Position Paper for Music Representation Panel

Lounette M. Dyer
CCRMA
Stanford University
Stanford, CA 94305
loon@parcplace.com

Introduction

A multitude of computer music software applications have been developed over the last two decades encompassing a wide variety of application areas, including music synthesis, music editing and printing, and computer aided composition. More recently, realtime control of music synthesis has become an important application area. Developers of computer music applications typically use music representations that are tailored to the specific application. In the case of synthesis applications, the representation was usually very tightly coupled to the particular synthesis algorithms and synthesis hardware. As a result it was difficult or in some cases impossible to share scores among applications or to perform scores on computer systems that were different from the one that originated the score. Attempts to write conversion programs between representations were not very successful as information was often lost between these application-specific representations. (For example, many applicatons used a *keynumber* to represent *pitch* which caused enharmonic pitches to be aliased.)

CAD Workstation for Musicians

The availability of powerful and inexpensive personal computers (with a graphics monitor and a mouse) and realtime synthesizers have caused computer music systems to rapidly evolve toward single user systems that support heterogenous synthesis hardware (MIDI synthesizers, DSP boards, etc.) and realtime control hardware. Thus, the time seems ripe for the development of the musician's equivalent of a CAD workstation (Music-CAD or M-CAD)—a workstation for the creation and manipulation of music and sound. Such a workstation would provide an integrated set of tools for all aspects of music and sound creation, from inception, through experimentation, to printing and performance. An M-CAD system would benefit greatly from a standard computer music representation by providing a foundation for the integration of applications.

Computer Music Representation

A music *notation* is a system of written symbols, a language if you will, by which musical ideas are represented and preserved for study and performance. Thus the notation acts as a set of instructions to performers who will create the sound of the music. A *computer music representation* (CMR) could be thought of as a digital encoding of a music notation, by which musical ideas are represented and preserved for performance and study by *machines.*

A CMR must provide at least three components: a standard ASCII score file format, run-time data structures, and a set of manipulation functions. M-CAD systems are somewhat unique in that new computer music hardware technology (both synthesis and realtime control hardware) is being introduced at a very rapid pace. Therefore, the envisioned M-CAD system would tie together the three above components with a design methodology that

would help developers to both build portable applications and to extend their applications to accomodate new hardware.

Basic Requirements

The CMR must be *portable* across applications as well as hardware platforms and synthesis hardware. To insure portability, the CMR must be:

- application independent;
- platform (workstation/OS/language) independent; and
- device (synthesis hardware and control interface) independent.

The CMR must provide some level of *compatibility* with the large body of existing score data in various formats (such as MIDI event lists). The compatibility constraint requires that the CMR support:

- extraction from and translation to existing file formats *without information loss*; and
- incorporation of new score I/O mechanisms.

The CMR must also support:

- a variety of styles of music; and
- structured score construction (including hierarchies with templates and instantiation).

CMR Data Model

The process of defining a CMR involves indentifying the basic elements in a score and representing the information so that it can be stored and manipulated by digital computers. Scores contain many different types of information. A partial list of these basic elements is shown below.

- **note**—pitch and duration (in *beats*)
- **note interpretation**—articulation and expressive nuance (e.g. tenuto, stacatto, accents, etc.)
- **event**—note and note interpretation
- **key**—tonality (but not limited to Western scales)
- **tempo**—time rate for beat (can be constant, a time varying function, or controlled via realtime input)
- **meter**—beat stress patterns
- **style**—overall *phrasing* or "feel"
- **global interpretation**—key, tempo, meter, and style

These basic score elements can be organized into *scores* with structural components, examples of which are shown below.

- **event list**—sequence of events
- **phrase**—event list and global interpretation
- **part**—sequence of phrases
- **score**—parallel set of parts

In synthesis applications, the above basic score elements comprise two categories: *static* elements that can be bound when the score is read from a file, and *dynamic* elements that may be controlled with realtime inputs. Non-realtime synthesis applications may have all static symbols, whereas realtime applications may have many (or all) dynamic symbols. Thus, realtime applications dictate further requirements, in that the CMR must:

- be based on an *abstract symbolic* representation;
- have an *extensible vocabulary*;
- separate the representation from the interpretation; and
- support late binding of symbols.

Object-Oriented Approach

There are several approaches to defining a CMR to encompass a variety of diverse applications. In the *intersection* approach the representation includes only those elements that are common to all application areas. Another approach is to provide the *union* of all elements that are in at least one application area. An approach that is preferable to either of these is to divide the representation into two parts: a generic core (approximately the intersection), and a mechanism for application-specific extensions. *Open* applications with a mechanism for extending the representation could be used not only by developers, but also by experienced users to customize and evolve applications to suit their specific needs.

Object-oriented languages (Smalltalk-80 in particular) are well suited to the needs described above because they provide:

- powerful abstraction and data modeling facility (via *objects* and *message passing*);
- inheritance (facilitating high code reuse and development by refinement); and
- unbounded polymorphism.

An object-oriented methodology is well suited for defining the CMR. The description language of the CMR would consist of class definitions, interface protocols, and an ASCII file format. Application developers could subclass any of the score objects listed above, and could selectively override protocol. For example, MIDI applications can subclass **Note** with **MIDINote** and use a **keynumber** in the **pitch** instance variable. (Unbounded polymorphy allows the instance variable to have any value, so the **MIDINote** methods would be written to manipulate **keynumber**.)

Conclusion

The benefits of defining and adopting a standard *computer music representation* for M-CAD systems are numerous, including support for the integration of applications and the portability of scores across applications and hardware. A set of requirements can be created by considering specific applications, giving special attention to realtime control applications as they present very specific needs. An object-oriented approach appears to best support the flexibility and dynamical properties of the M-CAD environment.