SPECTRAL PROCESSING
Using SAL to operate on spectra

From Sound to Spectra

Data flows this way

FFT iterator
Spectral frames (arrays) delivered on demand
This is the hidden object that lazily computes sound samples
OOP vs SAL

Data flows this way

CUSTOM OBJECT

SND-IFFT

FFT iterator

Sound

Sound

OOP vs SAL

Data flows this way

CUSTOM OBJECT

SND-IFFT

FFT iterator

FFT iterator

Some state

Sound

Sound

Sound
The Object Behavior We Need

Send: next method, get frame

CUSTOM OBJECT

Accept: next method, produce frame

Some state

How To Get Object Behavior From SAL

Send: next method, get frame

CUSTOM OBJECT

sal-function(...)

Accept: next method, produce frame

Some state
How To Get Object Behavior From SAL (2)

Call `sa-next()` function

CUSTOM OBJECT

Accept :next method, call (SAL) function with some state to produce frame

Template for Spectral Processing (1)

```plaintext
set sa = sa-init(input: "./rpdcello.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)

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