A SCRIPTING LANGUAGE DEVELOPMENT FOR MUSIC PERFORMANCE

[1]*Nima Darabi*, [2]*Naiem Khodabandelooyeganeh*, [3]*Sirous Rezaee* [4]*Mahdi ZiaeNia*

[1]Sharif University of Technology, Dept. of Computer Science, Azadi Ave., Tehran, Iran
[2]Qazvin IA University, Barajin Road, Qazvin, Iran
[3]Khaje Nasir University of Technology, Seyyed Khandan Bridge, Tehran, Iran
[4] Tehran Central IA University, Imam Hossein Square, Tehran, Iran

ABSTRACT

The present article discusses the general theory and the design model of the authors' approach to music performance software. The software developed within this framework enables a user to play back music with optional musical instruments, combine them, and add special effects to the musical acoustic waveforms.

The *Navazande* software, developed in Borland Delphi 5, is a scripting and programming environment which receives musical notes in its native scripting language in a text format and saves the output as a sound file. A Musician familiar with the software, besides choosing from a wide range of musical instruments, can specify properties of the musical waveform, such as its attack and decay times, damping and harmonic coefficients, volume envelopes, etc. The software will then playback the piece.

Our *Navazande* allows musicians to use complex mathematical operations in the sound creation process, allowing, essentially, for unlimited creative opportunities. This fusion of mathematical analysis and synthesis, together with simple use of performance capabilities, traditionally, has not been available to musicians, neither in current sound analysis software, nor in music performance and annotating software.

In order to demonstrate the potentials of our approach, we have also included a number of musical pieces performed with our software.

1. INTRODUCTION

Music is sound waves propagating in air. We can produce these sound naturally (mechanical) or artificially (software). Our tool is made for the second reasons. There are many tools producing music, but with some limitations: Most of these tools are developed in west and are not able to play eastern notes with frequency difference of less than half of the tone. There will be no Iranian instrument in those tools instrument lists. Some simulated instruments also look unnatural.

This tool does not have mentioned limitations and on the other hand, it has more unique and new abilities. For example, each instrument has the ability to apply special effects to the playing note. Most of the string and wind instruments have the ability of applying vibration and glissando to notes. Next point is that our tool provides all abilities of a single instrument for other instruments. It is possible to start a piece with some instruments and change the instrument type during the play continuously. It is also possible to define a new instrument. We can make novel changes to the method of playing the instrument. Programming capabilities gives the player to implement his ideas directly to the sound, instead of showing them on symbolic notes.

2. INSTRUMENT PLAYER AS SYSTEM

Music pieces are viewed as a sequence of notes, just like the way letters make a speech. The player reads all information about the piece by this language and then translates them to mechanical actions on the instruments. Music (generated sounds) is the output. This music is again used as feedback to correct later notes. Figure 1 shows the system of instrument and the playing piece.

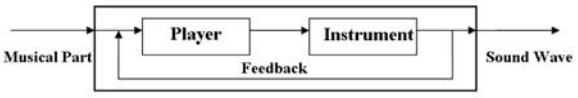


Figure1. *Block diagram of the system for playing a musical piece*

A tool which gets a list of notes and plays a sound signal as output can simulate the player and the instrument together. There is no distinction between the player and the instrument in this case. It is the main goal of our tool (*Navazande*). We can choose the input and output of our system from different file formats: we input the note sequences with a computer programming language in normal ASCII format and output the result as generic sound file formats.

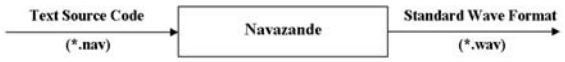


Figure 2. *Block diagram of Composer tool as interpreter for a programming code in order to generate music signals.*

Input file format: *Navazande* should translate the piece to a script of predefined commands. These files are saved as *.nav files in ASCII.

Output file format: We use generic wave file to store output of the tool. The most used file format to store sound waves is Standard Wave Format (*.wav) which is a sound signal stored as digital data with a constant granularity (sampling frequency, sampling accuracy, number of channels and etc.) [1].

3. USING COMPUTER TO RUN *NAVAZANDE*

We can simulate a group of instruments playing together just using the software. We start from playing of a simple signal up to playing of a whole orchestra masterpiece.

3.1. Development of a signal with the desired frequency and duration

Frequency, is the key factor of a note. Music sounds are usually made from a single frequency or they have a significantly stronger frequency. When we call A3 (A from the third octave), we mean the frequency 440Hz in any instrument. Intervals in the western music are based on semitones. A tone describes the difference of frequency between two consequent notes. Frequency of consequent notes forms a geometric progression. Each octave consists of six tones (12 semitones) and each note's frequency is twice the frequency of the same note in the lower octave. We can find the parameter of this progression as follows:

$$q = 2^{\frac{1}{12}} = 1.05946 \quad (1)$$

First line of figure 3 consists of the names of the notes forming an octave. Third line introduces the frequency of each note per source note (C for the lower octave). Second line is the linear distance of notes based on tone. This is calculated by calculating logarithm of the third line.

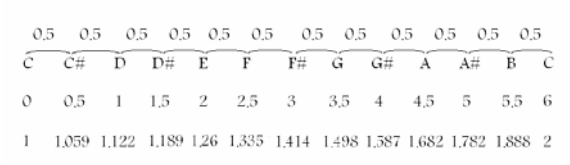


Figure 3. *Comparison of linear trend of western music intervals with logarithmic trend of frequencies.*

Consider a note with frequency of f1 to be the base note, frequency of a note which is m21 tones higher (f2) can be calculated as follows [3]:

$$\frac{f_2}{f_1} = 2^{\frac{2m_{21}}{12}} = 2^{\frac{m_{21}}{6}} \quad (2)$$

Now we can calculate the frequency of a note (f_0) by knowing its name based on the basic node and produce it using an algorithm based on specific amplitude (V_0). Then we can save it with the sampling frequency of fs and granularity of N bps and save the result in a wave file and listen to the sound signal which looks like an alarm tone. The i^{th} sample of sound sinus wave signal which is generated and has to be saved in the file is calculated like this:

$$DigitalSample = V_0 Cos(2\pi.i\frac{f_0}{f_s})2^n \quad (3)$$

3.2. Extracting signal's shape

Shape of a signal is the key factor to understand the sound's characteristics. Main factor of difference between the resulting sounds from two different instruments is the signals shape. Instruments from the same family which have same sound characteristics and physical structure; so, we need to extract the signal's shape of an instrument to be able to simulate that instrument.

Regarding to Fourier theory, a signal's shape is related to harmonic factors of the signal. Harmonic factors are simple musical signals which their frequency is relative to the base frequency. Having a periodic real function like *Func(t+T)=Func(t)*, we can generate it from sum of these harmonics. There are A_k and P_k for each wave satisfying the following equation:

$$Func(t) = A_0 + \sum_{K=\infty}^{K=1} A_k.Sin(2k\pi f_0 t + P_k) \quad (4)$$

The A_k coefficients are ignorable when k is increasing. We consider the first h harmonic factors A_1 to A_h. Moreover, the coefficients P_k describing the phase have not effects to how the synthesized wave sounds like. Besides, our signal has not DC level and we can omit the

factor A_0. Now knowing several harmonics for an instrument we can synthesize its sound. We can just save the i[th] sample calculated from the formula below:

$$DigitalSample = \sum_{K=h}^{K=1} V_0 A_k .Sin(2\pi .i \frac{f_0}{f_s} t) \quad (5)$$

3.3. Change of harmonics during the note playing

In order to synthesize the sound of an instrument more accurately, we divide a note into several durations such as starting noise, attack, steady part, and decay; Harmonics array differs from each other in each of these durations, so any sample should be generated based on the harmonics array of the duration it lies in. We can change the harmonics array continuously during playing the note, so that it can be heard more natural.

3.4 Simulating percussion instruments and the noise parameters for single frequency instruments

As percussion instruments usually do not have single frequency notes, their sound contains a wide range of frequencies. Even melodic instruments sound the similar voices especially when they start to play a note, such as blowing sound in the beginning of a flute note, the first hammer beat of a piano note, sound of bow on the string in a violin. We can use white noise for simulating such voices and add it to the synthesized signal. For producing such noises we perform the Inverse Fast Fourier Transform (IFFT) on the amplitude spectrums obtained from the real instruments.

3.5. Vibrating the sound

Most of the woodwind and string instruments are able to produce vibration in the both amplitude and frequency. This happens with vibrating the player's fingers over the holes of the woodwind or on the strings of the string instruments. After analyzing the signals of vibrating notes of different instruments we reached a simple and useful model covering all of them properly:
Vibration in amplitude or frequency acts as AM or FM modulation respectively. In these modulations a simple sinusoidal signal (with frequency of the vibration of the player's fingers) carries over the career signal (with the frequency of the playing note).
In fact, using the vibration in amplitude will cause the maximum of the picks to oscillate around the amplitude average. Also, using the vibration in frequency will cause the pitch of played note to oscillate around a central frequency. Studying of a vibrating note, one can detect both of these vibrations. We define three parameters in order to implement the vibration:

VibSpeed is the frequency of the message signal in AM or FM modulation, which can be the frequency of player's fingers over the string (around of a few cycles in the second) . *VibFreq* is used for FM modulation to measure the change ratio of the pitch and can be defined with $\Delta f/f_0$. *VibVol* is a similar quantity for AM modulation and can be defined with $\Delta V/V_0$.
To vibrate a simple signal with these parameters, the amplitude of the envelope to be applied to the sinusoidal signal with the frequency of f_0 and amplitude of V_0 can be calculated as follow:

$$V = V_0 + V_0 .VibVol .Sin(2\pi .VibSpeed) \quad (6)$$

Which is amplitude for resulting signal after carrying the message signal *Sin (2πt.VibSpeed)*. The parameter *V* should be multiplied by the main signal. In order to apply the frequency changes and implementing the FM modulation we should replace the following angle (φ) instead of $2\pi t.VibSpeed$ in the formula 6:

$$\varphi = 2\pi f_0 t + 2\pi VibFreq \int_0^t Sin(2\pi .VibSpeed .t)dt \quad (7)$$

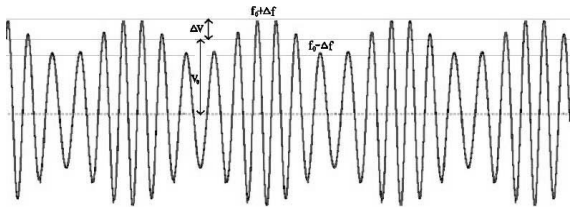


Figure 4. *Using AM and FM modulation to make vibration in the instrument's voice.*

3.6. Design of sound's envelope

Sound signal shape which is made from joining consequent maximums of periods is called the sound's envelope. This diagram is another important factor for understanding sound characteristics of an instrument. Different instruments will be different from this aspect. String instruments make sound by beating a string with two fixed ends. They make a constant wave which fades out by time. Fade out usually has a logarithmic tail; so we can set a logarithmic fade out factor to the wave showing that sound is fading out after rising fast (figure 5). This kind of fade out does not exist in wind instruments; while the air blows into the instrument, it makes sound (figure 6).

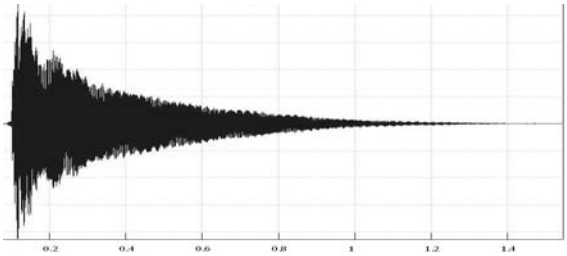


Figure 5. The harp note rises intensively and fades out with a logarithmic tale.

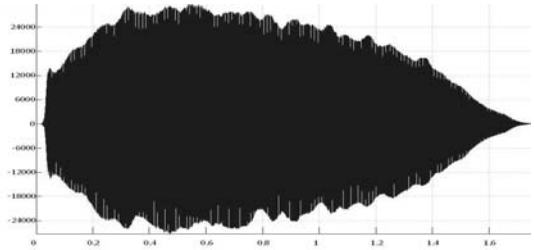


Figure 6. *The Oboe makes sound, while air blows in it.*

Studying the different kinds of envelopes for different instruments, one can find a general pattern which can cover all the possible shapes of the envelopes with changing in its parameters. We use the model shown in the figure 7. We normalize the time axis so that time unit becomes the duration of the played note. During the time this note played from 0 to 1, amplitude of the wave determined by the pattern of the (x_i, y_i) points. They are connected with straight lines. We just multiply each sample of the signal by the function shown in the figure 7 in order to apply the envelope to the synthesized simple signal.

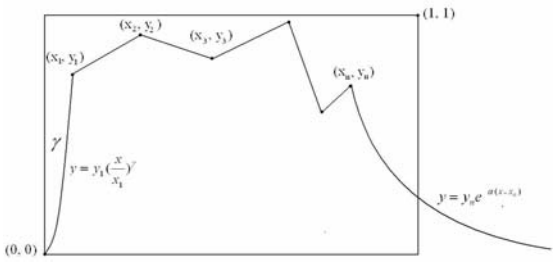


Figure 7: *The pattern used in our tool to model the envelope of a musical signal*

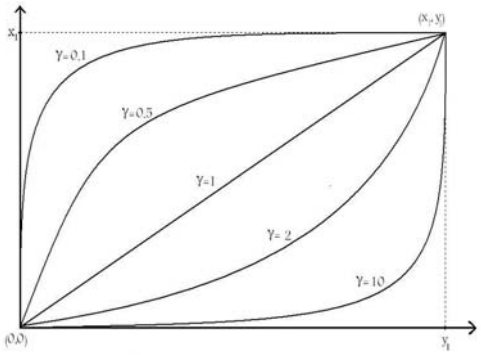


Figure 8. *Changing the parameter γ determines the concavity of the envelope in the attack duration of the note (from starting playing the note to the first point defined in the pattern)*

$$y = y_1(\frac{x}{x_1})^{\gamma} \quad (8)$$

Except offset all of the points (x_i, y_i) are connected to each other through straight lines. The connection's function definition is based on formula 8. This function meets the points $(0, 0)$ and (x_1, y_1) as expected, but the parameter γ should specify the concavity of the curve connected these two pointes. If $\gamma=1$ then this line is straight. The curve concaves upward for $\gamma>1$ and downward for $\gamma<1$. We defined parameter γ in order to simulate the attack of the synthesized wave to be more similar to the real wave. It is very clear in Figure 7. It is possible to hear the fading part of the wave after stopping the note's playing. The tale of the signal shows the fading of its amplitude based on the α factor. In order to customize envelope pattern for signal amplitude we should specify right values for γ parameter and right points for (x_i, y_i).

3.7. Simulation of simultaneous sounds

Usually different sounds mix together at same time In a part of music cause of three reasons:

1. Mixture of fading part of the previous played waves and current waves.
2. Playing different notes at same time with an instrument like piano
3. Playing different notes at same time with different instruments like an orchestral piece.

For simulating this mixture we should perform the calculation for a single sample of each signal seperatly and then save their summary in the wave file.

4. *NAVAZANDE*'S INTERFACE

Our interface is as simple as a common text editor. The text is a code which contains musical notes sequence (their pitch and duration) and definition of the instruments. After execution of this code a sound file will be generated as the output. The *Navazande* program executes all the commands as an interpreter and it will report in the case of occurring any error. The content of the *Navazande* program includes about 25 commands and some constants up to now. It has two parts: header and body. Error detection process is based on LALR compiler, which has a fine parsing and error detection speed [4]. In this paper it is not possible to describe the detailed procedures of compiling and error detection as well as syntax description of the developed scripting language. In the figure 9 the correspondent code for performing an Iranian musical piece "Zard -e- Malije" with the instrument "Iranian Tar" is illustrated.

4.1. A practical example

We aim to simulate the sound of an Iranian instrument "Tar". First, a pure and simple note of the instrument is needed. We choose a one second typical played note, then obtain its amplitude spectrum in the frequency domain with performing the Fast Fourier Transform (FFT).

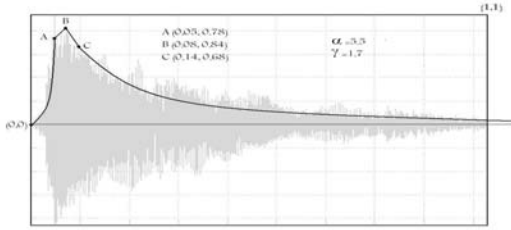
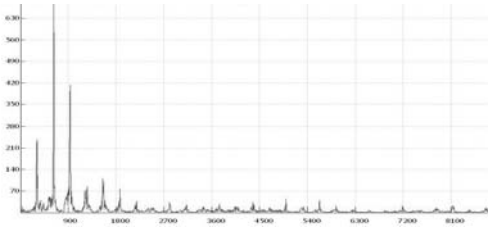


Figure 10. *Time diagram sample note for Iranian instrument; Tar. We can figure shape parameters for domain out of this diagra*m.



A_k=[0.31, 1, 0.6, 0.12, 0.16, 0.13, 0.04, 0.02, 0.05, 0.03]

Figure 11. *Frequency diagram. Main harmonic parameters can be figured out of this diagram.*

A good approximation of the wave can be found by multiplying numbers of A_k array (figure 10) by first to tenth harmonic sinus functions of formula number 5. Outcome will be an acceptable approximation for Tar instrument's sound. In addition we need to simulate the signal shape (figure 11), so we need to analyze Tar's sound versus time. For this reason, we put the whole signal shape of the sample note from beginning to end in a unit square, and then we choose enough point on the shape to have a good approximation for the covering curve. We find α using the dying part of wave and γ using the ascending part of wave from beginning to the first point. Parameters defined in figure 12 can simulate the domain covering function with a good approximation. Now we can have any music with any instrument's sound just by setting these parameters correctly.

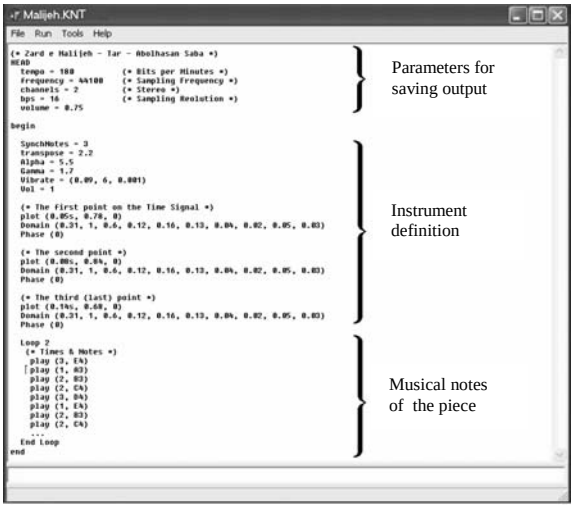


Figure 12. *Code for playing a famous Iranian musical piece "Zard-e-Malije"from "Abol-Hasan Saba"*

5. REFERENCES

[1] http://ccrma.stanford.edu/CCRMA/Courses/422/projects/WaveFormat

[2] Steven W. Smith, "The Scientist and Engineer's Guide to Digital Signal Processing", *California Technical Publishing*, pp. 169_184, pp. 311_318, 1999.

[3] N. Darabi, "Music from the perspective of digital signals" (English title), *Proceeding of 7th Iranian students conference on electrical engineering*, 2004.

[4] Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman , "Compilers: Principles, Techniques, and Tools", *Addison-Wesley Publishing Co.*, 1986

[5] N. Darabi, N. H. Azimi, H. Nojumi, "Recognition of Dastgah and Maqam for Persian Music with Detecting Skeletal Melodic Models", *The second annual IEEE BENELUX/DSP Valley Signal Processing Symposium (SPS-DARTS 2006)*