




# INTRODUCTION TO COMPUTER MUSIC SAMPLING SYNTHESIS AND FILTERS

---

Roger B. Dannenberg  
Professor of Computer Science, Art, and Music

ICM Week 7 Copyright © 2002-2013 by Roger B. Dannenberg **1**



# SAMPLING SYNTHESIS

---

Synthesis from pre-recorded sounds

ICM Week 7 Copyright © 2002-2013 by Roger B. Dannenberg **2**

## Sampling Synthesis



- FM and other techniques were “big” when computation and memory were expensive.
- But FM never produced satisfactory simulations of acoustic instruments.
- Sampling is a simple concept:  
Record actual sounds and play them back!
- Advantages:
  - Easily captured sounds
  - Works with noise, tones, anything

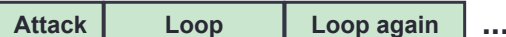
## How it works



- Base case: store a sound and play it back
- Desired parameters:
  - Duration
  - Pitch
  - Amplitude
  - Vibrato
  - Brightness

## Controlling Duration

- Repeat a portion of sound
  - Could be a single period
  - Could be much longer segment
  - Finding good “loop points” is tricky
- Use an envelope and multiply to decay at end

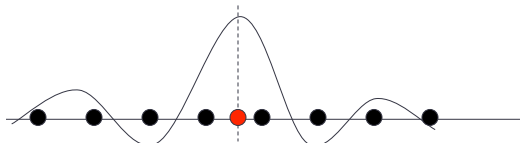


## Finding Loop Points

- Often done "by ear" interactively
- Tones are not often truly periodic
  - changes in amplitude, frequency, noise, spectrum
  - but you can pick where to start looping (a search problem)
  - you can cross-fade to make a smoother transition
- Periods are not always integer number of samples
  - At  $G_4$  (391.995 Hz), 1 period =  $44100/f_0 = 112.501$  samples
    - Upper harmonics are of course much shorter, so cutting out even half a sample is significant.
  - You can loop over multiple periods.
  - You can resample (interpolate) to get an integral length

## Controlling Pitch

- Resample to speed up and slow down
- Usually over limited range (2 to 12 semitones)
- Linear interpolation can cause aliasing
- Good samplers use multipoint interpolation using a weighted sum; number of points is considered trade secret



ICM Week 7

Copyright © 2002-2013 by Roger B. Dannenberg

7

## Controlling Amplitude

- Simple multiply
- Or, cross-fade between loud/soft samples
  - Cross-fades risk phase cancellation
- Can apply low-pass filter to make softer sounds less bright

ICM Week 7

Copyright © 2002-2013 by Roger B. Dannenberg

8

## Other Parameters, Modifications

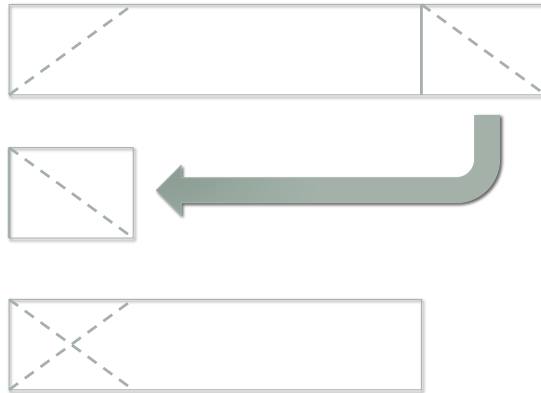
- Filters for various effects
- Frequency and Amplitude vibrato is easy to add
- Reverb, chorus, other effects
- Starting sample at offsets to emphasize/deemphasize attack transients
- Notice that these are all *synthesis* techniques

## Problems with Sampling Synthesis

- Most of the interesting sound quality is “frozen” in the samples
- Strings and woodwinds are controlled by bowing and blowing, not so much by the passage of time – bad model
- Samples can take lots of space: modern libraries take gigabytes

## Examples

- See `sampling.sal`

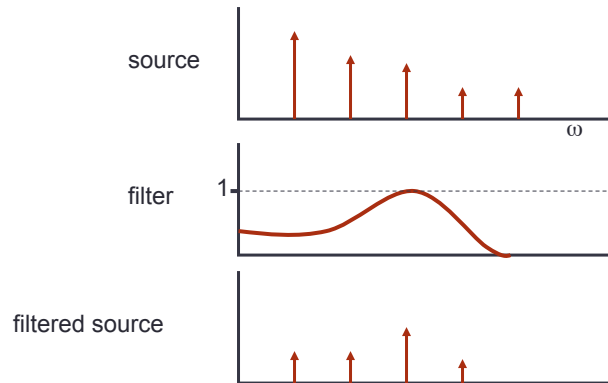


## FILTERS

Boosting and attenuating certain frequencies

## Filters

- What is filtering?



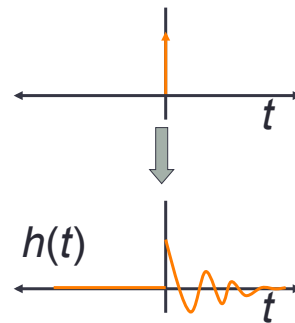
ICM Week 7

Copyright © 2002-2013 by Roger B. Dannenberg

13

## Some Intuition

- Suppose we know what a filter does to an impulse:



- This is called the *impulse response*  $h(t)$  of the filter

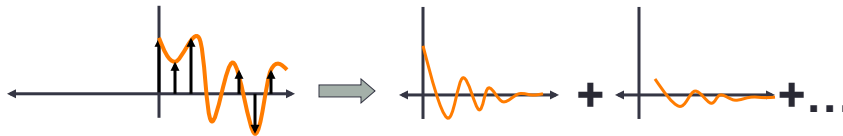
ICM Week 7

Copyright © 2002-2013 by Roger B. Dannenberg

14

## Superposition Principle

- Most filters are *linear time invariant* which implies that the superposition principle holds:  
Response(A+B) = Response(A) + Response(B)
- Using superposition principle, we can sum up impulse responses to get response to signal



See: <http://www.jhu.edu/signals/convolve/>  
And <http://ipsa.swarthmore.edu/Convolution/Convolution3.html>

## Convolution

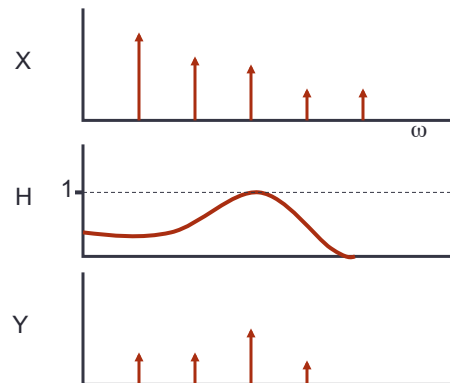
$$y(t) = \int x(\tau)h(t - \tau)d\tau$$

- This expresses the infinite superposition of  $h$  scaled by the input signal  $x$ .
- This is the *convolution* of  $x$  and  $h$ :  $y = x * h$
- Convolution in the time domain is multiplication in the frequency domain:  $Y = X \times H$



## Back to where we started...

Multiplication is easier to think about, so that's why we like to describe filters as operating in the frequency domain



ICM Week 7

Copyright © 2002-2013 by Roger B. Dannenberg

17

## What about phase?

- Yes, filters can change phase
- In fact an all-pass filter has an amplitude response of 1 (unity gain at all frequencies)
- For the complete story, represent  $H$  as complex...

ICM Week 7

Copyright © 2002-2013 by Roger B. Dannenberg

18

## Complex Frequency Response

$$H(\omega) = U(\omega) + iV(\omega)$$

$$|H(\omega)| = \sqrt{U^2(\omega) + V^2(\omega)}$$


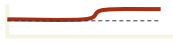

$$\angle H(\omega) = \arctan(V(\omega)/U(\omega))$$

## Filters in Nyquist

- LP – low-pass
- HP – high-pass
- RESON – resonance
- ARESON – complement



## More Filters

- LOWPASS2, HIGHPASS2, BANDPASS2, NOTCH2, ALLPASS2– biquad filter variants
- EQ-LOWSHELF 
- EQ-HIGHSHELF 
- EQ-BAND 
- LOWPASS[4,6,8], HIGHPASS[4,6,8]

## Time-Varying Filters

- Time-varying filters are based on time invariant filters; simply change parameters
- Filter equation parameters are often expensive to compute, so change parameters at control rate
- Parameter changes can insert energy into filter system, and interpolated parameters can create instabilities – numerically difficult

## Examples

```
play reson(buzz(20, c3, const(0)),  
          pwlv(100, 0.5, 3000, 1, 100),  
          1000, 1) ~ 5
```



```
play reson(buzz(20, c3, const(0)),  
          pwlv(100, 0.5, 3000, 1, 100),  
          200, 1) ~ 5
```



```
play lp(buzz(20, c3, const(0)),  
       pwev(20, 0.5, 3000, 1, 100)) ~ 5
```

