

Composition By Refinement



New Representations for Flexible Music Description Languages

Stephen Travis Pope
ParcPlace Systems, Inc.
1550 Plymouth St.
Mountain View, California, 94043 USA
stp@ParcPlace.{com, uucp}

Introduction



What is *Composition by Refinement*?

- A System for Simple Description of middle-level musical structures
- A Method of defining a musical structure in terms of EventGenerators
- A Collection of EventGenerator Objects with interesting default behavior
- A System which supports *Rapid Prototyping and Incremental Refinement* of Music

Background



The *HyperScore Toolkit* (~1986-88)

A Set of Objects for Music Description
Events/EventLists
EventGenerators/EventModifiers

DoubleTalk (~1984-86)

A Composer's Workbench
Music Structure Notations
Capturing the Process of Composition

Outline



- Introduction/Motivation/Background
- Music Description in the HyperScore ToolKit
- Specification of Generic EventGenerators
- Refinement of EventGenerators
- Examples

Motivation



Why Is He Doing This?

- Design of Flexible User Interfaces for the process of composition
- Exploration into Music Description Languages for new musical structures
- Building Representations of the *Composition Process*
- For use in his own compositions

The HyperScore ToolKit



Music Representation via Events and EventLists

- Events are Property Lists with Durations
- EventLists are Keyed lists of Events where the Keys are relative Start Times
- EventLists are Events (i.e., they can be hierarchical)
- EventLists are Smalltalk-80 Collections (i.e., one can send messages to access their events or operate on them)

Event and EventList Creation



Create a Note

```
"declaration of a NoteEvent"
NoteEvent duration: 1/8
      pitch: 'c3'
      amplitude: 'ff'
      accent: 'sfz'
```

Create an EventList

```
"create and play a simple named event list"
(EventList named: #ExampleMelody)
  add: (NoteEvent duration: 1/8 pitch: 'c3');
  add: (NoteEvent duration: 1/8 pitch: 'd3');
  add: (NoteEvent duration: 1/8 pitch: 'e3').

(EventList named: #ExampleMelody) play
```

Creating EventGenerators



• Message to a Class with creation parameters

```
Chord majorTriadOn: 'c3' inversion: '1'

Ostinato onEventList: someEventList

(Cluster from: 'c4' to: 'c5') eventList
```

• It is Possible to Send Messages to the EventGenerator or to its Events

```
((Cluster from: 'c4' to: 'c5') eventList)
  removePitch: 'g4'
```

Chord Creation Examples



Create some Chords

```
"Simple method, leads to very big Chord class."
Chord majorTriadOn: 'c3' inversion: '1'

"(possibly many) mode-specific Chord classes."
MajorChord tetradOn: 'c3' inversion: '1'

"Specify Chord Intervals."
Chord intervals: #(7 12 16)

"Specify Chord Events (useless)"
Chord events: someEventList
```

(There are many other possibilities.)

EventGenerators/Modifiers



There are Types of Events that can Create or Modify other Events.

EventGenerator examples

```
Chord
Ostinato
MarkhovTransitionTable
```

EventModifier examples

```
Rubato
Crescendo
```

Types of EventGenerators



Generic and Specific Behaviors

Example: the Hierarchy of Chords

```
Chord
  MajorChord
    ... major chord subclasses ...
  MinorChord
    ... minor chord subclasses ...
```

The Design Tension

One class with very many creation messages versus many classes with one message each

Cluster Creation Examples



Create a Whole-tone Cluster within an Octave

```
"create a cluster with a range of pitches"
aCluster Cluster from: 'c4' to: 'a5'

"new cluster creation message"
aCluster Cluster evenPitchesFrom: 'c4' to: 'a5'

"select the correct events with a block"
aCluster events: aCluster select:
  [ :anEvent |
    anEvent pitch asMidi isEven].
```

Cluster Manipulation



- Using Smalltalk-80 Collection Messages

aCluster collect: [aBlock]
 aCluster select: [aBlock]
 aCluster detect: [aBlock]
 aCluster reject: [aBlock]

- Using EventList Messages

Set Events, set Event parameters
 Collection operations on Events

- Using Graphical Editors

Example (cont'd)



Step 3—Split Parts, Add Performance Markings and Functional Markings

D: I VI I VII⁶ I V/V V

User Interfaces for Structures



- Event Editors/Inspectors

Event Editor

2/1 ff
 1/1 f
 1/2 mf
 1/4 mp
 1/8 p
 1/16 pp
 1/32

468 64

duration
 props
 voice
 pitch
 loudness

Evt Dur: 1/4 Beat
 Pitch: 'c#4'
 Amp: 64
 Voice: 'violin'
 Technique: 'con legno'

R.P and I.R. Example



Rapid Prototyping and Incremental Refinement of a Bach Chorale (Ein' feste Burg)

Step 1—Use Default Voicing for Chords

Step 2—Correct Voice Leading

Advantages of the Technique



- Rapid Prototyping provides audible results very early in the process of composition
- The Process of Refinement is Captured in the script of messages sent to EventGenerators in the process of refinement
- The EventGenerator Hierarchy grows into a personal library of musical structures

Structure U.I.s (cont'd)



- EventList Dictionaries/Browsers

Event List Dictionary

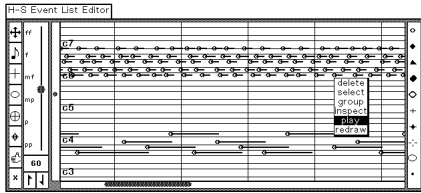
Bach
 BarbaraAllen
 Bournee
 Bransleashort
 demo1
 demo2
 Fugue3
 Gallarde
 is2
 MapleLeaf
 MapleLeaf2
 Merlions
 peal1
 peal1b
 peal2
 peal3
 peal3b
 peal3bshort
 progression1
 random

1 [EventList 'demo2' dur:5900 events: 3
 0 -> (EventList 'demo1' dur: 3400 events: 4
 0 -> Evt Dur: 400Ms. Pch: key 36 Amp: vel: 100
 1000 -> Evt Dur: 400Ms. Pch: key 40 Amp: vel: 100
 2000 -> Evt Dur: 400Ms. Pch: key 43 Amp: vel: 100
 3000 -> (EventList 'temp.1' dur: 400 events: 3
 0 -> Evt Dur: 400Ms. Pch: key 36 Amp: vel: 100
 0 -> Evt Dur: 400Ms. Pch: key 40 Amp: vel: 100
 0 -> Evt Dur: 400Ms. Pch: key 43 Amp: vel: 100
)
 1300 -> (EventList 'demo1' dur: 3400 events: 4
 0 -> Evt Dur: 400Ms. Pch: key 36 Amp: vel: 100
 1000 -> Evt Dur: 400Ms. Pch: key 40 Amp: vel: 100
 2000 -> Evt Dur: 400Ms. Pch: key 43 Amp: vel: 100
 3000 -> (EventList 'temp.1' dur: 400 events: 3

Structure U.I.s (cont'd)



- EventList Editors

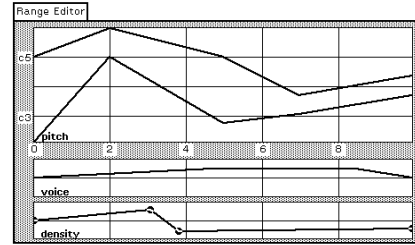


(Mention tools for building new UI
Components and editors)

Structure U.I.s (cont'd)



- Interfaces to Stochastic EventGenerators
(e.g., RangeEditors)



Conclusions



- The Framework includes simple Music Description Facilities as well as Objects for Representing Middle-level Musical Structures.
- It supports the Technique of Rapid Prototyping and Incremental Refinement.
- Classes for Describing Structures and Tools for Manipulating them are both Easily Extensible.