UMÁTICA. Revista sobre Creación y Análisis de la Imagen
{ISSN: 2659-5354 // D.L.: MA- 1628-2018}

Produción audio-visual y código libre

[DIYSIK] de VertexZenit

Audiovisual production and open source. [DIYSIK] by VertexZenit

JOSÉ ANTONIO VERTEDOR ROMERO
Universidad de Málaga, España.

Resumen

Este trabajo aborda la creación de un proyecto artístico vinculado a la producción audiovisual mediante código abierto, [DIYSIK]. Se analiza el proceso de trabajo de una pieza desarrollada durante una beca de creación artística en la Facultad de Bellas Artes de la Universidad de Málaga (2014). Se mostrarán los recursos básicos tanto de software como de hardware utilizados en la evolución de construcción de la pieza para ofrecer un repositorio de herramientas que permiten la implementación de este proyecto o el punto de inicio para la innovación de otros diferentes.

PALABRAS CLAVE: Código Abierto; Autoaprendizaje; "Vvvv" A Multipurpose Toolkit; Pure Data; Producción Audiovisual

Artículo original
Original Article

Correspondencia
Correspondence

Jose Antonio Vertedor

vertedor.78@gmail.com

Financiación
Fundings

Sin financiación

Received: 16.06.2017
Accepted: 10.01.2018

CÓMO CITAR ESTE TRABAJO / HOW TO CITE THIS PAPER

Vertedor, J.A. (2018). Produción audio-visual y código libre[DIYSIK] de VertexZenit. Umática. Revista sobre Creación y Análisis de la Imagen , 1: 155-170.
http://dx.doi.org/10.24310/Umatica.2018.voi1.2965

Umática. 2018; 1: 155-170

Audiovisual production and open source. [DIYSIK] by VertexZenit

José Antonio Vertedor Romero
Universidad de Málaga, España.

Abstract:

This work deals with the creation of an artistic project linked to audiovisual production using open source, [DIYSIK]. It analyzes the work process of a piece developed during a scholarship of artistic creation in the Faculty of Fine Arts of the University of Malaga (2014). It will show the basic resources of both software and hardware used in the evolution of the construction of the piece to offer a repository of tools that allow the implementation of this project or the starting point for innovation of different ones.

Key Words: Open Source; Self-Learning; vvvv, a multipurpose toolkit; Pure Data; Audiovisual production

Sumario:

1. Introducción

En el presente texto voy a exponer los puntos que llevé a cabo para el desarrollo de un proyecto artístico realizado durante el año 2013-14 en el marco de una beca como artista residente en la facultad de Bellas Artes de la Universidad de Málaga, [DIYSIK] (*do it yourself sincretic work*)[1]. La motivación principal para la realización de este trabajo era simplemente la de utilizar la tecnología de una manera diferente para la que originalmente había sido pensada, es decir, mezclar diferentes recursos para construir mi propia estación de "juego" teniendo presente la importancia de una formación auto-organizada. Esta pieza fue mostrada a modo de performance audiovisual ofreciendo varias sesiones de dos horas de duración cada una donde las personas que asistían al espacio podían interactuar con el sistema expuesto en la sala. En este texto, planteamos un análisis de la construcción y puesta en escena de esta obra interactiva.

Entiendo el uso de los nuevos medios en la producción artística como un recurso que permite romper las tradicionales dinámicas jerarquizadas que establecían una diferencia entre artistas y espectadores, donde los segundos quedaban relegados al simple acto de la observación de una obra. La tecnología convierte el arte en un potente medio de comunicación implementado por la filosofía del código libre. En este sentido, la intención es la de ofrecer un trabajo a la comunidad como obra abierta, como prototipo disponible para la modificación y uso de su código. [DIYSIK] es un trabajo que genera material sonoro que en este caso ha quedado registrado en la nube de SoundCloud (VertexZenit, 2017). Del mismo modo, las imágenes que se describirán en este trabajo, están tomadas de los propios *patches* de los programas y pueden descargarse del siguiente sitio web: https://vvvv.org/contribution/diysik.

En la producción de mis distintos proyectos, resulta imprescindible trabajar en torno al concepto de prototipo, ya que se refiere a un trabajo que no necesariamente tiene que estar terminado, sino que puede formar parte de un repositorio de ideas por desarrollar que, aunque mostradas en su precariedad, tienen la capacidad de poder adaptarse a cualquier espacio y a los medios ofrecidos en cada situación. Éste es un punto importante en mi trabajo, el prototipado y testeo de las interfaces programadas y/o construidas, dejándolas semiactivas en espera de implementar su función.

Traer el recurso de la tecnología al mundo del arte es uno de los propósitos del arte contemporáneo, haciendo un acopio de recursos tecnológicos en la manera que Lévi Strauss definiría como ciencia de lo concreto o del bricolaje con la figura del *bricoleur* (Levy-Strauss, 1966), es decir, emplear los recursos tecnológicos de una manera para la que no habían sido pensadas, esto puede dar lugar a nuevas construcciones creativas-comunicativas, que en este caso se van a utilizar para conducirlas hacia el factor lúdico-festivo del arte (Gadamer, 1991). Haciendo

1. [DIYSIK] ha participado en las muestras: Work in Problem (Málaga, BBAA, 2014) y en INT16 (Málaga, Centro Cultural Provicial MVA, 2016).

uso de diferentes tecnologías *open-source* que podemos encontrar en la red de manera gratuita tanto para una computadora como para dispositivo móvil, y de los diferentes recursos existentes para la comunicación entre programas, en este caso el protocolo UDP, he podido hacer posible la implementación de esta pieza audio-visual interactiva.

El protocolo UDP (User Datagram Protocol) es un protocolo de comunicación alternativo al Protocolo de Control de Transmisión (TCP) que se utiliza principalmente para el establecimiento de conexiones de baja latencia y tolerar la pérdida de datos entre aplicaciones usadas en Internet. UDP y TCP corren en la parte superior del Protocolo de Internet (IP) y se les denomina a veces como UDP/IP o TCP/IP. Ambos protocolos envían paquetes cortos de datos llamados datagramas. UDP proporciona dos servicios no proporcionados por la capa IP: números de puerto para ayudar a distinguir las diferentes peticiones de los usuarios y, opcionalmente, una suma de comprobación capaz de verificar que los datos llegaron intactos (Rouse, 2015).

Este recurso de comunicación ha sido de gran utilidad para la comunicación de datos entre los programas utilizados en el desarrollo de este proyecto, vvvv (v4) y *Pure Data* (PD). Ambos programas tienen la peculiaridad de trabajar con código modular, v4 enfocado a diseños interactivos y PD a síntesis de sonido. v4 me permitió comunicar los datos recogidos desde Arduino y Fiducials entre las dos computadoras que componían la instalación, para proceder a la sonificación de los mismos mediante el programa PD. Los datos procesados fueron renderizados a imágenes mediante el programa v4 y sonificados mediante osciladores programados en PD. La interacción entre las dos mesas iba a repercutir sobre la proyección de gráficos vectoriales diseñados con el programa v4, que generaría movimiento gracias a un algoritmo de análisis de sonido. Este soporte evolucionó hacia un proceso que ha ido tomando distintos caminos hasta aproximarse al terreno de la *performance* audiovisual.

En la siguiente imagen (fig.1) muestro la instalación montada en el espacio expositivo (Pueden verse dos vídeos en el siguiente enlace: https://diysik.wordpress.com/2014/08/07/diysik/). Las dos mesas de control permitían la manipulación y creación del sonido generativo, éste a su vez era analizado con un programa para generar datos que podían interpretarse en una sesión *live* y trabajarlos mediante un renderizado paramétrico. La instalación hace referencia a las prácticas de *Live Coding* mediante la experimentación con la manipulación de código en vivo utilizando hardware experimental para llevar a cabo esta acción. En este sentido me gustaría señalar los casos de *Live Coding* descritos por Collins, Mclean, Rohrhuber, y Ward (2003).

Es capital en este proceso de construcción atender al concepto de *bricoleur* que describe Levy-Strauss en *The Savage Mind* (1966), en el sentido de responder a lo que se está haciendo con los recursos accesibles en ese momento, ya que fue ese uno de los principales motivos de la producción de [DIYSIK], hacer acopio de *software* libre específico y del *hardware* del que podía disponer aprovechando los recursos de los procesadores que tenían que soportar ciertos procesos en ocasiones bastante pesados. Esto provocó la necesidad de depurar el código de manera que avanzara el prototipo hasta una versión estable.

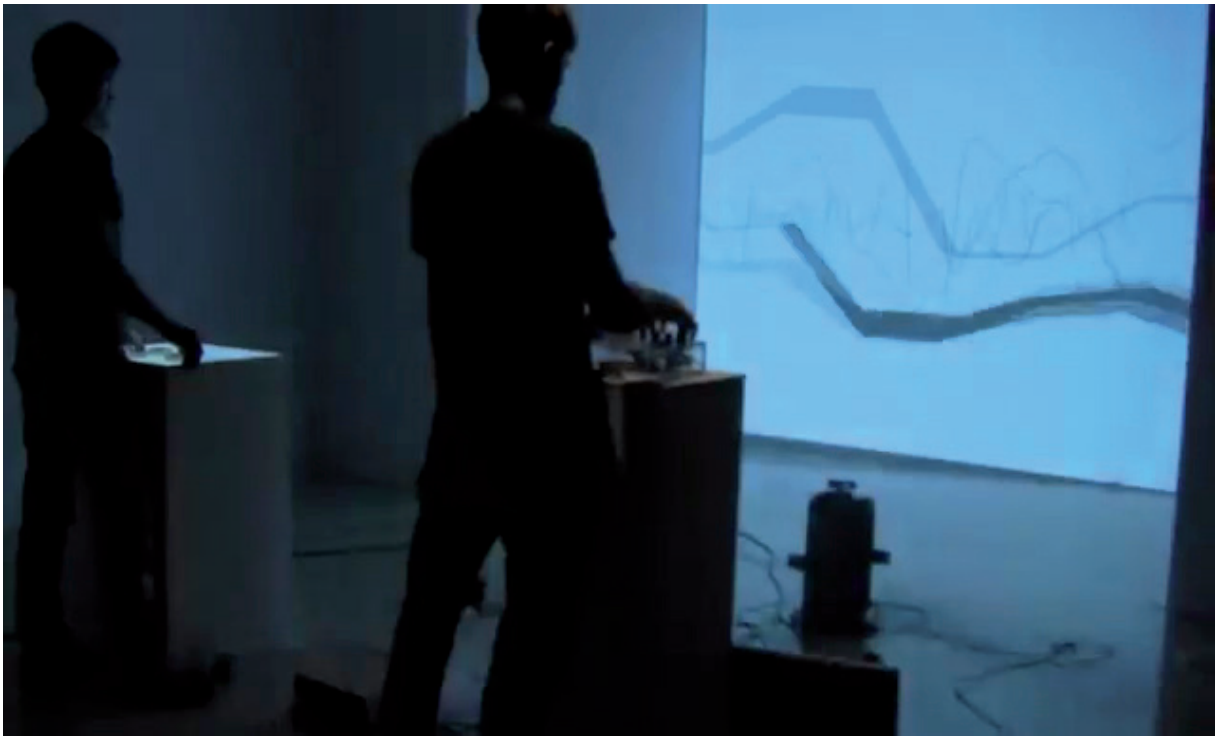


Fig.1
Vertedor. *Live Session* audiovisual con la disposición de las controladoras en sala. Exposición colectiva *Work in Problem* (Málaga: Facultad de Bellas Artes, 2014)

Fuente: Construcción y elaboración propia. (Málaga, 2014)

2. Origen y desarrollo de [DIYSIK]

La idea principal que me llevó al desarrollo de [DIYSIK] fue la de experimentar con los procesos internos de computadoras puestas en red a las que conecté algunos dispositivos para disponer de un sistema de producción audiovisual experimental en directo, con sonidos e imágenes basados en *glitches*, microsonido y *loops* principalmente. Un desarrollo de algoritmos generativos que repercutían de manera audio-reactiva en una imagen vectorial que se proyectaba en una de las paredes del espacio. De esta manera, el visitante tomaba el rol de creador audiovisual mediante el uso de estas controladoras experimentales dispuestas para la creación de estas prácticas creativas, no obstante, las máquinas presentadas guardan cierta relación con dispositivos comerciales de producción de música electrónica.

El primer paso para la construcción de este dispositivo fue la construcción de una controladora experimental desarrollada con *Arduino*. El microprocesador *Arduino* consiste en una plataforma electrónica de código abierto basado en hardware y software pensado para que cualquier persona pueda desarrollar proyectos interactivos[2]. Esto me permitía generar datos mediante unos potenciómetros y una serie de botones conectados al microprocesador *Arduino* e introducirlos en formato de datos en la computadora. Mediante este microprocesador y el programa v4, que describiré más adelante, conseguía visualizar ese flujo de datos en formato numérico. Esto supuso un reto interesante, el de sonificar y visualizar este flujo de datos mediante tecnologías de código abierto, de ahí el título, una máquina sincrética desarrollado con

2 Arduino. (Sin fecha). Arduino: What is Arduino? Recuperado de https://www.arduino.cc/

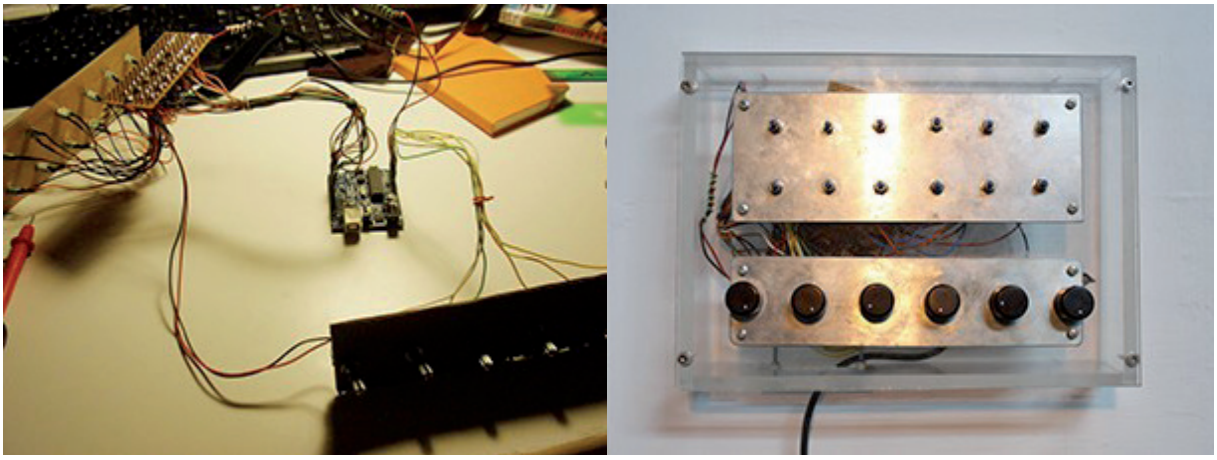


Fig.2 | Fig.3 Vertedor. Proceso de construcción de controladora desarrollada con Arduino. Fuente: Construcción y elaboración propia.

tecnología accesible que me permitiera trabajar sin que generase gastos desorbitados. Podría decir que fue ese el origen del proyecto. Las siguientes imágenes (fig.2, fig.3) muestran esta controladora en una etapa del proceso de su construcción y en el momento en el que ya estaba acabada.

A partir de la construcción de esta interfaz y de la correcta comunicación con el programa para síntesis de sonido del que también hablaré más adelante, PD, comienzo una labor de investigación, de recopilación de información y de herramientas, tanto digitales como físicas, de las que me sirvo para la construcción de esta máquina de estructura sincrética. La denomino máquina sincrética por aglutinar un conjunto de recursos tomados de diferentes proyectos de código abierto y que habían sido desarrollados para otros proyectos. Esta es una de las posibilidades interesantes del código abierto, la opción de poder recopilar y remezclar diferentes herramientas para darles un nuevo uso creativo. Entre estos recursos podemos encontrar la tecnología reacTIVision, un *framework* desarrollado como sensor primario para el instrumento electroacústico táctil, *reacTable*, que utiliza objetos fiduciarios para su control (Kaltenbrunner & Bencina, 2007).

3. Composición de la estructura sincrética

Como he apuntado antes, la construcción de esta *interface* me llevó a la investigación de recursos que me permitiesen reorganizar las diferentes herramientas que encontré para darles las funciones que iba a necesitar. Uno de los elementos principales que me ayudaron en el prototipado y desarrollo de esta máquina sincrética fue la herramienta multipropósito, vvvv, *a multipurpose toolkit*. Este programa trabaja con un código de gráficos orientados a objetos, o programación modular, lo permite a cualquier usuario con o sin conocimientos de código de texto adentrarse en el mundo de la programación creativa de una manera más visual. v4 trabaja siempre en modo *runtime*, lo que hace que el proceso de trabajo se compile cuando es necesario sin la necesidad de parar el programa para hacerlo (vvvv, sin fecha). Esta es una cualidad interesante del diseño paramétrico con este programa, ya que nos permite modificar o implementar el código en directo. Otra característica de este programa es su cualidad híbrida (fig. 4), lo

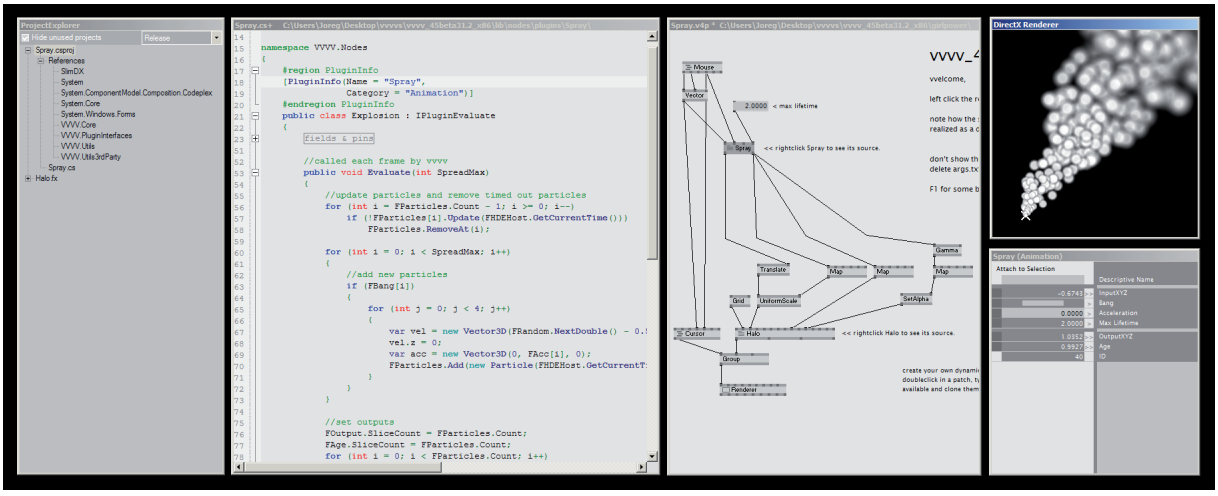


Fig.4
vvvv, entorno de desarrollo híbrido. Programación modular y código C#

que significa que debajo del código modular el programa, se alojan dos códigos de programación textual con los que se pueden conseguir resultados más complejos trabajando a un nivel más bajo con la máquina, estos son: *High-Level Shading Language* (HLSL) (Netware, sin fecha) y *C Sharp* (C#) (Wikipedia, 2017).

En la fig. 4 se muestra los dos códigos de trabajo con v4. Código de texto, a continuación, el código modular que está por encima de este texto y por último una pantalla *render* que muestra lo que el código genera[3].

Una de las principales funciones para las que se ha utilizado v4 ha sido la de crear la estructura del proyecto, ya que, además de haber realizado los diseños de imagen audio-reactiva para el trabajo que se mostraban por un proyector, su función principal ha sido la de ser el elemento con el que se han gestionado las distintas señales de entrada, esto es, las señales generadas por la controladora de *Arduino*, traducir las señales de los Fiducials captadas con una webcam, enviar datos por protocolo de comunicación UDP al programa PD para la sonificación de los mismos, también ha servido para crear una red de máquinas funcionando en espejo mediante el uso de su tecnología *Boygrouping*, que también describiré brevemente más adelante. Por último, destacar que la modelización paramétrica y variación de la imagen proyectada también se realiza con este programa, v4, mediante su analizador de sonido *fast Fourier transform* (FFT), lo que permite que el sonido modifique la imagen y al propio sonido de salida mediante la interacción con parámetros del propio código fuente.

Con respecto a los gráficos, v4 es también el programa con el que he trabajado el diseño de la parte visual de la instalación. Al usar el código HLSL, que es un lenguaje de programación para la unidad de procesamiento gráfico, hace que este programa esté dedicado principalmente a la producción visual mediante el uso de vectores, o implementado con diferentes archivos que son soportados por el programa, aunque también permite el trabajo con sonido y archivos de audio en varios formatos, aunque no han sido utilizados en este proyecto. El programa consta de diferentes módulos dedicados al modelado paramétrico en 2D o 3D.

3. Fuente: https://vvvv.org/screenshots

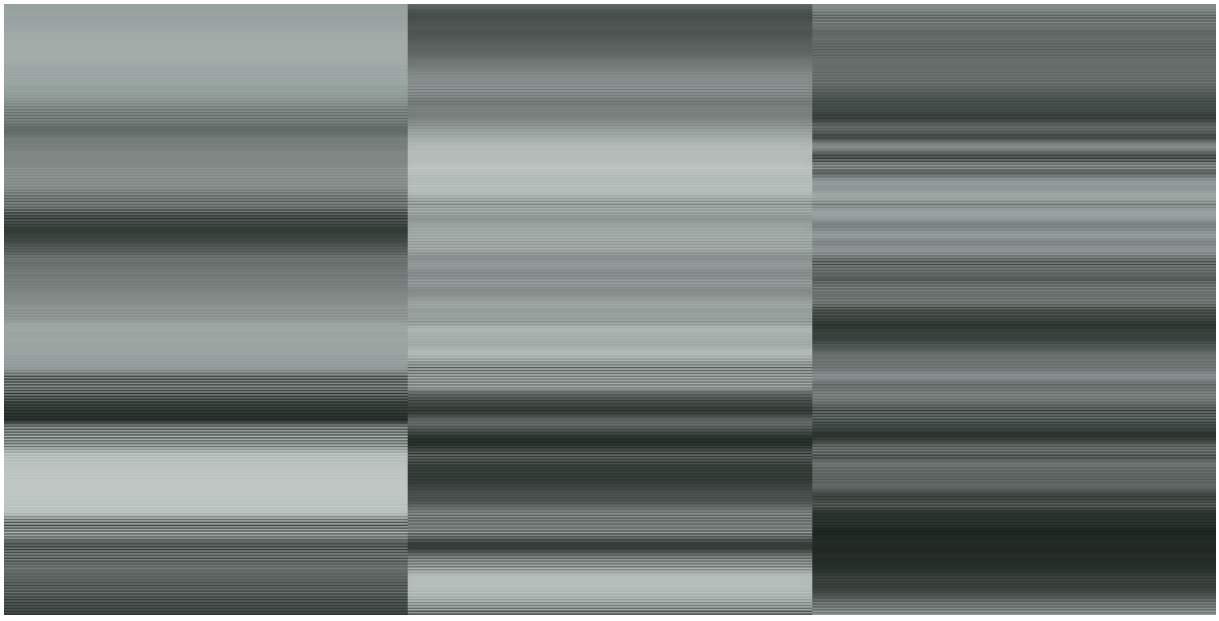


Fig.5
Fotograma tomado durante una prueba de diseño de imagen.
Fuente: Construcción y elaboración propia.

Módulos de animación, color, física, audio, matemáticas, transformación o vídeo entre otros, con los que puede trabajarse mediante la programación modular de una manera relativamente intuitiva. La siguiente imagen (fig.5) muestra un fotograma de renderizado de la imagen en movimiento modificada por el sonido generado.

En cuanto a la sonificación de estos datos de entrada, opté por el programa PD por utilizar código de programación modular similar al utilizado por v4. Esto permite, en estas herramientas de creación y síntesis de sonido, alcanzar una mayor complejidad en la creación de prototipos de audio. Un inconveniente de este programa es que resulta un poco más difícil de usar que su versión comercial Max/Msp, lo positivo es que se trata de código libre, además de ser multiplataforma, es decir, funciona en diferentes sistemas operativos. Al igual que v4, PD permite modificar parámetros y comportamientos mientras que el parche está en funcionamiento, creando un bucle de retroalimentación en tiempo real que agiliza la programación de estos prototipos de audición (Paul, sin fecha).

4. Desarrollo de la máquina sincrética

El proyecto [DIYSIK] se compone de un sistema que implica el procesado de datos a través de dos computadoras conectados en red a través de un *router* que permiten la lectura, procesado y renderizado independiente de estos datos. Por un lado, tenemos la gestión de entrada de datos y por otro lado el procesado de esta información para el renderizado audiovisual. A continuación, muestro el esquema unifilar de procesos que se ponen en funcionamiento en el proyecto [DIYSIK] así como un diagrama de flujo de la interacción que hay con el trabajo (fig.6), desde la manipulación por parte del usuario hasta la interacción de la información generada con estos datos que produce la salida, una interacción que repercute de manera

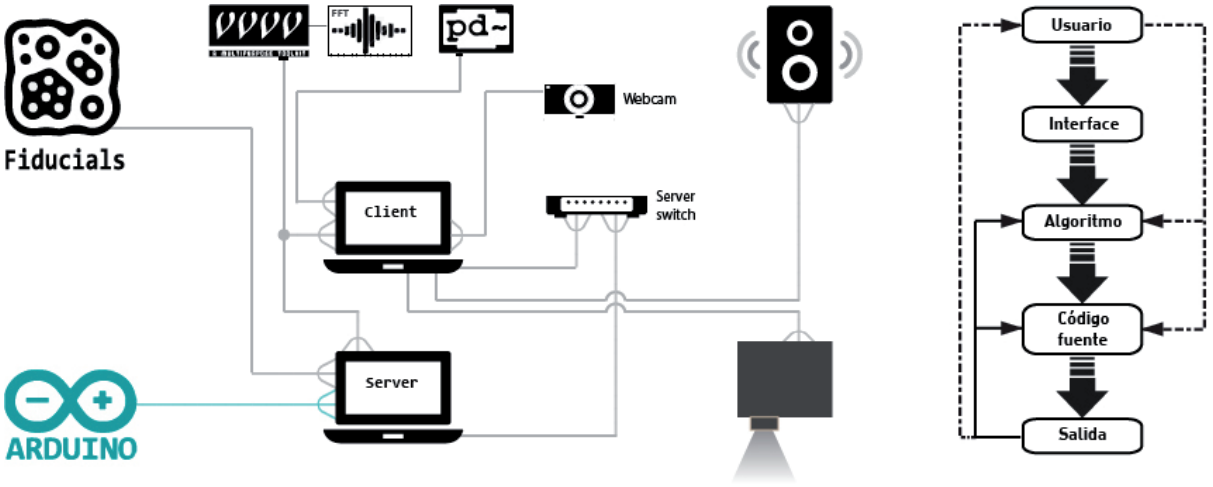


Fig.6
Esquema unifilar y diagrama de flujo de los programas y recursos que ponen en funcionamiento el proyecto [DIYSIK].
Fuente: Construcción y elaboración propia.

variada tanto en el propio algoritmo creado como en el usuario a través de los estímulos percibidos que le harán modificar los parámetros haciendo uso de los controles de la interface.

Entrando en un nivel más profundo del funcionamiento de [DIYSIK], pasaré a definir la tecnología que utiliza v4 para crear computadoras *server* y tantas *client* como se desee, esto me ha permitido repartir los procesos de las máquinas de una manera más óptima, ya que las computadoras no contaban con procesadores demasiado potentes. Me refiero al uso de la arquitectura servidor-cliente denominada *boygrouping* que le permite a v4 controlar el procesamiento de cualquier número de computadoras *render* conectadas en la red, es decir, clientes, desde un único servidor. Mientras el parcheo se realiza en el servidor, v4 se encarga de sincronizar todos los clientes, además, el parche espejo de la computadora cliente puede modificarse, con lo que me resultó muy útil este sistema para tener corriendo varios procesos a la vez de una manera sincronizada.

Para aprender estos procesos he necesitado profundizar tanto en la información ofrecida en la página como en los foros que los usuarios de v4 ofrecen para posibilitar el uso de estas herramientas. Me gustaría destacar en este punto la importancia que ha tenido el uso de internet para mi propio proceso de auto formación en cuanto a recursos que he ido descubriendo a medida que el proyecto iba creciendo y demandaba herramientas para su crecimiento coherente según mis intereses. Este texto me sirve para ver todos estos procesos expuestos de una manera más o menos ordenada y espero que sirva como referente a quien quiera adentrarse en procesos que en principio pueden ser complejos pero que sólo requieren disciplina e interés para poder aprender a desarrollarlos. Este inciso es para aquellos que piensen que sólo existe un camino a explorar y perfeccionar, porque en mi caso, yo venía del mundo de la pintura con conocimientos de electricidad, sin tener idea apenas de informática.

Me gustaría profundizar un poco más en los procesos de este proyecto con el objetivo de dejar lo suficientemente claro cuáles son los recursos que se ponen en práctica en el mismo. En el siguiente diagrama de flujo (fig.7) muestro cómo las dos principales fuentes de datos, que son *Fiducials* y la controladora construida con *Arduino*, trabajan como entradas en el programa v4 que es el encargado de gestionar los datos de entrada mediante varios proce-

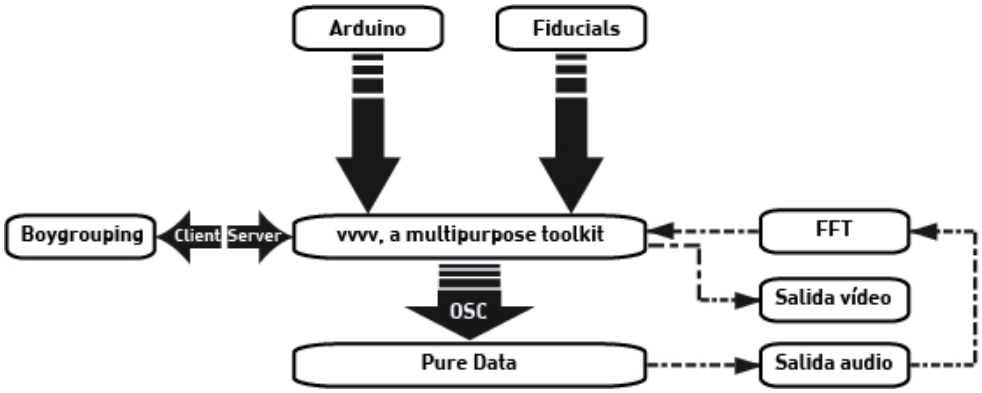


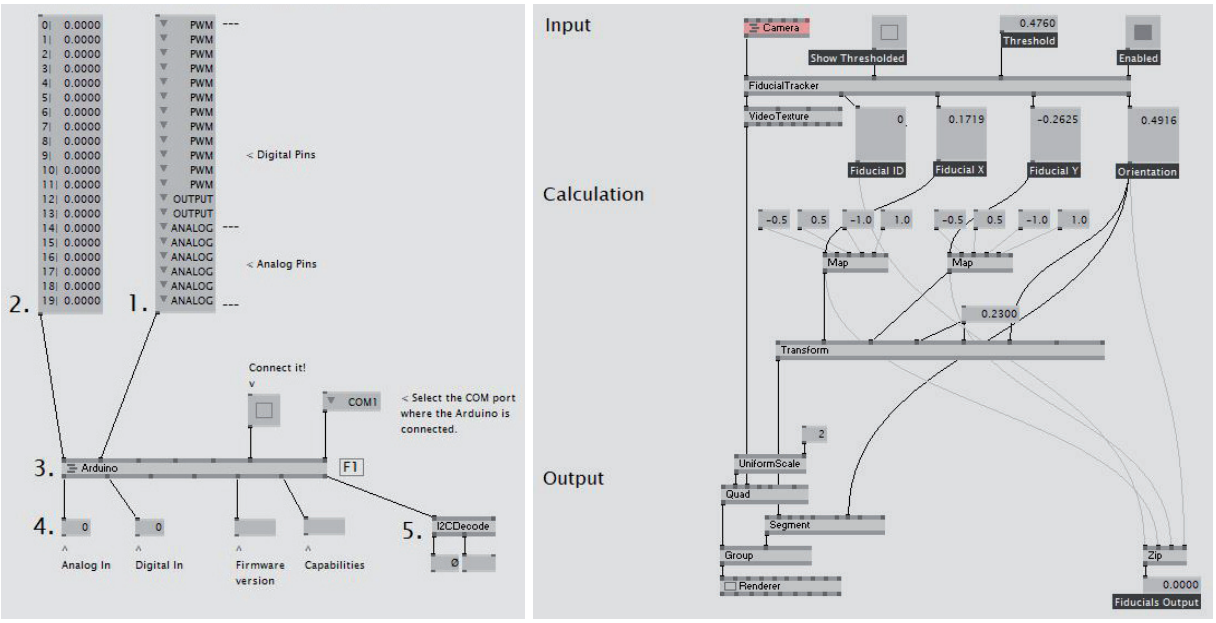
Fig.7
Diagrama de flujo de los procesos internos del algoritmo de [DIYSIK].
Fuente: Construcción y elaboración propia.

sos: el envío de datos vía UDP al programa PD, que se ejecuta en la computadora cliente para producir sonido generativo que es procesado de nuevo por v4 para generar datos y producir imagen audio reactiva en la salida de vídeo, también la conexión mediante *Boygrouping* con la otra computadora para reproducir los procesos de la máquina cliente.

Me gustaría destacar a continuación algunos de los *patches* del programa v4 y PD que me han servido para la construcción de esta máquina sincrética, el objetivo es mostrarlos de manera independiente a modo explicativo para que pueda entenderse mejor el proceso de construcción de este proyecto, ya que lo que hice fue combinar e implementar algunos de estos *patches* con otros que pueden encontrarse en los foros de la comunidad de v4 para la elaboración de [DIYSIK]. En primer lugar, muestro la comunicación del microprocesador Arduino con la computadora mediante este *patch* (fig.8). Para esto instalé a la placa un protocolo genérico para la comunicación con microcontroladores desde un programa instalado en una computadora, *firmata* (GitHub, sin fecha), de esta manera se puede tomar el control de Arduino utilizando cualquier programa, en este caso utilicé v4.

La otra fuente de entrada de datos al sistema estaba producida por los fiducials. La siguiente captura de pantalla (fig.9) muestra el *patch* utilizado para esta captura de datos. Mediante una cámara web conectada a la computadora este *patch* permite la lectura de las figuras fiduciarias para obtener sus datos de orientación y de giro X/Y. De esta manera programé los distintos comportamientos que iban a tener las diferentes figuras fiduciarias que utilicé. Tanto la entrada de datos de Arduino como la de fiducials eran leídas por v4 antes de pasar a procesarlas por el propio programa v4, una vez procesados serían convertidos en imagen en movimiento o enviadas mediante el protocolo UDP al programa PD para la sonificación de los mismos.

Las dos imágenes de arriba (fig.7 y fig.8) muestran la comunicación mediante puertos UDP, a la izquierda la interface de PD y a la derecha la de v4. Puede verse en la imagen cómo los datos enviados por v4 a través del puerto 4001 son recibidos por PD mediante el módulo *netreceive* y que ocurre lo mismo con los datos enviados por PD por el puerto 3001 hacia v4.



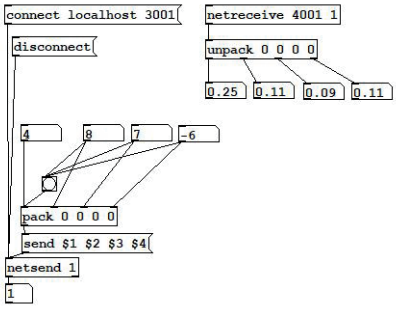
Este método me sirvió para sonificar los datos recibidos en v4 desde Arduino y los fiducials. Este protocolo es ideal para gestionar la red aplicaciones en las que la recepción de latencia es crítica, como ocurre en las comunicaciones de juegos, voz y video, que pueden sufrir alguna pérdida de datos sin afectar negativamente a la calidad percibida. En algunos casos, las técnicas de corrección de errores se utilizan para mejorar la calidad de audio y video, a pesar de una cierta pérdida.

Como último recurso voy a hacer referencia al análisis de audio mediante el analizador de sonido de v4 que utiliza el algoritmo FFT. El análisis de sonido mediante este algoritmo permite fragmentar la onda de sonido en 256 puntos que van desde el sonido grave hasta el agudo, además, mediante v4 se pueden obtener los golpes de un sonido, por todo esto, el análisis de sonido en v4 ha resultado una herramienta fundamental para aplicar movimiento a la imagen y generar un efecto sinestésico al hacer que la entrada de sonido modifique el comportamiento de la animación proyectada.

El uso de estos módulos permite comenzar a construir la máquina sincrética [DIYSIK] desde cero, claro está que las fuentes de datos introducidas a la computadora podrían ser cambiadas por otras, yo opté por éstas por el interés que tenía en ese momento por construir una máquina audio-visual generativa, aunque lo interesante de tener el código fuente es que siempre puede implementarse o hacerse más simple según las necesidades de la pieza a construir. La forma que se le dé a los datos tanto en imagen como en sonido va a depender del código empleado en estos dos programas, lo que aporto con este texto son recursos básicos para la construcción de mi proyecto que pienso que pueden servir a otras personas que estén interesadas en el trabajo con computación.

Fig.8
Patch del programa v4 para la gestión del microcontrolador Arduino mediante firmata. Captura de pantalla del patch de v4 para Arduino.
Fuente: Construcción y elaboración propia.

Fig.9
Patch del programa v4 para la gestión de datos generados por los fiducials. Captura de pantalla del patch FiducialTracker.
Fuente: Construcción y elaboración propia.



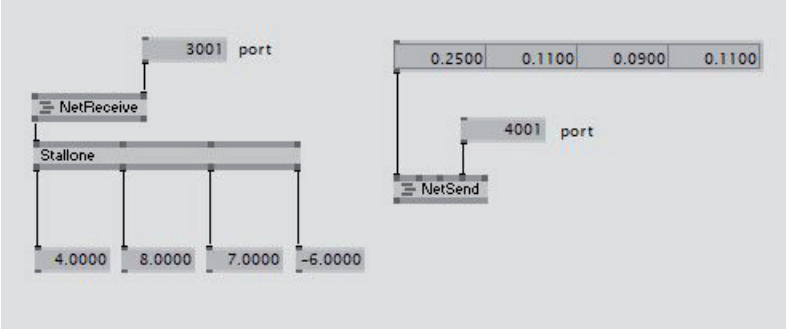


Fig.9 | Fig.10 Comunicación mediante protocolo de comunicación UDP entre PD (izquierda) y v4 (derecha). Captura de pantalla de los patches de v4 y PD.

Fuente: Construcción y elaboración propia.

Conclusiones

Con este proyecto y con la redacción de este artículo, he conseguido hacer acopio de un repositorio de herramientas que ofrecen la posibilidad de iniciar la elaboración de una "máquina sincrética" a cualquier usuario. También debo destacar que la bibliografía aportada contribuye a la comprensión de la conceptualización del algoritmo diseñado en [DIYSIK]. Analizar [DIYSIK] me ha permitido tomar consciencia de la complejidad utilizada en su construcción y ver nuevas vías de simplificación y extensión de la misma. [DIYSIK] me ha aportado una visión artística basada en la idea de prototipo en la que una obra permanece siempre abierta para poder ser re-configurada, es decir, al compartir los archivos que componen a la pieza, ésta podrá ser implementada con las diferentes aportaciones de quien la intervenga, redimensionando el concepto artístico hacia nuevos modos de entender el propio proceso creativo como un espacio común.

Uno de los puntos que más me interesa destacar de mi proceso de autoaprendizaje durante la construcción de [DIYSIK], es haber podido experimentar con la posibilidad de mutación del propio proceso de desarrollo que ofrece el trabajo generativo con programación, lo que permite varias cosas, por ejemplo, experimentar diferentes lecturas del espacio modificando el contenido del código o simplemente hacer un uso más creativo de la tecnología de *software* y de *hardware*. Este segundo punto me parece interesante ya que se podría llevar al terreno de los videojuegos serios, o gamificación, y encontrarle un sentido en el terreno de la educación.

Esta experimentación con el espacio podría implementarse en este trabajo haciendo uso por ejemplo de sistemas de sonido *surround*, esto permitiría trabajar el sonido en 3D, haciendo más rica la experiencia del sonido y dando juego a nuevas posibilidades, de entre las que me gustaría destacar la estrategia de redirección localizada del sonido a diferentes puntos. Incluir las nuevas tecnologías en el terreno artístico posibilita al espectador involucrarse de una manera más directa con la tecnología y con la propia obra. Esta involucración del espectador la he llevado a cabo en este trabajo mediante la construcción de sintetizadores a los que he dado distintos usos, como por ejemplo la aplicación para la construcción de espacios sensoriales donde sonido-imagen-presencia son el nudo principal sobre el que se trabaja, es

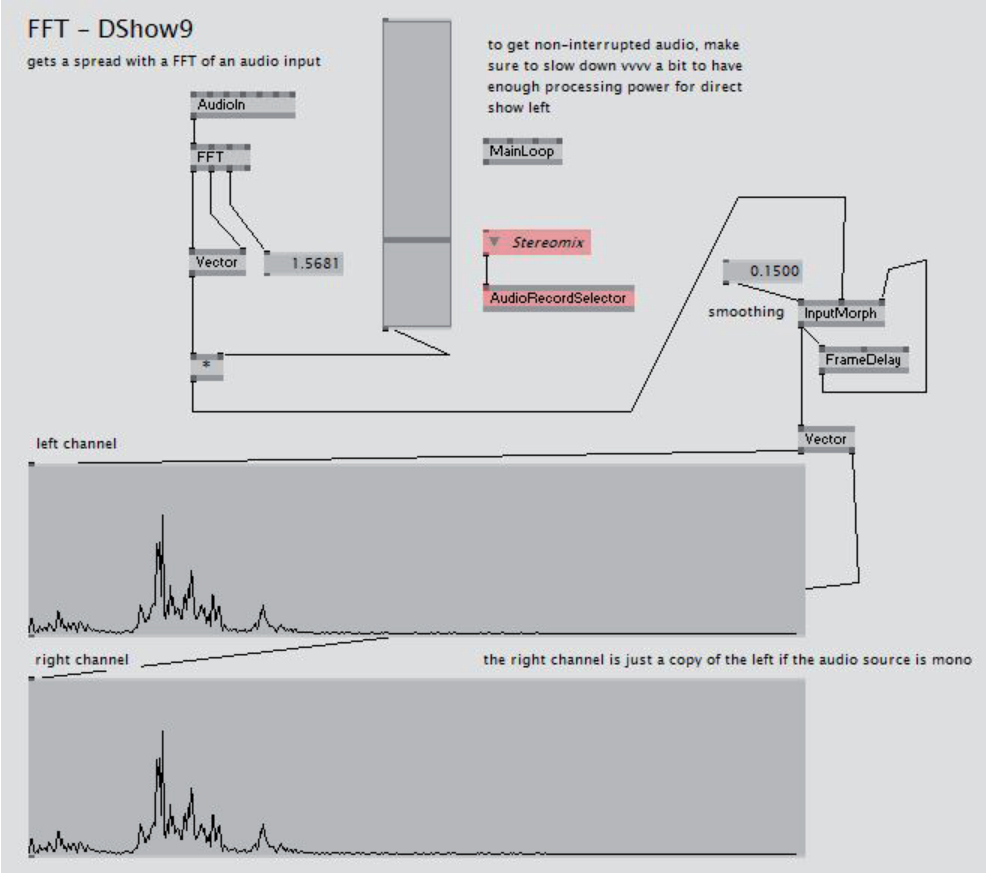


Fig.11
Módulo de análisis de audio mediante el analizador de sonido FFT. Captura de pantalla del patch FFT-DShow9 de v4.
Fuente: Construcción y elaboración propia.

decir, con el proyecto [DIYSIK] he conseguido generar "obra plástica sincrética" basada en imagen (estática o en movimiento), sonido e interacción con el espectador-actor.

He podido conocer un poco más a fondo con este trabajo diferentes tecnologías de protocolos de comunicación como es el caso como del UDP, utilizado en esta ocasión, o el protocolo OSC (*Open Sound Control*). El protocolo OSC se utiliza para la comunicación entre computadoras, sintetizadores de sonido y otros dispositivos multimedia optimizados para la tecnología de redes. El protocolo OSC es simple pero potente, ofrece todo lo necesario para el control en tiempo real en el procesamiento de sonido y otros medios de comunicación sin dejar de ser flexible y fácil de implementar. Incluye, entre otras particularidades, la interoperabilidad, la precisión, flexibilidad y una mejor organización (Opensoundcontrol, sin fecha).

El protocolo OSC me ofrece una nueva línea de investigación a explorar mediante la introducción en el proyecto de aplicaciones de control, desarrolladas para para Android o IOS, de emisión de datos por protocolo de comunicación OSC. Estas aplicaciones permiten, a través de una red WIFI, controlar desde la computadora el flujo de paquetes de datos que la aplicación genera en el dispositivo móvil, con esto cualquier usuario podría controlar con su propio dispositivo tanto el audio como el vídeo, pasando de esta manera a adoptar la figura de *performer*. Otras posibilidades de control pasarían por utilizar el propio movimiento del usuario para generar el flujo de datos de entrada haciendo uso de una cámara *Kinect*, con la

que se puede recoger una imagen de vídeo 3D del espacio. Esto abre diferentes posibilidades que fusionan expresión corporal con la creación audiovisual.

Por último, quiero señalar la aparición de dos conceptos en esta obra que son relevantes para mi trabajo posterior: la visualización de sonido y sonificación de flujo de datos. Me interesan especialmente por ser dos líneas de investigación sobre las que puede trabajarse en la actualidad ya que existe un amplio número de artistas que están avanzando continuamente con la tecnología de *software* y *hardware* en la dirección de estos conceptos. En este sentido, me gustaría señalar especialmente al artista alemán Alva Noto, interesado en la visualización de ondas sonido y el japonés Ryoji Ikeda, implicado en la sonificación de grandes masas de datos. Ambos artistas ofrecen nuevas maneras de entender el sonido y la imagen desde su respectivo opuesto.

Bibliografía / References

Arduino. (Sin fecha). Arduino: What is Arduino? Recuperado el 20 de febrero de 2017 de https://www.arduino.cc/

Collins, N., Mclean, A., Rohrhuber, J., & Ward, A. (2003). Live coding in laptop performance.
Organised Sound, 8(3), 321–329.

Gadamer, H. (1991). La actualidad de lo bello: El arte como juego, símbolo y fiesta. Barcelona, Paidos Ibérica.

GitHub. (Sin fecha). GitHub: Firmata firmware for Arduino.
Recuperado de https://github.com/firmata/arduino#firmata-client-libraries

Kaltenbrunner, M., & Bencina, R. (2007). reacTIVision: A Computer-Vision Framework for Table- Based Tangible Interaction. En Proceedings of the first international conference on "Tangible and Embedded Interaction (TEI07). (pp. 15–17). Baton Rouge, Louisiana.

Lévi-Strauss, C. (1966). The savage mind. Nature of Human Society. Chicago, The University Of Chicago Press.

Neatware. (Sin fecha). Neatware: HLSL Introduction.
Recuperado el 5 de marzo de 2017 de http://www.neatware.com/lbstudio/web/hlsl.html

Opensoundcontrol. (Sin fecha). Opensoundcontrol: Introduction to OSC. Recuperado el 20 de febrero de 2017 de http://opensoundcontrol.org/introduction-osc

Paul, L. (Sin fecha). Gamasutra: Audio Prototyping with Pure Data. Recuperado el 15 de marzo de 2017 de http://www.gamasutra.com/view/feature/131258/audio_prototyping_with_pure_data.php?print=1

Pure Data. (Sin fecha). Pure Data: Pd Community Site.
Recuperado el 12 de abril de 2017 de https://puredata.info/

Rouse, M. (mayo de 2015). Searchmicroservices: UDP (User Datagram Protocol). Recuperado de http://searchmicroservices.techtarget.com/definition/UDP-User-Datagram-Protocol

Smith, S. (Sin fecha). Capítulo 12: The Fast Fourier Transfotm. How the FFT works. En The Scientist and Engineer's Guide to Digital Signal Proccesing.
Recuperado el 12 de abril de 2017 de http://www.dspguide.com/ch12/2.htm

VertexZenit. (2017). SoundCloud: [DIYSIK].
Recuperado el 5 de marzo de 2017 de de https://soundcloud.com/vertexzenit/sets/diysik

VertexZenit (2015). vvvv: Audioanalysis-videomapping.
Recuperadoel 15 de noviembre de 2017 de https://vvvv.org/contribution/audioanalysis-videomapping

vvvv - a multipurpose toolkit | vvvv. (sin fecha). Recuperado el 10 de septiembre de 2016 de https://vvvv.org/

Wikipedia. (sin fecha). Wikipedia: C Sharp (programming language).
Recuperado el 29 de abril de 2017 de https://en.wikipedia.org/wiki/C_Sharp_%28programming_language%29