SHADOW: An Object-Oriented Performance System for the DMIX Environment

Daniel L. Oppenheim
Center for Computer Research in Music and Acoustics (CCRMA)
Stanford University
Dan@CCRMA.Stanford.Edu

SHADOW is an object-oriented performance system implemented in Smalltalk-80. It is unique in that it is an integral part of DMIX—a high level environment for music composition. Hence, there is no imposed distinction between the acts of composing and performing. Tools normally associated only with performance can now be used by the composer to enhance musical expression or to experiment with interactive performance settings. This was found especially useful for composing in a style we term Experimental Computer Music (ECM), for which the more ‘conventional’ (i.e. instrumental) paradigm, used in many existing score-tracking applications, did not prove adequate. SHADOW’s object-oriented design enables it to dispatch messages to any object in DMIX. Hence not only is it a score-tracker that synchronizes an accompaniment to a human performer, but also a general mechanism through which the performer can control the computer and interact with it in real-time. This opens new domains for both composition and performance-practice whereby not only tempo, but also such parameters as pitch, harmony, rhythm and texture can be controlled by the performer within the context of a live performance.

Motivation

Several excellent score-tracking programs have been developed in recent years (as examples see Dannenberg 86, Dyer 90, Mathews 89, Paukett 90, Vercoe 84). These are by and large stand-alone applications into which the composer enters his completed composition: i.e. a score that consists of the performer’s part and an accompaniment. The performer’s part is typically a list of triggers. A trigger waits for the performer to play some predefined musical gesture (a note, a dynamic level, a Midi controller, etc.) and then initiates some action—usually the playing of the accompaniment in the performer’s tempo. The order of the triggers is predetermined in the score and cannot be changed during a performance. This paradigm models a conductor performing ‘conventional’ instrumental music and we shall refer to it as Horizontal Tracking.

Although this approach works well for music that is more ‘conventional’, several limitations become apparent when composing in a more general style that we shall term Experimental Computer Music (ECM).

Linear Versus Horizontal Tracking

The computer opens new domains that can dramatically widen the horizon for both composer and performer. For example, the accompaniment might be generated on-the-fly by a process (algorithm) without requiring a predefined score. The performer’s musical-gestures can then be used as triggers that define parameters used by this process. Thus, the performer can have expressive control over new musical parameters that are not dealt with in traditional performance-practice. Performance gestures can be used as triggers that control pitches, harmonies, rhythms, timbres, textures and much more. This approach is somewhat akin to improvisatory sections in jazz or modern music.

ICMC 281
However, as the generating process (algorithm) is run by the composer, he still has full control over the resulting music. In other words, such music could still be perceived as the 'same' in each performance, as changes in tempo or gestures do not 'change' traditional music. Since the musical decisions that determine the new parameters are thus made within the context of a live performance, it no longer makes sense for the composer to predetermine the correct ordering of triggers.

We term situations in which new types of musical parameters are set by the performer in real-time, and within the context of a given performance, as vertical tracking. In DMIX, the class Echo and its subclasses are used to create processes that enable such interactive performance settings (Oppenheim 89). SHADOW can be used to spawn and terminate these processes, and hence the techniques of both linear and vertical tracking can be freely intermixed.

A Single Environment for Composition and Performance

The composer of conventional music writes in a musical style that is well defined and can therefore often predict how his music, the score, will be perceived. However, when using new musical resources that are now available in ECM, such as the interactive performance settings described above, things become problematic. In such situations, it can be very hard to predict how the resulting music will be perceived without trying it out (experimentation). It does not make much sense to enter ECM-composed in one environment into a separate application for the sake of performing it, since either the result will be unpredictable or the composer will have to limit his musical venture to a style that has already been well explored. In our view, it is therefore highly desirable to have a single environment in which the composer can experiment with the use of new musical materials and simulate their effect within the context of an actual performance.

Other important advantage are discussed in a separate presentation included in these proceedings (Oppenheim 91).

Object Oriented Design

Events, EventHolders and Scores

Music events are represented in Dmix by two main classes. The first represents SingleEvents, such as a MidiNote, and the second a collection of time-stamped music-events (single or non-continuous) and is called an EventFolder. Any EventFolder can be used by SHADOW as a score for tracking a performer. The EventHolder will be converted into a Score—a subclass of EventHolder that adds the ability to keep track of rehearsal marks (Score is capitalized to indicate that it is a class in Smalltalk). Thanks to the property of code-reuse, any Score can be played (to simulate a performance), edited in graphic or text editors, or saved on disk.

Triggers, Performances and Links

Once a Score has been created the composer can attach to it the accompaniment. In other words, the composer has to determine what events in the score will be used as triggers and then link to them a performance: the specific action that will take place when the trigger is fired. Events that are designated as triggers are removed from their Score and replaced by a PerformanceLink. Each PerformanceLink holds onto two objects: the original event that is now a trigger, and the performance.

When the Score is played from within Dmix, it simulates a performance; all of its events are played in the normal way, whereas PerformanceLinks play both the trigger and performance. However, when SHADOW is tracking a Score it will advance its position within the Score as it follows the notes played by the performer. Whenever the performer plays an event that is a trigger, SHADOW will first execute the attached performance and then advance its position in the Score.

Two different kinds of performances may be linked to a trigger: a music-event, or a DBlockContext. The first is used by and large for playing the accompaniment in synchronization with the performer and in his tempo. DBlockContexts are blocks of compiled Smalltalk code that get evaluated (executed) when their trigger is fired. This provides a completely
general mechanisms for interacting with the entire Dmx environment: messages can be sent to any object in Dmx, new objects can be created, processes that generate music on-the-fly can be spawned, graphic displays can be updated, and more.

The User Interface

The user interface design takes special care to meet the needs of both composer, in preparing a score, and performer, in performing it.

The PerformanceLink

A comment and message may be added to each PerformanceLink. Comments are strings of any size. Messages are short strings (3-4 characters long) intended as a brief reminder of what the performance is. This message is also used to print out a short string for displaying efficiently as much information about the PerformanceLink as possible (the TriggerString). For example, the string "s6(Do30)" indicates that the performance is a DBlockContext (""), the trigger is waiting for a note with a pitch of "s6", and the action that will take place is the one that was assigned with the message "Do30" (see the second to last trigger in Figure 2). This information is useful for both composer and performer.

Preparing a Score

Several mechanisms for attaching performances to a Score are provided; only two will be described. The simplest is via the standard graphic edit view, and is done by a mouse and click (Figure 1). Events that have been replaced by a PerformanceLink are colored black. CodeDictionaries provide a powerful mechanism for determining what type of performance will be linked—a music event or DBlockContext. After a PerformanceLink has been added to the Score, it can be edited further, as illustrated at the bottom of Figure 1.

![Figure 1](image1.png)

An alphanumeric trigger-list editor was found more suitable for music that employs many triggers that control interactive performance settings (Figure 2). In such situations, graphic views do not indicate with sufficient clarity what each trigger does, and it becomes increasingly difficult to manage large Scores. On the other hand, the trigger-list editor displays each PerformanceLink's message and comment. Furthermore, rehearsal marks indicate clearly the grouping of triggers within musical actions.

![Figure 2](image2.png)

Any item in the list—PerformanceLinks and rehearsal marks—can be selected and then edited or played. New triggers can be inserted into the list with the aid of CodeDictionaries.

ICMC 283
Performing a Score

The PerformanceTracker is the user-interface used by the performer (figure 3). The 'Display' panel shows the PerformanceLinks that are in its tracking window by printing out their triggerString. The 'Tempo' panel indicates the performer's tempo. The 'GoTo' panel updates the rehearsal mark position as the performer plays, and also enables him to position the tracker at any rehearsal mark. The 'Performer' panel includes a set of switches for pausing the tracker, returning to the previous rehearsal mark or jumping to the next mark. These switches can have triggers assigned to them, and then be activated during a performance. The 'Track' panel is a switch for turning tracking on and off. The 'Do' panel contains various utility features, such as play and on-line help.

Conclusions

SHADOW has been used successfully in performing the author's Concerto for MIDI Violin and Dmin. Via SHADOW, the violinist was able to play the combined role of soloist and conductor. By communicating with instances of Echo, SHADOW not only spawned processes that generated music on-the-fly, but also updated their parameters in real-time in response to the performer's gestures. This provided a flexible accompaniment which gave the performer much expressive freedom.

Smalltalk is a slow language that is not generally considered well suited for real-time applications. However, Smalltalk held up extremely well in a highly demanding performance situation that lasted over 20 minutes, running on a Macintosh IIx with 8MB of RAM. A further significant improvement in performance can be obtained by implementing a low-level driver and scheduler in C and linking it to Smalltalk via its primitives mechanism. Our experience has shown that the many benefits of working in Smalltalk greatly outweigh the slowdowns in performance.

References