Musicstring – tunes packed into a bunch of Ascii characters

Tomáš Mariančík

Bachelor Degree Programme (1), FIT BUT

E-mail: xmaria01@stud.fit.vutbr.cz

Supervised by: Miloš Eysselt

E-mail: eysselt@fit.vutbr.cz

**Abstract**: The most common notation used for writing music today is the modern musical notation, using graphical symbols called notes on a five-line staff. Writing these symbols on a computer requires specialized programs and they cannot be easily shared via text based media on the internet. MusicString is a language that implements a subset of the modern musical notation using only characters found in the ASCII table with emphasis on a very short representation while maintaining at least some level of clarity, allowing easy sharing of such tunes via instant messaging services, forums, internet relay chats or micro blogging services.

**Keywords**: EEICT, music, language, audio, synthesis, ASCII, plaintext, sharing, social

# Introduction

MusicString is a music programming language designed for writing music and simple melodies using as few ASCII characters as possible, while keeping the code human readable and straightforward. Project includes an official audio synthesis library that converts the MusicString code into raw audio that can be played on various platforms and will always sound the same, no matter where it was synthetized.

Purpose of this language is to allow for easy sharing of such tunes via text based media by simply copying and pasting short pieces of text. Such tunes can be shared on websites, forums, via instant messaging, internet relay chats or micro blogging services such as Twitter or Google Plus. Implementation for mobile systems also allows sharing via SMS messages. This paper presents the most important features of the MusicString language.

# writing Tones

Majority of music can be expressed using a series of musical notes on a five line staff, where vertical position of the symbol determines the pitch and the symbol itself determines duration of the tone. Thus, these are the two most important parameters that must be expressed using MusicString code. For this purpose, I chose the English alphabet to represent individual tones, where each letter represents a different pitch, whereas duration can be changed using digits. This way, a simple association letter - tone/pitch and digit – duration, is formed.

In order to synthetize audio the frequency of the signal corresponding to given pitch must be known. For this I used equation (1) from music theory which corresponds to majority of most Western-style music. [1]

(1)

The parameter *n* is equal to zero for pitch A4, which corresponds to frequency 440 Hz, which means that whole series is centered around A4. For every lower pitch, *n* is decremented by one and for every higher pitch, *n* is incremented. I used whole English alphabet, with alternating lower-case and uppercase letters and assigned value 0 to letter „n“. Every letter before has value decremented by one and every letter after has value incremented by one, which gives following range:

|  |  |  |  |
| --- | --- | --- | --- |
| Letter | n | f | Pitch |
| a | -26 | 98 Hz | G1 |
| n | 0 | 440 Hz | A4 |
| Z | 25 | 1864 Hz | A#6 |

1. Basic frequency range of MusicString

This range can be further extended by a shift statement, which allows the composer to shift the range of whole alphabet to a different scale, both up and down, using ampersand in combination with alphabet letter. Shifting frequency range needs to be done only once and is valid for all subsequent tones. Using this, the dynamic range has following minimal and maximal pitches.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Letter | Shift | n | m | n+m | f | Pitch |
| a | &-a | -26 | -53 | -79 | 4,59 Hz | D-3 |
| Z | &+Z | 25 | 51 | 76 | 35479 Hz | C#11 |

1. Dynamic frequency range of MusicString

Similarly to frequency, tone duration must be calculated in seconds. In modern musical notation, duration is represented by different symbols, however it has simple mathematical basis. We need to know the tempo in beats per minute, which most commonly determines how many quarter notes are there per minute. Using tempo, which is 120 bpm by default in MusicString but can be changed, we can calculate duration in seconds of the quarter note and then calculate longer notes by multiplying by two and shorter by dividing by two. By using all decimal digits with “5” being the quarter note, duration is calculated using equation (2) [1].

(2)

MusicString additionally supports altering the currently set duration using a ratio, where the current duration is multiplied by given ratio, or by using the dot notation, by specifying number of dots as in modern music notation, where each dot adds duration of a half of the current note to the overall duration.

# control statements

MusicString supports several control statements that simplify composing, shorten resulting string or add new functions.

## forking – from chords to channels

Because most music frequently uses more than one tone at once, MusicString provides a way to fork one sequence of tones into multiple independent ones, using parentheses with multiple sequences separated by commas. Once such statement is encountered, all sequences start playing at once. They can be as short as a single tone or longer, working as separate parallel channels. Because sequence can contain any valid MusicString statement, it is possible to combine various pitches, durations and also use multiple instruments and instrument sets.

## loops

Using square brackets, composer can repeat same sequence using a specified number of times, using an integer at the end of the statement. In case he needs to repeat portion of a sequence without knowing the number of repeats, infinite loop can be used. Such loop can be given a name, which can be later used to stop this loop from other channel. This way, composer can have a portion of sequence loop, providing a background rhythm for another part of the composition, stopping it automatically once it finishes playing.

## subroutines

A portion of the sequence can be repeated at several different places in the composition. Because this can’t be solved with loops, instead, it’s possible to predefine a subroutine with a simple name, which can be used to play the sequence repeatedly. In combination with support for arguments, additional tones can be inserted at any place as well as control statements which can modify the predefined sequence.

## instrument set

MusicString provides support for six different instruments by default, however it can be further extended with so-called instrument sets – packages that replace the six instruments with different sounds. The sound can be generated from audio samples or using a function *y = f(x)* that will be used to calculate individual samples. In combinations with forking, it is possible to use a different instrument set for each channels and because the number of channels is theoretically unlimited (practically limited only by memory and processing power), MusicString is capable of using theoretically unlimited number of instruments in a single composition.

# Toolset

Several software projects are part of the official MusicString implementation. At the core is the MusicString compiler, which is divided into two major parts. MusicString parser, which takes care of parsing the code and generating a linear series of tones, called MusicList, which is similar to machine code – a series of instructions that specify when should a new tone start playing and what are its properties. Second part is the audio synthesis engine, which takes MusicList as an input and produces raw audio data that can be copied into buffer and played in real time or stored in an audio file. However, additional modules can be written, so that MusicList can be transformed into variety of other formats, such as MIDI or module files, without having to worry about MusicString syntax and parsing, thus simplifying development of extensions.

MusicString compiler is written in Standard C++ with use of only standard libraries, which allows it to be compiled for variety of platforms, at the time of writing of this paper, MusicString compiler works on contemporary versions of Windows, Linux and Mac OS X.

Second project is Visual MusicString, an IDE for the MusicString language. It provides easy to use interface and various tools that simplify writing, debugging and playing of MusicString. Visual MusicString is written in Standard C++ as well, with addition of Qt extensions and uses Qt SDK for real-time playing and GUI features. Thanks to the usage of a multiplatform library, Visual MusicString is currently available also for Windows, Linux and Mac OS X. Versions for mobile devices and integration with web browsers is currently in progress.

# conclusion

MusicString implements an important subset of modern music notation with support for multiple channels and instruments and allows simple sharing via text based media thanks to succinct, but still human readable syntax.

reference

1. Nattiez, J. J.: Music and Discourse: Toward a Semiology of Music, Princeton University Press, 1990, ISBN 0691027145