SCORE PROCESSING

Lots of functions to manipulate scores

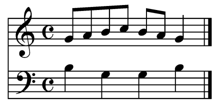
Score Processing Functions

- score-shift
- score-transpose
- score-sustain
- score-voice
- score-merge
- score-append
- score-select
- score-filter-length
- score-stretch-to-length
- score-filter-overlap
- score-adjacent-events
- score-sort
- score-repeat
- score-index-of
- score-last-index-of
- score-randomize-start
- score-read-smf
- score-write-smf

score-sort

- Score events *must* be sorted in order of increasing start times

```
exec score-play(score-sort(
  {{0.0 0.5 {plucked-string pitch: 67 vel:  90 cutoff: 4000}}
   {0.5 0.5 {plucked-string pitch: 69 vel:  95 cutoff: 5000}}
   {1.0 0.5 {plucked-string pitch: 71 vel: 100 cutoff: 6000}}
   {1.5 0.5 {plucked-string pitch: 72 vel: 105 cutoff: 7000}}
   {2.0 0.5 {plucked-string pitch: 71 vel: 100 cutoff: 6000}}
   {2.5 0.5 {plucked-string pitch: 69 vel:  95 cutoff: 5000}}
   {3.0 1.0 {plucked-string pitch: 67 vel:  90 cutoff: 4000}}
   {0.0 1.0 {note pitch: 59 vel: 100}}
   {1.0 1.0 {note pitch: 55 vel: 100}}
   {2.0 1.0 {note pitch: 55 vel: 100}}
   {3.0 1.0 {note pitch: 59 vel: 100}}}}))
```



score-shift

- add 3 seconds to all start times

```
print score-shift(my-score, 3.0)
```

- insert 3s rest at time 10

```
print score-shift(my-score, 3.0,
                  from-time: 10)
```

score-transpose

- Transpose pitch up one octave:

```
print score-transpose(my-score, keyword(pitch), 12)
```

- Increase cutoff freq. by 1000:

```
print score-transpose(my-score, keyword(cutoff),
                                          1000)
```

- Wrong:

```
print score-transpose(my-score, pitch:, 12)
print score-transpose(my-score, quote(pitch:), 12)
```

- OK: `print score-transpose(my-score, :pitch, 12)`

score-sustain

- Increase durations by 25% in the time interval from 1 to 3 seconds

```
print score-sustain(my-score, 1.25,
                          from-time: 1, to-time: 3)
```

score-voice

- Turn plucked-string into note and note into plucked-string

```
print score-voice(my-score,
                  {{note plucked-string}
                   {plucked-string note}})
```

score-merge

- Double every note an octave higher

```
print score-merge(my-score,
          score-transpose(my-score,
                         keyword(pitch), 12))
```

- Make my-score with 2 echoes

```
print score-merge(my-score,
                  score-shift(my-score, 0.1),
                  score-shift(my-score, 0.2))
```

score-append

- Play my-score as is, then transposed up 1 step, then up another step

```
print score-append(my-score,
                   score-transpose(my-score,
                                   keyword(pitch), 2),
                   score-transpose(my-score,
                                   keyword(pitch), 4))
```

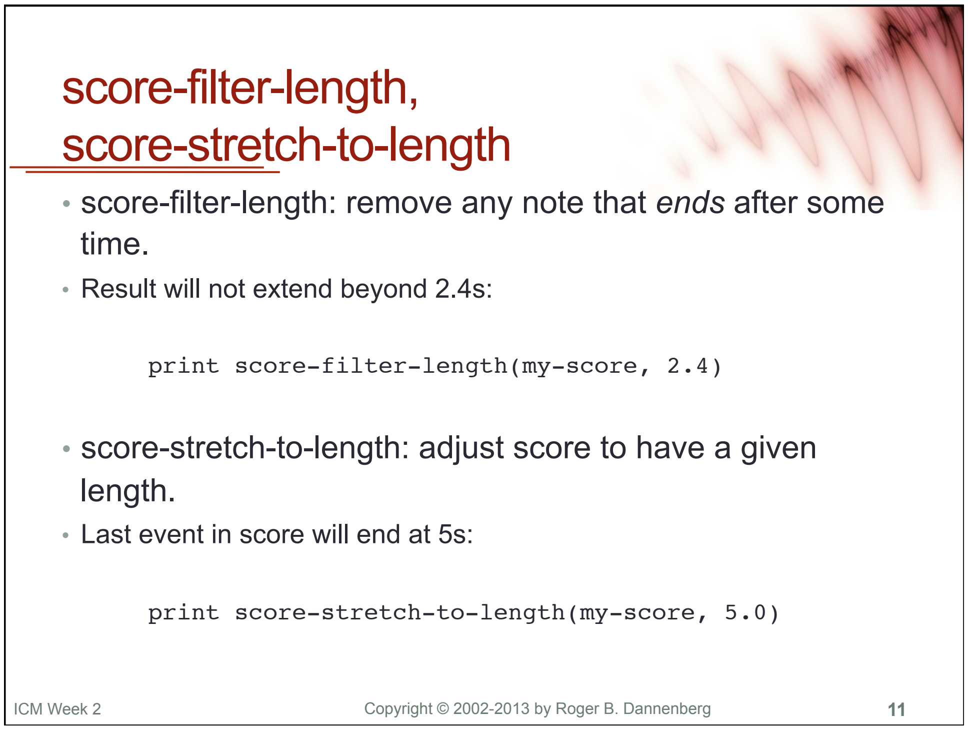
score-select

- A predicate that returns true when pitch is less than 70

```
define function not-very-high(time, dur, expr)
  return expr-get-attr(expr, keyword(pitch), 100) < 70
```

- Select all notes with pitch < 70 and time >= 2

```
print score-select(my-score, quote(not-very-high),
                   from-time: 2)
```



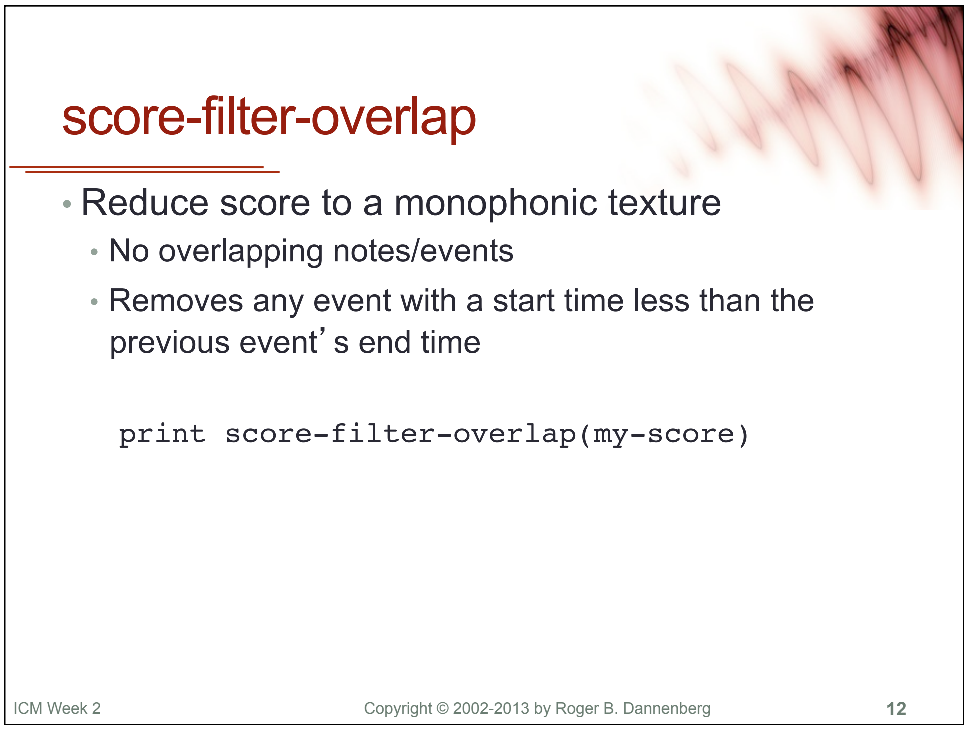
score-filter-length, score-stretch-to-length

- score-filter-length: remove any note that *ends* after some time.
- Result will not extend beyond 2.4s:

```
print score-filter-length(my-score, 2.4)
```

- score-stretch-to-length: adjust score to have a given length.
- Last event in score will end at 5s:

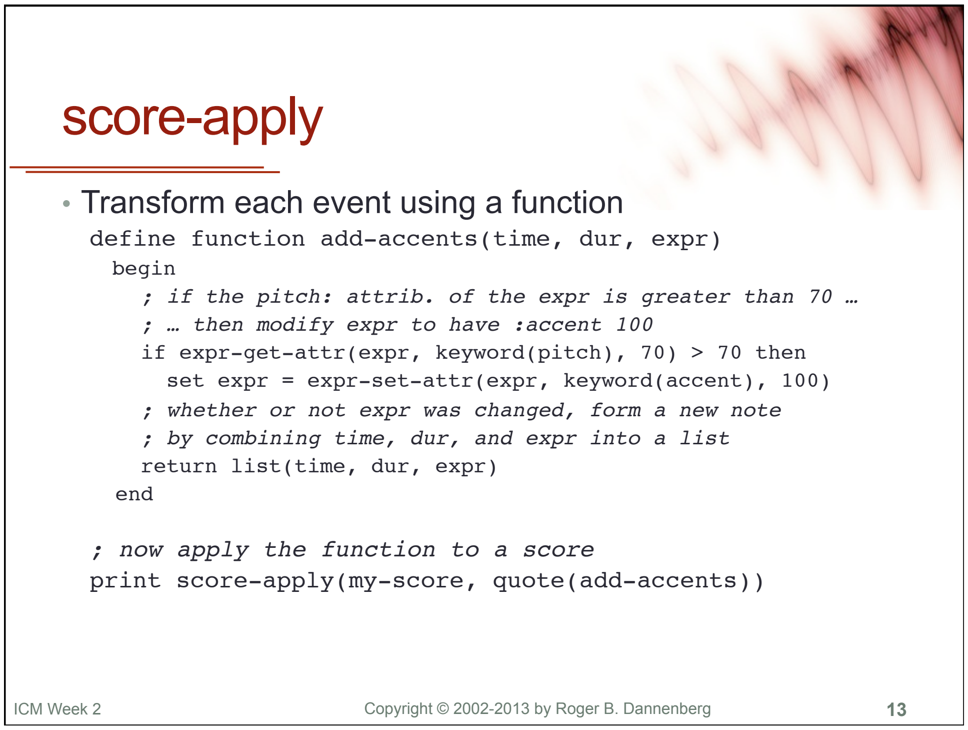
```
print score-stretch-to-length(my-score, 5.0)
```



score-filter-overlap

- Reduce score to a monophonic texture
  - No overlapping notes/events
  - Removes any event with a start time less than the previous event' s end time

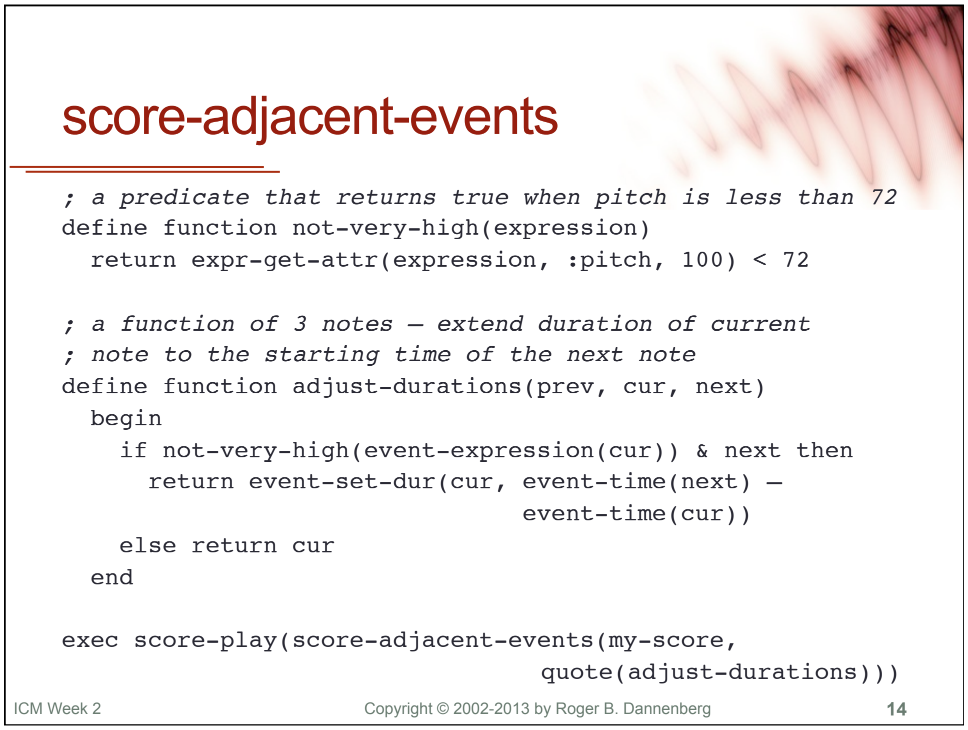
```
print score-filter-overlap(my-score)
```



score-apply

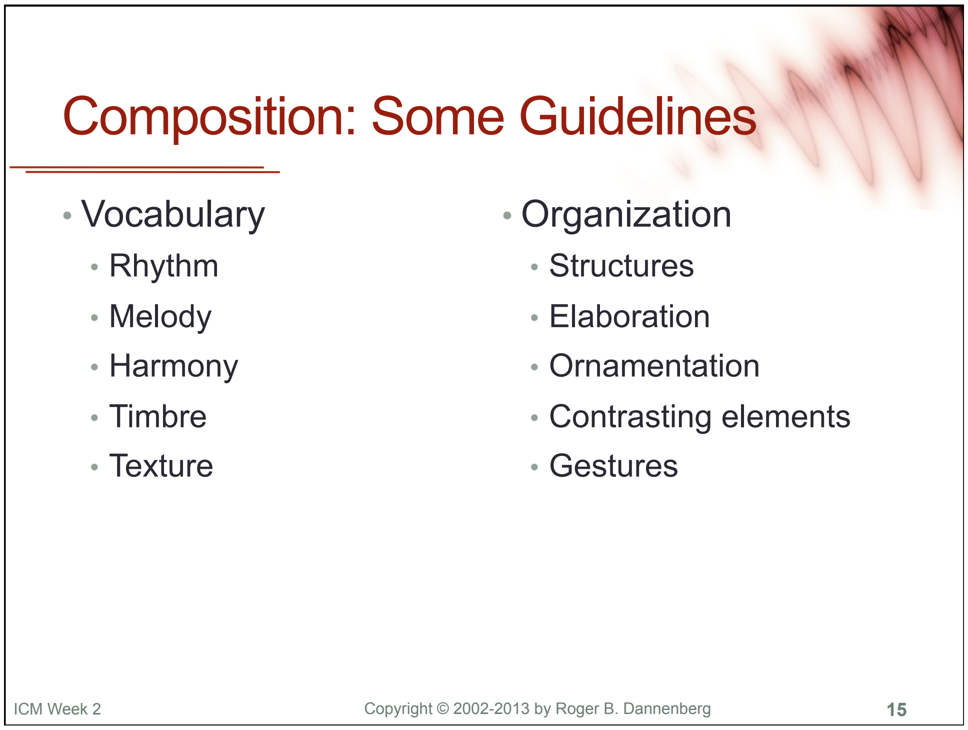
- Transform each event using a function

```
define function add-accents(time, dur, expr)
  begin
    ; if the pitch: attrib. of the expr is greater than 70 …
    ; … then modify expr to have :accent 100
    if expr-get-attr(expr, keyword(pitch), 70) > 70 then
      set expr = expr-set-attr(expr, keyword(accent), 100)
    ; whether or not expr was changed, form a new note
    ; by combining time, dur, and expr into a list
    return list(time, dur, expr)
  end

; now apply the function to a score
print score-apply(my-score, quote(add-accents))
```



score-adjacent-events

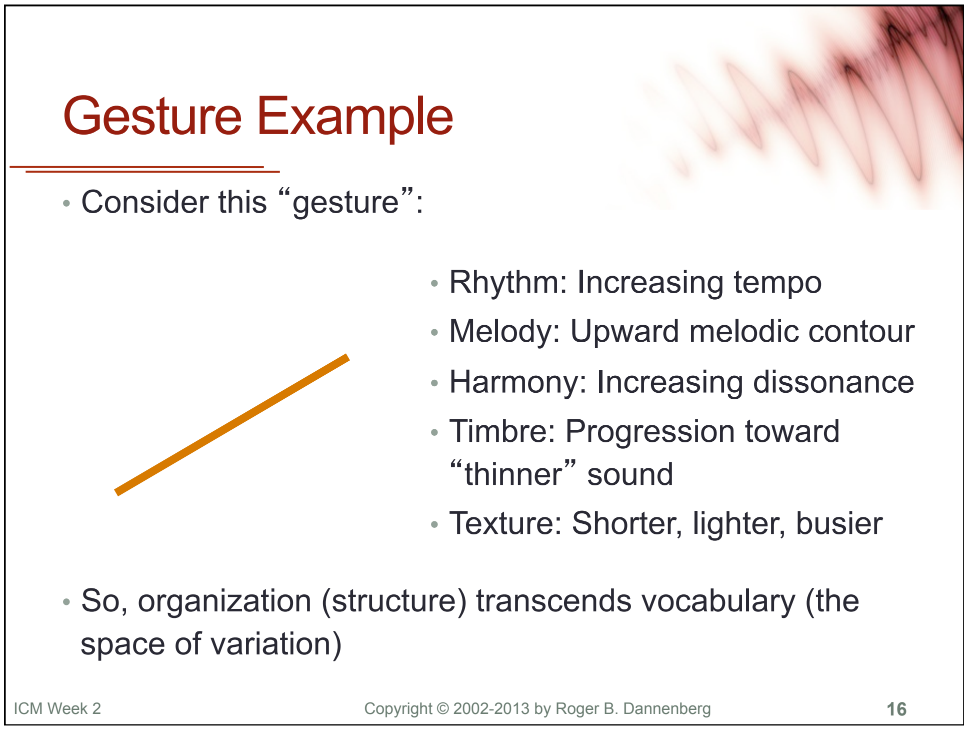
```
; a predicate that returns true when pitch is less than 72
define function not-very-high(expression)
  return expr-get-attr(expression, :pitch, 100) < 72

; a function of 3 notes – extend duration of current
; note to the starting time of the next note
define function adjust-durations(prev, cur, next)
  begin
    if not-very-high(event-expression(cur)) & next then
      return event-set-dur(cur, event-time(next) –
                                event-time(cur))
    else return cur
  end

exec score-play(score-adjacent-events(my-score,
                                  quote(adjust-durations)))
```



Composition: Some Guidelines

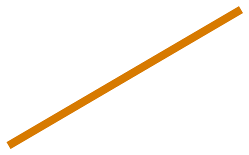
- Vocabulary
  - Rhythm
  - Melody
  - Harmony
  - Timbre
  - Texture
- Organization
  - Structures
  - Elaboration
  - Ornamentation
  - Contrasting elements
  - Gestures

ICM Week 2 15



Gesture Example

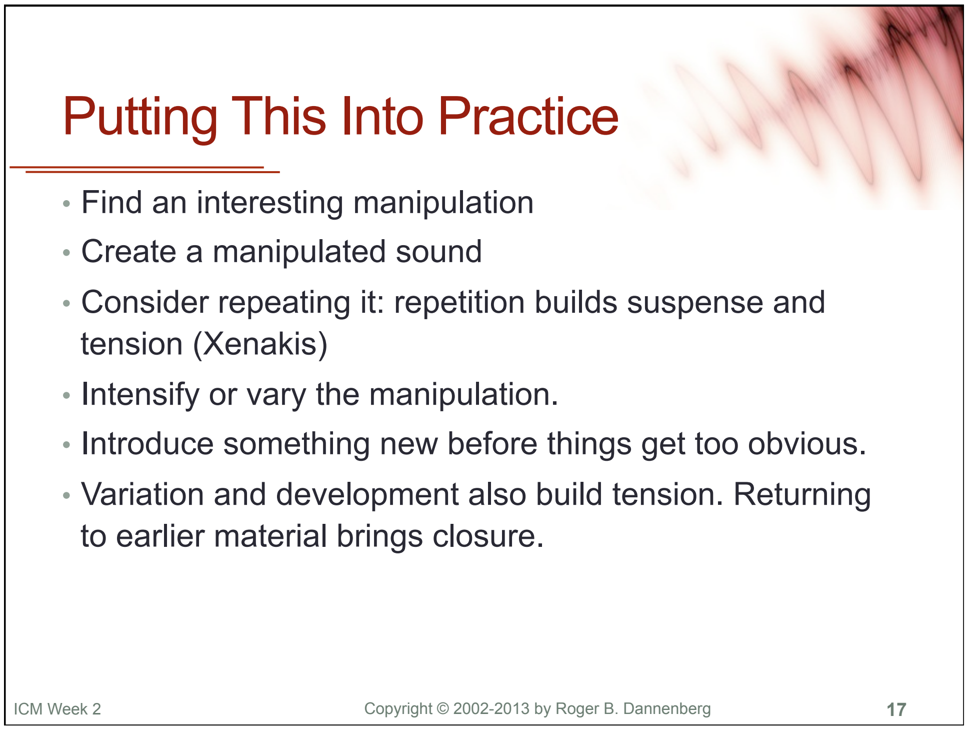
- Consider this "gesture":



- Rhythm: Increasing tempo
- Melody: Upward melodic contour
- Harmony: Increasing dissonance
- Timbre: Progression toward "thinner" sound
- Texture: Shorter, lighter, busier

- So, organization (structure) transcends vocabulary (the space of variation)

ICM Week 2 16



Putting This Into Practice

- Find an interesting manipulation
- Create a manipulated sound
- Consider repeating it: repetition builds suspense and tension (Xenakis)
- Intensify or vary the manipulation.
- Introduce something new before things get too obvious.
- Variation and development also build tension. Returning to earlier material brings closure.