CENTER FOR COMPUTER RESEARCH IN MUSIC AND ACOUSTICS
OCTOBER 1991

Department of Music
Report No. STAN-M-76

LECTOR: AN ECCLESIASTICAL LATIN CONTROL LANGUAGE FOR THE SPASM/SINGER INSTRUMENT

Perry R. Cook

CCRMA
DEPARTMENT OF MUSIC
Stanford University
Stanford, California 94305

LECTOR: An Ecclesiastical Latin Control Language for the SPASM/singer Instrument

by Perry R. Cook, Stanford CCRMA
PRC@CCRMA.STANFORD.EDU

A text reading program has been constructed which generates control files for a physical model of the human singing voice. The rule system for Ecclesiastical Latin determines pronunciation. In one processing paradigm, the text is processed in stream fashion, yielding a text file which controls the singer instrument. Note names or frequencies are imbedded directly in the text using parenthesis. Similar to common vocal notation practice, durations can be extended with dashes. Enclosing a message in curly braces places it verbatim into the singer instrument control stream. A more graphical control system uses music notation of a monophonic melody line and text below. The text/melody alignment process is aided by graphical cursors and faint lines connecting text with notes. The graphical editor generates LECTOR text files, which in turn generate singer control text files. This allows control representations to be edited and manipulated at many levels.

LECTOR: Ecclesiastical Latin to Speech/Singing

The purpose of the LECTOR project is to provide simple means of data input for the SPASM/singer articulatory-controlled voice synthesis systems. The SPASM and singer projects are described in [1][2][3]. These singing/speech synthesis systems are intended for use both as research and compositional tools. In order for the systems to be useable by composers, singers, and other non-technical users, it was necessary to create methods of data input which closely resemble the representations of spoken words and singing that are familiar to us all. Such representations include simple typed words in ASCII text, as well as note names and frequency values, and common music notation with text placed below the musical staff as it is in vocal music.

The rules for Ecclesiastical Latin are simple and rigid [4]. Very few 'exceptions' exist, and the context required to determine the pronunciation of a single character depends on at most two characters before or after the current character. This allows the parser to run linearly, processing the text input as a stream. The preservation of the stream paradigm is considered important in this project, anticipating future systems which might pass the low-bandwidth text/control stream over a network or similar low-data-rate channel, for rendering of spoken or sung text at the other end. Another hardware paradigm for such a processing scheme is the low-bandwidth connection between a host processor or controller and a DSP or other music synthesis engine, as would be the case in a MIDI control scheme. The LECTOR input code, and the corresponding singer control code generated by the LECTOR Latin parser for the word *Requiem* is shown below. Note the rolled r subroutine, and the variables *vowelTimeRef* and *consTimeRef*, which allow time scaling of vowel and consonant times independently.

Speech produced using this type of control strategy is monotone, but quite natural in pronunciation because of the nature of the model used for synthesis. The control file is a time-ordered list of vocal tract shapes, glottal input files, pitches, amplitudes, etc.. The voice synthesizer drives smoothly from one shape and set of parameters to the next, according to the script. Interpolation in the shape space provides articulation effects which sound acoustically natural [1].

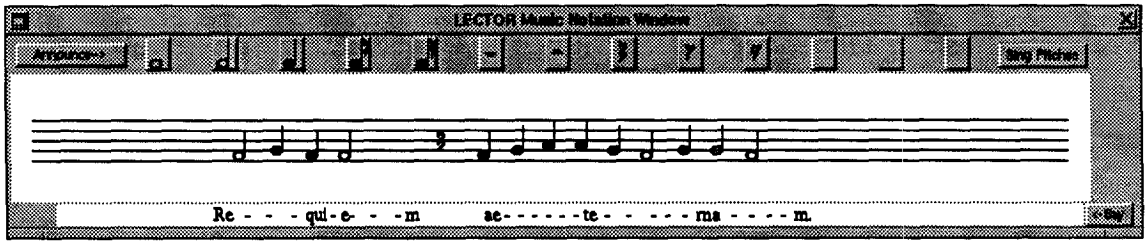
```
/********************** LECTOR Text Control File ******************************/
    Requiem
/********************** End LECTOR Text ***************************************/

/****************** Singer Control Code ***********************************************/
#include "singlib/singer.h"
#include "tuning/tuning.h"
singer(int fd)
{
    double vowlTimeRef=0.5,consTimeRef=0.5;
    setGlot("test");
    init();
    setup("rolledrc",lastGlot(),360.0,0.0,0.0,0.02);
    synthesize(0.10*consTimeRef,"rolledrc",lastGlot(),lastPitch(),0.00,0.00,lastVibrAmt(),fd);
    rollr(0.2*consTimeRef,lastGlot(),lastPitch(),0.5,lastVibrAmt(),fd);
    synthesize(0.05*vowlTimeRef,"ehh",lastGlot(),lastPitch(),0.80,0.00,lastVibrAmt(),fd);
    synthesize(0.15*vowlTimeRef,"ehh",lastGlot(),lastPitch(),0.80,0.00,lastVibrAmt(),fd);
    synthesize(0.05*consTimeRef,"kkk",lastGlot(),lastPitch(),0.00,0.00,lastVibrAmt(),fd);
    synthesize(0.05*consTimeRef,"kkk",lastGlot(),lastPitch(),0.00,0.00,lastVibrAmt(),fd);
    synthesize(0.02*consTimeRef,"kk+",lastGlot(),lastPitch(),0.00,1.00,lastVibrAmt(),fd);
    synthesize(0.08*consTimeRef,"kk+",lastGlot(),lastPitch(),0.00,0.20,lastVibrAmt(),fd);
    synthesize(0.05*vowlTimeRef,"ooo",lastGlot(),lastPitch(),0.80,0.00,lastVibrAmt(),fd);
    synthesize(0.15*vowlTimeRef,"ooo",lastGlot(),lastPitch(),0.80,0.00,lastVibrAmt(),fd);
    synthesize(0.05*vowlTimeRef,"eee",lastGlot(),lastPitch(),0.80,0.00,lastVibrAmt(),fd);
    synthesize(0.15*vowlTimeRef,"eee",lastGlot(),lastPitch(),0.80,0.00,lastVibrAmt(),fd);
    synthesize(0.05*vowlTimeRef,"ehh",lastGlot(),lastPitch(),0.80,0.00,lastVibrAmt(),fd);
    synthesize(0.15*vowlTimeRef,"ehh",lastGlot(),lastPitch(),0.80,0.00,lastVibrAmt(),fd);
    synthesize(0.05*consTimeRef,"mmm",lastGlot(),lastPitch(),0.80,0.00,lastVibrAmt(),fd);
    synthesize(0.15*consTimeRef,"mmm",lastGlot(),lastPitch(),0.80,0.00,lastVibrAmt(),fd);
    synthesize(0.10*vowlTimeRef,"mmm",lastGlot(),lastPitch(),0.00,0.00,lastVibrAmt(),fd);
    silence(0.1*vowlTimeRef,fd);
}
/********************** End of Singer Control Code ****************************/
```

Any ANSII C command that the singer instrument understands may be imbedded into the text stream in curly braces. An example which might be used at the beginning of a performance is {setClicking(0.4);}, which causes the singer instrument to write a click into the output soundfile each 0.4 seconds for metronome purposes. Pitches may be imbedded into the text stream by using parenthesis, such as (f4)re(g4)qui(440.0)em. All vowels have a default duration of *0.2*vowlTimeRef*, but may be extended by *0.2*vowlTimeRef* for each dash added after the vowel, such as in the example (f4)re---(g4)qui-----(440.0)e----m.

A Graphical Controller for the LECTOR System

The LECTOR Music Notation Window represents the first experiment in providing a graphical data input mode for the LATIN parser. The buttons at the top generate note symbols on the staff, which are then dragged with the mouse to the desired location. Text alignment with the note symbols determines duration, and the dashes in the text allow the parser to identify word continuations and boundaries.



Assuming that the clef of the example in the figure above is the G (treble) clef, the output of the LECTOR Music Notation Window is the following LECTOR text file:

```
/********************** LECTOR Text Control File ********************************/

    (f4)re-------(g4)---q(f4)ui--(f4)e-------m
    (f4)ae---(g4)----(a4)---t(a4)e---(g4)--------(f4)--------(g4)---r(g4)na------(f4)-------m.

/********************** End LECTOR Text ***************************************/
```

Future Directions

Much must still be done to make the system friendly for composition or instruction. The rules for parsing of other languages must be provided. The emphasis in the initial work has been upon providing many methods of generating and manipulating control code for the singer synthesis instrument. The graphical input system is the most promising, but has proven the most difficult to implement. It is planned to add features to the singer instrument which allow continuous scaling of time and frequency, such as envelope functions. This will provide control of natural deviations in the synthesis. It is planned to write rules within the Nutation system [5] to generate LECTOR and singer code.

References

[1] P. R. Cook, "Identification and Control of Parameters in an Articulatory Vocal Tract Model, With Applications to the Synthesis of Singing," Ph.D. Dissertation, Dept. of Electrical Engineering, Stanford Univ. 1991.

[2] P. R. Cook, "SPASM: a Real-Time Vocal Tract Physical Model Editor/Controller and Singer: the Companion Software Synthesis System," Colloque les Mod'eles Physiques Dans L'Analyse, la Production et la Cre'ation Sonore, ACROE, Grenoble, 1990, publication expected 1991.

[3] P. R. Cook, "Synthesis of the Singing Voice Using a Physically Parameterized Model of the Human Vocal Tract," Proc. of the International Computer Music Conference, pp. 69-72, Columbus, OH, 1989.

[4] J.C. Traupman, *The New College Latin and English Dictionary*, Bantam Books, NY, 1966.

[5] G.R. Diener, "Modeling Music Notation: A Three-Dimensional Approach," Ph.D. Dissertation, Dept. of Music, Stanford Univ. 1991.