DOSIS – PROCESSING LIVE CODING INTERFACE

Camilo Nemocón
Universidad de los Andes
canfcero@gmail.com

ABSTRACT

This article introduces the development of an interface called DOSIS for the sketchbook Processing software, where the interface allows to generate live coding for the creation of visuals and the connection with SuperCollider for the modification of sound synthesis through interaction devices like Kinect, Arduino and Joysticks.

1. INTRODUCTION

Dosis is an interactive environment focused on live coding, with a dynamic language (McLean, 2010), which allows algorithms to run while they are being modified in program execution time, generating audio and visuals in real time by code. This interface is open and free, and does not attempt to imitate or copy existing live coding tools (SuperCollider, Tidal, Praxislive, Hydra, Field) focused on the production of audio and visuals through code, but aims to be an environment of interaction of live coding, where you can use interaction devices (Kinect, Wired Glove, Gamepads, Joysticks, sensors, Arduino) in an easy and simple way, providing the artists with other types of controls parallel to the code, to create audio and visuals during the performance.

These interaction controls use input devices to the system such as the Kinect or Joysticks, where from the movements of the user or the public, or by pressing the Gamepad's buttons, it sends data to the interface to modify the audio synthesis or to create visual compositions. Likewise, Dosis can also generate as a system output, sounds from objects, also called, hardware hacking (Richards, 2013) through conversion from digital to analog by taking the digital information from live code and mapping them to voltage signals and turning on Arduino's analog and digital pins to generate Arduino musical performances.

The development of the DOSIS interface, as a tool of live coding for the generation of visuals and its integration with interaction devices for the public to intervene with the visual and sound compositions, arises from the need to generate live coding in the Processing software (Reas and Fry 2001), which allows an easy implementation of devices and a sound and visual software connected through the OSC (Open Sound Control) communication protocol.

Based on the fact that Processing is a platform to develop visuals, animations or interactivity (Reas and Fry 2007), as of the compilation of the code, therefore it does not allow to renderer the code in runtime (live coding), however Processing is an easy programming tool based on Java language with an object oriented programming structure, which allows to convert the sketchbooks developed in Processing into classes that can be integrated into the DOSIS interface, importing all the libraries that will be used, declaring and initializing the classes, for the program to be able to see any function that is being executed and therefore, to be able to use the functions in the Dosis Interface.

From the above characteristics, Dosis is developed in Processing with a graphical view to write codes and these codes are rendered at runtime, allowing to connect and use interaction devices to create visuals in real time and generating audio compositions by connecting Dosis with audio live coding software.

Although there are other live coding platforms that were developed from Processing, such as PraxisLive and Field, which are development environments for generating graphics and audio; these applications do not focus on providing the user with an easy use of syntax and connection to interaction devices, nor does it allow to generate in a simple way sound performance with Arduino output sensors.

PraxisLive and Field handle graphic and code programming, which complicates its use because of the multi paradigm code it uses (Wakefield and Roberts, 2014) and increases the user's learning curve, not only to learn the programming syntax of Processing in Praxis or JS in Field, but also to learn how to use the UI of both platforms and its graphic programming. Dosis on the other hand, only uses programming in code and reduces

the code syntax for the use of devices or creation of graphics or audio. The Dosis code syntax is always present at the bottom of the application to guide the user.

2. INTERFACE DOSIS DESIGN

The Dosis interface design was based on the concept of language patterns (Blackwell, 2015) around performance programming, introducing a medium between the artist and the audience to approximate both and generate a joint experience of construction and collaboration (Olaya and Zapata, 2018), taking into account creativity in programming and code as the main way of interaction, accompanied by devices with which the public participates and responds to performance.

As well as Ixi Lang (Magnusson, 2011) was designed for the performer to feel free to create on the live coding without having to think at the level of computer science. The Dosis interface was designed with the same objective, where the artist uses the interaction devices through simple codes, allowing performers to engage directly with Kinect, GamePads or Arduino sensors to create musical and graphical performances.

Next, the detailed interaction environment operation of live coding Dosis is presented, where its components are shown and the connection between them, to send the data that is typed in live code from the Client Dosis component to modify in real-time audio in SuperCollider, generate visuals in Dosis Server or activate interaction devices like Arduino, Kinect and Joysticks.

3. COMPONENTS AND CONNECTION

The Dosis interface is composed by two main components, a server application and a client application, where the server is responsible for displaying the interactive visualization of the implemented sketchbooks and these visualizations change from the codes that are sent by the Dosis Interface Client.

The client program is where the code is written at runtime, the one that detects and uses the interaction devices (Kinect, Arduino, Joysticks), furthermore it sends this data or codes to the server program to show the image and affect the visual from all the determined inputs by the user in the client program.

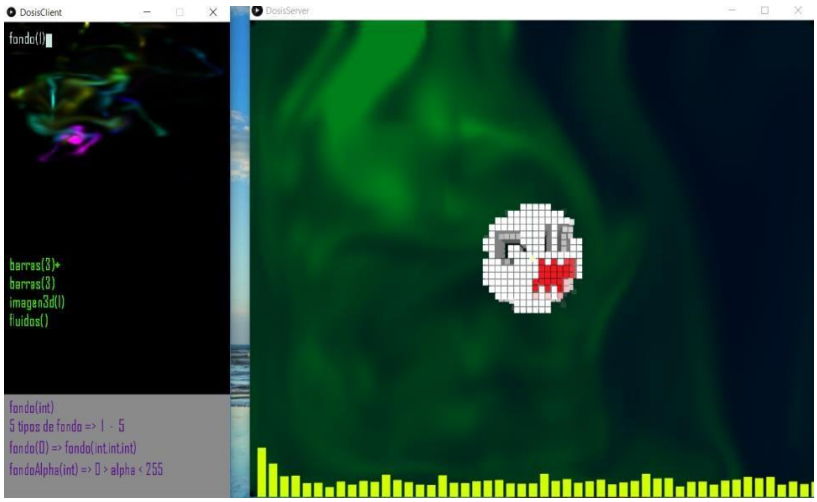


Figure 1. The Dosis Client Interface and the Dosis Server Interface.

The client-server connection is made through the OSC communication protocol using TCP transmission, in this way the client program sends OSC messages to the server program to generate visuals and can also send messages to the SuperCollider to modify sounds, as long as the computers are connected to the same network.

By using this network connection, messages can be sent and received by OSC, allowing all the interoperation, implementation and connection between several computers (Dannenberg and Chi 2016), improving interconnection problems in local area networks.

The client application connects to the network and sends data to the other server computers, which can generate visuals with the Dosis Server Interface or with the Hydra application (Jackson 2018), and/or create sounds in SuperCollider (McCartney 2002). Finally, all the programs are able to communicate through the Dosis Client Interface, forming a collaborative network environment where each computer can generate a

visual and/or sound creation simultaneously with the person who intervenes and interacts with the Dosis Client Interface.

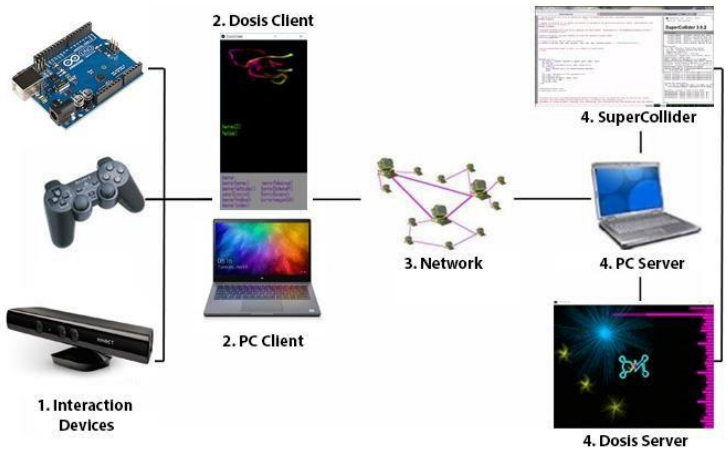


Figure 2. Client – Server connection scheme.

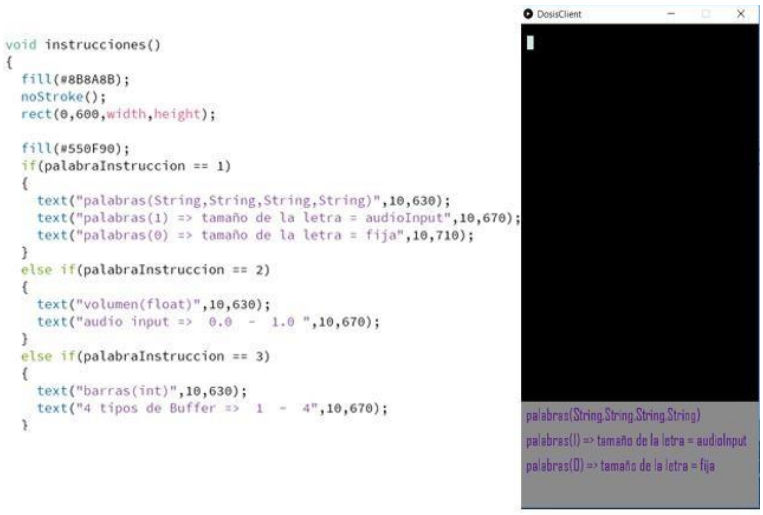
4. DEVELOPMENT AND IMPLEMENTATION OF THE DOSIS CLIENT INTERFACE

4.1 Client Interface

The client application has the possibility of integrating interaction devices or just using the pre-established code to become a visual musician making system (Lee, Essl and Martinez 2017), where the integration of actions with devices and the coding of algorithms create a real time performance.

This application was developed in Processing 3.3.7 and is composed of three netP5 libraries, oscP5 (Schlegel 2015) and msaFluids (Akten 2008) and has functions of connecting and sending messages from the Dosis Client Interface to the Dosis Server Interface, to generate the visuals. Both interfaces can run on the same computer, or they can run on separate computers by transmitting the information in real time from one interface to another or from the Dosis Client Interface to other audio and visual applications. This is done by specifying the port and the IP address of the computers to which it sends the information to activate the visuals, sounds or interaction devices by the performer's live coding codes.

The application shows the pre-established codes that the user can apply to create the visuals for LiveCoding, sending these codes to the Dosis Server Interface, to therefore create the visualization. Pre-established codes are used as part of mini-languages patterns (Collins and McLean 2014), to facilitate the identification of functions that can be used to create visualizations or audio compositions. These codes are displayed at the bottom of the interface, which guides the user during the 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);
  }
```


Figure 3. Instructions code as a guide of the pre-established codes.

The interface allows its connection with SuperCollider (SC) sound application, a program that is low-level language programming, therefore used to create and modify sound synthesis. This connection is made by OSC, by sending one or more integer values from the Dosis Client Interface to the SC interface, to modify sounds in "real time", since the messages are sent to SC every 500 milliseconds to not over saturate the communication buffer and give it enough time to process the data and execute the sound synthesis events.

```
//------------------------------ Dosis Client Interface ------------------------------
// Port and IP address configuration in the Dosis Client Interface To connect SuperCollider.
void setup()
{
  size(400,768);   frameRate(30);
  // start oscP5, listening for incoming messages at port 12000
 oscP5   =   new   OscP5(this,12000);                    superColliderConection   =   new
NetAddress("localhost",33333);

//---------------------------------- SuperCollider -----------------------------------
// Port configuration in SuperCollider to receive messages thisProcess.openUDPPort(33333);
```

This operation can be evidenced in the following example where the connection and interaction between Dosis and SuperCollider is shown, through the movement of the mouse or the written value typed on Dosis interface modify the frequency and the harmonic of the reproduced sounds in SuperCollider.

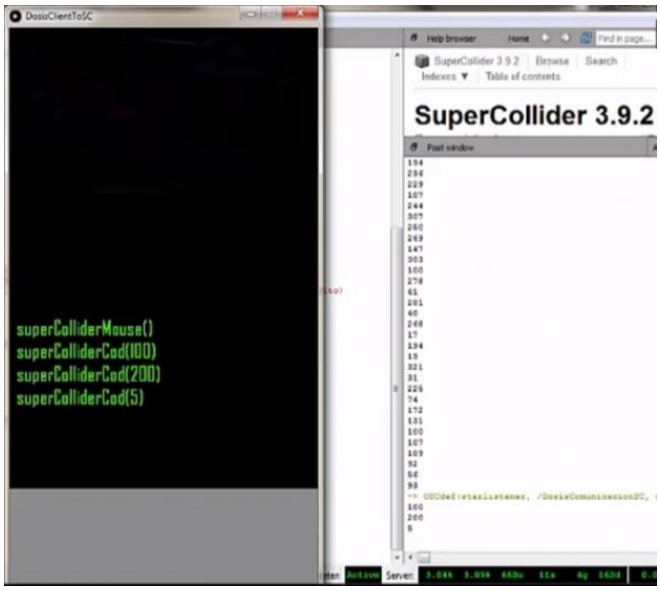


Figure 4. Dosis and SuperCollider. (https://vimeo.com/282070208)

On the other hand, one of the objectives of the client interface apart from sending Live Coding codes is to send data from interaction devices; therefore, the libraries of the Kinect ("OpenNI"), Arduino ("Firmata") and Joysticks (VrpnForProcessing) are added to the Dosis Client Interface application.

Three versions of the Dosis Client Interface were developed, one for each interaction device. Hereunder, the integration for each device will be explained.

4.1.1 Arduino Interface (Leonardo & UNO)

In Processing, Firmata (Steiner 2009) and Serial libraries are imported to make the connection between the Dosis and the Arduino device, in order to send bytes that are typing letters with the keyboard on the Dosis Client Interface where each letter reflects a digital pin or analog pin in Arduino. Thus, the keys from 2 to 9 turn on the digital pins from 2 to 9, the letters from A to D turn on the digital pins from 10 to 13, the letters from E to H turn on the analog pins from A0 to A3, and the other letters continue the sequence starting once again from the digital pin 10 to the analog pin A3.

Arduino Dosis Interface, has three main pre-established codes (Collins and McLean 2014), the first one is onceArduino(), which turns on the pins in Arduino only once, and the taken pins correspond to the letters typed subsequently after having written this code in the interface. The second code is loopArduino(), which turns on the pins corresponding to the letters iteratively in the order in which the message was typed in the Dosis Interface and finally the last code is pararLoopArduino(), which allows Arduino to not turn on any pins.

Next, an example case is shown where Dosis is used as a Live Coding tool to turn on led lights with Arduino, from the pin numbers written by code ion the interface.

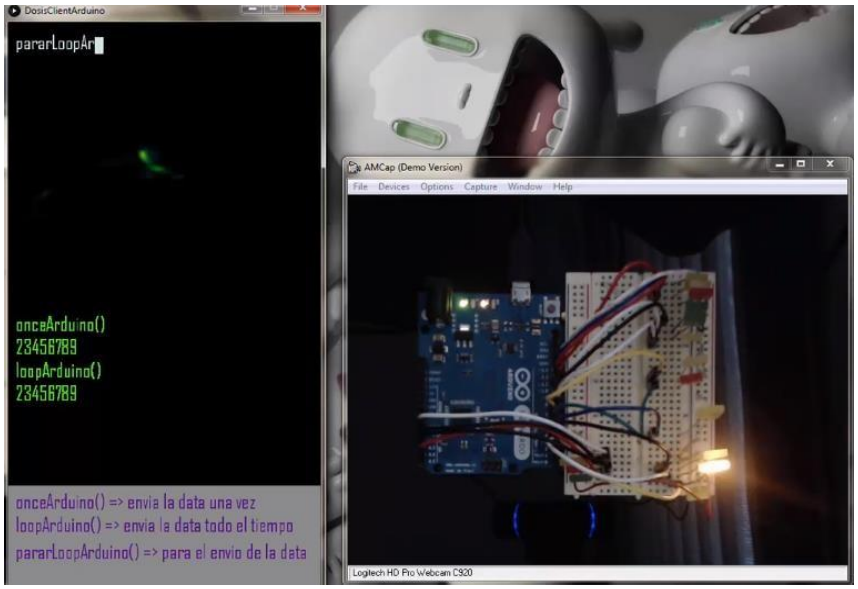


Figure 5. The letter typing on Dosis, turns on the led in Arduino. (https://vimeo.com/282589746)

Another use case, which was made as a result of a Dosis Workshop, were hardware music projects (Collins, N. 2006) where they focused on generating sound performances from hacking objects with Arduino, manipulated in real time by Dosis code, generating rhythms from the use of motors, speakers, security cameras, relay, drones, radios and typewriters among others.

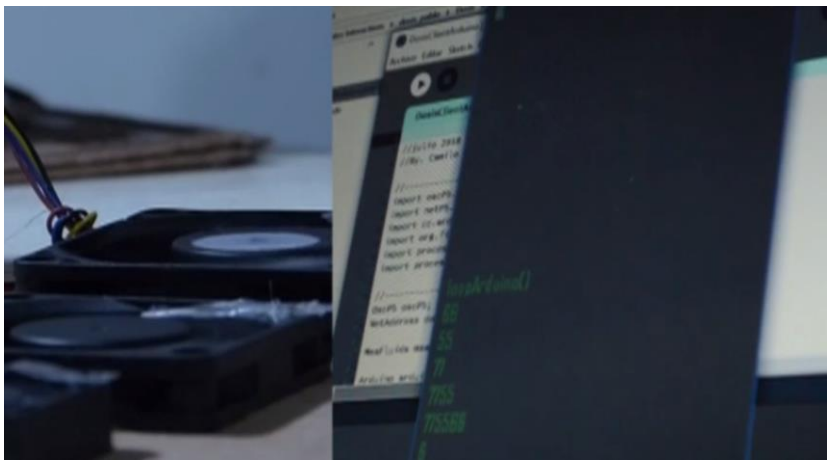


Figure 6. Dosis Hardware Hacking (https://vimeo.com/292636194)

4.1.2 Kinect 360 Interface

For the implementation of the Kinect (Zhang 2012) in the Dosis Client Interface, it was determined to use the SimpleOpenNI library (Vega 2017) for Processing, which needs the previous installation of the Kinect SDK v1.7, in order for the Windows computer to recognize the Kinect camera. The existing limitation is that the SKD drivers for the Kinect only work for Windows operating systems.

Through the library you can obtain the user's data, thus, each point of the articulations as a vector (X, Y, Z). This is how the Dosis Interface shows the depth image where the user is tracked and the data of the head, the torso, the right hand and the left hand are sent through the pre-established code, Kinect(), by which the information is transmitted by OSC message to the Dosis Server Interface to display a visual where it shows the user's head and hands movement and position.

On the other hand, this information is also sent to SuperCollider, whereby you can modify or reproduce the sound syntheses. This is how the audio engine receives four messages, the first one is the horizontal position of the right hand, the second one provides the horizontal position of the left hand, the third one, the "push" gesture generated by the user with the left hand (left hand extended in front of the Kinect camera) and the fourth message is the same gesture with the right hand.

This is evidenced in the following example, where Kinect's interaction with sound and graphics is shown, through the Dosis environment of live coding, where the movements of the body that can be used are determined from the code typed by the artist to reproduce the sounds and manipulate the graphics executed by code.

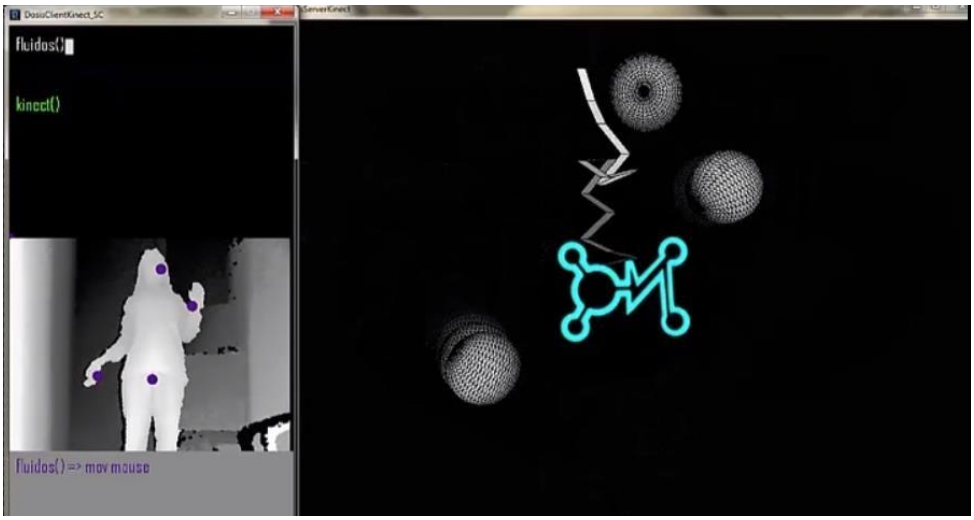


Figure 7. The Dosis Kinect Client & Server (https://vimeo.com/283835424) (https://vimeo.com/283835882)

4.1.3 Joystick Interface

To use the Joystick in the Dosis Interface, it is necessary to implement the VRPN's (Virtual Reality Peripherical Network) library (Taylor, Hudson and Seeger 2001) in Processing, with which you can use and receive input from different devices such as motion trackers (Phase Space, Flock of Birds Tracker, Data Gloves) or Joysticks controls.

The Joystick sends the coordinates in X and Y, to the computer through VRPN, with values ranging from 1 to -1, it also sends the data of the button pressed by the user.

With the previous installation of the VRPN on the computer and the implementation of the library in Processing, we have access to the Joystick with which the user can modify the sound or visuals from the Dosis Client Interface, allowing an additional tool for live coding.

The Joystick interface shows the visual feedback of the user's actions with the Joystick device, through two preestablished codes, the first is joystick(), which sends the data with the buttons' values and the lever of the Joystick. The second code is JoystickStop(), which stops sending messages from the Dosis Client Interface to the visual and audio applications.

5. DEVELOPMENT AND IMPLEMENTATION OF THE DOSIS SERVER INTERFACE

5.1 Interface Server

This application is in charge of receiving the OSC messages that the user typed in the Dosis Client Interface, by processing and converting the messages into visuals in real time. These visuals are interactive and change

from the pre-established codes, also from the location of the mouse in the client interface or from the ambient sound.

The structure of this application is object oriented programming (OOP), where a sketchbook made in Processing can be converted into a class and integrated into the Dosis Server Interface, declaring and initializing the class in the DosisServer.pde code, and in the protocol() function, the pre-established codes are determined to display the implemented visuals, when the message with the pre-established code is received.

```
//Dosis Server Interface protocol code void protocolo(String msn)
{
  String temp = "";
  String[] mensaje;   String[] mensajeDatos;   boolean mensajeValido = false;
  if(msn.substring(msn.length()-1).equals(")"))
  {
    temp = msn.substring(0,msn.length()-1);
    mensajeValido = true;
  }
```

The Dosis Server Interface had eleven visuals implemented, where each visual corresponds to a different class, but the structure allows the implementation of more visuals in order to have several options for composition in the live coding performance. Below are some examples of the type of visuals that can be done with Dosis.

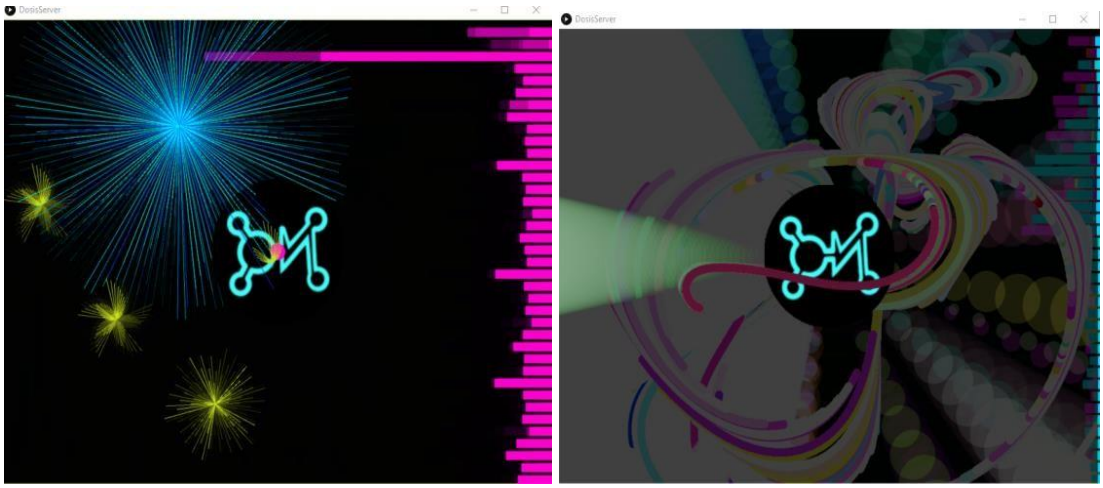


Figure 8. Visuals of Dosis server interface (https://vimeo.com/286624611)

6. CONCLUSIONS AND FUTURE WORK

This interface has been used in live coding samples as a result of the Synchronic Audiovisual Live Coding Workshop held in Bogotá - Colombia in 2018 (https://vimeo.com/278199568), where participants interacted through the Kinect device to modify the sounds in Super Collider and the visuals in Hydra, using the Dosis Interface. Dosis was used in this same sample, as a control interface for Sebastian Gómez's Machine Noise project (https://vimeo.com/278099527), where electronic writing machines were hacked with Arduino and data was sent to Arduino by Dosis interface to create rhythms through live coding.

Initially the interface allowed the integration of Kinect, then further functionalities were added for the use of other input devices such as Joysticks and output as Arduino, to use different tools to control, modify and generate audio and visuals.

The use of Processing allows an easy inclusion of any sketchbook inside the Dosis Server Interface in order to visualize any exercise in Processing, using the practice of live coding generating visuals in real time.

In performance, the Dosis interface allows collaborative productions using different computers that can be communicated by sending messages for the creation or modification of sounds and visuals through programs such as SuperCollider and Hydra, also allowing the participation of the public in the performance without the need for them to type coding but by means of the movement of the body with the Kinect, in this way an

inclusion is generated with people who wish to participate in performance but who do not know how to make a code in the audio or visual platforms.

This interface is an open source platform (https://github.com/camiloNemocon/DOSIS) and as future work it is proposed to link more interaction devices and implement more functionalities based on feedback received by users and developers.

REFERENCES

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-livewritingmusic
performance-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 Visualization 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”
Proceeding 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 http://www.nime.org/proceedings/2014/nime2014_328.pdf