Delicate Computations

Philip Corner: Larry Polansky


Stable URL:

Perspectives of New Music is currently published by Perspectives of New Music.

Your use of the JSTOR archive indicates your acceptance of JSTOR’s Terms and Conditions of Use, available at http://www.jstor.org/about/terms.html. JSTOR’s Terms and Conditions of Use provides, in part, that unless you have obtained prior permission, you may not download an entire issue of a journal or multiple copies of articles, and you may use content in the JSTOR archive only for your personal, non-commercial use.

Please contact the publisher regarding any further use of this work. Publisher contact information may be obtained at http://www.jstor.org/journals/pnm.html.

Each copy of any part of a JSTOR transmission must contain the same copyright notice that appears on the screen or printed page of such transmission.

JSTOR is an independent not-for-profit organization dedicated to creating and preserving a digital archive of scholarly journals. For more information regarding JSTOR, please contact support@jstor.org.
DELICATE COMPUTATIONS

PHILIP CORNER (with LARRY POLANSKY)

COME TO COMPUTER!

WAS NEVER ABLE before. Had to get simple enough.

A superhuman performer requires a superhuman listener . . . for, as
the limits are reached and surpassed of what our poor bodies can perform, so
they are for what our poor brains can appreciate. (So wire two machines to
each other, one to generate and the other taking dictation.) Years ago, Jim
Tenney got a few of us together—a FORTRAN class right out of Bell Labs
brought Downtown (and Alison Knowles, for one, made extraordinary use of
that), but I kept finding that whatever I wanted I could figure out myself; and
whatever I wanted to listen to I'd prefer the living kind of players for. (Human
improvisation remains the superior infinity process.) Jim even showed me that
my particular way of formulating indeterminacy—in pieces for piano, string
instruments, etc.—often resembled the way one programmed information in
any case. The major problem seemed to me be timbre. (I mean, how often
have you heard a synthesized piece whose sound you liked?)

Actually, synthetic is sterile—has to be.

It contrasts with the inexhaustible richness of life, of the world as given.

So, by artificially creating ever more virtuosic displays of complication which
begin to approach nature, the contrast becomes ever more evident. A sim-
ulacrum can never be "the real thing" (even when perfect!).

On the other hand, if you were to go in the opposite direction, really
desiring a simplification of sound so pure that it is a projection from the realm
of mental essences . . .

Actually, I'd learned this years ago, from a tape loop made by Jon Child
with staccato sine tones as a realization of my "open structure": Gamelan
VOX—where a beat is divided into progressively smaller regular divisions. (I
can execute this as a vocal mnemonic up to six. But the electronic loop took it
to seven. And of course could easily go on from there. Most importantly, it
sounded good. It was allowed into a performance.)

This made me think of the early Electronic Music Studies, as well as much
subsequent production. It seems to me that the reason I find them almost
unlistenable is that there is a contradiction between their simplicity of material
and the complexity of pitch and time parameters. (Although I do wish to
acknowledge the special beauty of Tenney's Ergodos and Stochastic Studies, which
seem to strike the right balance.)
With the connivance of Larry Polansky, I have just been lured into a studio.

The “meta-score” of Delicate Computations (also from my “Gamelan” series) had been made several years before. And it needs the computer.

A development from an earlier and simpler piece: pulse, it makes incredibly subtle precision serve an immediately graspable idea.

An eleven-phrase, approximately forty-minute version was typed into the computer by Larry from my dictation. This “performance” was practically in real time—although the program for it occupied him the better part of three days.

If the computer itself were to be brought into the concert, it could decide its own variations, endlessly. We are planning such an evening.

And now, I can see how to bring a specifically human sensibility back into the act.

NOTES

DELICATE COMPUTATIONS

The human complement
To computer generation

The basic procedures followed improvisationally
Replacing the computer's calculated precisions by awareness — directness of perception
to the degree that the identities and
progressions can be felt: Try to
keep a time-area constant: To
retard or accelerate by the smallest degree.

Acceptance of your instrument's limitations
of fixed intonation.

Use of all its resources, and your techniques
appropriately

Which can be played as an independent stratum
with another, for some other, tracks of
computerized sound. Also as solo,
or a number of independent versions.
Examples of "delicate computations".

Line of same length bent so as to expand time:

\[
\begin{array}{ccccccc}
\hline
& & & & & & \\
\end{array}
\]

Same line continued with even spaced pulsations, thus contracting interval scale:

\[
\begin{array}{ccccccc}
\hline
&&&&&&\end{array}
\]

Line constant in length and direction containing decreasing number (hence slowing rhythm) of notes:

\[
\begin{array}{ccccccc}
\hline
\vline&\vline&\vline&\vline&\vline&\vline&\vline\end{array}
\]

Decreasing number of notes at constant tempo, thus contracting the line:

\[
\begin{array}{ccccccc}
\hline
& & & & & & \\
\end{array}
\]

Increase of interval size and number of notes in constant tempo at constant speed:

\[
\begin{array}{ccccccc}
\hline
& & & & & & \\
\end{array}
\]
DELICATE COMPUTATIONS

Groups of patterns forming systematic subtle progressions in various manifold ways.

These can be expressed as lines (diagonal), giving the scalar (melody) neglect (which can be played up or down).

EXAMPLES

\[ \begin{align*}
\text{\textit{ex.}} & \quad - \quad \text{constant repeating} \\
\text{\textit{this}} & \\
\text{can only be permitted when the progression occurs in the remaining dimension: intensity} & \\
( \text{\textit{ffppoppmnpnffff}} ) & \quad \text{on the retrograde.} \\
\text{further subtly:} & \quad \text{\textit{fppmnnnffppff}} & \quad \text{etc.} \\
\text{These melody-shape lines are translatable either into pure glass (precisely!)} & \\
\text{or pulsations, evenly timed \ldots\ or evenly spaced, minutely-distant interval tones.} & \quad \text{- etc.}
\end{align*} \]

© 1972 Philip Corner
(Japanese) DELICATE COMPUTATIONS

PITCH
Start End #Steps
Low 304, avg 162.5 cp
High 120, avg 60 cp
up to 30 pm 8x

TEMPO
(Duration of):
Each Sound
Silence

0016 60
5 sec 0.5

TIMBRE
Triangle
Sawtooth
Goo (bad sine)
Rand. ("dirty")

PHRASES # of ; evolution

Silence length between phrases

INTENSITY
Dynamics are a live performance variable

1. 2040 967
2. 2791 860
3. 500 2.90
4. 9405 7670
5. 400 779
6. 940 703
7. 122 1155
8. 13,011 6101
9. 3000 4204
10. 11,040
11. 13,250 2000

8pm 25 1.4", 100%, and Δ 8x
21 pm 14" .9" " M 6x Tempom
23 pm 11" 1.5" " M 12x
int 44 → 30 3.2", 2.9:03 Δ 4x
int 31 → 40 1.47, 1.1:37 m 11x
int 19 → 301 1.05 100% Δ 16x
int 80 → 7671 .4, 1.2:18 m 3x .4 between
9 pm 111 1.1, 8:2 m 10x 3.3, .3
2.68 8pm 2 → 28pm m 2.7x
11.9, 100% Δ 3x 1.74, .3
Philip Corner's DELICATE COMPUTATIONS

author: Larry Polansky, 7/86

Much of the software for this work is written using the language
HMSL (Hierarchical Music Specification Language) developed by
Larry Polansky, David Rosenboom and Phil Burk, at the
Mills College Center for Contemporary Music

For this piece to run, HMSL must have first been loaded.
This piece also uses a simple linear interpolation algorithm,
not included in HMSL, but not shown here.

Ratios for high equal temperaments were derived with the assistance
of Dr. John Chalmers' "Computer Generated Tables for 1200 Tone
Equal Temperament" (originally published in Xenharmonikon)

new task-delicat  \ compilation fence

\ variables
v: p-tempo  \ general rate of change
v: p-#-steps  \ number of steps in each segment
v: p-start  \ starting pitch
v: p-end  \ ending pitch
v: p-envelope-length  \ length of envelope if amp. envelope used
v: p-gliss-on?  \ flag to determine if amp. envelope used or not
v: p-loudness-on?  \ flag to see if loudness gliss...
v: cresc/decresc?  \ flag
v: p-timbre-on?  \ timbral evolution enable
v: temp-index  \ counter for segment loop
v: numerator  \ for ratio divide
v: denominator
v: prev-period  \ previous period for taking succesive ratios
v: up/down?  \ melody ascending or descending
v: quit?  \ flag for ending segment when endpoint is reached

\ some simple waveforms: note that this piece uses only Amiga local sound
ob.waveform p-sawtooth
ob.waveform p-goo
ob.waveform p-rand
ob.waveform p-triangle

==============================================================================
Up to 30 equal divisions of the octave are used, and the following
table is a table of successive ratios, numerator then denominator
for easily computing the nearest ratio for a given equal temperament.
This method is used because at the time this piece was written, the
Beta version of Delta Research JFORTH used did not yet support
floating point

CREATE TUNING-TABLE
df1:larrypieces/delicate

99, 70, 63, 50,
44, 37, 1024, 891,
55, 49, 243, 220,
12, 11, 27, 25,
15, 14, 16, 15,
196, 185, 135, 128,
21, 20, 22, 21,
2673, 2560, 126, 121,
80, 77, 28, 27,
1034, 1000, 1032, 1000,
1031, 1000, 1030, 1000,
1029, 1000, 1028, 1000,
1027, 1000, 1026, 1000,
1025, 1000, 1024, 1000,
1023, 1000, 1022, 1000,
\ after 31 divisions do them arithmetically....

\ get ratio from current equal temperament for instrument
: GET.RATIO ( --- )
\ only do ratios for first thirty divisions of the octave...
p-#steps @ \ number of pitches in tuning
dup 30 >
IF
  drop \ don't need it
  1 numerator ! 1 denominator ! \ 1/1
ELSE \ use tuning table for ratio...
  2 - dup \ alter # of steps to offset into tuning table
  8 * tuning-table + @ numerator ! \ offset for num.
  8 * 4 + tuning-table + @ denominator ! \ for denom
THEN

\ compute actual period from stored ratio
: GET.PITCH ( --- )
  get.ratio
  \ determine whether ascending or descending for "getout"
p-start @ p-end @ <
  IF up/down? disable
  ELSE up/down? enable
  THEN
  p-#steps @ 30 >
  IF \ if too many steps use arithmetic interpolation
    temp-index @ interp \ interpolate period
dup prev-period ! \ store it for quit test
  put.period: ins-can-1 \ use a predefined "instrument"
  ELSE up/down? @
    IF \ melody ascending
      prev-period @ denominator @ numerator @
    *\ /* common way to do ratios in fixed point FORTH
dup prev-period !
    put.period: ins-can-1 \ use that period
  ELSE \ melody descending, same as above
      prev-period @ numerator @ denominator @
    */
dup prev-period !
put.period: ins-can-1
THEN
THEN
\ test for leaving begin-until loop in play.segment
\ look to see if period exceeds endpoints
up/down? @
IF
prev-period @ p-end @ <
IF quit? enable THEN
ELSE
prev-period @ p-end @ >
IF quit? enable THEN
THEN
;
\ start crescendo at 0 loudness
: P-CRESCEndo 0 da.loudness! cresc/decresc? enable ;
: P-DECRESCendo cresc/decresc? disable ;

\ set up simple, 8 byte waveform tables
: P-WAVEFORM$ INIT
 8 new: p-sawtooth
 8 new: p-goo
 8 new: p-rand
 8 new: p-triangle
 0 DO
    i 32 * i to: p-sawtooth
    255 choose i to: p-rand
    i 4 <
    IF i 32 * ELSE 255 i 32 * THEN
    i to: p-triangle
LOOP
0 DO
    i 45 * sin 4 / i to: p-goo
LOOP
;
\ initialization for simple instruments
: USE.GOO
  p-goo put.waveform: ins-can-1
  setup: ins-can-1
;
: USE.SAWTOOTH
  p-sawtooth put.waveform: ins-can-1
  setup: ins-can-1
;
: USE.RAND
  p-rand put.waveform: ins-can-1
  setup: ins-can-1
;
: USE.TRIANGLE
df1:larvpieces/delicap  28-Jan-87  09:40:29 am  Page# 4

p-triangle put.waveform: ins-can-1
setup: ins-can-1
;

\ ===============================================================================================

\ point by point deformation of the waveform table
 Definitions.WAVEFORM
 p-timbre-on?
 IF
  p-end @ p-start @ - abs \ get range of gliss
  choose \ pick a random value
  temp-index @ <
  IF \ is it time to pick a new timbre?
    255 choose \ pick sample point value
    8 choose \ pick which point
    get.waveform: ins-can-1 to: [] \ "late bind"
    \ get current waveform
  THEN
 THEN

\ ===============================================================================================

\ The segment generating routine "play.segment" uses the HMSL
\ Amiga-specific definition of instruments, though in a very simple
\ way. PLAY.SEGMENT assumes values in all the relevant parameters
\ and that segment out to the audio channel. This word is the
\ main body of the routine P-OD, used in all the phrase routines of
\ DELICAT.PIECE

 : PLAY.SEGMENT \ play line segment
  0 p-start @ p-#-steps @ p-end @
  set.interp \ if high division then interpolate
  \ set.interp needs x1,y1, x2,y2
  p-start @ prev-period ! \ if not then need successive ratios
  p-gliss-on? @ not
  IF \ if envelope then put the envelope in instrument
    env-bang put.envelope: ins-can-1
    setup: ins-can-1
 ELSE \ use null, or no envelope
    0 put.envelope: ins-can-1
    setup: ins-can-1
 THEN
 THEN
  0 temp-index !
 start: ins-can-1 \ start the first "canned" instrument" regardless
 BEGIN \ outer loop
  temp-index @ 1 + temp-index ! \ store value of "loop"
  \ for timbre deformation
  \ and arithmetic divisions...
  deform.waveform
  temp-index @ 0=
  IF p-start @ put.period: ins-can-1
  ELSE get.pitch \ does all pitch calculation and sending
 THEN
 p-gliss-on? @ not
IF \ if envelope then start it
    start: ins-can-1
THEN
p-loudness-on? @
    IF
       temp-index @ interp \ get the interpolated value
       p-start @ p-end @ min -
       \ the interpolated value - the smaller of the
       \ endpoints is the absolute...
       64 * \ scale it to the loudness range of the Amiga
       p-end @ p-start @
       - abs \ divide by total range of gliss
       /
       cresc/descresc? @
       IF \ crescendo or decrescendo
           64 swap -
       THEN
       da.loudness! \ store loudness of instrument
       THEN
       p-tempo @ msec \ tempo
       p-gliss-on? @ not
       IF \ if envelope then finish it
           finish: ins-can-1
       THEN
quit? @
UNTIL
    finish: ins-can-1 \ turn the instrument off at end of segment
    quit? disable
;
\ ================================================================================
\ do some preliminary initialization of the instruments, etc.
: DELICAT.INIT
    p-waveforms.init
    p-triangle put.waveform: ins-can-1 \ triangle wave is default
    \ for init purposes only, some arbitrary parameters
    12 p-#-steps ! \ 12-tone equal default
    2000 p-start ! \ medium starting period
    1000 p-end ! \ ending period
    100 p-tempo ! \ tempo
    0 put.envelope: ins-can-1 \ don't use envelopes!!!!!
    setup: ins-can-1 \ calls H&SL setup: method for instruments
    p-gliss-on? enable \ gliss, not envelopes
;
\ free allocated memory for waveform objects
: P.MELODY.TERM
    free: p-rand
    free: p-goo
    free: p-triangle
    free: p-sawtooth
;
\ ================================================================================
\ random generator of segments, for testing purposes only
df1:larrypieces/delicit  28-Jan-87  09:45:17 am  Page#  6

: TEST.DELICIT
BEGIN
    2 choose
    .  IF p-loudness-on? disable ELSE p-loudness-on? enable THEN
    2 choose
    .  IF p-timbre-on? enable ELSE p-timbre-on? disable THEN
    500 10 wchoose p-tempo !
    2 choose
    .  IF p-crescendo ELSE p-decrescendo THEN
    2 choose
    .  IF p-gliss-on? enable
    ELSE p-gliss-on? disable
    p-tempo @ dup 20 <
    .  IF drop 30 THEN
    20 - choose put.msec: env-bang
    THEN
    2000 200 wchoose p-start !
    2000 200 wchoose p-end !
    40 choose p-#-steps !
    4 choose dup 1 = IF use.goo drop THEN
        dup 2 = IF use.rand drop THEN
        dup 0 = IF use.triangle drop THEN
        dup 3 = IF use.sawtooth drop THEN
        drop
        play.segment
    5000 choose msec \ random milliseconds up to 5 seconds
    ?terminal
UNTIL
;

\ ================================
\ user utilities

: SET.POINTS \ starting ending #-steps tempo ---
    p-tempo !
    p-#-steps !
    p-end !
    p-start !
;

\ The following routine, P-GO, is the crux of the routines in the
\ "piece" file, called DelicitPiece. P-GO accepts the four parameters
\ of set.points (starting period, ending period, number of steps,
\ and tempo, and uses the current instrument definition (including
\ envelope, waveform, timbral evolution, etc.) to play one segment
: P-GO
    set.points
    play.segment
;
\ diagnostic
: SHOW.PARAMETERS
    ." timbre " p-timbre-on? ? . cr
    ." loudness " p-loudness-on? ? . cr
." gliss " p-gliss-on? ? . cr
." steps " p-##-steps ? . cr
." start " p-start ? . cr
." end " p-end ? . cr
." tempo " p-tempo ? . cr
 cr ." HI PHIL !!!!!!! "

\ phrases for Philip Corner's DELICATE COMPUTATIONS
\ author: larry polansky
\ 7/86
\nThis file is the actual piece, which uses the routines set up in
the file "Delicate". The piece, in this realization consists of
a series of "phrases", each of which has several initial parameters
including: glissando, tuning, waveform, timbral evolution, and others.
These parameters were specified by the composer, and programmed in
close to "real-time" by the author of the software.

Each of the phrases is, in a sense, a short and direct algorithmic
piece, a kind of software translation of the composer's description
of a sonic process

new task-delicate.piece \ fence for compilation

v: prev-event1 \ keep track of previous events in variable
v: prev-event2

\ Note: P-Go accepts the parameters: starting period, ending
\ period, #-of-steps to the octave, tempo

\ constant range, one increasing subdivision each time
: PHRASE.1
  p-gliss-on? enable \ yes for gliss...
  use.triangle \ triangle wave
  p-timbre-on? disable \ no timbral evolution
  9 0 DO \ st per end per #-steps tempo ---
     2020 967 17 i + 1400 p-go \ number of steps
     \ incremented each time through loop
  esc IF LEAVE THEN
  LOOP

;  
\ transposing up by 21st of an octave each time ( approx. 57 cents )
: PHRASE.2
  p-gliss-on? enable
  use.sawtooth
  p-timbre-on? disable
  2799 prev-event1 !
  880 prev-event2 !
  2799 880 21 900 p-go
  5 0 DO
      prev-event1 @ 1000 1038 */ dup prev-event1 !
      prev-event2 @ 1000 1038 */ dup prev-event2 !
     21 900 p-go
  LOOP

;  
\ successively larger subdivisions of a segment
: PHRASE.3
  p-gliss-on? enable
  use.sawtooth
  12 0 DO
    500 290 23 i - 150 p-go
df1:larrypieces/delicat.piece

```
LOOP

\ number of steps decreases slightly
: PHRASE.4
  use.triangle
  p-gliss-on? disable
  2900 put.msec: env-bang
  4 0 DO
  9405 6544 44 i - 3200 p-go
  esc IF LEAVE THEN
  LOOP

\ fast descending, getting slower and slower
: PHRASE.5
  p-gliss-on? disable
  use.goo
  60 put.msec: env-bang
  80 prev-event1 !
  400 799 29 80 p-go
  6 0 DO
  400 799 29 prev-event1 @ 12 11
  */
  dup prev-event1 ! p-go
  esc IF LEAVE THEN
  loop

\ successively smaller subdivisions
: PHRASE.6
  p-gliss-on? disable
  use.rand
  100 put.msec: env-bang
  11 0 DO
  940 703 31 i + 470 p-go
  esc IF LEAVE THEN
  LOOP

\ downward glissandi
: PHRASE.7
  use.triangle
  p-gliss-on? enable
  16 0 DO
  122 1156 19 i - 50 p-go
  esc IF LEAVE THEN
  LOOP

\ three phrases, 80, 76, and 71 steps respectively
: PHRASE.8
  p-gliss-on? disable
  use.sawtooth
  120 put.msec: env-bang
  13011 6101 80 400 p-go
```
400 msec
13011 6101 76 400 p-go
400 msec
13011 6101 71 400 p-go

\ phrase which continually "squishes" itself into smaller range...
: PHRASE.9
  p-gliss-on? disable
  use.rand
  800 put.msec: env-bang
  4204 prev-event1 !
  3000 4204 9 1100 p-go \ first sub-phrase
  3300 msec
  10 0 DO
    3000 prev-event1 ! 1080 1000 */ \ ending period changing
dup prev-event1 !
9 1100 p-go \ always 9 steps, same tempo
3100 300 i * - msec \ changing delays between sub-phrases
esc IF LEAVE THEN
  LOOP

\ envelope gets shorter, phrase gets quicker, #of-steps gets smaller,
\ ending period gets lower and lower
: PHRASE.10
  p-gliss-on? disable
  use.goo
  29 2 DO
    11040 \ starting period
    11040 4 / 20 + \ ending period
    i \ #of-steps, increasing each time
    2500 i / \ tempo, decreasing each time
dup 2 / \ divide by two for envelope length
    100 max put.msec: env-bang
    p-go
    esc IF LEAVE THEN
  LOOP

\ 3 sub-phrases, 11, 12, and 13 steps to different octaves...
: PHRASE.11
  p-gliss-on? enable
  use.triangle
  12206 24412 11 11900 p-go
  1700 msec
  13501 27002 12 11900 p-go
  1700 msec
  15000 30000 13 11900 p-go
  1700 msec

\ put them all together into one, approximately 45 minute piece, and
\ type PLAY.PIECE and sit back and enjoy...
: PLAY.PIECE
  ." phrase 1 " cr phrase.1 20000 msec
" phrase 2 " cr phrase.2 33000 msec
" phrase 3 " cr phrase.3 8000 msec
" phrase 4 " cr phrase.4 15000 msec
" phrase 5 " cr phrase.5 9000 msec
" phrase 6 " cr phrase.6 26000 msec
" phrase 7 " cr phrase.7 3000 msec
\ insert terminal wait loop for when next phrase starts....
" press return to play next phrase " key drop
" phrase 8 " cr phrase.8 19000 msec
" press return to play next phrase " key drop
" phrase 9 " cr phrase.9 39000 msec
" press return to play next phrase " key drop
" phrase 10 " cr phrase.10 51000 msec
" press return to play next phrase " key drop
" phrase 11 " cr phrase.11