livecoding, musique, python, em

FoxDot

FoxDot est un langage de livecoding qui permet d'écrire de la musique, de la transformer et de la jouer en temps réel.
FoxDot est développé en python par Ryan Kirkbride

Installation

Voir

- https://github.com/Qirky/FoxDot
- https://foxdot.org/installation/

Installation de supercollider et foxdot sur debian stretch 9.5

```
# installer supercollider
sudo apt-get install supercollider

# installer les sc3-plugins en suivant https://github.com/supercollider/sc3-plugins

# démarrer supercollider : démarrer / son et vidéo / supercollider IDE
# dans supercollider
# Quarks.install("https://github.com/Qirky/FoxDotQuark.git")
# Quarks.install("https://github.com/supercollider-quarks/BatLib.git")

pip3 install --user FoxDot --upgrade
sudo apt-get install python3-tk
```

Démarrage

```
# démarrer supercollider : démarrer / son et vidéo / supercollider IDE
# dans supercollider
# écrire dans la partie gauche de la fenêtre : FoxDot.start , pui faire CTRL+ENTER sur la ligne pour l'exécuter
# puis dans un terminal
python3 -m FoxDot &
```

Trucs et astuces

aide-mémoire

Un bloc de code est composé de **lignes qui se suivent sans ligne vide**, il est exécuté d'un coup.
On exécute un bloc avec Ctrl + ↵ Enter après avoir placé le curseur sur la ligne
On peut aussi exécuter une seule ligne avec Alt + ↵ Enter
Un «player» est toujours composé de deux caractères, pour jouer à plusieurs on recommande d'utiliser une lettre (initiale du pseudo) + un chiffre, exemples : d1, f4, etc.
Par défaut, Foxdot fonctionne avec les paramètres suivants :

- tempo à 120 BPM,
- tonalité en Do majeur,
- signature de temps en 4/4,
- tous les effets sont désactivés,
- la note jouée est 0,
- l'octave est 5,
- la gamme est do majeur (C major) tonalité est celle définie par défaut, la durée de la note est 1,

le volume est à 100% (1), et c'est joué en boucle

Un instrument

```
p1 >> pads((0,2,4))         // Les parenthèses définissent un accord
```

```
p1 >> pads([0,1,2,(3,5,7)]) // Les crochets carrés définissent une séquence
p1 >> pads([0,1,2,3], dur=[1,1/2,1/2], amp=[1.5,0.5], sus=2, pan=[-1,1], oct=[5,5,5,(4,6)])
```

Des rythmes

Chaque instrument de percussion est défini par un caractère qui correspond à un répertoire de samples

```
d1 >> play("x-o-")
d1 >> play("x-o-", sample=[0,1,2])
```

On peut varier les rythmes avec :

```
d1 >> play("x-o[--]")           # des crochets carrés pour jouer plusieurs samples sur un même pas
d1 >> play("x-o(-o)")           # des parenthèses pour alterner entre plusieurs sons
d1 >> play("x-o{-o*}")          # des accolades pour piocher un sample au hasard dans le répertoire correspondant
d1 >> play("o   ", sample=[3])  # changer l'index du sample dans le répertoire
d1 >> play("x-o{-[--]o}")       # mélange possible!
d1 >> play("x-|o2|-")           # entre barres verticales : jouer le sample 2 du répertoire o
d1 >> play("<x-o-><  + + [ +]>") # deux couches de percussions, entre chevrons
d1 >> play("<x x       xx   x><    o       o   ><: : : : : : : : >", amp=(1, 0.3, 0.4)) # amplification différente pour chaque couche
d1 >> play("<  x x           ><[::][: ]: : : : : [::][: ]: ><        o       >", amp=(1, 0.3, 0.8))
```

Rate modifie la vitesse de lecture

```
d1 >> play("x-(-[-o])", dur=[3/4,3/4,1/4,1/2], rate=[1,2,0.5,-1])
```

```
Clock.bpm = 120

h1 >> play("x      xx x     ")  # l'espace correspond à un silence
h2 >> play("    o       o   ")
h3 >> play("- - - - - - - - ")

h1.stop(); h2.stop(); h3.stop();
```

Deux instruments, le second suit le premier

```
y1 >> bass([0,2,3,4], dur=1/2, slide=0)
g1 >> varsaw(amp=0.3, oct=5).follow(y1)

y1.stop() ; g1.stop()
```

Stopper tous les players

```
Clock.clear()
```

Une suite d'accords (avec des patterns)

```
ac >> space(P[(0,3,5), (1,4,6), (2,5,7), (4,7,9)], dur=4)
```

Réinitialiser un player

```
ex.reset()
```

Ajouter un attribut

```
g1 >> varsaw(amp=0.6, oct=5, tremolo=3) # tremolo est un attribut
```

Ajouter un effet

```
y1 >> bass([0,2,3,4], amp=1, dur=[4, 1/2, 1/2, 3], slide=0, chop=5) # chop est un effet
```

Afficher le BPM

```
print(Clock.bpm)
```

Changer la gamme

```
Scale.default.set("minor")
# Identique à :
Scale.default = "minor"
```

Instruments, attributs, effets, samples, gammes

```
print(SynthDefs)               # afficher tous les instruments
print(Player.get_attributes()) # afficher tous les attributs
print(FxList)                  # afficher tous les effets
```

```
print(Samples)                # afficher tous les échantillons
print(Scale.names())          # afficher les gammes
```

Ce qui donne avec la version 0.8.3 :

foxdot : tous les instruments

foxdot_0.8.3_tous_les_instruments.txt

```
['bass', 'orient', 'spark', 'keys', 'glass', 'arpy', 'dbass', 'zap', 'nylon', 'blip', 'saw', 'loop', 'play2', 'scratch', 'marimba',
'bell', 'soft', 'scatter', 'lazer', 'sitar', 'play1', 'fuzz', 'pasha', 'creep', 'swell', 'feel', 'quin', 'dirt', 'squish', 'bug',
'pads', 'sinepad', 'dab', 'jbass', 'donk', 'crunch', 'razz', 'prophet', 'charm', 'stretch', 'snick', 'space', 'gong', 'soprano',
'rave', 'ripple', 'sawbass', 'pulse', 'karp', 'star', 'noise', 'ambi', 'audioin', 'varsaw', 'klank', 'pluck', 'growl', 'dub',
'twang', 'viola']
```

foxdot : tous les attributs

foxdot_0.8.3_tous_les_attributs.txt

```
('degree', 'oct', 'freq', 'dur', 'delay', 'buf', 'blur', 'amplify', 'scale', 'bpm', 'sample', 'env', 'sus', 'fmod', 'pan', 'rate',
'amp', 'midinote', 'channel', 'vibdepth', 'vib', 'slide', 'slidedelay', 'sus', 'slidefrom', 'glidedelay', 'glide', 'bend',
'benddelay', 'coarse', 'rate', 'striate', 'buf', 'pshift', 'hpf', 'hpr', 'lpr', 'lpf', 'swell', 'bpnoise', 'bpr', 'bpf', 'chop',
'tremolo', 'beat_dur', 'echotime', 'echo', 'spin', 'cut', 'mix', 'room', 'formant', 'shape', 'drive')
```

foxdot : tous les effets

foxdot_0.8.3_tous_les_effets.txt

```
<'bandPassFilter': keyword='bpf', other args=['bpnoise', 'bpr', 'sus']>
<'chop': keyword='chop', other args=['sus']>
<'coarse': keyword='coarse', other args=['sus']>
<'combDelay': keyword='echo', other args=['echotime', 'beat_dur']>
<'filterSwell': keyword='swell', other args=['sus', 'hpr']>
<'formantFilter': keyword='formant'>
<'glissando': keyword='glide', other args=['glidedelay', 'sus']>
<'highPassFilter': keyword='hpf', other args=['hpr']>
<'lowPassFilter': keyword='lpf', other args=['lpr']>
<'overdriveDistortion': keyword='drive'>
<'pitchBend': keyword='bend', other args=['benddelay', 'sus']>
<'pitchShift': keyword='pshift'>
<'reverb': keyword='room', other args=['mix']>
<'slideFrom': keyword='slidefrom', other args=['slidedelay', 'sus']>
<'slideTo': keyword='slide', other args=['slidedelay', 'sus']>
<'spinPan': keyword='spin', other args=['sus']>
<'striate': keyword='striate', other args=['rate', 'buf', 'sus']>
<'tremolo': keyword='tremolo', other args=['beat_dur']>
<'trimLength': keyword='cut', other args=['sus']>
<'vibrato': keyword='vib', other args=['vibdepth']>
<'wavesShapeDistortion': keyword='shape'>
```

foxdot : tous les samples

foxdot_0.8.3_tous_les_samples.txt

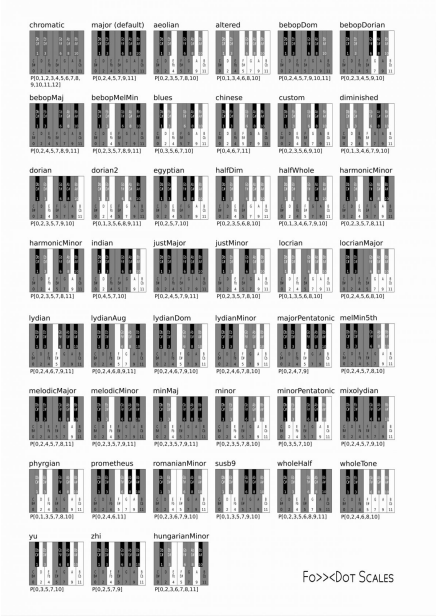
```
'!': Yeah!
'#': Crash
'$': Beatbox
'%': Noise bursts
'&': Chime
'*': Clap
'+': Clicks
'-': Hi hat closed
'/': Reverse sounds
'1': Vocals (One)
'2': Vocals (Two)
'3': Vocals (Three)
'4': Vocals (Four)
':': Hi-hats
'=': Hi hat open
'@': Gameboy noise
'A': Gameboy kick drum
'B': Short saw
'C': Choral
'D': Dirty snare
'E': Ringing percussion
'F': Trumpet stabs
'G': Ambient stabs
```

```
'H': Clap
'I': Rock snare
'J': Ambient stabs
'K': Percussive hits
'L': Noisy percussive hits
'M': Acoustic toms
'N': Gameboy SFX
'O': Heavy snare
'P': Tabla long
'Q': Electronic stabs
'R': Metallic
'S': Tamborine
'T': Cowbell
'U': Misc. Fx
'V': Hard kick
'W': Distorted
'X': Heavy kick
'Y': High buzz
'Z': Loud stabs
'\\': Lazer
'^': 'Donk'
'a': Gameboy hihat
'b': Noisy beep
'c': Voice/string
'd': Woodblock
'e': Electronic Cowbell
'f': Pops
'g': Ominous
'h': Finger snaps
'i': Jungle snare
'j': Whines
'k': Wood shaker
'l': Robot noise
'm': 808 toms
'n': Noise
'o': Snare drum
'p': Tabla
'q': Ambient stabs
'r': Metal
's': Shaker
't': Rimshot
'u': Soft snare
'v': Soft kick
'w': Dub hits
'x': Bass drum
'y': Percussive hits
'z': Scratch
'|': Hangdrum
'~': Ride cymbal
```

foxdot : toutes les gammes

foxdot_0.8.3_toutes_les_gammes.txt

```
['aeolian', 'altered', 'bebopDom', 'bebopDorian', 'bebopMaj', 'bebopMelMin', 'blues', 'chinese', 'chromatic', 'custom', 'default',
'diminished', 'dorian', 'dorian2', 'egyptian', 'freq', 'halfDim', 'halfWhole', 'harmonicMajor', 'harmonicMinor', 'hungarianMinor',
'indian', 'justMajor', 'justMinor', 'locrian', 'locrianMajor', 'lydian', 'lydianAug', 'lydianDom', 'lydianMinor', 'major',
'majorPentatonic', 'melMin5th', 'melodicMajor', 'melodicMinor', 'minMaj', 'minor', 'minorPentatonic', 'mixolydian', 'phrygian',
'prometheus', 'romanianMinor', 'susb9', 'wholeHalf', 'wholeTone', 'yu', 'zhi']
```



(Image d'après Foxdot,coding music part 1, Jens Meisner)

Et une petite cheatsheet à télécharger :

foxdot cheatsheet

Divers

Comment sont programmés les players ?

Chaque player est un SynthDef écrit en supercollider, il s'agit donc de synthèse sonore, excepté pour les sons de percussions qui sont des samples. On peut retrouver les définitions des SynthDefs dans le répertoire `/home/user/.local/lib/python3.5/site-packages/FoxDot/osc/scsyndef`
Un exemple avec **glass** :

```
SynthDef.new(\glass,
{|freq=0, peak=1, rate=0, level=0.8, blur=1, vib=0, sus=1, pan=0, fmod=0, atk=0.01, rel=0.01, decay=0.01, bus=0, beat_dur=1, amp=1|
var osc, env;
sus = sus * blur;
freq = In.kr(bus, 1);
freq = [freq, freq+fmod];
sus=(sus * 1.5);
amp=(amp * 1.5);
freq=(freq * [1, (1 + (0.005 * rate))]);
osc=Klank.ar(`[[2, 4, 9, 16], [1, 1, 1, 1], [2, 2, 2, 2]], (PinkNoise.ar(0.0005).dup * SinOsc.ar((freq / 4), mul: 0.5, add: 1)), freq);
env=EnvGen.ar(Env(levels: [0, amp, 0],times: (sus * 2),curve: 'lin'), doneAction: 0);
osc=(osc * env);
osc = Mix(osc) * 0.5;
osc = Pan2.ar(osc, pan);
    ReplaceOut.ar(bus, osc)}).add;
```

Ajouter des répertoires de samples

on peut modifier le fichier ci-dessous pour indiquer de nouveaux répertoires qui pourront être utilisés avec play (plutot que loop)

```
/home/emoc/.local/lib/python3.5/site-packages/FoxDot/lib/Buffers.py
```

Les échantillons sont rangés dans

```
/home/emoc/.local/lib/python3.5/site-packages/FoxDot/snd
```

Enregistrer ?

On peut enregistrer depuis la fenêtre de SuperCollider

```
Server.default.record
```

Pour arrêter

```
Server.default.stopRecording
```

L'enregistrement se fait en AIFF et le chemin vers le fichier est indiqué dans la fenêtre de supercollider, le fichier est automatiquement horodaté.

Ressources

FAQ : https://forum.toplap.org/t/frequently-asked-questions/504

Forum anglophone : https://forum.toplap.org/c/communities/foxdot/17

Tutoriel, en français de Marvin Wortman : http://marvinwortman.me/idlabs_foxdot

Tutoriels pour workshops : https://github.com/Qirky/FoxDot-Worksheet

Quelques exemples :

- exemples en moins de 10 lignes : https://github.com/Qirky/ten-lines-or-less
- snippets : https://forum.toplap.org/t/share-your-snippets/711

Article de l'auteur sur Foxdot : http://users.sussex.ac.uk/~thm21/ICLI_proceedings/2016/Colloquium/68_FoxDot.pdf

Autre tutoriel complet (de Jens Meisner) : https://gitlab.com/iShapeNoise/foxdot_codingmusic_part1

D'autres ressources sur : https://fablabo.net/wiki/Livecoding et https://ressources.labomedia.org/live_coding

à propos de supercollider : https://github.com/madskjeldgaard/awesome-supercollider

Troop

Troop est un serveur qui permet de jouer à plusieurs, collaborativement. Il peut s'utiliser avec FoxDot ou d'autres langages de livecoding.

Installation de Troop : https://github.com/Qirky/Troop/

Il est recommandé d'utiliser la même version de Troop sur chaque machine qui se connecte au serveur et que le serveur et les clients soient de la même version.

Voir configuration d'un serveur troop sur debian 10