J2ME M3G 中 Animation 的设计与实现*)

徐中礼　谢丹铭　李冰峰　高传善
(复旦大学计算机科学与工程系　上海 200433)

摘　要　为了 Java 三维图形程序能在不同的移动设备和平台上运行，JCP (Java Community Process)制定了 J2ME M3G(Mobile 3D Graphics API)，规范了移动设备上 Java 三维图形程序的 API 和框架。而 M3G 中由于动画部分的独特性以及它在设计和实现中的复杂性，我们结合在 XORP 上开发 M3G 类库经验的基础，总结并给出了其设计和实现的参考模型和理论基础，也适用于其它 3D 类库的设计和开发。

关键词　Java，JCP，J2ME，M3G，动画，XORP

Designing and Implementation of Animation in J2ME M3G

XU Zhong-Li　XIE Dan-Ming　LI Bing-Feng　GAO Chuan-Shan
(Computer Science and Engineering Department, Fudan Univ, Shanghai 200433)

Abstract　The JCP (Java Community Process) constitutes the J2ME M3G (Mobile 3D Graphics API) and specifies the Java 3D graphics API and framework on the mobile facilities in order to make applications programmed in M3G run on different mobile equipments and platforms. However, because it is unique and complicated in the designing and implementation of Animation in M3G, we summed up the reference model and theory according to the M3G library that we have developed on XORP. It also applies to other 3D libraries.

Keywords　Java, JCP, J2ME, M3G, Animation, XORP

1　引言

随着手机、PDA 等移动设备的普及和性能的不断提高，运行在它们之上的应用程序也日趋多样化。Java 作为与平台无关、面向对象的语言而受到青睐。开放式运行平台 ORP 是一个高性能的可控制运行环境软件，是用来研究垃圾收集和动态编译技术的开放资源研究平台，它支持执行类型安全字节码，能够运行 Java 程序。经过改进，已可以运行 J2ME 程序。ORL 就是我们在改进的 ORP 之上，遵循了 Clean Room 的原则，严格按照 Sun 公司的 J2ME 规范，开发出的 J2ME 类库。目前我们在 ORL 的基础上开发 M3G 类库。

2　M3G 与 Animation 概述

为使 J2ME 支持三维图形的应用程序开发，多家公司和组织联合制定了 Mobile 3D Graphics API (M3G)，规范了三维图形类库的 API 和框架。

M3G 共支持 30 个类，其中以 Object3D 为绝大数类的基类，在图 1 中给出了用于描述物体到主要类之间的继承关系。对于继承 Transformable 的类都有自己的变换矩阵，只有 Node 及其子类可被显示出来。Camera 用来设定观察视角和位置；Group 用来构建 Node 场景，可以包含一组 Node 对象，并且可以嵌套；Light 用来设定光照效果；Mesh 用来构造物体，包括物体的顶点信息，颜色，和纹理(Texture2D)信息等；Sprite3D 用来把 2 维的图片显示在 3 维空间中。如果要构建一个完整的场景，所有的信息都保存在类 World 中，然后再依次渲染。SkinnedMesh 和 MorphingMesh 用来刻画运动的物体，前者通过制定骨骼框架而后者通过起始和终止位置的线性插值来决定某一时刻的物体位置其它的类还包括背景，雾，材料，坐标变换等辅助类。

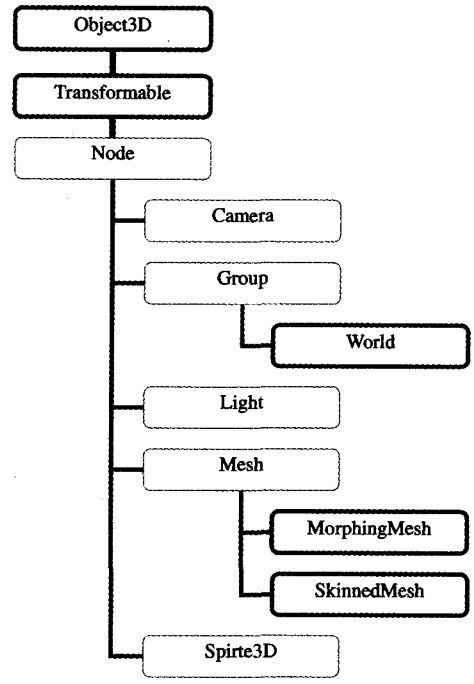

图 1　类的继承关系

M3G 是通过构建树来描述要渲染的场景，只有 Group 对象才能有自己的子树(孩子节点)，即只有 Group 对象才能是树的内部节点，而页节点可以是除了 World 的任何 Node 对象。

M3G 中有 2 种渲染模式：立即(immediate)模式和延迟(retained)模式。前者是直接渲染一个 Node 对象，而后者是渲染一个以 World 为根节点的场景树。

*)Intel 公司基金资助项目"J2ME Claa Libraries with Small Footprint, Low Power and High Performance on XScale Processor"。**徐中礼，谢丹铭，李冰峰**　硕士研究生，研究方向：计算机网络和信息工程；**高传善**　教授，博士生导师，研究方向：数据通信、计算机网络和分布式系统及其应用。

动画在生活中随处可见，典型的例子就是电影。动画令静止的场景令不显得单调，而在很多游戏中更需要动画来使画面更加逼真。在 M3G 中动画是直接作用在各种 Object3D 对象的实例上，包括 Node，Group，World，Mesh，SkinnedMesh，MorphingMesh，Sprite3D，Light 等。任何属性都可以被动画，包括坐标，光照，颜色等，M3G 把这些属性成为动画属性。动画主要是通过 AnimationTracker、KeyframeSequence 和 AnimationController 这三个类来实现的。

3 Animation 的设计与实现

3.1 AnimationTracker

用于关联一个拥有 AnimationController 的 KeyframeSequence 和一个动画属性。

(1)三者的关系

三者的关系可以由图 2 来表示。

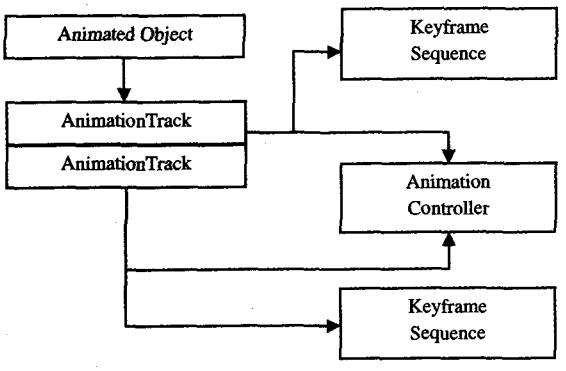

图 2　三者的关系

一个动画属性可以是向量或是标量的，这个由特定的动画系统直接更新，譬如，一个节点(Node)的方向。AnimationTracker 中定义了 21 种动画属性。动画属性可以由一系列的符号变量来标识，一些动画属性是仅对一个类而言的，譬如 Material 的 SHININESS 属性。绝大多数继承了 Object3D 的类都拥有一个或者多个动画属性。一个拥有动画属性的 Ojbect3D 实例被称作动画对象(animated object)。一个动画对象中的每个动画属性组成了一个唯一的动画目标(animation target)。每个动画对象可能使用 0 个或多个 AnimationTracks。它们中的每个对象在被它们的各自的 AnimationControllers 激活时，都掌管着动画对象的一个动画目标的更新操作。赋予目标的值将通过对由 AnimationTrack 对象引用的 KeyframeSequence 对象的取样来决定。每个 KeyframeSequence 可以被多个 AnimationTracks 引用，关键帧的数据是可以共享的。每个 AnimationTrack 只和一个 AnimationController，一个 KeyframeSequence，以及一个动画属性相关联，但是它可以和多个动画目标相关。换句话说，它可以使多个不同对象中的相同属性同时运动起来。还有另外一种情况，就是许多 AnimationTrack 对象和一个单独的动画目标相关，在这种情况下，最终的动画目标是一系列来自于每个单独的 AnimationTracks 的值的线性组合，权重就是各自的 AnimationController 的权重。

(2)插值处理

动画关键帧作为浮点数输入，经过插值和混合后的值也是如此；如果一些属性值将被应用到它们所对应的目标属性时，这些值必须被映射成在相应属性中可以表达的值，例如，浮点属性的值必须限制相应的精度，整型的属性性值必须由浮点型近似到相应的整型，逻辑属性的值必须这样判断，当属性值大于或等于 0.5 时表示真，否则为假。其他的属性根据自身得特点进行处理。

3.2 KeyframeSequence

该类封装了一个拥有时间戳和向量表示的关键帧序列，每个关键帧表示在一个特定时刻的一个动画量的值。

一个关键帧可以和多个动画目标相关联。有效的动画目标包括节点和纹理的变换，材质(Material)参数，镜头(Camera)参数等等。当应用一个动画于它的目标时，目标的真实值往往是通过在相关的 KeyframeSequence 对象中的关键帧的插值来得到的。

每个关键帧中的向量组件数量是在构造器中标定的，在序列中的所有关键帧保持一致。关键帧的插值由这个序列所附属的动画对象所决定。譬如，具有 4 个组件的关键帧在应用到 ORIENTATION 目标时将会被描述为一个四元组。

有五种对关键帧的插值方法：LINEAR 和 SPLINE，它们的等价四元组 SLERP 和 SQUAD；以及简单 STEP 函数；还有两种影响插值的重复模式 LOOP 和 CONSTANT。

(1)序列时间和世界时间

一个 KeyframeSequence 内部序列时间 t 是由与之相关联的 AnimationController(s)中定义的世界时间映射过来的。这个序列时间是用来在多个关键帧值之间使用已经选定的插值算法进行插值运算。

在一个 KeyframeSequence 中所有的有效关键帧必须落于[0，D]这样一个序列时间的区间范围内，其中 D 是序列时间的持续时间。这个持续时间可以使用 setDuration 方法进行设定。对于使用 LOOP 循环模式的序列，序列时间 t 是使用一个取模操作来限制到这个区间的，也就是说通过加或者减多个 D 来满足 $0 \leqslant t < D$，对于使用 CONSTANT 模式的序列，t 的值是不受限定的。

(2)序列重复模式

在一个 CONSTANT 序列中的第一个有效关键帧定义了插值在该时间点之前的返回值，也就是说，在 t_0 时刻以初始值 v_0，则对于时间 t，满足 $t < t_0$，那么插值 $v = v_0$。

在一个 CONSTANT 序列中的最后一个有效关键帧定义了插值在该时间点之后的返回值，也就是说，在 t_{N-1} 时刻以初始值 v_{N-1}，则对于时间 t，如果满足 $t \geqslant t_{N-1}$，那么插值 $v = v_{N-1}$。

对于一个使用 LOOP 循环模式的序列，当关键帧在一个与给定的序列持续时间相等的间隔内被向前或向后任意次数的复制时，将被插值；在这种情况下，一个关键帧在 $t + nD$ 时刻将会被看作如同在 t 时刻一样，其中 n 是任意的正或者负整数，D 是一个单循环动画的持续时间。

在一个有 N 个关键帧[0，$N-1$]的循环序列中，帧 $N-1$ 的下一帧是帧 0，而帧 0 的前一帧是帧 $N-1$。

(3)共存帧

允许在时间的相同位置上几个关键帧共存。这就允许了动画序列的不连续性，这样对譬如相机动画的合并剪裁是很有用的，在多个帧共存的情况下，具有最小索引值的帧总是被用来作为在相应时间位置上的段结束值；对于从该点开始的段，具有最大索引值的帧总是被用来作为初始值。在 LOOP 模式下的帧序列，前一个或下一个重复序列的关键帧可能与当前序列中的某些帧共存，当它们恰好是前一帧的结束帧，或是后一帧的开始帧时。前一循环的关键帧的索引被视作拥有比当前循环中的关键帧的索引都小，而相应的后一循环的关键帧索引则认作当前循环中的关键帧的索引都要大。

3.3 AnimationController

该类用于控制一个动画序列的位置,速度和权重。一个动画序列需要控制多个对象的多种属性,例如一个链接的形体展现一个单一动作通常被认为是一个单一的动画,然而其中包含中对多个不同对象的位置和方向的复杂的坐标控制。我们定义一个动画序列为一系列独立的 AnimationTracks,它们有一个单独的 AnimationController 控制,每个 AnimationTrack 对象囊括着针对单一对象的单一动画属性控制的所有数据。一个 AnimationController 对象使它所相关的动画序列可以作为一个整体来执行暂停、停止、重新启动、快进、后退、任意定位,或分解动作。正式一些讲,它定义了一个由 world time 到 sequence time 的线性映射。

(1)动画应用

在立即模式和延迟模式下,通过调用在 Object3D 目标本身的 animate(int time)方法来实现对目标对象明确的动画应用,这样做将会重新评估带有一个或多个动画的对象的属性值。动画处理还会应用在目标对象的孩子节点上,故此应用程序可以自由地选择调用 World 的 animate 方法来使每个在 World 中的对象发生动画,亦或是通过应用已经存在的 Group 对象上使之发生动画。

当整个场景或是一棵对象的子树执行更新操作时,由控制程序所维护的世界时间将被传递给每一个动画对象,接着由动画对象将世界时间创送给每个与之绑定的 AnimationTrack 对象。

接下来 AnimationTrack 对象检查当前的世界时间是否落在与之绑定的 AnimationController 对象所确定的活动时间间隔内,如果不是则不会进行下一步动作,如果没有活动的 AnimationTrack 对象绑定在一个动画目标上,则目标属性将不发生变化。需要注意的是由于运动对象是独立的,因此可能在同一个对象中的其他的动画目标还是会因为某些原因发生变动的。

动画控制器拥有一个活动间隔值,这个值由世界时间(World Time)的最小值和最大值来确定的,它将用来标识动画控制器的活动时间间隔;相对而言,由非活动的动画控制器来控制的动画将对目标对象不发生作用,而只是在动画应用过程中被简单的忽略掉了。

(2)动画权重

每个动画控制器有一个相应的权重,所有活动的动画控制器在同一时间作用于同一个属性时,是用它们所占的权重来加以混合的。P_i 作为一个单独的动画控制器的贡献,w_i 代表权重,则总的标量属性 P 的表达式如下:$P=\sum(w_iP_i)$。

对于向量属性来说,上面的公式将会应用到每个独立的向量单元上;对于绝大多数的动画,上述简单的加权和已经足够表示,然而,对于方位值,实现要求对结果四元组进行归一化。

如果 AnimationController 是活动的,相应动画的序列时间就可以决定,序列时间接下来可以用于获得一个来自于 KeyframeSequence 对象的内插值,这个采样将会乘以 AnimationController 对象的权重因子并应用于目标属性;如果多个 AnimationTrack 对象指向一个相同的属性,它们将会按照它们各自 AnimationController 对象的权重加以混合。

(3)时间和速度控制

AnimationController 标定了一个由世界时间(world time)到序列时间(sequence time)的线性映射关系,世界时间事先已经传入 Object3D 对象的 animate 方法中,而序列时间用于采样关键帧数据而标定的。

在每次调用 animate 方法时,序列时间由给定的世界时间直接计算而得来,而并非作为内部存储形式存在。这样一来可以避免不期望得舍入误差积累,同时也可以简化动画系统的使用策略,仅仅依靠使动画系统处于有效的无状态情况下(相对于传统的状态机设计)。

上述映射关系由三个 AnimationController 中标定的参数变量,以及一个世界时间变量 tw 来表示计算序列时间 ts 的公式如下:

$$ts=tsref+s(tw-twref)$$

其中,ts 是序列时间,tw 是世界时间,$tsref$ 是参考序列时间,$twref$ 是参考世界时间,s 是速度,表示序列时间与世界时间之比。

它们之间的关系可由图 3 来描述。参考点($twref$,$tsref$)由 setPosition 方法来表示,速度由 setSpeed 方法来设置。(注意:设置速度将会影响到参考点,使之相应改变)

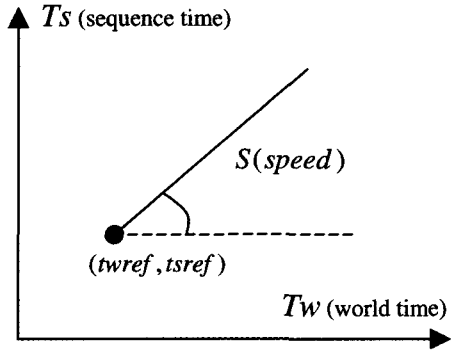

图 3 序列时间与世界时间

例:设想在一个世界(World)中,当前的时间为从开始算起的 5000ms,一个动画从 300ms 时开始(开始的序列时间记作 0ms),并以半速运动。也就是说动画开始与世界时间的 2000ms 以前,但是速度是 0.5,所以在序列时间范畴应该是 1000ms 以前,因此,当前的序列时间为 1000ms − 0ms = 1000ms,对应上面的公式中,tw=5000ms,$twref$=3000ms,$tsref$=0ms,and s=0.5。

需要注意的是时间的单位并没有标定,建议应用程序以及内容建立工具使用毫秒作为默认的时间单位;事实上,出入范围和精度的考虑任意的时间标称都是可以在应用中具体指定的。

(4)同步动画

与其他不同媒体类型进行动画的同步是可以基于由控制程序的世界时间来工作的,并没有什么同步事件或是其他的机制来满足这种需求服务,在试图将动画与音乐进行同步是,当前所流逝的时间可以直接来自与音乐播放器的库。

总结 本文提出了如何设计和实现 J2ME M3G 类库中的相关的动画部分。在设计和实现的过程中必须要遵守正确性,可扩展性,统一性和简洁性。本文中列举的设计和实现方法都是从实际开发中总结出来的,有很好的借鉴作用。

以上的一些设计和实现方法不仅仅适用于 J2ME M3G 类库的开发,它们更具有普遍的意义,也适用于其它类库的设计和开发。

参 考 文 献

1 http://www.jcp.org/en/jsr/detail?id=184, JSR184 Specification
2 Bloch J. Effective Java Programming Language Guide. Sun Microsystems, Inc. 2001
3 Gosling, James, Joy B, Steele G. The Java Language Specification, second edition, Addison-Wesley, Boston, 2000