Audio Synthesis methods
Digital Signal Processing with Computer Music

Christophe Rhodes

Tuesday 16th November 2021
Outline

Introduction

Additive synthesis

Unit generators

Physical modelling

Conclusions
General perspective

- Overview of audio synthesis
  - relate to DSP concepts and techniques
General perspective

• Overview of audio synthesis
  • relate to DSP concepts and techniques
• Audio production must relate to perception
  • how do we hear?
  • psychology of music (discuss with Dr Spiro)
General perspective

- Overview of audio synthesis
  - relate to DSP concepts and techniques
- Audio production must relate to perception
  - how do we hear?
  - psychology of music (discuss with Dr Spiro)
- Building blocks of today’s sonic/music programming environments
  - (discuss with Dr Aaron)
Why building blocks?

• Why synthesise audio at all?
  • (just take the Fourier Transform of what we want and call it a day)
Why building blocks?

- Why synthesise audio at all?
  - (just take the Fourier Transform of what we want and call it a day)

- Why consider audio in terms of frequency?
  - (the eardrum responds to changes in pressure, that’s all)
The inner ear

Basilar membrane cross-section

- not to scale
- unrolled
The inner ear

Basilar membrane cross-section

- not to scale
- unrolled
The inner ear

Basilar membrane cross-section

- not to scale
- unrolled
First try

\[ A_1 \sin f_1 \]

\[ A_2 \sin f_2 \]

\[ A_2 f_2 \]
Second try

\[ A_1 \times f_1 \rightarrow A \sin f_1 \]

\[ A_2 \times f_2 \rightarrow A \sin f_2 \]

\[ + \]

\[ A_1 \sin f_1 + A_2 \sin f_2 \]
Efficiency

How long does it take to compute $\sin(x)$?
Efficiency

How long does it take to compute $\sin(x)$?

• (it depends)
• around 60 cycles for $x$ in [0,1]

• around 180 cycles for $x$ in [0,1000]

• 2.5M-7.5M cycles for 44100 $\sin$ computations

• "only" at most 1000 different oscillators, even on high-end laptops

• (and very few indeed on old computers)
Efficiency

How long does it take to compute \( \sin(x) \)?

- (it depends)
- around 60 cycles for \( x \) in \([0,1]\)
- around 180 for \( x \) in \([0,1000]\)
How long does it take to compute \( \sin(x) \)?

- (it depends)
- around 60 cycles for \( x \) in \([0,1]\)
- around 180 for \( x \) in \([0,1000]\)
- 2.5M-7.5M cycles for 44100 \( \sin \) computations
  - ”only” at most 1000 different oscillators, even on high-end laptops
  - (and very few indeed on old computers)
Efficiency: wavetables

\[ A_1 \times f_1 \]

\[ A_2 \times f_2 \]

\[ AT_{f_1} \]

\[ AT_{f_2} \]
Wavetables

• recover a factor of 10 or so by using table-lookup
• reuse tables for common functions (e.g. \( \sin(x) \))
• SNR depends on interpolation strategy:
  • truncation (no interpolation): SNR 48dB
  • linear: 109dB
Origin of unit generator

Max Mathews and Joan Miller, Music III (1960)

- *Bicycle Built for Two*

A unit generator is

(...) *a small block of computer instructions performing a given operation such as that of an oscillator, an adder or a random noise generator* [...] 

(Mathews and Miller, 1964)

- Combine unit generators into *patches* (or *instruments*)
Example: amplitude modulation

\[ A \times AT_{fm} \]

\[ A_1 \times \]

\[ f_c \]

\[ AT_{fc} \]

\[ f_m \]
Amplitude modulation

\[ S(t) = A_c \cos(2\pi f_c t) \times [1 + I \cos(2\pi f_m t)] \]
Amplitude modulation

Index of modulation $I$: “how much modulation”

$$\cos(A) \cos(B) = \frac{1}{2} \left[ \cos(A + B) + \cos(A - B) \right]$$

$$S(t) = A_c \cos(2\pi f_c t) + \frac{A_c I}{2} \left[ \cos(2\pi(f_c + f_m)t) + \cos(2\pi(f_c - f_m)t) \right]$$
Amplitude modulation

Index of modulation $I$: “how much modulation”

$$\cos(A) \cos(B) = \frac{1}{2} [\cos(A + B) + \cos(A - B)]$$

$$S(t) = A_c \cos(2\pi f_c t) + \frac{A_c I}{2} [\cos(2\pi (f_c + f_m) t) + \cos(2\pi (f_c - f_m) t)]$$

- position of sideband frequencies = $f_c \pm f_m$
- number of non-zero sideband pairs = 2
- relative amplitude of each sideband = $\frac{I}{2}$
- bandwidth = $2f_m$
Example: frequency modulation

\[ D f_m \]

\[ DT_{f_m} \]

\[ A_1 x \]

\[ AT_f \]

after Chowning (1973)
Frequency modulation

\[ S(t) = A \sin \left( 2\pi \int_{t_0}^{t} f(\tau) d\tau \right) \]

\[ f(t) = f_c + D \sin(2\pi f_m t) \]

\[ S(t) = A \sin \left( 2\pi \int_{t_0}^{t} f(\tau) d\tau \right) \]

\[ S(t) = A \sin \left( 2\pi f_c t + \frac{D}{2\pi f_m} \sin(2\pi f_m t) \right) \]
Frequency modulation

Write \( D = 2\pi I f_m \)

\[
S(t) = A \sin(2\pi f_c t + I \sin(2\pi f_m t))
\]
Frequency modulation

\[ S(t) = A \sin(2\pi f_c t + I \sin(2\pi f_m t)) \]

- position of sideband frequencies = \( f_c \pm k f_m \)
- number of non-zero sideband pairs \( \sim (I + 1) \)
- relative amplitude of each sideband \( \sim J_k(I) \)
- bandwidth \( \sim 2(D + f_m) = 2f_m(I + 1) \)
Example: sampled instruments

\[ A_1 \times f \rightarrow AT_f \]
Sampled instruments

• record target sound
• identify steady-state sample
• use envelope to generate transients
  • e.g. ADSR: Attack, Decay, Sustain, Release
  • (arbitrary envelopes are available)
Example: granular synthesis
Waveguide

A waveguide is

\[ \frac{\partial^2 y}{\partial x^2} = \frac{1}{c^2} \frac{\partial^2 y}{\partial t^2} \]

\[ y(x, t) = f(x + ct) + g(x - ct) \]

*a computational model of medium along which waves travel*

(“Digital Synthesis Waveguide”; cf. waveguides used in transmission engineering)
Waveguide

excitation

delay

scatterings

delay

delay

filter

output
Example: Karplus-Strong
Example: Karplus-Strong

- typical filter: two-sample average
Karplus-Strong

• Why does Karplus-Strong even sound pitched?
Karplus-Strong

• Why does Karplus-Strong even sound pitched?
Karplus-Strong

- why initialize buffer with white noise?
- how to decouple period from decay time?
- (more interesting sounds using the same setup?)
Waveguides and unit generators

Can be made compatible:

- switch (noise vs feedback) based on zero-crossing of trigger signal;
- delay modifiable (up to some maximum);
- parameterise filters;

Include physical modelling (including impossible instruments) in unit generator paradigm
Perspectives

• historical:
  • how to get “rich” sounds cheaply?
  • how to mimic/simulate/model physical processes?
  • how to explore possible sound worlds?
Perspectives

• historical:
  • how to get “rich” sounds cheaply?
  • how to mimic/simulate/model physical processes?
  • how to explore possible sound worlds?

• musical:
  • how can we use digital signals to produce music?
  • why do we find some sounds pleasing?
Perspectives

• historical:
  • how to get “rich” sounds cheaply?
  • how to mimic/simulate/model physical processes?
  • how to explore possible sound worlds?

• musical:
  • how can we use digital signals to produce music?
  • why do we find some sounds pleasing?
  • who is “we” anyway? (and what is “computer music”?)
Further Reading

- Hermann Helmholtz (tr. Ellis), *On the sensations of tone*, Longmans (1885)
- Max Mathews and Joan Miller, *Music IV programmer’s manual*, Bell Labs (1964)
- Scott Wilson, David Cottle and Nick Collins (eds.), *The SuperCollider Book*, MIT (2011)