Objed and the Design of Timbral Resources

W. Buxton, S. Patel, W. Reeves, and R. Baecker
Structured Sound Synthesis Project (SSSP)
Computer Systems Research Group
University of Toronto
Toronto, Ontario, Canada M5S 1A1

Introduction

One of the main attractions of electroacoustic music is the potential for composers to design and control their own palette of timbral resources. A key frustration, however, results from the difficulty in actually doing so. Consequently, in spite of technological advances, systems appearing on the market are reminiscent of organs and conventional instruments in their reliance on preset timbres. Very little progress seems to have been made in the development of tools to aid composers in "rolling their own" timbres.

With analog equipment, the musician could use a "hands-on" approach to exploring the timbral potential of various configurations. The adoption of digital technology provided greater precision, sophistication, and potential, but initially made timbral exploration more remote. With the current trend toward highly interactive real-time digital systems, the composer should now have the best of both worlds. It is toward such an end that this paper is directed.

Ideally, timbre should be controlled according to perceptual rather than acoustic attributes. However, our limited understanding of perception, cognition, and acoustics makes this difficult at present, although work by Grey (1975; 1977) and Wessel (1979) is making considerable headway in this direction. If we must deal with timbral specification in mainly acoustic terms, it is important to provide an environment for doing so that minimizes the nonmusical problems of the task and that permits the composer to develop an ability to understand and predict the perceptual consequences of changing acoustic parameters. The realization of such a timbre editing environment requires that:

1. The data being edited is clearly represented.
2. There is a good correspondence between the things that the composer wants to do and the operators that the editing environment provides.
3. The bookkeeping and administrative tasks of the user are minimal.
4. The syntax for all transactions is succinct, intuitive, and consistent.
5. The system have a high tolerance for user error, and that it encourage exploration and experimentation (i.e., learning).

We will examine one program that attempts to fulfill these requirements. The intention is to provide a case study that will serve as the basis for discussion of the issues involved. It is hoped that as much will be learned from the program's deficiencies as from its strengths.

Objed and Objects

Objed is a program that permits named sets of timbral characteristics (called *objects*) to be defined, auditioned, and modified. Objects have three properties:

1. Each is contained in a separate, uniquely named file.
2. Each defines a set of timbral characteristics (including time-varying functions) that can be used to orchestrate one or more notes.
3. The parameters determining pitch, maximum amplitude, and duration of each instance of an object are determined by the

Patel is currently with Human Computing Resources, Toronto, Canada. Reeves is now with Lucasfilm Ltd., San Rafael, California. Baecker is with both University of Toronto and Human Computing Resources.

Computer Music Journal, Vol. 6, No. 2, Summer 1982,
0148-9267/82/020032-13 $04.00/00

orchestrated note rather than the object itself.

Objects can be compared to Music V instruments (Mathews 1969), and *patches* on analog synthesizers. The usage described derives directly from Truax's POD system (1976). The concept is general, but the current SSSP implementation restricts the composer to working with the following limited set of object types (type is determined by the method of sound synthesis employed): *fixed waveform, frequency modulation* (FM) (Chowning 1973), *additive synthesis* (Moorer 1977), *Vosim* (Kaegi and Tempelaars 1978), and *waveshaping* (Arfib 1979; Le Brun 1979). Each object type is like a Smalltalk "class" (Krasner 1980), although the two were developed independently.

The composer's task in object editing is one of plugging values into the fixed template defined for the object type being used. As a result, the composer works at a level halfway between the two extremes of instrument definition at the unit-generator level (as in Music V-type programs) and the use of presets. (*Unit generators* are signal processing elements, which are the lowest-level modules used in defining Music V instruments.) In this approach, the composer is restricted by the object types available as to the range of timbres that can be generated. However, if the available object types are well chosen, a musically rich palette of timbres is possible. Furthermore, the number of object types or the synthesis techniques used are not restricted by the object concept. New methods can and have been introduced. (In fact, the concept of a meta-object editor would be worth developing, so as to facilitate the specification of new object types, using the unit-generator concept, for example.)

Our rationale for taking the somewhat restrictive template approach of the object formalism is based on the resulting ability to build a powerful user interface for timbre specification. Since each object has a system-defined template, the program can provide an "intelligent" editing environment that "understands" what the user is trying to do. Errors are detected more easily, diagnostic messages are clearer, and the assignment of default values is simplified. While the timbre space offered by this approach is limited when compared to the *potential* of the Music V instrument paradigm, in many cases the tools offered by Objed, for example, give the composer control over a broader timbral range. Potential is only meaningful to composers if it falls within their "threshold of patience." The approach exemplified by Objed is intended to extend this threshold as far as possible.

The use of objects, as described, is not incompatible with either Music V-type instruments or presets. Objed provides a middle ground between the two. As has already been pointed out, the unit-generator approach could be used to define the template of new object types. On the other hand, objects whose parameters have been previously defined can be used as presets. In this case, the composer can select the preset closest to the timbre desired and use Objed to "clone" a new object that is a variation of this existing one.

Objed in Perspective

Before progressing, it is important to put the task of editing objects in perspective with respect to the overall task of composing using the SSSP system. The description given below is brief. Those readers wishing detailed information on the software available in the SSSP system are referred to *Music Software User's Manual* (Buxton 1981).

We view the SSSP software as providing suitable environments for performing three basic tasks:

1. Defining and editing a palette of timbres to be used in a composition. This is referred to as *object definition.*
2. Defining and editing musical scores, including the task of orchestrating the notes of these scores, using the lexicon of objects defined by the composer.
3. Defining the performance information affecting a composition, usually by real-time interpretation, or conducting, of the material.

Each computer program is designed to assist the composer in the performance of one of these three tasks. Objed, for example, is the primary environment for performing the first task. Usually, how-

ever, there are alternative ways of performing a particular task. The most appropriate environment for expressing one musical idea during score definition, for example, may not be the best for expressing some other (equally valid) idea. Hence, the two different programs Sced and Scriva (Buxton et al. 1979), support alternative approaches to performing the second task of score editing. For carrying out the third task (performance), the Conduct program (Buxton et al. 1980) is the environment available.

There are two benefits gained from viewing the software in terms of different task environments. First, composers are provided a simple mental model with which to view what they are doing; that is, in terms of one of these three tasks. As the number of alternative ways of performing a task increases, having this mental model available becomes more and more helpful to the composer. Second, this approach allows the software to be structured in such a way that no order of performing these three tasks is forced upon the composer. In fact, the composer can ignore tasks that are not of immediate interest. For example, if the composer just wants to deal with scores, the material can be auditioned with automatically assumed default timbres from within the score-editing environments. Similarly, as we shall see in more detail, the composer who wants to work on timbres without having to think about scores may do so, by using Objed. The point is that the system can be introduced and compositions built up incrementally. The composer need not be confronted with detail until it is needed.

In practice, the expert composer usually jumps from environment to environment. As stated earlier, the system is designed to facilitate this type of action as well. For example, one can jump into a score-editing environment from within Objed. This is often useful if one suddenly wants to specify a special score to test the object being designed. Similarly, one can temporarily jump into Conduct or Objed from any of the score editors, without ever having to suspend explicitly or save the material being worked upon. It would be fruitless to try to design one environment that fulfills all musical needs. Our philosophy is to make a number of strong, specifically oriented, software packages and obtain completeness by allowing the composer to access one environment from within another.

On Entering Objed

On initial entry, the Objed display is as shown in Fig. 1. The display is divided into five regions, each with a different function. The main region, which occupies the upper two-thirds of the screen, is where the actual data being edited is displayed. In the example, a simple, fixed-waveform object is being edited. Its components, all of which are graphically displayed, are a waveform; a time-varying function, controlling pitch; and an envelope, controlling the contour of the note's volume.

To the left, below the main region, is an area containing data pertaining to the pitch, volume, and duration at which the object being edited can be auditioned. These values are not part of the object: they are simply conveniences for exploring its behavior at different pitches, durations, and volumes.

The central panel in the lower part of the screen contains various options that allow the user to change the state of the editing environment. The user can change the type of object being edited and change the way in which the edited object can be auditioned.

The panel to the lower right is dedicated to the saving and retrieving of objects with a minimum of effort. The elongated region along the bottom left edge is a *window* that permits the composer to access the "outside world" without leaving Objed.

Objed is designed to minimize the amount of rote learning that must be undertaken by the user. Rather than memorize a large number of commands, the user need only remember a simple strategy that forms the basis for all interactions: when one wants to change something, one just points at the diagram or word that represents it on the display and depresses the selection (Z) button on the tablet's cursor. Any consequent options will then be presented, and the same method of interaction is applied to them.

Fig. 1. Screen image generated by Objed upon entry to the program.

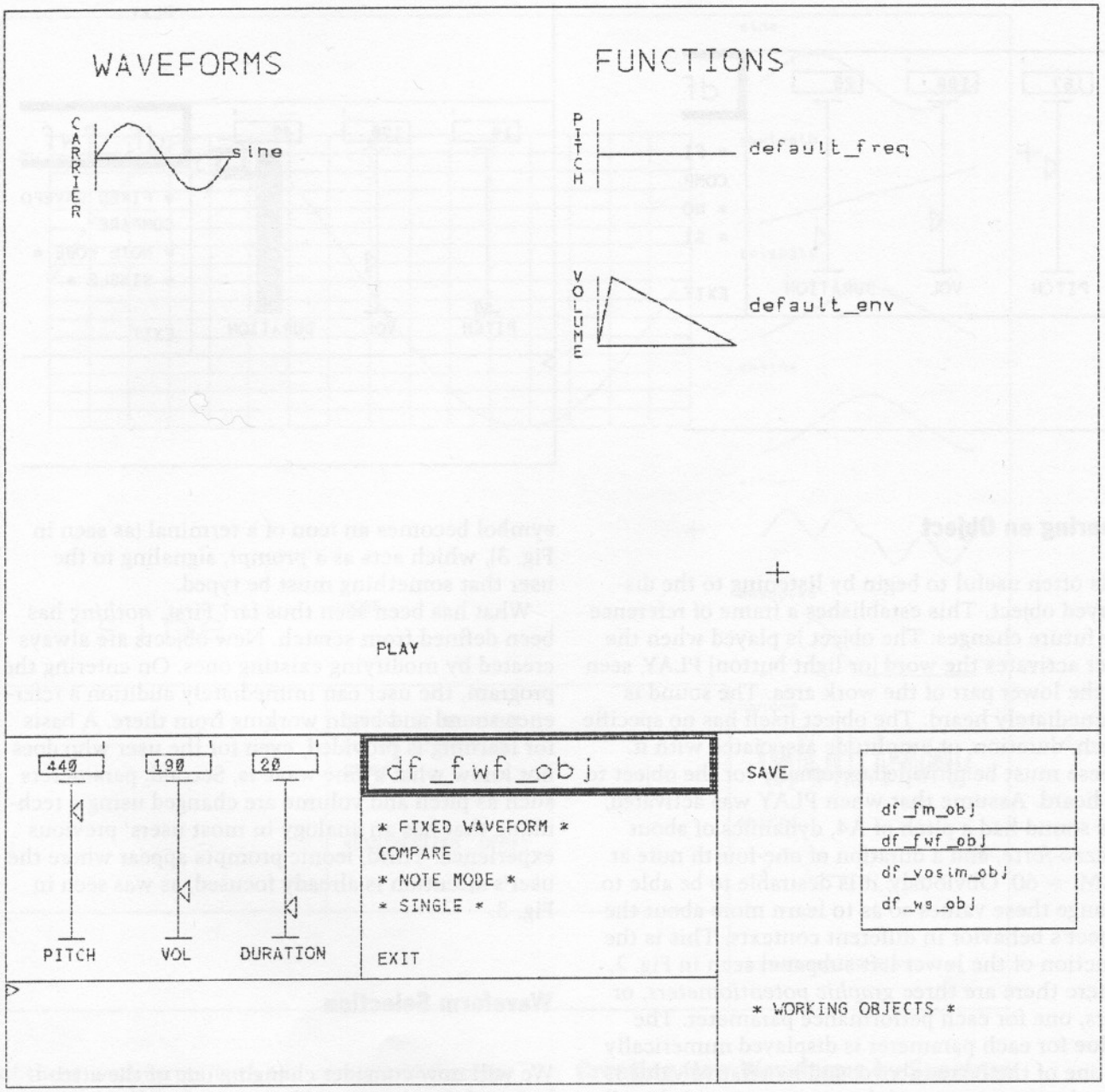


Fig. 2. Graphic potentiometers, used to control parameters of the sound.

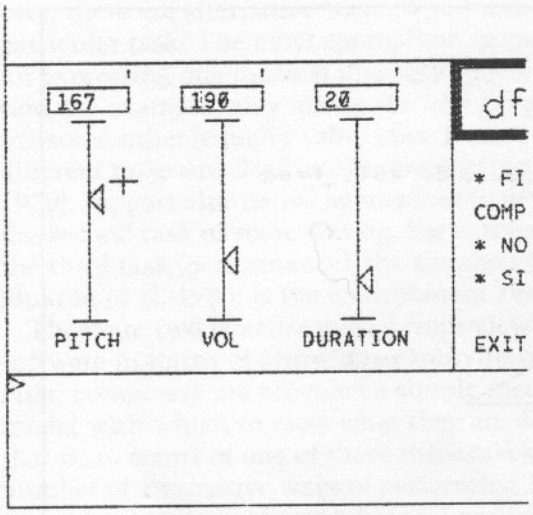
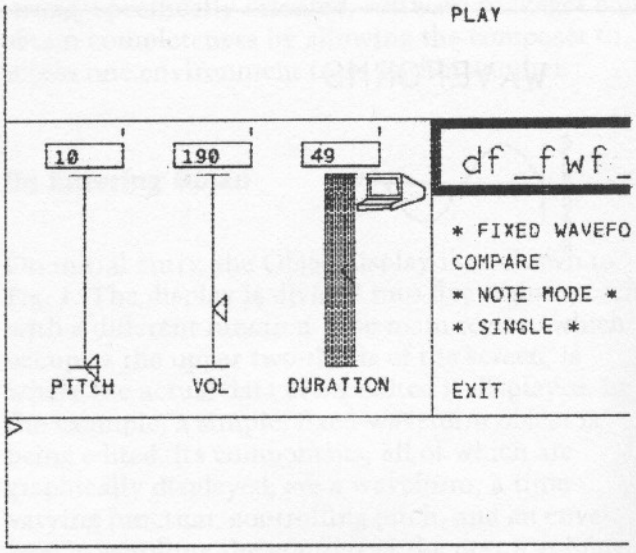


Fig. 3. Setting a level on a graphic potentiometer by typing. The tracking symbol has changed into the icon of a terminal.



Hearing an Object

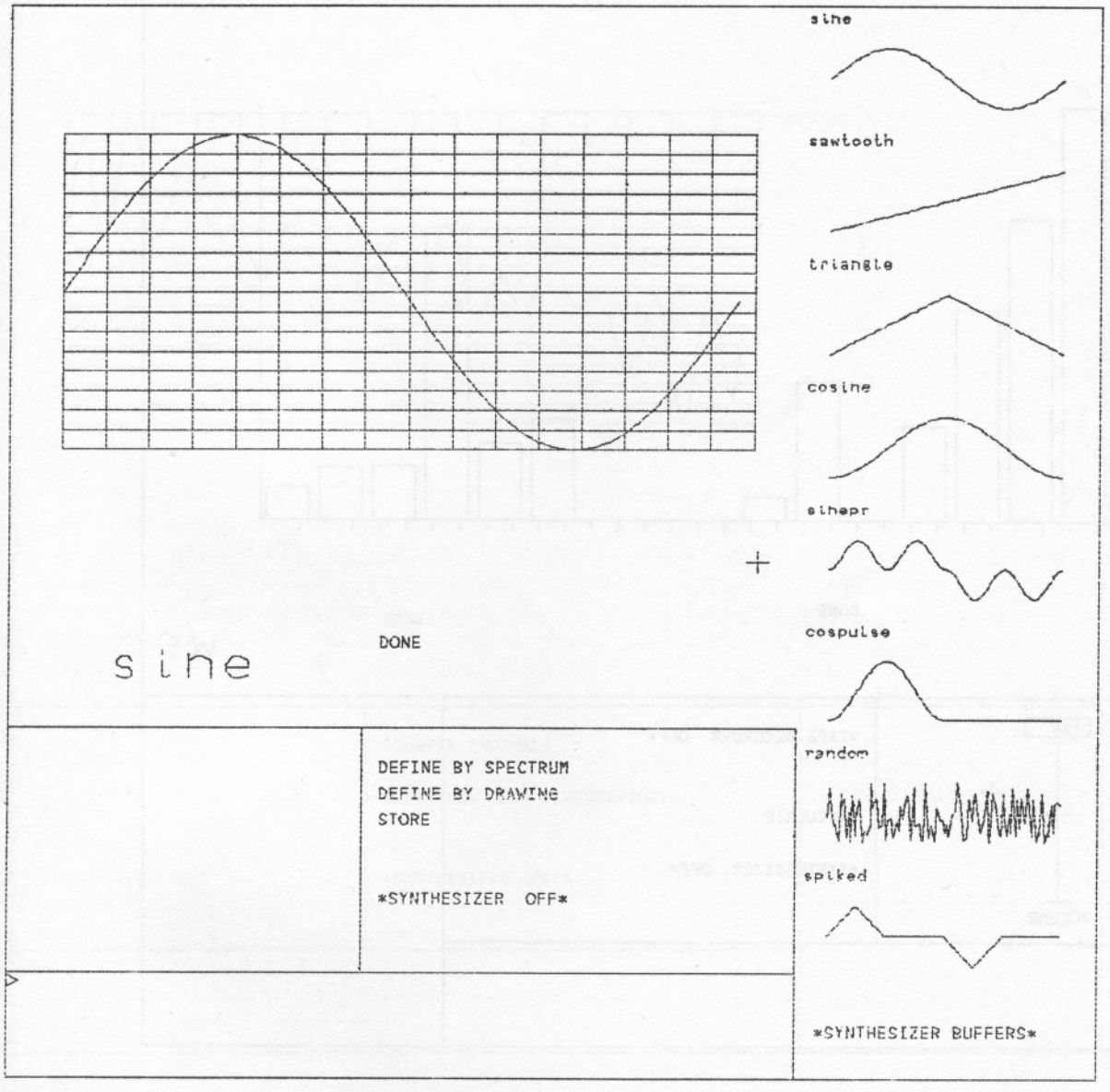
It is often useful to begin by listening to the displayed object. This establishes a frame of reference for future changes. The object is played when the user activates the word (or light button) PLAY, seen in the lower part of the work area. The sound is immediately heard. The object itself has no specific pitch, duration, or amplitude associated with it. These must be provided externally for the object to be heard. Assume that when PLAY was activated, the sound had a pitch of A4, dynamics of about *mezzo-forte*, and a duration of one-fourth note at M.M. = 60. Obviously, it is desirable to be able to change these values so as to learn more about the object's behavior in different contexts. This is the function of the lower-left subpanel seen in Fig. 2, where there are three *graphic potentiometers*, or *pots*, one for each performance parameter. The value for each parameter is displayed numerically in one of the boxes above, and its relative value is seen by the position of the potentiometer's *handle* (the triangular pointer). The user can change the value of any graphic potentiometer by "dragging" its handle up or down, using the cursor. Alternatively, the user may point at the box above the graphic potentiometer, activate the Z button, and type in a numerical value. In this case, the tracking symbol becomes an icon of a terminal (as seen in Fig. 3), which acts as a *prompt*, signaling to the user that something must be typed.

What has been seen thus far? First, *nothing* has been defined from scratch. New objects are always created by modifying existing ones. On entering the program, the user can immediately audition a reference sound and begin working from there. A basis for learning is provided, even for the user who does not know what a sine wave is. Second, parameters such as pitch and volume are changed using a technique that has an analogy in most users' previous experience. Third, iconic prompts appear where the user's attention is already focused, as was seen in Fig. 3.

Waveform Selection

We will now consider changing one of the attributes of the object itself. As an example, let us alter the waveform associated with the object. Pointing at the picture of the current waveform and depressing the Z button will cause the panel seen in Fig. 4 to appear. A menu consisting of the eight waveforms currently loaded in the synthesizer appears down the right margin of this panel. Selecting one

Fig. 4. Window for waveform selection.

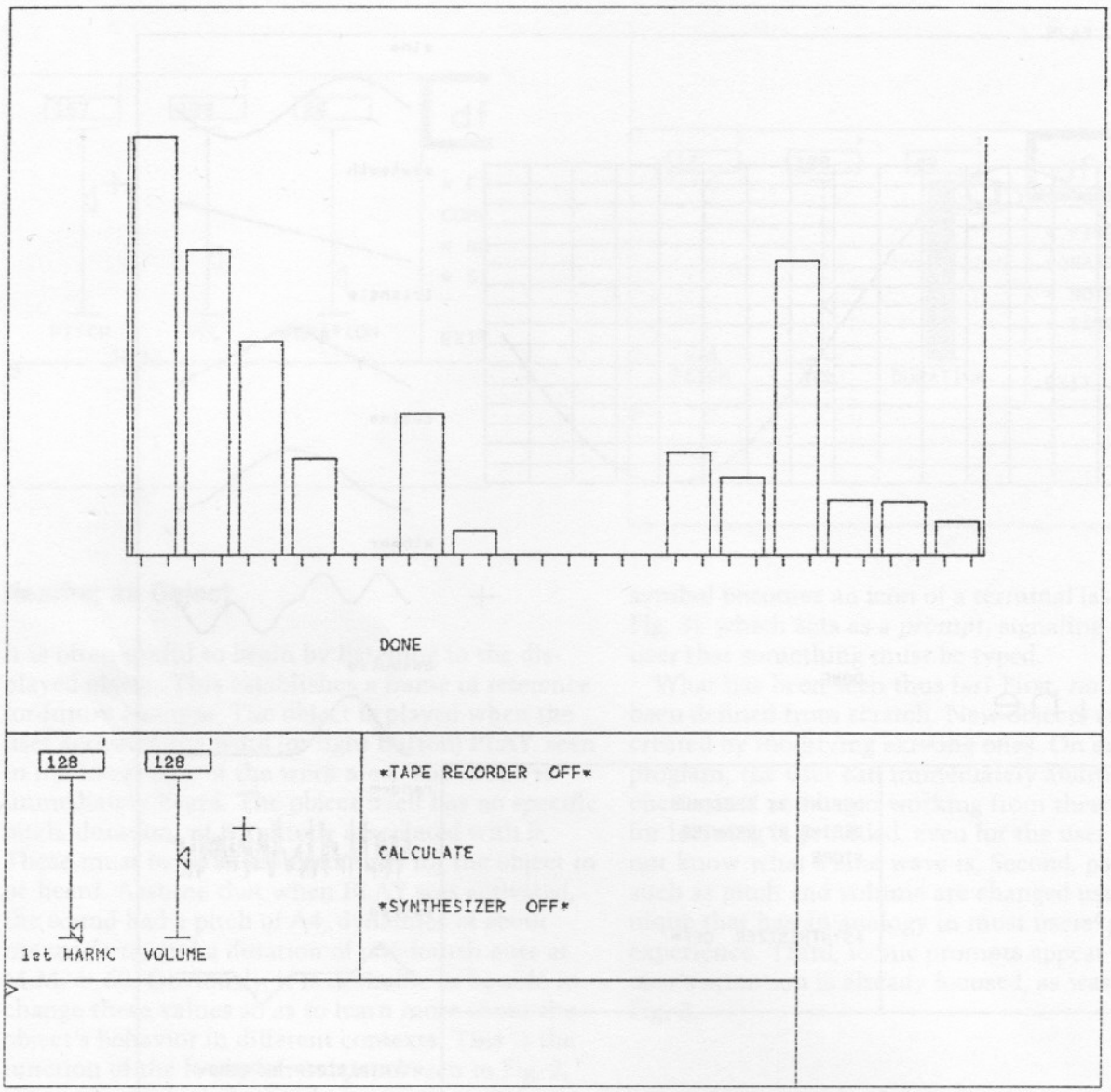


of these waveforms with the cursor and then activating the light button DONE restores the original panel of Fig. 1, with the one difference that a drawing of the selected waveform has replaced that of the sinusoid. This brings up an important point: the current state of the object being edited is always clearly displayed, thereby eliminating the mental task of remembering its current attributes.

Creating New Waveforms by Spectrum

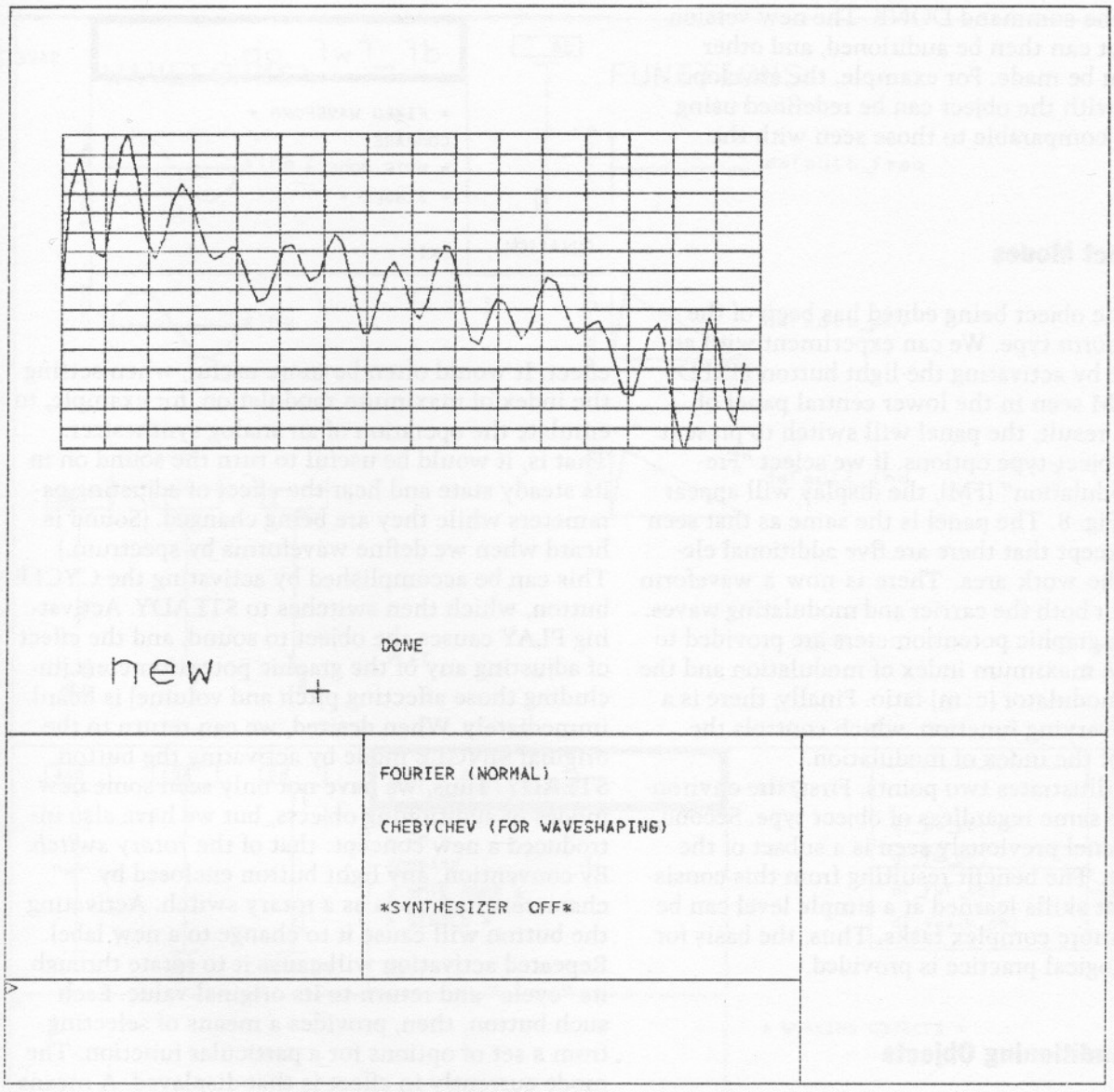
In the previous example (Fig. 4), if we had wanted to define a new waveform rather than use one of those in the synthesizer, we could have done so. By activating the light button DEFINE BY SPECTRUM in Fig. 4, we cause the panel shown in Fig. 5 to appear. What is seen in the work area is a bar graph in

Fig. 5. Defining a waveform by spectral content. Each bar represents a harmonic; its height is the amplitude of the harmonic relative to the others.



which the height of the bars represents the relative amplitude of the harmonics of a sound. Harmonics 1–16 appear from left to right. Any bar will jump to the height of the cursor when the cursor passes over the bar while the Z button is depressed. Thus, a simple hand gesture can sketch the spectral envelope of a waveform. More importantly, the waveform is being synthesized all the while, and we can use the graphic potentiometers in the lower left-hand panel to adjust the overall amplitude and the frequency of the fundamental. (Placement and use of the latter are consistent with the main panel, thereby simplifying learning of the control structure.) Throughout, the spectral content of the sound and the graphics display are updated in real time. The result is that musicians can quickly de-

Fig. 6. Example of a waveform defined by spectral content.



velop a sense of the perceptual effect of spectral content on steady-state tones.

Once the desired spectrum is defined, the composer activates the CALCULATE light button, and the new waveform is calculated and displayed (in the time domain), as seen in Fig. 6. Activating CALCULATE presents the user with the option of having the waveform calculated according to the weighed sum of the Fourier series or the Chebychev polynomials. In each case, the bar heights represent the relative weights of the different-order functions. The Chebychev polynomials are used to generate transfer functions to be employed in waveshaping synthesis.

The waveform is then loaded into the synthesizer, and the user returns to the main panel by

activating the command DONE. The new version of the object can then be auditioned, and other changes can be made. For example, the envelope associated with the object can be redefined using techniques comparable to those seen with the waveform.

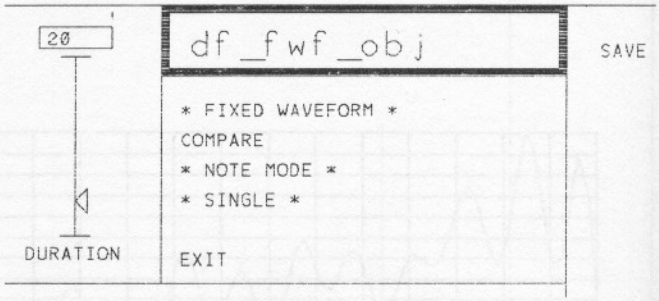


Fig. 7. Editing environment control subpanel.

Other Object Modes

Thus far, the object being edited has been of the *fixed-waveform* type. We can experiment with another mode by activating the light button FIXED WAVEFORM seen in the lower central panel of Fig. 7. As a result, the panel will switch to present the set of object-type options. If we select "Frequency Modulation" (FM), the display will appear as seen in Fig. 8. The panel is the same as that seen in Fig. 1, except that there are five additional elements in the work area. There is now a waveform displayed for both the carrier and modulating waves. In addition, graphic potentiometers are provided to set both the maximum index of modulation and the carrier-to-modulator (c : m) ratio. Finally, there is a third time-varying function, which controls the evolution of the index of modulation.

Figure 8 illustrates two points. First, the environment is the same regardless of object type. Second, the work panel previously seen is a subset of the current one. The benefit resulting from this consistency is that skills learned at a simple level can be applied to more complex tasks. Thus, the basis for good pedagogical practice is provided.

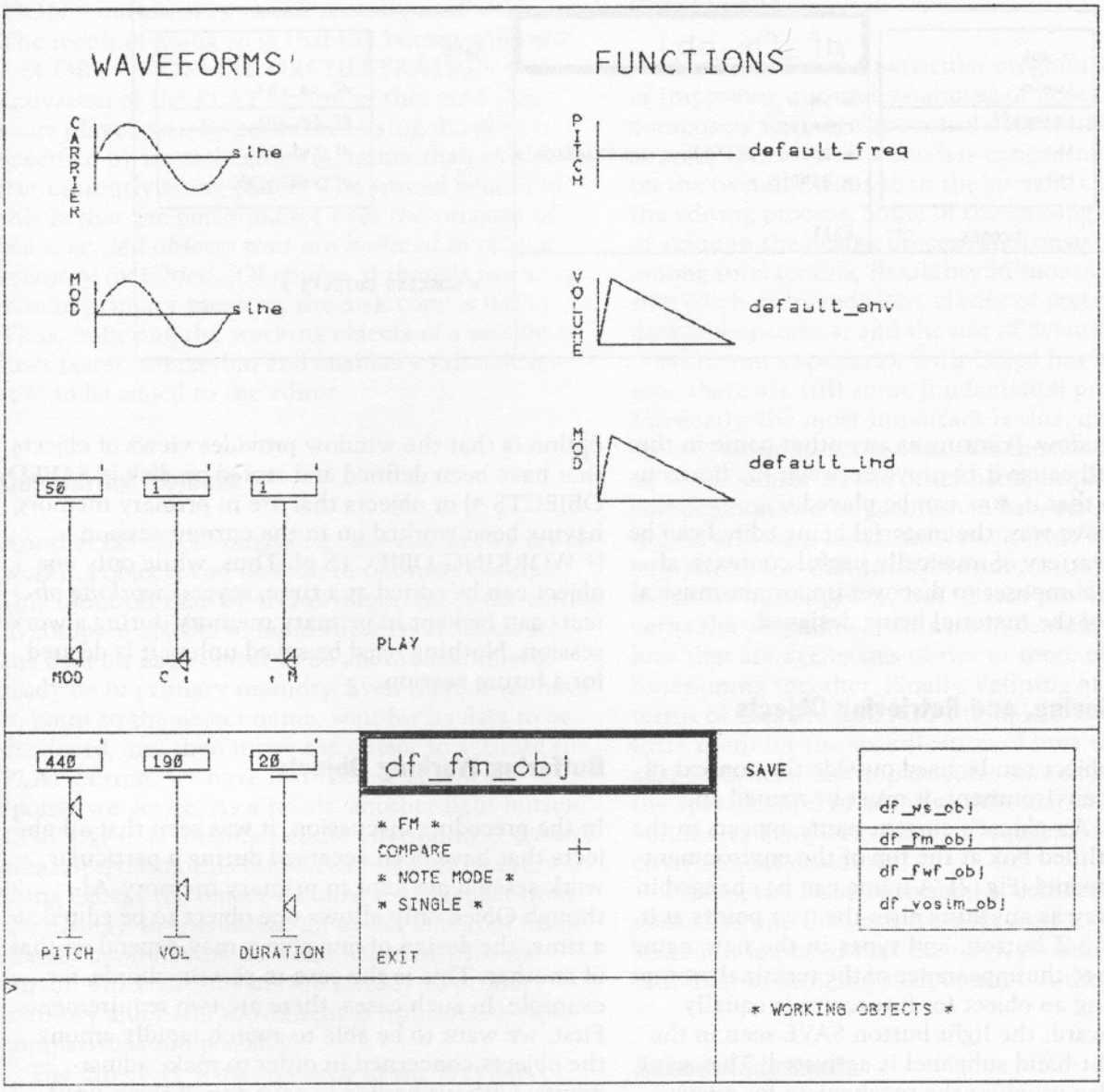
More on Auditioning Objects

During an editing session, it is often desirable to be able to audition the object repeatedly. One way to do this is to activate the SINGLE button at the bottom of the environment control panel (Fig. 7). The button will be renamed CYCLE, indicating that when PLAY is activated the sound will play repeatedly until stopped by the user.

One problem that still exists, however, is that all of our interactions with the object's data take the two-part form: (1) change a value, (2) listen to the effect. It would often be more useful, when setting the index of maximum modulation, for example, to emulate the operation of an analog synthesizer. That is, it would be useful to turn the sound on in its steady state and hear the effect of adjusting parameters while they are being changed. (Sound is heard when we define waveforms by spectrum.) This can be accomplished by activating the CYCLE button, which then switches to STEADY. Activating PLAY causes the object to sound, and the effect of adjusting any of the graphic potentiometers (including those affecting pitch and volume) is heard immediately. When desired, we can return to the original SINGLE mode by activating the button STEADY. Thus, we have not only seen some new modes of auditioning objects, but we have also introduced a new concept: that of the *rotary switch*. By convention, any light button enclosed by "*" characters functions as a rotary switch. Activating the button will cause it to change to a new label. Repeated activation will cause it to rotate through its "cycle" and return to its original value. Each such button, then, provides a means of selecting from a set of options for a particular function. The mode currently in effect is that displayed. A means is provided, therefore, of accessing significant complexity that can be "hidden" until the user is ready for it.

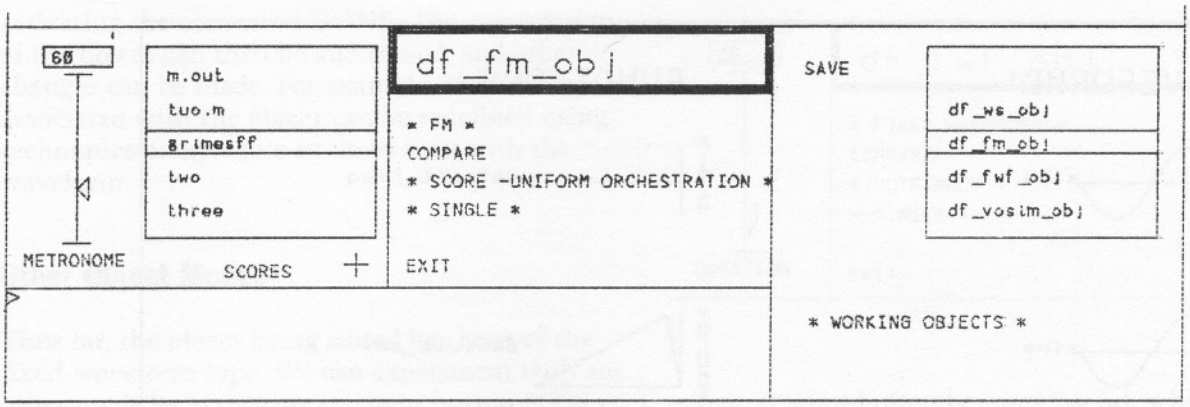
The effect of hearing an object in isolation is often (usually?) very different from that of hearing it in some musical context. So far, we have only been able to hear an object as a single note. This can be altered by activating the NOTE MODE button (Fig. 7). Once activated, the button is renamed SCORE – UNIFORM ORCHESTRATION, and the

Fig. 8. Work panel for editing an FM object.



subpanel controlling note parameters is switched, as seen in Fig. 9. The box that appears can be thought of as a window that looks out on all of the scores the composer has created. The name of each score is listed in this directory window, and if there are more names than will fit, a means of scrolling through them is provided. When the PLAY button is activated, the score whose name appears between the horizontal lines of the window is heard. More importantly for our purposes, the score is temporarily orchestrated with the object currently being edited! Furthermore, the metronome marking controlling the tempo of the performance can be changed by adjusting the graphic potentiometer be-

Fig. 9. Control panel for auditioning objects in the context of user-defined scores.



side the window. Pointing at any other name in the window will cause it to move between the horizontal lines so that it, too, can be played.

In the above way, the material being edited can be heard in a variety of musically useful contexts, allowing the composer to discover important musical properties of the material being designed.

Naming, Saving, and Retrieving Objects

Before an object can be used outside the context of the editing environment, it must be named and then saved. An object's current name appears in the heavily outlined box at the top of the environment-control subpanel (Fig. 7). A name can be changed in the same way as anything else: the user points at it, activates the Z button, and types in the new name in response to the appearance of the terminal prompt (icon). Saving an object for future use is equally straightforward: the light button SAVE seen in the bottom right-hand subpanel is activated. This same subpanel also provides the mechanism for retrieving previously defined objects. The technique is, again, use of the directory window. In this case, instead of previously defined scores, the window provides selective vision of objects. The name between horizontal lines is always the same as that of the object being edited. Selecting a different name causes that object to be displayed in the work area for purposes of editing or audition. One interesting option is that the window provides views of objects that have been defined and stored on disk (* SAVED OBJECTS *) or objects that are in primary memory, having been worked on in the current session (* WORKING OBJECTS *). Thus, while only one object can be edited at a time, several *working objects* can be kept in primary memory during a work session. Nothing need be saved unless it is desired for a future session.

Buffering Working Objects

In the preceding discussion, it was seen that all objects that have been accessed during a particular work session are kept in primary memory. Although Objed only allows one object to be edited at a time, the design of one object may depend on that of another. This is the case in certain chords, for example. In such cases, there are two requirements. First, we want to be able to switch rapidly among the objects concerned in order to make adjustments, without having to save any of them until they are in an acceptable state. Second, we want to be able to audition the chord with the current version of these objects.

Objed supports both of these features. The first is accomplished using the WORKING OBJECTS window and the buffering mechanism already described. The second feature is achieved by activating the * SCORE – UNIFORM ORCHESTRA-

TION * button seen in the central panel of Fig. 9. The result of doing so is that the button will read * SCORE – NORMAL ORCHESTRATION *. On activation of the PLAY button in this mode, the score played will be performed using the objects specified by its orchestration, rather than by the object currently being edited. The special benefit of this is that the performance *uses the versions of the specified objects that are buffered in primary memory by Objed.* (Of course, if there is not a version in primary memory, the disk copy is used.) Thus, buffering the working objects of a session allows faster interaction and enables a valuable feature to be added to the editor.

Comparing Objects

Another benefit of providing easy access to a set of working objects can be seen in our next example. One function that we feel is important is the ability to compare objects in rapid succession. To do so, the data for each object to be auditioned must already be in primary memory. Even then, if we have to point to the object name, wait for its data to be displayed, and then move the cursor to activate the PLAY button, we have lost the instantaneous response we desire. As a result, another light button, COMPARE, is provided in the environment control area. Activating this button causes almost everything except the object window to disappear from the display. In this mode, an object is played immediately upon having its name selected with the cursor. Since the objects are already in primary memory and hand movement is minimized, rapid comparisons are possible.

The directory windows seen in the previous examples are the most important feature of the system with respect to one of our initial demands: that the cognitive effort required for administration and bookkeeping be minimized. In computer science terms, this represents file input and ouput. The windows allow files to be input without the user having to type names, remember spelling, or extract the score or object files from among the text files and other items in the directory.

Conclusions

We have examined a particular program as a means of improving our understanding of how to provide composers with better control over timbral resources. The examination has concentrated more on the overall means than the specific content of the editing process. Some of the guiding principles of value in the design process are consistency among interactions, flexibility in modes of operation (such as in audition), clarity of presentation of data and operators, and the use of defaults.

While our experience with Objed has been positive, there are still some fundamental problems. Musically, the most important is this: designing a system around a score editor on the one hand, and an object editor on the other hand, implies that composition can be partitioned between the sonological and deep-structural levels. This is a large assumption; one that is clearly not justified in some music. A more specific but related problem concerns the program's weakness in dealing with timbres that are aggregates of two or more objects functioning together. Finally, defining objects in terms of data plugged into a template leaves very little room for the specification of how the object adapts its behavior in particular contexts. How can the specification of adaptive information (such as volume varying with pitch) be combined with the environment described?

Clearly, the issue of timbral specification and control is one that still requires a great deal of research. It is hoped that the work presented here will help in bringing such control to the composer.

Acknowledgments

The results reported in this paper have benefited from the input of numerous composers who have worked on the SSSP system, especially James Montgomery and Philippe Menard. Several conversations with David Wessel have also strongly influenced our approach. Finally, Bob Pritchard and M. R. Lamb have made many helpful comments during the preparation of this manuscript.

The research reported in this paper has been undertaken as part of the SSSP of the University of Toronto. This research is supported by the Social Sciences and Humanities Research Council of Canada. This support is gratefully acknowledged.

References

Arfib, D. 1979. "Digital Synthesis of Complex Spectra by Means of Multiplication of Non-linear Distorted Sine Waves." *Journal of the Audio Engineering Society* 27(10):757–768.

Buxton, W. 1981. "Music Software User's Manual." Tech. Note No. 22. Toronto: University of Toronto Computer Systems Research Group.

Buxton, W. et al. 1979. "The Evolution of the SSSP Score Editing Tools." *Computer Music Journal* 3(4):14–25.

Buxton, W. et al. 1980. "A Microcomputer-based Conducting System." *Computer Music Journal* 4(1):8–21.

Chowning, J. 1973. "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation." *Journal of the Audio Engineering Society* 21:526–534. Reprinted in *Computer Music Journal* 1(2):46–54.

Grey, J. 1975. "Exploration of Musical Timbre." Tech. Rep. STAN-M-2. Stanford, California: Stanford University Department of Music.

Grey, J. 1977. "Multidimensional Perceptual Scaling of Musical Timbre." *Journal of the Acoustical Society of America* 61:1270–1277.

Kaegi, W., and S. Tempelaars. 1978. "VOSIM—A New Sound Synthesis System." *Journal of the Audio Engineering Society* 26:418–424.

Krasner, G. 1980. "The Design of a Smalltalk Music System." *Computer Music Journal* 4(4):4–14.

Le Brun, M. 1979. "Digital Waveshaping Synthesis." *Journal of the Audio Engineering Society* 27:250–266.

Mathews, M. V. 1969. *The Technology of Computer Music*. Cambridge, Massachusetts: MIT Press.

Moorer, J. A. 1977. "Signal Processing Aspects of Computer Music—A Survey." *Proceedings of the IEEE* 65:1108–1137. Reprinted in *Computer Music Journal* 1(1):4–37.

Truax, B. 1976. "A Communicational Approach to Computer Sound Programs." *Journal of Music Theory* 20(2):227–300.

Wessel, D. 1979. "Timbre Space as a Musical Control Structure." *Computer Music Journal* 3(2):45–52.