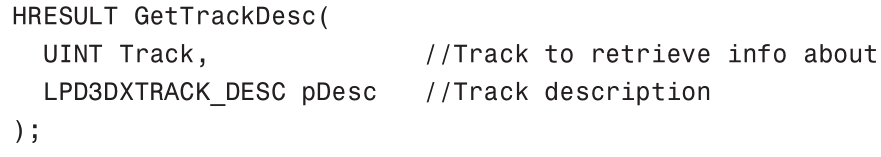
分配给单独动画轨道的三个动画集。

你并不局限于在每个轨道中设置不同的动画。有时，将相同的动画分配给多个轨道是有意义的。例如，查看图5.3; Walk动画被分配给轨道1和轨道2，不同之处在于轨道2中的轨道速度是200%(即动画将播放两倍的速度)。

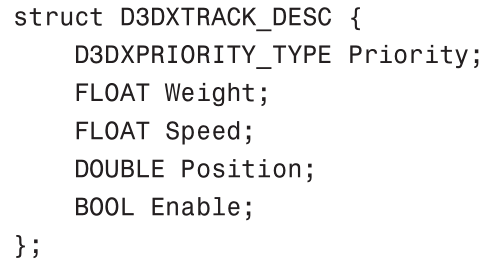


轨道的速度属性会影响动画播放。

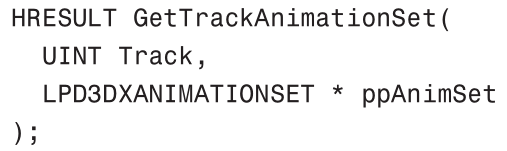
要检索轨道的当前状态，可以使用以下动画控制器功能：



该函数将填充以下结构：

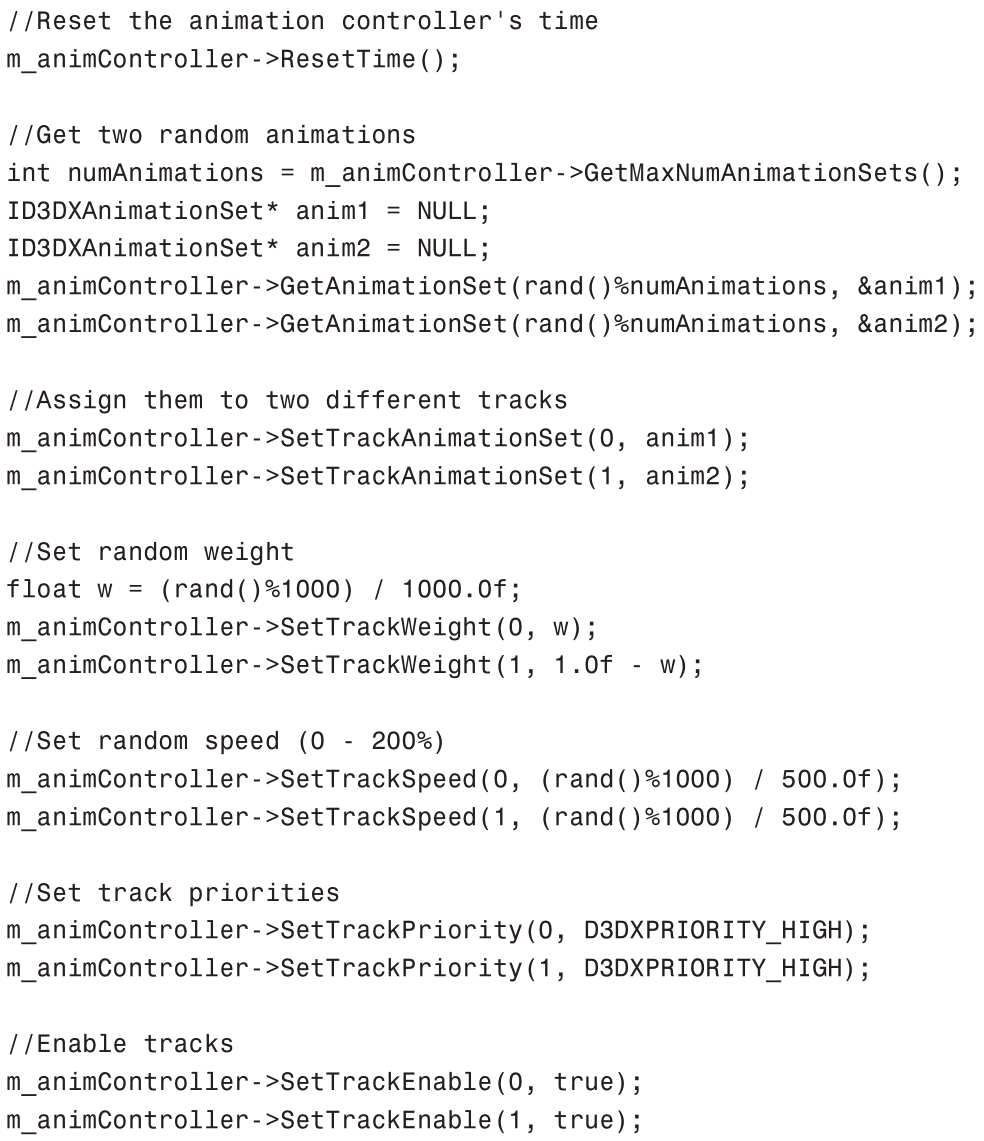


这个结构中唯一不包含的关于轨道的信息是分配给它的当前动画集。为此，您可以使用ID3DXAnimationController接口中定义的此函数：

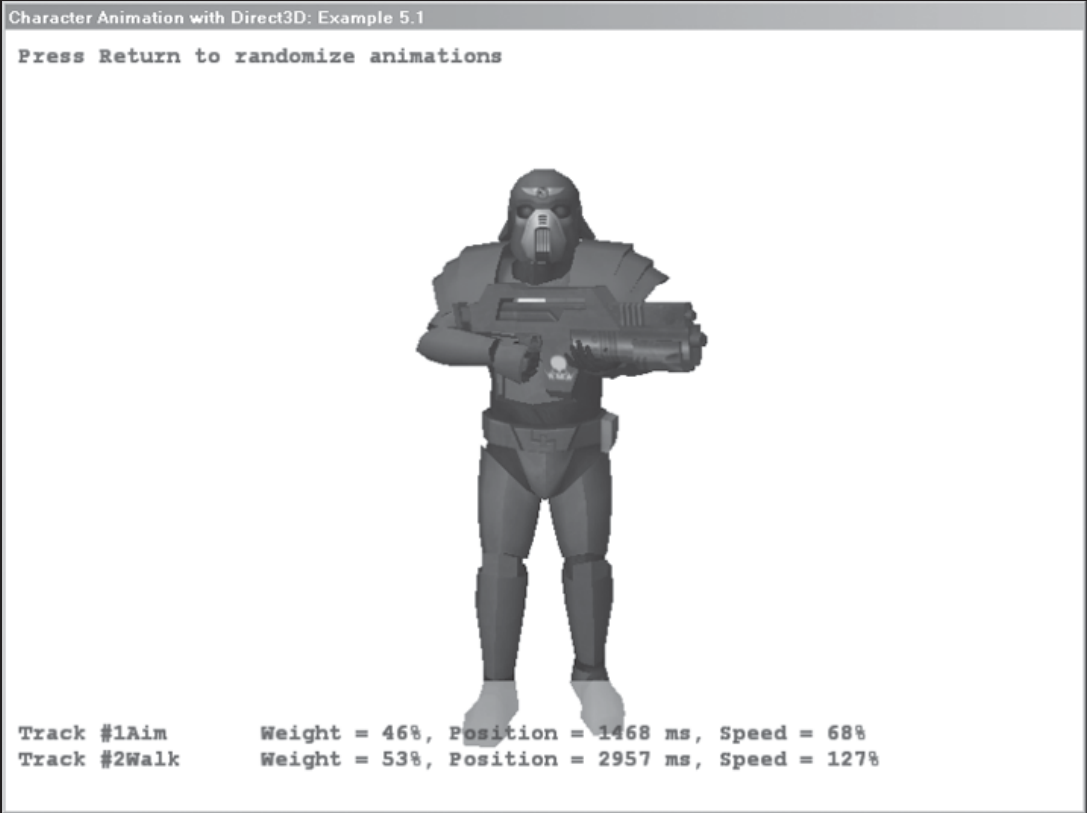


动画控制器的GetTrackAnimationSet（）函数返回指向当前分配给特定轨道的动画集的指针。好的，现在你知道如何查询动画控制器的所有必要的轨道属性。 现在是时候继续前进并尝试将两个轨道混合在一起。

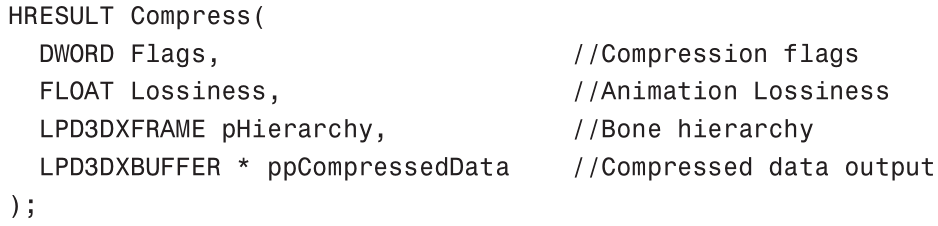
要将多个动画混合在一起，您需要检索要使用的不同动画集。然后将它们分配到不同的轨道并设置不同轨道的权重，优先级和速度。下面的代码随机将两个动画混合在一起：



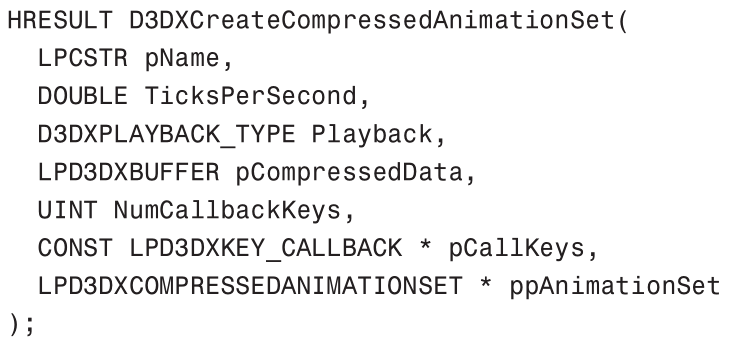
如果两个动画尝试为同一骨骼设置动画，则它们各自的权重将决定骨骼的动画效果。例如，如果一个动画轨道的权重为5而另一个轨道的权重为2.5，则受第一个轨道影响的任何骨骼将受到第二个轨道影响的两倍。



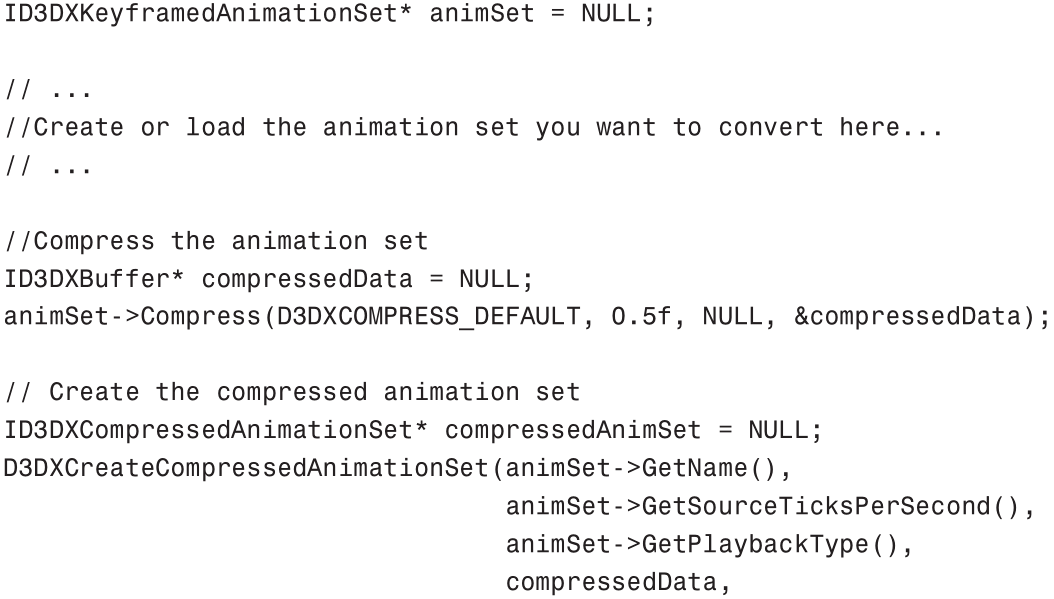
您已经了解了ID3DXKeyframedAnimationSet接口，并了解了如何为其添加关键帧。在拥有大量动画数据的大型游戏中，有时压缩动画数据以允许更多的动画数据是明智的。同样，D3DX库是一个很好的帮助。对于压缩动画集，您可以使用ID3DXCompressedAnimationSet接口。要将关键帧动画集转换为压缩动画集，需要调用要压缩的关键帧动画集的Compress（）函数。

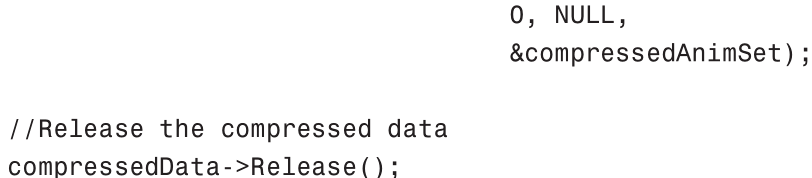


压缩标志可以是D3DXCOMPRESS\_DEFAULT，这是一种快速压缩方案，或D3DXCOMPRESS\_STRONG，这是一种更慢但更准确的压缩方法。（注意：尚不支持强压缩，但可能会在将来的DirectX版本中支持。）您还可以将所需的损耗（即，允许压缩方案改变数据的程度）设置为0到1之间的值。作为这个函数的输出，您不会得到ID3DXCompressedAnimationSet—相反，您会得到包含所有压缩动画及其关键帧等的数据块。获得此压缩数据后，可以使用以下D3DX库函数创建新的压缩动画集：



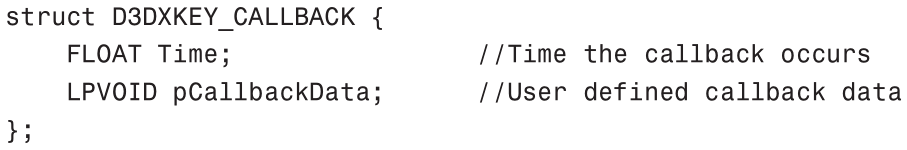
提供名称，每秒刻度，回放类型，压缩动画数据和可选回调函数（稍后将详细介绍），您将获得新的压缩动画集作为结果。这里有一些代码显示如何使用这些函数将关键帧动画集转换为压缩动画集。



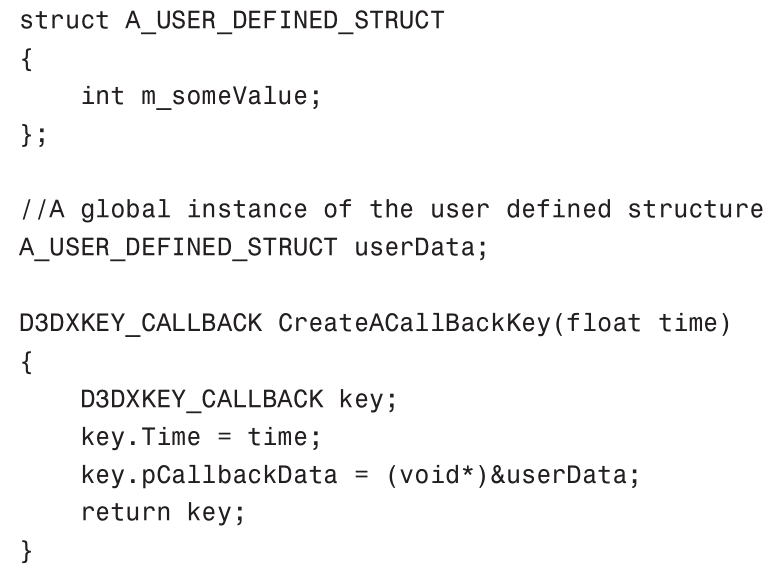


如您所见，每秒的名称，播放类型和刻度都取自原始动画集。您只需提供额外的压缩动画数据，即可获得压缩动画集。仅仅为了减小动画集的大小，这似乎很麻烦。但是，一旦动画的数量开始大幅增加，压缩动画集是一个很好的伎俩。

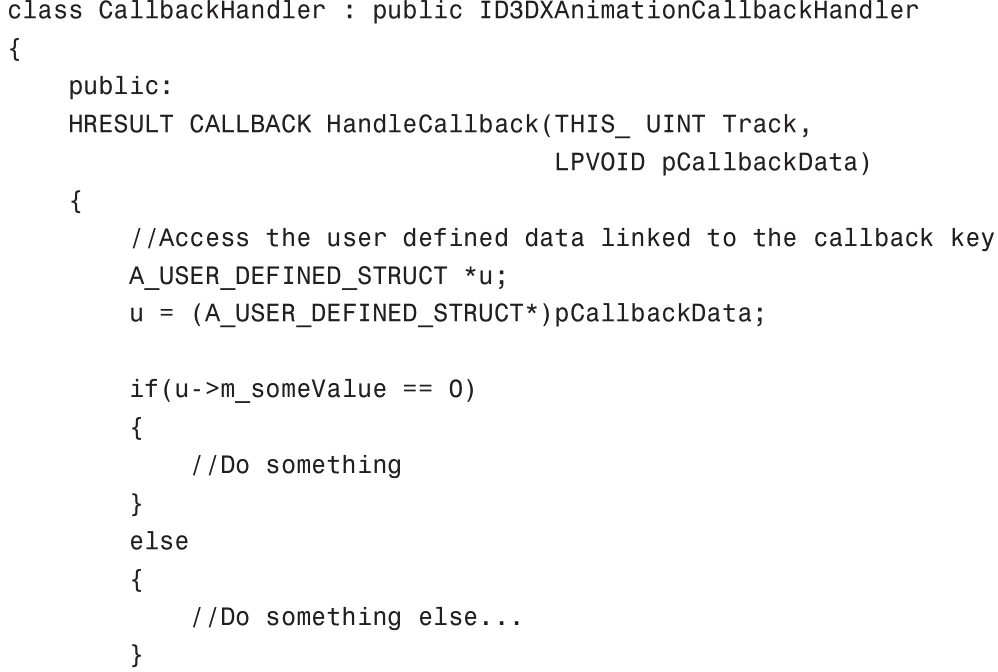
动画回调是与动画同步的事件。一个例子可能是播放脚步声。想象一下，你有一个类似于本章前面的步行动画。请记住，您可以以不同的速度播放此动画。如果你必须手动将脚步声的声音连接到动画上，你将不得不经历各种各样的烦恼来计算你需要播放声音的时间。这就是动画回调的作用所在。创建一个回调函数并将其添加到动画集中。每次动画传递这个回调函数时，都会生成一个播放声音的事件。您还可以自定义此事件 - 例如，如果角色踩踏砾石而不是木质表面，则播放不同的声音。回调函数使用以下结构定义：



D3DXKEY\_CALLBACK结构包含一个包含时间戳的浮点值，以及一个指向任何用户定义结构的指针。如前一章所述，这些动画键结构的时间戳以刻度而非秒为单位。因此，请记住将事件发生的实际时间（以秒为单位）乘以动画的每秒刻度值。

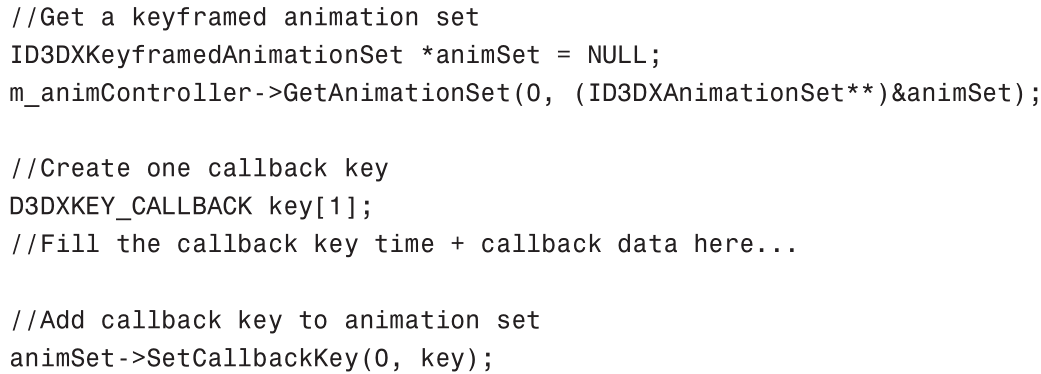


此代码创建用户定义的结构，并定义一个函数，该函数创建链接到此用户定义结构的新回调键。添加了大量回调事件之后，需要创建自己的回调处理程序来处理传入的事件。为此，您需要实现自己版本的ID3DXAnimationCallbackHandler接口。

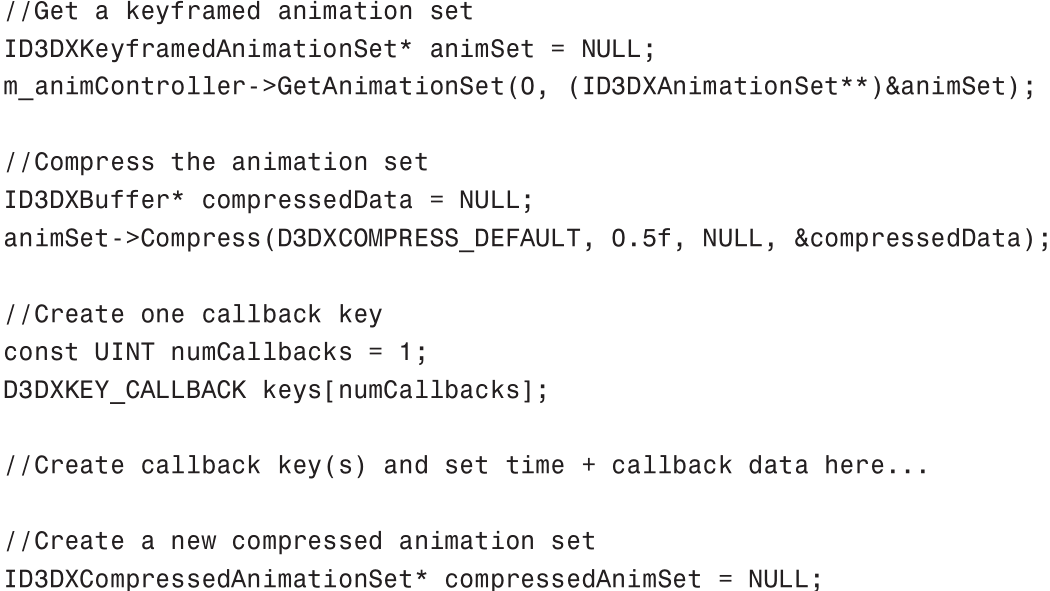


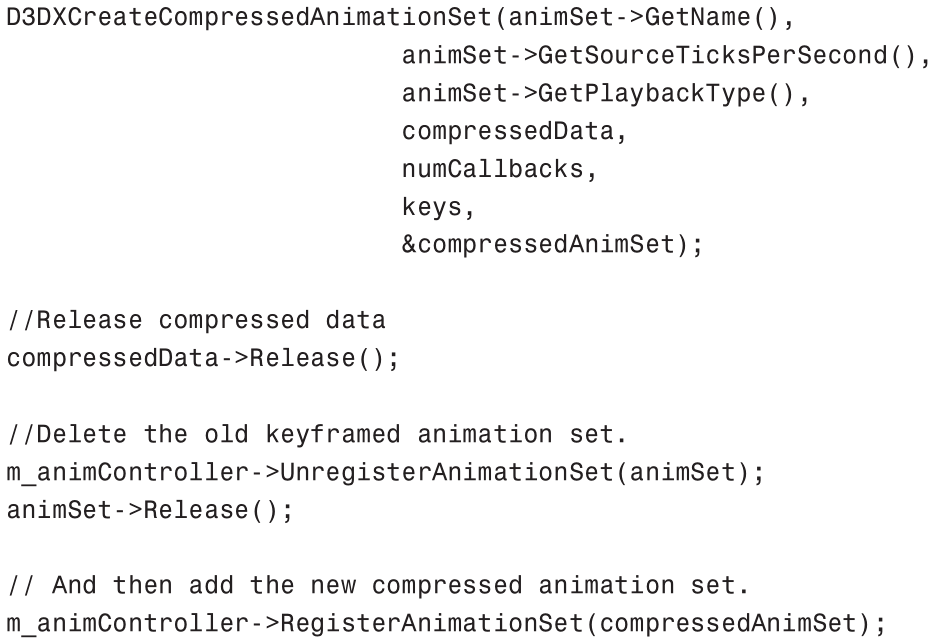


在这里，您可以看到如何实现ID3DXAnimationCallbackHandler接口来处理您自己的用户定义数据结构。所有事件处理都在HandleCallback（）函数中完成，该函数是ID3DXAnimationCallbackHandler接口中定义的唯一函数。好的，现在您知道如何创建回调键以及如何在触发事件后处理它们，但尚未涉及的是如何向现有动画添加新的回调键。



SetCallbackKey（）函数将回调键添加到普通的关键帧动画集。您还可以将回调键添加到压缩动画集，如下所示：

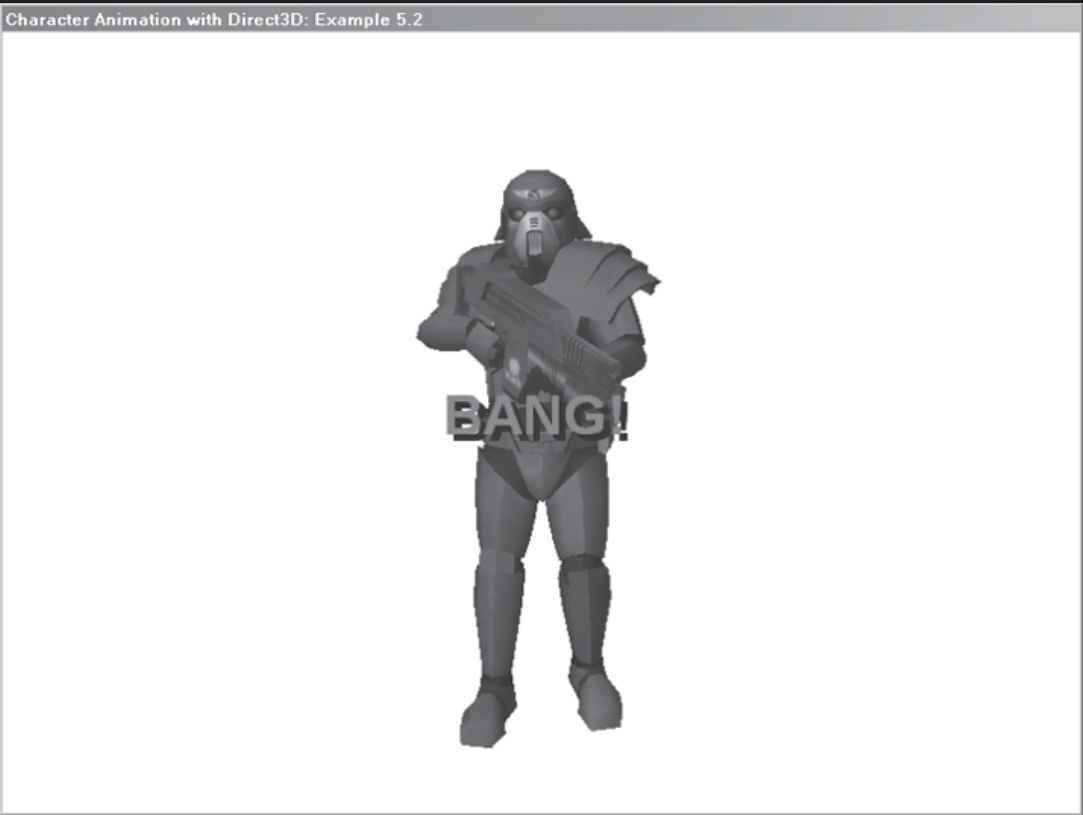




像以前一样，当您压缩动画集时，您将执行相同的步骤。只有这次你还使用一组回调键提供D3DXCreateCompressedAnimationSet（）函数。创建新的压缩动画集之后，在动画控制器中注销旧的动画集，并在其位置注册新的压缩动画集。它之前的最后一件事就是将回调处理程序发送到动画控制器的AdvanceTime（）函数。



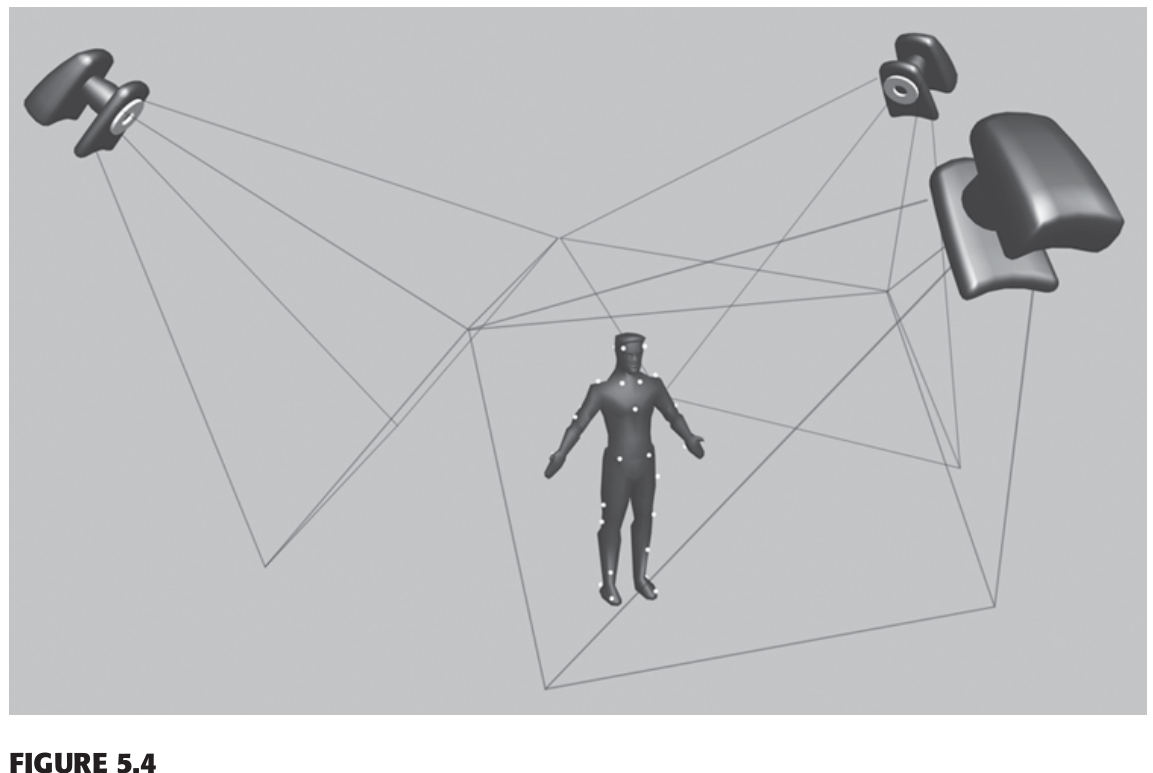
这实际上意味着您还可以使用不同的回调处理程序来处理相同的回调事件。因此，例如，如果你的角色受伤，你可以拥有一个与角色健康时不同的回调处理程序。换句话说，每次触发某个回调事件时，都可以执行不同的代码，具体取决于您使用的回调处理程序。



本节简要介绍了动作捕捉的高级主题，也称为Mocap。动作捕捉是记录现实演员的动作并将这些动作/动画应用于3D角色的过程。动作捕捉的使用在电影和游戏行业中最为常见。使用Mocap设备，您可以使用更传统的3D动画工具制作更逼真的动画。您可以创建新角色动画的速度也比传统方式快得多。

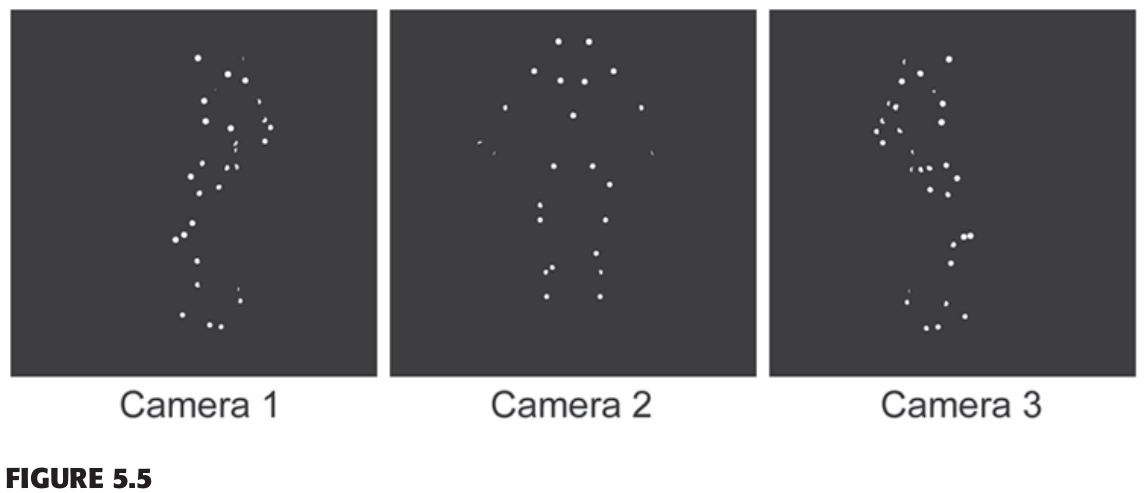
有几种不同类型的Mocap系统。一般来说，它们可以分为三类：光学，磁性和机械。虽然这些系统有许多不同之处，但它们也有一些共同点。它们都非常昂贵，需要大量的技术专业知识，还需要大量的校准。因此，游戏公司（和其他公司）将其动作捕捉需求外包给专门从事Mocap的工作室是很常见的。在本章的最后是对拉普兰工作室的一些人的采访，他们为其他公司做了很多Mocap。

简而言之，光学Mocap系统可与安装在面向中心的房间墙壁上的多个摄像头配合使用。这些相机通常是非常昂贵的高对比度相机。然后演员穿着一套衣服，衣服上有许多白色的小球(标记)。这些标记由摄像机捕获，摄像机使用三角剖分法计算标记在三维空间中的位置。图5.4显示了如何设置这样的系统。标记通常有两种形式，具体取决于系统：活动（包含小的红外光）和非活动（反射标记）。



光学Mocap系统。

图5.4仅显示了三个摄像头（这是一个像这样的系统的理论最小值）; 但是，您拥有的摄像机越多，系统就越准确和强大。图5.5显示了图5.4中三个摄像头的图像。

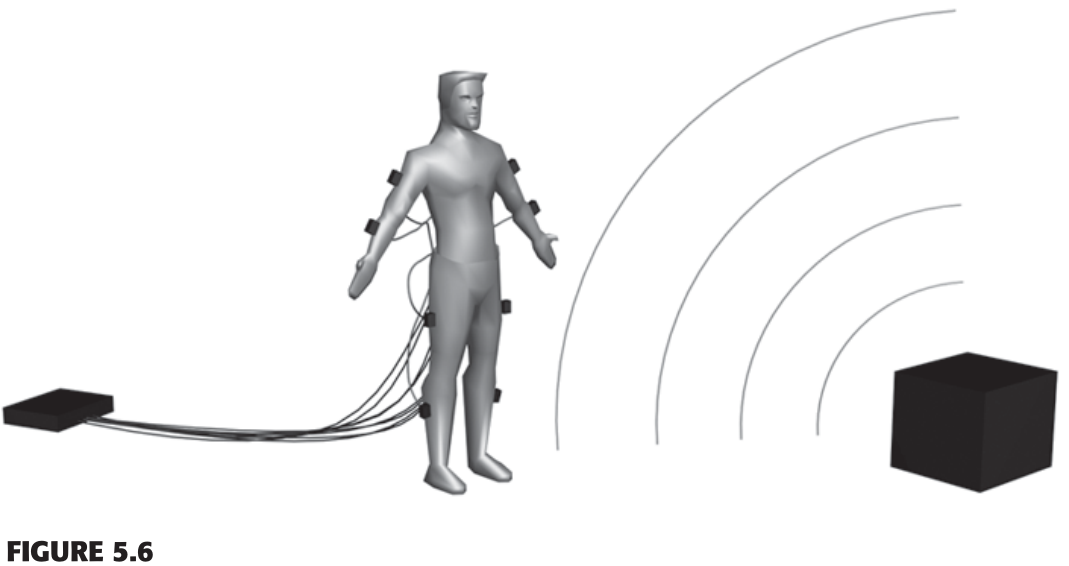


从光学Mocap系统记录的图像

即使从这些简单的图像中，您也可以轻松看到佩戴标记的人的轮廓。当您观看这些标记移动的实时馈送时，它会变得更加明显。一旦使用来自所有相机的图像来计算许多标记的3D位置，就将这些图像映射到虚拟骨架上。然后导出该虚拟骨架的运动，并且可以在3D建模软件中使用，最后在游戏或电影中使用。

最近，在无标记运动捕捉领域进行了大量研究。在撰写本文时，这项技术刚刚开始进入市场[1]。基本上，无标记Mocap可以像任何其他光学系统一样工作，但没有标记。使用多个摄像头和先进的计算机视觉算法提取动作，聚焦于你身体的某些部位，轮廓检测等。无标记Mocap尤其适用于面部动画等[2]。

磁性运动捕捉系统几乎像光学系统一样工作。不同的是替代衣服上的许多 标记，磁性运动捕捉系统用的是有线传感器附着在人的四肢上。这些传感器通过屏蔽电缆连接到控制单元，控制单元测量它们在低频磁场中的位置和方向。电磁场由静磁发射器产生。磁性Mocap的优点在于它还可以为您提供传感器的方向（在光学系统种必须离线计算）。这使磁性Mocap系统适用于实时动作捕捉（用于不同的直播电视节目等此类行业种）。磁性Mocap系统的一大缺点是它们是有线的（并且传感器也可以重量很大）。这意味着它们很麻烦，并且在录制时限制了演员的动作。磁性Mocap系统的另一个重大缺点是它们对噪声和其他磁场非常敏感。任何金属表面都会干扰磁场并导致传感器读数错误。一个磁力运动捕捉系统如图5.6所示。



磁性Mocap系统

机械运动捕捉系统通常建立在演员佩戴的外骨骼上。记录外骨骼的不同关节取向并用于产生Mocap数据。这项技术的主要缺点是没有记录位置数据，因此跳跃，逼真的运行动画等内容无法直接记录，之后需要进行一些手动修改。这项技术的另一个缺点是外骨骼往往体积庞大，可能会限制演员。然而，机械Mocap系统并非一切都不好。它是机械的这一事实意味着它不会受到干扰，遮挡和类似问题的困扰。还有机械Mocap系统的例子，它们在背包中具有记录计算机和电源，有效地使得套装完全独立于位置。您可以在图5.7中看到一个带有外骨骼的机械Mocap系统示例。



机械Mocap系统

Moven[3]最近实现了一种使用紧身衣的运动捕捉。他们已经制造了一套带有微型惯性传感器的超薄纤薄套装，这根本不麻烦。由于该系统不依赖于摄像头等，它的最大优点是可以在任何地方使用。

毋庸置疑，所有这些技术都各有利弊。每种方法也有不同的变化，都有各自的优点和缺点。

尽管光学系统存在缺陷，但就Mocap的游戏和电影而言，它们的优点超过了它们的优点。随着时间的推移，无标记的Mocap可以取代常规的光学系统。但目前而言，高采样率和高精度似乎使光学技术成为游戏角色动作捕捉的最佳方法。

我有机会参观了位于芬兰罗瓦涅米的拉普兰工作室的动作捕捉工作室。他们有一个VICON [4]光学动作捕捉系统，使用14个摄像头，捕捉面积为4 x 4 x 3米。

在本章中，您了解了一些更高级的角色动画主题。你学习了动画混合，如果你想要创造一个真实的角色，你肯定需要它。有了它，您可以通过将其他两个混合在一起来创建新的动画。您还可以通过在两个动画之间进行混合来创建两个动画之间的过渡。本章介绍的另一个高级主题是动画回调，您可以使用它来计算游戏中特定动画发生时的事件。例如，如果您制作了一款格斗游戏，您可能想要在punch动画中将“punch”声音计时到特定的时间。最后，简要介绍了动作捕捉的主题及其变化。以上就是本书骨骼动画部分的全部内容，也是本书最后三章的重点。现在是时候转向动态动画，看看如何实现一个布娃娃系统了。

练习

尝试修改示例5.1，使您可以在两个不同的轨道上运行相同的动画，使用不同的权重和稍微不同的速度。看看会发生什么?

为士兵的步行动画中的脚步声创建动画回调。