Lightweight Classes Without Programming

Carla Sealeti
Symbolic Sound Corporation • P. O. Box 2530 • Champaign, IL 61825-2530 • USA
Tel: (217) 328-6645 Electronic Mail: symbolic.snd@applelink.apple.com

In an earlier version of Kyma, creating new Sound classes required an extensive knowledge of the Smalltalk-80 programming language. The addition of variables to Kyma makes it possible to define new classes of Sound without programming and to construct sound objects algorithmically using a score language.

A class can be thought of as a set of functions (methods) with formal parameters (instance variables). An instance of that class is an association of the formal parameters with specific values. A sound object with one or more variable arguments can be thought of as a class of sound objects, and the variables can be thought of as the formal parameters of the class. In the current version of Kyma, a sound object's arguments can be constants, variables, or expressions involving constants and variables. If a variable appears in more than one argument, it sets up a constraint among those arguments. The formal parameters of a user-defined subclass of Sound can be bound to specific values in three ways: interactively (through dialog boxes); through the use of a user-defined, graphic editor for the newly defined sound class; or through a Music-N style language in which lightweight classes serve as instruments, and events in the score instantiate those classes, binding specific values to their formal parameters.

Introduction

Kyma is a visual language for specifying and manipulating sound [1-4]. One of the premises of Kyma is that a composer's language must be customizable and extendable. Another is that composers should be able to learn about a system as they are using it to compose; they should not have to master the entire language before making the first sound. In an early version of Kyma, complex sounds could be designed in a visual language, but extensions to Kyma required a considerable amount of programming in Smalltalk-80 [2-5].

The goals of this work were: one, to make it possible for composers to extend the Kyma language without programming, and two, to add an algorithmic score language object to Kyma.

Class and Instance

A class of objects is like a category or generalization; it describes what several instances of that class have in common. A class describes the structure and behavior of its instances.

In the Smalltalk-80 language, a class is a set of functions (methods) with formal parameters (instance variables). An instance of a class is an association of the formal parameters with specific values. A new instance is created by sending an instance creation message to the class. Classes can be defined hierarchically, and they inherit characteristics of their super classes [6].

Each Kyma class is a subclass of the class Sound. All Sound subclasses share a common set of characteristics:

- A name
- An icon
- A help string describing the class
- An array of instance parameter names and types and some common behavior; for example, each Sound subclass understands the play message.

Since these are the only elements essential to a Sound subclass, it is not necessary to create a full-fledged Smalltalk class in order to create a new subclass of Sound. "Lightweight" classes — objects containing only those characteristics necessary for Sound subclasses — could serve just as well.

In the visual Kyma language, new instances of Sound subclasses are created by cloning existing instances rather than by sending a creation message to the class. The Prototype Strip is a palette containing an instance of each Sound subclass. A new instance is cloned by dragging one of the prototype instances into the working window and then modifying it as desired.

Lightweight Sound Classes

A Kyma sound object represents a stream of samples; a sound object is either a source of samples or a function of one or more sample streams. In some respects, sound objects are analogous to arithmetic expressions. Unlike arithmetic expressions, how-
ever, sound objects can also contain time offsets and indications of simultaneity or sequence. Sound objects are used to define all levels of a composition, from the microstructure of timbre, to the macrostructure of an entire composition.

Lifting

When an arithmetic expression has one or more formal parameters represented by variables, it represents an entire class of expressions. Similarly, a sound object in which one or more parameters are variable represents an entire class of sound objects, not just one specific sound object. It is as if the specific ground instance has been "lifted" to a higher level of abstraction.

A lifted sound represents a class of sound objects. Since a sound object represents anything from a cymbal to a composition, a lifted sound may represent a class of timbres, a class of structures, or even a class of compositions.

Variables

Any parameter of a sound object can be a variable or an expression involving variables. When messages are sent to variables, they are saved and evaluated later, when the value of the variable is known.

Variables representing typeable parameters are entered as strings preceded by question marks (e.g., Figure 1); variables representing sound objects or collections of sound objects are represented by icons that can be dragged into a sound structure (as in Figure 2).

Each occurrence of a variable within a sound represents the same object; it is true whether the variable appears more than once at the same level or whether it occurs elsewhere in one of the subsounds. Constraints between the parameters of a sound object can be specified as expressions involving variables. As a simple example, Figure 2 shows a sound object in which the RightScale is constrained to be 1.0 minus the LeftScale. Parameters can be related to each other through any Smalltalk expression or sequence of expressions.

While examples shown in this paper are, of necessity, relatively simple, one of the biggest advantages of using variables and variable expressions is in encapsulating and generalizing complex interrelationships between sound objects and parameters.

Figure 1. Simple Parameter Variables. In this sound object, the frequency and duration have been set to the variables Tone and Time respectively. The left half of this editor shows the structure of the sound object, and the right half shows the parameters of the gray-highlighted subsound.

Installation

How can specific values be attached to the formal parameter of a lifted Sound? A mapping from variable names to values can be used to substitute actual values for the variable parameters. Such a mapping is usually referred to as a binding list, since it binds variables to values [7]. The binding list can be constructed by querying the user, by constructing a new lightweight class, or by interpreting a "scores" program.

User-supplied Values

Before a lifted sound object can be played, it must have specific values for each of its parameters. If

Figure 2. A Simple Example of Constraints Between Parameters. The RightScale is constrained to be 1.0 minus the LeftScale. Notice that a Variable=Sound represents a sound object rather than a typeable parameter value.
Figure 3. A Binding List. This is one mapping of variable names to values for the sound object shown in Figure 2.

the user sets to play a sound object that has free variables in it, dialog boxes request values for each of the free variables, and a substitution mapping is constructed (as, for example, the mapping shown in Figure 3).

Class editor
All sound classes share a common set of characteristics and behavior, making it possible to automate the creation of new sound classes.

The Kyma Class Editor (Figure 4) uses a lifted sound as the basis for a new class description. The sound object’s free variables become the parameter names of the new class. Each parameter name is assigned a type, a default value and a description (that serves as on-line help). A name, an icon, and a class description complete the definition of the new class.

An instance of the new class consists of a mapping of parameter names to values and a pointer to the lifted sound that serves as the class of the object.

Sound objects defined in this way are indistinguishable from the sound objects provided with the Kyma System.

Score language
Another interpretation of a lifted sound object is as an "instrument" in the sense of the Music N family of languages. In the Music N languages [8], the model is that of an orchestra—a setup made up of calls to subroutines or functions—and a score—a file containing values to the orchestra program.

A Kyma Score-Language sound differs from a traditional Music N language in three ways. First, the instrument/score model is not central to Kyma; it is just one of the many compositional paradigms supported by Kyma. An instance of Score-Language can be treated like any other sound object in Kyma; it can even be used as an "instrument" in another instance of Score-Language. Second, an "instrument" can be altered from within the score. Third, a Kyma score is not a data file but a Smalltalk-80 program. Events can be generated algorithmically, directly in the score.

A Kyma Score-Language sound uses its subsounds as "instruments" and instantiates them from a "score" which is a Smalltalk-80 program.

An "event" in the score is a Smalltalk message sent to the lifted sound. The message provides the lifted sound with a mapping of variable names to values. Each event also supplies a start time for that particular instance of the lifted sound.

For example, the event:

```
setStereoPlacementSound
start: 0 seconds
left: 0.25
variableSound: (gran dur: 2 seconds freq: 2 a pitch)
```

would schedule an instance of "setStereoPlacementSound" at time 0 and with the following mapping of variable names to values:

```
left => 0.25
variableSound => gran
```

Notice that one of the arguments to "setStereoPlacementSound" is another sound object; "gran"; the lifted sound "gran" is also instantiated, in this case with the mapping

```
gran => 2 seconds
freq => 2 a pitch
```

"gran", however, is not scheduled at a specific time since it derives its start time from "setStereoPlacementSound". The variables in "gran" are bound first, before the substitutions of "setStereoPlace-

Figure 4. The Class Editor. This is an editor for constructing a class based on the lifted sound of Figure 2. The newly defined class has its own name, icon, list of parameters, and instance editor (painted in the Field Layout pane).
A variant of the ScoreLanguage, called the FileInter-preter, can be used to interpret or present data files stored on the disk.

Applications

Using a ScoreLanguage or FileInterpreter, composers can generate sound objects either event-by-event or algorithmically using Smalltalk-80 [9, 10].

The class definition facility is used in the Kyma development system to integrate new assembly language primitives into the graphical environment [11]. Class definition has also been used to create application specific tools, for example, tools for mapping scientific data into sound [12].

Summary

By seeing some parameters to variables, one can generalize and encapsulate complex sound objects in Kyma. Expressions involving variables can be used to define constraints between the parameters of a sound object. A sound object with variable parameters represents an entire class of sounds, and it can be used as the basis for a user-defined class or as an "instrument" in a ScoreLanguage or FileInterpreter environment.

References


ICMC 508