Departamento de Música

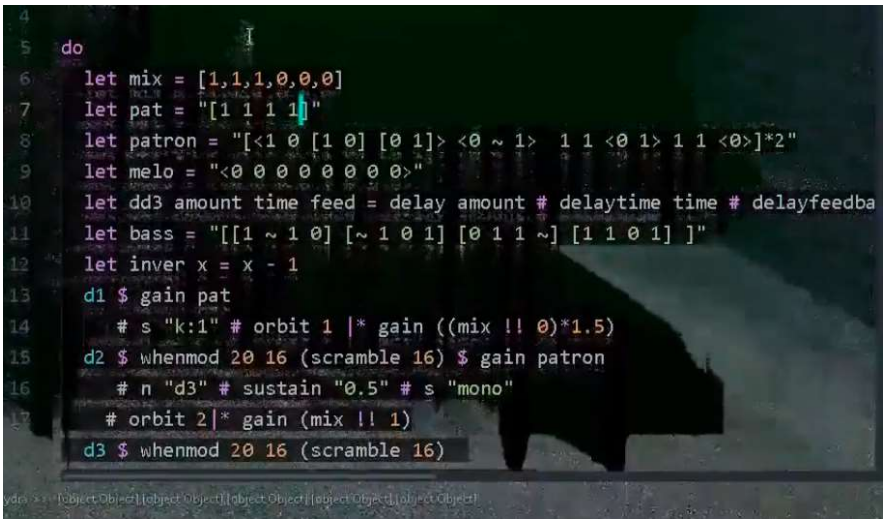
FACULTAD DE ARTES

UNIVERSIDAD NACIONAL DE LA PLATA

Trabajo de Graduación

Performances audio-visuales en entornos de programación

El código fuente como instrumento musical.



Tesis de Licenciatura en Música, orientación Música Popular

Prof. Pedro Iñaki Sagasti

DNI: 35034431

Tel: 221-4551755

E-mail: torotumbo.musica@gmail.com

Legajo: 57334/6

Director: Prof. Federico Jaureguiberry

Titular Producción y Análisis V: Julio Schinca

Resumen

El presente trabajo consta de la performance de live coding 'Nuevas Canciones de Bandidxs' y del análisis de las prácticas artísticas englobadas bajo este término, enfocando en la relación intrínseca del live coding con la composición algorítmica en tiempo real. Para esto se aborda la comprensión de la lógica algorítmica como objeto digital artístico basado en los conceptos que presenta Carlos Gutiérrez López en su trabajo. A su vez se estudia al live coding musical enmarcado en una algorave como ámbito de difusión de estas obras. Se presenta el análisis de cinco elementos sintácticos de Tidal Cycles presentes en la performance antedicha.

Palabras Clave: Live Coding, Algorave, Programación en tiempo real, Tidal Cycles.

Indice

[1] El término 'on-the-fly programming' se suele usar como alternativa al de live coding. (Ge Wang & Perry R. Cook, 2004).

1. Fundamentación

Este trabajo surge desde una búsqueda personal para llevar adelante obras que incluyan, en el medio técnico y en el discurso poético, herramientas relacionadas con las tecnologías digitales. Dentro de esa búsqueda, el live coding se presenta como un espacio de creación donde las posibilidades compositivas y discursivas son maleables a los intereses particulares de quienes practican esta disciplina. En este trabajo las herramientas utilizadas se basan en la filosofía del Software Libre[2], las cuales otorgan mayor autonomía a quienes crean con ellas al abrir el proceso constitutivo de la herramienta en sí. Además, entender al código fuente como un instrumento permite ampliar la paleta de recursos a los estudiantes de las carreras de grado. El software elegido para trabajar en este proyecto (Tidal Cycles) posee tutoriales y guías de uso mayoritariamente en inglés, por lo que este trabajo aporta a la comunidad hispano parlante que lo utiliza.

2. Propósitos

- Presentar la práctica del live coding a la comunidad universitaria de la Facultad de Artes (FDA-UNLP)

- Introducir la lógica algorítmica y su relación con el objeto digital artístico propuesto por Carlos Gutiérrez López.

- Abordar a la escritura de código como un instrumento musical, con sus aportes estético-compositivos propios.

- Dar a conocer el software Tidal Cycles y sus usos para la composición algorítmica en tiempo real.

3. Objetivos

- Comprender la dinámica discursiva del live coding musical.

[2]La filosofía del Software Libre se puede resumir en 4 puntos:
- La libertad de ejecutar el programa como se desee, con cualquier propósito (libertad 0).
- La libertad de estudiar cómo funciona el programa, y cambiarlo para que haga lo que usted quiera (libertad 1).
- La libertad de redistribuir copias para ayudar a otros (libertad 2).
- La libertad de distribuir copias de sus versiones modificadas a terceros (libertad 3).

- Contextualizar la práctica del live coding musical en el ámbito de algorave.
- Desarrollar un acercamiento teórico-práctico a la composición algorítmica en entornos de live coding.
- Llevar adelante una performance de live coding musical en clave algorave.

4. Marco Teórico

4.1 Algoritmos on-the-fly[3] y Live Coding

Se define a un algoritmo como "… cualquier procedimiento computacional bien definido que toma algún valor, o conjunto de valores, como entrada y produce algún valor, o conjunto de valores, como salida" (Cormen, Leiserson, Rivest & Stein 2001). Así, el live coding es la práctica artística de generar obras a través de dar instrucciones a computadoras, en forma de algoritmos. Escribir instrucciones algorítmicas en forma de código fuente para reescribirlas mientras la performance sucede y la instrucción es llevada adelante. Es por esta razón que se denomina live coding (codificación en vivo) u on-the-fly programming (programación al vuelo) a esta disciplina artística.

4.2 Rasgos característicos del Live Coding

El live coding tiene 3 rasgos característicos. El primero de ellos ya ha sido mencionado, es la posibilidad de modificar los algoritmos a medida que se ejecutan. El segundo es la visualización del código que genera la obra a medida que la performance se desarrolla, como si pudiéramos ver la partitura de una improvisación a medida que sucede. Esto es tan intrínseco a las presentaciones que el manifiesto de la comunidad internacional de live coding (TOPLAP) demanda:

"...Dennos acceso a la mente del performer, a todo el instrumento humano.

El oscurantismo es peligroso. Muéstrennos su pantalla…"[4]

[3] El término 'on-the-fly programming' se suele usar como alternativa al de live coding. (Ge Wang & Perry R. Cook, 2004).

[4] TOPLAP MANIFESTO. https://toplap.org/wiki/ManifestoDraft (Traducción por el autor)

El tercer rasgo del live coding, es que al abarcar diversas disciplinas artísticas, permite que las performances incluyan varias capas de código pertenecientes a las disciplinas involucradas. En síntesis, el live coding nos ofrece la posibilidad de generar performances audio-visuales en entornos de programación y, a su vez, es un campo fecundo para desarrollar el trabajo interdisciplinario a partir de la mixtura de lenguajes y técnicas.

4.3 El Objeto Digital, el Objeto Técnico y el Objeto Digital Artístico

El live coding se inserta en la tradición artística del arte digital que se remonta a los años 60, principalmente a las animaciones asistidas por computadoras y la figura de Charles Csuri[5]. El medio en el que se producen estas obras "(…) no es meramente un nuevo soporte, sino que incluye una nueva forma de pensamiento (…)" (Gutiérrez López, 2018). En referencia a esta forma de pensamiento, el autor retoma el concepto de *Objeto Digital* acuñado por Yuk Hui[6], entendiéndolo no solo como el resultado final del trabajo en una computadora, sino "(…) también aquellos procesos y mecanismos que dan vida a un programa(...)," o en nuestro caso, una obra.

Para interpretar la lógica de los objetos digitales es importante comprender la teoría de los *Objetos Técnicos* del filósofo Gilbert Simondon[7]. Para este autor, un objeto técnico es aquel que permite a la persona que lo utiliza comprender su funcionamiento interno, modificarlo, mejorarlo y/o adaptarlo. Propone a su vez que todo objeto técnico necesita de una red de intercambio de conocimiento interpersonal donde los logros y métodos de trabajo son puestos a disposición para que otras personas los comprendan y/o repliquen. Con relación a esto, Gutiérrez López (2018) propone que:

[5] https://proyectoidis.org/charles-csuri/
[6] Ver ¿Que es un objeto digital? Yuk Hui
https://www.revistavirtualis.mx/index.php/virtualis/article/view/221
[7] Ver El modo de existencia de los objetos técnicos, Gilbert Simondón. Ed Prometeo.

> (…) si se considera al objeto digital una especie de continuación del objeto técnico y al objeto estético un a posibilidad de construir el objeto artístico, se infiere que hay una posibilidad de decantar un objeto digital como objeto del arte. (p. 6)

Sobre la base de lo antedicho entenderemos como objeto digital artístico a aquellas obras, lenguajes de programación o dispositivos que permiten comprender su funcionamiento, modificarlo, replicarlo y se encuentren inscriptos en una red de intercambio donde estos conocimientos sean compartidos.

4.4 Live Coding Musical

Hernani Villaseñor considera al live coding musical como "...la práctica de escribir código fuente con la intención de crear música en el momento." (Villaseñor, 2014, p. 7). Para esto se utilizan distintos lenguajes de programación[8] que nos ofrecen una lógica de funcionamiento particular, ya sea por el modo en que se escriben los comandos (sintaxis) o por los procesos que puedan aplicarse a los materiales sonoros[9]. Quién "livecodea[10]" opera estos comandos en vivo, haciendo que la creación y modificación de los mismos sea parte troncal de la performance: con una secuencia de comandos determinada se crea un instrumento. Modificando la secuencia, se modifica el instrumento.

4.5 Algoraves

Dentro de la escena del live coding existe una vertiente que realiza performances en fiestas electrónicas llamadas algoraves (del inglés algorythmic rave). Estos eventos se caracterizan por la realización de live coding musical pensado para ser bailable. Los géneros elegidos dependen del livecoder, aunque suele haber preponderancia de música electrónica de baile (EDM por sus siglas en inglés), reggaeton y cumbia. Es usual encontrar proyecciones del código que genera la

[8] https://github.com/toplap/awesome-livecoding/blob/master/README.md Lista de lenguajes y herramientas para realizar live coding

[9] Esto se puede realizar mediante síntesis sonora, utilización de audios pre-grabados (o grabados en vivo) y/ o control de dispositivos externos vía protocolos MIDI u OSC.

[10] Se suele utilizar este término en el ambiente para hablar de la persona que hace live coding.

música, animaciones generadas mediante código y prácticas experimentales cercanas al live coding.

5. Análisis de Caso: Tidal Cycles

Para que el programa funcione de manera correcta, tiene que estar escrito de una manera determinada, cada lenguaje de programación tiene su propia sintaxis. Tidal Cycles funciona como un secuenciador que organiza patrones de eventos (listas) en SuperCollider a través de SuperDirt. Estas listas se ven afectadas por distintas funciones que trae el software o que pueden ser modificadas por el usuario. El funcionamiento de estos programas escapa a los límites de este trabajo sin embargo se presenta el análisis de la sintaxis de 5 elementos presentes en la performance 'Nuevas Canciones de Bandidxs':

1. Lista de eventos sonoros y Organización temporal

Se entiende por lista a una secuencia de eventos con datos asociados. En el software analizado nos enfocaremos en listas de eventos sonoros donde los datos asociados serán el tipo de archivo que se utiliza y su organización temporal. Esto se realiza de la siguiente manera:

```
d1 $ s "bd arpy@3 fander ! bd*2 print/4 <~ sn:3> [~ jvbass]"
```

FIGURA 1. Ejemplo de lista.

El comando d1 funciona como canal de audio, pudiendo usar hasta d8 (o más dependiendo el set-up).
El signo $ permite contener la lista de eventos que pondremos luego. La letra s, que es por sound, sirve para decirle a Tidal Cycles que la lista que sigue, son archivos de audio, sintetizadores o silencios.

- Archivo de audio: Nombre de la carpeta donde está el archivo "bd"; si la misma tiene varios archivos se usa índice cero siendo "bd:1" el segundo

archivo de la carpeta bd, “bd:2” el tercero y así. Estás archivos están en formato .wav y se alojan en un directorio específico[11].

- Sintetizadores: el motor de audio SuperDirt con el que funciona Tidal Cycles viene con un conjunto de sintetizadores por defecto[12] a los que le podemos sumar aquellos que escribamos o consigamos por internet. Los mismos se agregan en la configuración de Tidal Cycles y se denominan SynthDef.

- Virgulillas “~” que equivale a un silencio.

Es importante señalar que este software funciona de manera cíclica (por eso Tidal Cycles, ciclos de mareas en inglés). La organización del tiempo en nuestra secuencia dependerá de cómo los samples, synthDefs y silencios se relacionen a partir de los siguientes elementos:

- Corchetes “[]”: Funcionan como listas dentro de la lista, resultando como la división en la escritura tradicional, por lo que sí tenemos:

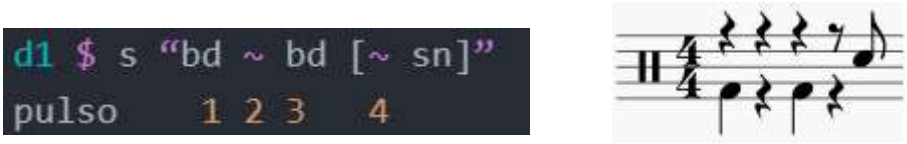


FIGURA 2. Ejemplo de organización temporal

- Diples “<>”: Funciona como una lista que es interpretada una vez por ciclo, por lo que lo escrito en la figura 3 va a hacer que el golpe en el contratiempo suene un ciclo si y el siguiente no.

FIGURA 3. Alternancia de eventos: diples.

- Signo de admiración “!”: Permite repetir el evento anterior.

[11] Para ubicar nuestra carpeta de samples tenemos que ir, en SuperCollider, a File/Open user support directory/downloaded-quarks/Dirt-Samples

[12] https://tidalcycles.org/index.php/All_effects_and_synths#SuperDirt_synths

```
d1 $ s "bd ! ! !"
```

FIGURA 4. Repetición de eventos

- Arroba “@”: Permite extender la duración del evento dependiendo el número que pongamos luego de él, haciendo que en la figura 5 suene el patrón de la figura 1 sin el primer silencio.

```
d1 $ s "bd@2 bd [~ sn]”
```

FIGURA 5. Modificación de la duración del evento.

- Signo de multiplicar “*”: Genera una figura regular o irregular dependiendo del contexto. Por lo que la figura 6 Va a generar un tresillo de corcheas en el tiempo 3.

```
d1 $ s "bd ~ bd*3 [~ sn]"
```

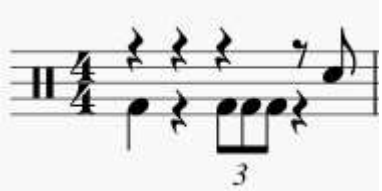


FIGURA 6. Repetición de eventos.

- Barra “/”: hace que el evento suene cada tantos ciclos como se escriban. Por lo que en lo escrito en la figura 7 va a hacer que el segundo evento suene cada dos ciclos.:

```
d1 $ s "bd  sn/2 bd [~ sn]"
```

FIGURA 7. Distinta ejecución dependiendo el ciclo.

2. Procesamiento de sonido

```
d1 $ s "bd ~ bd [~ sn]" # lpf "2000" # lpq "0.2"
```

FIGURA 8. Procesamiento de sonido

El símbolo # permite que el programa asigne un proceso determinado a la lista que lo precede, funcionando de manera similar que el $ pero aquí la nueva lista (# lpf “2000”) es aplicada a la lista pre-existente ($ s “bd ~ bd [~ sn]”). Los procesos de sonidos que trae el software son filtros, reverberación, delay, dentro de una larga lista de opciones[13]. Cada proceso tiene ciertos parámetros de control. En el ejemplo, la lista de eventos sonoros se ve afectada por el filtro pasa bajos (lpf) que tiene el punto de corte en la frecuencia 2000 Hz, y la resonancia (lpq) que está al 20%.

3. Creación de nuevas funciones

```
let filt f r = lpf f # lpq r

d1 $ s "arpy*8" # filt 2000 0.5
```

FIGURA 9. Nuevas funciones

Se denomina función a cada elemento del software general que tiene una tarea específica, por ejemplo, se presentó la función s para trabajar con sonidos entre otras. Quien usa este software puede crear sus propias funciones que van a ayudar a que la performance tenga su identidad particular. Es importante poner el nombre que queremos darle y cada letra hace referencia al valor que le daremos cuando la utilicemos. En el ejemplo, f es la frecuencia de corte del filtro pasa bajos (lpf) y r es la cantidad de resonancia que le aplicaremos a ese filtro (lpq).

4. Condicionales

```
d1 $ every 3 (# lpf "2000")
    $ whenmod 8 6 ((# lpf "2500").(#lpq "0.5"))
    $ "bd ~ bd [~ sn]"
```

[13] https://tidalcycles.org/index.php/All_effects_and_synths

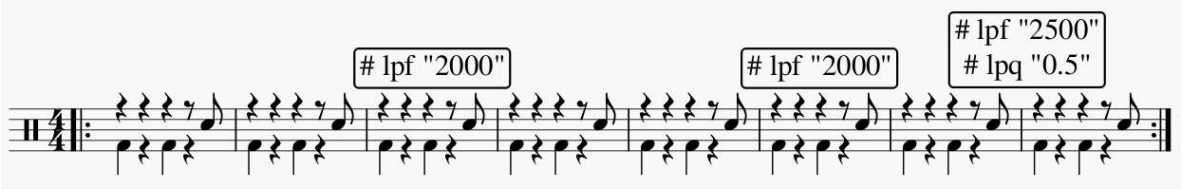


FIGURA 10. Uso de condicionales

Las funciones *every* y *whenmod* permiten poner determinadas condiciones temporales que funcionan de la siguiente manera:

- every se utiliza para que, luego de un número dado de ciclos, (every n, siendo n el número de ciclos) suceda una acción. En el ejemplo cada tres ciclos se aplicará el lpf.
- whenmod, interpola el proceso definido entre paréntesis en los ciclos delimitados por valores dados. En el ejemplo se aplican los efectos, durante el sexto y séptimo ciclo. Es importante señalar que los ciclos se cuentan con índice cero, lo que implica que comienza a contar desde ese número.

5. Bloque *do* y *stack*

```
do
  let filt f r = lpf f # lpq r
  d1 $ every 3 (fast 2)
      $ whenmod 8 6 (jux (rev))
      $ stack[
        "bd ~ bd [~ sn]"
        , s "{jvbass jvbass:1 jvbass:2}%4" # gain "0.95"
        , s "[808:1 [808:1 808:1]]*2"
      ] # filt 2000 0.3
  d2 $ s "arpy*8" # filt 2500 0.1
```

FIGURA 11. Bloque Do y Stack en simultáneo

Estos elementos permiten juntar varias líneas de comandos y ejecutarlas en simultáneo, aspecto que es muy útil para la performance en vivo.

- *stack* ‘apila’ líneas de código donde podremos aplicar efectos por separado. Por otra parte se pueden poner efectos para todo el stack luego de cerrar corchetes (como se ve en el ejemplo).

- El bloque *do* funciona de manera similar pero podremos incluir otros canales de audio, d1, d2, etc. Y además podremos incluir nuestras funciones para modificarlas en el transcurso de la obra.

6. Performance: ‘Nuevas Canciones de Bandidxs’

Canciones de bandidxs es un EP lanzado en 2019[14] donde se encuentran una serie de obras en las que parte de los audios utilizados, son grabaciones de la resistencia en Chile durante ese año. Además se incorporaron fragmentos de ‘el derecho a vivir en paz’ por Victor Jara[15]. En esta ocasión se retoman dos de esos temas (‘canciones de bandidxs’ y ‘vea lo que está haciendo’) sumándole dos piezas nuevas compuestas de manera similar, trabajando con samples de los sucesos recientes en relación con la reforma constitucional que se está por llevar adelante en el país vecino. La idea central de esta obra es re-contextualizar los sonidos y eventos que los mismos evocan en el marco de una música electrónica bailable, acompañando y celebrando el movimiento popular y, a su vez, reinterpretando el archivo de las protestas a través del live coding.

Esta performance utiliza Tidal Cycles para generar la música e Hydra[16] para realizar las visuales. Dado el contexto de distanciamiento social preventivo y obligatorio (pandemia de coronavirus 2020), la performance se realizará vía streaming, transmitiendo un video pregrabado para asegurar la estabilidad y disponibilidad de la misma. La misma se estrenará en el festival artimañas 2020, quedando luego alojada en el canal de youtube *Torotumbo*[17] y el sitio https://torotumbo.glitch.me/ .

7. Conclusión

[14] https://torotumbo.bandcamp.com/album/canciones-de-bandidxs
[15] https://youtu.be/GssGeCif3fk
[16] https://github.com/ojack/hydra
[17] https://www.youtube.com/channel/UCeV8EyNpF56e8E_GXo9YSFw

Este texto acompaña al registro de una performance de live coding musical en clave algorave cuyo estreno se realizará al momento de evaluar el trabajo.
Si entendemos como instrumento musical a los objetos con los que se produce y desarrolla música, y retomamos la idea del algoritmo como el objeto digital artístico, podemos afirmar que el código es el instrumento con el que se ejecutan los algoritmos para realizar piezas de live coding musical.

En este trabajo se propuso comprender la dinámica discursiva del live coding musical. Esta fue abordada como el hecho de escribir código fuente para crear música, donde la construcción de secuencias de eventos y la modificación de las mismas son la centralidad de esta disciplina. El ámbito de algorave, se comprendió como ese espacio donde circulan estas obras, siendo los géneros predominantes cercanos a la música electrónica de baile.
El acercamiento teórico-práctico a la composición algorítmica en entornos de live coding quedó circunscripto en el análisis de 5 usos sintácticos del software Tidal Cycles:

- Lista de eventos sonoros y organización temporal
- Procesamiento de sonido
- Creación de nuevas funciones
- Condicionales
- Bloque do y stack

En la escena internacional, el live coding está en crecimiento desde hace 20 años, a partir del trabajo de TOPLAP. En Argentina existe hace 5 años, teniendo mayor circulación desde el surgimiento del Colectivo de Live Coders (CLiC) y de la colaboración de sus integrantes con miembrxs de TOPLAP. Estos espacios tienen un gran componente colaborativo donde el aprendizaje colectivo es central y se busca el apoyo mutuo para llevar adelante la comunidad.

Este es un primer trabajo de difusión sobre estas prácticas artísticas, con el cual espero acercar esta disciplina a la comunidad de la FDA. El live coding podría desarrollarse de manera interdisciplinar con actores de los distintos departamentos

de esta unidad académica y del Laboratorio de Software Libre para el Arte y Diseño (SLAD). A su vez, este trabajo puede aportar a que esta casa de estudios se ponga en contacto con las distintas comunidades e instituciones que practican e investigan live coding.

8. Glosario

Algorave: Serie de eventos donde se llevan adelante prácticas de live coding. https://algorave.com/

CLiC: Comunidad de live coding originada en la ciudad de La Plata, actualmente cuenta con miembros de distintas ciudades del mundo, principalmente de Argentina. https://colectivo-de-livecoders.gitlab.io/

Livecoder: Término con el que se denomina a las personas que practican live coding.

SuperCollider: Plataforma para la síntesis de audio y la composición algorítmica. https://supercollider.github.io/

SuperDirt: Motor de audio de Tidal Cycles.
https://github.com/musikinformatik/SuperDirt

Synthdef: Nombre con el que se denominan los sintetizadores dentro de SuperCollider. https://youtu.be/jirSBMaQ7Ug

Tidal Cycles: Lenguaje de programación para realizar live coding. https://tidalcycles.org/

Toplap: Comunidad internacional dedicada al estudio y practica del live coding. https://toplap.org/

9. Indice de figuras

10. Bibliografia

Cormen, T. et al. (2001). *Introduction to Algorithms*, Second Edition. MIT.

Gutiérrez López, C. (2018): *El algoritmo como objeto digital artístico prototípico*, Universidad Autónoma del Estado de México.

Hui, Y. (2017). *¿Qué es un objeto digital?* (Ethel Rueda, Katheryn Picón, trad.). Virtualis, [online] (15, Enero- Junio 2017), pp.81-96. Disponible en: https://www.academia.edu/34107670/_QU%C3%89_ES_UN_OBJETO_DIGITAL [Último acceso 15 Nov. 2020]. (Obra original publicada en 2012).

Simondon, G. (2008). *El modo de existencia de los objetos técnicos* (2.ª edición). Buenos Aires: Prometeo Libros.

Villaseñor, H. (). *Live Coding en red*, Universidad Nacional Autónoma de México.