DOSIS - Processing Live Coding Interface

Camilo Andrés Nemocón Farfán (canfcero@gmail.com)

Bogotá, Colombia

Resumen— **Este artículo da a conocer el desarrollo de una interface llamada DOSIS, para el software sketchbook Processing, donde la interface permite generar live coding para la creación de visuales y la conexión con SuperCollider para la modificación de síntesis de sonidos a partir del uso de dispositivos de interacción como el Kinect, Arduino y Joysticks.**

Palabras Claves— **Processing, Live Coding, Kinect, Arduino, Joysticks, GamePads, SuperCollider, OSC, TidalCycles.**

I. INTRODUCCION

Dosis es un ambiente interactivo enfocado al live coding, con un lenguaje dinámico (McLean, 2010), que permite correr algoritmos mientras estos son modificados en tiempo de ejecución del programa, generando audio y visuales en tiempo real por código. Esta interface es abierta y libre, y no intenta imitar o copiar las herramientas existentes de live coding (SuperCollider, Tidal, Praxislive, Hydra, Field) enfocados a la producción de audio y visuales mediante código, sino tiene como objetivo ser un ambiente de interacción de live coding, en donde se puedan usar dispositivos de interacción (Kinect, Wired Glove, Gamepads, Joysticks, sensores, Arduino) de forma fácil y simple, proporcionándole a los artistas otro tipo de controles paralelos al código para crear audio y visuales durante el performance.

Estos controles de interacción usar dispositivos de entrada al sistema como el Kinect o Joysticks, donde a partir de los movimientos del usuario o el público, o al oprimir los botones del Gamepad, le envía datos a la interface para que se pueden modificar la síntesis de audio o crear composiciones visuales. Así mismo, Dosis también puede generar como salida del sistema, sonidos a partir de objetos, también llamado, hardware hacking (Richards, 2013) por medio de la conversión de digital a análogo, tomando la información digital del código y mapeando lo datos en señales de voltaje y prendiendo los pines análogos y digitales de Arduino para generar performances musicales con Arduino.

El desarrollo de la interface DOSIS, como herramienta de live coding para la generación de visuales y sonido, y su integración con dispositivos de interacción, surge a partir de la necesidad de generar live coding por medio de la herramienta Processing (Reas and Fry, 2001), el cual permite la implementación con dispositivos y conexión con software de sonido por medio del protocolo de comunicación OSC (open sound control).

Teniendo como base que Processing es una plataforma para desarrollar imágenes, animaciones o interactividad (Reas and Fry, 2007), a partir de la compilación del código, por tanto no permite renderizar el código en tiempo de ejecución (live coding), sin embargo es una herramienta de programación básica, fácil de usar y aprender, con una estructura orientada a objetos y se basa en lenguaje de programación Java, con lo cual permite convertir los skecthbooks desarrollados en Processing en clases para que se puedan integrar en el programa DOSIS, importando todas las librerías que se van a utilizar y declarando e inicializando las clases para que el programa las pueda ver desde cualquier función que se realice y así poderlas utilizar en la interface DOSIS.

A partir de las anteriores características se desarrolla un programa en Processing que genere una interface gráfica para escribir código y que éste se renderice en tiempo de ejecución, permitiendo conectar y usar dispositivos de interacción para crear visuales en tiempo real y generar composiciones de audio conectando Dosis con programas de audio de Live Coding.

Aunque existen otras plataformas de live coding desarrolladas a partir de Processing, como PraxisLive y Field, los cuales son ambientes de desarrollo para generación de gráficas y audio; estas aplicaciones no se enfocan en proporcionarle al usuario una fácil sintaxis de uso y conexión a dispositivos de interacción, ni tampoco permite generar de forma sencilla perfomance sonoros con sensores de salida de Arduino.

PraxisLive y Field maneja programación gráfica y en código, lo cual complejiza su uso debido al multiparadigm code que utiliza (Wakefield and Roberts, 2014) y aumenta la curva de aprendizaje del usuario, no solo para aprender la sintaxis de programación de Processing en Praxis o de JS en Field, sino también para aprender a usar el UI de ambas plataformas y su programación gráfica. Dosis por el contrario sólo usa programación en código y reduce la sintaxis de código para el uso de dispositivos o de creación de graficas o audio. La sintaxis de código de Dosis siempre está presente en la parte inferior de la aplicación para guiar al usuario.

II. Diseño de la interface Dosis

El diseño para la interface Dosis se baso en el concepto de patrones del lenguaje (Blackwell, 2015) en torno a la programación performance, introduciendo un medio entre el artista y la audiencia para aproximarlos a ambos y generar una experiencia en conjunto de construcción y colaboración (Olaya and Zapata, 2018), teniendo en cuenta la creatividad en la programación y el código como medio principal de interacción, acompañado a dispositivos con los cuales el público participa y responde al performance.

Así como Ixi Lang (Magnusson, 2011) fue diseñado para que el perfomer en el live coding sonoro, se sintiera libre para crear, sin tener que pensar a nivel de la computación científica. La interface Dosis fue pensada con el mismo objetivo, donde el artista usa los dispositivos mediante códigos sencillos para enfocarse en la interacción con Kinect, GamePads o sensores de Arduino para crear sus performance visuales y musicales.

A continuación, se presenta el funcionamiento en detalle del ambiente de interacción de live coding Dosis, en donde se muestran sus componentes y la conexión entre ellos, para enviar los datos que se tipean en código en vivo desde el componente Dosis Cliente para modificar en tiempo real el audio en SuperCollider, generar visuales en Dosis Servidor o activar dispositivos de interacción como Arduino, Kinect y Joysticks.

III. Componentes y conexión

La interface DOSIS, esta compuesta por dos componentes principales, un programa servidor y un programa cliente, donde el programa servidor es el que se encarga de mostrar las visualizaciones interactivas de los skecthbooks implementados y estas visualizaciones cambian a partir de los códigos enviados por parte del cliente.

El programa cliente es en donde se escribe el código en tiempo de ejecución, y el que detecta y usa los dispositivos de interacción (Kinect, Arduino, Joysticks), y le envía la orden o los códigos al programa servidor para mostrar la visual y afectar la visual a partir de todos los inputs realizados en el programa cliente.

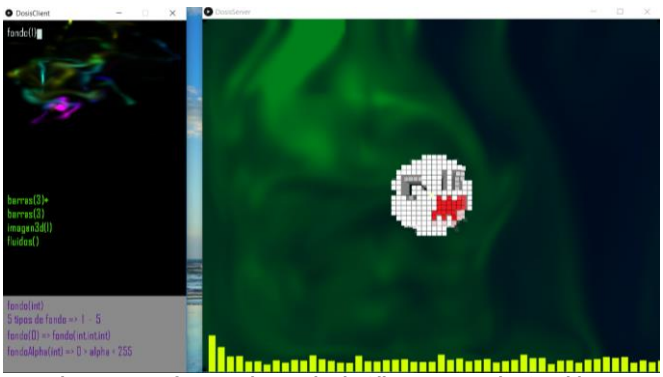

Figura 1. Interface Dosis. Izquierda Cliente – Derecha Servidor.

La conexión cliente – servidor se realiza mediante el protocolo de comunicación OSC usando la transmisión por TCP, de esta forma el programa cliente envía mensajes OSC al programa servidor para generar visuales y también puede enviar mensajes a SuperCollider para modificar sonidos, siempre y cuando los computadores estén conectados a la misma red.

Al utilizar esta conexión en red, se puede enviar y recibir mensajes por OSC, permitiendo toda la interoperación, implementación y conexión entre varios computadores (Dannenberg and Chi, 2016), mejorando los problemas de interconexión en redes de área local.

La aplicación cliente se conecta a la red y envía datos a los demás computadores servidores, los cuales pueden generar visuales con la interfaz Dosis o con la aplicación Hydra (Jack, 2017), y/o crear sonidos en SuperCollider (McCartney, 2002). Finalmente, todos los programas se logran comunicar mediante la interfaz cliente de DOSIS, formando un ambiente colaborativo en red donde en cada computador se puede generar una creación visual y/o sonora simultáneamente con la persona que interviene e interactúa con la interfaz cliente.

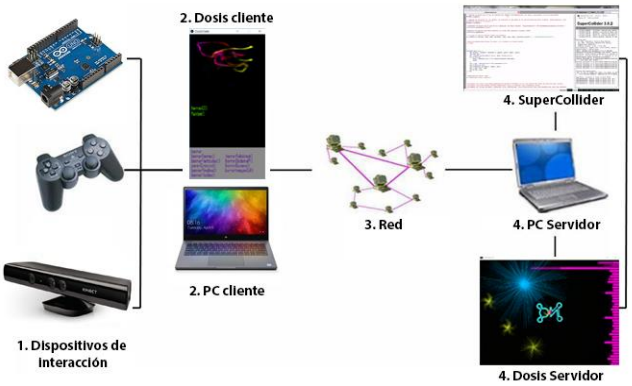

Figura 2. Esquema conexión en red cliente - servidor.

IV. Desarrollo e implementación de la Interface Cliente Dosis

Interface Cliente

La aplicación cliente tiene la posibilidad de integrar dispositivos de interacción o solo utilizar el código preestablecido para convertirse en una visual musician making system (Lee and Essl, 2017), donde la integración de las acciones con los dispositivos y la codificación de algoritmos crean un performance en tiempo real.

Ésta aplicación se desarrolló en Processing 3.3.7 y está compuesta por 3 librerías netP5, oscP5 y msaFluids (Akten, MSA Fluid, 2008) y posee funciones para la conexión y envío de los mensajes desde la interface cliente DOSIS a la interface servidor DOSIS para generar las visuales. Ambas interfaces se pueden correr en el mismo computador, o pueden correr en computadores independientes transmitiendo la información en tiempo real de una interface a otra o de una interface a otras aplicaciones de audio y visuales. Para ello se especifica el puerto y la dirección IP

de los computadores a los cuales envía la información de los códigos escritos por el performer para activar las visuales, los audios y/o los dispositivos de interacción.

DosisClient | msaFluids

```
//------------------------Setup--------------------
void setup()
{
  size(400,768);
  frameRate(30);

  noCursor();

  // start oscP5, listening for incoming messages at port 12000
  oscP5 = new OscP5(this,12000);

  //se conecta al servidor por la dirección ip,port
  // dosisConection = new NetAddress("192.168.0.101",12000);
  dosisConection = new NetAddress("localhost",12000);
```

Figura 3. Configuración del puerto y dirección IP.

La aplicación le muestra al usuario cuales son los códigos preestablecidos de Live Coding que puede usar para crear las visuales, enviando estos códigos a la interface Dosis servidor, para así generar la visualización. Se utilizan códigos preestablecidos como parte de mini lenguajes de patrones (Collins and McLean, 2014), para facilitar la identificación de funciones que se pueden usar para crear las visualizaciones o composiciones de audio. Estos códigos se muestran en la parte inferior de la interfaz, lo que le sirve de guía al usuario durante el performance.

```
void instrucciones()
{
  fill(#8B8A8B);
  noStroke();
  rect(0,600,width,height);

  fill(#550F90);
  if(palabraInstruccion == 1)
  {
    text("palabras(String,String,String,String)",10,630);
    text("palabras(1) => tamaño de la letra = audioInput",10,670);
    text("palabras(0) => tamaño de la letra = fija",10,710);
  }
  else if(palabraInstruccion == 2)
  {
    text("volumen(float)",10,630);
    text("audio input =>  0.0  -  1.0 ",10,670);
  }
  else if(palabraInstruccion == 3)
  {
    text("barras(int)",10,630);
    text("4 tipos de Buffer =>  1  -  4",10,670);
  }
```

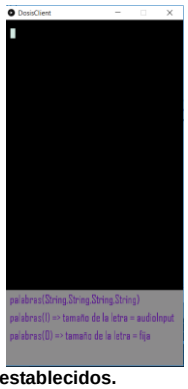

Figura 4. Código y visual de los códigos preestablecidos.

La interface permite su conexión con la aplicación de sonido SuperCollider (SC), el cual es un programa de bajo nivel de lenguaje de programación que sirve para la creación y modificación de síntesis de sonidos. Esta conexión se realiza por OSC, enviando uno o varios valores enteros desde la interface Dosis cliente a SC, para modificar sonidos en "tiempo real", ya que los mensajes o datos son enviados a SC cada 500 milisegundos para no sobresaturar el buffer de comunicación y darle el tiempo suficiente para que procéselos datos y ejecute los eventos de síntesis de sonido.

DosisClientToSC | msaFluids

```
//------------------------Setup--------------------
void setup()
{
  size(400,768);
  frameRate(30);

  noCursor();

  // start oscP5, listening for incoming messages at port 12000
  oscP5 = new OscP5(this,12000);

  //se conecta al servidor por la dirección ip,port
  // dosisConection = new NetAddress("192.168.0.101",12000);
  dosisConection = new NetAddress("localhost",12000);

  //superColliderConection = new NetAddress("172.16.24.80",33333);
  superColliderConection = new NetAddress("localhost",33333);
```

OscRecieveDosis.scd

```
//colocamos el puerto por el que se va a comunicar con Dosis Cliente: "dosisConection = new NetAddress(dirección IP,33333);"
thisProcess.openUDPPort(33333);

//muestra los puertos que estan abiertos, en donde debe aparecer el puerto 33333
thisProcess.openPorts;

// se crea la funcion OSC que recibe los mensajes
o = OSCFunc({ arg msg, time, addr, recvPort; [msg, time, addr, recvPort].postln; }, '/"DosisComunicacionSC"');

(
SynthDef.new(\tone, {
    arg freq=40, nharm=12, detune=0.2, gate=0, pan=0, amp=1, out=0;
    var sig, env;
    env = EnvGen.kr(Env.perc(0.01,2), gate, doneAction:2);
    sig = Blip.ar(
        freq * LFNoise1.kr(0.2!16).bipolar(detune).midiratio,
        nharm
    );
    sig = sig * LFNoise1.kr(0.5!16).exprange(0.1,1);
    sig = Splay.ar(sig);
    sig = Balance2.ar(sig[0], sig[1], pan);
    sig = sig * env * amp;
    Out.ar(out, sig);
}).add;
)

//se genera una funcion que queda pendiente de recibir el mensaje con los dos valores del mouse de parte de Dosis Cliente
//el primer valor del mensaje recivido modificará \freq y el segundo valor modificará \nharm
//se mapean los valores enviados .linexp(min valor recibido,max  valor recibido,min valor que necesito,max valor que necesito)
(
OSCdef('starlistener',
    {
        arg msg, time, addr, port;
        Synth('tone',[\gate,1, \freq, msg[1].linexp(0,400,20,500), \nharm, msg[2].linexp(0,300,1,500)] );
        msg[1].postln;
        msg[2].postln;
    },"/DosisComunicacionSC");
)
```

Figura 5. La imagen superior muestra el código de Processing para la conexión y la imagen inferior muestra el código de SC para la conexión.

Este funcionamiento se puede evidenciar en el siguiente ejemplo en donde se muestra la conexión e interacción entre Dosis y SuperCollider, donde a partir del movimiento del mouse o del valor escrito en la interface Dosis se modifica la frecuencia y el armónico de los sonidos reproducidos en SuperCollider.

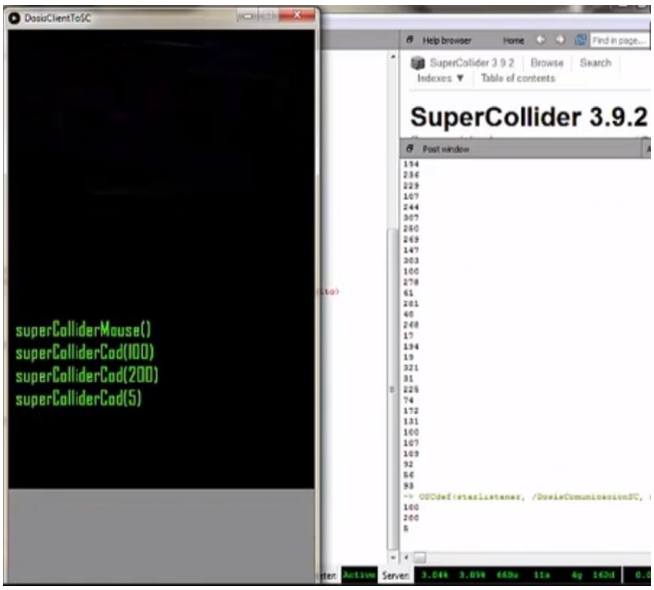

Figura 6. Dosis & Super Collider.
(https://vimeo.com/282070208)

Por otro lado, uno de los objetivos de la interface cliente aparte de enviar códigos de Live Coding para generar sonidos, es enviar datos de dispositivos de interacción; por lo cual a la aplicación base de la interface cliente se le

adicionan las librerías del Kinect (“OpenNI”), Arduino (“Firmata”) y/o Joysticks (VrpnForProcessing).

Se desarrollaron 3 versiones de la interface cliente DOSIS, una por cada dispositivo de interacción. A continuación, se explicará el funcionamiento para cada dispositivo.

A. Interface Cliente Arduino (Leonardo y Uno)

En Porcessing se importa la librería Firmata (Steiner, 2009) y Serial, para realizar la conexión entre Dosis y el dispositivo Arduino, para poderle enviar bytes a éste, mediante la escritura de letras con el teclado sobre la interface donde cada letra refleja un pin digital o análogo en Arduino, por tanto, se configuraron las teclas del 2 al 9 y prenden los pines digitales del 2 al 9, las letras de la A hasta la D prenden los pines digitales del 10 al 13, las letras desde la E hasta la H prende los pines análogos desde A0 hasta A3, y las demás letras continúan el orden empezando nuevamente desde el pin digital 10 hasta el pin análogo A3.

La interface Dosis Arduino, tiene 3 principales códigos preestablecidos (Collins and McLean, 2014), el primero es onceArduino(), el cual prende los pines en Arduino una sola vez, y los pines que prende corresponde a las letra que se escriban posteriormente de haber escrito este código en la interface. El segundo código es loopArduino(), el cual prende los pines correspondientes a las letras iterativamente en el orden en que se escribió el mensaje en la interface y finalmente el último código es pararLoopArduino(), el cual genera que Arduino no prenda ningún pin.

A continuación, se muestra un caso de ejemplo en donde se utiliza Dosis como herramienta de Live Coding para prender leds con Arduino, a partir de los números de los pines escritos por código en la interface.

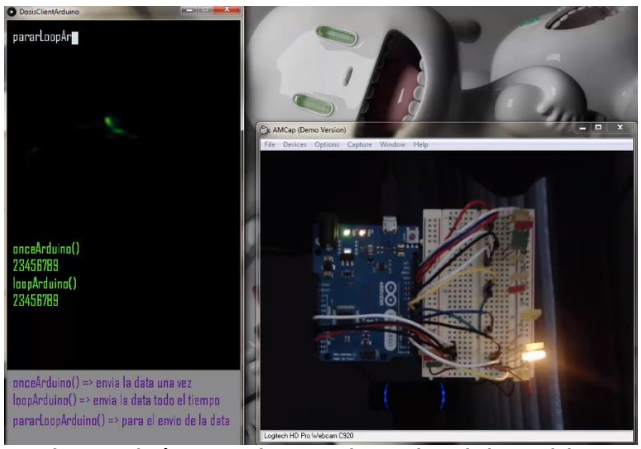

Figura 7. El número escrito en Dosis prende un led en Arduino. (https://vimeo.com/282589746)

Otro caso de uso, que se realizó como resultado de un Workshop de Dosis, fueron proyectos de hardware music (Collins, N. 2006) en donde se enfocaron en generar performances sonoros a partir del hacking de objetos con Arduino, manipulados en tiempo real por código mediante Dosis, generando ritmos a partir del uso de motores, parlantes, cámaras de seguridad, ventiladores, relay, drones, radios, máquinas de escribir entre otros.

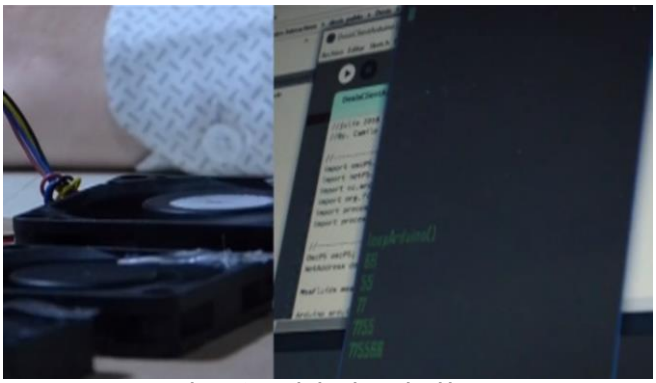

Figura 8. Dosis hardware hacking (https://vimeo.com/292636194)

B. Interface Cliente Kinect 360

Para la implementación del Kinect en la interface Dosis, se determinó usar la librería SimpleOpenNI (Vega, 2018) para Processing, la cual necesita de la instalación previa del Kinect SDK v1.7, para que el computador Windows reconozca la cámara Kinect. La limitante existente es que los drivers SKD para el Kinect sólo funciona para sistemas operativos Windows.

Por medio de la librería podemos obtener los datos del esqueleto del usuario, por tanto, la posición como un vector (X,Y,Z) de cada una de las articulaciones. Es así como la interface muestra el depth image en donde realiza el tracking del usuario y envía los datos de la cabeza, el torso, la mano derecha y la mano izquierda.

Estos datos son enviados mediante el código preestablecido, Kinect(), mediante el cual se envía la información por mensaje OSC a la interface Dosis Servidor para desplegar una visual en donde muestra la posición y movimiento de la cabeza y manos del usuario.

Por otro lado, esta información también es enviada a SuperCollider, con lo cual se puede modificar o reproducir las síntesis de sonido. Es así como este motor de audio recibe cuatro mensajes, el primero es la posición horizontal de la mano derecha, el segundo provee la posición horizontal de la mano izquierda, el tercero el gesto de “push” generado por el usuario con la mano izquierda (mano izquierda extendida frente a la cámara kinect) y el cuarto mensaje es el mismo gesto con la mano derecha.

Esto se evidencia en el siguiente ejemplo, en donde se muestra la interacción del Kinect con el sonido y la gráfica, mediante el ambiente de live coding Dosis, donde a partir del código usado por el artista se determinan los movimientos del cuerpo que se pueden usar para reproducir los sonidos y manipular las gráficas ejecutadas por código.

Figura 9. Dosis Kinect Client & Server (https://vimeo.com/283835424) (https://vimeo.com/283835882)

C. Interface Cliente Joystick

Para poder utilizar el Joystick en la interface, por ende, en Processing se implementa la librería de VRPN "Virtual Reality Peripherical Network" (Taylor and Hudson, 2001), con la cual se puede utilizar y recibir el input de distintos dispositivos como motion trackers (Phase Space, Flock of Birds Tracker, Data Gloves) o controles Joysticks.

El Joystick, envía al VRPN coordenadas en X y en Y, con valores que van del 1 al -1, también posee botones, los cuales envían el dato del botón oprimido por el usuario.

Con la instalación previa del VRPN en el computador y la implementación de la librería en Processing (Nemocon, 2018), tenemos acceso al Joystick con el cual el usuario puede modificar el sonido o las visuales desde la interface cliente, permitiendo una herramienta adicional para el live coding.

La interface cliente muestra la retroalimentación visual de las acciones que realiza el usuario con el dispositivo Joystick, y cuando se crearon dos códigos predeterminados, el primero es joystick(), con el cual le envía los datos con los valores de los botones y la palanca del Joystick. El segundo código es joystickParar(), el cual para de enviar mensajes a la interface servidor Dosis.

V. Desarrollo e implementación de la Interface Servidor Dosis

Interface Servidor

Esta aplicación se encarga de recibir los mensajes OSC que escribió el usuario en la interface Dosis cliente, procesando y convirtiendo los mensajes en visuales en tiempo real. Estas visuales son interactivas y cambian a partir del código preestablecidos escrito, también de la ubicación del mouse en la interface cliente o del sonido ambiente.

La estructura de esta aplicación está hecha a partir de clases, donde un sketchbook hecho en Processing puede ser convertido en una clase e integrarlo dentro de la inteface Dosis servidor, declarando e inicializando la clase dentro del código DosisServer.pde, así mismo, en la función protocolo() se coloca la palabra clave que se desea recibir desde la interfaz cliente para que se despliegue la visual de la clase implementada, cuando se reciba el mensaje con el código preestablecido.

DosisServer

```
272
273 void protocolo(String msn)
274 {
275   String temp = "";
276   String[] mensaje;
277   String[] mensajeDatos;
278   boolean mensajeValido = false;
279
280   if(msn.substring(msn.length()-1).equals(")"))
281   {
282     temp = msn.substring(0,msn.length()-1);
283     mensajeValido = true;
284   }
285
286   if(mensajeValido == true )
287   {
288       mensaje = split(temp,'(');
289
290
291       //SI LLEGA ESA PALABRA ENTONCES VA A COLOCAR LAS PALABRAS
292       if (mensaje[0].equals("palabras"))
293       {
294         //inicializa los textos
295         textos = new Textos();
296
297         mensajeDatos = split(mensaje[1],',');
```

Figura 10. Código de la función protocolo().

La Interfaz del Servidor de Dosis tiene once visuales implementados, donde cada visual corresponde a una clase diferente, pero la estructura permite la implementación de más visuales para tener varias opciones de composición en el rendimiento de la codificación en vivo. A continuación, se muestran unos ejemplos del tipo de visuales que se pueden realizar con Dosis.

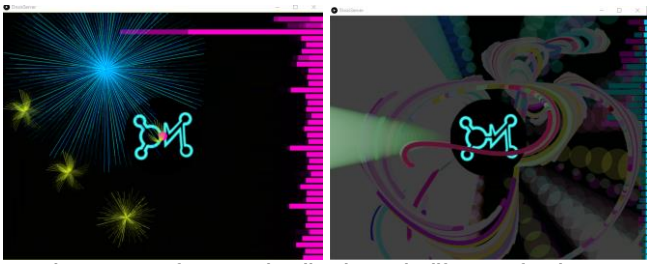

Figura 11. Interface con visualizaciones de diferentes funciones. (https://vimeo.com/286624611)

VI. Conclusiones y Trabajo futuro

Esta interface se ha utilizado en muestras de live coding como resultado del Taller Live Coding Asincronia Audiovisual realizado en Bogotá, Colombia en el 2018 (https://vimeo.com/278199568), en donde los participantes interactuaban por medio del dispositivo Kinect para modificar los sonidos generados en live coding por medio Super Collider y las visuales en Hydra, haciendo uso de la interface Dosis.

En esta misma muestra se utilizó Dosis, como interface de control del proyecto Machine Noise de Sebastián Gómez (https://vimeo.com/278099527), donde se Hackearon máquinas electrónicas de escribir con Arduino y se enviaban

datos a Arduino por medio de Dosis para crear ritmos a través del live coding.

Inicialmente la interface permitió la integración del Kinect, y se le fueron adicionando funcionalidades para el uso de otros dispositivos de entrada como Joysticks y de salida como Arduino, para utilizar diferentes herramientas de control, modificación y generación de audio y visuales.

El uso de Processing permite una fácil inclusión de cualquier sketchbook dentro de la interface servidor de Dosis para así poder visualizar cualquier ejercicio hecho en Processing haciendo uso de la práctica de live coding generando visuales en tiempo real.

A nivel de performance permite poder realizar producciones colaborativas ya que a partir del uso de la interface Dosis se pueden comunicar diferentes computadores enviándose mensajes para la creación o modificación de sonidos y visuales mediante programas como SuperCollider e Hydra, permitiendo también la participación del público dentro del performance sin necesidad de que ellos tengan que realizar coding sino por medio del movimiento del cuerpo con el Kinect o del uso de Joysticks, de esta forma se genera una inclusión con las personas que desean participar pero que no saben hacer código en las plataformas de audio o sonido.

Esta interface se deja como plataforma de código abierto (https://github.com/camiloNemocon/DOSIS) y como trabajo a futuro se propone vincular más dispositivos de interacción e implementar más funcionalidades a partir de la retrolaimentación recibida por usuarios y desarrolladores.

VII. Referentes

Akten, Memo. 2008. “MSA Fluid”. URL:
http://www.memo.tv/msafluid/

Blackwell, Alan. 2015. “Patterns of User Experience in Performance Programming.”
Proceedings of First International Conference on Live Coding. Zenodo.
http://doi.org/10.5281/zenodo.19315

Collins, Nick, and Alex McLean. 2014. “Algorave: Live Performance of Algorithmic Electronic Dance Music.”
Proceedings of the International Conference on New Interfaces for Musical Expression.
http://www.nime.org/proceedings/2014/nime2014_426.pdf.

Collins, Nicolas. 2006. “Handmade Electronic Music: The Art of Hardware Hacking.”
Published in Computer Music Journal. Pp: 96-99.
http://loliel.narod.ru/DIY.pdf

Dannenberg, Roger and Zhang Chi. 2016. “O2: Rethinking Open Sound Control”
Proceedings of the 42nd International Computer Music Conference, Utrecht: HKU University of the Arts Utrecht, 2016, pp. 493 - 496.
https://www.cs.cmu.edu/~music/cmsip/readings/o2-web.pdf

Jackson, Olivia. 2018. “Hydra.”
https://github.com/ojack/hydra.

Lee, Sang. George Essl, and Mari Martnez. 2017. “Live Writing as a Real-Time Audiovisual performance.”
Proceedings of the New Instruments for Musical Expression (NIME).
https://cpb-us-w2.wpmucdn.com/people.uwm.edu/dist/0/236/files/2016/09/Lee-NIME16-livewritingmusicperformance-1309rq3.pdf

McCartney, James. 2002. “Rethinking the computer music language: SuperCollider”.
Computer Music Journal Volume 26: 61-68
https://www.mitpressjournals.org/doi/abs/10.1162/014892602320991383?journalCode=comj

Magnusson, Thor. 2011. “ixi lang: A SuperCollider Parasite for Live Coding”
Proceedings of International Computer Music Conference 2011. Pp: 503-506.
http://hdl.handle.net/2027/spo.bbp2372.2011.101

McLean, Alex. 2010. “Visualization of live code”
Proceeding of EVA'10 Proceedings of the 2010 international conference on Electronic Visualisation and the Arts: 26-30
https://dl.acm.org/citation.cfm?id=2227185

Olaya, Juan. Zapata, Laura and Nemocón, Camilo. 2018. “Full-Body Interaction for Live Coding”
Poceeding of the International Conference of Live Coding, ICLC 2019.

Reas, Casey and Ben Fry. 2001. “Processing.” 2001.
https://processing.org/.

Reas, Casey and Ben Fry. 2007. “Processing: Programming for the Media Arts.”
Proceeding of AI and Society 20 (4): 526–38.
https://doi.org/10.1007/s00146-006-0050-9

Richards, John. 2013. “Beyond DIY in Electronic Music”.
Proceeding of Organized Sound: 274-281.
https://doi.org/10.1017/S1355771813000241

Schlegel, Andreas. 2015. “OscP5: An OSC Library for Java and the Programming Environment Processing”.
https://doi.org/10.5281/ZENODO.16308.

Steiner, Hans. 2009. “Towards making microcontrollers act like extensions of the computer.”
Proceedings of the New Instruments for Musical Expression (NIME). pp. 125-130.
http://archive.notam02.no/arkiv/proceedings/NIME2009/nime2009/pdf/author/nm090182.pdf

Taylor, Russel, Thomas Hudson and Adam Seeger. 2001. “Device-independent, network-transparent VR peripheral system”.

Proceedings of the ACM symposium on Virtual reality software and technology: 55-61. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.298.6271&rep=rep1&type=pdf

Vega, Antoino. 2017. “OpenNI library for Processing”. URL: https://github.com/totovr/SimpleOpenni/tree/Processing_3.4

Wakefield, Graham and Roberts, Charlie. 2014. “Collaborative Live-Coding with an Immersive Instrument.” Proceeding of the Conference on New Interfaces for Musical Expression. Pp:505-508 https://www.researchgate.net/profile/Graham_Wakefield/publication/320736396_Collaborative_Live-Coding_Virtual_Worlds_with_an_Immersive_Instrument/links/59f8a0aea6fdcc075ec99015/Collaborative-Live-Coding-Virtual-Worlds-with-an-Immersive-Instrument.pdf