Digital Signal Processing for Music

Part 18: Source Generation

Andrew Beck

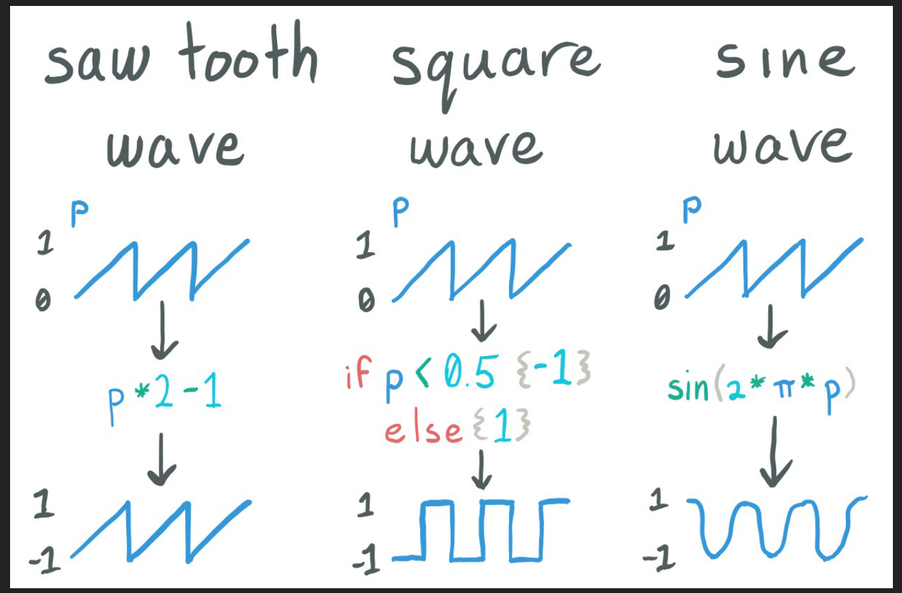
Introduction

- **Tone Generation**
  - *Direct Generation*
    - Naive function generator
    - Additive synthesis
    - Oversampled naive
    - BLEP / PolyBLEP
  - *Wavetable*
    - Single cycle
    - Morphing
    - Mixed direct & wavetable
  - *Specific techniques*
    - ~~FM synthesis~~
    - Karpluss-strong
- **Source playback**
  - Sample playback
  - Granular synthesis

Naive Function Generation

Process

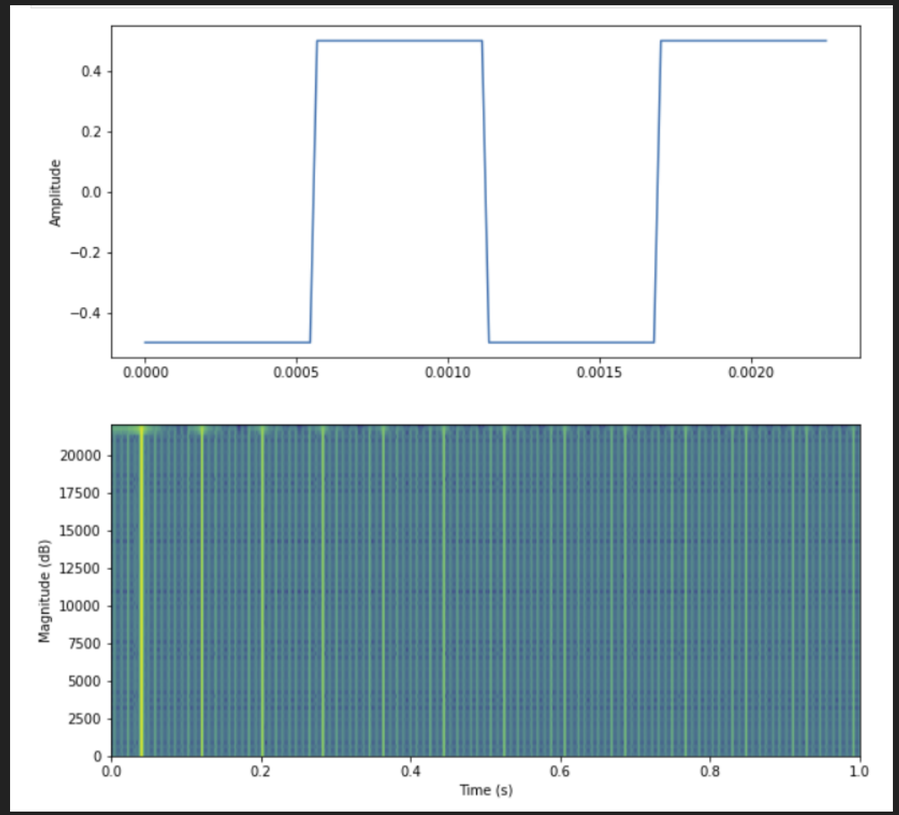
- Keep track of phase ϕ
- Transform to desired function



Naive downsides

Simple to implement but:

- Potentially expensive
- Causes aliasing



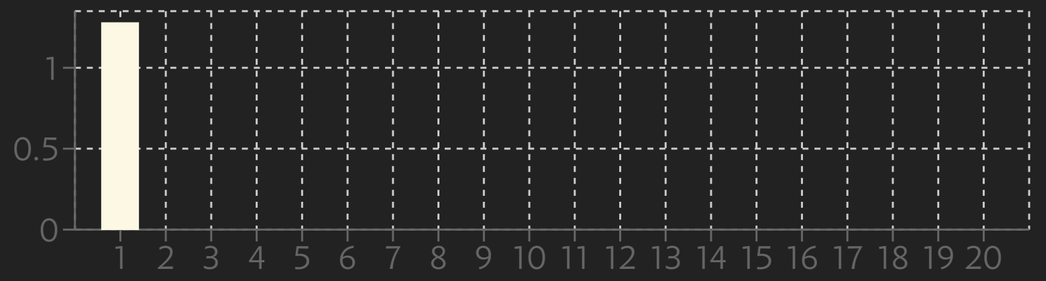
Additive Synthesis

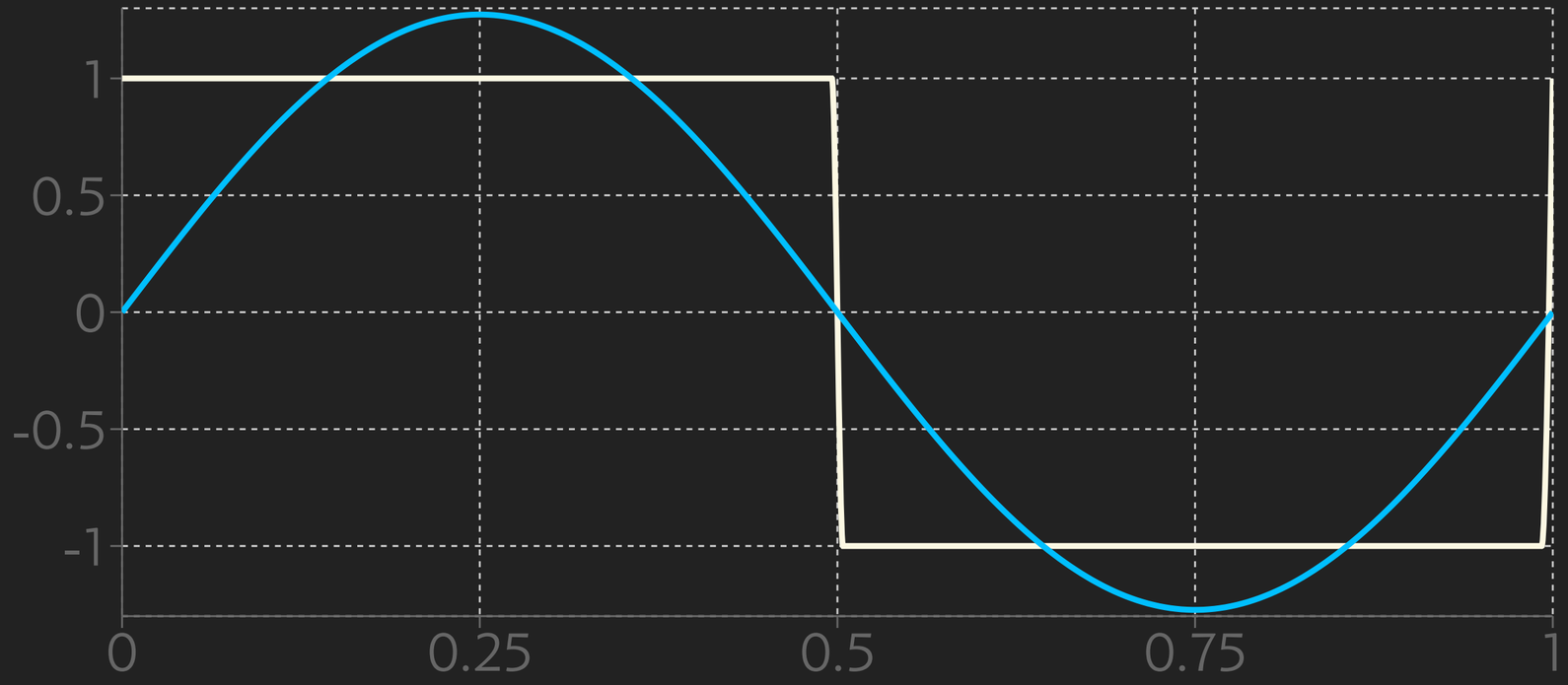
Build function from list of sinusodials

Num Harmonics

1

Harmonic Amplitudes

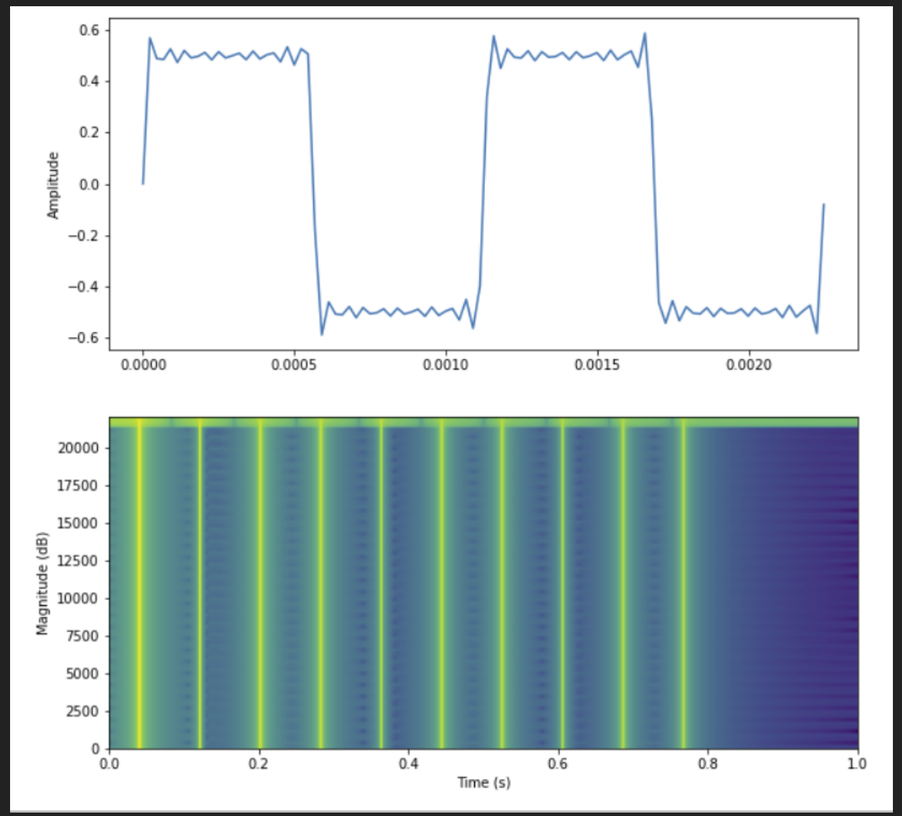




Additive downsides

Offers tight control over spectral domain but:

- Expensive
- Can still alias



Oversampling Naive

Process

- Generate naive function at higher sample rate
- Apply anti-aliasing filter
- Downsample

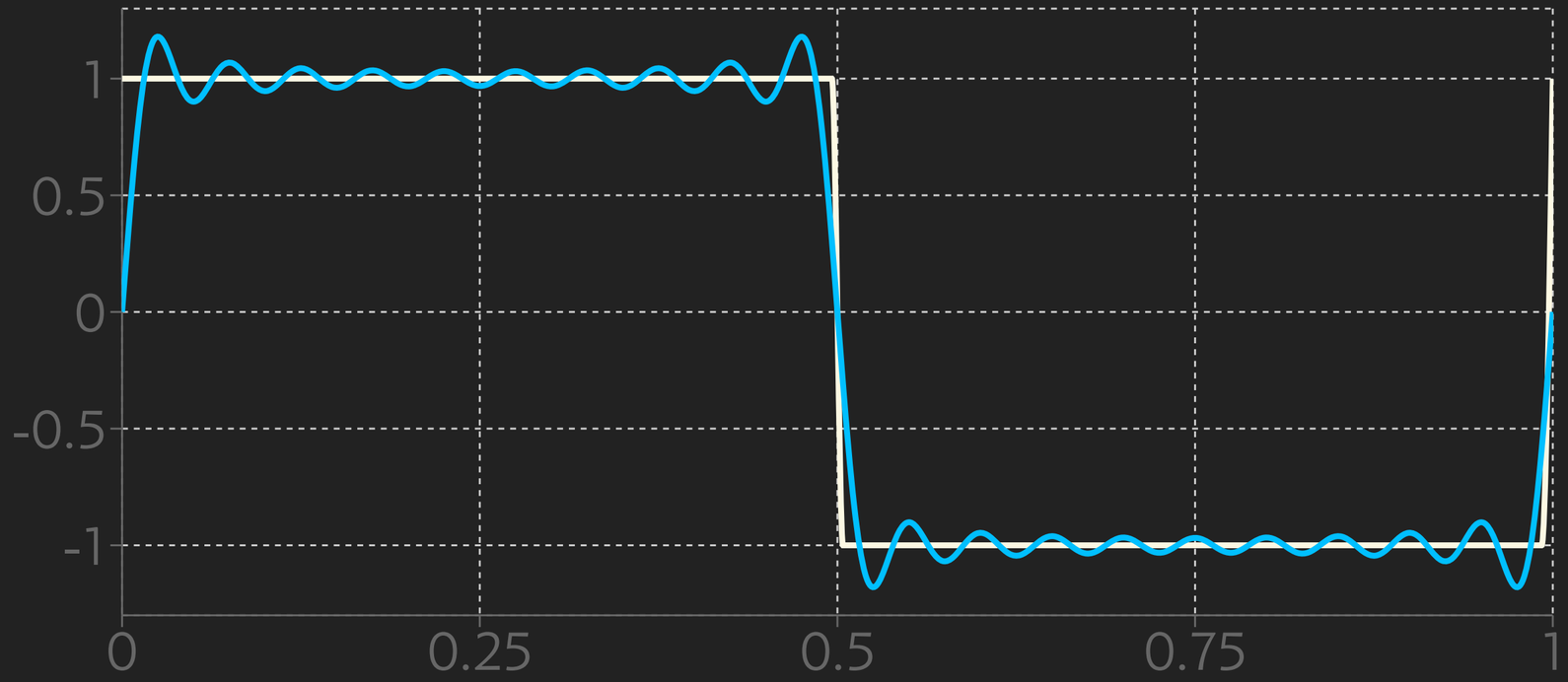
Flexible approach but:

- Expensive
- Aliasing artifacts still exist

BLIT/BLEP/PolyBLEP

Observations:

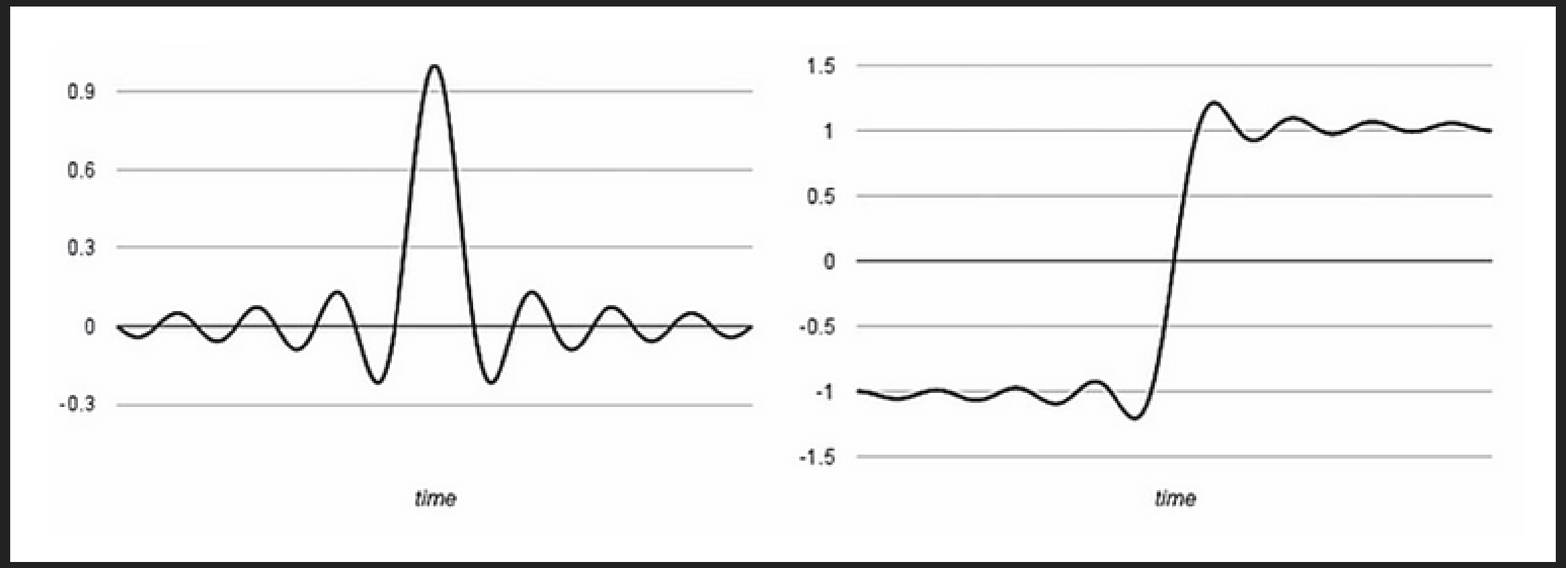
- Aliasing is caused by sharp jumps (IE high frequencies)
- Smoothing (lowpassing) sharp jumps still causes aliasing
- True anti-aliased signals have ripples



BLEP

Solution: Add ripple across edges

- **B**and **L**imited st**EP** function
- Adding in idealized lowpass filter of step function
- Integrate sinc function and add to naive signal



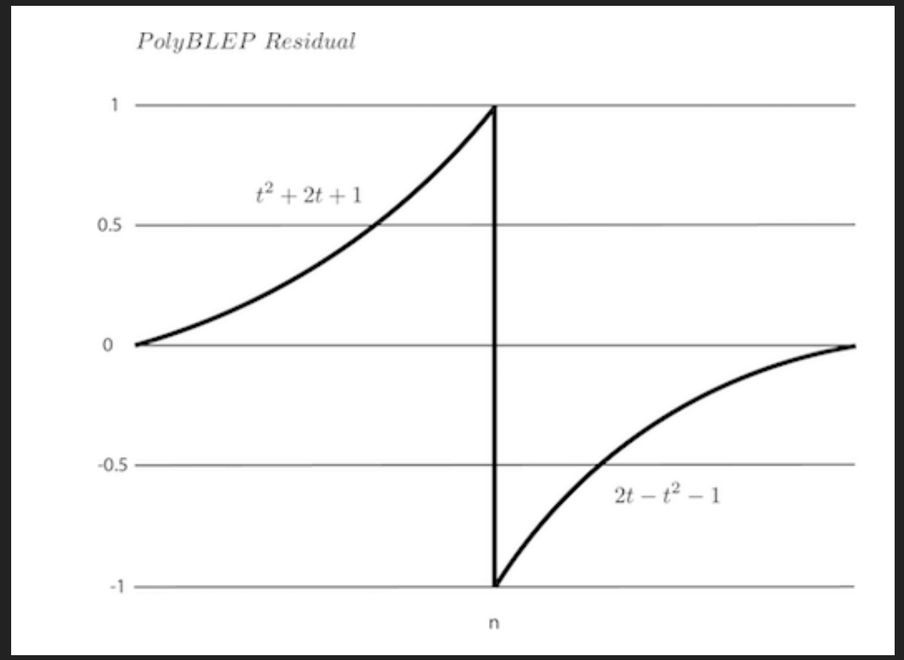
PolyBLEP

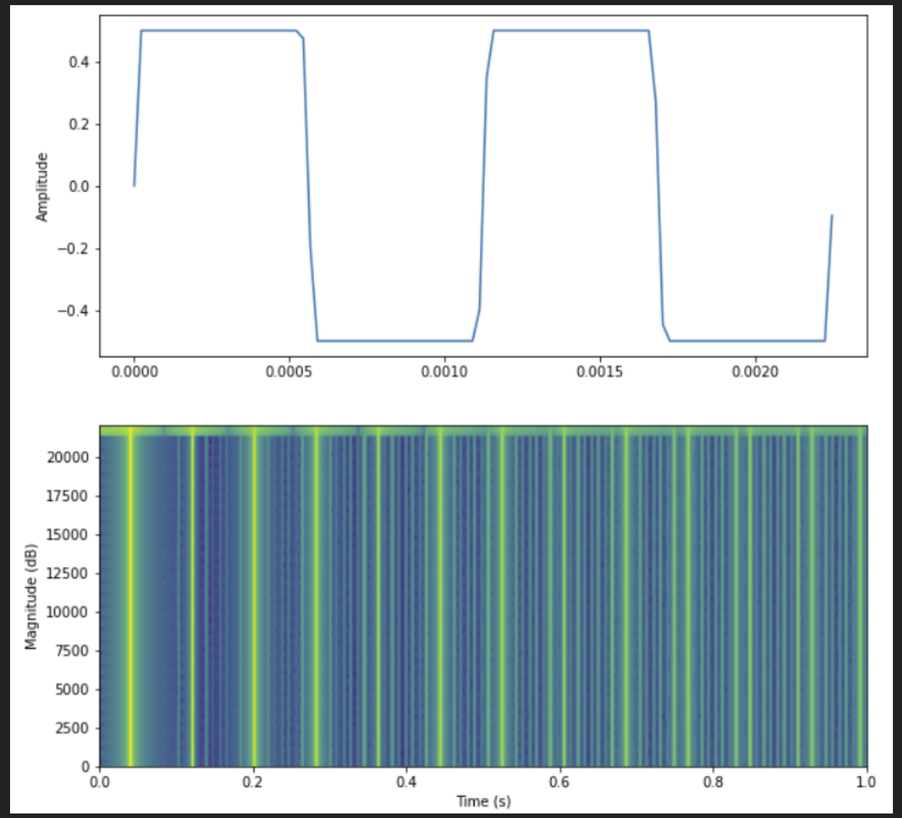
Special case of BLEP, smoothing discontinuity

Where t is normalized distance to the discontinuity (typically use one sample)

$$polyblep(t) = \begin{cases} 0, & t < -1 \\ \frac{t^2}{2} + t + \frac{1}{2}, & -1 \leq t \leq 0 \\ t - \frac{t^2}{2} + \frac{1}{2}, & 0 < t \leq 1 \\ 1 & t > 1 \end{cases}$$

PolyBLEP





- Not suitable for modulating control values
- Often good comprimise between speed and quality

Wavetable Synthesis

- **Problem**: Some functions (particularly sine) can be expensive
- **Solution**: Render into a single cycle buffer and use ϕ to index into wavetable

```
phi = phasor(hz, sr)
out = wavetable[int64(phi * (len(wavetable) - 1))]
```

Wavetable Morphing

- **Time Domain**: Interpolating between two wavetables
- **Frequency Domain**: Interpolate complex results of FFT, IFFT for playback

Combining Wavetable and Direct

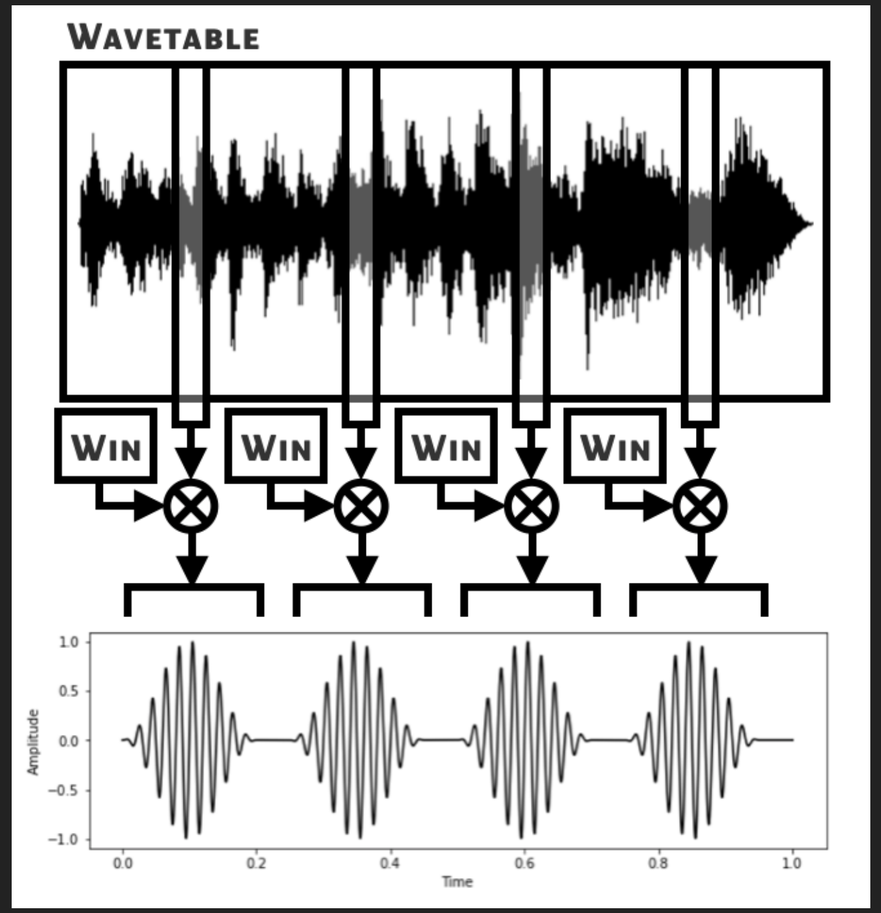
- Example: Casio keyboards

Sample Playback

Simply a large wavetable that doesn't repeat

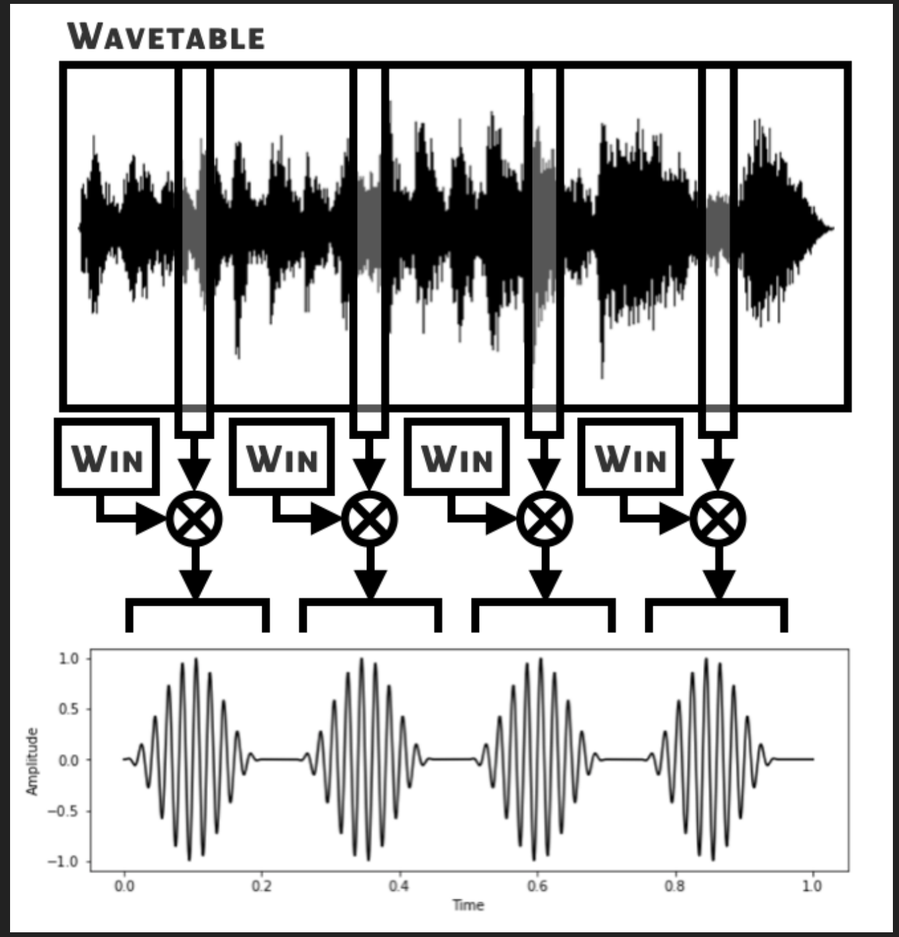
Granular Synthesis

Indexing into small segments (aka Grains) from a wavetable



Granular Synth Properties

- Grain Size
- Hop Size
  - Periodic
  - Stochastic
  - Random
- Grain Pitch



Karpluss-Strong

Process

- Fill single cycle buffer with random noise
- As you index, run low pass filter (averaging filter) over buffer, either in time with playback index or independently
- Periodicity of buffer creates tone
- Simulates noisey impulse and decay

Summary

- No "one-size-fits-all" solution to function generation
- Overlap between different techniques