INTRODUCTION TO COMPUTER MUSIC
SPECTRAL PROCESSING

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

Review of Fourier Transform

Real part:

$$R(\omega) = \int_{-\infty}^{\infty} f(t)\cos\omega t\, dt$$

Imaginary part:

$$X(\omega) = -\int_{-\infty}^{\infty} f(t)\sin\omega t\, dt$$

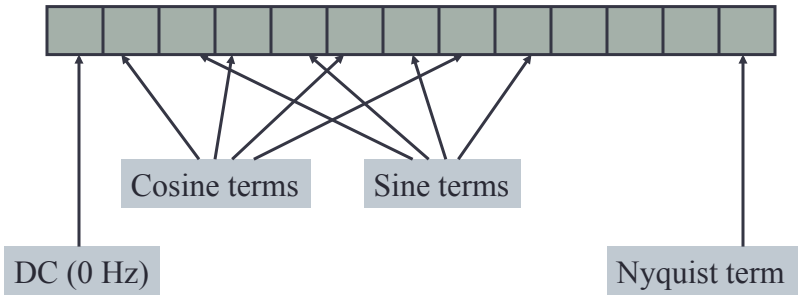
Discrete Fourier Transform

$$R_k = \sum_{i=0}^{N-1} x_i \cos(2\pi ki / N)$$

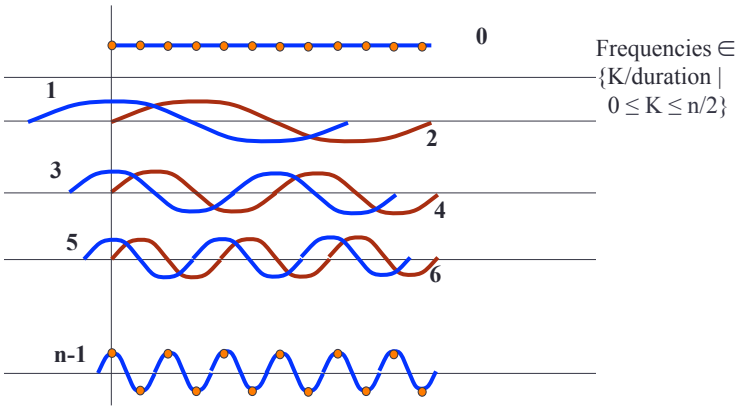
$$X_k = -\sum_{i=0}^{N-1} x_i \sin(2\pi ki / N)$$

Computing Spectra in Nyquist

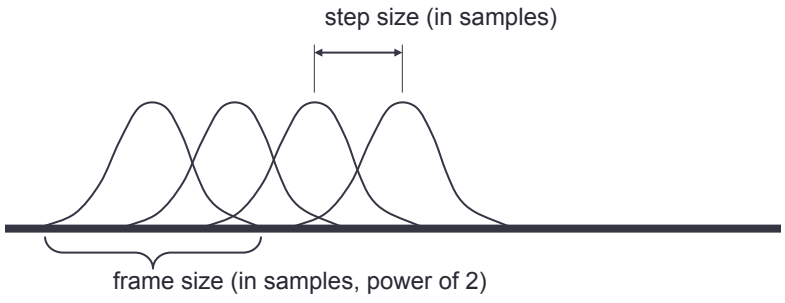
- Representation: spectra appear as floating point arrays.
- (More detail in Week 5 slides)



What does the array mean?

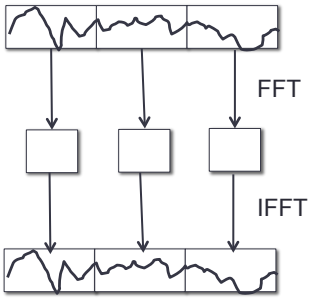


Windows and Step Size



Perfect Reconstruction?

- Converting to frequency domain and back opens up many possibilities. Can we do this without loss?
- FFT can be inverted: IFFT
- Simple but flawed approach:
  - Square windows, no overlap

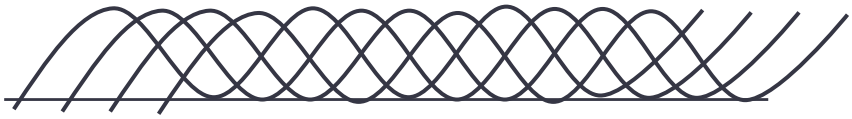


- What about windowing?

Perfect Reconstruction? (2)

- There are many overlapping windows that sum to one:

- But windows are applied twice!
  - Window ➔ FFT ➔ (alter signal) ➔ IFFT ➔ Window

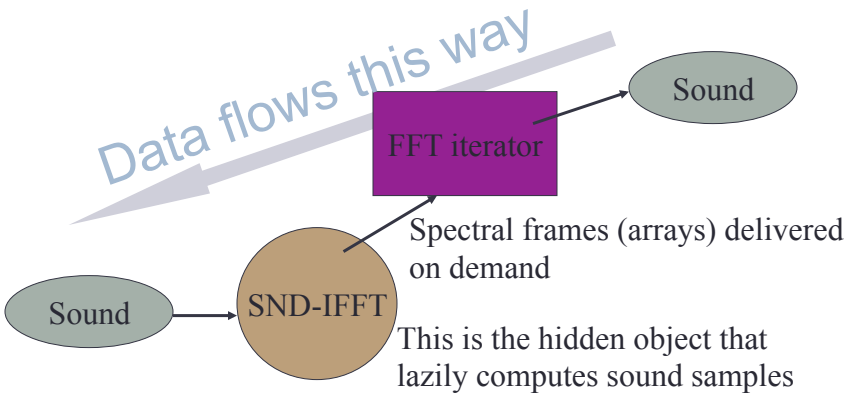


- Raised $\cos^2$ with 25% overlap sums to one!

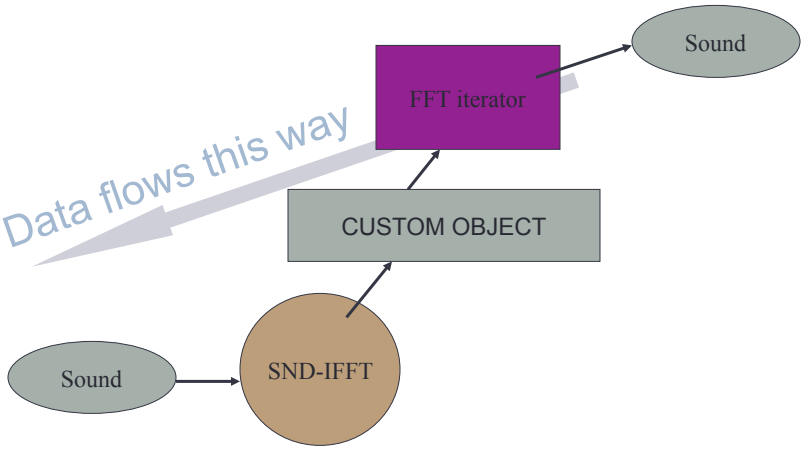
SPECTRAL PROCESSING

Using SAL to operate on spectra

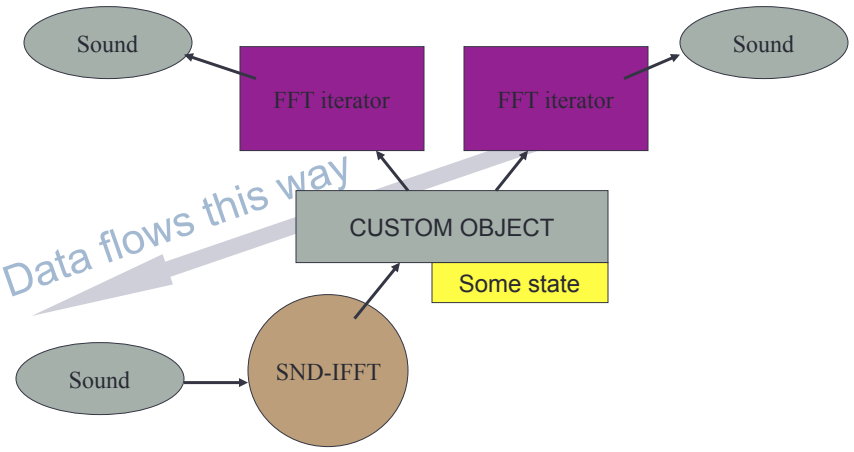
From Sound to Spectra



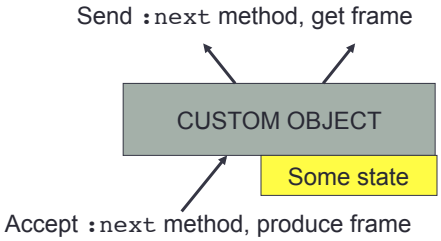
OOP vs SAL



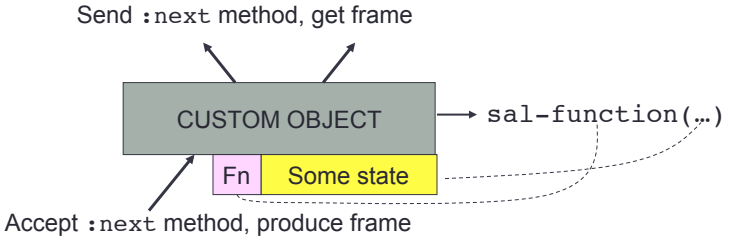
OOP vs SAL



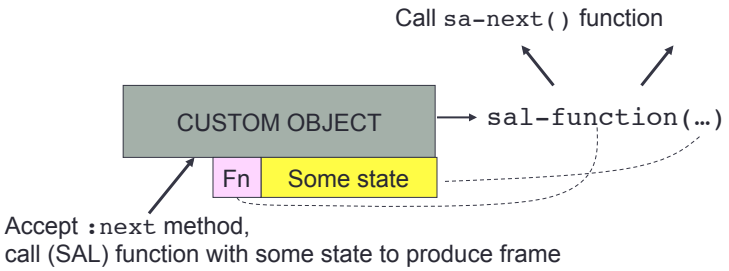
The Object Behavior We Need



How To Get Object Behavior From SAL



How To Get Object Behavior From SAL (2)



Template for Spectral Processing (1)

```
set sa = sa-init(input: "./rpd-cello.wav",
                 fft-dur: 4096 / 44100.0,
                 skip-period: 512 / 44100.0,
                 window: :hann)
```


Template for Spectral Processing (2)

```
set sp = sp-init(sa, quote(processing-fn), 0, 0)
```

Spectral processing object

Spectral analysis object

SAL Function

Initial State

```
function processing-fn(sa, frame, p1, p2)
  begin
    ... Process frame here ...
    set frame[0] = 0.0 ; simple example: remove DC
    return list(frame, f(p1), g(p2)) ; state change
  end

play sp-to-sound(sp)
```


Simple analysis/synthesis examples

- See spectral-process.sal
  - Note: requires spectral-process.lsp and spectral-analysis.lsp as well.

Cross-Synthesis, Morphs, etc.

- In general, combine features from two sounds
- Common approach: separate sounds into *excitation* and *filter* parts.
- Combine filter of one signal with excitation of the other, e.g.
  - Vocal tract filter applied to noise, orchestra, etc.
  - Cello body applied to woodwind sound
- See spectral-process.sal