The YaPPP Manuel

Version 0.7

Emanuel Wittersheim

Inhaltsverzeichnis

Teil I.

Einleitung

1. Was und warum ist YaPPP?

TBC

2. Installation

2.1. Abhängigkeiten

Bevor Sie YaPPP installieren, sollten Sie sicher stellen, dass folgende Programme lauffähig auf Ihrem System installiert sind:

- csound (aktuelle Version: 5.18)[1]
- scheme (ich empfehle Guile, jede andere Version sollte es jedoch auch tun[2])

Nicht zwangsläufig notwendig, aber empfehlenswert ist außerdem der Texteditor Emacs (aktuelle Version: 24.2)[3]

2.2. Installation

Die Installation von YaPPP ist denkbar einfach: Laden Sie die ZIP-Datei herunter und extrahieren Sie sie irgendwohin. Öffnen sie den Texteditor Ihres Vertrauens (ich rate Ihnen dringendst zu Emacs), starten Sie Scheme und legen Sie eine Datei z.B. mit dem Namen "`expample01.scm`" an. Sagen Sie Scheme, wo Ihre YaPPP-Dateien liegen (passen Sie den String in `YaPPP-In` an ihren Pfad an):

```
(define YaPPP-In "/home/emanuel/bin/YaPPP_0.7/")

(define (LoadYaPPP)
  (load (string-append YaPPP-In "Object_Framework.scm"))
  (load (string-append YaPPP-In "tierchen_160113.scm"))
  (load (string-append YaPPP-In "ugens_030213.scm"))
  (load (string-append YaPPP-In "schedule_030213.scm"))
  (load (string-append YaPPP-In "formant_ugens_220113.scm")))

(LoadYaPPP)
```

Evaluieren sie nun diesen Quellcode. Soweit keine Fehlermeldungen kommen: Los geht's!

[1] siehe: `www.csounds.com`
[2] siehe: `www.schemers.org` und: `www.gnu.org/software/guile/`
[3] siehe: `www.gnu.org/software/emacs/`

2.3. Bekannte Fehler

- Der Csound-Aufruf aus YaPPP heraus scheint nicht richtig zu funktionieren, weshalb er in `run_csound` auskommentiert ist. Csound sollte also extern übers Terminal aufgerufen werden.
- Die Track-Funktion scheint ebenfalls nicht richtig zu arbeiten. Kann gerne korrigiert werden, ansonsten empfehle ich, mit dem Verwenden bis zum nächsten Release zu warten.

2.4. Upcoming Things

Neben der Ausgabe im Csound-Score-Format ist auf längere Sicht eine Unterstützung des Fomus-Formates[4] und somit eine Ausgabe nach Lilypond[5] bzw. MusicXML[6] (zur Weiterverarbeitung in Finale oder Sibelius) geplant.

[4] siehe: `http://fomus.sourceforge.net/`
[5] siehe: `www.lilypond.org`
[6] siehe: `http://libmusicxml.sourceforge.net/`

3. YaPPP — So Geht's

TBC

Teil II.

Unit Generators

4. Linien und Funktionen

4.1. LineMaker

```
(LineMaker x0 y0 x1 y1)
```

Implementiert eine lineare Funktion der Form $f(x) = mx + b$.

- Initialisierung:

```
(define <name> (LineMaker x0 y0 x1 y1))
```

- Methoden:

```
(send <name> 'type)
(send <name> 'next [this-time])
(send <name> 'reset!)
```

4.2. ExptLineMaker

```
(ExptLineMaker a r)
```

Implementiert eine Potenzfunktion der Form $f(x) = ax^r$

- Initialisierung:

```
(define <name> (ExptLineMaker a r))
```

- Methoden:

```
(send <name> 'type)
(send <name> 'next [this-time])
(send <name> 'reset!)
```

5. Zufall

5.1. RandomMaker

```
(RandomMaker ug og)
```

Implementiert einen Zufallsgeerator, welcher Werte zwischen ug und og (exklusive) ausgibt.

- Initialisierung:

  ```
  (define <name> (RandomMaker ug og))
  ```

- Methoden:

  ```
  (send <name> 'type)
  (send <name> 'next [this-time])
  ```

5.2. JitterPositionMaker

```
(JitterPositionMaker pos max-step)
```

Implementiert einen Zufallsgenerator, welcher Werte im Beriech $]-$ max-step $<$ pos $<$ $+$max-step$[$ ausgibt.

- Initialisierung:

  ```
  (define <name> (JitterPositionMaker pos max-step))
  ```

- Methoden:

  ```
  (send <name> 'type)
  (send <name> 'next [this-time])
  ```

5.3. RandomWalkMaker

```
(RandomWalkMaker init-pos max-step)
```

Implementiert einen Zufallsgenerator, welche Zufallswerte im Beriech $]-$ max-step $<$ pos $<$ $+$max-step$[$ ausgibt und für jeden nächsten Werte den vorigen als Grundwert verwendet.

- Initialisierung:

  ```
  (define <name> (RandomWalkMaker init-pos max-step))
  ```

- Methoden:

```
(send <name> 'type)
(send <name> 'next [this-time])
(send <name> 'reset!)
```

5.4. TendencyMaker

```
((TendencyMaker) LineProc '(args-ug) '(args-og))
```

Implementiert eine Tendenzmaske, welche von `TIME = 0` bis `MAX_TIME` geht. Als Grenzen für die Zufallswerte werden zwei Prozeduren `LineProc` verwendet (in der Regel `LineMaker` oder `ExptLineMaker`), welche mittels `'(args-ug)` und `'(args-og)` initialisiert werden.

- Initialisierung:

```
(define <name> ((TendencyMaker) <LineProc> '(args-ug)
   '(args-og)))
```

- Methoden:

```
(send <name> 'type)
(send <name> 'next this-time)
(send <name> 'reset!)
```

6. Repertoire

6.1. ShuffleListMaker

```
(ShuffleListMaker rep)
```

Implementiert einen Zufallsgenerator, welcher Werte aus einem gegebenen Repertoire (Liste) zieht.

- Initialisierung:

  ```
  (define <name> (ShuffleListMaker liste))
  ```

- Methoden:

  ```
  (send <name> 'type)
  (send <name> 'next [this-time])
  (send <name> 'last)
  ```

6.2. PatternMaker

```
(PatternMaker rep)
```

wiederholt sein Repertoire (Liste).

- Initialisierung:

  ```
  (define <name> (PatternMaker liste))
  ```

- Methoden:

  ```
  (send <name> 'type)
  (send <name> 'next [this-time])
  (send <name> 'last)
  ```

7. Dynamische Veränderungen

7.1. ChangeMaker

```
(ChangeMaker '(time-list) procs)
```

Ändert das Verhalten (die Definition) eine Parameters nach Ablauf der angegebenen Zeiten.

- Initialisierung:

  ```
  (define <name> (ChangeMaker '(t0 t1 ... tn) '(proc0 args0)
     '(proc1 args1) '(procn argsn)))
  ```

  `tn` muss dabei $\geq$ `MAX_TIME` sein; außerdem muss die Anzahl der in der Liste angegebenen Zeiten der Anzahl der angegebenen Prozeduren entsprechen.

- Methoden:

  ```
  (send <name> 'type)
  (send <name> 'next this-time)
  ```

7.2. ChangeTrackMaker

```
(ChangeTrackMaker procs)
```

Ändert das Verhalten (die Definition) eines Parameters für jeden vorgesehenen Track.

- Initialisierung:

  ```
  (define <name> (ChangeTrackMaker '(proc0 args0) '(proc1 args1)
     '(procn argsn)))
  ```

  `procn` muss dabei der in `TRACKS` vorgesehenen Anzahl der Tracks entsprechen.

- Methoden:

  ```
  (send <name> 'type)
  (send <name> 'next COUNT_TRACKS)
  ```

7.3. Interpolate

```
(Interpolate from-proc to-proc '(from-args) '(to-args))
```

Interpoliert in der Zeit von $[\texttt{TIME} = 0, \texttt{MAX\_TIME}]$ zwischen `from-proc` und `to-proc`.

- Initialisierung:

```
(define <name> (Interpolate <from-proc> <to-proc> '(arga0
   arga1 ... argan) '(argb0 argb1 ... argbn)))
```

- Methoden:

```
(send <name> 'type)
(send <name> 'next this-time)
```

8. Misc

8.1. AccumulatorMaker

```
(AccumulatorMaker init-value change-proc)
```

Akkumuliert seine Werte mithilfe der Prozedur `change-proc` (in der Regel Addition oder Multiplikation)

- Initialisierung:

```
(define <name> (AccumulatorMaker <init-value> <change-proc>)))
```

- Methoden:

```
(send <name> 'type)
(send <name> 'update!)
(send <name> 'show)
(send <name> 'reset!)
```

8.2. FindNextMaker

```
((FindNextMaker rep) elem)
```

Findet das nächste Element zu einem gegebenen Element in einer Liste (hilfreich teilweise, um Dauern zu berechnen, wenn eine Liste mit Startwerten gegeben ist).

- Initialisierung:

```
(define <name> (FindNextMaker <liste>))
```

- Methoden:

```
(send <name> 'type)
(seend (<name>  <element>) 'next)
```

8.3. ObjectMaker

```
(ObjectMaker proc . args)
```

deprecated

Teil III.

Der Scheduler

9. Prozeduren

9.1. ScoreLineMaker

```
(ScoreLineMaker procs)
```

9.2. write-score

```
(write-score)
```

9.3. run_csound

```
(run_csound aiff-file orc-file sco-file)
```

Teil IV.

Lizenzen

A. GNU GPL v3 (YaPPP)

TBC

B. GNU Free Documentation License

TBC