FAT-PROC
An Interactive, Pattern-Process, Algorithmic
Composition Program

Phil Winsor
Center for Experimental Music and Intermedia
University of North Texas
Denton, Texas 76203 (FAI2@UNTVAX.BITNET)
and
Computer Music Studio
National Chiao Tung University
Hsin-Chu, Taiwan, R.O.C. (PWINSOR@TWNCTU01.BITNET)

ABSTRACT

Fat-Proc, in use at the University of North Texas Center for Experimental Music and Intermedia since 1985, is an interactive pattern/process oriented program originally designed to produce ensemble scores using "minimalist" techniques. Over the years, it has grown into a general compositional utility program which allows wide stylistic latitude. It consists of two parts, a pitch gamut generation routine and a melody generation routine.

INTRODUCTION

Fat-Proc is an interactive pattern/process oriented program originally written in BASIC and subsequently translated into C. At its inception, the program was designed to output scores (1 to 20 voices) using compositional devices commonly found in so-called minimal music: the application of conceptual, operational, and physical loop processes to user-entered or algorithmically composed melodic material. As the program evolved, however, it became clear that the constituent routines could be used to produce stylistically diverse compositions by allowing maximum flexibility with regard to input data and internal parameter controls.

Presently, the program has grown large enough to require interactive execution in two parts: Part 1, which generates and stores a chordal, scalar, or synthetic pitch gamut file for later retrieval by the score compilation routines; and Part 2, which derives, embellishes, loop-processes, and coordinates the melodies which constitute the final score. In broad terms, Pat-Proa now functions as a general compositional utility program with a pattern-processing bias; that is, features have been added which allow the user wide latitude in terms of musical style and the transformation of motivic material.
PROGRAM FUNCTION

Part 1 of the program offers three methods of generating a background chord/scale/pitch gamut file:

1. manual entry of a series of articulated aggregates that can be retrieved individually;

2. application by the computer of a constrained interval-size sieve to generate pitch gamuts;

3. octave-modular or extended pitch scales built (low to high range over the total chromatic) from a user-input interval set (Figure 1.)

The program user entered an interval set (5,4,7,3,6), start pitch, start octave, and number of tones in each gamut:

Thus, an entire set of registraed background materials for melody derivation and elaboration in Part 2 may be generated in a single execution of Part 1. Naturally, such usage requires a degree of planning prior to the interactive computer session; in practice, some composers choose to build their source file by concatenating shorter program runs or by alternating runs of Parts 1 and 2.

Part 2 of the program retrieves the chords/scales/gamuts filed in Part 1, then prompts the user to enter the number of coordinated score voices and the preferred method of assigning part pitch ranges (which may be stratified, interlocking, or common to all voices). At this juncture the program invokes the melody writing routine once for each constituent voice and once for each component chord/scale/gamut of the piece. During execution of the routine the user enters formative specifications chosen from the following procedures:

* method of melody derivation from chordfile: serial, random-order distribution of available pitches, or minimum/maximum interval choice sieve applied to pitches in the current chord/gamut;
Prime

Ornamented: 50% of notes, 50% each of types 3 & 4

Serial order gamut element selection
Figure 2.

Pitch sequence generated using interval-controlled gamut element selection:
Interval-size minimum 3, maximum 5 (in 1/2 steps)

Prime

Ornamented: 25% each, types 1-4

Intervalically controlled gamut element selection
Figure 3.

* occurrence probability weights for rests, number of notes to be embellished and type of embellishment to be selected from four available ornaments; (See Figures 2. & 3.)

* rhythmic complexity level (1-4) ranging from uniform pulse to multiple beat-subdivision (micropulse regrouping) assigned to the derived pitch sequence; (Figure 4.)

Rhythm patterns for 2 co-terminating melodies, each having its own internal meter (median):

median: 4, complexity level 4 (increased pulse regrouping)

median: 7, complexity level 8 (near maximum pulse subdivisions regrouping)

Complex rhythm generation mode
Figure 4.

* type of rhythmic loop processing to be applied (there are three choices: 1) accretions to a melodic nucleus; 2) progressive subtractions from the total sequence; and, 3) constant-length, metrically justified (among all the ensemble voices), element rotation; (Figure 5.)
Original melody = 8 notes/rests; program user has requested retrograde-order rotation of 2 elements over 16 copies;

Parameter-element rotation

\[ \text{Figure 5.} \]

* specific parameters to be affected by the selected loop process: pitch, rhythm, articulation, volume and/or timbre;

* number of iterations over which the selected loop process is to take place. (See Figure 5.)

PROGRAM OUTPUT FORMAT

After all orchestral parts have been composed, Pat-Proc compiles an alphanumeric notelist score in Script Music Language, which may be easily converted to a MIDI sequence, performed as is by a Synclavier digital synthesizer, or read into standard notation software for printout and performance by acoustic instruments.

PREVIOUS COMPOSITIONAL USES

The author has composed several pieces using Pat-Proc, including "Dulcimer Dream" for amplified piano, "Dervish #2" for Synclavier, "Les Chemins des Suages" for dancer, projections, and Synclavier, and "Suite for Disklavier". A number of graduate students have also produced stylistically diverse compositions using the program.