可控式数字音乐系统模拟的研究与实现

罗誉家,李 樾,马春燕
(西北工业大学 软件与微电子学院,陕西 西安 710068)

摘 要:目前,数字音乐系统很大程度上仍依赖专业硬件设备,高效率的软件模拟方式仍有待探索。文中从数字音乐的文件结构、运作流程、封装方式、编程接口、逻辑结构模拟、用户接口几个方面,研究了以面向对象方式描述一个程序可控的数字音乐模拟系统的研发框架,并为数字音乐系统模拟流程提供了完整性定义和具体的实现技术。通过该文提出的解决方案可实现在脱离专业硬件条件的支持下实现对MIDI文件结构的封装,并以较优的性能搭建一个数字音乐系统。

关键词:数字音乐;音乐设备数字接口系统;模拟;程序可控
中图分类号:TP37 **文献标识码**:A **文章编号**:1673-629X(2010)09-0201-05

Research and Realization of Digital Music Simulation System under Program Control

LUO Yu-jia, LI Yue, MA Chun-yan
(College of Software and Microelectronics, Northwestern Polytechnical University, Xi'an 710068, China)

Abstract: With the popularity of Internet applications, digital music has entered our daily life deep. Research in the field of digital music draws more attention of interested people; however, there still has no set of complete description about the digital music system simulation. As a result, many people interested in digital music and programmers have difficulty to study and research in this field. Describe a complete simulation process of the digital music system under program control and in an object oriented way. Take a bottom-up approach to describe the file structure, running process, encapsulation way, program interface, logical structure simulation and user interface of digital music system.

Key words: digital music; MIDI system; simulation; program control

1 数字音乐与MIDI

数字音乐是用数字格式存储的,可以用互联网和无线网络来传输的音乐文件[1]。数字音乐以产生于20世纪80年代的MIDI为基础。MIDI是"音乐设备数字接口"(Musical Instrument Digital Interface)的英文简写,是一个工业标准的电子通讯协定,为电子乐器等演奏装置定义各种音符或弹奏码,容许电子乐器、电脑或者它的演奏设备彼此连接、调节和同步,得到即时交换演奏资料。初期的数字音乐制作主要以电子乐器为主,随着计算机技术的日益成熟,数字音乐制作流程中的许多硬件相关设施已经完全可以由模拟软件替代,数字音乐的制作在脱离了MIDI键盘等昂贵的电子器材之后也渐渐进入普通用户视线[2]。

收稿日期:2009-12-28;**修回日期**:2010-03-16
基金项目:国家大学生创新实验计划(071069930)
作者简介:罗誉家(1988-),男,四川安岳人,研究方向为计算机科学与技术。

2 数字音乐系统结构描述

图1描述了经典数字音乐系统的结构。该系统由计算机和必要的外设组成,主要包含以下几个部分:

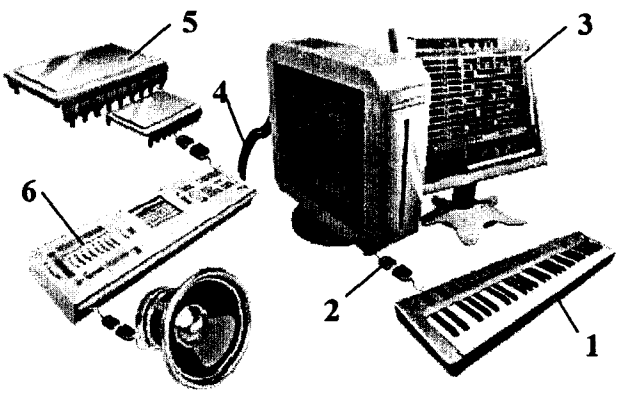

图1 数字音乐系统硬件结构图

1)MIDI输入设备:"MIDI键盘(或电子琴)"可说是最普遍MIDI输入设备,其主要工作是产生MIDI序列并向音序器发送。

2)MIDI接口:个人计算机与MIDI输入设备之间

数据传输的通道。

3）音序器（Sequencer）：通常是计算机中用于存储、回放、修改、控制 MIDI 序列的软件。

4）通道（MIDI Channel）：连接音序器与合成器，实现 MIDI 序列的有规则传输。

5）音色库（Sound bank）：是多种音色的集合。目前大多数音色库均为遵循“GM 音色排列标准”的软波表，音色以可被计算机识别的方式压缩并存储为音色文件。

6）合成器（Synthesizer）：播放 MIDI 序列，控制声音生成。

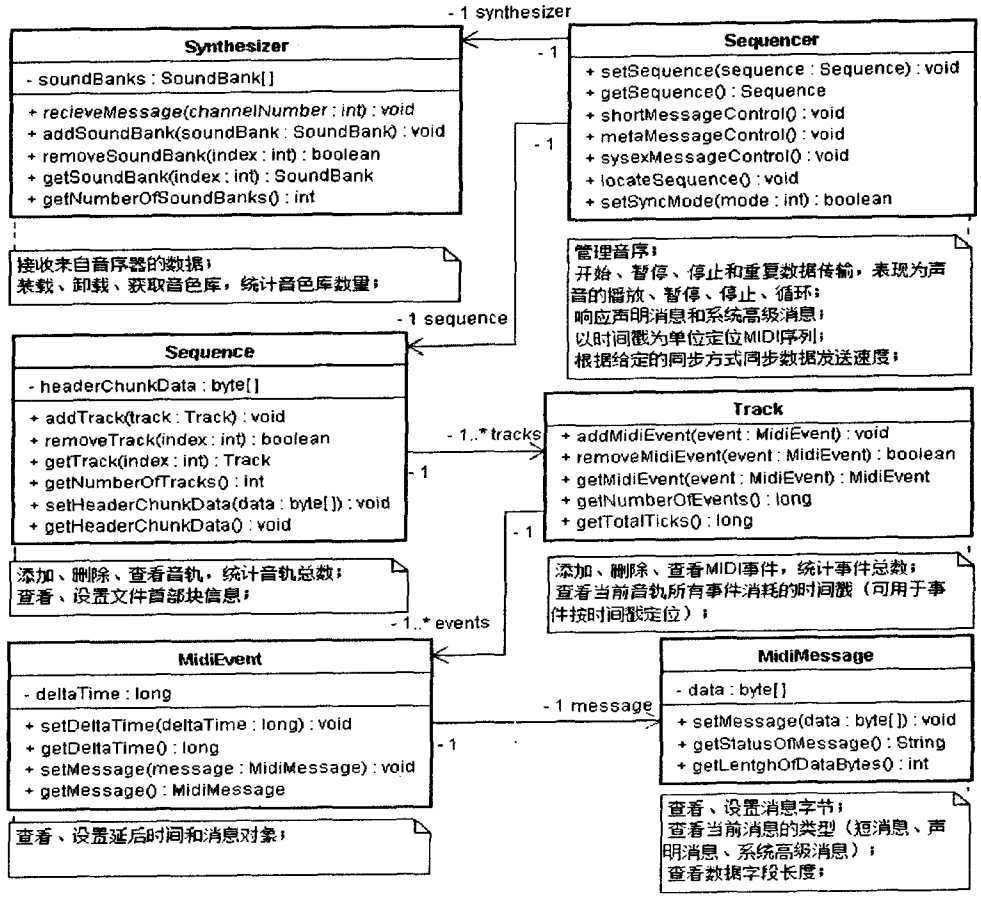

图 2 数字音乐系统模拟结构图

目前，各大主流的操作系统均已提供了自己的音序器、合成器和音色库，以实现对数字音乐系统的支持（例如，Windows 下自带的“GS 波表软件合成器”）。数字音乐系统便可简化为一个 MIDI 输入设备、一台个人计算机和必要的软件搭建起来的简易系统。

该文章将以数字音乐文件结构及播放原理为基础，从编程角度出发探究创建更加高效、更加符合用户操作习惯的数字音乐系统，探索通过程序可控的方式实现对“.mid”格式文件的支持，以创建完全脱离硬件设备的数字音乐系统的搭建流程。

3 程序可控式数字音乐系统实现

随着被计算机技术的发展所推动的数字音乐技术的成熟，现在的音乐制作者仅通过“个人音乐工作站”便可以实现一整个乐队的演奏任务。目前的绝大多数个人音乐工作站是一个集成了音序器、合成器、音色库及声卡和录音设备的全套设施，可实现数字音乐制作、编辑、加工和录制的全部工作[3]。图 2 为数字音乐系统模拟结构图。

程序可控方式的数字音乐支持依赖已有的各种模拟部件（如果操作系统不支持数字音乐，则需要通过编程方式[4]创建简易的音乐工作站，或者使用目前已有的相关软件产品）。下面，文章将基于数字音乐文件结构和播放原理，详细阐述利用程序可控方式实现对数字音乐的支持。

3.1 数字音乐文件的封装

“.mid”格式的文件以字节为单位进行存储[5]。众所周知，文件 IO 操作需要占用大量系统资源并且极易引发错误，因此从效率和稳定性性角度来说对 MIDI 文件的程序可控的最佳策略绝对不是直接使用 IO 控制字节文件本身。通常采用的方式是创建一个数据结构模拟“.mid”格式文件中的各个逻辑结构，并且直接对模拟数据进行操作，操作结束后通过 IO 按照规范将模拟数据写出为“.mid”格式的字节文件。规定了“.mid”格式文件的模拟数据结构之后，数字音乐模拟系统的所有操作都将围绕对此数据结构的操作展开。

3.1.1 基本封装

为了使写出操作更高效、更稳定，映射“.mid”格式文件的数据结构应当符合以下条件：

1）尽可能地模拟图 3[6]所描述的 MIDI 文件的逻辑结构；

2）可实现 MIDI 文件系统中规定的所有类型的数据存储，并具备一定的容错、纠错能力；

3)以字节为单位存储数据;

4)数据可控,增、删、改、查、遍历等操作的性能良好。

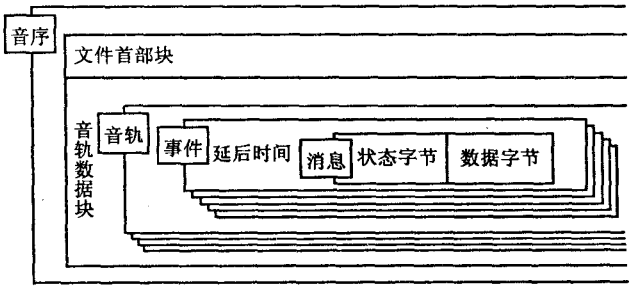

图 3　MIDI 文件逻辑结构图

除此之外,还应当在遵循 MIDI1.0 规范的前提下具备一定的可扩展性,以便向后兼容更新版本的 MIDI 规范。参照以上条件,利用模拟数据结构封装 MIDI 文件的逻辑结构可按照以下步骤进行(所有类的编程接口请参见图 2):

(1)创建"MIDI 消息"类作为基本数据单元,维护一个以字节为单位的容器以包含 MIDI 消息中的数据,还可包含一些诸如消息类型(短消息、声明消息等)、数据域长度等信息。

(2)创建"MIDI 事件"类。组合 MIDI 消息类,包含 MIDI 消息的基本管理功能,并维护一个延后时间域。由于 MIDI 事件是顺序相关的,为了避免重复和提高效率,在使用延后时间和 MIDI 消息作为事件对象的唯一标识时,可将延后时间定义为相对于音轨起始位置的绝对时间。以绝对时间为标志对所有事件进行排序,可实现事件的精确识别、定位,并提高写出效率。

(3)创建"音轨"类作为 MIDI 事件类的容器。首先,音轨中维护的 MIDI 事件是顺序相关的,因此事件的添加操作必须保证 MIDI 事件以绝对后延时间为标志被排序;其次,音轨结束事件应不被用户可见,并且应随音轨的创建同时被创建。

(4)创建"音序"类作为音轨类的容器,并提供完整的音序封装。其中,将文件首部块抽象为以字节方式存储的音序类的私有域(而不是创建独立的"文件首部块"类)可提高效率。同样,对默认信息的初始化过程(例如默认的时间戳时长定义)应当对用户透明。

3.1.2　基于数据封装的数字音乐输入输出控制

模拟软件或通过 MIDI 接口与 PC 机相连的数字音乐硬件是音序器的直接输入设备,音序器是合成器的直接输入设备。音乐制作者在 MIDI 输入设备中调节音量、调整音调、定义音效并手工弹奏乐谱(或使用软件谱曲),这些操作均被定义为事件并被音序器记录,音序器然后按照给定规则向合成器输出相应事件。在程序可控方式的数字音乐系统中,MIDI 序列的输入过程将完全脱离专业硬件设备而由计算机相关输入设备(如鼠标、键盘等)取代。

在输入模拟阶段,键盘、鼠标等设备的动作应当可以实现以下操作:

1)发送音序器可识别的 MIDI 序列;

2)MIDI 序列生成控制,可实现各种消息类型、消息数据的生成。

输出操作相对较易实现。输出包括两个部分:".mid"格式文件的输出、声音输出。文件写出此处不再赘述,声音输出的详细实现方案请参见"合成器程序可控"一节。

3.2　音序器程序可控

在数字音乐播放阶段,音序器是用于解析 MIDI 文件的核心组浸。音序器通过 MIDI 接口接收来自 MIDI 输入设备中的 MIDI 序列并存储,必要时对其进行修改、控制或者发送至合成器。一般而言,音序器将提供如图 2 所描述的编程接口。

如果希望音序器程序可控,则需要通过操作系统接口获取已有的音序器。如果当前系统未提供音序器,则需要通过以下步骤创建模拟音序器:

1)编写一个基于模拟数据结构的容器,该容器可接收、存储、修改、转发模拟数据;

2)提供对音序中各个数据结构的解析功能和对 MIDI 消息的分析功能,定义系统高级消息的处理方式;

3)提供对音序中各个数据结构的基本控制功能,并可对各个事件进行精确定位;

4)提供对合成器的支持,可以获得合成器所控制的一组通道并向其发送数据;

5)封装底层实现原理,提供基本的编程接口。

3.3　MIDI 序列在通道上的传输控制

3.3.1　音序器对合成器的支持

音序器对合成器的支持实际上是定义 MIDI 序列在通道上的发送方式。通道以 MIDI 消息为单位传输数据[7]。音序器利用 MIDI 序列的文件首部块声明信息和音轨中的节奏声明信息确定 MIDI 事件的最小时间单位 tick,并且在运行时按顺序以 MIDI 事件中给定的 tick 处理所有事件。该处理过程按照 MIDI 消息的类型分为三种情况:

1)短消息的处理:任何一个短消息中均包含一则通道信息,因此称短消息是"通道相关"的。音序器从短消息中解析出该消息对应的通道并将其发送至该通道。

2)声明消息的处理:一则声明将被音序器处理,但由于声明消息中并不包含通道信息,因此声明消息将不被发送。所以,声明消息只会对音序器起作用,故可以使用某些声明信息通过控制音序器实现 MIDI 序列的播放控制。

3)系统高级消息的处理:大多数系统高级消息属于保留域或不具备实际意义,但是某些特殊的系统高级消息要求立即被处理。模拟音序器可自定义对系统高级消息的处理方式,以实现对 MIDI 指令的功能扩展。

3.3.2 合成器对音序器的支持

合成器对音序器的支持是指(电子或模拟)合成器对由通道上接收到的数据的响应方式的实现。其中,将由合成器解释的通道力度控制消息(状态位的值从 208 到 223)和音调调节轮(状态位的值从 224 到 239)已不常用,其他需要关注的消息类型有[8]:

1) 打开(Note on)/关闭(Note off)某个音色:消息状态位的值位于 128 到 143 的 16 个短消息表示关闭由数据位指定的某个音色,位于 144 到 160 间的 16 个短消息则表示打开指定音色。如果两个分别表示打开、关闭的消息具有相同的数据位,则这两个消息为一组消息。一组开/闭消息在宏观上表现为具备特定音量、音调的一个音符。MIDI1.0 标准明确表示,一个音色一旦被打开,则必须被关闭。

2) 触后量控制(Polyphonic aftertouch):消息状态位的值从 160 到 175 间的 16 个短消息表示触后量信息更新。在当前通道上正处于打开状态的音符在关闭之前,其压力信息可被调整。模拟电子琴的琴键所受到的压力宏观表现为音符的音量,即通过该类消息可简单地调整当前正在被播放的音符的音量。

3) 控制模式切换(Control mode change):消息状态位值为 176 到 191 的控制信息是很常用的字段。控制消息可根据数据位的描述更改当前通道在合成器中已注册的播放方式,达到控制音效的目的。大多数音效均需要使用控制字段实现,例如滑音、颤音、左右声道、踏板效果等。

4) 音色切换(Program change):消息状态位值为 192 到 207 的短消息将通知合成器切换消息中指定的通道在音色库中所注册的音色。

3.4 合成器程序可控

合成器是直接控制发声的部件。该部件接收来自通道的 MIDI 序列并调用音色库中的相关数据以声音的方式解释 MIDI 序列。合成器的程序可控应当包含以下三个方面内容:提供对音序器的支持、提供基本的编程接口、管理音色库操作接口。

合成器利用音色库控制声音的生成。按照 MIDI1.0 规范,在初始化过程中,与合成器直接相连的 16 条通道需要被注册为 GM 标准中所约定的 16 种音色。如果一个音轨未指定音色切换信息,那么合成器使用默认音色播放音符。合成器对音序器的支持、初始化过程等操作都应当是用户透明的,并提供如图 2 所示的编程接口。

模拟合成器的主要操作对象是软波表。许多合成器通常自带软波表(大型软波表生产商也可能令软波表中自带合成器),也支持扩展软波表。如果当前系统未提供合成器,则一般不会单独提供软波表,此时则需要通过网络获取由专业软波表制造公司提供的资源,不推荐自己模拟软波表。因此模拟合成器对软波表的支持主要表现为通过外部软波表的操作接口对其进行控制,以实现其对音序器的完全支持。

目前大多数操作系统均提供了合成器,如果当前系统未提供,则需通过以下步骤创建[9]:

1)使用特定的数据对象模拟 16 条数据通道,并将这些对象与音序器相连以实现数据传输;

2)编写一个 MIDI 序列处理对象,创建到指定软波表的访问接口(不同数据格式的软波表有不同的访问方式,这些信息在软波表生产商提供的帮助文档中可以获取);

3)创建对通道的监听端口,以便可以在任何时候接受来自音序器的数据;

4)实现对音序器的支持(即数据解析、响应方式支持);

5)定义对象的初始化动作,注册模拟通道的默认音色;

6)实现基本编程接口。

4 数字音乐模拟系统的封装和扩展

4.1 用户操作接口设计

完整的数字音乐系统模拟最终应为用户提供如下操作接口以实现透明地操纵该系统:

1)基于计算机输入设备的动作输入(例如音符输入、音效调整等,这些动作可产生 MIDI 序列)和实时声音输出;

2) 音序基本属性(例如 tick 精度)初始化控制;

3) 存储模拟数据结构的中介文件的输出和已存在文件的输入[10];

4) ".mid"格式文件输出和已存在文件的输入;

5) 已输入数据的增、删、改、查操作;

6) 已输入数据的回放、定位、控制;

7) 添加、卸载音色库。

除此之外,许多个人音乐工作站还提供录音接口,可将数字音乐录制、压缩为通用音乐格式文件。

4.2 模拟过程中性能相关因素的解决方案

4.2.1 性能制约因素

实践证明,在整个模拟系统的运作流程中,如下操作最有可能成为效率瓶颈:

1) 原始数据导入:".mid"格式的音乐文件在导入模拟系统时需要被编译为模拟数据结构。

2) 容器扩容:一首 MIDI 音序中可能包含大量事件,需要使用大容量数据结构处理。如果使用数组作为容器类的实现,扩容将产生一次遍历。

3) 数据插入、删除,目标事件查找:使用线性数据结构实现容器类时目标事件查找操作很可能导致一次遍历,插入、删除则依赖于线性数据结构的类型;非线性数据结构可优化此操作性能[11]。

4) 插入数据的时间戳排序:这是一个往已排序的顺序容器中插入新对象的问题,该操作可保证事件的顺序相关性。

5) 文件写出:文件写出操作必然导致一次遍历,然而相比于 IO 操作,数据结构的选择对该过程影响并不十分明显。

4.2.2 解决方案及性能评优

模拟系统的 IO 操作可能导致的性能问题主要表现在".mid"格式文件的导入和导出,受 IO 本身的资源开销和".mid"格式文件到模拟数据结构的编译过程开销两方面影响,其中后者可被改善:通过 IO 操作直接将模拟数据结构写出为中介文件,必要时才将其编译为".mid"格式文件。不过,相比于 IO 操作而言编译步骤的资源消耗仍然很有限。

从性能制约因素中可看出,容器类的选择对整个系统的性能有巨大影响。模拟系统存在三个容器:MIDI 序列的容器、MIDI 事件的容器和音轨的容器。其中,由于不同类型的 MIDI 序列长度是可知的,音轨的最大数量也是可知的,因此这两类容器均可以使用简单的可扩容数组实现,因此 MIDI 事件的容器就成了影响系统性能的关键因素[12]。

现文章对以下实例进行简要分析:设用户将创建一个含一条音轨的音序,该音轨中包含 N 个事件,各个事件的绝对延后时间允许相同;创建结束后,将在随机位置对事件进行添加、删除、修改操作。设完成一次简单操作(创建、修改、添加、删除、写出事件)的时间复杂度为 $O(1)$。

在采用数组实现方案时,设数组初始长度为 $L(L < N)$,每次扩容长度变为原来的 2 倍。创建时需要对事件按照时间戳排序,必要时进行扩容,此时创建过程时间复杂度为 $O(N\log_2 N)$,修改、删除、添加过程使用二分法搜索目标位置,时间复杂度为 $O(\log_2 N)$;相比之下,使用以时间戳为键值的哈希表来设计系统时这两项数据值分别为 $O(N\log_2 N)$ 和 $O(1)$。

可见,好的数据结构选择对系统性能有相当大的影响。当然,在此可能会有其他更高效率的封装方式,期待有兴趣的读者对此进行更深入的研究和实践。

5 结束语

该系统所建立的一整套模拟流程已在"基于开源技术的数字音乐在线制作平台"和"数字音乐模拟系统"项目中得到成功实现。

实践证明,针对数字音乐系统的自下向上模拟过程具备较完善的理论基础。然而,数字音乐系统的外部扩展性(例如添加音序器对录音的控制、超过 16 位的数字音乐模拟)及相关实现方案中的性能优化等内容仍有待进一步研究。

参考文献:

[1] MIDI 百科[EB/OL]. 2009. http://baike.baidu.com/view/7969.htm? fr=ala0_1.

[2] International Federation of the Phonographic. Digital Music Report 2009: New Business Models for a Changing Environment[R]. IFPI, 2009.

[3] 浅谈 MIDI 技术的应用[EB/OL]. 2007. http://www.go-gddq.com/html/2007-01/410057.htm.

[4] 刘永志. MIDI 播放编辑器的实现[J]. 电声技术:2006(7):46-48.

[5] Cunningham S. An ethnographic study of music information seeking: implications for the design of a music digital library [M]. Washington, DC, USA: IEEE Computer Society, 2003.

[6] Rothstein J. MIDI: a comprehensive introduction[J]. Computer Music and Digital Audio Series, 1995, 7: 23-27.

[7] 徐云强. 基于 S3C2410 的 MIDI 音乐播放系统研究与开发[D]. 武汉:华中师范大学, 2007.

[8] Yamaha Corporation. Motif-rack ES Data List, 411MWAP 28.2-01A0[R]. Yamaha Corporation, 2004.

[9] 潘晓利. 基于 MIDI 模块的音乐发生器设计[J]. 电子测量技术, 2006, 30(2): 108-110.

[10] 沈昌松. MIDI 音乐合成的研究与实现[J]. 微机发展(现更名为计算机技术与发展), 2001, 11(5): 1-3.

[11] 赵　芳. 基于音轨特征量的多音轨 MIDI 主旋律抽取方法[J]. 计算机工程, 2007, 33(2): 165-167.

[12] 李　涛. 基于 MIDI 的汉语普通话语音合成算法[D]. 上海:复旦大学, 2004.