Rina Refianti
Achmad Benny Mutiara

Bahasa Pemrograman ChucK

Penerbit Gunadarma

Rina Refianti
Achmad Benny Mutiara

Bahasa Pemrograman ChucK

Penerbit Gunadarma

Bahasa Pemrograman ChucK

Penyusun : Rina Refianti dan Achmad Benny Mutiara

Desain : Restu Ibu

Desain dan Layout : Toto Bes

Diterbitkan pertama kali oleh Universitas Gunadarma

Jakarta 2018

ISBN 978-602-9438-90-1

Kata Pengantar

Puji syukur kehadirat Tuhan Yang Maha Esa, karena dengan rakhmat dan berkat Hidayahnya catatan kuliah ini dapat diselesaikan.

Dalam suatu institusi pendidikan, proses utama yang sangat perlu diperhatikan dan merupakan tolok ukur dari kualitas institusi tersebut adalah proses belajar mengajar yang terjadi antara mahasiswa dan dosen. Guna menunjang proses tersebut kami team pengajar pemrograman multimedia menyusun buku ini.

Selain diperuntukkan bagi mahasiswa, buku ini juga diharapkan dapat digunakan sebagai acuan keseragaman materi antar dosen yang mengajar pada beberapa kelas parallel di Jurusan Teknik Informatika.

Ucapan terima kasih penulis sampaikan pada Pimpinan Universitas Gunadarma, Mahasiswa Teknik Informatika, serta isteri dan anak-anakku tercinta, karena atas dorongan dan semangat dari mereka buku ini tertulis.

Buku ini masih jauh dari sempurna. Kritik dan saran yang membangun dapat para pembaca sampaikan ke penulis.

Semoga buku ini bermanfaat bagi para ahli di bidang teknologi informasi yang berminat pada bidang spesifik multimedia digital. Amin.

Jakarta, 2018
Rina Refianti
Achmad Benny Mutiara

DAFTAR ISI

Kata Pengantar ……....……...i

1 Intro-ChucK-tion 1

2 Instalasi 3

Instalasi Binary ……...……3

Instalasi Source ……...……...6

3 Tutorial ChucK 9

Sebuah tutorial ChucK ……...9

Konvensi ……...13

Pemrograman On-the-fly ……..14

Pemodifikasian dasar patch…… .17

LFOs dan Blackhole ……...……..19

Bekerja dengan MIDI ……...….. 20

Penulisan ke Disk ……24

Stereo ……25

4 Ikhtisar 27

Menjalankan ChucK ……27

Komentar ……28

Debug print ….. ……...…….29

Reserved words…..…….. 30

5 Tipe data, Nilai, and Variabel 33

Tipe data primitif ……..……... 33

Nilai (literal) ……..……...34

Variabel…….34

Tipe data acuan ... …...……...36

6 Array 37

Deklarasi ……..……..37

Array multi dimensi …….……...38

Lookup ……...…… 39

Array assosiatif ……41

Operator array …… 43

7 Operator dan Operasi 45

Cast …….. ……48

8 Struktur Kontrol 53

If / else ……..53

While ...…….54

Until ……..54

For …...…….55

Break / continue ……...……55

9 Fungsi 57

Writing . . . . . .…....57

Calling . . . . . .…..…58

Overloading . . . . . .…....58

10 Concurrency and Shreds 61

Sporking shreds (dalam code) 61

The ’me’ keyword 63

Menggunakan machine.add() 64

Komunikasi Inter-shred 65

11 Time and Timing 67

Time and duration . . . . . .…..67

Operations on time and duration (aritmatika) 69

The keyword ‘now’ 70

Mempercepat time . . . . . .…..71

12 Events 75

Siapakah mereka… 75

Penggunaan 71

Midi events ……... 78
Osc events . . . . . ……. 79
Membuta kustom events . . …. 80
13 Objects 83
Pengenalan …. 83
Built-in class …….. 84
Bekerja dengan objects 84
Menulis class…….. 86
Anggota (instance data + fungsi) …. 87
Class constructors . . . . . ……... 89
Static (data + fungsi) ……... 90
Pewarisan . . . . . ……. 91
Overloading . . . . . ……... 94
14 The ChucK Compiler + Virtual Machine 95
15 Perintah-perintah Pemrograman On-the-fly 99
16 Standard Libraries API 103
17 Unit Generators 109

BAB 1

Intro-ChucK-tion

Apakah ini: ChucK adalah bahasa pemrograman general-purpose, yang diharapkan untuk real-time audio syntesis dan pemrograman grafik atau multimedia. Chuck memperkenalkan model pemrograman secara bersamaan yang melekatkan pemilihan waktu secara langsung dalam alur program (kita menyebutnya strongly-timed). Potensi lain dari fitur yang bermanfaat meliputi kemampuan untuk menulis atau mengubah program-program on-the-fly.

Untuk siapa: para peneliti audio atau multimedia, pengembang, penggubah, pemain seni peran. Platform-platform yang mendukung:

- MacOS X (CoreAudio)
- Linux (ALSA/OSS/Jack)
- Windows/also Cygwin (DirectSound)

Strongly-timed

Model pemrograman ChucK menyediakan para programmer langsung, tepat, dan kendali menarik dari waktu ke waktu, durasi, tingkatan, dan hanya sekedar hal lain yang meyertakan waktu. Hal ini membuat ChucK adalah potensi yang menyenangkan dan tool-tool yang sangat fleksibel untuk pendeskripsian, perancangan, dan pengimplementasian sintesis suara dan pembuatan musik pada level rendah dan tinggi.

On-the-fly Programming

Pemrograman on-the-fly adalah gaya pemrograman dimana programmer/pemain seni peran/penggubah meningkatkan dan memodifikasi program saat saat sedang berjalan, tanpa menghentikan atau me-restart, agar menyatakan ekspresif, kendali programmable untuk penampilan, komposisi, dan eksperimentasi saat run-time. Karena kekuatan fundamental dari bahasa pemrograman, kita percaya aspek teknik dan estetik dari pemrograman on-the-fly adalah eksplorasi yang berharga.

BAB 2

Instalasi

Kita mencoba membuat ChucK mudah dan memungkinkan untuk membangun (jika diinginkan), meng-instal dan penggunaan ulang. Semua source file-header source untuk compiler, vm, dan mesin audio harus ditempatkan pada direktori yang sama. Perbedaan platform diringkas pada level paling dasar (terima kasih untuk Gary Scavone). Tak satupun compiler atau vm mempunyai beberapa kode OS-depedent. Ada juga executable pre-compiled yang tersedia untuk OS X dan Windows. 'chuck' klasik berjalan sebagai sebuah program command line. Terdapat pengembangan pengintegrasian dengan GUI-based dan lingkungan pencapaian yang baik yang dapat digunakan sebagai virtual mesin chuck standalone, atau dalam penghubung dengan 'chuck' versi command. Lingkungan GUI_based meliputi miniAudicle (http://audicle.cs.princeton.edu/mini). Bagian ini sebagian besar berhadapan dengan the classic yaitu command line dari versi chuck.

Instalasi Binary

Distribusi binary meliputi direktori yang disebut bin yang berisi binary precompiled dari ChucK untuk system operasimu. Distribusi binary adalah jalan terbaik untuk mendalami ChucK.

OS X

1. Terminal adalah lokasi dalam Utilities/ folder dalam Applications/ folder di dalam hardiskmu. Bukalah terminal(buat shortcut untuk itu jika kamu ingin, karena kita akan sering menggunakannya dengan chuck commang-line). Dalam terminal akan dilanjutkan ke bin/direktori(meggantikan chuck-x.x.x.x-exe dengan nama direktori yang sebenarnya):

```
%>cd chuck-x.x.x.x-exe/bin
```

2. Install-lah dengan commad berikut ini.

```
%>sudo cp chuck /usr/bin/
```

(masukan password saat prompt)

```
%>sudo chmod 755 /usr/bin/chuck
```

Sekarang kamu harus bisa menjalankan ‘chuck dari direktori tersebut.

3. Tes untuk meyakinkan bahwa telah terinstal dengan benar.

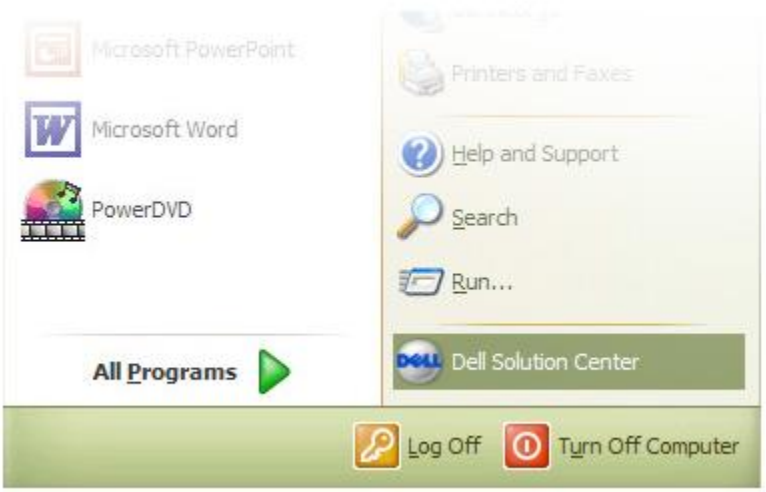
```
%>chuck
```

Kamu harus melihat pesan berikut(yang merupakan perilaku yang benar):

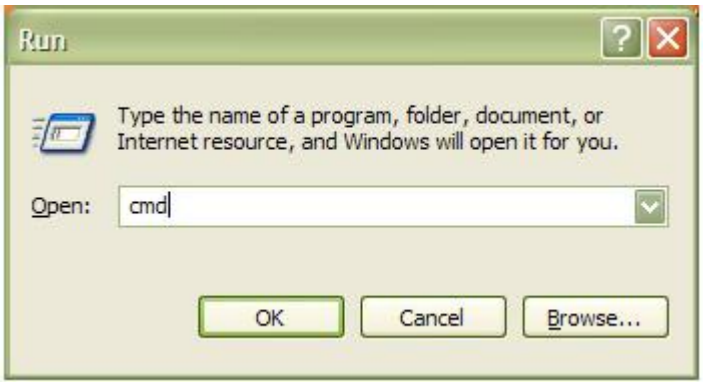
```
[chuck]: no input files... (try –help)
```

Windows

1. Tempatkan chuck.exe (temukan dalam folder ‘bin’) didalam c:\windows\system32\
2. Bukalah command window yang ditemukan di start-> run.



3. Ketikkan cmd dan tekan enter.



3. Ketikkan chuck dan tekan enter, kamu harus melihat:

```
%>chuck
[chuck]: no input files... (try –help)
```

Instalasi Source

Untuk membuat chuck dari source (pemakai Windows: hal itu mungkin membuat ChucK dari keduanya Visual C++ 6.0 dan dari cygwin- bagian ini mendeskripsikan pembuatan dengan cygwin):

1. Carilah src/direktori (gantilah chuck-x.x.x.x dengan nama direktori yang sebenarnya):

```
%>cd chuck-x.x.x.x/src/
```

2. Jika kamu ketikkan ‘make’ disini, kamu harus mendapatkan pesan berikut:

```
%>make
[chuck] : please use one of the following configurations:
make osx-ppc, make osx-intel, make win32,
make linux-oss, make linux-alsa, make linux-jack
```

Sekarang, ketikkan command yang sesuai dengan platformmu…
contohnya untuk MacOS X (power-pc):

```
%>make osx-ppc
```

contohnya untuk MacOS X (intel):

```
%>make osx-intel
```

contohnya untuk Windows (under cygwin):

```
%>make win32
```

3. Jika kamu ingin meng-install chuck (cp into /usr/bin by default). Jika kamu tidak suka dengan membuat tujuannya, edit makefile dibawah ‘install’, atau lompati langkah ini semuanya. (Kita merekomendasikan meletakkannya di suatu tempat di path kamu, itu membuat pemrograman on-the-fly lebih mudah)

```
# (optional: edit the makefile first)
%>make install
```

Kamu boleh mempunyai administrator khusus dalam menginstal ChucK. Jika kamu mempunyai akses admin lalu kanu dapat menggunakan sudo command untuk menginstal.

```
%>sudo make install
```

4 Jika kamu belum pernah mendapat pesan kesalahan terkenal manapun sampai kepada titik ini, kemudian kamu harus laksanakan.

Harus ada ‘chuck’ executable dalam directori sekarang. Agar memeriksanya cepat, eksekusi seperti berikut. (use ‘./chuck’ jika chuck tidak berada di path kamu), dan lihatlah apakah kamu mendapatkan output yang sama:

```
%>chuck [chuck]: no input files...
```

(jika yang kamu lakukan ada kesalahan selama kompilasi, atau kamu menjalankan yang sama dengan masalah lain, maka biarkan kamu mengetahuinya dan kita akan melakukan yang terbaik dari kita dengan menyediakan support). Kamu siap

dengan ChucK. Jika ini adalah pertama kalinya kamumemprogram dengan ChucK, kamu mungkin memerlukan dokumentasi atau mengambil tutorialdari ChucK di(http://chuck.cs.princeton.edu/doc/). Terima kasih banyak. Go forth and ChucK kirim email ke kami untuk dukungan atau membuatusul atau hubungi kita.
Ge + Perry

BAB 3
Tutorial ChucK

Sebuah Tutorial ChucK

Hello ChucK:

Tutorial ini ditulis untuk versi command line dari ChucK (sekarang yang paling stabil dan secara luas didukung). Jalan lain untuk menjalankan ChucK meliputi menggunakan miniAudicle (download dan dokumentasi: http://audicle.cs.princeton.edu/mini/) dan Audicle (in pre-pre-alpha). Kode ChucK sama, tapi jalan untuk menjalankannya yang berbeda, tergantung dari system ChucK tersebut. Hal pertama yang harus kita lakukan adalah memabangkitkan gelombang sinus dan mengirimnya ke speaker sehingga kita bisa mendengarnya. Kita dapat melakukannya dengan mudah dalam ChucK dengan menghubungkan modul pemrosesan audio (unit pembangkit) dan mereka bekerja bersama menghasilkan suara. Kita mulai dengan program ChucK kosong dan menambahkan baris kode berikut:

```
// connect sine oscillator to D/A convertor (sound card)
SinOsc s => dac;
```

NOTE: Sebagai defaultnya, program ChucK memulai eksekusi dari instruksi pertama dari level atas lingkup (global).

Beberapa hal yang harus dikerjakan:

1. Membuat sebuah unit pembangkit baru dengan tipe ‘SinOsc’ (sine oscillator), dan menyimpannya dalam variable ‘s’.
2. ‘dac’ (D/A convertor) adalah special unit pembangkit (dibuat oleh sistem)yang mana abstaksi kita untuk dasar dari tampilan audio.
3. kita menggunakan operator ChucK (=>) untuk ChucK ‘s’ menuju ‘dac’. Dalam ChucK, saat satu unit pembangkit ChucKed kepada lainnya, kita menghubungi mereka. Kita dapat pikirkan dari garis ini untuk menentukan alur data dari ‘s’,

sebuah sinar pembangkit menuju ‘dac’, yaitu sound card atau speaker. Secara bersama disebut dengan ‘patch’.

Yang di atas adalah program ChucK yang valid, tapi semua itu bekerja begitu jauh dalam membuat hubungan- jika kita menjalankan program ini maka dia akan berhenti dengan seketika. Dalam urutan ini apa yang kita inginkan, kita memerlukan menjaga dari satu lagi hal yang sangat penting yaitu waktu. Tidak seperti bahasa-bahasa lainnya, kita tidak harus dengan tegas berkata “play” untuk mendengarkan hasilnya. Dalam ChucK, kita sederhananya harus mengijinkan waktu untuk lewat atau istilahnya “allow time to pass” untuk data yang dihasilkan. Kita akan melihat, waktu dan data audio kedua-duanya adalah hubungan yang inextricably dalam ChucK (sebagaimana dalam kenyataan), dan terpisah dalam cara mereka dimanipulasi. Namun untuk sekarang, mari bangkitkan gelombang sinus kita dan mendengarkannya dengan menambahkan satu baris lagi:

```
// menghubungkan sinus pada  D/A convertor (sound card)
SinOsc s => dac;
// mengijinkan 2 detik untuk lewat
2::second => now;
```

Mari kita jalankan ini (mengumpamakan kamu telah menyimpan file dengan nama ‘foo.ck’):

```
%>chuck foo.ck
```

Ini akan menyebabkan sound untuk memainkan selama 2 detik.(Operator :: untuk mengalikan argument dengan mudah), selama waktu data audio diproses (dan terdengar), setelah program keluar (karena itu menjangkau sampai selesai). Untuk sekarang, kita dapat mengambil kode baris kedua untuk mengartikan “mengijinkan 2 detik untuk lewat (dan membiarkan audio menghitung selama waktu itu)”. Jika kamu ingin memainkannya secara tak terbatas, kita dapat menuliskan loop:

```
// menghubungkan sinus oscillator ke D/A convertor (sound card)
SinOsc s => dac;
// loop in time
while( true ) {
```

```
2::second => now;}
```

In ChucK, this is called a ‘time-loop’ (in fact this particular one is an ‘infinite time loop’). This program executes (and generate atau process audio) indefinitely. Try running this program.

PENTING: barangkali yang lebih penting dari sekedar bagaimana menjalankan ChucK adalah bagaimana menghentikan program ChucK yang berkelanjutan dari command line, dengan menekan (ctrl c). Sejauh ini, sejak semua yang kita lakukan adalah mempercepat waktu, itu tidak masalah (untuk sekarang) nilai apa yang membantu kita – (kita gunakan 2 :: detik , tapi kita dapat menggunakan ‘ms’, ‘second’, ‘minute’, ‘hour’, ‘day’, dan ‘week’), dan hasilnya akan sama.hal ini baik untuk menjaga pikiran dari contoh ini yang hampir semua terjadi ChucK terjadi secara natural dari pemilihan waktu. Mari mencoba mengubah frekuensi secara acak tiap 100 ms:

```
// membuat patch
SinOsc s => dac;
// time-loop, yang mana frekuensi  Osc diubah tiap 100 ms
while( true ) {
100::ms => now;
Std.rand2f(30.0, 1000.0) => s.freq;
}
```

Seharusnya ini kedengaran seperti computer mainframe dalam film old sci-fi. Ada dua hal yang perlu dicatat disini:

(1) Kita mempercepat waktu disamping dengan loop dengan durasi 100::ms.

(2) Sebuah nilai acak diantara 30.0 dan 1000.0 dibangkitkan dan ‘assigned ke frekuensi oscillator, tiap 100::ms. Lanjutkan dan jalankan ini (ganti foo.ck dengan nama file kamu):

```
%>chuck foo.ck
```

Mainkan dengan parameter-parameter di program. Ubahlah 100::ms ke yang lain (seperti 50::ms atau 500::ms, atau 1::ms, atau 1::samp(tiap contoh)), atau ubah 1000.0 sampai 5000.0. Jalankan dan dengarkan :

```
%>chuck foo.ck
```

Sekali lagi hal ini bekerja, tetap pada file ini - kita akan menggunakanya. Konkurensi di ChucK:

Sekarang mari menulis program lain (yang lebih panjang):

```
// impulse untuk filter  ke dac
Impulse i => biquad f => dac;
// aturlah kutub radius filter
.99 => f.prad;
// aturlah sama dengan nol
1 => f.eqzs;
// inisialisasi variabel float
0.0 => float v;
// batasan time-loop
while( true )
{
// mengatur impulse arusnya
1.0 => i.next;
// membersihkan frekuensi filter resonansi
Std.fabs(Math.sin(v)) * 4000.0 => f.pfreq;
// penambahan v
v + .1 => v;
// mempercepat waktu
100::ms => now;
}
```

Namakan dengan moe.ck, dan jalankan:

```
%>chuck moe.ck
```

Sekarang membuat dua salinan dari moe.ck - larry.ck dan curly.ck. membuat modifikasi berikut:

1. Mengubah percepatan waktu dari larry.ck dengan 99::ms (sebagai gantinya 100::ms).
2. Mengubah kecepatan waktu curly.ck menjadi 101::ms (dari sebelumnya 100::ms).

3. Secara optional, mengubah 4000.0 ke lainnya (contohnya 400.0 untuk curly).
   Menjalankan semua modifikasi di atas secara parallel:

   %>chuck moe.ck larry.ck curly.ck

Apa yang kamu dengar (jika berjalan lancar) tahapan antara moe, larry, dan curly, dengan pancaran curly pulsa dengan frekuensi yang rendah. ChucK mendukung sample-synchronous concurrency via pemilihan waktu mekanik dari ChucK. Diberikan beberapa source file yang menggunakan pemilihan waktu mekanik di atas, Virtual mesin (VM) ChucK dapat menggunakan informasi pemilihan waktu untuk sinkronisasi semuanya secara otomatis. Lagipula, konkurensi adalah 'sample synchronous', berarti bahwa pemilihan waktu inter-process audio dijamin menjadi sample yang tepat. Sample audio dibangkitkan oleh tiga pembantu dalam contoh ini merupakan sinkronisasi yang lengkap. Sebagai catatan bahwa tiap proses tidak perlu tahu tentang lainnya- itu hanya menyetujui dengan waktu local. VM akan meyakinkan dengan apa yang terjadi berjalan dengan benar dan global.

KONVENSI

ChucK didukung beberapa system operasi yang berbeda. Sedang kode ChucK ditujukan untuk "platform-independent", tiap sisitem operasi yang berbeda itu mempunyai"features" mereka sendiri yang membuat pengalaman bekerja dengan ChucK jadi berbeda. Bab ini akan menjelaskan garis besar beberapa perbedannya. ChucK digunakan sebagai sebuah aplikasi terminal dal tutuorial ini. Sehingga kamu akan perlu mengetahui bagaimana mengakses dan navigasi dalam terminal. Ini beberapa isyarat tentang bagaimana mendapat terminal dalam system operasimu.

OS X

Terminal berada di dalam Utilities atau folder dalam Applications atau folder di hard drivemu. Double click di terminalnya. Kamu dapat click dan tahan ikonnya dalam Dock dan pilihlah pilihan "Keep in Dock". Sekarang aplikasi terminal akan berada di Dock.

http://www.macdevcenter.com/pub/ct/51

http://www.atomiclearning.com/macosxterminalx.shtml

Windows

Terminal di akses dengan meng-klik Start Menu dan meng-klik run. Setelah window terbuka ketikan cmd.

http://www.c3scripts.com/tutorials/msdos/

http://www.ss64.com/nt/

Linux

Tidak dijelaskan disini.

Pemrograman On-the-fly

oleh Adam Tindale

Navigasilah pada folder contoh dalam distribusi ChucK lalu jalankan command berikut:

```
%>chuck moe.ck
```

Di kasus ini, ChucK akan menjalankan apapun yang ada pada moe.ck. kamu bisa mengganti moe.ck dengan nama yang lain dalam file ChucK. Jika script ini hanya berupa yang takkan pernah berhenti lalu kita mememrlukan ChucK untuk menghentikannya. Hanya dengan menenkan CTRL-C (hold control and press c). ini adalah cara cepat untuk ”kill process” dalam terminal. Hal pertama yang harus dicoba adalah tes konkurensi (menjalankan file ChucK secara parallel) yaitu moe, larry, dan curly. Pertama menjalankan mereka sendiri-sendiri (menjalankan ChucK yang moe.ck, larry.ck atau curly.ck). Lalu menjalankan mereka semua secara parallel, seperti ini:

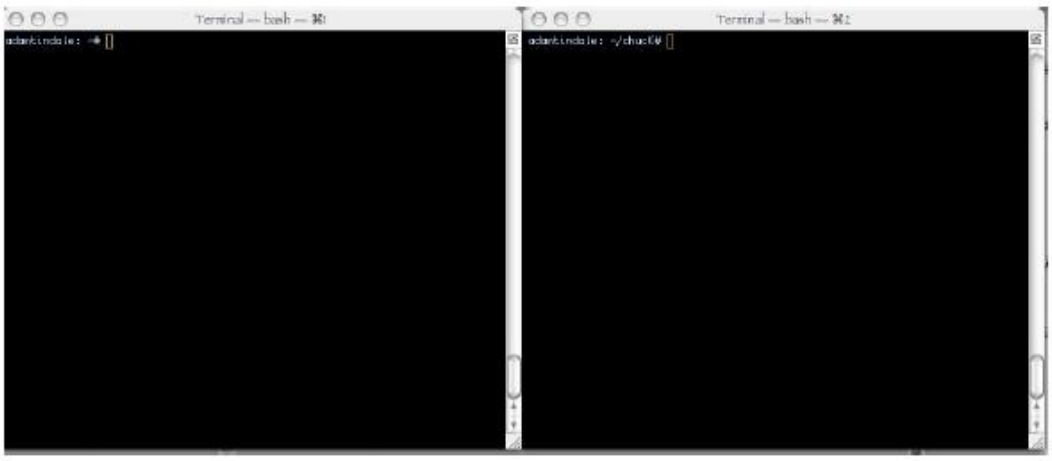
```
%>chuck moe.ck larry.ck curly.ck
```

Mereka telah ditulis dalam dan luar tahap dengan lainnya. Sekali lagi, jika beberapa script ini akan hilang selamanya tekanlah CTRL_C untuk menghentikan ChucK.

Cobalah juga versi improvisasi untuk teman kecil kita: larry++.ck curly++.ck moe++.ck

ChucK dengan Dua Window

Sekarang mari kita putar lengan baju kita sedikit dan lihat bebebrapa kekuatan ChucK sebenarnya. Kita akan menuju dua jendela ChucKdan on-the-fly. Bagian ini akan menjalankanmu menuju pada sesi dalam ChucK



Ini yang harus kamu lakukan: membuka jendela terminal lain seperti jendela yang lalu. Pada jendela yang baru ketikan:

%>chuck - -loop

Ini akan mulai menjalankan ChucK. Chuck sedang menunggu untuk melakukan sesuatu. Kembalilah ke jendela semula dimana kamu terdapat home dari ChucK. Hati-hati ya. Jika kamu mengetikkan chuck test1.ck kamu akan mulai menjalankan ChucK yang kedua tes1.ck. apa yang akan kita lakukan untuk menambah script ke ChucK yang dapat kita set menjalankannya dalam jendela yang kedua kita. Kita akan menggunakan operator + untuk manambah script kita dan operator – untuk menghapus script-nya.

%>chuck + test1.ck

%>chuck - 1

%>chuck test.ck

%>chuck test.ck

%>chuck test.ck

Apa yang terjadi? Itulah kekuatannya. Kita telah menambahkan test1.ck. hal itu telah ditambahkan sebagai potongan pertama pada ChucK kita. Sejak kita telah mengetahui potongan yang pertama tadi, kita menghapusnya dengan mengetik chuck -1. great. Selanjutnya kita menambah 3 salinan dari script yang sama, bukankah itu keren? kamu juga dapat melakukannya dengan chuck + test1.ck test1.ck test1.ck. Bagaimana kita menjejaki potongan itu?

Kamu dapat bertanya pada ChucK bagaimana dia melakukan dengan mengetikkan chuck- -status dari shortcut dan ChucK akan menjawab pada window lain dimana kita dan membiarkannya dia berjalan. ChucK akan memberitahumu potongan apa aja yang ada dan apa id number mereka. Dia juga akan mengatakan berapa lama dia berjalan. Jika kamu telah cukup dengan ChucK kamu dapat menuju ke jendela lain dengan menggunakan khayalanmu trik menekan CTRL-C atau kamu dapat ketikan chuck --kil dalam jendela asli.

```
%>chuck - -kill
```

ChucK dengan Satu Window

Satu jendela ChucK hanya untuk hardcore yang paling keras. Kamu telah diperingatkan. Konsepnya hampir sama dengan dua jendela di ChucK: pertama, mulailah ChucK kamu lalu kamu menangani penambahan dan penghapusan script di dalamnya. Bagaimana kamu memulai ChucK dan mendapatkan command prompt untuk kembali, kamu bertanya? Pada shellmu kamu dapat menambahkan sebuah ampersand (&) setelah command dan itu akan mengatakan pada shell untuk menjalankan perintah sebagai proses background dan memberi kita prompt lagi.

```
%>chuck - -loop &
```

Kamu harus berhati-hati saat menulis potongan program kamu jangan meletakkan terlalu banyak statement print. Saat kamu mencetak secara berkala kamu kehilangan control dari prompt sebagai shell mencetak. Ini akan menjadi buruk jika kamu mencetak input MIDI. Perlakuan yang sama saat kamu mengunakan perintah - -status ke ChucK. Itu juga menyenangkan untuk bertanding pada command line. Siapa yang menang?

Memodifikasi Potongan Patch

Oleh Adam Tindale

Kita mempunyai suatu potongan program dasar yang berjalan di ChucK tetapi masih terdengar tidak baik. Pada bab ini kita akan menutupi dengan beberapa jalan sederhana untuk mengoreksi masalah itu. ChucK mengijinkan satu dengan cepat membuat modifikasi ke potongan yang secara drastis dapat mengubah bunyi. Pertama yang kita dapat lakukan adalah merubah jenis osilator. Ada banyak perbedaan osilator yang tersedia untuk penggunaan: SinOsc (gelombang sinus), Sawosc (sawtooth), SqrOsc (gelombang kuadrat) dan PulseOsc (gelombang pulsa). Kita dapat dengan mudah mengubah jenis osilator seperti di bawah.

```
SawOsc s => dac;
```

Cobalah mengubah osilator ke semua jenis yang berbeda dan rasakan bagaimana suara mereka. Manakala mengubah Ugens yang berbeda selalu pastikan untuk memeriksa sisa potonganmu sedemikian sehingga pemanggilan parameternya benar. Jika kamu akan menggunakan metoda .width dari pulseOsc dan yang lain pada SinOsc ChucK akan mengeluh. Kamu dapat berkomentar ke luar tentang kerusakan yang periodik dengan penggunaan garis miring ganda (//).

Sekarang mari menambahkan beberapa efek kepada potongan program kita. ChucK mempunyai banyak perbedaan efek standard yang dapat ditambahkan ke rantai Ugen. Efek paling sederhana yang dapat kita tambahkan adalah suatu amplifier. Dalam ChucK, obyek ini disebut Gain.

```
SawOsc s => Gain g => dac;
```

Sekarang kita dapat mengubah parameter dari efek kita. Gain mempunyai suatu parameter .gain yang dapat digunakan untuk mengubah gain dari sinyal yang melintas obyek itu. Mari mengubah gain.

```
.5 => g.gain;
```

Ini berlebih-lebihan semua Ugens mempunyai kemampuan untuk mengubah gain mereka menjadi sebuah cara yang serupa. (Lihat bagian UGEN dalam referensi untuk informasi lebih tentang parameter UGEN).

```
.5 => s.gain;
```

Bagaimanapun, ini bermanfaat saat kita mempunyai berbagai Ugens menghubungkan kepada satu tempat. Jika kita akan menghubungkan 2 osilator kepada dac kemudian kita akan mendapatkan penyimpangan atau distorsi. Sebagai defaultnya, osilator ini bergerak-gerak antara - 1 dan 1. Saat mereka dihubungkan dengan input yang sama yang mereka tambahkan, maka sekarang mereka menengahi - 2 dan 2. Ini akan klip output kita. Harus berbuat apa? Gain sebagai penolong!

```
SinOsc s1 => Gain g => dac;
sinsocs s2 => g;
.5 => g.gain;
```

Sekarang osilator kita skalanya antara - 1 dan 1 dan segalanya benar di dunia ini.
Efek lebih yang dijanjikan, sekarang kamu akan lihat beberapa aksi. Sekali lagi, salah satu keajaiban ChucK betapa mudah untuk mengubah Ugens. Kita dapat mengambil potongan program di atas dan mengubah ‘Gain’ untuk ‘PRCRev’.

```
SinOsc s1 => PRCRev g => dac;
sinsocs s2 => g;
.5 => g.gain;
```

Ugen Gain telah digantikan oleh suatu reverb dan outputnya adalah skala dengan menggunakan parameter ‘.gain’ parameter yang merupakan posess dari semua Ugens. Sekarang kita dapat menambahkan beberapa rempah-rempah kepada resep itu. ‘PRCRev’ telah mempunyai parameter ‘. mix’ yang dapat diubah antara 0. dan 1. Jika kita ingin mempunyai set parameter ini menjadi nilai yang sama sebagai apa yang kita meninggalkan ke g.gain. Setelah penugasan sebuah Ugen akan mengembalikan nilai yang ditinggalkan kepadanya. Kita dapat menggunakan metoda ini ke propogate parameter untuk osilator kita.

```
.5 => g.gain => g.mix;
500 => s1.freq => s2.freq;
```

Teknik lain untuk pengaturan parameter adalah untuk membaca sebuah parameter, kemudian memodifikasinya dan kemudian ChucK kembali ke dirinya sendiri. Pengaksesan parameter memerlukan penambahan tanda-kurung () setelah nama

dari parameter. Di sini adalah suatu contoh menggandakan frekuensi dari suatu osilator.

```
s1.freq() * 2 => s1.freq;
```

Mari mengubah jenis osilator untuk tambahan kesenangan lagi. Kita dapat dengan sederhana menggantikan ‘SinOsc’ dengan osilator jenis lain. Memeriksa bagian Ugen dalam referensi gagasan. Cobalah mengubah frekuensi dari osilator dan parameter mix dari reverb untuk masing-masing jenis osilator yang kamu coba.
Jam kesenangan tak ada akhirnya!

LFOS dan Blackhole

oleh Adam Tindale

Adalah suatu teknik umum untuk menambahkan variasi ke sintesis yaitu modulasi. Modulasi adalah proses dalam mengubah sesuatu, pada umumnya parameter dari sinyal seperti frekuensi. Suatu Low Frequency Oscillator (LFO) adalah tipe yang digunakan untuk tugas ini sebab variasinya secara cukup cepat untuk menarik, sekalipun begitu melambat menjadi perceptable. Manakala suatu sinyal diatur dengan cepat (dengan kata lain. di atas 20Hz atau kira-kira segitu) hal itu cenderung untuk mengubah warna nada dari sinyal dibanding penambahan variasi.

Ok, mari menggunakan gagasan ini. Apa yang kita harus dilakukan adalah menyediakan dua osilator dan mempunyai satu pengatur parameter yang lain. ChucK tidak mendukung koneksi dari sinyal output Ugen ke parameter input. Potongan kode ini tidak akan bekerja:

```
SinOsc s => dac;
SinOsc lfo => s.freq;
```

Yang digagalkan. Apa yang kita harus lakukan adalah memberikan suara lfo kita pada beberapa tingkat yang kita putuskan, untuk sekarang kita akan memperbarui frekuensi tiap 20 seperseribu detik. Ingatlah bahwa suatu SinOsc bergerak-gerak antara -1 dan 1, sehingga jika kita baru saja meletakkan secara langsung kepada frekuensi s kita tidak bisa mendengar itu (kecuali jika kamu sedang menggunakan

ChucK dalam suatu tricked ke luar kewarganegaraan...). Apa yang kita akan lakukan adalah mengalikan keluaran dari lfo dengan 10 dan menambahkannya kepada frekuensi 440. Sekarang Frekuensi akan bergerak-gerak antara 430 dan 450.

```
SinOsc s => dac;
SinOsc lfo;
// mengatur frekuensi LFO
5 => lfo.freq;
while (20::ms => now){
( lfo.last() * 10 ) + 440 => s.freq;
}
```

ChucK adalah suatu smart little devil. Ini tidak bekerja dan sekarang kita akan mempelajari alasannya. Mengapa? Ugens dihubungkan dalam suatu jaringan dan pada umumnya diberikan kepada dac. Manakala suatu potongan program di-compile ChucK meneliti apa yang dihubungkan kepada dac dan seperti masing-masing contoh yang dihitung ChucK memeriksa jaringan Ugens dan merebut contoh yang berikutnya. Dalam hal ini, kita tidak ingin Ugen kita menghubungkan kepada dac, sekalipun begitu kita ingin ChucK untuk merebut contoh darinya. Masukkan blackhole: the sample sucker. Jika kita menghubungkan lfo kita ke blackhole segalanya akan berkembang baik.

```
SinOsc lfo => blackhole;
```

Berkutat dengan potongan program ini dalam form sekarang dan temukan nilai-nilai menarik untuk tingkatan pemberian suara, frekuensi lfo dan jumlah lfo. Cobalah mengubah Ugens agar suara lebih menarik

Bekerja dengan MIDI

Oleh Adam Tindale

Menambahkan suatu pengontrol MIDI adalah suatu cara bagus untuk menambahkan variasi ke potongan program ChucK kamu. Sebaliknya, ChucK menawarkan suatu cara kuat atau tangguh dan sederhana untuk menggunakan suatu pengontrol MIDI untuk pembuatan musik.

Yang mula-mula untuk lakukan saat bekerja dengan MIDI adalah meyakinkan bahwa ChucK lihat semua alatmu. Kamu dapat melakukan ini dengan menggunakan - - probe start flag. Seperti ini:

```
%>chuck --probe
```

Chuck akan menampilkan daftar audio yang dihubungkan dan peralatan MIDI dan referensi ID mereka. Kita akan berasumsi bahwa pengontrolmu ditemukan untuk mempunyai ID 0. Pertama, kita harus membuka suatu koneksi antara ChucK dan portnya. Kita dapat memenuhi ini dengan menciptakan suatu obyek MidiIn dan kemudian menghubungkannya ke portnya.

```
//membuat obyek
MidiIn min;
//menghubungkan ke port 0
min.open(0);
```

Jika kamu ingin mengirimkan MIDI ke luar ChucK gunakanlah obyek Midiout dan kemudian membuka port.

```
//membuat obyek
MidiOut mout;
//menghubungkan ke  port 0
mout.open(0);
```

Saat pembukaan port hal itu diusulkan bahwa kamu memeriksa apakah fungsi .open mengembalikan dengan baik. Dalam beberapa situasi hal itu tidak membuat beberapa perasaan untuk shred untuk mempertahankan hidup jika tidak ada data MIDI yang siap dikirim. Kamu dapat memeriksa nilai kembalian dari fungsi .open dan kemudian keluar dari shred menggunakan kata kunci me dengan fungsi exit().

```
MidiIn min;
min.open( 0 ) => int AmIOpen;
if( !AmIOpen ) { me.exit(); }
```

Kita dapat melakukan ini di dalam fewer lines dari kode. Kita dapat meletakkan min.open(0) di dalam statement if. Dengan cara ini min.open akan

mengembalikan nilai benar atau salah (yang mana diwakili seperti ints dengan suatu nilai 1 atau 0). Suatu tanda ! akan memberi nilai kembalian kebalikan dari min.open. Sekarang statement akan berarti jika min.open tidak mengembalikan nilai benar kemudian keluar. Yah?

```
if( !min.open(0) ) { me.exit(); }
```

Menjadi MIDI

Dalam rangka menerima beberapa data yang kamu salurkan ke dalam ChucK kita perlu membuat obyek MidiMsg. Obyek ini digunakan untuk menjaga data yang dapat masuk ke dalam ChucK atau keluaran pada port MIDI. Kecuali jika kamu mempunyai ketrampilan tinggi memanage status dari pesan ini (atau kamu menikmati sakit kepala yang kamu dapatkan dari debugging) hal itu direkomendasikan agar kamu membuat sedikitnya satu MidiMsg untuk masing-masing port yang kamu gunakan. Apa yang kita harus lakukan adalah mendapatkan data dari obyek MidiIn ke dalam sebuah pesan yang kita gunakan di dalam ChucK. Obyek MidiMsg adalah sebagai kontainer dengan tiga slot: data1, data2 dan data3. Kita memenuhi slot ini dengan meletakkan pesan kita di dalam fungsi .recv (MidiMsg) dari obyek MidiIn. MidiIn menyimpan pesannya di dalam suatu antrian sedemikian sehingga kamu dapat menanyai nya untuk beberapa pesan dan itu akan tetap memberi pesan sampai itu kosong. Fungsi .recv(MidiMsg) mengembalikan nilai benar dan salah sehingga kita dapat meletakkan nya di dalam loop untuk sementara dan itu akan terus berlanjut untuk di jalankan melalui loop sampai tidak ada lagi pesan yang ditinggalkan.

```
// memeriksa pesan tiap 10 ms
while(10::ms => now){
//
while( min.recv(msg) ){
<<<msg.data1,msg.data2,msg.data3,"MIDI Message">>>;
}
}
```

Sistem event membuat hidup menjadi sedikit lebih mudah dan juga meyakinkan bahwa input MIDI itu dihadapkan dengan secepat ChucK menerimanya. Semua

yang harus dilaksanakan oleh ChucK pada obyek MidiIn sekarang dan dia akan menunggu sampai suatu pesan diterima untuk pergi lebih lanjut di dalam program itu.

```
while(true){
// menggunakan event MIDI dari MidiIn
min => now;
while( min.recv(msg) ){
<<<msg.data1,msg.data2,msg.data3,"MIDI Message">>>;
}
}
```

Keluaran Midi

Jika kamu mempunyai suatu synthesizer yang mana kamu ingin untuk mencetuskan dari ChucK kamu dapat mengirimkan pesan MIDI kepadanya untuk sederhananya. Semua kamu harus dilakukan adalah mempunyai sebuah MidiMsg yang akan bertindak sebagai kontainer untuk datamu dan kemudian kamu akan menyampaikannya ke MidiOut menggunakan fungsi.send (MidiMsg).

```
MidiOut mout;
MidiMsg msg;
// memeriksa apakah port sudah terbuka
if( !mout.open( 0 ) ) me.exit();
// isilah data dengan data
144 => msg.data1;
52 => msg.data2;
100 => msg.data3;
// bugs setelah point ini bisa terkirim
// kepada mufaktur sari sintesismu
mout.send( msg );
```

Penulisan ke Disk

oleh Adam Tindale dan Ge Wang

Di sini adalah suatu pengajaran tambahan ringkas penulisan ke disk...

— 1. Rekamanlah sesi ChucKmu ke untuk mudahnya!

contoh: kamu ingin merekam berikut:

```
%>chuck foo.ck bar.ck
```

Semua yang harus kamu lakukan adalah shred dari ChucK yang ditulis ke file:

```
%>chuck foo.ck bar.ck rec.ck
```

tidak ada perubahan yang diperlukan pada file ada, sebuah contoh rec.ck dapat ditemukan di dalam contoh /, guy /gal ini metulis untuk “foo.wav”. Editlah file agar berubah. Jika kamu tidak ingin cemas tentang overwriting pada file yang sama, kamu dapat:

```
%>chuck foo.ck bar.ck rec2.ck
```

rec2.ck akan menghasilkan suatu nama file menggunakan waktu yang sekarang. Kamu dapat mengubah awalan dari file name dengan

```
"data/session" => w.autoPrefix;
```

w adalah Wvout di dalam potongan tersebut.

Kamu dapat tentu saja ChucK rec.ck dengan pemrograman on-the-fly...

dari terminal 1

```
%>chuck --loop
```

dari terminal 2

```
%>chuck + rec.ck
```

— 2. Silent mode

Kamu dapat menulis secara langsung ke disk tanpa mempunyai audio real-time dengan menggunakan - - silent atau – s

```
%>chuck foo.ck bar.ck rec2.ck -s
```

ini tidak akan mensinkronkan kepada kartu audio, dan akan menghasilkan contoh secepat dia dapat..

— 3. start dan stop

Kamu dapat mulai dan menghentikan penulisan ke file dengan:

```
1 => w.record; // start
0 => w.record; // stop
```

seperti dengan semua Chuckian, ini bisa dilakukan sample-synchronously.

— 4. masalah penghentikan yang lain

akibatnya bagaimana jika aku mempunyai waktu loop yang tanpa batas, dan ingin mengakhiri VM, akankah fileku dituliskan dengan tepat? jawaban:

Ctrl-C bekerja baik.

Modul STK ChucK menjaga track dari penangann file terbuka dan menutup mereka bahkan dengan penghentian abnormal, seperti Ctrl-C. Benar-benar untuk banyak, Ctrl-C adalah cara yang alami untuk mengakhiri sesi ChucKmu. Bagaimanapun juga, itu bekerja. Perihal seg-faults dan peristiwa malapetaka lain, seperti komputer yang terbakar dari kemarahan ChucK, file yang mungkin adalah roti panggang.

hmmmm, roti panggang...

— 5. silent sample sucker membentur lagi

seperti di rec.ck, satu potongan untuk menulis ke file adalah:

```
dac => Gain g => WvOut w => blackhole;
```

blackhole memandu Wvout, yang pada gilirannya menghisap contoh dari Gain dan kemudian dac itu. Wvout dapat juga ditempatkan sebelum dac:

```
Noise n => WvOut w => dac;
```

Wvout menulis ke file, dan juga menerobos contoh yang berikutnya itu.

Stereo

oleh Adam Tindale

Pada akhirnya, ChucK adalah stereo! Mengakses kemampuan stereo ChucK relatif sederhana. dac sekarang mempunyai tiga poin-poin akses.

ugen u;

```
// standard mono connection
u => dac;
// cukup mudah
u => dac.left;
u => dac.right;
```

adc kemampuannya mencerminkan dac.

```
// switcheroo
adc.right => dac.left;
adc.left => dac.right;
```

Jika kamu mempunyai jaringan UGEN yang bagus dan kamu ingin melemparkannya di suatu tempat di dalam bidang stereo kamu dapat menggunakan Pan2. Kamu dapat menggunakan fungsi .pan untuk memindahkan gerakkan bunyimu antara kiri (- 1) dan kanan(1).

```
// ini adalah sebuah hubungan stereo ke dac
SinOsc s => Pan2 p => dac;
1 => p. pan;
while(1::second => now){
// ini akan jatuh pada pan dari ke kanan
p.pan() * -1. => p.pan;
}
```

Kamu dapat juga mencampur sinyal bawah stereomu kepada mono sinyal menggunakan obyek Mix2.

```
adc => Mix2 m => dac.left;
```

Jika kamu memindahkan Mix2 di dalam rantai dan menggantikannya dengan obyek gain itu akan bertindak dengan cara yang sama. Manakala kamu menghubungkan obyek stereo ke obyek mono itu akan menjumlahkan masukan itu. Kamu akan mendapatkan efek yang sama seolah-olah kamu menghubungkan dua sinyal mono kepada masukan sinyal mono yang lain.

BAB 4

Ikhtisar

ChucK adalah suatu audio strongly-typed, strongly-timed, adalah bahasa pemrograman yang bersamaan antara audio dan multimedia bahasa program. Itu di-compile ke dalam instruksi virtual, yang mana dengan seketika dijalankan dalam Virtual Mesin ChucK. Dokumen ini sebagai pemandu fitur dari Bahasa, Compiler, dan Virtual Mesin untuk programmer ChucK.

Menjalankan ChucK

Beberapa catatan cepat:

Kamu dapat menginstal ChucK (lihatlah membangun instruksi) atau menjalankannya dari direktori lokal.

ChucK adalah suatu aplikasi command line disebut chuck. (juga lihat Audicle)

Gunakanlah command line prompt/terminal untuk menjalankan ChucK: (contoh Terminal atau xterm pada OS X, cmd atau cygwin pada Windows, pada Linux, kamu tentu saja sudah terminal lebih kamu sukai.)

Ini adalah suatu ikhtisar cepat, lihat pilihan pada VM untuk suatu pemandu yang lebih lengkap untuk pilihan command line.

Untuk menjalankan ChucK dengan suatu program atau patch memanggil foo.ck menjalankan ChucK dengan mudah dan kemudian nama dari file:

```
%>chuck foo.ck
```

Untuk menjalankan ChucK dengan berbagai program yang secara bersamaan (atau satu yang sama berulang kali):

```
%>chuck foo.ck bar.ck bar.ck boo.ck
```

Ada beberapa flag yang dapat kamu spesifikasikan untuk mengendalikan bagaimana mengoperasikan ChucK, atau untuk menemukan tentang sistemnya. Untuk contohnya, yang mengikuti probe sistem audio dan prints out semua alat audio yang tersedia dan peralatan MIDI. Kamu boleh mengacu pada mereka (dengan nomor pada umumnya) dari command line atau dari programmu. (sekali lagi, lihat pilihan VM untuk daftar yang lengkap)

```
%>chuck - -probe
```

ChucK dapat dijalankan dalam terminal yang berbeda sebagai host atau listener yang programnya boleh dikirim. Server perlu memohon flag loop untuk menetapkan bahwa virtual mesin mestinya tidak berhenti secara otomatis (saat program yang sekarang keluar).

```
%>chuck - -loop
```

(Lihat pemandu pada pemrograman On-The-Fly untuk informasi yang lebih)
Jika listener ChucK sedang berjalan, kita dapat (dari terminal kedua) mengirimkan program/patch pada listener dengan penggunaan + pilihan command line :

```
%>chuck + foo.ck
```

Dengan cara yang sama, kamu dapat menggunakan – dan = untuk menghapus atau mengganti program di listener, dan penggunaan untuk menemukan status itu. Sekali lagi, lihat pemrograman On-The-Fly untuk informasi lebih. Untuk menjalankan kebanyakan dari kodenya atau contoh dalam spesifikasi bahasa ini, kamu hanya perlu menggunakan program dasar ChucK.

Komentar

Komentar adalah bagian kode yang diabaikan oleh compiler. Bantuan ini para programmer lain (dan dirimu sendiri) menginterpretasikan kodemu. Slash ganda menunjukkan kepada compiler untuk melompati sisa dari baris. Untuk sekarang, tidak ada blok berkomentar.

```
// ini adalah komentar
int foo; // komentar lain
```

Debug Print

Untuk sementara waktu, stdout dan chout telah di-disabled untuk sementara untuk release saat ini. Di dalam tempat mereka kita sudah menyajikan suatu sintak debug print :

```
// tampilan dari nilai ekspresi
<<< expression >>>;
```

Ini akan mencetak nilai-nilai dan jenis manapun ungkapan yang ditempatkan di antara mereka. Konstruksi debug print ini mungkin ditempatkan di sekitar ungkapan manapun pada beberapa ekspresi non-declaration (non l-value) dan tidak akan mempengaruhi pelaksanaan dari kode. Ungkapan yang menghadirkan suatu obyek akan mencetak nilai (menyangkut) referensi alamat dari obyek:

```
 // masukan 5 pada deklarasi variabel yang baru
5 => int i;
// mencetak"5 : (int)"
<<<i>>>;
// mencetak "hello! : (string)"
<<<"hello!">>>;
// mencetak "3.5 : (float)"
<<<1.0 + 2.5 >>>=> float x;
```

Untuk format output data lebih lanjut, sebuah daftar pemisah koma dari ungkapan akan mencetak hanya nilai-nilai masing-masing mereka (dengan satu spasi diantaranya):

```
// mencetak "the value of x is 3.5" (x from above)
<<<"the value of x is" , x >>>;
// mencetak "4 + 5 is 9"
<<<"4 + 5 is", 4 + 5>>>;
// mencetak "here are 3 random numbers ? ? ?"
<<<"here are 3 random numbers",
Std.rand2(0,9),
Std.rand2(0,9),
Std.rand2(0,9) >>>;
```

Reserved Words

(tipe primitif)

. int

. float

. time

. dur

. void

. same (tidak diimplementasikan)

(struktur kontrol)

. if

. else

. while

. until

. for

. break

. continue

. return

. switch (tidak diimplementasikan)

(class keywords)

. class

. extends

. public

. static

. pure

. this

. super (tidak diimplementasikan)

. interface (tidak diimplementasikan)

. implements (tidak diimplementasikan)

. protected (tidak diimplementasikan)

. private (tidak diimplementasikan)

(kata kunci chuck lain)
. function
. fun
. spork
. const
. new

(nilai-nilai khusus)
. now
. true
. false
. maybe
. null
. NULL
. me
. pi

(special : default durations)
. samp
. ms
. second
. minute
. hour
. day
. week

(special : global ugens)

. dac

. adc

. blackhole

(operators)

. +

. -

. *

. /

. %

. =>

. =<

. ||

. &&

. ==

. ^

. &

. |

. ~

. ::

. ++

BAB 5
Tipe, Nilai, dan Variabel

ChucK adalah suatu bahasa strongly-typed, maksudnya jenisnya dapat dipecahkan pada compile-time. Bagaimanapun, itu adalah tidak yang sungguh statically-typed, sebab compiler atau type sistem menjadi bagian dari Virtual mesin ChucK, dan merupakan komponen runtime. Sistem jenis ini membantu ke pemaksaan ketepatan dan kejelasan dalam kode, dan secara alami meminjamkan ke organisasi dari program kompleks. Pada waktu yang sama, ini juga dinamis yang di dalam perubahan itu kepada tipe sistem dapat berlangsung (di dalam cara yang dapat dirumuskan dengan baik) pada runtime. Aspek dinamis ini membentuk basis untuk pemrograman on-the-fly.

Bagian ini berhadapan dengan jenis, nilai-nilai, dan deklarasi dan pemakaian variabel. Seperti di bahasa program strongly-typed lain, kita dapat berpikir tentang suatu jenis itu ketika dihubungkan dengan perilaku data. (Sebagai contoh, suatu ’int’ adalah suatu jenis yang berarti bilangan bulat atau integer, dan menambahkan dua bilangan bulat atau integer dikenalkan untuk menghasilkan sepertiga bilangan bulat mewakili penjumlahan.). Kelas dan obyek mengijinkan kita untuk meluaskan jenis sistem dengan jenis kebiasaan kita sendiri, tetapi kita tidak ingin menutup mereka dalam bagian ini. Kita akan memusatkan sebagian besar pada tipe primitif di sini, dan meninggalkan diskusi tentang tipe yang lebih rumit untuk obyek dan kelas.

Tipe Primitif

Adalah yang primitif, atau jenis hakiki adalah tipe data yang sederhana (mereka tidak punya atribut data tambahan). Object adalah bukan tipe primitif. tipe primitif dilalui nilai. Tipe primitif tidak bisa diperluas. Tipe primitif dalam ChucK adalah:

. int : integer (bertanda)

. float : nilai floating point (dalam ChucK, float defaultnya adalah presisi ganda)

. time : waktu ChucKian

. dur : durasi ChucKian

. void : (bukan tipe)

Untuk ringkasan operasi pada tipe ini, pergi ke operator dan operasi. Semua tipe lain diperoleh dari ’obyek’, baik sebagai bagian dari ChucK perpustakaan standard, atau sebagai suatu kelas baru yang kamu menciptakan. Untuk spesifikasi, pergi ke obyek dan kelas.

Nilai-nilai (Literal)

Nilai-Nilai literal ditetapkan dengan tegas di dalam kode dan ditugaskan suatu tipe oleh compiler itu. Berikut adalah beberapa contoh tentang nilai-nilai literal:

int:

```
42
```

int (hexidecimal):

```
0xaf30
```

float:

```
1.323
```

dur:

```
5.5::second
```

Dalam kode di atas, detik atau second adalah durasi variabel ada. Untuk durasi lainnya, lihat bagian manipulasi waktu.

Variabel

Variabel adalah penempatan di memori yang menjaga data. Variabel harus dideklarasikan di ChucK sebelum mereka digunakan. Sebagai contoh, untuk mendeklarasikan tipe variabel int memanggil foo:

```
// mendeklarasikan sebuah ‘int’ yang disebut dengan ’foo’
int foo;
```

Kita dapat menugaskan suatu nilai kepada suatu variabel ada dengan penggunaan operator (=>) ChucK. Ini merupakan salah satu menyangkut operator yang digunakan di dalam ChucK, ini adalah cara untuk bekerja dan mulai bertindak! Kita akan mendiskusikan keluarga operator ini didalam operasi dan operator.

```
// memasukkan nilai 2 untuk ’foo’
2 => foo;
```

Dimungkinkan untuk mengkombinasikan kedua statemen menjadi satu:

```
// memasukkan 3 ke dalam variable baru ’foo’ dengan mengetikkan ’int’
2 => int foo;
```

Untuk menggunakan variabelnya, hanya dengan mengacu padanya sesuai nama:

```
// nilai debug-print dari foo
<<< foo >>>;
```

Untuk memperbaharui nilai foo, contohnya adalah:

```
// mengalikan ’foo’ dengan 10, dan dimasukkan lagi pada ’foo’
foo * 10 => foo;
```

Kamu dapat juga melakukan hal di atas dengan penggunaan a * => (mult-chuck):

```
// mengalikan ’foo’ dengan 10, lalu memasukkannya pada ’foo’
10 *=> foo;
```

Di sini adalah contoh dari durasi:

```
// memasukkan nilai assign value of ’5 seconds’ to new variable bar
5::second => dur bar;
```

Sekali anda mempunyai bar, kamu dapat secara induktif menggunakannya untuk membangun durasi yang baru:

```
// 4 bar dimasukkan ke measure
4::bar => dur measure;
```

Karena waktu adalah pusat untuk memrogram ChucK, hal itu penting untuk memahami waktu, dur, hubungan dan operasi antara mereka. Ada informasi lebih di dalam bagian manipulasi waktu.

Tipe Data Acuan

Jenis acuan adalah jenis yang mewarisi dari kelas obyek. Beberapa default tipe acuan meliputi:

- Obyek: dasar tipe yang mewarisi dari semua kelas (secara langsung atau secara tidak langsung)
- array: N-Dimensional yang dipesan satuan data (dari tipe yang sama)
- Event: pokok, bisa meluas, mekanisme sinkronisasi
- UGen: generator unit bisa meluas yang berdasarkan kelas
- string: string (tentang karakter)

Kelas baru dapat diciptakan. Semua kelas adalah tipe acuan. Kita akan meninggalkan diskusi yang penuh kepada bagian kelas dan obyek.

BAB 6
Array

Larik atau array digunakan merepresentasikan N-Dimensional yang dipesan satuan data (dengan tipe yang sama). Bagian ini menetapkan bagaimana larik diumumkan dan digunakan di dalam ChucK. Beberapa catatan cepat:

- larik dapat diberi indeks dengan bilangan bulat (0-indexed).
- Beberapa array dapat juga digunakan sebagai peta assosiatif, yang diberi indeks dengan string.
- Hali itu penting untuk catatan bahwa porsi indeks integer dan porsi assosiatif dari array disimpan di dalam namespaces atau spasi nama yang terpisah.
- larik adalah obyek (lihat kelas dan obyek), dan akan bertindak dengan cara yang sama di bawah penugasan acuan dan operasi umum lainnya ke obyek.

Deklarasi

Larik dapat dideklarasi dapat dengan cara berikut:

```
//mendeklarasikan 10 array unsur dengan int, disebut foo
int foo[10];
//karena arraynya int (jenis primitif), muatan
//secara otomatis diinisialisasikan ke 0
```

Larik dapat juga diinisialisasi:

```
// penginisialisasi ChucK ke acuan array
[1, 1, 2, 3, 5, 8] @=> int foo[];
```

Di dalam kode di atas, ada beberapa hal yang perlu dicatat ;

- penginisialisasi harus berisi tipe yang sama atau serupa. compiler akan mencoba untuk menemukan tipe basis umum yang paling tinggi dari semua unsur-unsur itu. Jika tidak ada unsur umum yang ditemukan, kesalahan akan dilaporkan.
- Tipe dari penginisialisasi [1, 1, 2, 3, 5, 8] adalah int[]. Penginisialisasi adalah suatu array yang dapat digunakan secara langsung manakala larik itu diharapkan.
- pada operator (@ =>) chuck berarti penugasan, dan dibahas panjang lebar pada operasi dan operator.
- int foo[] sedang mendeklarasikan suatu acuan kosong kepada array. statemen menugaskan penginisialisasi array ke foo.
- larik adalah obyek.

Saat mendeklarasikan sebuah obyek array, obyek di dalam array secara otomatis instantiated.

```
// obyek dalam larik secara otomatis instantiated
Object group [10];
```

Jika kamu hanya ingin array dari acuan obyek:

```
// array dari acuan obyek NULL
Object @ group [10];
```

Memeriksa informasi lebih disini pada deklarasi obyek dan instantation di ChucK. Contoh di atas adalah array 1-dimensional (atau vektor). Meningkat berikutnya adalah array multi-dimensional!

Array Multi-dimensional

Itu mungkin (dan mudah) untuk mendeklarasikan array multi-dimensional:

```
// mendeklarasikan array 4, 6, 8 dengan tipe float
float foo3D[4][6][8];
```

Initializers bekerja dengan cara yang serupa:

```
// mendeklaraikan array 2 x 2 dengan tipe int
[ [1,3], [2,4] ] @=> int bar[][];
```

Dalam kode di atas, mencatat kedua [] [] sejak kita membuat sebuah matriks.

Lookup

Unsur-Unsur di dalam array dapat diakses dengan penggunaan [] (dalam jumlah yang sesuai).

```
// mendeklarasikan array dengan tipe floats
[ 3.2, 5.0, 7 ] @=> float foo[];
// mengakses elemen ke-0  (debug print)
<<< foo[0] >>>; // hopefully 3.2
// mengeset elemen kedua
8.5 => foo[2];
```

mengulang unsur-unsur dari array:

```
// sekali lagi array mengunakan tipe floats
[ 1, 2, 3, 4, 5, 6 ] @=> float foo[];
// mengulang keseluruhan array
for( 0 => int i; i < foo.cap(); i++ )
{
// melakukan sesuatu (debug print)
<<< foo[i] >>>;
}
```

Mengakses array multi-dimensional:

```
// array 2D
int foo[4][4];
// mengeset sebuah elemen
10 => foo[2][2];
```

Jika index melebihi batas-batas dari array dalam beberapa dimensi, suatu perkecualian dikeluarkan dan shred yang sekarang dihentikan.

```
// kapasitas array adalah 5
int foo[5];
// hal ini disebabkan oleh ArrayOutOfBoundsException
```

```
// mengakses 6 elemen (indeks 5)
<<< foo[5] >>>;
```

suatu program lebih panjang: otf 06.ck dari contoh:

```
// periode
.5::second => dur T;
// mensinkronisasi pada periode (untuk sinkronisasi on-the-fly)
T - (now % T) => now;
// program kita
SinOsc s => JCRev r => dac;
// inisialisasi
.05 => s.gain;
.25 => r.mix;
// skala (pentatonic; dalam semitones)
[ 0, 2, 4, 7, 9 ] @=> int scale[];
//looping waktu yang tak terbatas
while( true )
{
// mengambil sesuatu dari skala
scale[ Math.rand2(0,4) ] => float freq;
//  mendapatkan final freq
Std.mtof( 69 + (Std.rand2(0,3)*12 + freq) ) => s.freq;
// mereset phase untuk extra bandwidth
0 => s.phase;
// percepatan waktu
if( Std.randf() > -.5 ) .25::T => now;
else .5::T => now;
}
```

Array Assosiatif

Beberapa array dapat digunakan juga sebagai suatu array assosiatif, yang diberi indeks pada string. Sekali ketika array reguler instantiated, tidak ada pekerjaan lebih lanjut yang harus dilaksanakan untuk membuatnya assosiatif juga- hanya pada start dengan menggunakannya sedemikian.

```
 // mendeklarasikan array regular (kapasitas tidak terlalu menjadi masalah)
float foo[4];
// digunakan  sebagai int-based
2.5 => foo[0];
// digunakan sebagai array assosiatif
4.0 => foo["yoyo"];
// sebagai pengaksesan assosiatif (print)
<<< foo["yoyo"] >>>;
// mengakses elemen kosong
<<< foo["gaga"] >>>;
// -> harus dicetak  0.0
```

Hali itu penting untuk dicatat (lagi), bahwa ruang alamat dari membagi-bagikan bilangan bulat dan pembagian assosiatif dari array dengan pemisah secara lengkap. Contohnya adalah:

```
 // mendeklarasikan  array
int foo[2];
// meletakkan sesuatu dalam elemen 0
10 => foo[0];
// meletakkan sesuatu dalam elemen dengan "0"
20 => foo["0"];
// ini harus dicetak10 20
<<< foo[0], foo["0"] >>>;
```

Kapasitas dari suatu array menghubungkan hanya untuk porsi dari bilangan bulat darinya. Suatu array dengan suatu porsi bilangan bulat dari kapasitas 0, sebagai contoh, masih dapat mempunyai beberapa angka dari indeks assosiatif.

```
//mendeklarasikan array dengan kapasitas 0
int foo[0];
// meletakkan sesuatu dalam elemen dengan "here"
20 => foo["here"];
// ini harusnya 20
<<< foo["here"] >>>
// ini disebabkan oleh sebuah eksepsi
<<< foo[0] >>>
```

Catatan: Kapasitas assosiatif dari suatu array tidak digambarkan, maka obyek digunakan dalam ruang nama atau namespaces assosiatif harus dengan tegas instantiated, berlawanan dengan mereka yang namespaces bilangan bulat. Pengaksesan suatu unsure yang uninstantiated dari array assosiatif akan mengembalikan sebuah acuan null. Tolong periksalah halaman dokumentasi kelas untuk suatu penjelasan acuan dan obyek dari ChucK.

```
class Item {
float weight;
}
Item box[10];
// indeks integer  (up to capacity) adalah pre-instantiated.
1.2 => box[1].weight;
// instantiate element "lamp";
new Item @=> box["lamp"];
//pengaksesan diijinkan untuk "lamp"
2.0 => box["lamp"].weight;
// pengaksesan disebabkab oleh sebuah  NullPointerException
2.0 => box["sweater"].weight;
```

Operator Array

Array adalah obyek. Maka ketika kita mendeklarasikan suatu array, kita benar-benar (1) mendeklarasikan sebuah acuan ke array (variabel acuan) dan (2) instantiating suatu array baru dan menugaskan acuan kepada variabel itu. Suatu acuan yang (null) adalah suatu variabel acuan yang tidak menunjuk pada obyek atau null. Suatu null acuan pada suatu array dapat diciptakan di dalam fasion ini:

```
 // mendeklarasikan acuan array (dengan tidak menetapkan kapasitasnya)
int foo[];
// kita sekarang dapat menugaskan beberapa int[] untuk foo
[ 1, 2, 3 ] @=> foo;
// mencetak elemen ke-0
<<< foo[0] >>>;
```

Ini adalah juga bermanfaat dalam mendeklarasikankan fungsi yang mempunyai array sebagai argumentasi atau mengembalikan tipe.

```
//fungsi kita
fun void print( int bar[] )
{
// cetaklah
for( 0 => int i; i < bar.cap(); i++ )
<<< bar[0] >>>;
}
// kita dapat memanggil fungsi dengan sebuah literal
print( [ 1, 2, 3, 4, 5 ] );
// atau kita dapat melewati sebiah variabel acuan
int foo[10];
print( foo );
```

Seperti obyek lain, hal itu dimungkinkan membuat berbagai acuan ke array tunggal. Seperti obyek lain, semua penugasan adalah penugasan acuan, maksudnya adalah muatan tidak dicopy, hanya suatu acuan ke array disalin.

```
//array tunggal kita
int the_array[10];
// menugaskan acuan untuk  foo dan bar
the_array => int foo[] => int bar[];
// (the_array, foo, dan bar sekarang semua acuan adalah array yang sama)
// kita mengubah the_array dan mrncetak foo...
// mereka mereferensikan array yang sama, mengubah satu sama saja dengan mengubah yang lain
5 => the_array[0];
<<< foo[0] >>>; // should be 5
```

Itu dimungkinkan untuk mereferensikan sub-bagian array multi-dimensional.

```
// sebuah array 3D
int foo3D[4][4][4];
// kita dapat membuat sebuah referensi ke sebuah sub-bagian
foo3D[2] => int bar[][];
// (catatan bahwa dimensi harus ditambahkan!)
```

BAB 7

Operator dan Operasi

Operasi pada data dicapai melalui operator. Bagian ini menggambarkan bagaimana operator bertindak pada berbagai tipe data. Kamu mungkin telah melihat banyak dari operator di dalam bahasa program lain (C atau Java). Beberapa yang asli lainnya untuk ChucK. Kita mulai dengan keluarga operator ChucK.

=> (operator ChucK)

Operator ChucK (=>) adalah suatu operator yang dimuati secara berlebihan, tergantung pada jenis yang terlibat, melaksanakan berbagai tindakan. Itu menandakan tindakan, dapat dihubungkan, dan memaksakan dan menjelaskan pesanan (selalu berangkat dari kiri ke kanan). Operator ChucK berarti bahwa pekerjaan dilakukan dalam ChucK. Lagipula, operator ChucK adalah bukan operator tunggal, hanyalah suatu keluarga dari operator.

=> (foundational ChucK operator)

Kita mulai dengan standard itu, operator ChucK plain-vanilla (=>). Itu left-associative (semua operator ChucK), yang mana mengijinkan kita untuk menetapkan beberapa arus dipesan data atau tasks atau modules (seperti koneksi unit generator) dari left-to-right, seperti yang tertulis dalam teks (Bahasa Inggris). Apakah => yang tergantung pada konteks itu. Itu selalu tergantung pada jenis dari entity pada sisi kiri (chucker) dan satu pada sisi kanan (chuckee), dan itu kadang-kadang juga tergantung pada sifat alami entity (seperti apakah ini merupakan suatu variabel atau bukan).

Beberapa contoh:

```
// sebuah program unit generator – arus sinyal adalah nyata
// (dalam kasus ini, => menghubungkan dua unit generator)
SinOsc b => Gain g => biquad f => dac;
```

```
// menambahkan 4 pada foo, hasil chuck untuk  'int' variabel 'bar' yang
baru
// (dalam kasus ini, => menugaskan sebuah value pada variabel (int)
4 + foo => int bar;
// nilai chuck diberikan pada function == pemanggilan fungsi
// (sama dengan Math.rand2f( 30, 1000))
(30, 1000 ) => Math.rand2f;
```

Ada banyak penggunaan yang dirumuskan dengan baik menyangkut operator ChucK, tergantung pada konteks itu. @=> (penugasan secara eksplisit operator ChucK). Di dalam ChucK, tidak ada standar operator penugasan(=), yang ditemukan dalam banyak bahasa pemrograman lain. Penugasan dilaksanakan dengan menggunakan operator ChucK. Pada contoh yang sebelumnya, kita sudah menggunakan => untuk penugasan:

```
 // menugaskan 4 pada variabel foo
4 => int foo;
// menugaskan 1.5 untuk variabel bar
1.5 => float bar;
// menugaskan durasi  100 millisecond untuk duh
100::ms => dur duh;
// assign the time "5 second from now" to later
5::second + now => time later;
```

@ => adalah operator ChucK untuk penugasan secara eksplisit yang persisnya bertindak sama untuk tipe di atas (int, float, dur, time). Bagaimanapun, perbedannya adalah bahwa @=> dapat juga digunakan untuk penugasan referensi dari obyek (lihat kelas dan obyek) sedangkan => hanya mengerjakan penugasan pada tipe primitif (int, float, dur, time). Perilaku dari => pada obyek adalah sepenuhnya context-dependent atau bergantung pada konteks.

```
// menggunakan @=> sama seperti => untuk tipe primitif
4 @=> int foo;
// menugaskan 1.5 untuk variabel bar
1.5 @=> float bar;
```

```
// (hanya @=> dapat menunjukkan penugasan referensi pada obyek)
// menugaskan acuan moe untuk larry
// (seperti keduanya acuan moe dan larry adalah obyek yang sama)
Object moe @=> Object @ larry;
// inisialisasi array
[ 1, 2 ] @=> int ar[];
// menggunakan yang baru
new Object @=> moe;
```

Dalam caranya screwed-up miliknya, ini adalah jenis yang manis sebab tidak ada kebingungan antara penugasan (@=> atau =>) dan persamaan(==). Sesungguhnya program berikut bukan tidak statemen ChucK yang valid:

```
// bukan statemen ChucK yang valid
int foo = 4;
```

+=> -=> *=> /=> etc. (operator ChucK aritmatika)

Operator ini digunakan dengan variabel (menggunakan ’int’ dan ’float’) untuk melaksanakan satu operasi dengan penugasan.

```
// menambahkan  4 untuk foo dan menugaskan hasil pada foo
foo + 4 => foo;
// menambahkan  4 untuk foo dan menugaskan hasil pada foo
4 +=> foo;
// mengurangi  10 dari foo dan menugaskan hasilnya untuk foo
// ingatlah ini adalah (foo-10), bukan (10-foo)
10 -=> foo;
// 2 kali foo menugaskan hasil untuk foo
2 *=> foo;
// membagi 4 dari foo dan menugaskan hasilnya foo
// ingatlah lagi ini adalah (foo/4), bukan (4/foo)
4 /=> foo;
```

Ini penting untuk mencatat hubungan antara variabel dan nilai manakala penggunaan -=> dan atau =>, karena operasi ini adalah tidak komutatif.

```
// mod dari foo oleh T dan menugaskan hasilnya untuk foo
T %=> foo;
// bitwise AND 0xff dan bar dan mengaskan hasilnya untuk bar
0xff &=> bar;
// bitwise OR 0xff dan bar dan menugaskan hasilnya untuk bar
0xff |=> bar;
```

Itu cukup memungkinkan penggunaan operator yang salah untuk sekarang...

+ - * / (aritmatika)

Dapatkah kamu menambahkan, mengurangi, mengalikan dan membagi? Maka dapat ChucK!

```
// pembagian (dan penugasan)
16 / 4 => int four;
// perkalian
2 * 2 => four;
// penambahan
3 + 1 => four;
// pengurangan
93 - 89 => four;
```

Cast

ChucK secara implisit mengubah nilai int menjadi float adalah diharapkan, tetapi bukan di sekeliling yang lain. Yang terakhir dapat mengakibatkan hilangnya informasi dan memerlukan suatu cast atau mengubah yang eksplisit.

```
//menambahkan  float dan int menghasilkan  float
9.1 + 2 => float result;
// bagaimanapun, perubahan dari float ke int memerlukan cast
4.8 $ int => int foo; // foo == 4
// fungsi ini mengharapkan dua  float
Math.rand2f(30.0, 1000.0);
// ini adalah bagus karena cast yang implisitcast
Math.rand2f(30, 1000);
```

% (modulo)

Operator modulo % menghitung sisa setelah integer, floating point, durasi, dan time atau divisi durasi.

```
// 7 mod 4 (hasil seharusnya 3)
7 % 4 => int result;
// 7.3 mod 3.2 floating point mod (hasil seharusnya .9)
7.3 % 3.2 => float resultf;
// mod durasi
5::second % 2::second => dur foo;
// mod waktu/durasi
now % 5::second => dur bar;
```

Belakangan ini (mod waktu atau durasi) adalah satu dari banyak jalan menuju ke sinkronisasi pemilihan waktu yang dinamis di dalam shred. Contohnya otf 01.ck sampai otf 07.ck (lihat contoh di bawah) menggunakan ini ke sinkronisasi on-the-fly berbagai komponennya, kapanpun juga masing-masing shred ditambahkan kepada virtual mesin:

```
// mendefinisikan periode (yang disetujui oleh beberapa  shred)
.5::second => dur T;
// menghitung sisa dari periode sekarang
// dan mempercepat waktu dengan jumlahnya
T - (now % T) => now;
// saat kita menjangkau titik ini, kita mensinkronisasikan untuk batasan periode T
// kode yang istirahatthe rest of the code
// ...
```

Ini adalah satu dari banyak jalan untuk menghitung dan memberi alasan sekitar waktu di dalam ChucK. Solusi yang sesuai pada setiap kasus tergantung pada kemampuan yang diharapkan itu. Bersenang-senanglah!

&& || == != > < => >= (logika)

Operator logika masing-masing kebutuhan ini dua operand. Hasil adalah suatu bilangan bulat nilai 0 atau 1.

- && : dan
- || : atau
- == : sama dengan
- != : tidak sama dengan
- > : lebih besar dari
- >= : lebih besar atau sama dengan
- < : lebih kecil dari
- <= : lebih kecil sama dengan

```
// tes beberapa kebenaran umum
if( 1 <= 4 && true )
<<<"horray">>>;
```

>> << & | ^ (bitwise)

Ini digunakan pada nilai int pada tingkatan bit, sering untuk penutup bit.

- >>: shift bits right (8 >>1 = 4)
- <<: shift bits left (8 <<1 = 16)
- & : bitwise AND
- | : bitwise OR
- ^ : bitwise XOR

++ - - (inc / dec)

Nilai - nilai mungkin dinaikkan atau diturunkan dengan menambahkan operaor ++ atau - - .

```
4 => int foo;
foo++;
foo--;
```

! + - baru (unary)

Operator ini datang di depan satu operand.

```
 //pembalik logika
if( !true == false )
<<<"yes">>>;
// negative
-1 => int foo;
// instantiate object
new object @=> object @ bar;
```

BAB 8
Struktur Kontrol

Struktur Kontrol ChucK meliputi standard yang mengendalikan struktur yang serupa pada kebanyakan bahasa program. Suatu kondisi (dari tipe ’int’) dievaluasi dan kemudian suatu blok kelanjutan berpotensi dieksekusi. Blok dipisahkan baik oleh titik koma maupun oleh tanda-kurung kurawal.

If / Else

Statemen if memgeksekusi suatu blok jika kondisi dievaluasi adalah bukan nol.

```
if( kondisi)
{
// memasukkan/menyisipkan kode di sini
}
```

Pada kode di atas, kondisi adalah beberapa ungkapan yang mengevaluasi suatu int. Statemen else dapat diletakkan setelah blok if untuk menangani kasus di mana kondisi mengevaluasi 0.

```
if( kondisi)
{
// kode mu di sini
}
else
{
// kodemu yang lain di sini
}
```

Jika statemen tersarang.

While

Statemen while adalah suatu loop yang berulang-kali mengeksekusi badan sepanjang kondisi mengevaluasi bukan nol.

```
 // ini adalah loop tak terbatas
while( true )
{
// kode loopmu selamanya
// (kadang ini diinginkan karena kita dapat membuat
// ini adalah cara loop dengan waktu tak terbatas dan karena kita mempunyai
// konkurensi
}
```

While loop akan memeriksa kondisi pertama kali, dan mengeksekusi badan sepanjang kondisi mengevaluasi bukan nol. Untuk mengeksekusi badan dari loop sebelum mengecek kondisi, kamu dapat menggunakan suatu do atau while loop. Ini menjamin untuk badan yang dieksekusi paling sedikit sekali.

```
do {
// kodemu dieksekusi paling sedikit sekali
} while( condition );
```

Beberapa poin-poin:

- Statemen while dapat dibuat bersarang.
- Memeriksa break atau continue untuk kendali yang tambahan dengan teliti pada loopmu.

Until

Statemen until adalah kebalikan dari while statemen, secara semantis. Suatu loop until yang berulang-ulang mengeksekusi badan sampai kondisi mengevaluasi bukan nol.

```
// sebuah loop tak terbatas
until( false )
{
```

```
// kode terbaikmu loop selamanya
}
```

While loop itu akan cek pertama adalah kondisi, dan mengeksekusi badan sepanjang kondisi mengevaluasi nol. Untuk mengeksekusi badan dari loop sebelum mengecek kondisi, kamu dapat menggunakan suatu do atau until loop. Ini menjamin yang mengeksekusi badan paling sedikit sekali.

```
do {
// kodemu dieksekusi paling sedikit sekali
} until( condition );
```

Beberapa poin-poin:

Statemen until dapat dibuat bersarang.

Memeriksa break atau continue untuk kendali yang tambahan dengan teliti loopmu.

For

Suatu loop yang iterasinya ditentukan banyaknya. Suatu variabel temporari dideklarasikan itu menjejaki track dari index sekarang dan dievaluasi dan incremen pada masing-masing iterasi berulang-ulang.

```
//untuk loop
for( 0 => int foo; foo < 4 ; foo++ )
{
// debug-print nilai dari ’foo’
<<<foo>>>;
}
```

Break / Continue

Break mengijinkan program mengalir untuk melompat dari loop.

```
// loop tak terbatas
while( 1 )
```

```
{
if( condition )
break;
}
```

Continue mengijinkan sebuah loop untuk continue looping tapi tidak mengeksekusi iterasi sisa dari blok for di mana continue dieksekusi.

```
// loop tak terbatas lainnya
while( 1 )
{
// cek kondisi
if( condition )
continue;
// beberapa kode hebat yang mungkin diinginkan (jika continue dijalankan)
}
```

BAB 9
Fungsi

Fungsi menyediakan jalan untuk menghentikan kode atau tugas umum atas ke dalam unit individu. Ini membantu untuk mempromosikan kode re-usability dan keadaan dapat dibaca.

Writing

Mendeklarasikan fungsi dengan kata kunci fun (atau fungsi) yang diikuti oleh tipe pengembalian dan kemudian nama dari fungsi. Setelah nama dari fungsi tanda kurung harus dibuka untuk mendeklarasikan tipe dari argumentasi masukan.

```
// mendefinisikan fungsi pemanggilan ’funk’
fun void funk( int arg )
{
// masukan kode disini
}
```

Fungsi di atas tidak mengembalikan nilai-nilai (tipe pengembaliannya adalah void). Jika fungsi dideklarasikan untuk mengembalikan beberapa nilai tipe lain, fungsi badan harus mengembalikan suatu nilai yang menyangkut tipe yang sesuai itu.

```
 // mendefinisikan fungsi ’addOne’
fun int addOne(int x)
{
// hasil
return x + 1;
}
```

Calling

Untuk pemanggilan suatu fungsi menggunakan nama dari fungsi dengan argumen yang sesuai.

```
// mendefinisikan ’hey’
fun int hey(int a, int b)
{
// melakukan sesuatu
return a + b;
}
// memanggil fungsi; menyimpan hasil
hey(1, 2) => int result;
```

Kamu dapat juga menggunakan operator ChucK untuk memanggil fungsi!

```
// memanggil hey
(1, 2) => hey => int result;
// sama
hey(1, 2) => int result;
// beberapa dalam satu baris
(10, 3000) => Std.math => Std.abs => float foo;
// sama
Std.abs(Std.math(10, 3000)) => float foo;
```

Overloading

Overloading atau pemuatan adalah suatu fungsi mengijinkan fungsi dengan nama yang sama untuk digambarkan dengan argumentasi yang berbeda. Fungsi harus ditulis dalam memisahkan instance untuk menangani masukan, dan tipe pengembalian harus setuju.

```
// funk(int)
fun int add(int x)
{
```

```
return x + x;
}
// funk(int, int)
fun int add(int x, int y)
{
return x + y;
}
// kompilator secara otomatis memilih salah satu untuk dipanggil
add( 1 ) => int foo;
add(1, 2) => int bar;
```

BAB 10
Konkurensi dan Shred

ChucK bisa menjalankan banyak proses secara bersamaan (proses bertindak seolah-olah mereka sedang menjalankan di dalam paralel). Suatu proses Chuckian disebut suatu shred. To spork a shred mempunyai arti menciptakan dan menambahkan suatu proses baru kepada mesin virtual itu. Shred mungkin sporked dari berbagai tempat, dan boleh dari diri mereka spork shred baru.

ChucK mendukung sample-synchronous, non-preemptive concurrency. Via mekanisme pemilihan waktu. Beberapa program atau shred dapat secara otomatis dijadwalkan dan disinkronisasi menggunakan informasi pemilihan waktu. Suatu penjadwalan di dalam virtual mesin mengerjakan penjadwalan itu. Concurrency adalah 'sample synchronous', maksudnya bahwa pemilihan waktu inter-process audio adalah dijamin menjadi tepat untuk contoh itu. Catatan bahwa masing-masing process atau shred tidak perlu harus memahami tentang satu sama lain- itu hanya harus berhadapan dengan waktu lokal. Virtual mesin akan meyakinkan hal-hal terjadi dengan tepat "across the board". Akhirnya, konkurensi-seperti pemilihan waktu- adalah deterministik di dalam ChucK.

Cara yang paling sederhana untuk menjalankan shred secara bersamaan akan menetapkan merekanya pada command line:

```
%>chuck foo.ck bar.ck boo.ck
```

Perintah di atas untuk menjalankan chuck yang foo.ck, bar.ck, dan boo.ck secara bersamaan. Ada cara-cara lain untuk menjalankan shred secara bersamaan (lihat perintah pemrograman on-the-fly). Berikutnya, kita menunjukkan bagaimana untuk menciptakan shred baru dari dalam program ChucK.

Sporking Shred (dalam Kode)

Spork berarti menjadwalkan ke suatu shred baru.

Spork suatu shred, menggunakan kata kunci atau operator spork:

Spork secara dinamis sporks shred dari suatu pemanggilan fungsi

Operasi ini adalah sample-synchronous, shred baru adalah dijadwalkan untuk menjalankan dengan seketika

Shred orang tua melanjutkan untuk menjalankan, sampai beberapa waktu ke depan (lihat manipulasi waktu) atau sampai orangtua mendapat hasil dengan tegas (lihat bagian berikutnya).

Dalam implementasi yang sekarang, saat shred orang tua keluar, semua shred anak keluar juga semuanya (perilaku ini akan ditingkatkan di masa datang.)

Sporking suatu fungsi mengembalikan acuan kepada shed yang baru, catatan bahwa operasi ini tidak mengembalikan apa yang dikembalian fungsi - kemampuan untuk mendapat balasan nilai kembalian pada beberapa titik kemudiannya pada waktunya akan disiapkan dalam pelepasan atau release dimasa depan.

```
//mendefinisikan fungsi go()
fun void go()
{
// memasukkan kode
}
// spork sebuah shred baru untuk menjalankan dari fungsi go()
spork ~ go();
// spork lainnya, menyimpan referensi untuk shred baru dalam offspring
spork ~ go() => Shred @ offspring;
```

Suatu contoh yang sedikit lebih panjang:

```
// mendefinisikan fungsi
fun void foo( string s )
{
//loop dengan waktu tak terbatas
while( true )
```

```
{
// mencetak s
<<< s >>>;
// percepatan waktu
500::ms => now;
}
}
// spork shred, melewati "you" sebagai argumen untuk foo
spork ~ foo( "you" );
// mempercepat waktu dengan  250 ms
250::ms => now;
// spork shred yang lain
spork ~ foo( "me" );
// loop dengan waktu tak terbatas untuk menjaga shred
while( true )
1::second => now;
```

Lihat juga bagian fungsi untuk lebih banyak informasi pada bekerja dengan fungsi.

Kata Kunci ’me’

Kata kunci ’me’ (tipe shred) menunjukkan shred yang sekarang itu. Kadang-kadang berguna bagi memenjarakan shred yang sekarang tanpa mempercepat waktu, dan membiarkan shred lain dijadwalkan untuk waktu yang sekarang untuk menjalankannya. me.yield() mengerjakan secara persis. Ini sering bermanfaat secara seketika setelah sporking shred baru, dan kamu menginginkan shred itu mempunyai kesempatan untuk menjalankan tetapi kamu tidak ingin mempercepat waktu sekalipun untuk kamu sendiri.

```
// spork shred
spork ~ go();
// memenjarakan shred yang sekarang
```

```
// ... memberikan shred lain (shreduled untuk ’now’) sebuah kesempatan
untuk menjalankan
me.yield();
```

Mungkin juga akan berguna untuk mengeluarkan shred yang sekarang itu. Sebagai contoh jika suatu peralatan MIDI gagal untuk terbuka, kamu boleh mengeluarkan shred yang sekarang itu.

```
//membuat sebuah obyek MidiIn
MidiIn min;
// mencoba untuk membuka peralatan 0 (chuck – memeriksa untuk
mendaftar semua peralatan)
if( !min.open( 0 ) )
{
// mencetak pesan kesalahan
<<< "can’t open MIDI device" >>>;
// keluar dari shred sekarang
me.exit();
```

Kamu dapat mendapatkan id:

```
// mencetak shred id
<<< me.id(); >>>;
```

Fungsi ini adalah umum untuk semua shred, tetapi yield() dan exit() biasanya digunakan dengan shred yang sekarang itu.

Menggunakan Machine.Add()

Machine.add (string path) mengambil alur untuk program chuck, dan sporks itu. Tidak sama dengan spork, tidak ada hubungan parent-child antara shred yang memanggil fungsi dan shred baru yang ditambahkan. Ini bermanfaat untuk secara dinamis menjalankan program disimpan.

```
// spork "foo.ck"
Machine.add( "foo.ck" );
```

Segera, ini akan mengembalikan id dari shred baru, bukan sebuah acuan shred itu. Ini akan mungkin diubah di masa datang. Dengan cara yang sama, kamu dapat memindahkan shred dari virtual mesin.

```
// menambahkan
Machine.add( "foo.ck" ) => int id;
//memindahkan shred dengan id
Machine.remove( id );
// menambahkan
Machine.add( "boo.ck" ) => id
// mengganti shred dengan "bar.ck"
Machine.replace( id, "bar.ck" );
```

Komunikasi Inter-shred

Shred sporked di dalam file yang sama dapat berbagi variabel global yang sama. Mereka dapat menggunakan event dan waktu untuk mensinkronkan ke satu sama lain. (lihat event) shred sporked dari file yang berbeda dapat berbagi data (termasuk event). Untuk sekarang, ini dilaksanakan melalui suatu kelas publik dengan data statis (lihat kelas). Data statis adalah tidak sepenuhnya diterapkan! Kita akan memperbaiki ini dengan segera!

BAB 11

Waktu dan Pemilihan Waktu

Chuck adalah suatu bahasa *strongly-timed*, artinya adalah pada dasarnya ditempelkan pada atau didalam suatu bahasa. Chuck mengijinkan programmer untuk dengan tegas memberi alasan tentang waktu dari kode tersebut. Ini memberi kendali yang sangat tepat dan fleksibel dari waktu ke waktu dan (oleh karena itu) bunyi sintesis.

Di (dalam) Chuck :

- waktu dan durasi adalah tipe asli pada bahasa ini
- kata kunci *now* [memegang/menjaga] waktu logis yang umum
- waktu adalah kelebihan dari manipulasi
- kita mempunyai kendali yang tepat dan fleksibel

Waktu dan Durasi (Jangka Waktu)

Waktu dan durasi adalah tipe asli di (dalam) Chuck. Waktu menyatakan suatu titik mutlak pada / di dalam waktunya (dari permulaan Chuck). dur menyatakan suatu durasi (dengan logika yang sama sebagai waktu).

```
// sebuah durasi untuk satu detik
1::second => dur foo;

// a point in time (duration of foo from now)
now + foo => time later;
```

Pada bagian ini, kita menguraikan secara singkat berbagai operasi perhitungan untuk melaksanakan atau menjalankan waktu dan durasi.

Suatu durasi dapat digunakan untuk membangun suatu durasi baru, yang kemudian digunakan untuk secara induktif membangun durasi lainnya. Sebagai contoh:

```
// .5 second is a quarter
.5::second => dur quarter;

// 4 quarters is whole
4::quarter => dur whole;
```

Secara *default*, Chuck menyediakan nilai-nilai durasi yang telah ditetapkan yaitu :

- samp : salah satu contoh durasi di (dalam) waktu Chuck
- ms : durasi dari 1 millidetik
- second : durasi dari 1 detik
- minute : 1 menit
- hour : 1 jam
- day : 1 hari
- week : 1 minggu

Berikut ini digunakan untuk menyatakan berbagai durasi.

```
// the duration of half a sample
.5::samp => dur foo;

// 20 weeks
20::week => dur waithere;
```

// use in combination

2::minute + 30::second => dur bar;

// same value as above

2.5::minute => dur bar;

Operasi yang terdapat di dalam Waktu dan Durasi (Aritmatika)

Di (dalam) Chuck, ada operasi perhitungan yang dirumuskan dengan baik untuk memberikan nilai-nilai pada tipe waktu dan dur.

Contoh 1 (offset waktu):

// time + dur yields time

now + 10::second => time later;

Contoh 2 (pengurangan waktu) :

// time - time yields dur

later - now => dur D;

Contoh 3 (penambahan) :

// dur + dur yields dur

10::second + 100::samp => dur foo;

Contoh 4 (pengurangan) :

// dur - dur yields dur

10::second - 100::samp => dur bar;

Contoh 5 (pembagian) :

// dur / dur yields number

10::second / 20::ms => float n;

Contoh 6 (sisa hasil pembagian waktu) :

// time mod dur yields dur

now % 1::second => dur remainder;

Contoh 7 (mensikronisasikan ke periode) :

// synchronize to period of .5 second

.5::second => dur T;

T - (now % T) => now;

Contoh 8 (perbandingan waktu) :

// compare time and time

if(t1 < t2)

// do something...

Contoh 9 (perbandingan durasi) :

// compare dur and dur

if(900::ms < 1::second)

<<< "yay!" >>>;

Kata Kunci ‘now’

Kata kunci ‘*now*’ adalah kunci untuk memikirkan dan mengendalikan waktu di dalam Chuck.

Beberapa properti pada ‘***now’*** meliputi :

- ‘*now*’ adalah suatu variabel spesial pada tipe waktu
- ‘*now*’ menahan waktu Chuck pada suatu saat (ketika membaca)
- mengubah atau memodifikasi ‘*now*’ dapat menyebabkan :
  - mempercepat waktu (lihat dibawah)
  - menghentikan proses yang sedang berjalan saat itu (disebut juga sebagai potongan) sampai waktu yang diinginkan tercapai
  - potongan dan sintesis audio di hitung
  - nilai dari ‘*now*’ hanya berubah ketika dimodifikasi secara terperinci.

(lihat juga bagian berikutnya untuk mempercepat waktu).

Contoh :

// compute value that represents "5 seconds from now"

now + 5::second => time later;

```
// while we are not at later yet...
while( now < later )
{
    // print out value of now
    <<< now >>>;

    // advance time by 1 second
    1::second => now;
}
```

Mempercepat Waktu

Mempercepat waktu mengijinkan potongan lain (proses) untuk menjalankan dan mengijinkan audio untuk dihitung adalah suatu cara dikendalikan. Ada tiga jalan mempercepat waktu di dalam Chuck :

- Menchuck (=>) suatu durasi untuk sekarang: ini akan membantu mempercepat waktu oleh durasi itu.
- Menchuck (=>) suatu waktu untuk sekarang: ini akan membantu mempercepat waktu untuk titik itu. (catatan bahwa waktu yang diinginkan harus dibandingkan dengan waktu yang sekarang, atau sedikitnya memadai atau sama dengan itu).
- Menchuck (=>) suatu event untuk sekarang: waktu akan membantu sampai event dicetuskan. (lihat juga event).

Mempercepat Waktu dengan Durasi

```
// advance time by 1 second
1::second => now;
// advance time by 100 millisecond
100::ms => now;
// advance time by 1 samp (every sample)
1::samp => now;
```

```
// advance time by less than 1 samp
.024::samp => now;
```

Mempercepat Waktu dengan Waktu Mutlak

```
// figure out when
now + 4::hour => time later;
// advance time to later
later => now;
```

Suatu waktu yang dichuck-kan untuk sekarang akan mempunyai masa tunggu chuck sampai waktu yang ditetapkan [itu]. Chuck tidak menunjukkan kehilangan (kecuali jika rusak)! Lagi, waktunya menchuck untuk sekarang harus lebih besar dari atau sepadan dengan sekarang, jika tidak suatu perkecualian diberikan.

Mempercepat Waktu dengan Event

```
// wait on event
e => now;
```

Lihat peristiwa untuk suatu diskusi yang lebih lengkap menggunakan peristiwa! Kemajuan waktu dapat terjadi pada titik manapun di dalam kode itu.

```
// our patch: sine oscillator -> dac
SinOsc s => dac;
// infinite time loop
while( true )
{
    // randomly choose frequency from 30 to 1000
    Std.rand2f( 30, 1000 ) => s.freq;
    // advance time by 100 millisecond
    100::ms => now;
}
```

Lagipula, tidak ada pembatasan (selain dari mendasari ketepatan titik-dasar yang mengapung) pada beberapa waktu dipercepat. Maka adalah mungkin untuk membantu waktu oleh suatu microsecond, suatu samp, 2 jam, atau 10 tahun. Sistem oleh sebab itu akan bertindak dan *deterministically*.

Mekanisme ini mengijinkan waktu untuk dikendalikan pada tingkat nilai manapun yang diinginkan, menurut pola *programmable* manapun. Berkenaan dengan sintesis bunyi, mungkin untuk mengendalikan unit generator manapun, sesunguhnya pada tingkat nilai manapun, bahkan tingkat nilai contoh bagian.

Kekuatan mekanisme pemilihan waktu diperluas oleh kemampuan untuk menulis kode paralel, di mana dibahas *concurrency* dan potongan.

BAB 12

Event

Sebagai tambahan terhadap pembangunan mekanisme pemilihan waktu untuk pengawasan intern, Chuck mempunyai suatu kelas event untuk mengijinkan sinkronisasi secara tepat ke seberang suatu jumlah potongan secara sembarangan.

Apakah itu

Chuck event adalah suatu kelas asli di dalam bahasa Chuck. Kita dapat menciptakan suatu *object* event, dan kemudian menchuck (=¿) peristiwa itu untuk sekarang. Event menempatkan potongan pada daftar tunggu event, menghentikan potongan yang sekarang (waktu pembiaran membantu dari segi pandangan potongan (*shred's*)). Ketika event dicetuskan, satu atau lebih potongan berada pada daftar tunggunya adalah untuk dijalankan dengan seketika. Picu ini boleh di mulai dari potongan Chuck yang lain, atau dari aktivitas yang berlangsung di luar Mesin virtual itu (MIDI, OSC, atau IPC).

```
// declare event
Event e;
// function for shred
fun void eventshred( Event event, string msg )
{
    // infinite loop
    while ( true )
    {
        // wait on event
```

```
            event => now;
            // print
            <<<msg>>>;
        }
}

// create shreds
spork ~ eventshred ( e, "fee" );
spork ~ eventshred ( e, "fi" );
spork ~ eventshred ( e, "fo" );
spork ~ eventshred ( e, "fum" );
// infinite time loop
while ( true )
{
        // either signal or broadcast
        if( maybe )
        {
            <<<"signaling...">>>;
            e.signal();
        }
        else
        {
            <<<"broadcasting...">>>;
            e.broadcast();
        }

        // advance time
        0.5::second => now;
}
```

Penggunaan

Menchuck suatu event dengan menghentikan potongan pada saat itu, membiarkan waktu dipercepat :

```
// deklarasi event
Event e;
// ...
// menunggu dalam event
e => now;
// after the event is trigger
<<< "I just woke up" >>>;
```

seperti ditunjukkan di atas, event dapat dicetuskan melalui dua arah, tergantung pada perilaku yang diinginkan.

```
// signal one shred waiting on the event e
e.signal();
```

sinyal() melepaskan potongan yang pertama pada antrian event, dan menjalankan di waktu yang sekarang, merespon perintah di mana potongan telah ditambahkan kepada antrian.

```
// wake up all shreds waiting on the event e
e.broadcast();
```

broadcast() melepaskan semua potongan yang ada pada antrian oleh event itu, di dalam perintah yang telah ditambahkan, dan di saat yang sama pada waktunya

Potongan yang dilepaskan adalah dijadwalkan untuk dijalankan dengan seketika. Tetapi hanya event yang akan merespon potongan lain juga trejadwal untuk berjalan pada waktu yang sama. Lagipula, potongan yang memanggil sinyal() atau broadcast() akan terus berjalan sampai suatu event mempercepat waktunya

sendiri, atau menghasilkan mesin virtual tanpa mempercepat waktu. (lihat me.yield() di bawah konkruensi).

MIDI events

Chuck berisi kelas pembangun MIDI untuk memungkinkan interaksi dengan MIDI berdasarkan perangkat lunak atau alat.

```
MidiIn min;
MidiMsg msg;
MidiOut mout;
// open midi receiver, exit on fail
if ( min.open(0) ) me.exit();
while( true )
{
    // wait on midi event
    min => now;
    // receive midimsg(s)
    while( min.recv( msg ) )
    {
        // print content
        <<< msg.data1, msg.data2, msg.data3 >>>;
    }
}
...
```

MidiIn adalah suatu sub kelas event, sama seperti menchuck pada saat itu. MidiIn kemudian mengambil suatu objek **MidiMsg** dimana metoda .recv() untuk mengakses MIDI data. Sebagai *default*, MidiIn event mencetuskan event broadcast() yang sama.

OSC events

Sebagai tambahan terhadap MIDI, Chuck mempunyai kelas event OSC juga:

…

```
// create our OSC receiver
OscRecv orec;
// port 6449
6449 => orec.port;
// start listening (launch thread)
orec.listen();
function void rate_control_shred()
{
    // create an address in the receiver
    // and store it in a new variable.
    orec.event("/sndbuf/buf/rate,f") @=> OscEvent oscdata;
    while ( true )
    {
        oscdata => now; //wait for events to arrive.
        // grab the next message from the queue.
        while( oscdata.nextMesg() != 0 )
        {
            // getFloat fetches the expected float
            // as indicated in the type string ",f"
            buf.play( oscdata.getFloat() );
            0 => buf.pos;
        }
    }
}
…
```

Kelas **OscRecv** mendengarkan untuk paket OSC yang berikutnya pada port yang spesifik. Masing-Masing event OscRecv dapat menciptakan objek OscEvent menggunakan metoda eventnya() untuk mendengarkan pada paket pola penunjukan OSC manapun yang valid.

Suatu event OscEvent bisa dichuckkan untuk menantikan pesan yang tiba, di mana metoda **nextMesg()** dan **get{Float|String|Int}()** dapat digunakan untuk mengambil pesan data.

Membuat Bentuk Event pada umumnya

Event, seperti umumnya kelas yang lain, dapat dijadikan sub kelas untuk menambahkan kemampuan dan mengirimkan data :

```
// extended event
class TheEvent extends Event
{
    int value;
}
// the event
TheEvent e;
// handler
fun int hi( TheEvent event )
{
    while( true )
    {
        // wait on event
        event => now;
        // get the data
        <<<e.value>>>;
    }
}
```

```
// spork
spork ~ hi( e );
spork ~ hi( e );
spork ~ hi( e );
spork ~ hi( e );

// infinite time loop
while( true )
{
    // advance time
    1::second => now;
    // set data
    Math.rand2( 0, 5 ) => e.value;
    // signal one waiting shred
    e.signal();
}
```

BAB 13

Objek

Pengenalan

Chuck mengimplementasikan suatu sistem objek yang meminjam dari konvensi C++ dan Java. Pada kasus kita ini berarti :

- Kamu dapat menggambarkan kelas pada umumnya sebagai tipe yang baru dan *instantiate* objek
- Chuck mendukung *inheritance polymorphic* (ini adalah model yang sama digunakan dalam Java, dan juga mengenal sebagai *inheritance* virtual di dalam C++)
- Semua variabel objek adalah acuan (seperti Java), tetapi *instantiation* menyerupai C++. Kita akan mendiskusikan ini secara detil di bawah.
- Ada suatu perpustakaan kelas *default*.
- Semua objek menerima warisan dari Kelas Objek (seperti di Java)

Demi kejelasan kita akan menggambarkan terminologi ini :

- suatu kelas adalah suatu abstraksi data (anggota) dan perilaku (metoda)
- suatu kelas adalah suatu tipe.
- suatu obyek adalah suatu *instantiation* (menyangkut) kelas tersebut
- suatu variabel acuan menunjuk secara tidak langsung kepada suatu obyek yang bukan obyeknya sendiri. Semua Chuck variabel objek adalah variabel acuan (seperti Java).
- dengan cara yang sama, tugas acuan menyalin suatu acuan bagi suatu obyek dan menugaskan acuan tersebut kepada suatu variabel acuan.

Obyeknya sendiri tidaklah disalin. Semua tugas obyek Chuck adalah tugas acuan.

Built-in Kelas

Chuck mempunyai sejumlah kelas yang digambarkan di dalam bahasa tersebut.

- Obyek : Kelas Dasar bagi semua objek Chuck.
- Event : mekanisme sinkronisasi dasar Chuck's; kemungkinan diperluas untuk menciptakan Kemampuan event pada umumnya (yang dibahas di sini).
- Potongan: abstraksi dasar untuk suatu proses *non-premptive* Chuck.
- UGen: kelas generator unit dasar (yang dibahas di sini).
  Kelas yang biasanya digunakan di dalam Chuck.

Bekerja dengan Objek

dimulai dengan beberapa contoh. Karena contoh ini, diasumsikan *Foo* sebagai definisi kelas.

// membuat sebuah objek foo; stored in reference variable bar
Foo bar;

Kode dapat mengerjakan dua hal :

- suatu indeks variabel acuan diumumkan; tipenya adalah *Foo*.
- suatu kejadian *Foo* yang baru diciptakan, dan referensinya ditugaskan untuk menghalangi.

Catatan bahwa ini berlawanan dengan Java, statemen ini kedua-duanya mengumumkan suatu variabel acuan dan *instantiates* suatu kejadian yang menyangkut kelas itu dan menugaskan acuannya kepada variabel tersebut. Juga catatan bahwa ini berlawanan dengan C++, indeks adalah suatu acuan, dan tidak menghadirkan obyeknya sendiri.

Untuk mengumumkan suatu variabel acuan yang tidak mengacu pada apapun (juga disebut suatu acuan batal):

// create a null reference to a Foo object
Foo @ bar;

Kode hanya mengumumkan suatu acuan dan inisialisasi null tersebut. (catatan acak: di atas statemen adalah mungkin dibaca seperti ”*Foo* pada indeks”)

Kita dapat menugaskan suatu kejadian baru kepada variabel acuan :

// assign new instance of Foo to bar
new Foo @=> Foo @ bar;

// (this statement is equivalent to 'Foo bar', above)

Kode di atas persisnya setara dengan *Foo* indeks; seperti ditunjukkan di atas. Operator yang baru membuat suatu kejadian dari sebuah kelas, dalam hal ini *Foo*. Operator @=¿ melaksanakan tugas acuan tersebut. (lihat Bab Operator untuk informasi lebih lengkap mengenai @=¿).

Itu adalah mungkin untuk membuat banyak acuan ke obyek yang sama :

// make a Foo
Foo bar;
// reference assign to duh
bar @=> Foo @ duh;
// (now both bar and duh points to the same object)

Objek acuan Chuck terhitung dan koleksi tidak berguna berlangsung secara otomatis. (catatan: namun diterapkan !)

Seperti dinyatakan di atas, suatu kelas boleh berisi data dan perilaku, dalam wujud variabel anggota dan fungsi anggota, secara berturut-turut. Anggota diakses dengan penggunaan ’notasi titik’ *reference.memberdata* dan *reference.memberfunc()*. Untuk memohon suatu fungsi anggota dari suatu obyek (diumpamakan kelas **Foo** mempunyai suatu fungsi anggota yang disebut menghitung dengan mengambil dua bilangan bulat dan mengembalikan suatu bilangan bulat) :

```
// make a Foo
Foo bar;
// call compute(), store result in boo
bar.compute( 1, 2 ) => int boo;
```

Menulis Sebuah Kelas

Jika suatu kelas telah digambarkan didalam mesin virtual Chuck (file yang sama manapun atau sebagai suatu kelas publik di dalam suatu file berbeda) kemudian dapat serupa *instantiated* ke tipe primitif.

Kecuali jika publik diumumkan, definisi kelas adalah memandang kepada potongan dan tidak akan bertentangan dengan nama kelas yang identik lainnya di dalam menjalankan potongan.

Kelas *Encapsulate* satu set perilaku dan data. Untuk menggambarkan suatu tipe obyek baru, kata kunci kelas digunakan dengan diikuti oleh nama yang kelas.

```
// define class X
class X
{
    // insert code here
}
```

Jika suatu kelas digambarkan sebagai publik, maka kelas tersebut terintegrasi ke dalam *namespace* pusat (sebagai ganti yang lokal), dan dapat di *instantiated* dari

program lainnya yang sesudah itu di-compile. Paling banyak satu kelas publik per file.

```
// define public class MissPopular
public class MissPopular
{
        // ...
}

// define non-public class Flarg
class Flarg
{
        // ...
}

// both MissPopular and Flarg can be used in this file
// only MissPopular can be used from another file
```

Kita menggambarkan data anggota dan metoda untuk menetapkan kemampuan dan jenis data yang memerlukan kelas itu. Anggota atau data kejadian dan fungsi kejadian dihubungkan dengan kejadian individu suatu kelas, sedangkan fungsi dan data statis hanya dihubungkan dengan kelas (dan yang bersama oleh kejadian).

Anggota (Data Event + Fungsi)

Data Event dan Metoda dihubungkan dengan suatu obyek.

```
// define class X
class X
{
        // declare instance variable 'm_foo'
        int m_foo;
        // another instance variable 'm_bar'
```

```
        float m_bar;
        // yet another, this time an object
        Event m_event;

        // function that returns value of m_foo
        fun int getFoo() { return m_foo; }

        // function to set the value of m_foo
        fun void setFoo( int value ) { value => m_foo; }

        // calculate something
        fun float calculate( float x, float y )
        {
                // insert code
        }

        // print some stuff
        fun void print()
        {
                <<< m_foo, m_bar, m_event >>>;
        }
}

// instantiate an X
X x;
// set the Foo
x.setFoo( 5 );
// print the Foo
<<< x.getFoo() >>>;

// call print
```

```
x.print();
```

Kelas Konstruktor

Di (dalam) pelepasan atau *release* awal, kita masih tidak mendukung pembangun. Bagaimanapun, kita mempunyai *pre-constructor* tunggal. Kode yang dengan seketika di dalam suatu kelas *definiton* (dan bukan di dalam fungsi manapun) dijalankan setiap kali suatu kejadian menyangkut kelas itu diciptakan.

```
// define class X
class X
{
    // we can put any ChucK statements here as pre-constructor
    // initialize an instance data
    109 => int m_foo;
    // loop over stuff
    for( 0 => int i; i < 5; i++ )
    {
        // print out message how silly
        <<< "part of class pre-constructor...", this, i >>>;
    }
    // function
    fun void doit()
    {
        // ...
    }
}
// when we instantiate X, the pre-constructor is run
X x;

// print out m_foo
<<< x.m_foo >>>;
```

Static (Data + Fungsi)

Fungsi dan Data Statis dihubungkan dengan suatu kelas, dan bersama oleh semua kejadian yang sesungguhnya menyangkut kelas itu, unsur-unsur statis dapat diakses tanpa suatu kejadian, dengan penggunaan nama kelas tersebut : **Classname.Element**.

```
// define class X
class X
{
        // static data
        static int our_data;
        // static function
        fun static int doThatThing()
        {
                // return the data
                return our_data;
        }
}
// do not need an instance to access our_data
2 => X.our_data;
// print out
<<< X.our_data >>>;
// print
<<< X.doThatThing() >>>;
// create instances of X
X x1;
X x2;
// print out their static data - should be same
<<< x1.our_data, x2.our_data >>>;
// change use one
5 => x1.our_data;
```

// the other should be changed as well

<<< x1.our_data, x2.our_data >>>;

Pewarisan

Warisan di dalam kode berorientasi obyek menolak mengijinkan programmer untuk mengambil suatu kelas yang ada dan menambah atau mengubah kemampuannya. Dalam pelaksanaannya kita dapat menciptakan suatu taksonomi kelas bahwa semua bagian adalah suatu spesifik satuan perilaku, sedang perilaku diterapkan secara berbeda, namun dirumuskan dengan baik, sejalan. Kita mengindikasikan bahwa suatu kelas baru menerima warisan dari kelas lain menggunakan penambahan kata kunci. Kelas yang dari mana kita menerima warisan dikenal sebagai kelas orangtua, dan kelas yang menerima warisan kelas adalah kelas anak. Kelas Anak menerima semua anggota data dan fungsi dari kelas orangtua, walaupun fungsi dari kelas orangtua adalah mungkin dikesampingkan (di bawah). Sebab kelas anak berisi kemampuan dari kelas orangtua, acuan ke kejadian sebuah kelas anak adalah mungkin ditugaskan untuk suatu jenis acuan kelas orangtua.

Untuk sekarang, mengakses modifier (public, protected, privatei) adalah termasuk tetapi tidak secara penuh diterapkan.

Segalanya adalah publik oleh *default*.

```
// define class X
class X
{
        // define member function
        Fun void doThatThing()
        {
                <<<"Hallo">>>;
        }
        // define another
        fun void hey()
```

```
        {
        <<<"Hey!!!">>>;
        }
        // data
        int the_data;
}

// define child class Y
class Y extends X
{
        // override doThatThing()
        fun void doThatThing()
        {
        <<<"No! Get away from me!">>>;
        }
}
// instantiate a Y
Y y;
// call doThatThing
y.doThatThing();
// call hey() - should use X's hey(), since we didn't override
y.hey();
// data is also inherited from X
<<< y.the_data >>>;
```

Warisan menyediakan kita suatu cara secara efisien berbagi kode antar kelas yang melaksanakan peran serupa. Kita dapat mendefinisikan pola perilaku kompleks tertentu, serta mengubah cara perilaku aspek tertentu beroperasi.

```
// parent class defines some basic data and methods
class Xfunc
```

```
{
        int x;
        (
                fun int doSomething( int a, int b ) {
                return 0;
        }
}

// child class, which overrides the doSomething function with an addition
operation
class Xadds extends Xfunc
{
        fun int doSomething ( int a, int b )
        {
                return a + b ;
        }
}

// child class, which overrides the doSomething function with a multiply
operation
class Xmuls extends Xfunc
{
        fun int doSomething ( int a, int b )
        {
                return a * b;
        }
}

// array of references to Xfunc
Xfunc @ operators[2];
// instantiate two children and assign reference to the array
```

```
new Xadds @=> operators[0];
new Xmuls @=> operators[1];
// loop over the Xfunc
for( 0 => int i; i < operators.cap(); i++ )
{
        // doSomething, potentially different for each Xfunc
        <<< operators[i].doSomething( 4, 5 ) >>>;
}
```

sebab **Xmuls** dan **Xadds** masing-masing mendefinisikan kembali sesuatu (*int* a, *int* b) yang dilakukan dengan kode mereka sendiri, kita katakan bahwa mereka sudah mengesampingkan perilaku kelas orangtuanya.

Overloading

Fungsi yang *overloading* di dalam kelas adalah serupa dengan fungsi reguler. lihat fungsi.

BAB 14

Kompiler Chuck + Mesin Virtual

Mari mulai dengan *compiler* atau mesin virtual, kedua-duanya dapat menjalankan proses yang sama. Sekarang juga, kamu perlu mempunyai *built* atau *installed* Chuck (pemandu), dan barangkali mengambil pengajaran tambahan tentang Chuck. Pemandu ini dimaksudkan untuk menjadi pelengkap dan petunjuk lebih dibandingkan pengajaran tambahan tersebut.

RINGKASAN (suatu halaman man-esque)

pemakaian:

chuck - - [options|commands] [+-=^] file1 file2 file3 ...

[options] = halt|loop|audio|silent|dump|nodump|about|

srate<N>|bufsize<N>|bufnum<N>|dac<N>|adc<N>|

remote<hostname>|port<N>|verbose<N>|probe

remote<hostname>|port<N>

[commands] = add|remove|replace|status|time|kill

[+-=^] = shortcuts for add, remove, replace, status

URAIAN

[sumber file Chuck]: Chuck dapat menjalankan 1 atau lebih proses yang paralel dan secara interaktif. Programmer hanya harus menetapkan mereka semua pada baris perintah, dan mereka akan di-compile dan berjalan di VM. Masing-Masing file sumber masukan (akhiran .ck oleh konvensi) akan dijalankan terpisah sebagai ’potongan’ (*user-level* Chuck benang atau ulir) di dalam VM.

Mereka mendapat potongan tambahan *'spork'* dan saling berhubungan dengan potongan yang ada. Terima kasih kepada mekanisme pemilihan waktu Chuck, suatu potongan tidak perlu harus memahami tentang satu sama lain untuk dengan tepat menjadwalkan pada waktunya mereka hanya harus menjejaki waktu milik mereka, boleh dikatakan seperti itu.

Untuk tambahan, banyak potongan dapat ditambahkan atau dipindahkan atau digantikan secara manual pada *run-time*, menggunakan *on-the-fly programming* [Wang dan Cook 2004] - (lihat penerbitan dan http://on-the-fly.cs.princeton.edu/).

[pilihan] :
- -halt / -h
(secara default) - diberitahukan kepada VM untuk berhenti dan keluar dari VM jika tidak ada potongan-potongan di dalam VM.
- -loop / -l
beritahukan kepada Chuck VM untuk melanjutkan pelaksanaan sekalipun di sana tidak ada potongan saat ini di dalam VM. Ini adalah bermanfaat sebab potongan dapat ditambahkan kemudian melalui *on-the-fly*. Lagipula, itu adalah sah untuk menetapkan pilihan ini tanpa file masukan. Sebagai contoh :
%>chuck - -loop
atas kehendak VM *'time-loop* tanpa batas', menantikan datangnya potongan.
- -audio / -a
(secara default) - memungkinkan keluaran *real-time* audio
- -silent / -s
menonaktifkan keluaran *real-time* audio - perhitungan di dalam VM tidaklah diubah, kalau tidak pemilihan waktu yang nyata adalah yang tidak lagi *clocked* oleh mesin audio *real-time*. Manipulasi Pemilihan waktu (seperti operasi terpasang 'sekarang') masih berfungsi secara penuh. Ini bermanfaat untuk menyatukan audio ke disk atau jaringan. Juga, adalah ringkas untuk menjalankan suatu program non-audio.

- -dump / +d

mengabaikan instruksi virtual yang dipancarkan ke **stderr**, untuk semua file setelah *flag* ini berada pada baris perintah, sampai sebuah *'nodump'* ditemui (lihat di bawah). Sebagai contoh:

%>chuck foo.ck +d bar.ck

akan mengabaikan Chuck virtual instruksi untuk **bar.ck** (saja), dengan menilai argumentasi, ke *stderr*. - *-dump* dapat digunakan bersama dengan- *-nodump* untuk memilih membuang file.

- -nodump / -d

Secara *default* membuat pembuangan berhenti dari instruksi sebetulnya untuk file yang datang setelah *flag* di dalam baris perintah, sampai sebuah '*dump*' ditemui (lihat di atas). Sebagai contoh:

%>chuck +d foo.ck -d bar.ck +d doo.ck

akan membuang **foo.ck**, kemudian **doo.ck** - tetapi bukan **bar.ck**.

Ini adalah bermanfaat untuk debug Chuck sendiri, dan bermaksud untuk pertunjukan lain.

- -srate(N)

set contoh internal *rate* untuk (N) Hz. dengan default, Chuck berjalan pada 44100Hz untuk OS X dan Windows, dan 48000Hz untuk linux/ALSA. sekalipun VM sedang menjalankan- -17, contoh rate masih digunakan oleh beberapa unit *generaters* untuk menghitung audio, ini adalah penting untuk menghitung contoh dan penulisan untuk memfile. Tidak semua contoh rate didukung oleh semua alat!

- -bufsize(N)

set ukuran penyangga audio internal untuk (N) contoh frame. ukuran penyangga lebih besar sehingga sering mengurangi audio *artifacts* dalam kaitan dengan system / program pemilihan waktu. penyangga lebih kecil mengurangi audio latency. Default adalah 512. Jika (N) bukanlah suatu power dari 2, power yang berikutnya 2 kali lebih besar dari (N) yang digunakan. Sebagai contoh:

%>chuck - -bufsize950

menetapkan ukuran penyangga pada 1024.

- -dac(N)

membuka alat keluaran audio #(N) untuk *real-time audio*. oleh default, (N) adalah 0.

- -adc(N)

membuka alat masukan audio #(N) untuk *input real-time audio*. Oleh default, (N) adalah 0.

- -chan(N) / -c(N)

membuka N untuk sejumlah channel masukan dan keluaran pada alat audio. Oleh default, (N) adalah 2.

- -in(N) / -i(N)

membuka N untuk sejumlah channel masukan pada alat audio. Oleh default (N) adalah 2.

- -out(N) -o(N)

membuka N untuk sejumlah channel keluaran pada alat audio. Oleh default (N) adalah 2.

- -about / - -help

mencetak pesan pemakaian, dengan *URL Chuck*.

- -callback

Gunakan suatu callback untuk penyangga (default).

- -blocking

Gunakan hambatan untuk penyangga.

BAB 15

Perintah On-the-fly Programming

Ini digunakan untuk memprogram *on-the-fly* (lihat http://on-the-fly.cs.princeton.edu). Oleh default, memprogram *on-the-fly* memerlukan bahwa suatu mesin virtual Chuck telah siap dijalankan pada localhost tersebut. komunikasikan via soket ke *add* atau *remove* atau *replace* potongan di dalam VM, dan untuk menyatakan query VM. Cara yang paling sederhana untuk menyediakan suatu mesin virtual Chuck untuk menerima perintah ini adalah dengan permulaan suatu VM kosong dengan - - *loop* :

%>chuck - -loop

saat akan memulai start suatu VM, pengulangan (dan mempercepat waktu), menantikan perintah berikutnya. Pemanggilan berurutan ‘Chuck’ dengan perintah yang sesuai akan berkomunikasi dengan pendengar VM ini.

(untuk mengendalikan operasi di atas TCP, lihat di bawah)

- -*add* / +

menambahkan potongan baru dari sumber file kepada pendengar VM. Proses ini kemudian keluar. Sebagai contoh:

%>chuck + foo.ck bar.ck

mengintegrasikan foo.ck dan bar.ck ke dalam pendengar VM. Potongan yang bertanggung jawab untuk menemukan tentang pemilihan waktu dan potongan lain via mekanisme pemilihan waktu dan alat penghubung VM.

- *-remove / -*

memindahkan potongan yang ada dari VM oleh ID. Bagaimana cara menemukan tentang id? (lihat- -status di bawah) sebagai contoh:

%>chuck - 2 3 8

memindahkan 2, 3, 8 potongan.

- *-replace / =*

menggantikan potongan yang ada dengan suatu potongan baru. sebagai contoh:

%>chuck = 2 foo.ck

menggantikan 2 potongan dengan foo.ck

- *-status / ^*

query status keluaran-VM pada pendengar VM. sebagai contoh :

%>chuck ^

ini mencetak start potongan yang internal di pendengar VM, kira-kira :

[chuck](VM): status (now == 0h:2m:34s) ...

[shred id]: 1 [source]: foo.ck [sporked]: 21.43s ago

[shred id]: 2 [source]: bar.ck [sporked]: 28.37s ago

- *-time*

mencetak ke luar nilai saat ini pada pendengar VM. sebagai contoh:

%>chuck - -time

kira-kira :

[chuck](VM): the value of now: now = 403457 (samp)

= 9.148685 (second)

= 0.152478 (minute)

= 0.002541 (hour)

= 0.000106 (day)

= 0.000015 (week)

- *-kill*

semi-gracefully menghancurkan pendengar VM - memindahkan semua potongan dulu.

- -remote /

menetapkan di mana untuk mengirimkan perintah on-the-fly. Harus nampak baris perintah sebelum perintah on-the-fly manapun. Sebagai contoh:

%>chuck @192.168.1.1 + foo.ck bar.ck

(or)

%>chuck @foo.bar.com -p8888 + foo.ck bar.ck

Mengirimkan foo.ck dan bar.ck ke VM pada 192.168.1.1 atau *foo.bar.com: 8888.*

BAB 16

Standar Library API

Chuck Standar Library API, library ini adalah menyediakan default dengan Chuck hal-hal baru dapat juga diimport penghubung dinamis Chuck (segera untuk didokumentasikan...). Library yang ada di organisir oleh namespaces di dalam Chuck. Mereka adalah sebagai berikut.

namespace: Std

Std adalah suatu library standar di (dalam) Chuck, yang meliputi fungsi, generasi nomor atau jumlah acak, konversi unit dan nilai mutlak.

[contoh]

```
/* a simple example... */
// infinite time-loop
while( true )
{
    // generate random float (and print)
    <<< Std.rand2f( 100.0, 1000.0 ) >>>;
    // wait a bit
    50::ms => now;
}
```

[function] int abs (int value);

Mengembalikan nilai mutlak dari integer

[function] float fabs (float value);

Mengembalikan nilai mutlak dari bilangan real

[function] int rand ( );

Menghasilkan bilangan bulat acak

[function] int rand2 (int min, int max);

Menghasilkan bilangan bulat acak di dalam cakupan [min, max]

[function] float randf ( );

Menghasilkan bilangan real acak di dalam cakupan [-1, 1]

[function] float rand2f (float min, float max);

Menghasilkan bilangan real acak di dalam cakupan [min, max]

[function] float srand (int value)

benih nilai untuk menghasilkan bilangan yang acak

[function] float sgn (float value);

Menghitung tanda masukan sebagai - 1.0 (negatif), 0 (nol), atau 1.0 (positif)

[function] int system (string cmd);

Memberikan suatu perintah yang dieksekusi pada shell

[function] int a to i (string value);

Mengubah nilai ascii (string) menjadi nilai integer (int)

[function] float a to f (string value);

Mengubah nilai ascii (string) menjadi nilai real (float)

[function] string getenv (string value);

Mengembalikan nilai dari suatu lingkungan variabel, seperti ” PATH ”

[function] int setenv (string key, string value);

Menetapkan lingkungan variabel nama ’key’ untuk ’nilai’

[function] float mtof (float value);

Mengkonversi suatu catatan nomor MIDI ke frekwensi (Hz)

catatan nilai masukan itu dari tipe ’real’ (mendukung catatan nomor fraksional)

[function] float ftom (float value);

mengkonversi frekwensi (Hz) ke ruang catatan nomor MIDI

[function] float powtodb (float value);

mengkonversi rasio kekuatan sinyal ke desibel (dB)

[function] float rmstodb (float value);

Mengkonversi amplitudo linear ke desibel (dB)

[function] float dbtopow (float value);

Mengkonversi desibel (dB) ke rasio kekuatan sinyal

[function] float dbtorms (float value);

Mengkonversi desibel (dB) ke amplitudo linear

namespace: Machine

Machine adalah Chuck runtime penghubung ke virtual mesin. alat penghubung ini dapat digunakan untuk mengatur potongan. Mereka adalah serupa kepada Perintah Memprogram On-The-Fly, kecuali ini dilibatkan dari dalam suatu program Chuck, dan adalah subyek untuk pemilihan waktu *mech anism*.

[function] int add (string path);

compile dan *spork* adalah suatu potongan baru dari file *'path'* ke dalam VM sekarang mengembalikan ID potongan

(lihat *example/machine.ck*)

[function] int spork (string path);

Sama dengan menambah /+

[function] int remove (int id);

memindahkan potongan dari VM oleh ID potongan (yang dikembalikan oleh *add/spork*)

[function] int replace (int id, string path);

menggantikan potongan dengan potongan baru dari file

mengembalikan ID potongan, atau 0 pada kesalahan

[function] int status ( );

menampilkan arus status dari VM

memiliki fungsi yang sama ketika status/^

(lihat *example/status.ck*)

[function] void crash ( );

iterally menyebabkan VM untuk dihancurkan. tempat peristirahatan terakhir; menggunakan dengan penuh perhatian. Terima kasih.

namespace: Math

Math berisi fungsi *math* standar. semua fungsi trignometri sudut untuk di lingkaran.

[contoh]

// print sine of pi/2

<<<< Math.sin(pi / 2.0) >>>;

[function] float sin (float x);

Menghitung sinus x

[function] float cos (float x);

Menghitung kosinus x

[function] float tan (float x);

Menghitung tangen x

[function] float asin (float x);

Menghitung arc sinus x

[function] float acos (float x);

Menghitung arc kosinus x

[function] float atan (float x);

Menghitung arc tangen x

[function] float atan2 (float x);

menghitung nilai dengan prinsip arc tangen y/x, menggunakan tanda dari kedua argumentasi untuk menentukan kwadrant dari nilai kembali

[function] float sinh (float x);

Menghitung sinus hiperbola x

[function] float cosh (float x);

Menghitung kosinus hiperbola x

[function] float tanh (float x);

Menghitung tangen hiperbola x

[function] float hypot (float x, float y);

menghitung jarak euclidean garis vektor orthogonal (x,0) dan (0,y)

[function] float pow (float x, float y);

menghitung x mengambil pada kekuatan y-th

[function] float sqrt (float x);

menghitung akar dua nonnegative x (x harus¿= 0)

[function] float exp (float x);

menghitung eˆx, berbasis-e exponensial x

[function] float log (float x);

Menghitung logaritma natural x

[function] float log2 (float x);

Menghitung logaritma x untuk basis 2

[function] float log10 (float x);

Menghitung logaritma x untuk basis 10

[function] float floor (float x);

mengembalikan nilai integral paling besar (dikembalikan sebagai real) bukan lebih besar dari x

[function] float ceil (float x);

Mengembalikan nilai integral paling kecil (dikembalikan sebagai real) bukan lebih kecil dari x

[function] float round (float x);

mengembalikan ke asal nilai integral paling dekat (dikembalikan sebagai real)

[function] float trunc (float x);

mengembalikan ke asal nilai integral paling besar (dikembalikan sebagai real) tidak lebih besar dalam magnitude dibanding x

[function] float fmod (float x, float y);

menghitung sisa hasil bagi bilangan real x / y

[function] float remainder (float x, float y);

menghitung nilai r dengan r = x- n* y, di mana n adalah bilangan bulat yang paling dekat tepat dengan nilai x/y. Jika ada dua bilangan bulat terdekat ke x / y, n akan menjadi satu. Jika r adalah nol, diberi tanda yang sama sebagai x

[function] float min (float x, float y);

memilih paling sedikit untuk dua buah nilai

[function] float max (float x, float y);

Memilih paling banyak untuk dua buah nilai

[function] int nextpow2 (int n);

menghitung integral (dikembalikan sebagai int) yang lebih kecil dari 2 lebih besar dari nilai x

[function] int isinf (float x);

apakah x tidak terbatas?

[function] int isnan (float x)

Apakah x bukan angka?

[function] float pi ;

(meniadakan - penggunaan pi (22/7))

[function] float twopi ;

(meniadakan)

[function] float e ;

(meniadakan- penggunaan Math.Exp(1))

BAB 17

Unit Generator

Generator Unit (*ugens*) dapat dihubungkan dengan menggunakan Chuck operator(=>)

[contoh]

adc => dac;

di atas menghubungkan *ugen 'adc'* (*a/d convertor*, atau inputan audio) untuk ' *dac'* (*d/a convertor*, atau keluaran audio).

Ugens dapat juga tidak dihubungkan (menggunakan=<) dan berhubungan kembali (lihat *examples/unchuck.ck*).

Suatu unit generator mungkin punya 0 atau lebih parameter kendali. Suatu *Ugen'S* parameter dapat juga di-set menggunakan Chuck operator(=>, atau->)

[contoh]

```
//connect sine oscillator to dac
SinOsc osc => dac;
// set the Osc's frequency to 60.0 hz
60.0 => osc.freq;
```

(lihat *examples/osc.ck*)

Semua *ugen's* mempunyai sedikitnya empat parameter:

[ctrl param]

- *.gain* - (real, BACA/TULIS)- *gain* yang di-set
- *.op* - (int, BACA/TULIS)- tipe operasi yang di-set
  - -1 *passthrough*
  - 0 tidak diproses
  - 1 diproses secara normal

- *.last* - (real, BACA/TULIS)- contoh yang terakhir dihitung oleh generator unit
- *.channel* - (int, BACA saja) - jumlah *channel* pada *UGEN*
- *.chan*- (int)- mengembalikan suatu acuan pada suatu channel (0 -> N-1)

Multichannel UGENS adalah *adc, dac, Pan2, Mix2*

[contoh]

Pan2 p;

// assumes you called chuck with at least --chan5 or -c5

p.chan(1) => dac.chan(4);

Audio Output

dac

- digital/analog konverter
- abstrak untuk mendasari alat keluaran audio

[ctrl param]

- *.left* - (*UGEN*)- masukan untuk saluran kiri
- *.right* - (*UGEN*)- masukan untuk saluran kanan
- *.chan* - () - masukan untuk saluran N pada alat (0 -> N-1)
- *.channel* - (Int, BACA saja) - mengembalikan jumlah saluran yang terbuka pada alat

Adc

- analog/digital konverter
- abstrak untuk mendasari alat masukan audio

[ctrl param]

- *.left* - (*UGEN*) - keluaran untuk saluran kiri
- *.right* - (*UGEN*) - keluaran untuk saluran kanan
- *.chan* - () - keluaran untuk saluran N pada alat (0 -> N-1)
- *.channel* - (Int, BACA saja) - mengembalikan jumlah saluran terbuka pada alat

blackhole

- contoh *rate*, contoh pengisap
- (seperti *dac*, detak *ugens*, tetapi tidak ada lagi)
- lihat *examples/pwm.ck*

Gain

- kontrol *gain*
- (CATATAN - semua generator unit bisa dengan sendirinya mengubah *gain* mereka)
- (ini adalah suatu cara untuk menggabungkan N keluaran dan mengelupas mereka)
- penggunaan di dalam *examples/i-robot.ck*

[ctrl param]

- *.gain* - (real, BACA/TULIS) - *gain* yang di-set (semua *ugen's* mempunyai ini)
- *.op* - (int, BACA/TULIS) - tipe operasi yang di-set
  - -1 *passthrough*
  - 0 tidak ada proses
  - 1 menambahkan masukan (default)
  - 2 mengurangi masukan
  - 3 mengalikan masukan
  - 4 membagi masukan

[contoh]

```
Noise n => Gain g => dac;
SinOsc s => g;
.3 => g.gain;
while( true ) { 100::ms => now; }
```

Wave Forms

Noise

- pembangkit noise putih
- lihat *examples/noise.ck, examples/powerup.ckts*

Impulse

- generator *pulse* - dapat menetapkan nilai contoh yang sekarang
- default untuk masing-masing contoh adalah 0 jika tidak di-set

[ctrl param]

- *.value* - (real, BACA/TULIS) - nilai yang sekarang di-set (sekarang ini dirusakkan)
- *.next* - (real, BACA/TULIS) - nilai yang di-set dari contoh berikutnya

[contoh]

Impulse i => dac;
while(true) {
1.0 => i.next;
100::ms => now;
}

Blit

- rombongan membatasi generator *pulse* - dapat menetapkan nilai contoh yang sekarang
- default untuk masing-masing contoh adalah 0 jika bukan *setf* contoh berikutnya

[ctrl param]

- *.value* - (real, BACA/TULIS) - nilai yang sekarang di-set (sekarang ini dirusakkan)
- *.next* - (real, BACA/TULIS)- nilai yang di-set dari contoh berikutnya

[contoh]

Impulse i => dac;
while(true) {
    1.0 => i.next;

100::ms => now;

}

Step

- langkah generator - seperti Impuls, tetapi sekali nilai di-set, dipegang untuk semua contoh berikut, sampai nilai di-set lagi
- lihat *examples/step.ck*

[ctrl param]

- *.value* - (real, BACA/TULIS) - set nilai yang sekarang
- *.next* - (real, BACA/TULIS)- set langkah *value*

[contoh]

Step s => dac;

-1.0 => float amp;

// square wave using step

while(true) {

-amp => amp => s.next;

800::samp => now;

}

Dasar Memproses Sinyal

HalfRect

- penyearah gelombang-paruh
- untuk penyearah gelombang-paruh

FullRect

- penyearah gelombang-penuh

ZeroX

- nol memotong detektor
- memancarkan *pulse* tunggal di nol persimpangan di dalam arah nol persimpangan
- (lihat *examples/zerox.ck*)

filters

BiQuad

- *Biquad* (dua-kutub, dua-nol) menyaring kelas.

Filter dilindungi *subclass implements* dua-kutub, dua-nol filter digital. Suatu metoda disediakan untuk menciptakan suatu resonansi di dalam respon frekuensi sedang pemeliharaan suatu keuntungan saringan tetap.

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

[ctrl param]

- .b2 (real, BACA/TULIS) menyaring koefisien
- .b1 (real, BACA/TULIS) menyaring koefisien
- .b0 (real, BACA/TULIS) menyaring koefisien
- .a2 (real, BACA/TULIS) menyaring koefisien
- .a1 (real, BACA/TULIS) menyaring koefisien
- .a0 (real, BACA saja) menyaring koefisien
- .*pfreq* (real, BACA/TULIS) frekuensi resonansi yang di-set (kutub)
- .*prad* (real, BACA/TULIS) radius kutub(¡= 1 untuk menjadi stabil)
- .*zfreq* (real, BACA/TULIS) bentuk frekuensi
- .*zrad* (real, BACA/TULIS) nol radius
- .*norma* (real, BACA/TULIS) normalisasi
- .*eqzs* (real, BACA/TULIS) keuntungan sama nol

Satu Kutub (*OnePole*)

- STK satu kutub (*one-pole*) menyaring kelas.

Filter Ini dilindungi *subclass* implementasi suatu satu kutub saringan digital. Suatu metoda disediakan untuk menentukan kutub itu memposisikan sepanjang poros yang riil *z-plane* sedang pemeliharaan suatu puncak tetap menyaring keuntungan.

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

[ctrl param]

- .a1 (real, BACA/TULIS) menyaring koefisien
- .b0 (real, BACA/TULIS) menyaring koefisien
- .*pole* (real, BACA/TULIS) kutub yang di-set memposisikan sepanjang poros *z-plane* yang riil

Dua Kutub (*TwoPole*)

- STK dua kutub menyaring kelas.
- lihat *examples/powerup.ck*

filter dilindungi *subclass implements* suatu saringan digital dua kutub. Suatu metoda disediakan untuk menciptakan suatu resonansi di dalam merespon frekuensi sedang pemeliharaan suatu keuntungan saringan yang tetap.
Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.
[ctrl param]

- . a1 (real, BACA/TULIS) menyaring koefisien
- . a2 (real, BACA/TULIS) menyaring koefisien
- . b0 (real, BACA/TULIS) menyaring koefisien
- . *freq* (real, BACA/TULIS) menyaring frekuensi resonansi
- . *radius* (real, BACA/TULIS) menyaring radius resonansi
- . *norm* (real, BACA/TULIS) kotak penyaring normalisasi

SatuNol (*OneZero*)

- STK satu-nol menyaring kelas.

Saringan dilindungi *subclass* implementasi suatu saringan digital satu-nol. Suatu metoda disediakan untuk menentukan posisi nol sepanjang poros yang riil *z-plane* sedang pemeliharaan suatu keuntungan saringan tetap.
oleh Perry R. Cook dan Gary P. Scavone, 1995- 2002.

[ctrl param]

- *.zero* (real, BACA/TULIS) posisi nol yang di-set
- .b0 (real, BACA/TULIS) menyaring koefisien
- .b1 (real, BACA/TULIS) menyaringkoefisien

DuaNol (*TwoZero*)

- STK dua-nol menyaring kelas.

Saringan dilindungi *subclass* implementasi suatu saringan digital dua nol. Suatu metoda disediakan untuk menciptakan sebuah "bentuk" di dalam respon frekuensi sedang pemeliharaan suatu keuntungan saringan tetap.

oleh Perry R. Cook dan Gary P. Scavone, 1995- 2002

[ctrl param]

- .b0 (real, BACA/TULIS) menyaring koefisien
- .b1 (real, BACA/TULIS) menyaring koefisien
- .b2 (real, BACA/TULIS) menyaring koefisien
- *.freq* (real, BACA/TULIS) menyaring bentuk frekuensi
- *.radius* (real, BACA/TULIS) menyaring bentuk radius

Kutub Nol

- STK satu-kutub, satu-nol menyaring kelas.

Saringan dilindungi *subclass* implementasi suatu satu-kutub, satu-nol saringan digital. Suatu metoda disediakan untuk menciptakan suatu allpass menyaring dengan koefisien ditentukan. Metoda yang lain disajikan untuk menciptakan suatu saringan penghalang DC.

oleh Perry R. Cook dan Gary P. Scavone, 1995- 2002

[ctrl param]

- .a1 (real, BACA/TULIS) menyaring koefisien
- .b0 (real, BACA/TULIS) menyaring koefisien
- .b1 (real, BACA/TULIS) menyaring koefisien
- *.blockZero* (real, BACA/TULIS) saringan penghalang DC dengan posisi kutub diberikan

- .*allpass* (real, BACA/TULIS) allpass menyaring dengan diberi koefisien Filter
- STK menyaring kelas.

Kelas ini implementasi suatu yang struktur umum dapat digunakan untuk menciptakan suatu cakupan luas dari filter. Itu dapat berfungsi dengan bebas atau menjadi subclassed untuk menyediakan kendali yang lebih spesifik berdasarkan pada tipe filter tertentu .

Khususnya, kelas ini implementasi standar menyamakan perbedaan :

a[0]*y[n] = b[0]*x[n] + ... + b[nb]*x[n-nb] -
a[1]*y[n-1] - ... - a[na]*y[n-na]

Jika a[0] tidak sama dengan 1, koefisien penyaringan dinormalisir oleh a[0].

parameter \e keuntungan diterapkan di masukan saringan dan tidak mempengaruhi koefisien itu menilai. Nilai Keuntungan default adalah 1.0. Struktur ini mengakibatkan satu ekstra mengalikan per perhitungan contoh, hanyalah mengijinkan kendali mudah keseluruhan keuntungan saringan.

oleh Perry R. Cook dan Gary P. Scavone, 1995- 2002

[ctrl param]

- .*coefs* (string, TULIS juga)

LPF

- Saringan Pass rendah resonansi. 2nd memesan *Butterworth*. (Di masa datang, kelas ini mungkin diperluas sedemikian sehingga memesan dan tipe saringan dapat di-set.)

meluas *FilterBasic*

[ctrl param]

- .*freq* (real, BACA/TULIS) memotong frekuensi (Hz)
- .Q (real, BACA/TULIS) resonansi (default adalah 1)
- .set (real, TULIS saja) *freq* yang di-set dan Q

HPF

- Saringan Pass tinggi resonansi. 2nd memesan *Butterworth*. (Di masa datang, kelas ini mungkin diperluas sedemikian sehingga memesan dan jenis saringan dapat di-set.)

meluas *FilterBasic*

[ctrl param]

- .*freq* (real, BACA/TULIS) memotong frekuensi (Hz)
- .Q (real, BACA/TULIS) resonansi (default adalah 1)
- .set (real, TULIS saja) *freq* yang di-set dan Q

BPF

- Rombongan lewat saringan. 2nd memesan *Butterworth*. (Di masa datang, kelas ini mungkin diperluas sedemikian sehingga memesan dan jenis saringan dapat di-set.)

meluas *FilterBasic*

[ctrl param]

- .*freq* (real, BACA/TULIS) pusat frekwensi (Hz)
- .Q (real, BACA/TULIS) Q (default adalah 1)
- .set (real, TULIS saja) *freq* yang di-set dan Q

BRF

- Rombongan menolak saringan. 2nd memesan *Butterworth*. (Di masa datang, kelas ini mungkin diperluas sedemikian sehingga memesan dan jenis saringan dapat di-set.)

meluas *FilterBasic*

[ctrl param]

- .*freq* (real, BACA/TULIS) frekwensi pusat (Hz)
- .Q (real, BACA/TULIS) Q (default adalah 1)
- .set (real, TULIS saja) *freq* yang di-set dan Q

ResonZ

- Filter resonansi. Sama seperti *Biquad* dengan *gain* sama dengan nol.

meluas *FilterBasic*

[ctrl param]

- .*freq* (real, BACA/TULIS) frekwensi pusat (Hz)
- .Q (real, BACA/TULIS) Q (default adalah 1)
- .set (real, TULIS saja) freq yang di-set dan Q

FilterBasic

- dasar kelas, tidak instan.

[ctrl param]

- .*freq* (real, BACA/TULIS) memotong / pusat frekuensi (Hz)
- .Q (real, BACA/TULIS) resonansi / Q
- .set (real, TULIS saja) *freq* yang di-set dan Q

File Bunyi

SndBuf

- penyangga bunyi (sekarang menyisipkan)
- membaca dari berbagai format file
- lihat *examples/sndbuf.ck*

[ctrl param]

- .*read* - (string, TULIS saja) - memuat file untuk membaca
- .*Chunks* - (int, BACA/TULIS) - ukuran chunk (# tentang frame) untuk yang dibaca atas permintaan; 0 menyiratkan
- keseluruhan file, default; harus menyajikan/mengemukakan membacakan untuk berlaku; terjadi.
- .*write* - (*string*, TULIS saja) - memuat suatu file untuk menulis (atau bukan)
- .*pos* - (int, BACA/TULIS)- posisi yang di-set (0 < p <.sample)
- .*valueAt* - (int, BACA saja) - kembalikan nilai itu pada index contoh

- .*loop* - (int, BACA/TULIS) - kotak pengulangan
- .*interp* - (int, BACA/TULIS) - interpolasi yang di-set (0=*drop*, 1=*linear*, 2=*sinc*)
- .*rate* - (real, BACA/TULIS)- memainkan kembali tingkat tarip (sehubungan dengan kecepatan alami file)
- .*play* - (real, BACA/TULIS) – permainan (sama seperti tingkat tarip)
- .*freq* - (real, BACA/TULIS) - memainkan kembali tingkat tarip (file *loops/second*)
- .*fase* - (real, BACA/TULIS)- tahap yang di-set memposisikan (0 - 1)
- .*channel* - (int, BACA/TULIS) - memilih saluran (0 < x <.channel)
- .*phaseOffset* - (real, BACA/TULIS) - yang di-set suatu fase offset
- .*example* - (int, BACA saja) - mengambil jumlah contoh
- .*length* - (dur, BACA saja) - mengambil panjangnya sebagai jangka waktu
- .*channel* - (int, BACA saja) - mengambil jumlah saluran

Oscillators

Phasor

- generator lereng sederhana (0 ke 1)
- ini dapat diberi umpan ke dalam lain osilator (dengan mode sync 2)
- sebagai tahap kendali. lihat *examples/sixty.ck* untuk suatu contoh

[ctrl param]

- .*freq* (real, BACA/TULIS) frekuensi osilator (Hz)
- .*Sfreq* (real, BACA/TULIS) frekwensi osilator (Hz), fase-matched
- .*fase* (real, BACA/TULIS) fasa-arus
- .*sync* (int, BACA/TULIS) (0) masukan mengatur frekwensi, (1) (yang dipesan), (2) fase mengatur masukan
- .*width* (real, BACA/TULIS) jangka waktu yang di-set lereng pada setiap siklus. (default 1.0)

SinOsc

- osilator sinus
- (lihat *examples/osc.ck*)

[ctrl param]

- .*freq* (real, BACA / TULIS) frekwensi osilator (Hz)
- .*Sfreq* (real, BACA / TULIS) frekwensi osilator (Hz), fase-*matched*
- .*fase* (real, BACA / TULIS) fasa-arus
- .*sync* (int, BACA / TULIS) (0) masukan mengatur frekwensi, (1) (yang dipesan), (2) masukan mengatur tahap

PulseOsc

- pulse osilator
- suatu pulse melambaikan osilator dengan lebar variabel.

[ctrl param]

- .*freq* (real, BACA / TULIS) frekwensi osilator (Hz)
- .*Sfreq* (real, BACA / TULIS) frekwensi osilator (Hz), fase-matched
- .*fase* (real, BACA / TULIS) fasa-arus
- .*sync* (int, BACA / TULIS) (0) masukan mengatur frekwensi, (1) (yang dipesan), (2) masukan mengatur tahap
- .*width* (real, BACA / TULIS) panjangnya tugas beredar (0- 1)

SqrOsc

- osilator gelombang kuadrat

[ctrl param]

- .*freq* (real, BACA / TULIS) frekwensi osilator (Hz)
- .*Sfreq* (real, BACA / TULIS) frekwensi osilator (Hz), Tahap- yang ditarungkan
- .*fase* (real, BACA / TULIS) fasa-arus
- .*sync* (int, BACA / TULIS) (0) masukan mengatur frekwensi, (1) (yang dipesan), (2) masukan mengatur tahap
- .*width* (real, BACA / TULIS) panjangnya tugas beredar (0- 1)

BlitSquare

- rombongan membatasi osilator gelombang kuadrat

[ctrl param]

- .*freq* (real, BACA / TULIS) frekwensi osilator (Hz)
- .*fase* (real, BACA / TULIS) fasa-arus
- .selaras (int, BACA / TULIS) jumlah selaras

TriOsc

- osilator gelombang segitiga

[ctrl param]

- . *freq* (float, BACA / TULIS) frekwensi osilator (Hz)
- . *Sfreq* (float, BACA / TULIS) frekwensi osilator (Hz), Tahap - yang dibandingkan
- . *fase* (float, BACA / TULIS) fasa-arus
- . *sync* (int, BACA / TULIS) (0) mengatur frekuensi masukan, (1) (yang dipesan), (2) mengatur tahap masukan
- .*width* (float, BACA / TULIS) kendali midpoint segitiga (0- 1)

SawOsc

- *sawtooth* osilator gelombang(segitiga, lebar yang dipaksa ke 0.0 atau 1.0)

[ctrl param]

- .*freq* (float, BACA / TULIS) frekuensi osilator (Hz)
- .*Sfreq* (float,MBACA / TULIS) frekuensi osilator (Hz), Tahap- yang dibandingkan
- .*fase* (float, BACA / TULIS) fasa-arus
- .*sync* (int, BACA / TULIS) (0) masukan mengatur frekwensi, (1) (yang dipesan), (2) masukan mengatur tahap
- .*width* (float, BACA / TULIS) terus meningkat (w > 0.5) atau mengurangi (w < 0.5)

BlitSaw

- Rombongan membatasi *sawtooth* osilator gelombang

[ctrl param]

- .*freq* (float, BACA / TULIS) frekuensi osilator (Hz)
- .*phase* (float, BACA / TULIS) fasa-arus
- .harmonis (int, BACA / TULIS) jumlah selaras

Network

netout

- UDP - berbasiskan pemancar audio jaringan

[ctrl param]

- .*addr* (string, BACA / TULIS) alamat target
- .*port* (int, BACA / TULIS) port target
- .*size* (int, BACA / TULIS) ukuran paket
- .*name* (string, BACA / TULIS) nama

netin

- UDP - berbasiskan penerima audio jaringan

[ctrl param]

- .*port* (int, BACA / TULIS) port yang di-set untuk menerima
- .*name* (string, BACA / TULIS) nama

Mono <- -> Stereo

Pan2

- sinyal mono yang disebar ke stereo
- lihat *examples/stereo/moe2.ck*

[ctrl param]

- .*left* (UGEN) saluran kiri (mono) ke luar
- .*right* (UGEN) saluran kanan (mono) ke luar
- .*pan* (float, BACA / TULIS) penempatan nilai pan (- 1 ke 1)

Mix2

- campuran stereo masuk hingga menuju ke saluran mono

[ctrl param]

- *.left* - (UGEN) saluran kiri (mono) didalam
- *.right* - (UGEN) saluran kanan (mono) didalam
- *.pan* - (float, BACA / TULIS) parameter campuran bernilai (0- 1)

STK - Instruments

StkInstrument (di impor dari *Instrmnt*)

Super - kelas untuk STK instrumen

Berikut subkelas *UGENS Stkinstrument* :

- *BandedWG*
- *BlowBotl*
- *BlowHole*
- *Bowed*
- *Brass*
- *Clarinet*
- *Flute*
- *FM* (dan semua subkelasnya : *Beethree, FMVOICES, Hevymetl, Percflut, Rhodey, Tubebell, Wurley*)
- *Mandolin*
- *ModalBar*
- *Moog*
- *Saxofony*
- *Shakers*
- *Sitar*
- *StifKarp*
- *VoicForm*

[ctrl param]

- *.noteOn* - (percepatan float) - catatan picu terpasang
- *.noteOff* - (percepatan float) - catatan picu batal
- *.freq* - (frekuensi float)- diset / mendapatkan frekwensi (Hz)
- *.controlChange* - (Int Nomor, nilai float) - menyatakan perubahan kendali- angka-angka adalah instrumen spesifik, cakupan nilai : [0.0- 128.0]

BandedWG

- Panduan-Gelombang yang memperagakan kelas

Penggunaan Kelas ini merangkum teknik pandu-gelombang untuk model berbagai bunyi, termasuk membungkuk bar, kacamata, dan mangkuk. Untuk informasi lebih lanjut, lihat Essl, G. dan Cook, P. "Pandu-Gelombang Yang terangkum: Ke arah Modeling *Phisik* Instrumen Perkusi", diproses pada Konferensi Musik Komputer Internasional 1999.

Angka-Angka Perubahan Kendali:

- *Bow Pressure* = 2
- *Bow Motion* = 4
- *Strike Position* = 8 (tidak diterapkan)
- *Vibrato Frekuensi* = 11
- *Gain* = 1
- *Bow Velocity* = 128
- *Set Striking* = 64
- *Instrument Presets* = 16
- *Uniform Bar* = 0
- *Tuned Bar* = 1
- *Glass Harmonica* = 2
- *Tibetan Bowl* = 3

dengan Georg Essl, 1999- 2002.

yang dimodifikasi Untuk Stk 4.0 oleh Gary Scavone.

meluas *Stkinstrument*

[ctrl param]

- .*bowPressure* (float, BACA / TULIS) tekanan haluan/busur [0.0- 1.0]
- .*bowMotion* (float, BACA / TULIS) gerakan haluan/busur [0.0- 1.0]
- .*bowRate* (float, BACA / TULIS) Posisi serangan
- .*Strikeposition* (float, BACA / TULIS) Posisi serangan
- .*Integrationconstant*-(float, BACA / TULIS)-?? [0.0- 1.0]
- .*modesGain* (float, BACA / TULIS) amplitudo untuk gaya [0.0- 1.0]
- .*preset* (int, BACA / TULIS) instrumen *preset* (0- 3, lihat di atas)
- .*pluck* (float, BACA / TULIS) instrumen *pluck* [0.0- 1.0]
- .*startBowing* (float, BACA / TULIS) bowing start [0.0- 1.0]
- .*stopBowing* (float, BACA / TULIS) bowing stop [0.0- 1.0]

(pewarisan dari *Stkinstrument*)

- .*noteOn* - (percepatan float) - catatan picu terpasang
- .*noteOff* - (percepatan float) - catatan picu batal
- .*freq* - (frekuensi float)- di-set / mendapatkan frekuensi (Hz)
- .*controlChange* - (Int Nomor, nilai float)- menyatakan perubahan kendali

BlowBotl

- STK kelas instrumen botol tiup

Kelas ini implementasi suatu *helmholtz* resonator (*biquad* filter) dengan suatu polinomial pancaran eksitasi (a la Cook).

Angka-angka pengubah kontrol :

- *Noise Gain* = 4
- *Vibrato Frekuensi* = 11
- *Vibrato Gain* = 1
- *Volume* = 128

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

Meluas *StkInstrument* *[ctrl param]*

- .*noiseGain* -(float, BACA / TULIS)- komponen gangguan gain [0.0- 1.0]
- .*vibratoFreq* -(float, BACA / TULIS)- vibrato frekuensi (Hz)

- .*vibratoGain* - (float, BACA / TULIS)- vibrato memperoleh [0.0- 1.0]
- .*volume* - (float, BACA / TULIS)- namun tombol volume lain [0.0- 1.0]
- .*startBowing* (float, BACA / TULIS) mulai bowing [0.0- 1.0]
- .*stopBowing* (float, BACA / TULIS) perhentian bowing[0.0- 1.0]
- .*rate* (float, BACA / TULIS)- tingkat serangan (detik)

(pewarisan dari *StkInstrument*)

- .*noteOn*- (percepatan float)- catatan picu terpasang
- .*noteOff*- (percepatan float)- catatan picu batal
- .*freq*- (frekwensi float)- di-set / mendapatkan frekuensi (Hz)
- .*controlChange*- (Int Nomor, nilai float)- menyatakan perubahan kendali

BlowHole

- STK *phisik* alat musik tiup/klarinet model dengan satu lubang daftar dan satu *tonehole*. Daftar lubang dan satu *tonehole*.

Kelas ini didasarkan pada model alat musik tiup/klarinet, dengan penambahan suatu *two-port* mendaftarkan lubang dan suatu *three-port tonehole* dinamis implementasi, [seperti/ketika] dibahas oleh Scavone dan Cook (1998).

Di dalam implementasi ini, jarak antara lubang *reed/register* dan *tonehole/bell* adalah yang ditetapkan. Sebagai hasilnya, kedua-duanya *tonehole* dan daftar lubang akan mempunyai variabel mempengaruhi terpasang bermain frekwensi, yang mana [adalah] *dependent* terpasang panjang kolom udara. Sebagai tambahan, permainan yang paling tinggi *freqeuency* terbatas oleh ini panjangnya ditetapkan. Ini adalah suatu pandu-gelombang digital model, membuatnya penggunaan yang mungkin tunduk kepada hak paten berpegang kepada Stanford Universitas, Yamaha, dan yang lainnya.

Angka-angka pengubah kontrol :

- *Reed Stiffness* = 2
- *Noise Gain* = 4
- *Tonehole State* = 11
- *Register State* = 1
- *Breath Pressure* = 128

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

Meluas *StkInstrument [ctrl param]*

- .*reed* (float, BACA / TULIS) kekakuan reed [0.0- 1.0]
- .*noiseGain*-(float, BACA / TULIS)- komponen gangguan gain [0.0- 1.0]
- .*vent* (float, BACA / TULIS) frekuensi vent [0.0- 1.0]
- .*pressure* (float, BACA / TULIS) tekanan [0.0- 1.0]
- .*tonehole* (float, BACA / TULIS) ukuran tonehole [0.0- 1.0]
- .*startBlowing* (float, BACA / TULIS) mulai blowing [0.0- 1.0]
- .*stopBlowing* (float, BACA / TULIS) berhenti blowing [0.0- 1.0]
- .*rate* (float, BACA / TULIS) tingkat perubahan (detik)

(pewarisan dari *StkInstrument*)

- .*noteOn*- (percepatan float)- catatan picu terpasang
- .*noteOff*- (percepatan float)- catatan picu batal
- .*freq*- (frekwensi float)- di-set/mendapatkan frekuensi (Hz)
- .*controlChange*- (Int Nomor, nilai float)- menyatakan perubahan kendali

Bowed

- STK kelas instrumen *string*.

Kelas implementasi suatu model string dibungkukkan, a la Smith (1986), setelah Mcintyre, Schumacher, Woodhouse (1983).

Ini adalah suatu pandu-gelombang model digital, membuat penggunaannya yang mungkin tunduk kepada hak paten berpegang kepada Stanford Universitas, Yamaha, dan yang lainnya.

Angka-angka pengubah kontrol :

- *Bow Pressure* = 2
- *Bow Position* = 4
- *Vibrato Frekuensi* = 11
- *Vibrato Gain* = 1
- *Volume* = 128

Oleh Perry R. Cook and Gary P. Scavone, 1995 – 2002.

Meluas *StkInstrument [ctrl param]*

- .*bowPressure*-(float, BACA / TULIS)- tekanan haluan/busur [0.0- 1.0]
- .*bowPosition*-(float, BACA / TULIS)- posisi haluan/busur [0.0- 1.0]
- .*vibratoFreq*-(float, BACA / TULIS)- vibrato frekwensi (Hz)
- .*vibratoGain*-(float, BACA / TULIS)- vibrato memperoleh [0.0- 1.0]
- .*volume*-(float, BACA / TULIS)- volume [0.0- 1.0]
- .*startBowing* (float, BACA / TULIS) mulai bowing [0.0- 1.0]
- .*stopBowing* (float, BACA / TULIS) berhenti bowing [0.0- 1.0]
- .*rate* (float, BACA / TULIS)- tingkat serangan (detik)

(pewarisan dari *StkInstrument*)

- .*noteOn*- (percepatan float)- catatan picu terpasang
- .*noteOff*- (percepatan float)- catatan picu batal
- .*freq*- (frekuensi pelampung)- di-set/mendapatkan frekuensi (Hz)
- .*controlChange*- (Int Nomor, nilai float)- menyatakan perubahan kendali

Brass

- STK kelas instrumen brass sederhana.

Kelas implementasi suatu instrumen brass sederhana model pandu-gelombang, a la Cook (TBONE, Hoseplayer).

Ini adalah suatu pandu-gelombang model digital, membuat penggunaannya mungkin tunduk kepada hak paten berpegang kepada Stanford Universitas, Yamaha, dan yang lain.

Angka-angka pengubah kontrol :

- *Lip Tension* = 2
- *Slide Length* = 4
- *Vibrato Frekuensi* = 11
- *Vibrato Gain* = 1
- *Volume* = 128

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

Meluas *StkInstrument* *[ctrl param]*

- .*lip*-(float, BACA / TULIS)- tegangan lip [0.0- 1.0]
- .*slide*-(float, BACA / TULIS)- panjangnya slide [0.0- 1.0]
- .*vibratoFreq*-(float, BACA / TULIS)- vibrato frekuensi (Hz)
- .*vibratoGain*-(float, BACA / TULIS)- vibrato gain [0.0- 1.0]
- .*volume*-(float, BACA / TULIS)- volume [0.0- 1.0]
- .*clear*-(float, TULIS saja)- instrumen bersih
- .*startBlowing* (float, BACA / TULIS) start blowing [0.0- 1.0]
- .*stopBlowing* (float, BACA / TULIS) berhenti blowing [0.0- 1.0]
- .*rate* (float, BACA / TULIS) tingkat perubahan (detik)

(pewarisan dari *StkInstrument*)

- .*noteOn*- (percepatan float)- catatan picu terpasang
- .*noteOff*- (percepatan float)- catatan picu batal
- .*freq*- (frekwensi float)- set / mendapatkan frekuensi (Hz)
- .*controlChange*- (int nomor, nilai float)- menyatakan perubahan kendali

Clarinet

- STK phisik alat musik tiup/klarinet model kelas.

Kelas implementasi suatu alat musik tiup/klarinet sederhana model phisik, seperti dibahas oleh Smith (1986), Mcintyre, Schumacher, Woodhouse (1983), dan yang lain.

Ini adalah suatu pandu-gelombang model digital, membuat penggunaannya yang mungkin tunduk kepada hak paten berpegang kepada Stanford Universitas, Yamaha, dan yang lain.

Angka-angka pengubah kontrol :

- *Reed Stiffness* = 2
- *Noise Gain* = 4
- *Vibrato Frekuensi* = 11
- *Vibrato Gain* = 1
- *Breath Pressure* = 128

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

Meluas *StkInstrument* *[ctrl param]*

- .*reed*-(float, BACA / TULIS)- kekakuan reed [0.0- 1.0]
- .*noiseGain*-(float, BACA / TULIS)- komponen gangguan gain [0.0- 1.0]
- .*clear*-()- instrumen jelas
- .*vibratoFreq*-(float, BACA / TULIS)- vibrato frekuensi (Hz)
- .*vibratoGain*-(float, BACA / TULIS)- vibrato gain [0.0- 1.0]
- .*pressure*-(float, BACA / TULIS)- tekanan/volume [0.0- 1.0]
- .*startBlowing*-(float, TULIS saja)- start blowing [0.0- 1.0]
- .*stopBlowing*-(float, TULIS saja)- berhenti blowing [0.0- 1.0]
- .*rate*-(float, BACA / TULIS)- tingkat serangan (detik)

(pewarisan dari *StkInstrument*)

- .*noteOn*- (percepatan float)- catatan picu terpasang
- .*noteOff*- (percepatan float)- catatan picu batal
- .*freq*- (frekwensi float)- di-set / mendapatkan frekuensi (Hz)
- .*controlChange*- (Int Nomor, nilai float)- menyatakan perubahan kendali

Flute

- STK *phisik* flute model kelas.

Kelas implementasi suatu flute sederhana model phisik, [seperti dibahas oleh Karjalainen, Smith, Waryznyk, dan lain-lain. Pancaran model penggunaan suatu polynomial, ala Cook.

Ini adalah suatu pandu-gelombang model digital, membuat penggunaannya yang mungkin tunduk kepada hak paten berpegang kepada Stanford Universitas, Yamaha, dan yang lain.

Angka-angka pengubah kontrol :

- *Jet Delay* = 2
- *Noise Gain* = 4
- *Vibrato Frekuensi* = 11

- *Vibrato Gain* = 1
- *Breath Pressure* = 128

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

Meluas *StkInstrument [ctrl param]*

- .*jetDelay*-(float, BACA / TULIS)- penundaan pancaran[...] — item
- .*jetReflection*-(float, BACA / TULIS)- cerminan/pemantulan pancaran[...]
- .*endReflection*-(float, BACA / TULIS)- berakhirlah penundaan[...]
- .*noiseGain*-(float, BACA / TULIS)- komponen gangguan gain [0.0- 1.0]
- .*clear*-()- instrumen jelas
- .*vibratoFreq*-(float, BACA / TULIS)- vibrato frekuensi (Hz)
- .*vibratoGain*-(float, BACA / TULIS)- vibrato gain [0.0- 1.0]
- .*pressure*-(float, BACA / TULIS)- tekanan/volume [0.0- 1.0]
- .*startBowing* (float, BACA / TULIS) mulai bowing [0.0- 1.0]
- .*stopBowing* (float, BACA / TULIS) berhenti bowing [0.0- 1.0]
- .*rate* (float, BACA / TULIS)- tingkat serangan (detik)

(pewarisan dari *StkInstrument*)

- .*noteOn*- (percepatan float)- catatan picu terpasang
- .*noteOff*- (percepatan float)- catatan picu batal
- .*freq*- (frekuensi float)-di-set / mendapatkan frekuensi (Hz)
- .*controlChange*- (Int Nomor, nilai float)- menyatakan perubahan kendali

Mandolin

- STK instrumen mandolin model kelas.
- lihat *examples/mand-o-matic.ck*

Kelas ini menerima warisan dari *Plucktwo* dan penggunaan "sintesa yang diubah" teknik untuk model sebuah instrumen mandolin.

Ini adalah suatu pandu-gelombang model digital, membuat penggunaannya mungkin tunduk kepada hak paten berpegang kepada Stanford Universitas, Yamaha, dan yang lain.

Sintesa Yang diubah, khususnya, mencakup hak paten, mengabulkan, menunggu keputusan, dan/atau penggunaan. Semua ditugaskan kepada Dewan Komisaris dari Universitas Stanford. Untuk informasi, hubungi Kantor Lisensi Teknologi, Universitas Stanford.

Angka-angka pengubah kontrol :

- *Body Size* = 2
- *Pluck Position* = 4
- *String Sustain* = 11
- *String Detuning* = 1
- *Microphone Position* = 128

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

Meluas *StkInstrument [ctrl param]*

- .*bodySize* (float, BACA / TULIS) ukuran badan (persentase)
- .*pluckPos* (float, BACA / TULIS) posisi keberanian [0.0- 1.0]
- .*stringDamping* (float, BACA / TULIS) string damping [0.0- 1.0]
- .*stringDetune* (float, BACA / TULIS) detuning string pair [0.0- 1.0]
- .*afterTouch* (float, BACA / TULIS) aftertouch (sekarang ini tanpa pendukung)
- .*pluck* - (float, TULIS saja) - instrumen pluck [0.0- 1.0]

(pewarisan dari *StkInstrument*)

- .*noteOn*- (percepatan float)- catatan picu terpasang
- .*noteOff*- (percepatan float)- catatan picu batal
- .*freq*- (frekwensi float)- di-set / mendapatkan frekuensi (Hz)
- .*controlChange*- (Int Nomor, nilai float)- menyatakan perubahan kendali

ModalBar

- STK kelas instrumen bar resonan.
- lihat *examples/modalbot.ck*

Kelas ini implementasi sejumlah instrumen bar diserang berbeda. Itu menerima warisan dari Kelas Modal.

Angka-angka pengubah kontrol :

- *Stick Hardness* = 2
- *Stick Position* = 4
- *Vibrato Gain* = 11
- *Vibrato Frekuensi* = 7
- *Direct Stick Mix* = 1
- *Volume* = 128
- *Modal Presets* = 16
    - *Marimba* = 0
    - *Vibraphone* = 1
    - *Agogo* = 2
    - *Wood1* = 3
    - *Reso* = 4
    - *Wood2* = 5
    - *Beats* = 6
    - *Two Fixed* = 7
    - *Clump* = 8

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

Meluas *StkInstrument [ctrl param]*

- .*stickHardness*-(float, BACA / TULIS)- kekerasan tongkat [0.0- 1.0]
- .*strikePosition*-(float, BACA / TULIS)- posisi serangan [0.0- 1.0]
- .*vibratoFreq*-(float, BACA / TULIS)- vibrato frekwensi (Hz)
- .*vibratoGain*-(float, BACA / TULIS)- vibrato memperoleh [0.0- 1.0]
- .*directGain*-(float, BACA / TULIS)- direct gain [0.0- 1.0]
- .*masterGain*-(float, BACA / TULIS)- master gain [0.0- 1.0]
- .*volume*-(float, BACA / TULIS)- volume [0.0- 1.0]
- .*preset*-(int, BACA / TULIS)- memilih preset (lihat di atas)
- .*strike*-(float, TULIS saja)- strike bar [0.0- 1.0]
- .*damp*-(float, TULIS saja)- damp bar [0.0- 1.0]
- .*clear*-()- reset [tidak ada]

- .*mode*-(int, BACA / TULIS)- memilih mode [0.0- 1.0]
- .*modeRatio*-(float, BACA / TULIS)- mengedit perbandingan mode terpilih[...]
- .*modeRadius*-(float, BACA / TULIS)- mengedit radius mode terpilih [0.0-1.0]
- .*modeGain*-(float, BACA/TULIS)- mengedit mode terpilih memperoleh [0.0 - 1.0]

(pewarisan dari *StkInstrument*)

- .*noteOn*- (percepatan float)- catatan picu terpasang
- .*noteOff*- (percepatan float)- catatan picu batal
- .*freq*- (frekuensi float)- di-set / mendapatkan frekuensi (Hz)
- .*controlChange*- (Int Nomor, nilai float)- menyatakan perubahan kendali

Moog

- STK *moog-like* kelas sintesa sampling saringan disapu
- lihat *examples/moogie.ck*

Instrumen ini menggunakan satu serangan gelombang, satu gelombang yang berulang, dan suatu *ADSR* amplop (yang menerima warisan dari Kelas *Sampler*) dan menambahkan dua bisa menyapu formant *(Formswep)* Saringan.

Angka-angka pengubah kontrol :

- *Filter Q* = 2
- *Filter Sweep Rate* = 4
- *Vibrato Frequency* = 11
- *Vibrato Gain* = 1
- *Gain* = 128

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

Meluas *StkInstrument [ctrl param]*

- .*filterQ* - (float , BACA/TULIS) - filter Q value [0.0 - 1.0]
- .*filterSweepRate* - (float , BACA/TULIS) - filter sweep rate [0.0 - 1.0]
- .*vibratoFreq* - (float , BACA/TULIS) - vibrato frequency (Hz)

- .*vibratoGain* - (float, BACA/TULIS) - vibrato gain [0.0 - 1.0]
- .*afterTouch* - (float , TULIS saja) - aftertouch [0.0 - 1.0]

(pewarisan dari StkInstrument)

- .*noteOn*- (percepatan float)- catatan picu terpasang
- .*noteOff*- (percepatan float)- catatan picu batal
- .*freq*- (frekwensi float)- set/mendapatkan frekuensi (Hz)
- .*controlChange*- (Int Nomor, nilai float)- menyatakan perubahan kendali

Saxofony

- STK *faux* yang berbentuk kerucut mengandung kelas instrumen *reed*.

Kelas ini implementasi suatu *"hybrid"* digital instrumen pandu-gelombang yang dapat menghasilkan sebuah variasi tentang bunyi. Itu telah pula dikenal sebagai *"string blowed"* model. Bagian Pandu-Gelombang sangat utama itu suatu *string*, dengan satu kaku dan satu merugi penghentian. Fungsi non-linear adalah sebuah tabel *reed*. *String* dapat *"yang dipukul"* pada manapun menunjuk antara terminal, meskipun demikian sama halnya dengan *string*, adalah mustahil untuk menggerakan sistem akhir manapun. Jika eksitasi adalah yang ditempatkan di *string mid-point*, bunyi itu adalah suatu alat musik tiup/klarinet. Pada poin-poin yang semakin dekat kepada "jembatan", bunyi adalah yang semakin dekat untuk itu sebuah *saxophone*. Lihat Scavone (2002) untuk detail yang lebih.

Ini adalah suatu pandu-gelombang model digital, membuat penggunaannya yang mungkin tunduk kepada hak paten berpegang kepada Universitas Stanford, Yamaha, dan yang lain.

Angka-angka pengubah kontrol :

- *Reed Stiffness* = 2
- *Reed Aperture* = 26
- *Noise Gain* = 4
- *Blow Position* = 11
- *Vibrato Frequency* = 29
- *Vibrato Gain* = 1
- *Breath Pressure* = 128

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

Meluas *StkInstrument [ctrl param]*

- .*stiffness* - (float , BACA/TULIS) - reed stiffness [0.0 - 1.0]
- .*aperture* - (float , BACA/TULIS) - reed aperture [0.0 - 1.0]
- .*blowPosition* - (float , BACA/TULIS) - lip stiffness [0.0 - 1.0]
- .*noiseGain* - (float , BACA/TULIS) - noise component gain [0.0 - 1.0]
- .*vibratoFreq* - (float , BACA/TULIS) - vibrato frekuensi (Hz)
- .*vibratoGain* - (float , BACA/TULIS) - vibrato gain [0.0 - 1.0]
- .*clear* - ( ) - clear instrument
- .*pressure* - (float , BACA/TULIS) - pressure/volume [0.0 - 1.0]
- .*startBowing* (float, BACA/TULIS) begin bowing [0.0 - 1.0]
- .*stopBowing* (float, BACA/TULIS) stop bowing [0.0 - 1.0]
- .*rate* (float, BACA/TULIS) - rate of attack (sec)

(pewarisan dari StkInstrument)

- .*noteOn* - (float velocity) - trigger note on
- .*noteOff* - (float velocity) - trigger note off
- .*freq* - (float frequency) - set/get frequency (Hz)
- .*controlChange* - (int number, float value) - assert control change

Shakers

- kelas *Phisem* dan *Pholies*.
- lihat *examples/shake-o-matic.ck*

PhISEM (Physically Informed Stochastic Event Modeling) adalah suatu pendekatan algoritma untuk menirukan benturan berbagai mandiri bunyi yang memproduksi objek. Kelas ini adalah suatu meta-model yang dapat menirukan suatu *Maraca, Sekere, Cabasa, Bamboo Wind Chimes, Water Drops, Tambourine, Sleighbells,* dan suatu *Guiro*.

PhOLIES (Physically-Oriented Library of Imitated Environmental Sounds) adalah suatu pendekatan serupa untuk sintesa dari lingkungan bunyi. Kelas ini

implementasi simulasi dari *breaking sticks*, *crunchy snow* (atau tidak), *suatu wrench*, *sandpaper*, dan lebih.

Angka-angka pengubah kontrol :

- *Shake Energy* = 2
- *System Decay* = 4
- *Number Of Objects* = 11
- *Resonance Frequency* = 1
- *Shake Energy* = 128
- *Instrument Selection* = 1071
    - *Maraca* = 0
    - *Cabasa* = 1
    - *Sekere* = 2
    - *Guiro* = 3
    - *Water Drops* = 4
    - *Bamboo Chimes* = 5
    - *Tambourine* = 6
    - *Sleigh Bells* = 7
    - *Sticks* = 8
    - *Crunch* = 9
    - *Wrench* = 10
    - *Sand Paper* = 11
    - *Coke Can* = 12
    - *Next Mug* = 13
    - *Penny + Mug* = 14
    - *Nickle + Mug* = 15
    - *Dime + Mug* = 16
    - *Quarter + Mug* = 17
    - *Franc + Mug* = 18
    - *Peso + Mug* = 19
    - *Big Rocks* = 20
    - *Little Rocks* = 21

- *Tuned Bamboo Chimes* = 22

Oleh Perry R. Cook, 1996 - 1999.

Meluas *StkInstrument* *[ctrl param]*

- .*preset* - (int , BACA/TULIS) - pemilihan instrument (0 - 22; lihat diatas)
- .*energy* - (float , BACA/TULIS) - shake energy [0.0 - 1.0]
- .*decay* - (float , BACA/TULIS) - system decay [0.0 - 1.0]
- .*objects* - (float , BACA/TULIS) - number of objects [0.0 - 128.0]

(pewarisan dari *StkInstrument*)

- .*noteOn* - (float velocity) - trigger note on
- .*noteOff* - (float velocity) - trigger note off
- .*freq* - (float frequency) - set/get frequency (Hz)
- .*controlChange* - (int number, float value) - assert control change

Sitar

- STK kelas model string sitar

Kelas ini implementasi suatu sitar dawai petik model phisik berdasar pada algoritma *Karplus-Strong*.

Ini adalah suatu pandu-gelombang digital model, membuat penggunaannya yang mungkin tunduk kepada hak paten berpegang kepada Universitas Stanford, Yamaha, dan yang lain.

Di sana ada sedikitnya dua hak paten, menugaskan untuk Stanford, tegas nama Karplus dan/atau Strong.

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

Meluas *StkInstrument* *[ctrl param]*

- .*pluck* (float, TULIS saja) pluck string [0.0 - 1.0]
- .*clear* () reset

(pewarisan dari *StkInstrument*)

- .*noteOn* - (float velocity) - trigger note on
- .*noteOff* - (float velocity) - trigger note off
- .*freq* - (float frequency) - set/get frequency (Hz)

- .*controlChange* - (int number, float value) - assert control change

StifKarp

- STK memetik instrumen dawai kaku.
- lihat *examples/stifkarp.ck*

Kelas ini implementasi suatu dawai dipetik sederhana algoritma (Karplus Strong) dengan peningkatan (Jaffe-Smith, Smith, dan yang lain), termasuk kekakuan dawai dan posisi keberanian mengendalikan.

Kekakuan diperagakan dengan *allpass* saringan.

Ini adalah suatu pandu-gelombang digital model, membuat penggunaannya yang mungkin tunduk kepada hak paten berpegang kepada Universitas Stanford, Yamaha, dan yang lain.

Angka-angka pengubah kontrol :

- *Pickup Position* = 4
- *String Sustain* = 11
- *String Stretch* = 1

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

Meluas *StkInstrument [ctrl param]*

- .*pickupPosition* - (float , BACA/TULIS) -posisi pickup [0.0 - 1.0]
- .*sustain* - (float , BACA/TULIS) - string sustain [0.0 - 1.0]
- .*stretch* - (float , BACA/TULIS) - string stretch [0.0 - 1.0]
- .*pluck* - (float , TULIS saja) - pluck string [0.0 - 1.0]
- .*baseLoopGain* - (float , BACA/TULIS) - ?? [0.0 - 1.0]
- .*clear* - ( ) - reset instrument

(pewarisan dari *StkInstrument*)

- .*noteOn* - (float velocity) - trigger note on
- .*noteOff* - (float velocity) - trigger note off
- .*freq* - (float frequency) - set/get frequency (Hz)
- .*controlChange* - (int number, float value) - assert control change

VoicForm

- Empat formant instrumen sintese.
- lihat *examples/voic-o-form.ck*

Instrumen ini berisi suatu eksitasi yang menyanyi *wavetable* (pengulangan gelombang dengan acak dan *vibrato* berkala, memperlancar pada frekwensi, dll.), gangguan eksitasi, dan empat yang bisa menyapu resonansi kompleks.

Formant yang diukur Data dimasukkan, dan cukup data ada di sana untuk mendukung baik paralel maupun sintese air terjun kecil. Di (dalam) kasus *floating point* sintese air terjun kecil adalah [yang] yang paling alami maka itu adalah apa yang kamu temukan di sini.

Angka-angka pengubah kontrol :

- *Voiced/Unvoiced Mix* = 2
- *Vowel/Phoneme Selection* = 4
- *Vibrato Frequency* = 11
- *Vibrato Gain* = 1
- *Loudness (Spectral Tilt)* = 128

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

Meluas *StkInstrument [ctrl param]*

- .*phoneme* (string, BACA/TULIS) pemilihan phoneme (above)
- .*phonemeNum* - (int , BACA/TULIS) - pemilihan phoneme berdasarkan angka [0.0 - 128.0]
- .*speak* (float, TULIS saja) start singing [0.0 - 1.0]
- .*quiet* (float, TULIS saja) stop singing [0.0 - 1.0]
- .*voiced* (float, BACA/TULIS) set mix for voiced component [0.0 - 1.0]
- .*unVoiced* (float, BACA/TULIS) set mix for unvoiced componenet [0.0 - 1.0]
- .*pitchSweepRate* (float, BACA/TULIS) pitch sweep [0.0 - 1.0]
- .*voiceMix* (float, BACA/TULIS) voiced/unvoiced mix [0.0 - 1.0]
- .*vibratoFreq* (float, BACA/TULIS) vibrato frequency (Hz)
- .*vibratoGain* (float, BACA/TULIS) vibrato gain [0.0 - 1.0]

- .*loudness* (float, BACA/TULIS) 'loudness' of voice [0.0 - 1.0]

(pewarisan dari *StkInstrument*)

- .*noteOn* - (float velocity) - trigger note on
- .*noteOff* - (float velocity) - trigger note off
- .*freq* - (float frequency) - set/get frequency (Hz)
- .*controlChange* - (int number, float value) - assert control change

STK - FM Synths

FM

- STK memisahkan FM kelas dasar sintese

Kelas ini mengendalikan suatu jumlah semaunya gelombang dan amplop, menentukan via suatu argumentasi pembangun.

Angka-angka pengubah kontrol :

- *Control One* = 2
- *Control Two* = 4
- *LFO Speed* = 11
- *LFO Depth* = 1
- *ADSR 2 & 4 Target* = 128

basis dasar *CHOWNING/STANFORD FM* hak paten berakhir pada 1995, tetapi di sana ada follow-on hak paten, kebanyakan ditugaskan ke Yamaha. Jika kamu menjadi tipe siapa yang cemas akan ini (pembuatan uang) keraguan pergi.

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

[ctrl param]

- .*lfoSpeed* (float, BACA/TULIS) modulation speed (Hz)
- .*lfoDepth* (float, BACA/TULIS) modulation depth [0.0 - 1.0]
- .*afterTouch* (float, BACA/TULIS) aftertouch [0.0 - 1.0]
- .*control1* (float, BACA/TULIS) FM control 1 [instrument specific]
- .*control2* (float, BACA/TULIS) FM control 2 [instrument specific]

Ini (pewarisan dari *StkInstrument*)

- *.noteOn* - (float velocity) - trigger note on
- *.noteOff* - (float velocity) - trigger note off
- *.freq* - (float frequency) - set/get frequency (Hz)
- *.controlChange* - (int number, float value) - assert control change

BeeThree

- *Stk Hammond-oid* Organ/ bagian badan FM instrumen sintese.

Kelas ini implementasi suatu topologi 4 operator sederhana, juga dikenal sebagai algoritma 8 dari TX81Z.

\code

Algoritma 8 adalah :

```
1 --.
2 -\|
+-> Out
3 -/|
4 --
```

\endcode

Angka pengubah kontrol :

- *Operator 4 (feedback) Gain* = 2
- *Operator 3 Gain* = 4
- *LFO Speed* = 11
- *LFO Depth* = 1
- *ADSR 2 & 4 Target* = 128

Dasar hak paten *CHOWNING/STANFORD FM* berakhir di 1995, tetapi di sana eksis mengikuti pada hak paten, kebanyakan ditugaskan ke Yamaha. Jika kamu menjadi tipe dari siapa yang cemas akan ini (pembuatan uang) tidak perlu ragu.

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

Meluas *FM*

[ctrl param]

- (lihat super kelas)

FMVoices

- STK instrumen sintesa yang menyanyi FM.

Kelas ini implementasi 3 pengangkut dan modulator umum, juga dikenal sebagai algoritma 6 dari TX81Z.

\code

Algoritma 6 adalah :

```
/->1 -\
4-|-->2 - +-> Out
\->3 -/
```

\endcode

Angka-angka pengubah kontrol :

- *Vowel* = 2
- *Spectral Tilt* = 4
- *LFO Speed* = 11
- *LFO Depth* = 1
- *ADSR 2 & 4 Target* = 128

Dasar hak paten *CHOWNING/STANFORD FM* berakhir di 1995, tetapi di sana ada yang eksis mengikuti pada hak paten, kebanyakan ditugaskan ke Yamaha. Jika kamu adalah jenis siapa yang cemas tentang ini (pembuatan uang) keraguan pergi.

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

Meluas *FM*

[ctrl param]

- .*vowel* (float, TULIS saja) pemilihan vowel [0.0 - 1.0]
- .*spectralTilt* (float, TULIS saja) spectral tilt [0.0 - 1.0]
- .*adsrTarget* (float, TULIS saja) adsr targets [0.0 - 1.0]

HevyMetl

- STK instrumen sintese metal heavy FM.

Kelas ini implementasi 3 operator cascade dengan modulasi umpan balik, juga dikenal sebagai algoritma 3 dari TX81Z.

Algoritma 3 adalah : 4--\

3-->2-- + -->1-->Out

Angka-angka pengubah kontrol :

- *Total Modulator Index* = 2
- *Modulator Crossfade* = 4
- *LFO Speed* = 11
- *LFO Depth* = 1
- *ADSR 2 & 4 Target* = 128

Dasar hak paten *CHOWNING/STANFORD FM* berakhir di 1995, tetapi di sana ada yang eksis mengikuti pada hak paten, kebanyakan ditugaskan ke Yamaha. Jika kamu menjadi jenis siapa yang cemas akan ini (mencari uang) tidak ada kekuatiran.

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

Meluas FM

[ctrl param]

- (lihat super kelas)

PercFlut

- STK instrumen sintese perkusi flute FM.

Kelas ini implementasi algoritma 4 dari TX81Z.

\code

Algoritma 4 adalah : 4->3--\

2-- + -->1-->Out

\endcode

Angka-angka pengubah kontrol :

- *Total Modulator Index* = 2
- *Modulator Crossfade* = 4
- *LFO Speed* = 11
- *LFO Depth* = 1

- *ADSR 2 & 4 Target* = 128

Dasar hak paten *CHOWNING/STANFORD FM* berakhir di 1995, tetapi di sana ada yang eksis mengikuti pada hak paten, kebanyakan ditugaskan ke Yamaha. Jika kamu menjadi jenis siapa yang cemas akan ini (pembuatan uang) jangan kuatir.

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

Meluas FM

[ctrl param]

- (lihat super kelas)

Rhodey

- *STK Fender Rhodes*-seperti piano elektris FM
- lihat examples/rhodey.ck

instrumen sintesa.

Kelas ini implementasi dua pasangan FM sederhana yang dijumlahkan bersama-sama, juga dikenal sebagai algoritma 5 dari TX81Z.

\code

Algoritma 5 adalah : 4->3--\

+ --> Out

2->1--/

\endcode

Angka-angka pengubah kontrol :

- *Modulator Index One* = 2
- *Crossfade of Outputs* = 4
- *LFO Speed* = 11
- *LFO Depth* = 1
- *ADSR 2 & 4 Target* = 128

Dasar hak paten *CHOWNING/STANFORD FM* berakhir di 1995, tetapi di sana ada yang eksis mengikuti pada hak paten, kebanyakan ditugaskan ke Yamaha.

Jika kamu menjadi jenis siapa yang cemas akan ini (pembuatan uang) jangan kuatir.

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

Meluas FM

[ctrl param]

- (lihat super kelas)

TubeBell

- STK bel berbentuk pipa (bunyi genta berkenaan dengan orkes) FM
- meluas FM

instrumen sintesa.

Kelas ini implementasi pasangan dua FM sederhana yang dijumlahkan bersama-sama, juga dikenal sebagai algoritma 5 dari TX81Z.

\code

Algoritma 5 adalah : 4->3--\

+ --> Out

2->1--/

\endcode

Angka-angka pengubah kontrol :

- *Modulator Index One* = 2
- *Crossfade of Outputs* = 4
- *LFO Speed* = 11
- *LFO Depth* = 1
- *ADSR 2 & 4 Target* = 128

Dasar hak paten *CHOWNING/STANFORD FM* berakhir di 1995, tetapi di sana ada yang eksis mengikuti pada hak paten, kebanyakan ditugaskan ke Yamaha. Jika kamu menjadi jenis siapa yang cemas akan ini (pembuatan uang) jangan kuatir.

Oleh Perry R. Cook and Gary P. Scavone, 1995 - 2002.

[ctrl param]

- (lihat super kelas)

Wurley

- STK *Wurlitzer* piano elektris FM
- lihat *examples/wurley.ck* meluas
  instrumen sintesa.

Kelas ini implementasi pasangan dua FM sederhana yang dijumlahkan bersama-sama, juga dikenal sebagai algoritma 5 dari TX81Z.

\code

Algoritma 5 adalah : 4->3--\
+ --> Out
2->1--/

\endcode

Angka-angka pengubah kontrol :

- *Modulator Index One* = 2
- *Crossfade of Outputs* = 4
- *LFO Speed* = 11
- *LFO Depth* = 1
- *ADSR 2 & 4 Target* = 128

Dasar hak paten *CHOWNING/STANFORD FM* berakhir di 1995, tetapi di sana ada yang eksis mengikuti pada hak paten, kebanyakan ditugaskan ke Yamaha. Jika kamu menjadi jenis siapa yang cemas akan ini (pembuatan uang) jangan kuatir.

Oleh Perry R. Cook and Gary P. Scavone, 1995 – 2002.

Meluas FM

[ctrl param]

- (lihat super kelas)

STK - Delay

Delay

- STK kelas garis penundaan yang *non-interpolating*
- lihat *examples/net relay.ck*

Ini Saringan dilindungi *subclass* implementasi suatu *delay-line* digital *non-interpolating*. Panjang maksimum ditetapkan dari 4095 dan suatu penundaan nol di-set menggunakan konstruktor default. Sebagai alternatif, penundaan dan panjangnya maksimum dapat menata instantiation suatu konstruktor yang dimuati berlebihan.

Suatu garis penundaan non-interpolating secara khas digunakan menetapkan aplikasi panjang delay, seperti untuk gema.

Oleh Perry R. Cook and Gary P. Scavone, 1995 – 2002.

[ctrl param]

- .*delay* (float, BACA/TULIS) panjang dari delay
- .*max* (float, BACA/TULIS) max delay (buffer size)

DelayA

- STK allpass menyisipkan kelas garis penundaan.

Penundaan Ini *subclass* implementasi suatu panjang fraksional *delay-line* digital yang menggunakan suatu *first-order allpass* saringan. Panjang maksimum ditetapkan dari 4095 dan suatu penundaan 0.5 di-set menggunakan konstruktor default. Sebagai alternatif, penundaan dan panjang maksimum dapat menata instantiation suatu konstruktor dimuati berlebihan.

Suatu allpass saringan mempunyai kesatuan penting *gain* tetapi variabel fase memperlambat properti, pembuatan itu bermanfaat dalam mencapai delay fraksional tanpa mempengaruhi suatu respon sinyal frekuensi penting. Di dalam memesan untuk mencapai suatu respon delay suatu fase flat secara maksimal, penundaan minimum yang mungkin di dalam implementasi terbatas ini pada suatu nilai 0.5.

Oleh Perry R. Cook and Gary P. Scavone, 1995 – 2002.

[ctrl param]

- *.delay* (float, BACA/TULIS) panjang dari delay
- *.max* (float, BACA/TULIS) max delay (buffer size)

DelayL

- STK linier yang menyisipkan kelas garis penundaan.
- lihat *examples/i-robot.ck*

Penundaan Ini subclass implementasi suatu panjang fraksional *delay-line* digital yang menggunakan *first-order* sisipan linier. Panjang maksimum ditetapkan dari 4095 dan suatu penundaan nol di-set menggunakan konstruktor default. Sebagai alternatif, menunda dan panjang maksimumnya dapat di-set instantiation dengan suatu konstruktor dimuati berlebihan.

Sisipan linier adalah suatu teknik efisien untuk menuju keberhasilan panjangnya penundaan fraksional, meskipun demikian itu memperkenalkan isyarat frekuensi tinggi pelaifan bagi bermacam-macam derajat tingkat yang tergantung pada seting penundaan faksional. Penggunaan dari order *Lagrange interpolators* lebih tinggi dapat secara khas meningkatkan (memperkecil) karakteristik pelaifan ini.

Oleh Perry R. Cook and Gary P. Scavone, 1995 – 2002.

[ctrl param]

- *.delay* (float, BACA/TULIS) panjang dari delay
- *.max* (float, BACA/TULIS) max delay (buffer size)

Echo

- STK kelas efek echo.

Kelas ini implementasi suatu efek echo.

Oleh Perry R. Cook and Gary P. Scavone, 1995 – 2002.

[ctrl param]

- *.delay* (float, BACA/TULIS) panjang dari echo
- *.max* (float, BACA/TULIS) max delay
- *.mix* (float, BACA/TULIS) mix level (wet/dry)

STK - Envelopes

Envelope

- STK kelas basis envelope.
- Lihat *examples/sixty.ck*

Ini kelas implementasi suatu envelope sederhana generator yang mana adalah mampu dari ramping untuk suatu nilai target oleh suatu spesifikasi \e rate. Itu juga bereaksi terhadap *keyOn* sederhana \e dan \e *keyOff* pesan, ramping untuk 1.0 pada *keyOn* dan untuk 0.0 pada *keyOff*.

Oleh Perry R. Cook and Gary P. Scavone, 1995 – 2002.

[ctrl param]

- .*keyOn* (int, TULIS saja) ramp to 1.0
- .*keyOff* (int, TULIS saja) ramp to 0.0
- .*target* (float, BACA/TULIS) ramp to arbitrary value
- .*time* (float, BACA/TULIS) time to reach target (in second)
- .*duration* (dur, BACA/TULIS) time to reach target
- .*rate* (float, BACA/TULIS) rate of change
- .*duration* (dur, BACA/TULIS) duration of steady state
- .*value* (float, BACA/TULIS) set immediate value

ADSR

- STK kelas envelope ADSR.
- Lihat *examples/adsr.ck*

Envelope Ini *subclass* implementasi suatu ADSR tradisional (Serangan, Kebusukan, Mendukung, Lepaskan) envelope. Itu bereaksi terhadap *keyOn* sederhana dan *keyOff* pesan, mengawasi tentang statusnya. Status \e= *ADSR::DONE* setelah envelope menghargai jangkauan 0.0 di dalam status *ADSR::RELEASE*.

Oleh Perry R. Cook and Gary P. Scavone, 1995 – 2002.

[ctrl param]

- .*keyOn* (int, TULIS saja) start the attack for non-zero values
- .*keyOff* (int, TULIS saja) start the release for non-zero values
- .*attackTime* (float, TULIS saja) attack time
- .*attackRate* (float, BACA/TULIS) attack rate

- .*decayTime* (float, BACA/TULIS) decay
- .*decayRate* (float, BACA/TULIS) decay rate
- .*sustainLevel* (float, BACA/TULIS) sustain level
- .*releaseTime* (float, BACA/TULIS) release time
- .*releaseRate* (float, BACA/TULIS) release rate
- .*state* (int, BACA saja) attack=0, decay=1, sustain=2, release=3,done=4

STK - Reverbs

JCRev

- kelas pemantulan suara John Chowning's.

Kelas ini diperoleh dari fungsi *CLM JCREV*, yang mana adalah didasarkan pada penggunaan jaringan dari *allpass* sederhana dan menyisir penundaan menyaring. Kelas ini implementasi tiga rangkaian unit *allpass*, yang diikuti oleh empat sisir paralel saringan, dan dua *decorrelation* menunda bentuk dalam paralel di keluaran itu.

Oleh Perry R. Cook and Gary P. Scavone, 1995 – 2002.

[ctrl param]

- .*mix* (float, BACA/TULIS) mix level

NRev

- Kelas pemantulan suara *CCRMA's NRev*.

Kelas ini diperoleh dari fungsi *CLM NREV*, yang mana adalah didasarkan pada penggunaan jaringan dari allpass sederhana dan penundaan sisir menyaring. Pengaturan tertentu ini berisi tentang 6 sisir menyaring paralel, yang diikuti oleh 3 *allpass* saringan, suatu *lowpass* saringan, dan *allpass* yang lain secara urut, yang diikuti oleh dua *allpass* menyaring yang paralel dengan bersesuaian keluaran kanan dan kiri.

Oleh Perry R. Cook and Gary P. Scavone, 1995 – 2002.

[ctrl param]

- .*mix* (float, BACA/TULIS) mix level

PRCRev

- kelas pemantulan suara *Perry'S* sederhana.

Kelas ini didasarkan pada sebagian dari pengembalian kata kerja *Stanford/Ccrma* yang terkenal itu *(NREV, Kiprev*), Yang telah didasarkan pada *Chowning/Moorer/Schroeder* pemantulan suara yang menggunakan jaringan dari *allpass* sederhana dan penundaan sisir menyaring. Kelas ini implementasi dua rangkaian unit *allpass* dan dua sisir paralel saringan.

Oleh Perry R. Cook and Gary P. Scavone, 1995 – 2002.

[ctrl param]

- .*mix* (float, BACA/TULIS) mix level

STK - Components

Chorus

- STK kelas efek *chorus*.

Kelas ini implementasi suatu efek *chorus*.

Oleh Perry R. Cook and Gary P. Scavone, 1995 – 2002.

[ctrl param]

- .*modFreq* (float, BACA/TULIS) frekuensi modulation
- .*modDepth* (float, BACA/TULIS) kedalaman modulation
- .*mix* (float, BACA/TULIS) effect mix

Modulate

- STK modulator periodik/acak.

Kelas ini kombinasi modulasi periodik dan acak untuk memberi suatu fungsi modulasi manusia alami.

Oleh Perry R. Cook and Gary P. Scavone, 1995 – 2002.

[ctrl param]

- .*vibratoRate* (float, BACA/TULIS) set rate of vibrato
- .*vibratoGain* (float, BACA/TULIS) gain for vibrato
- .*randomGain* (float, BACA/TULIS) gain untuk kontribusi acak

PitShift

- STK kelas efek bergeser titi nada sederhana.

Kelas ini implementasi suatu titi nada bergeser sederhana yang menggunakan bentuk penundaan.

Oleh Perry R. Cook and Gary P. Scavone, 1995 – 2002.

[ctrl param]

- .*mix* (float, BACA/TULIS) efek dry/wet mix level
- .*shift* (float, BACA/TULIS) degree of pitch shifting

SubNoise

- STK pembangkit suara gaduh *sub-sampled*.

Hasilkan suatu nomor baru yang acak setiap *"rate"* detak menggunakan fungsi C rand(). Kualitas dari fungsi rand() bervariasi dari satu OS ke OS lainnya.

Oleh Perry R. Cook and Gary P. Scavone, 1995 – 2002.

[ctrl param]

- .*rate* (int, BACA/TULIS) subsampling rate

STK - File I/O

WvIn

- STK kelas basis input data audio.

Kelas ini menyediakan pendukungan masukan untuk berbagai format file audio. Itu juga bertindak sebagai suatu dasar kelas untuk *"realtime"* subkelas *streaming*.

Wvin memuat indeks dari suatu file audio untuk keluaran yang berikut. Sisipan linier digunakan untuk fraksional "rate yang dibaca".

Wvin mendukung *multi-channel* data terselip di antara halaman format. Adalah penting untuk mencirikan metoda tick(), yang mengembalikan contoh produksi dengan rata-rata ke seberang bingkai sederhana, dari metoda tickFrame(), yang

kembali ke pointer untuk frame sederhana *multi-channel*. Untuk *single-channel* data, metoda ini mengembalikan nilai-nilai padanan.

File kecil dibaca selengkapnya kepada memori lokal selama *instantiation*. File besar dibaca *incrementally* dari disk. Batas ukuran file dan nilai-nilai ukuran kenaikan didefinisikan didalam Wvin.H.

Wvin yang sekarang ini mendukung WAV, AIFF, SND (AU), MAT-FILE (Matlab), dan STK Format File mentah. Bilangan bulat yang ditandai (8-, 16-, dan 32-bit) dan *floatingpoint* (32- dan 64-bit) adalah jenis data yang didukung. Jenis Data yang tidak dimampatkan tidaklah didukung. Jika penggunaan MAT-FILES, data harus diselamatkan didalam suatu array dengan saluran data masing-masing yang mengisi suatu baris acuan atau matriks.

Oleh Perry R. Cook and Gary P. Scavone, 1995 – 2002.

[ctrl param]

- .*rate* (float, BACA/TULIS) memainkan kembali rate
- .*path* (string, BACA/TULIS) specifies file to be played

WaveLoop

- STK kelas osilator waveform.
- Lihat *examples/dope.ck*

[ctrl param]

- .*freq* (float, BACA/TULIS) frequency dari playback (loops/second)
- .*addPhase* (float, BACA/TULIS) offset dari phase
- .*addPhaseOffset* (float, BACA/TULIS) set phase offset

WvOut

- STK kelas basis output data audio.

Kelas ini menyediakan pendukungan keluaran untuk berbagai format file audio. Itu juga bertindak sebagai suatu dasar kelas untuk *"realtime"* subkelas streaming.

Wvout menulis contoh untuk suatu file audio. Itu mendukung data *multi-channel* terselip di antara halaman format. Adalah penting untuk mencirikan metoda tick(),

dimana contoh tunggal keluaran untuk semua saluran di dalam suatu contoh frame, dari metoda tickFrame(), yang mengambil suatu pointer ke *multi-channel* frame data sederhana.

Wvout yang sekarang ini mendukung WAV, AIFF, AIFC, SND (AU), MAT-FILE (Matlab), dan STK format File mentah. Bilangan bulat yang ditandai (8-, 16-, dan 32-bit) dan *floating-point* (32- dan 64-bit) adalah jenis data yang didukung. STK File mentah menggunakan 16-bit bilangan bulat menurut definisi. MAT-FILES akan selalu menjadi tertulis seperti 64-bit float. Jika suatu spesifikasi jenis data tidak memenuhi tipe file yang ditetapkan, tipe data akan secara otomatis menjadi dimodifikasi. Jenis Data yang tidak dimampatkan adalah tidak didukung.

Sekarang ini, Wvout adalah *non-interpolating* dan rate keluaran selalu *Stk::sampleRate()*.

Oleh Perry R. Cook and Gary P. Scavone, 1995 – 2002.

[ctrl param]

- .*matFilename* (string, TULIS saja) membuka sebuah file matlab untuk menulis
- .*sndFilename* (string, TULIS saja) membuka file snd untuk menulis
- .*wavFilename* (string, TULIS saja) membuka file WAVE untuk menulis
- .*rawFilename* (string, TULIS saja) membuka file raw untuk menulis
- .*aifFilename* (string, TULIS saja) membuka file AIFF untuk menulis
- .*closeFile* (string, TULIS saja) menutup file properly

Events

Event

- *event handler*

[ctrl param]

- .*signal()* sinyal pertama menunggu shred
- .*broadcast()* sinyal semua shreds

MidiIn

- Penerima MIDI

Meluas *Event* *[ctrl param]*

- .*open* (int, BACA/TULIS) set port untuk menerima
- .*recv* (MidiMsg, BACA) menerima MIDI input

(lihat MIDI tutorial)

MidiOut

- Pengirim MIDI

Meluas *Event* *[ctrl param]*

- .*open* (int, BACA/TULIS) set port untuk kirim
- .*send* (MidiMsg, TULIS) pengiriman MIDI output

(lihat tutorial MIDI)

MidiMsg

- Pemilik data MIDI

[ctrl param]

- .*data1* (int, BACA/TULIS) byte pertama dari data (member variable)
- .*data2* (int, BACA/TULIS) byte kedua dari data (member variable)
- .*data3* (int, BACA/TULIS) byte ketiga dari data (member variable)

(lihat tutorial MIDI)

OscRecv

- Membuka Penerima Kendali Bunyi

[ctrl param]

- .*port* (int, BACA/TULIS) set port untuk menerima
- .*listen* () starts object listening to port
- .*event* (string(name), string(type), BACA/TULIS) mendefinisikan string dari event untuk menerima

(lihat tutorial events)

OscSend

- Membuka pengirim kendali bunyi

[ctrl param]

- .*setHost* (string(host), int(port), BACA/TULIS) set port di dalam the host untuk menerima
- .*startMsg* (string(name), string(type), BACA/TULIS) mendefinisikan string dari event untuk mengirim

(lihat tutorial events)

OscEvent

- Membuka event kendali bunyi

Meluas *event* *[ctrl param]*

- .*nextMsg* (int, BACA) the number of messages in queue
- .*getFloat* (float, BACA) gets float from message
- .*getInt* (int, BACA) gets int from message

(lihat tutorial events)

KBHit

- *ascii keyboard event*

meluas *event* *[ctrl param]*

- .*getchar* (int, BACA) ascii value
- .*more* (int, BACA saja) returns 1 if multiple keys have been pressed

(lihat tutorial events)

HidIn

- Penerima HID

Meluas *event* *[ctrl param]*

- .*openJoystick* (int(which), TULIS saja) open joystick number
- .*openMouse* (int(which), TULIS saja) open mouse number
- .*openKeyboard* (int(which), TULIS saja) open keyboard number
- .*num* () return joystick/mouse/keyboard number

- *.recv* (HidMsg, BACA saja) menulis message selanjutnya available dari device ini dari argumen
- *.name* () return device name

(lihat tutorial events)

HidMsg

- Pemilik data HID

[ctrl param]

- *.isAxisMotion* (int, BACA saja) non-zero jika message ini korespondensi untuk movement dari ajoystick axis
- *.isButtonDown* (int, BACA saja) non-zero jika message ini korespondensi untuk button down atau key down dari beberapa tipe device
- *.isButtonUp* (int, BACA saja) non-zero jika message ini korespodensi untuk button up atau key up dari beberapa tipe device
- *.isMouseMotion* (int, BACA saja) non-zero jika message ini korespodensi untuk motion dari sebuah pointing device
- *.isHatMotion* (int, BACA saja) non-zero jika message ini korespodensi untuk motion dari a joystick hat, point-of-view switch, atau directional pad
- *.which* (int, BACA/TULIS) HID element number (member variable)
- *.axisPosition* (float, BACA/TULIS) posisi dari joystick axis didalam range [-1.0, 1.0] (member variable)
- *.deltaX* (float, BACA/TULIS) mengubah X axis dari pointing device (member variable)
- *.deltaY* (float, BACA/TULIS) mengubah Y axis dari pointing device (member variable)
- *.deviceNum* (float, BACA/TULIS) device number which produced this message (member variable)
- *.deviceType* (float, BACA/TULIS) device type which produced this message (member variable)
- *.type* (int, BACA/TULIS) message/HID element type (member variable)

- .*idata* (int, BACA/TULIS) data (member variable)
- .*fdata* (int, BACA/TULIS) data (member variable)

(lihat tutorial events)

DAFTAR PUSTAKA

1. http://chuck.cs.princeton.edu/
2. http://chuck.cs.princeton.edu/doc/
3. http://chuck.cs.princeton.edu/doc/learn/
4. http://audicle.cs.princeton.edu/mini/
5. http://audicle.cs.princeton.edu/
6. http://chuck.cs.princeton.edu/wiki
7. http://www.sourceforge.net/chuck
8. http://en.wikipedia.org/wiki/music
9. http://id.wikipedia.org/wiki/musik

DAFTAR PUSTAKA

1. http://chuck.cs.princeton.edu/
2. http://chuck.cs.princeton.edu/doc/
3. http://chuck.cs.princeton.edu/doc/learn/
4. http://audicle.cs.princeton.edu/mini/
5. http://audicle.cs.princeton.edu/
6. http://chuck.cs.princeton.edu/wiki
7. http://www.sourceforge.net/chuck
8. http://en.wikipedia.org/wiki/music
9. http://id.wikipedia.org/wiki/musik

Penerbit Gunadarma