SONIC-PI : SEQUENCE PEDAGOGIQUE

Université du Luxembourg – Projet "Mutechlab" – Matteo Olivo

1. Live Coding : définition

La pratique du **live coding** a été définie de plusieurs manières, parmi lesquelles :

- coder des logiciels (algorithmes) en temps réel
- modifier des logiciels en temps réel
- projeter à l'écran l'interface de l'ordinateur du musicien, afin favoriser la participation du public
- utiliser l'écriture de code comme pratique d'improvisation
- composer de la musique en utilisant un système de notation textuelle

Cependant, selon Blackwell et al.[1], le live coding est *« une pratique performative basée sur l'aventure et l'exploration, **rejetant délibérément les définitions formelles** et conservant sa nature hétérogène, remettant constamment en question sa propre compréhension à travers la pratique de l'écriture et de la réécriture, de la définition et de la redéfinition, comme une performance publique »*.

Au niveau général, le live coding consiste à **générer en temps réel** du matériel audio-visuel, en **écrivant du code** sur un ordinateur. L'artiste doit rendre visible au public tout le processus de codage.

[1] Blackwell A., Cocker E., Cox G. et al., *Live Coding: A User's Manual*, Cambridge, The MIT Press, 2022.

2. Live Coding : histoire

- La pratique du live coding se développe principalement en Angleterre à partir des années '1990, initialement au sein des universités (Plymouth University).

- En septembre 2000, une première version du **manifeste programmatique** du live coding (« the Generative Manifesto ») a été présentée à l'Institut d'art contemporain de Londres.

- Au début des années '2000, des performances de live coding commencent à être jouées dans des pubs à Londres, ainsi qu'à Berlin, à New York et dans d'autres villes. La pratique du live coding commence également à être enseignée dans certaines universités (par exemple, à la Middlesex University). Pendant la même période, le terme « live coding » apparaît (à l'époque, l'expression « live programming » était aussi utilisée).

- En 2001, la librairie « Just In Time » pour le langage de programmation **SuperCollider** a été développée.

- En 2004, dans le cadre du symposium « Changing Grammar », organisé à la Hamburg Art Academy (HfbK) autour de la thématique « Code-as-art », la communauté **TOPLAP** (Terrestrial Organisation for the Permanence of Live Art Programming) a été fondée. Dans la même année, le **TOPLAP Manifesto** (version « draft ») a été publié.

- Aujourd'hui, le thème du live coding est abordé dans plusieurs conférences scientifiques, parmi lesquelles l'International Conference on Live Coding (ICLC).

3. Sonic-pi

Sonic-pi est un environnement de live coding créé en 2013 pour des buts pédagogiques. Il a été conçu par **Sam Aaron**, chercheur à l'université de Cambridge (Laboratoire d'informatique), pour fonctionner sur des ordinateurs Raspberry Pi (contrôleurs de petite taille et peu couteux, donc accessibles pour les étudiants). L'application Sonic-pi est également disponible pour les ordinateurs Windows et Mac OSX, et peut être téléchargée gratuitement. Sonic-pi utilise comme base le langage **Ruby**, mais la syntaxe a été simplifiée pour rendre la programmation plus accessible.

Sonic Pi a été développé en étroite collaboration avec les **enseignants**. Le processus de design de cet environnement numérique a vu la réalisation de nombreuses itérations, et de centaines d'heures d'**observation des élèves** en classe. Le but des concepteurs de Sonic-pi était de permette aux enseignants de transmettre, à travers l'utilisation de ce langage, les notions principales du curriculum ministériel anglais concernant l'enseignement de l'informatique[2].

Demo: https://www.youtube.com/watch?v=suH_goWVBeA

4. L'utilisation d'échantillons enregistrés

Sonic-pi met à disposition plusieurs types **d'échantillons enregistrés**. Pour lire un échantillon, il suffit d'écrire la ligne de code suivante :

```
sample :ambi_piano
```

Faites toujours bien attention aux **espaces entre les mots**.

[2] Bănuț M., "Using the Sonic Pi Application for Educational Purposes - A Literature Review", *Educatia 21 Journal*, n° 26(12), 2023.

Cliquez sur le bouton 

pour déclencher le son.

Plusieurs types d'échantillons sont disponibles. Pour les sélectionner, une fois le code « **sample :** » tapé, vous pouvez ajouter les préfixes écrits ci-dessous, et un menu contenant la liste d'échantillons disponibles pour chaque section va s'afficher :

```
sample :ambi_
sample :bass_
sample :elec_
sample :perc_
sample :guit_
sample :drum_
sample :misc_
sample :bd_
```

Pour **lire en boucle** un fichier enregistré, il suffit d'écrire le code suivant :

```
live_loop :sample do
  sample :loop_amen, amp: 1.5
  sleep 1.8
end
```

Il est possible de régler les paramètres des échantillons suivants :
- **amp** : amplitude de l'échantillon ➔ 0 (silence) / 1 (niveau d'amplitude moyen). Vous pouvez monter l'amplitude à 2 et plus.
- **pan** : stéréophonie du son ➔ -1/+1
- **rate** : vitesse de lecture ➔ valeurs positives (augmentation de la vitesse) ou négatives (lecture en reverse).

La lecture de chaque échantillon doit prévoir à chaque fois une pause (**sleep**).

N.B. Pour que l'échantillon soit lu correctement, la durée de la pause (sleep) **ne doit pas être inférieure à celle de l'échantillon.** Pour éviter ce problème, vous pouvez déclarer que la longueur de la pause est égale à la durée de l'échantillon :

```
live_loop :sample do
  sample :loop_amen
  sleep sample_duration :loop_amen
end
```

Pour créer **un loop** de batterie, en plus d'utiliser des boucles pre-enregistrées, il est aussi possible d'utiliser des **échantillons** de sons percussifs. Voici un exemple :

```
live_loop :rock do
  sample :drum_heavy_kick
  sleep 0.5
  sample :drum_snare_hard, amp: 0.5
  sleep 0.5
  sample :drum_heavy_kick
  sleep 0.25
  sample :drum_heavy_kick
  sleep 0.25
  sample :drum_snare_hard, amp: 0.5
  sleep 0.5
end
```

Le **tempo en Bpm de la session** peut être déclaré en amont, en écrivant le code suivant :

```
use_bpm 120
live_loop :tick do
  sleep 1
end
```

Ensuite, il faut **synchroniser la boucle de batterie** avec le tempo de la session :

```
live_loop :hiphop, sync: :tick do
  sample :drum_heavy_kick
  sleep 0.5
  sample :drum_snare_hard, amp: 0.5
  sleep 0.5
  sample :drum_heavy_kick
  sleep 0.25
  sample :drum_heavy_kick
  sleep 0.25
  sample :drum_snare_hard, amp: 0.5
  sleep 0.5
end
```

5. Notes, accords, synthétiseurs, effets

Pour **jouer une note** il suffit d'écrire la ligne de code suivante (faire attention aux **espaces** entre les mots) :

```
play :c4
```

N.B. Sonic-pi utilise le système de **notation anglo-saxon**.

Notation française	Do	Ré	Mi	Fa	Sol	La	Si	Do
Notation anglophone	C	D	E	F	G	A	B	C

Pour **jouer un accord** (un accord de Do Majeur), il suffit d'ajouter les deux lignes de code en jaune :

```
play :c4
play :e4
play :g4
```

Il est également possible d'utiliser la **notation MIDI** pour jouer des notes. Par exemple, pour jouer un DO4, nous pouvons écrire :

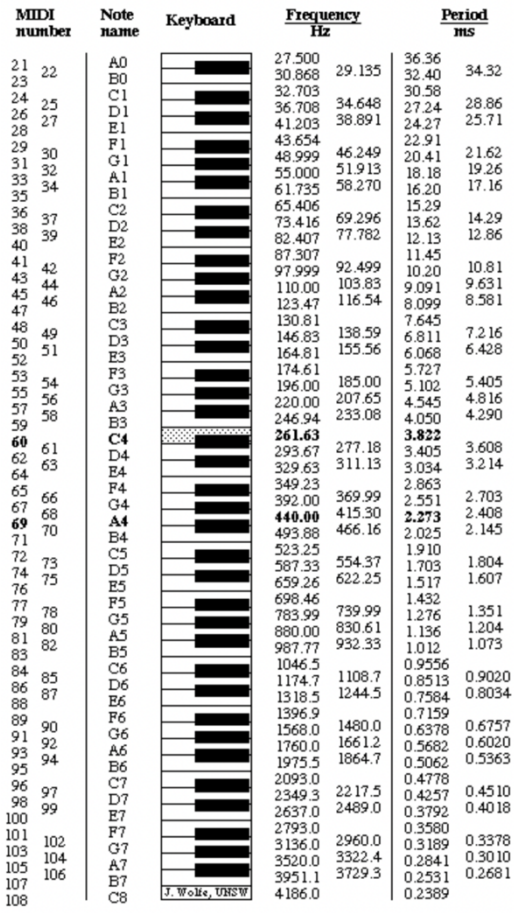
```
play 60
```

Et pour **jouer un accord** de Do Majeur en utilisant la notation MIDI, nous pouvons écrire le code suivant :

```
play 60
play 64
play 67
```

La notation Midi permet d'écrire des **altérations**. Par exemple, un Do dièse correspond à

```
play 61
```



Si nous ajoutons des pauses entre les notes, nous jouons une **mélodie** :

```
play :c4
sleep 0.5
play :e4
sleep 0.5
play :g4
sleep 0.5
```

N.B. La pause de 0.5 correspond à la **moitié d'un temps** en musique.

Pour **créer une boucle**, comme pour les fichiers enregistrés, il faut ajouter au début du bloc le code **live_loop** :

```
live_loop :loop do
    play :c4
    sleep 0.5
    play :e4
    sleep 0.5
    play :g4
    sleep 0.5
end
```

Le code **n.times** permet de créer des **itérations** : une itération permet de répéter N fois un bloc de code. L'itération peut être créée à l'intérieur d'une boucle, comme dans l'exemple ci-dessous :

```
live_loop :jacques do
2.times do
    play :c4
    sleep 0.5
    play :d4
    sleep 0.5
    play :e4
    sleep 0.5
    play :c4
    sleep 0.5
end
```

```
2.times do
     play :e4
     sleep 0.5
     play :f4
     sleep 0.5
     play :g4
     sleep 1
end
end
```

Enfin, pour modifier le type de **synthétiseur** utilisé, il faut écrire le code **use_synth :** au début du bloc, comme dans l'exemple ci-dessous :

```
use_synth :saw
live_loop :jacques do
     ……
     ……
end
```

Lorsque vous avez écrit la commande **use_synth :** il suffit de taper la première lettre (ex. b, c, d, etc.) et la **liste des synthétiseurs** mis à disposition par Sonic-pi va s'afficher.

Comme pour les fichiers enregistrés, il est possible de régler les paramètres **amp** (amplitude du signal), **pan** (stéréophonie) et d'autres.

```
use_synth :saw
play 60, amp: 0.5, pan: -1
```

Il est aussi possible de traiter le signal avec des **effets**. Pour ce faire, il faut écrire le code « **with_fx :nom_de_l'effet do** » et ensuite « **end** » à la fin. Par exemple, la deuxième ligne de code ci-dessous ajoute un **echo :**

```
use_synth :saw
with-fx :echo do
play 60, amp: 0.5
end
```

Pour chaque effet, certains **paramètres** peuvent être contrôlés. Par exemple, dans le code ci-dessous, nous avons saisi la valeur 1 pour le paramètre « **room** » de la réverbération, qui détermine la taille de la pièce :

```
use_synth :saw
with_fx :reverb, room: 1 do
  play 60, amp: 0.5
end
```

Dans la section « **effets spéciaux** » du tutoriel de Sonic-pi (bouton « **Help** » en haut à droite de l'interface), il est possible de consulter la liste des effets utilisables, ainsi que des différents paramètres modifiables.



Pour ajouter plusieurs effets, le code « **with_fx :** » doit être répété, ainsi que la commande **end** à la fin du bloc :

```
use_synth :saw
with_fx :reverb, room: 1 do
  with_fx :echo do
    play 60, amp: 0.5
  end
end
```

6. Enchaîner des sons : la structure du liveset

Tous les instruments et les blocs de code écrits sur l'éditeur sont exécutés en même temps. Pour arrêter la lecture d'un bloc de code, deux manières sont possibles :

1) utiliser le code « **stop** », comme dans l'exemple ci-dessous :

```
live_loop :melodie do
  stop
  ……….
  ……….
end
```

2) Utiliser le symbole # devant chaque ligne de code qui permet de déclencher un son, comme dans l'exemple ci-dessous :

```
live_loop :melodie do
  #play :c4
  sleep 1
  #play :e4
  sleep 1
  #play :g4
  sleep 1
end
```

En éliminant le code **stop** ou le symbole #, vous réactivez le bloc de code en pause.

7. Capter le signal d'un microphone

Pour capter le signal d'un microphone branché à votre ordinateur (à travers une carte son), il faut écrire le code « **live_audio :** », comme dans l'exemple ci-dessous :

```
live_audio :piezo1, input: 1
end
```

« **input: 1** » représente la première entrée de la carte son
Pour que Sonic Pi puisse recevoir le son du microphone branché à la carte son, il faut vérifier que la case « **Activer les entrées audio** » soit cochée dans la section « **Audio** » des préférences, et que le nom de la carte son soit bien affiché dans la section « **Entrée/Sortie** ».





Contraintes : live_audio doit avoir un nom unique (dans le cas ci-dessus « **piezo1** »). Un seul flux « live_audio » avec ce nom peut exister.

Le signal du micro peut etre traité avec des **effets**, en utilisant le même code utilisé pour les synthétiseurs.

```
with_fx :echo, decay: 7 do
with_fx :level, amp: 1.9 do
  live_audio :piezo1, input: 1
end
end
```

8. Performance de Live coding

3 instruments

- 1 instrument à vent
- 1 ordinateur (Sonic-pi)
- 1 micro piézo-électrique

Effets conseillés

- instrument à vent : echo / ping_pong – reverb
- micro piézo-électrique : distorsion – ring_mod – ping_pong - flanger