Lisp in Summer Projects Submission

Submission Date	2013-10-17 11:47:17
Full Name	Bruno Burke
Country	Deutschland - Germany
Project Name	Musilisp
Type of software	library
General category	music
LISP dialect	Commmon Lisp
GitHub URL	https://github.com/BurkeB/MusiLisp
Did you start this project?	Yes, all the code is written by me
Project Description	I want to describe my project in this form.
Purpose	Musilisp – A Software Sound Synthesizer or Music Generator, written in Common Lisp.
Function	This program generates a .wav Music File from a given Melody Text-String with a self created oscillating Function - Sine, Squarewave, Triangle, or whatever you define.
Motivation	I'm a computer science student and musician. I always wanted a modular synthesizer keyboard but never had the money. Last semester I finished an A.I. Course in my university where I began to learn Common Lisp programming. To get more into things like audio modular synthesis and to get more experience in Common Lisp, I combined this two goals and began to develop Musilisp.
Audience	People who are interested in Music and Computer Science. Or perfect for people who are traveling, want to compose a little Melody and dont have an instrument with them. This will also be designed as a library, that it can be used by other programs – more about this in Section “Conclusion”

Methodology

This Project can be divided into several parts.

1. Note informations
By a given note like “c'8” the program have to calculate different informations for this note. The other parts of the program are working with absolute frequency, so you have to calculate which frequency is represented by a note like c'. Also the song which will be generated has a special tempo, given in beats per minute, with this information you can calculated how long a note lasts in (milli-)seconds.
So its possible to define with which frequency the tone will be generated and how long it sounds.

2. Sample Generator
This will read the melody note by note and feed an oscillating function (made by a make generator function) with the calculated frequency and duration of the note. This samples will be collected in a list and later append to the Data Header of the wave file.

3. Generator Functions
The tone-sampleswill be generated by a simple function, for example Sine. The Sample Generator needs a function where he can get a volume value for a sample with a specific frequency. The Generator Function will generate this function, for example a Sine function which oscillates in the right frequency depending on the needed frequency and the samples per second of the music file.

4. Various other functions
My program uses different number formats, for example double, signed-int, int or the automatic standard format in which lisp calculates with numbers. Also because of the rules of the Wave-Header the program must represent numbers in a specific number of bytes and you have to convert numbers from big-endian to little-endian or the other way.
Also there are some functions to manipulate or merge lists to get the lists in the right way the program needs them.

Program flow:
1. Generate the Sample-List
2. Generate the fmt-Chunk-List
3. Combine the data-Chunk with the Sample-List
4. Generate the Wave-Header.
5. Combine everything in the right direction and write it to the byte-stream.

Conclusion

At the moment the program has many limitations, for example the melody has no volume dynamic. It's always maximum volume. Also it's not possible to create real polyphonic music. I think, because musilisp is able to work with biphonic-melodies, it is a small step to implement support for real polyphonic melodies.
Another bad thing of this program is the performance, if it's possible I will write the code more efficient and maybe add some multithreading concepts.

While programming Musilisp I learned so many new Lisp techniques, actually I will rewrite many functions. Because I'm new to Lisp there are so many techniques I don't know (Macros, CLOS...), so that my future tasks on this program will be learning Lisp and refactor the code with the new

knowledge.
There are two directions im focused on for the future of this project.

1. Additional Composing Tools
There should be a more abstract way to write music. For example, just enter some Chords and the programm will automatically generate a Walking-Bass-Line to the given Chord-Sequence.

2. Computer Vision
This semester I visit a course about computer vision. My idea is that I can make a photo with my webcam/smartphone of a music sheet and then the programm generates a wave-file or maybe directly output this musical information to the soundcard.

Build Instructions

This project doesn't use asdf or some other project packaging software, so you have to load each file by hand. Load and interpret the files in this order:
1. functions_tone.lisp
2. functions_instruments.lisp
3. musilisp.lisp
4. bourree.lisp
The main development was done on a 32 Bit Windows 8 with LispIDE + SBCL.
This project is platform independent, but there is one little bug that only appears on non-windows systems. Please see the Section “Bugs or caveats” if you're running a Linux System.

Test Instructions

This project doesn't use any unit-testing framework or something similar. But below some functions you find comments with example function-calls and their expected results.
For testing this project load and interpret the four files, described in section “Build Instructions”. Now you can generate the song Bourrée by Johann Sebastian Bach with this function call:

```
(musilisp "bourree_sinus.wav" (bourree) :bpm 150
:instrument #'make-mysin-octave)
```

Please ensure that you have write rights in the folder of the given file-name. If you just enter a file-name (like in this test) instead of a path+filename as the first function parameter, it probably uses the folder of your lisp-interpreter.
Bourree is the function from bourree.lisp which generates a list of two strings. Bourree is a musical piece with only two voices, so it can be written as a list of two strings.

If everything worked well you can play the wave file and hear some Bach-Music played by a sinus-generator.

Execution Instructions

Executing this program mostly happens with the function named “musilisp”.
This functions needs two parameters, first one is the file (path+filename) where you want to write your music, and the second one is the melody which will be written. The melody can be a string or a list of 2 strings. At this moment the program is only able to play monophonic or biphonic melodies.
If you want to generate your own song you can simply enter

a String, for example "c4 e4 g4 c'1" or if you want to use two voices you can enter a list of two strings, for example (list "c4 e4 g4 c'1" "c16 e16 g16 e16 e16 g16 h16 g16 g16 h16 d'16 h16 c1")

There are two key-parameter, bpm and instrument. With :bpm you can set the tempo of the song. Standard value will be 120. With :instrument you can say which function should be used to generate the music. Standard Instrument will be #'make-mysin (see functions_instrument.lisp).
These instrument maker functions will get 3 parameters from the program:
-frequency
-maxvolume
-samples per second
As the return value there should be a new function which only has one parameter that represents the sample for which you want to have the function value.
With other words: The maker function returns a function which will oscillate with the given frequency parameter (in relation to samples per second), with an amplitude of “maxvolume”.
With this concept it is possible to create Instruments which will add overtones to the normal frequency or you can modulate the signal. This can be complex stuff, or just like in #make-mysin-octave a normal octavation of the melody.

Describe any bugs or caveats

Due to the fact that I'm new to lisp programming, this program has some mistakes and isn't working on every operating system. Most development was done on a 32 Bit Windows 8 with LispIDE + Clozure Common Lisp or SBCL, where the program runs without errors.
On Fedora Linux I get an error because of the function write-tone in musilisp.lisp. I think this error will be on all Linux systems. In this function the parameter frequency is declared as Integer. On my Windows System double values will be converted to integer by this declare, but on Linux this seems to not work and will be handled as an error if you put a non-integer value as the frequency-parameter.
So if the jury don't have a Windows machine with CCL, or the error also appears on Windows, you can use this workaround to get the program running.
Replace the function
~~~~~~~~~~~~~~~~~~~~~~
(defun write-tone (frequency seconds instrument)
(declare (integer frequency) (float seconds))
(let* ((samples_per_second 44100)
(samples (* samples_per_second seconds))
(mysquarewave (funcall instrument frequency 32767 samples_per_second)))
(loop for i
from 0
to samples
collect (funcall mysquarewave i))
)
)
~~~~~~~~~~~~~~~~~~~~~~

with this function:

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
(defun write-tone (frequency seconds instrument)
(declare (float seconds))
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

	(let* ((frequency (round frequency)) (samples_per_second 44100) (samples (* samples_per_second seconds)) (mysquarewave (funcall instrument frequency 32767 samples_per_second))) (loop for i from 0 to samples collect (funcall mysquarewave i)))) ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ This is the only known error. The program is not finished so its very pedantic in terms of the input melody format and other function parameters. In future versions it will be more flexible.
Screen shots	musilisp.jpg
Official	I have read rules and have abided by them. I am 18 years of age or older. I am not living in Brazil, Quebec, Saudi Arabia, Cuba, Iran, Myanmar (Burma), North Korea, Sudan, or Syria.