The Use of Wave-Table Synthesis

Použití Wave-Table syntézy

Autor: Ing. Vladimír Arnošt, 1. ročník PDS, zpracováno na ÚIVT FEI
Školitel: Doc. Ing. František Zbořil, CSc.

Abstract: This paper describes main methods used during development of a pure software musical synthesizer. The program simulates real synthesizers by the means of Wave-table synthesis. A brief overview of the Wave-table synthesis method is given here. Software can be an inexpensive alternative to special musical equipment if lower sound quality is not an issue.

1 Introduction

The aim of this project was to create a program that would make it possible to replace expensive hardware musical synthesizers. This solution would give its users a chance to improve the computer music (MIDI) playback quality at a substantially smaller cost than if they bought equivalent piece of hardware. Software is also more flexible than hardware and can be upgraded or completely replaced very easily.

2 Principles of Operation

A software synthesizer tries to work in a similar fashion as its hardware counterpart. Its operation can be described as a pipeline of functional units working together to get expected results. Unlike hardware, a program cannot execute different pieces of code in parallel if only one processor is available. The biggest difference is the inability to parallelize the work, so different parts of the algorithm must be executed sequentially.

Multiple simultaneous voices must be calculated at once to reach required level of *polyphony*. Values typically range from 16 to 32 voices. In hardware, all these voices could be theoretically assigned independent synthesis units and thus work in parallel. Software must process all the voices in sequence and then combine the results into an output stream.

Hardware synthesizers are usually independently working devices. They can spend 100% of their time on synthesis and other tasks are handled by other devices. Software usually cannot do the same, because other programs might be executing in parallel and thus only a fraction of CPU power is available to the synthesizer.

2.1 Wave-table Synthesis

Wave-table synthesis (WT synthesis from here on) makes it possible to imitate the sound of real musical instruments with only moderate computational power requirements. The method is based on a table of sound samples recorded from real instruments or artificially calculated. These samples can be played back at various frequencies and the output sound can resemble the original timbre.

Playing back the samples without any modification rarely satisfies any musician so *modulation* was introduced. WT synthesis differs from playing whole samples in the use of

modulation to recreate the characteristic timbres from small sound samples and a set of parameters. The most common modulation methods include:

- pitch shifting, resampling
- amplitude envelopes
- filtering
- periodic changes of pitch and/or volume (vibrato, tremolo)
- sound effects (reverberation, chorusing, etc.)

The project was focused mainly on the problem of *pitch shifting*. For real-time operation the algorithms used to accomplish the task must be as simple as possible. On the other hand their quality and precision must be high enough to avoid audible *artifacts* from appearing on the output. These two requirements are obviously contradictory, so a compromise solution must be found.

2.2 Resampling

Resampling is a process in which the sample rate of a discrete signal is changed. Let f_{s1} be the original sampling frequency and f_{s2} the new one. If $f_{s2} < f_{s1}$, the signal must be band-limited to $f_{s2}/2$ prior to conversion to satisfy *Nyquist sampling theorem* requirements. This should be done by a brick-wall low-pass FIR filter with cut-off frequency of $f_{s2}/2$. Such a filter is very difficult to realize, though, so the requirements must be a bit relaxed. The real filter cut-off frequency is set below $f_{s2}/2$ and the transition band frequency response is expected to be less steep and a bit rippled.

Unfortunately, it is quite difficult to change the sample rate without introducing additional noise and distortion. The best compromise between low processor demands and high quality seems to be simple *linear interpolation* scheme. Despite its far-from-ideal response, it is considerably better than the simplest possible method *sample and hold*. Other methods (e.g. interpolation by higher order curves, oversampling, etc.) seem to be too time consuming to be used in real-time synthesis, but considerable effort has already been spent to create better and yet usable methods.

3 Conclusions

The project has shown that the use of Wave-table synthesis to create a pure software synthesizer software is a perspective way which can lead to very good results. The program still exhibits some problems typical to early research and development phase, but they are being continually solved. It will be necessary to concentrate on the algorithms that form the program core and try to improve them further.

As an average computer power rises, it will be viable to use more sophisticated algorithms for sample interpolation and filtering soon. It seems to be absolutely necessary to have enough experience and working algorithms in order to be prepared for the time when almost any computer sold will be able to bear the load of such software.

References

1. Arnošt, V.: *Software Musical Synthesizer* (*Programový syntetizér hudebních nástrojů*). [diploma thesis], FEECS Technical University of Brno, 1998 (in Czech)
2. Rothacher, F. M.: *Sample-Rate Conversion: Algorithms and VLSI Implementation*. [Ph.D. thesis], Swiss Federal Institute of Technology, Zurich, 1995