Computationally Efficient Music Synthesis – Methods and Sound Design

Jussi Pekonen

May 31, 2007

Outline

- **Introduction**
- **Synthesis Techniques**
- **Synthesizer Design**
  - *Anti-Aliasing Waveform Synthesis using* PolyBLEP *Approach*
- **Sound Design**
  - *Design Rules for Different Timbres*
  - *Parameter Variation in Generation of Sound Variants*
- **Conclusions and Future Work**

Introduction

Computational power of microprocessors increasing – Why bother?

- Power consumption is a major issue in, e.g., mobile devices
  - Limits maximum possible computational power and memory capacity
- Capability to play polyphonic music
  - System limitations even stricter
  - Synthesizer usually controlled with MIDI-format files
- Sound design using applicable synthesis techniques
  - Wide range of different sounds
  - Trade off between sound quality and computational efficiency

SYNTHESIS TECHNIQUES

Practical techniques in computationally efficient music synthesizer?

Table look-up techniques

- Involve reading sample values from memory
- Sampling & wavetable synthesis
- Computationally efficient, memory consumption high, generality good

Spectral modeling techniques

- Model the spectrum of the target signal
- Additive, granular & source-filter (subtractive) synthesis
- Computationally fairly efficient, memory consumption fair, generality fair

Modulation techniques

- Generate the target signal by applying a transformation to a signal
- Amplitude & frequency modulation, waveshaping synthesis
- Computationally efficient, memory consumption extremely low, generality poor

Physical modeling techniques

- Model the physical behavior of an acoustical instrument
- Modal & waveguide synthesis
- Computationally fairly efficient, memory consumption high, generality fair

Which one is the best?

- One technique is not better than some another
- Trade off between computational efficiency and parameter control and memory usage

Practice

- Small number of sinusoidal components – Additive synthesis
- Harmonic spectrum – Source-filter (subtractive) synthesis
- If nothing else can be done – Sampling synthesis
- Additional effects – Amplitude and frequency modulation, waveshaping

The applicable technique ultimately determined by the sound to be imitated!

SYNTHESIZER DESIGN

What components are needed for a synthesizer?

- Oscillators
- Filters
- Control signals
- Effects

The largest problem in the use of classic waveforms in source-filter synthesis

- Sawtooth, rectangular pulse & triangular pulse waves
- Harsh aliasing due to discontinuities

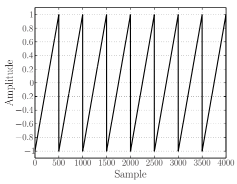
Bandlimited waveform synthesis

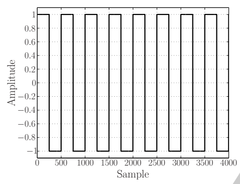
How to obtain a bandlimited classic waveform?

- Strictly bandlimited methods
  - Only frequencies up to Nyquist limit are generated
  - Additive & wavetable techniques, discrete summation formulae
- Quasi-bandlimited methods
  - Some aliasing allowed mainly at high frequencies
  - BLIT & BLEP
- Alias-suppressing methods
  - Some aliasing allowed at all frequencies, but sufficiently suppressed at low and middle frequencies
  - Trivial sampling at high sampling rate, DPW

Quasi-bandlimited waveform synthesis

- Based on filtering of the continuous-time waveform
- In closed form with classic waveforms
- Two approaches
  - Integration of bandlimited impulse train – BLIT
  - Summation of bandlimited step functions – BLEP





PolyBLEP – *Polynomial* BLEP

Quasi-bandlimited waveform synthesis methods manipulate the neighbouring samples of the discontinuity instant

- ♭ BLIT and BLEP traditionally implemented with tables
  - ⇒ Memory consumption!
- ♭ Could it be implemented with closed-form formulas? YES!

Design of a polynomial bandlimited step function

- ♭ Design a symmetric, smoothly piecewise continuous polynomial, which approximates the sinc function near origo
- ♭ Integrate the polynomial, take care of continuities, and scale it to saturate to unity

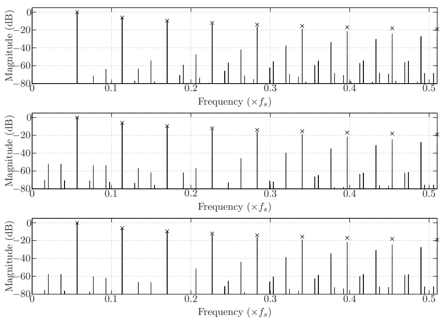
In practice, only the residual, the difference from unit step function, is used in the algorithm

PolyBLEP – *Simple low order approximations*

Simplest approach: manipulate only the previous and next samples

- ♭ First-order approximation
  - ♮ Linear interpolation function
- ♭ Third-order approximation
  - ♮ Spline
    - ♯ Values of the polynomial and its first order derivative at the interval end points match the respective values of the target function
  - ♮ Lagrange polynomial
    - ♯ One of the Lagrange interpolation coefficients
    - ♯ Values at the interval end points match the target function

PolyBLEP – *Comparison of low order approximations*



Sound Design

MIDI compatible system must be capable of produce the sound set defined in the MIDI specification

- General MIDI Level 1 (GM-1) – "The MIDI"
  - 128 melodic sounds, one percussion set with 47 different sounds
- General MIDI Level 2 (GM-2) – Extension to GM-1
  - 128 additional melodic sounds, mainly variants of the GM-1 melodic sound
  - Extends the GM-1 percussion set, defines eight additional sets

MIDI synthesizer with high sound quality requires careful sound design!

Design rules for different timbres

Hints from the type of the sound to be imitated

- String, Pipe, Brass and Reed Instruments
  - Harmonic spectrum → Source-filter synthesis
- Organs, Mallets and Bells
  - Usually only a few components → Additive synthesis
- Electronic Instruments and Synthetic Sounds
  - Implementation varies from a synthesizer to another
- Percussion Instruments
  - Various sound production mechanisms, noise-like spectra
- Sound Effects
  - Very heterogeneous group of sounds

Parameter variation in generation of sound variants

Synthesizers produce many similar sounding timbres, which are still different from each other

- Obtained via parameter variation
- Changing the brightness
  - Shifting the spectral centroid
- Widening of the timbre
  - Increasing the decay rate of the higher components
- Adding power
  - Concentrate the power to the beginning
- Adding reverberation
  - Modulate the amplitude with a LFO

Conclusions

- Computational efficiency must also consider memory consumption and parameter control
- Applicable synthesis techniques limited to additive and source-filter synthesis
  - In special cases sampling synthesis
  - Additional effects with modulation techniques
- Largest issue in generation of classic waveforms for source-filter synthesis
  - Anti-aliasing waveform synthesis with closed form formulas possible
- Sound design based on the type of the timbre
  - Variants with rather simple parameter variations

Future work

Source-filter synthesis the most useful technique

- Analysis methods for the control data
- Evaluation of applicable source signal algorithms
  - Applicable polynomials for PolyBLEP
  - Classic waveforms with inharmonicity control
- More accurate modeling of the tone characteristics
- Evaluation of the method with psychoacoustic modeling and listening tests

Reduction of required delay elements in sound effect algorithms

- Applicability of delay filters
- New algorithms?

References

[1] T. Tolonen, V. Välimäki, and M. Karjalainen, "Evaluation of modern sound synthesis methods," Tech. Rep. 48, Laboratory of Acoustics and Audio Signal Processing, Helsinki University of Technology, Finland, 1998.

[2] V. Välimäki and A. Huovilainen, "Antialiasing oscillators in subtractive synthesis," *IEEE Signal Processing Magazine*, vol. 24, pp. 116–125, March 2007.

Questions, comments?