N-Delta Builder : A Multiplatform Graphical Rapid Prototyping Tool for the Development of Voice Synthesis

DIONYSIOS B. POLITIS
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, GR-540 06
GREECE
dpolitis@csd.auth.gr http://www.csd.auth.gr

Abstract: - N-Delta Builder is a graphical, extensible, multimodal, platform-independent application for interactive, real- time design of voice signals. It consists of two parts: the server part, which is a physically modeled voice synthesizer and the graphical client part which drives in-situ the server. This paper apart from describing the arity N-Delta interfacing protocol between the client and the server, optimizes the process of synthesis by implementing a multi-platform, event driven Tool which is capable of visualizing subtle voice quantity and quality elements and handling as part of a generalized object oriented hierarchy musical events that describe or accompany the lyrics.

Key-Words: - Rapid Prototyping, Music Interfaces, Formant Synthesis, Music Delta Systems, Tcl/Tk.

1 Introduction

The major issue in the technology of Synthetic Speech systems is the ability to reproduce naturally and adequately synthetic speech. As far as naturalness is concerned, the current research trend focuses in physical modeling [1], i.e. in simulating the voice production mechanism as more accurately as possible.

On the other hand, the variation of human speech according to prosodic features and the musical reproduction and accompaniment, have to do with another more subtle "dimension" of pitch fluctuation, the melodic one. A "natural" synthetic utterance is not adequate unless it imprints precisely the melodic line.

When the concept of melody is used, a coarse description of the pitch fluctuations is perceived according to the staff lines. However, the musical patterns have been shaped and categorized through practice in many musical traditions rendering idiomatic prosodic and melodic patterns. This results to the following major issues in composing synthetic melodies [2][3]:

- the scales that are used in vocal reproduction may differ from the well-tempered ones used by the usual music notation in using smaller or incompatible intervals,
- vocal transitory phenomena cannot be depicted by using coarse descriptions of intervalistic pitch bending, and
- voice "quality" issues arise out of fixed idiomatic pattern performance.

In order to resolve these issues we have devised object oriented, real-time formant synthesizers that are capable of reproducing voice entities of big resolution in their time-frequency representation. These entities are perceived as voice quanta that virtually express a voice interval that is hardly audible or distinguishable. The synthesizer is accompanied by a multi-modal and multi-platform graphical user-interface which is the basic tool for the voice synthesis improvisation. This module serves as a client to the synthesizer-server. The graphical client has been developed under the Tcl/Tk environment in Unix and WinXX environments and incorporates visual technology to describe the melodic improvisation in terms of scales, notes, transitional phenomena and phonemes. The combined information is deciphered in a symbolic vector language of N arity [4] which is transmitted either locally or via TCP/IP *sockets* (if the client and the server are not on the same computer) to the synthesizer module in order to produce the melodic output.

2 The Hierarchy of the Object Oriented Formant Synthesizer

The entities that are exchanged in the client-server multimedia framework are the phonemes

of the uttered melody. Although the phonemes are the basic units of speech, they are not the atomic entities of the server module. The server uses two basic vocal sources, one for consonants (white noise generator) and one for vowels (impulse generator) [5], as seen in fig. 1.

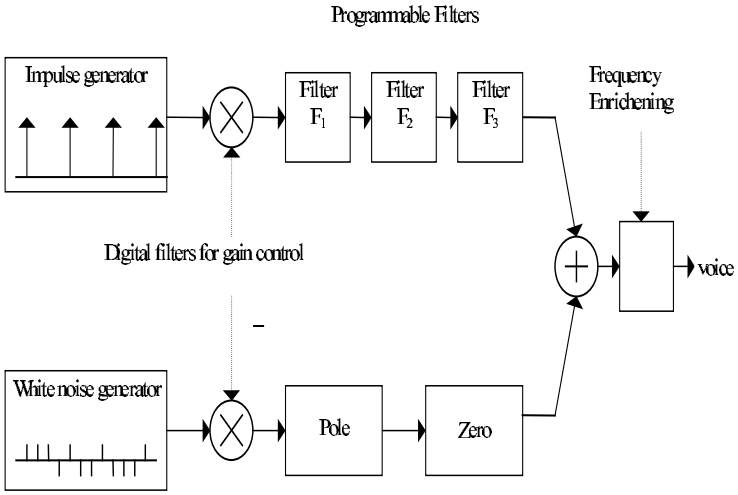

Fig. 1. Diagram of a cascaded formant synthesizer unit.

The shaping of the vowels takes place by passing through a filter bank of at least 3 digital filters with the characteristics of the naturally uttered phoneme.

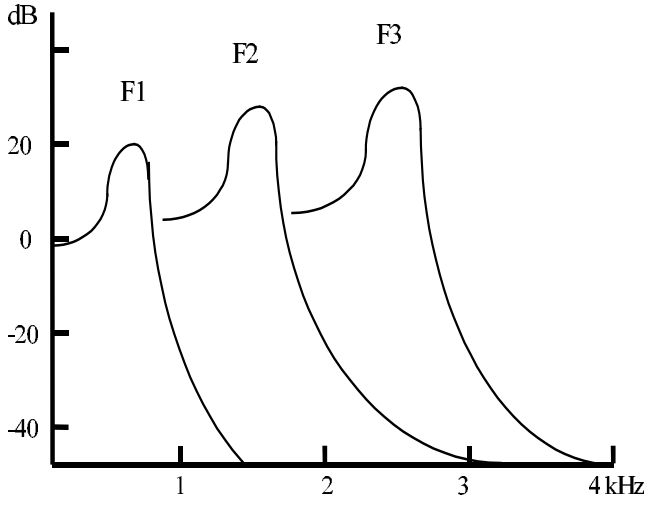

Fig. 2. Formant shaping.

The digital oscillator formed by the two zero and two pole filter bank is used to produce the resonance and anti-resonance characteristics of the vocal tract.

The oscillator described in fig. 3 is the Z transform of the two pole-circuit described by Gold and Rabiner [5]:

$$H(s) = \frac{s_n . s_n^*}{(s - s_n)(s - s_n^*)} \qquad (1)$$

where

$s = -\sigma + j\omega$,

$s_n = -\sigma_n + j\omega_n$ και $s_n^* = -\sigma_n - j\omega_n$. (2)

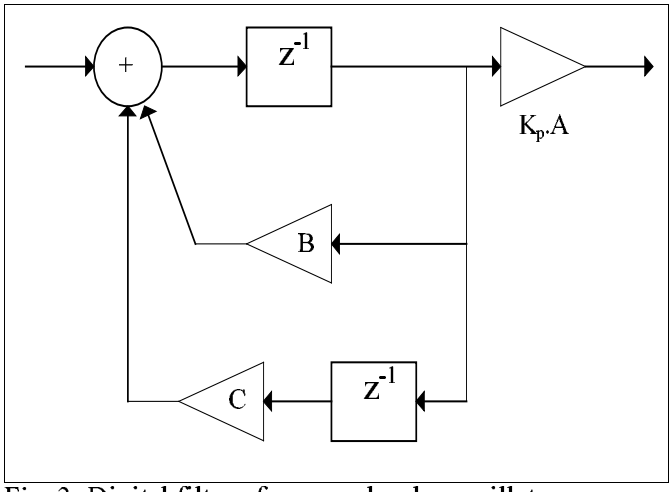

Fig. 3. Digital filter of a second order oscillator.

In (1) and (2) the analog pole circuit is described in its frequency domain, i.e. with its Laplace transform. The digital equivalent is readily calculated:

$$H(z) = K_p \frac{A.z^{-1}}{(1 + B.z^{-1} + C.z^{-2})} \qquad (3)$$

with

$$K_p = \frac{\sigma_n^2 + \omega_n^2}{\omega_n}, \quad A = e^{-\sigma_n T} \sin(\omega_n T)$$

$$B = e^{-2\sigma_n T}, \quad C = 2e^{-\sigma_n T} \cos(\omega_n T) \qquad (4)$$

Thus, the pole circuit of fig. 3 is described by (5).

$$y[nT] = K_p A\, x[nT-T] - B\, y[nT-T] - C\, y[nT-2T] \qquad (5)$$

where $T = f_s^{-1}$, the sampling period [5].

Following these relations, the basic object of the hierarchy is defined as:

```
class Object{
public:
int MIDI_control1;
int MIDI_control2;
int MIDI_control3;
int MIDI_mod_wheel;
int MIDI_after_touch;
protected:
Object();
~Object();
};
/* Sampling rate */
#define SRATE 22050.0
/* States for Envelopes, etc.        */
#define ATTACK 0
#define DECAY 1
#define SUSTAIN 2
#define RELEASE 3
```

Fig. 4. Basic class definition of the primarily audible object.

As it can be seen, there is MIDI compatibility and phenomena like *glissando* [2] can be expressed in terms of control sequences like ATTACK, DECAY, SUSTAIN etc. When it comes to voicing, this approach is insufficient, and derived classes are incorporated in order to shape the envelope of the synthetic utterance. Various transitory and "quality" phenomena have been reproduced by using elaborate time-frequency descriptions [6][7][8].

3 Visual Programming at client level

The shift towards black-box reuse of framework components has been implemented at the upper level of the application. N-Delta Builder is a completely graphical tool for voice synthesis. The main screen of the application can be seen at fig. 5.

As seen in fig. 5, the main screen is divided in three regions: **The menu bar**, **the button region** and the **graph region**. The menu bar contains all the commands not related to event manipulation. It contains file manipulation commands, sound output commands and various general commands. The button region contains the indicators, buttons and entry fields relating to event manipulation. Finally, the graph region apart from containing the visual representation of the events sequence, serves as an alternative way of event manipulation.

The conceptual space of the application is the graph region which is a schema of the time-frequency curvature of the fundamental frequency F0. Although the F0 plot is seemingly a continuous curve, neither intonation nor harmony is. Consequently, inherent to prosody is musical scale quantization, which defines the notes and the musical system. Since the usual music notation relies on staff views, the majority of interfacing protocols exchange MIDI sequencer data extracted from staff notation, with interactive editing capabilities. This means that musical events correspond to quanta of the time-frequency domain based on the centroid description of a note event by a key number, a velocity number and a duration one. These elements are included in the basic class definition for compatibility reasons with the MIDI interface which is used to drive the instruments (fig. 4).

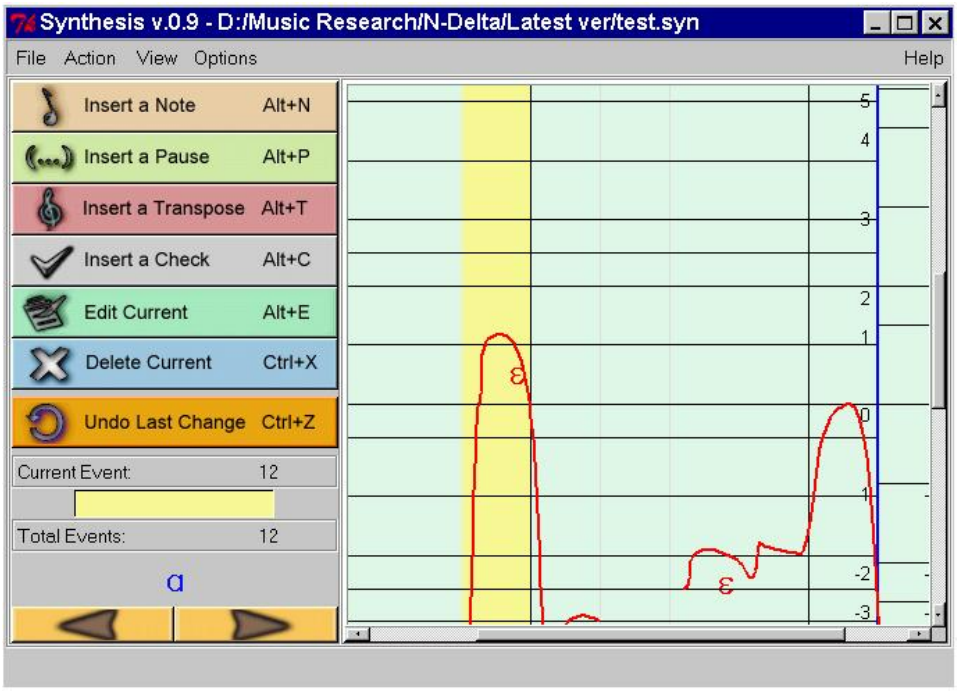

Fig. 5. The user interface of N-Delta Builder.

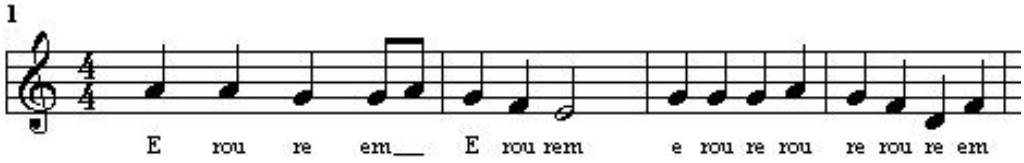

Fig. 6. A part of lyrics and staff notation from traditional Greek carols.

For the reasons explained previously, the definition of a note in the N – Delta [4] interface is somewhat different. For instance, the measures of the carols at fig. 6 are transcribed in the N-Delta language with the following sequence of musical events

S(d [d:12,10,8,12,12,10,8],2), Δ(+4, 1) [e], Δ(0,1,pi) [rou], Δ(-1,1) [re], Δ(0,1,pi) [rou], Δ(-1,1) [re], Δ(0,0.5,sigma) [em], Δ(+1,0.5), Δ(-1,1) [e], Δ(-1,1) [rou], Δ(-1,2) [em], C(+1) (6)

where

S(d,2) : defines the diatonic scale, and the second note of the middle octave; the diatonic scale used is defined by 72 **cents**, with scale quantization of 7 notes [12,10,8,12,12,10,8],

Δ(+4,1) [e] : is an alteration of a fifth, a time duration of a *noire* (♪) and [e] is the corresponding to the note *phoneme* /ε/ according to the notation of the International Phonetic Alphabet,

Δ(0,1,pi) [rou] : means the same note level for *morpheme* [rou] corresponding to *phonemes* /r//u/ and pronounced with the quality characteristic of a **petasti** (see reference [2]),

…

Δ(0,0.5,sigma) [em] : is the same note with the previous one, with a time duration half that of a *noire*, and the utterance of *morpheme* [em], analyzed to phonemes */ε//m/*; this note and the next are bound with the **sigma** operator, which implies a prolonged utterance of the same morpheme [em].

The phonemes used by the N-Delta interface are described with their first 4 formants. In table 1 the description of phoneme /ε/ can be seen.

In sequence (6) the arity of the interface is :

$$N = \max(\dim v_1, \dim v_2, \ldots, \dim v_n) \quad (7)$$

where

v1 = (+4, 1)

v2 = (0,1,pi)

… …

Although sequence (6) has an arity of 3, N-Delta Builder can cope with far more complicated scores ranging up to arity of 8. Presumably, the musical arity of a morpheme has to do with its expected performance complexity.

In order to compose sequence (6) with the N-Delta Builder, the **Note Attributes** menu is invoked which can concisely assist voice synthesis, as it can be seen in figure 7.

The curve that corresponds to each note operation on F0 will be imprinted on the curve of the melodic sequence, as seen in the **graph** pane of fig. 5. After the melody is given at the user interface level of fig. 5, the compiler is invoked yielding the sequence described at (6). This sequence will be pipelined to the real time synthetic singer and a voiced output

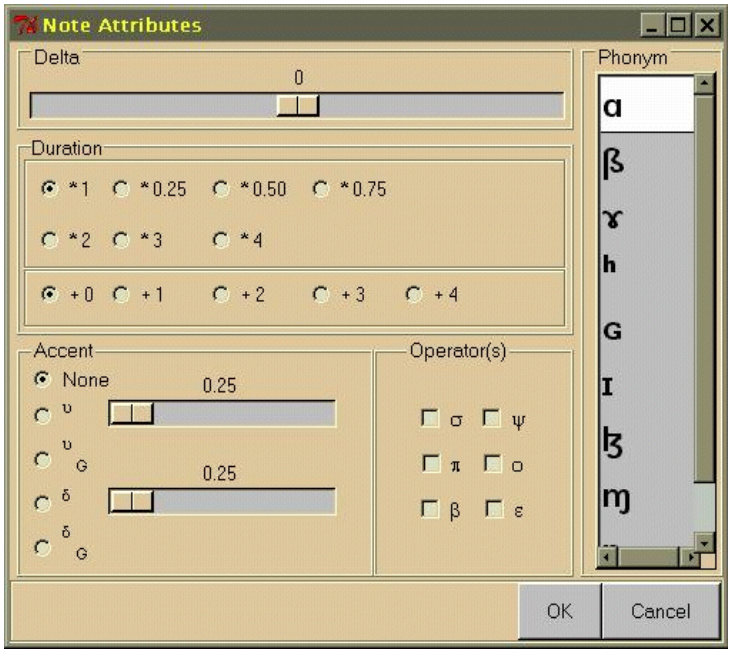

Fig. 7. Selecting note attributes according to the N-Delta will be produced.

4 Overview and future directions

Although only the basic functions of the N-Delta Builder prototype have been presented, the other functions not exploited are easily deduced. Lately, many products have been presented that incorporate visual interfaces [7][8]. The basic difference in their

Phoneme	ASCII description	formants (Hz)	Frequency width	Relative amplitudes (dB)
/ε/	ehh	515	0.977	0
		1805	0.810	-10
		2526	0.875	-10
		3103	0.400	-13

Table 1. Estimated formant values for a phoneme /ε/ pitch levels out of 5 speakers samples. The frequency width of the formants is not denoted in Hz but as it is inserted into the synthesizer, i.e. as the pole coefficient ($\exp(-\pi B f_s^{-1})$).

approach is the encompassed interface. Most of them are set out for the usual music notation without handling capabilities for alternate music sources [6][7][9]. The N-Delta prototype is a model for the standardization of future musical interfaces on a more 'chromatic' basis for their vocal performance.

References:

[1] Cook, P.R., "SIGGRAPH: Synthesis ToolKit in C++, Version 1.0", Department of Computer Science, Princeton University, May 1996.

[2] Pikrakis, A., Theodoridis, S., Kamarotos, D., "Recognition of Isolated Musical Patterns in the context of Greek Traditional Music", Special Session: Voice/Audio Processing Systems, *Third IEEE International Conference on Electronics, Circuits and Systems ICECS '96*, Rhodes, Vol. II, pp. 1223-1226, October 13-16, 1996.

[3] Spyridis, H.C., Politis, D.V., "Information Theory Applied to the Structural Study of Byzantine Ecclesiastical Hymns", *ACUSTICA*, Vol. 71, No. 1, May 1990, pp. 41-49.

[4] N. Mastorakis (editor), *Recent Advances in Information Science and Technology*, World Scientific Pub. Co., 1998. (Politis, D. et al., "A Voice Scripting Language Interface of N-Arity for an OO Formant Synthesizer", pp. 247-251).

[5] Klatt, D.H., "Software for a cascade / parallel formant synthesizer", *Journal of the Acoustical Society of America*, Vol. 67, pp. 971-995, 1980.

[6] Politis, D., Tsoukalas, A., Linardis, P., "Interpretation of Byzantine Music Notatuion as Adaptive Δ-Modulation", Special Session: Voice/Audio Processing Systems, *Third IEEE International Conference on Electronics, Circuits and Systems ICECS '96*, Rhodes, Vol. II, pp. 1219-1222, October 13-16, 1996.

[7] Politis, D., Tsoukalas, A., Linardis, P., "VIDI-A Voice Instrument Digital Interface for Byzantine Music", *International Computer Music Conference ICMC97*, Thessaloniki, Proceedings, pp. 403-407, September 25-30, 1997.

[8] Porcaro, N., Jaffe, D., Scandalis, P., Smith, J., Stilson, T., Van Duyne, S., "SynthBuilder: A Graphical Rapid-Prototyping Tool for the Development of Music Synthesis and Effects Patches on Multiple Platforms", *Computer Music Journal*, Vol. 22, No. 2, Summer 1998, pp. 35-43.

[9] Mastorakis, N.E., Gioldasis, K.D., Koutsouvelis, D. and Theodorou, N.J., " Study and Design of an Electronic Musical Instrument which accurately produces the spaces of the Byzantine music", *IEEE Transactions on Consumer Electronics,* Vol.41, No.1, pp.118-124, February 1995.