PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

Escuela de Ingeniería Eléctrica
PONTIFICIA UNIVERSIDAD CATOLICA DE VALPARAISO

Cristian Ignacio Tapia Macaya

Sistema de reconocimiento de gestos mediante Kinect para interpretación de obras musicales

Informe Proyecto de Título de Ingeniero Civil Electrónico

Escuela de Ingeniería Eléctrica
Facultad de Ingeniería

Valparaíso, 17 de agosto de 2017

Sistema de reconocimiento de gestos mediante Kinect para interpretación de obras musicales

Cristian Ignacio Tapia Macaya

Informe Final para optar al título de Ingeniero Civil Electrónico,

aprobada por la comisión de la

Escuela de Ingeniería Eléctrica de la

Facultad de Ingeniería de la

Pontificia Universidad Católica de Valparaíso

conformada por

Sr. Francisco Pizarro Torres

Profesor Guía

Sr. Andrés Rivera Fernández

Segundo Revisor

Sr. Sebastián Fingerhuth Massmann

Secretario Académico

Valparaíso, 17 de agosto de 2017

Dedicado a mi familia y amigos.

Resumen

El documento que se presenta a continuación, muestra la creación y propuesta de un proyecto orientado para usuarios que les interese la idea de transformar su cuerpo en una interfaz que envíe datos a través del protocolo MIDI; todo esto, mediante la utilización de Kinect. Sintetizando el concepto general de la idea: primero, se desea cargar en el ordenador una pista de audio en formato .wav para luego poder controlarle los parámetros de play, stop, control de volumen y filtro mediante el movimiento de los brazos. Esta tarea se lleva a cabo a través de la utilización de cuatro softwares que trabajan simultáneamente (Kinectar, Pure Data, loopMIDI y OSCeleton). Es por ello, que se trabajará en la creación de una interfaz gráfica de usuario (GUI) que ayudará a reducir la complejidad de manipular cuatro softwares por separados, uniéndolos a todos dentro de una sola plataforma. La finalidad de este proyecto es facilitar al usuario el manejo de estos programas para que, quién desee probar y crear sus propios proyectos musicales, pueda hacerlo sin necesidad de tener conocimiento sobre estos softwares.

A medida que se avance en la lectura, se podrán verificar los antecedentes generales de la problemática de trabajar con interfaces en el procesamiento de contenido musical en vivo. Luego, se revisarán y evaluarán proyectos ya desarrollados en años anteriores para tomar una referencia de cómo empezar a abordar la problemática. Posteriormente, se elaborará un marco teórico especificando todo lo necesario para que el ordenador (Windows 7 en este caso) pueda leer correctamente a Kinect. A continuación, se presentará el desarrollo del proyecto mediante la definición de los softwares que se utilizarán para la creación de la GUI y su respectiva puesta en marcha. Finalmente, se presentarán los resultados de lo obtenido, mostrando cómo instalar todo lo necesario para que el sistema pueda ser ejecutado sin mayores problemas. Todo esto será visto con mayor detalle en los capítulos siguientes.

Se espera que este documento pueda ser leído como un instructivo que vaya guiando al lector sobre cómo operar la plataforma, de manera de reducir los inconvenientes que se tengan al interactuar por primera vez con esta interfaz.

Palabras claves: Protocolo MIDI, Kinect, parámetros, softwares, interfaz gráfica de usuario (GUI).

Abstract

The document presented below shows the creation and proposal of a project oriented to users who are interested in transforming their body into an interface that sends data through the MIDI protocol; all this, by using Kinect. Synthesizing the general concept of the idea: first, it is desired to load an audio track in .wav format so then it can be controled by the parameters of play, stop, volume control and filter by moving the arms. This task is carried out through the use of four software's that work simultaneously (Kinectar, Pure Data, loopMIDI and OSCeleton). That is why it will be worked on the creation of a graphical user interface (GUI) that will help to reduce the complexity of manipulating four separate software, joining them all within a single platform. The purpose of this project is to facilitate the user to manage these programs so that whoever wants to try and create their own musical projects, can do it without having to know about these software.

As you progress in reading, you can check the general background of the problem of working with interfaces in the processing of live music content. Then, will be reviewed and evaluated projects already developed in previous years to take a reference on how to start addressing the problem. Subsequently, a theoretical framework will be elaborated specifying everything necessary so that the computer (Windows 7 in this case) can read Kinect correctly. Next, the development of the project will be presented by defining the software that will be used for the creation of the GUI and its respective implementation. Finally, the results will be presented, showing how to install everything necessary so that the system can be executed without major problems. All this will be seen in more detail in the following chapters.

It is hoped that this document can be read as an instruction that guides the reader on how to operate the platform, in order to reduce the inconveniences that are had when interacting with this interface for the first time.

Keywords: MIDI protocol, Kinect, parameters, software, graphical user interface (GUI).

Índice general

Introducción

La tecnología va cambiando y evolucionando cada vez más. La forma que interactuamos con ella nos indica que siempre es posible crear algo nuevo, que sea innovador y que esté a disposición del público para solucionar toda clase de problemáticas. Si nos remontamos a 10 años atrás, RIM (Research In Motion Limited) estaba empezando a testear las capacidades de teléfonos móviles de su línea BlackBerry. Ahora tenemos móviles con pantalla táctil capaces de brindarnos una completa experiencia informática en la palma de nuestras manos. Es fácil imaginar que lo que nos depara el futuro probablemente sea una interacción hombre/máquina sin necesidad de plataformas táctiles. Es a esto lo que apunta Kinect, generar una nueva interfaz que logra conectar al humano con la tecnología de una forma similar a recursos táctiles, pero sin el uso de ellos.

Microsoft, empresa especializada en el sector de la informática, fue fundada en Estados Unidos en el año 1975 por Bill Gates, Steven Ballmer y Paul Allen. Esta empresa ha revolucionado el mundo de la informática con sus softwares, de los que destacan los sistemas operativos Windows, Microsoft Office y Windows Live. A finales del año 2001 sale a la venta en Estados Unidos Xbox, la primera videoconsola de sobremesa producida por Microsoft, en colaboración con Intel y Nvidia. Esta consola fue la manera de competir con Sony (PlayStation 2) y Nintendo (GameCube) en el campo de los videojuegos. Posteriormente Microsoft empieza a trabajar en su sucesor: Xbox 360, que no sería lanzada hasta finales del año 2005. Pero no fue hasta el año 2009 que dio un golpe en una de las ferias de videojuegos más importantes del año, la E3[1]. En esa instancia presentó su "Natal Project", que prometía controlar los menús con el cuerpo y voz, escanear la ropa y tener un sistema de reconocimiento facial. Es decir, la idea de una IA[2] mucho más elaborada y que a su vez pueda interactuar con el usuario.

A finales del 2010 llega el tan esperado Kinect que, pese a mostrar problemas por la necesidad de tener un gran espacio para jugar, sorprende a todos por no requerir mandos y además por la calidad del reconocimiento de movimientos. Con esto viene la apertura de su SDK (System Development Kit), a principios del 2011, capaz de ayudar a la creación de nuevos proyectos innovadores mediante la conexión Kinect con Windows, influyendo en la simplificación para

[1] Feria más importante del mundo elaborada para computadores, videojuegos, móviles y productos relacionados con tecnología.

[2] Inteligencia Artificial. Área multidisciplinaria que estudia la creación y diseño de sistemas capaces de resolver problemas cotidianos por sí mismos.

comenzar nuevos proyectos. Con esto viene una inusual liberación de Microsoft: “Una versión beta gratuita para aplicaciones no comerciales". En otras palabras, fue una creación para los aficionados, científicos e inventores, en lugar de socios de la industria, desarrolladores de software y fabricantes de equipos originales [1].

Antiguamente no era fácil desarrollar melodías digitales, puesto que no existía la tecnología que hoy se maneja. Sin embargo, esta tarea se encuentra potenciada en la actualidad, ya que como bien se sabe existen variadas herramientas computacionales que permiten al usuario elaborar cómodamente contenido musical. Como bien se sabe, la música se presenta en vivo y para ello se necesita de manera obligatoria de una interfaz para procesarla. Esta interfaz cumple la función de enviar y recibir datos MIDI desde y hacia el ordenador. Casi todas funcionan mediante el protocolo USB y son compatibles con los sistemas operativos más usados. Sin embargo, el desarrollo musical mantiene un patrón: el uso particularmente de OS X, desarrollado por Apple, que brinda una mayor comodidad al usuario, ¿Por qué?

En los años 80 Atari lanzó un ordenador al mercado, el ST. Tuvo mucho éxito en Europa y en Estados Unidos. Este ordenador se diferenció fuertemente con respecto a la competencia (Commodore Amiga[3]) porque tenía incorporado el puerto MIDI (IN y OUT), por lo que rápidamente fue adoptado por muchos músicos y otros profesionales de la música. Su principal característica era que usaba un sistema operativo propio y el mismo tipo de chip que en esos tiempos utilizaba Apple (Motorola de la serie 68000). De esta manera, cuando Atari dejó de existir en el mercado, trasladarse hacia Apple era lo más fácil y lógico para los consumidores, puesto que ya tenían la experiencia de trabajar con un sistema operativo similar. Además, se ahorraba el esfuerzo de aprender a ocupar otro tipo de entorno, es por eso que el mercado musical derivó fuertemente hacia Apple [2]. Por otro lado, si se compara MS-DOS –el sistema operativo de Microsoft en los años 80– con el de Apple, en esa misma época, este último era más amigable y fácil de entender por los músicos.

Más tarde aparecieron compañías como Opcode o Digidesign, que al principio sólo desarrollaban productos para Apple. Cabe mencionar que el primer sistema de grabación digital fiable basado en ordenadores fue Sound Tools, desarrollado por Digidesign en 1989; y no fue hasta después de 1995, cuando Avid (compañía de video que también trabajaba sólo sobre Mac) compró a Digidesign, y a raíz de una inyección económica por parte de Microsoft se empezaron a desarrollar softwares musicales para Windows. No obstante, a esas alturas, quienes estaban involucrados en el sector profesional del audio no podían perder el tiempo intentando aprender de una máquina con este último sistema operativo, que aún permanecía inestable.

A pesar de esto, en la actualidad es indiferente utilizar cualquiera de los dos sistemas operativos, ya que ambos cuentan con la mejor de las tecnologías. No hay absolutamente ninguna razón por la que no se pueda hacer música en un computador con Windows. Después de todo, los componentes electrónicos en el interior de la máquina son, a menudo, idénticos. Es más, en el

[3] Fue un ordenador personal que debido a sus extraordinarias capacidades multimedia, cosechó grandes éxitos en las décadas de los años 80 y 90.

proyecto que se presenta para la definición de este informe, se controla contenido musical a través del sistema operativo Windows 7.

Objetivo general

- Modelar e implementar un sistema de detección y reconocimiento de gestos del cuerpo humano (manos, cabeza, entre otros) para manipular virtualmente contenido musical.

Objetivos específicos

- Realizar un análisis bibliográfico y el estado del arte de algoritmos de clasificación y detección usados mediante Kinect (controlar música, reproducción, etc.).

- Estudiar el SDK de Kinect y las ventajas de su utilización.

1 Antecedentes generales

1.1 Descripción del problema

Para procesar sonido en vivo, se requiere de una interfaz obligatoriamente. Existen múltiples interfaces comerciales para el control de procesos y efectos para la interpretación musical en vivo. En general, se usan mouse y teclado o un controlador MIDI manejando un software en el ordenador, así como pedaleras de efectos, que son cadenas de efectos de sonido conectadas unas a otras y manipuladas con los pies. Si bien estas interfaces son eficientes, no funcionan como instrumentos (en el sentido escénico).

Cuando se está trabajando en el computador produciendo sonidos, el hecho de arrastrar el mouse y hacer clics ocasiona que el elemento escénico se vea disminuido o eliminado. Además no existe nada de la gestualidad corporal del músico que tenga una relación directa con el sonido que se escucha. Es decir, se puede hacer un sonido muy grave y potente, sin embargo, lo único que logra apreciar el público es el sonido que fue causado por un clic o el movimiento de una perilla en un controlador (teclado MIDI, por ejemplo). Es por esto que se desea construir un "instrumento" que trabaje como interfaz y permita editar y procesar sonido en tiempo real a partir del reconocimiento de gestos de las manos y el cuerpo, pensado para la interpretación en vivo. Para esta primera etapa sólo se busca que pueda editar y procesar sonidos pre-grabados (archivos de audio). La idea de esta interfaz es que sea un aporte a la performance de la música electrónica y electroacústica en vivo, consiguiendo que la gestualidad del músico convierta lo escénico en algo más novedoso y útil a la vez, y que logre generar una coherencia entre el sonido y el despliegue corporal en el escenario [3].

Kinect conformaría la interfaz, con la cual el músico podría incluso estar tocando un instrumento en vivo, mientras usa algunos gestos del cuerpo para controlar el procesamiento de sonidos en el computador. El aporte de este proyecto pondría al alcance de los músicos una tecnología que ya está bastante difundida y que cada vez se torna más accesible al público, también resolvería sus necesidades de interpretación de música electroacústica y electrónica en vivo.

1.2 Estado del arte

En la actualidad se pueden encontrar una gran variedad de trabajos interesantes relacionados con la música y la tecnología. Algunos desarrollados con la interfaz de Kinect mediante el reconocimiento de gestos del cuerpo. Estos son:

1.2.1 Ethno Tekh

Es un equipo interdisciplinario formado por Chris Vik, Brad Hammond y Stephen Burns con sede en Melbourne, Australia, y que se centra en romper barreras entre el mundo humano y el digital; dedicando todos sus esfuerzos en la creación de nuevos métodos de expresión e interacción.

Sus habilidades están puestas en la tecnología, arte y música, una combinación que hace que trabajen en prácticas artísticas, entretenimiento, salud y educación. Han desarrollado actuaciones de movimiento controlado, instalaciones, guantes de datos inalámbricos, software de simulación de movimiento secuenciado, un sintetizador visual así como sus propias plataformas para sistemas de interacción.

El equipo cuenta con más de siete años de experiencia en el uso de dispositivos de captura de movimientos para el desarrollo de aplicaciones creativas. Además han participado mostrando sus trabajos en festivales, galerías de arte, espacios públicos y discotecas en Estados Unidos, Europa y en todo Australia [4].

Este equipo ha desarrollado dos módulos interactivos relacionados con Kinect:

Connections

Fue la primera iteración del sistema de música interactiva de Ethno Tekh, lanzada en el año 2013. Su temática fue la de un viaje audio visual a través de un mundo evolutivo en la superficie de un universo abstracto. En este trabajo los usuarios podían interactuar usando los movimientos del cuerpo para generar diferentes tipos de sonidos que se contrastaban con la música de fondo, y que venían incluidos el set desarrollado. En la Figura 1-1 se muestra una serie de imágenes del proyecto desarrollado por el equipo de Ethno Tekh y que fue mostrado al público por primera vez en el año 2013.

Figura 1-1: Connections mostrado en CeBit, 2013[4] (fuente: https://www.youtube.com/)

[4] Feria más importante del mundo para la exposición de computadores, tecnologías de la información, telecomunicaciones, software y servicios.

Gravitate

Siguiendo los pasos de Connections, Gravitate es un sistema interactivo de música completamente dinámico que también utiliza sensores de Kinect para controlar los movimientos del usuario. Las diferencias y mejoras del anterior modelo son: primero, que la interacción se ve reforzada a través de plataformas que vibran cuando se ejecutan notas, con luces que coinciden con el movimiento del usuario. Lo segundo es que ya no es un usuario, sino dos quienes interactúan con la interfaz al mismo tiempo.

Con este proyecto es posible realizar sonidos de instrumentos de percusión, además de los sonidos del trabajo anterior de Ethno Tekh. En la Figura 1-2 se exponen imágenes que clarifican la línea programática de Gravitate mediante la prueba dirigida al público en el año 2014.

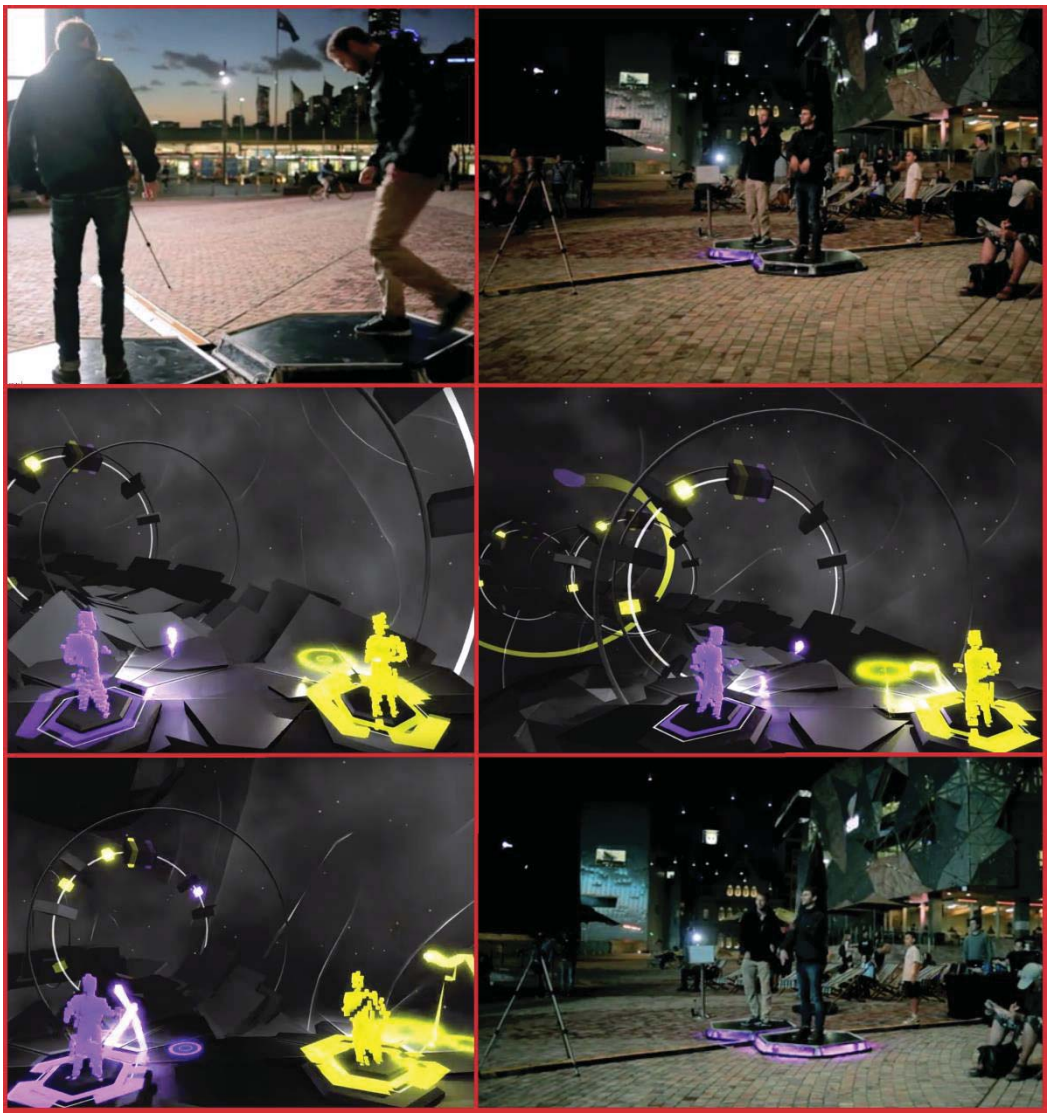

Figura 1-2: Gravitate mostrado al público en Melbourne, 2014 (fuente: https://vimeo.com/)

1.2.2 The Computer Orchestra

Es un proyecto de un grupo de estudiantes de la ECAL (École Cantonale d'Art de Lausanne, en español: Escuela Cantonal de Arte de Lausana, Suiza) en el que, por medio de Kinect, el usuario puede dirigir y monitorear toda una orquesta computarizada.

Simon de Diesbach, Jonas Lacôte y Laura Perrenoud son los responsables de esta idea. Kinect detecta los movimientos del director y transmite los datos a los equipos que forman la orquesta a través de Wi-Fi. Los sonidos han sido pregrabados como base de *samples*[5] de voces (de la misma forma que haría un sintetizador digital) y se activan de acuerdo a los movimientos del cuerpo del director. Además del sonido, los computadores envían diferentes animaciones. De esta manera el director no solo recibe sonido, sino también información visual [5]. Su proceso de trabajo se refleja en la secuencia de imágenes seleccionadas al azar y mostradas en la Figura 1-3.

Figura 1-3: Disposición y características de la orquesta (fuente: http://computer-orchestra.com/)

[5] Porción de un sonido grabado para reutilizarla posteriormente como instrumento musical o una diferente grabación de sonido.

1.2.3 The Space Palette

Es un instrumento musical y gráfico creado por Tim Thompson, ingeniero de software, músico y artista. Permite componer música y dibujar efectos visuales simultáneamente al introducir las manos en orificios de distintos tamaños en un marco de madera (4 grandes como cruz y 12 pequeños en las esquinas), como se muestra en la Figura 1-4. No existen sonidos pre-grabados, pero sí son usadas las secuencias y los loops. Todo esto es generado en tiempo real mediante la interacción de Kinect con las manos.

El marco de madera es usado como una referencia para el usuario, mientras Kinect es utilizada para detectar la posición de cualquier mano (u objeto) que se introduzca en los agujeros del marco. La profundidad de introducción de las manos es importante (brazo completo por ejemplo), así como también la posición (izquierda/derecha/arriba/abajo). En el fondo, es como tener múltiples mousepads 3D en el aire. Además puede ser usada más de una mano.

Musicalmente, los orificios grandes mantienen un parecido a las teclas de un piano de izquierda a derecha, en los cuales se pueden tocar notas individuales y controlar sonidos con la profundidad de la mano tales como el vibrato[6] y filtros. Visualmente, los agujeros permiten al usuario dibujar formas gráficas (procesadas como efectos visuales) y controlar el tamaño de los bosquejos con la profundidad de la mano. Los 12 agujeros pequeños en las esquinas del marco son usados para seleccionar diferentes conjuntos de sonidos en simultaneidad con el dibujo de diferentes gráficos [6].

[6] Ondulación del sonido que se produce en algunos instrumentos de cuerda mediante un movimiento del dedo que pisa la cuerda e intensifica su vibración.

Figura 1-4: Muestra de la forma de trabajar Space Palette (fuente: http://spacepalette.com/)

1.2.4 The V Motion Project

Es un trabajo producido por Assembly, una compañía formada por un capital humano de múltiples disciplinas que combina habilidades para formular proyectos atractivos. Su sede está en Auckland, Nueva Zelanda.

El proyecto en sí es muy similar a los realizados por Ethno Tekh; con la diferencia de que, en este trabajo, el usuario elige en la pantalla qué sonido quiere reproducir. Su funcionamiento rige en la utilización de dos Kinect conectadas a dos ordenadores distintos (Windows y Mac): una usada para calcular la posición del esqueleto y la otra para acceder a los datos profundos del sensor de infrarrojos [7]. La Figura 1-5 muestra parte del desarrollo de este proyecto, exponiendo el objetivo del uso de las dos Kinect.

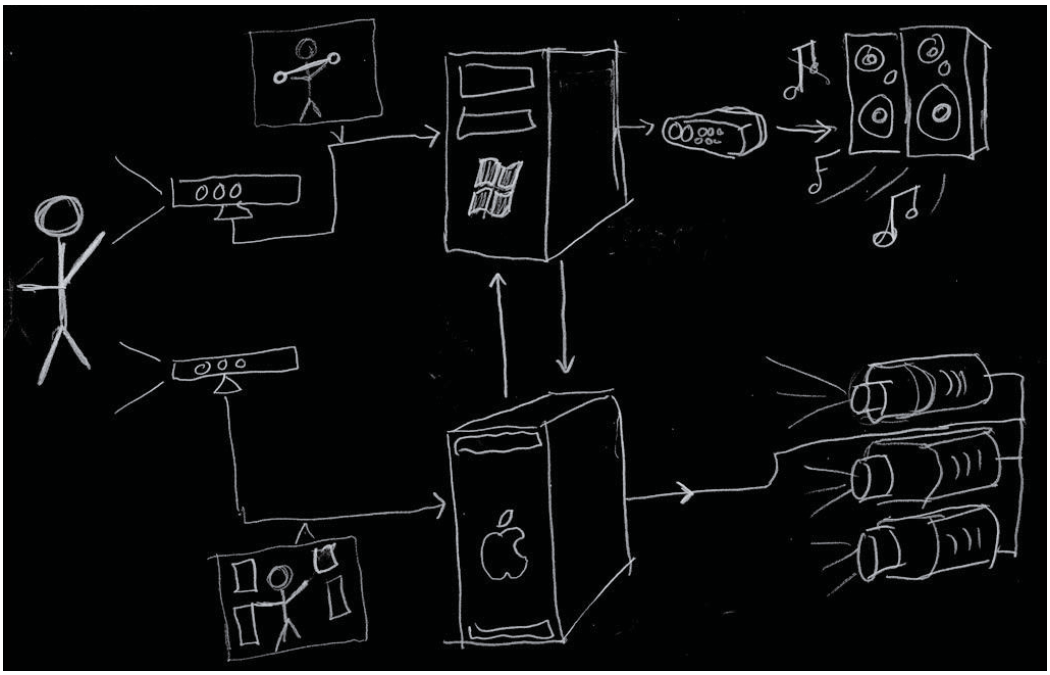

Figura 1-5: Esquema de The V Motion Project (fuente: http://custom-logic.com/)

Para que se pueda formular música correctamente, primero, se trabaja con una serie de datos de forma individual y posteriormente se unen para lograr la composición. Como ejemplo estos datos son: bajos, vox samples, baterías filtradas, etc. Finalmente queda un trabajo como el que se muestra en la Figura 1-6.

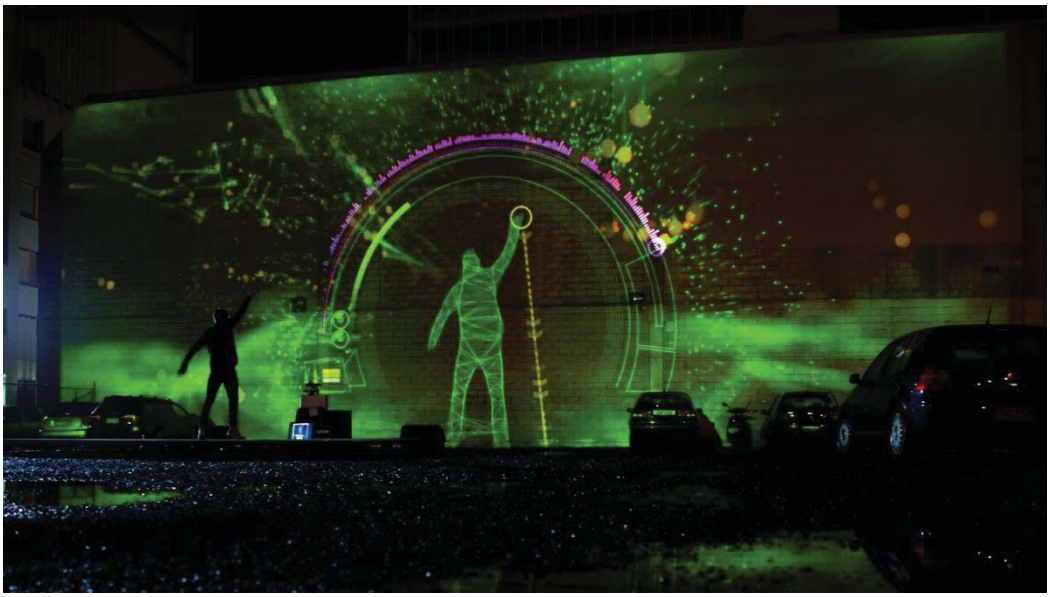

Figura 1-6: The V Motion Project mostrado en un estacionamiento en Australia (fuente: http://assemblyltd.com/)

1.2.5 MOTIV

Es un proyecto con código abierto que también utiliza el sensor Kinect. Actualmente se encuentra desactualizado, sin embargo, no impide que desarrolladores puedan modificar e incluso mejorar su funcionamiento, puesto que se puede encontrar su código de programación en GitHub[7]. Su objetivo, es brindar a los usuarios un control de las expresiones emocionales que se manifiestan en la interpretación de sus movimientos físicos en tiempo real.

MOTIV permite controlar la rapidez de reproducción de la nota y la estratificación orquestal (la sección de cuerdas), basado en la velocidad y la magnitud de los gestos. Mientras más rápido se mueva la mano derecha del usuario, mayor será la intensidad del sonido. Por otro lado, mientras más se mueva el cuerpo (giros en 360° por ejemplo), más fuerte se convierte el sonido de la sección de cuerdas [8].

Operación

Para explicar de mejor manera cómo opera este proyecto, desde la Figura 1-7 hasta la Figura 1-10 se presentan esquemas de desarrollo para llevar a cabo cada paso, además de clarificarlos.

- Primer paso: Componer música en un dispositivo digital (teclado, batería, incluso iPhone). Luego crear un archivo MIDI.

Figura 1-7: Esquema del primer paso (fuente: http://musicwithmotiv.com/)

- Segundo paso: Abrir MOTIV y cargar la secuencia digital. Seguido por seleccionar los parámetros de expresión que se desean controlar durante la utilización de Kinect.

[7] Es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. GitHub opera bajo el nombre de *GitHub, Inc.* El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago.

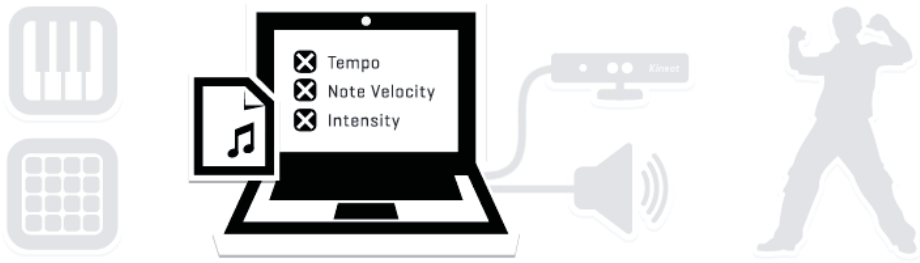

Figura 1-8: Esquema del segundo paso (fuente: http://musicwithmotiv.com/)

- Tercer paso: Conectar la Kinect al computador y utilizarla. MOTIV sigue los movimientos y los traduce a expresiones.

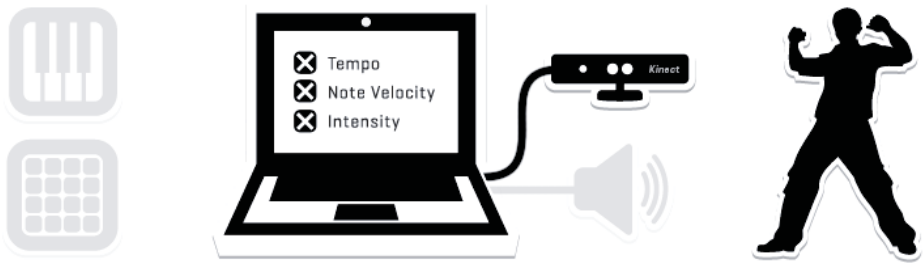

Figura 1-9: Esquema del tercer paso (fuente: http://musicwithmotiv.com/)

- Cuarto paso: MOTIV conecta las expresiones con la secuencia digital en tiempo real, enviándolos al instrumento virtual mientras se visualiza la entrada para obtener una perfecta presentación.

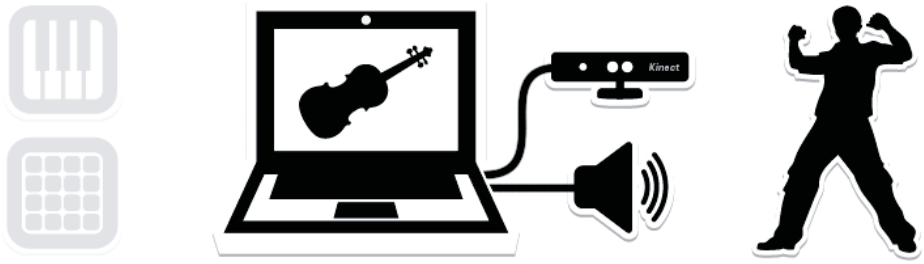

Figura 1-10: Esquema del cuarto paso (fuente: http://musicwithmotiv.com/)

1.3 Solución propuesta

Lo expuesto en el estado del arte es bastante importante, puesto que permite generar una base que será de ayuda para plantear un proyecto y poder solucionar la problemática que existe escénicamente en el universo musical. Por otro lado, obtener toda esta información previa permite que se puedan seguir abarcando ideas para cumplir con el objetivo general y los objetivos específicos; por tanto, agregar una propuesta de valor será de suma importancia para que, en el proyecto, se integre el concepto de facilidad de operación y control para usuarios que no han tenido la oportunidad de trabajar con Kinect.

Para que los dos trabajos creados por Ethno Tekh (Connections y Gravitate) pudieran funcionar y ser expuestos al público, la compañía digital desarrolló un software capaz de leer el movimiento de las manos, además de permitir el control de audios pregrabados: Kinectar. Este software se tomará como referencia para elaborar la solución propiamente tal. En los capítulos siguientes se irá explicando el procedimiento de la propuesta con un mayor grado de profundidad.

2 Marco Teórico

2.1 Sensor Kinect

2.1.1 Historia

Kinect fue lanzado en Estados Unidos el 04 de noviembre de 2010. En la Figura 2-1 se puede observar este controlador de juego libre y entretenimiento creado por Alex Kipman.

Desarrollado por Microsoft para la videoconsola Xbox 360, permitiendo a los usuarios controlar e interactuar con la consola sin necesidad de tener contacto físico con un controlador de videojuegos tradicional, mediante una interfaz natural de usuario (INU) que reconoce movimientos del cuerpo, gestos, objetos e imágenes, comandos de voz, etc. Esto hace que el jugador se envuelva más en el rol, brindando experiencias de entretenimiento extraordinarias [9].

Figura 2-1: Kinect completo (fuente: http://www.elotrolado.net/)

2.1.2 Funcionamiento y especificaciones técnicas

La empresa PrimeSense, con sede en Tel-Aviv, Israel, alcanzó la hazaña tecnológica a través de su diseño "PrimeSensor" que brinda la posibilidad de percibir el mundo en tres dimensiones y traducir estas percepciones en una imagen sincronizada. Dentro de esta idea básica, la sala donde se ubica el sistema de Kinect y sus usuarios, está compuesta con un patrón de puntos invisibles y que son generados por un láser que trabaja en la frecuencia del infrarrojo cercano. Esto es, un dispositivo láser de clase 1 que proporciona un enfoque a una distancia útil sin generar condiciones peligrosas para los jugadores. Un sensor de imagen CMOS detecta los reflejos del haz infrarrojo, dentro de un patrón de puntos y segmentos, y genera mapas que informan sobre la intensidad de cada punto y segmento. Este proceso se realiza a una distancia de pocos metros y se obtiene una resolución de hasta 1 centímetro dentro de la dimensión de la profundidad (Eje Z). La resolución espacial (Ejes X e Y) es del orden de milímetros, y se utiliza la entrada de información del RGB (Rojo, Verde y Azul) desde un segundo sensor de imagen CMOS (como la cámara de un teléfono móvil) para añadir color a los datos adquiridos. El frente de Kinect, en donde se ubican todos dispositivos y características mencionadas, se ve en la Figura 2-2 [10].

Figura 2-2: Distribución externa de Kinect (fuente: http://www.elotrolado.net/)

Las características de Kinect pueden ser especificadas con un mayor grado de profundidad, éstas se han agrupado en una tabla para mostrar y agrupar el contenido por secciones comunes; las que se presentan en la Tabla 2-1.

Tabla 2-1: Características y especificaciones técnicas de Kinect

Sensores/Cámaras	Cámara RGB. Cámara VGA con resolución de 640 · 480p a 30 FPS. Doble cámara de profundidad con resolución de 640 · 480p a 30 FPS. Sensor de profundidad. Micrófono multi-arreglo.
Campo de visión	Campo de visión horizontal de 57°. Campo de visión vertical de 43°. Rango de inclinación física de ± 27°. Rango de profundidad de 1,2 - 3,5 metros.
Flujo de datos	320 · 240 a 16 bits de profundidad a 30 FPS. 640 · 480 a 32 bits de color a 30 FPS. Audio de 16 bits a 16 [kHz]. Acelerómetro 2G/4G/8G configurado para rango de 2G, con un límite superior de precisión de 1°.
Sistema de audio	Sistema con cancelación de eco. Permite chat en vivo y voz dentro de la aplicación. Reconocimiento de voz múltiple. Array de cuatro micrófonos con un convertidor análogo a digital de 24 bits (ADC) y procesamiento de señales de Kinect con eliminación de ruido de fondo.
Sistema de seguimiento	Rastreo de hasta 6 personas, incluyendo a 2 usuarios utilizándola. Rastreo de 20 articulaciones por usuario activo. Capacidad de modelar avatares a imagen y semejanza del usuario

Requisitos del sistema [9]

- Requerimientos de Hardware
  - Sistema operativo : Windows 7 o superior (32 o 64 bits)
  - Procesador : Dual Core a 2,66 GHz o superior
  - USB : 2.0
  - Memoria RAM : 2 GB o superior

- Requerimientos de Software
  - Microsoft Visual Studio C# 2010 Express
  - .NET FrameWorks
  - OpenNI o alguna librería compatible con Kinect (el SDK de Windows es muy utilizado).

Como otras funcionalidades, se pueden mencionar librerías y sus características que son bastante útiles para desarrolladores e interesados en el trabajo con Kinect. Estas se muestran en la Tabla 2-2.

Tabla 2-2: Librerías compatibles con Kinect

Librería	Lenguajes	Características
Open Kinect/Libfreenect	C, Python, ActionScript, C#, C++, Java JNI y Java JNA, JavaScript, CommonLisp	Imágenes de color y profundidad, control de la base motorizada.
Robot Operating System	UNIX	Imágenes de color y profundidad, control de la base motorizada.
Open NI/NITE Middleware	Windows, Linux, Mac OS X	Identificación de usuario, detección de gestos, seguimiento y orientación de articulaciones de usuario, imágenes de color y profundidad.
SDK Microsoft Kinect	Windows	Identificación de usuario, detección de gestos, seguimiento y orientación de articulaciones de

		usuario, imágenes de color y profundidad.
openFrameworks	C++, Windows, Mac OS X, Linux, iOS, Android	Identificación de usuario, detección de gestos, seguimiento y orientación de articulaciones de usuario, imágenes de color y profundidad.

2.2 Librerías de Kinect

Existen múltiples librerías que logran conectar Kinect con un ordenador. Sin embargo, para estudios de interés de este proyecto se definirán las dos más importantes: la primera de ellas es OpenNI/NiTE, y corresponde a la que se utilizará para llevar a cabo el trabajo definido en este informe. Seguida por la librería del SDK de Windows, que está más actualizada y puede ser usada por quién desee seguir desarrollando trabajos con Kinect.

2.2.1 OpenNI

Es un SDK (Kit de desarrollo de software) con código abierto usado para el desarrollo de bibliotecas de detección 3D usando las librerías middleware y aplicaciones. La página web de OpenNI, provee una comunidad activa de programadores involucrados en la creación de herramientas, apoyo a quienes recién inician, posibles redes de socios y una plataforma de distribución para abordar el ciclo completo del desarrollo [11].

OpenNI facilita la comunicación con los sensores de audio, video y profundidad de dispositivos hardware compatibles como Kinect (basado en el chip PS1080 de PrimeSense[8]). Así como también la comunicación con la librería de middleware, encargada de analizar y comprender los datos obtenidos de la escena mediante algoritmos [12]. Esto se ilustra mejor en la Figura 2-3.

[8] Fue una compañía israelí dedicada a la detección 3D. Era la asociación más conocida, y que licenciaba el diseño de hardware y chip utilizado para la detección de movimiento de Kinect de Microsoft en Xbox 360, el año 2010.

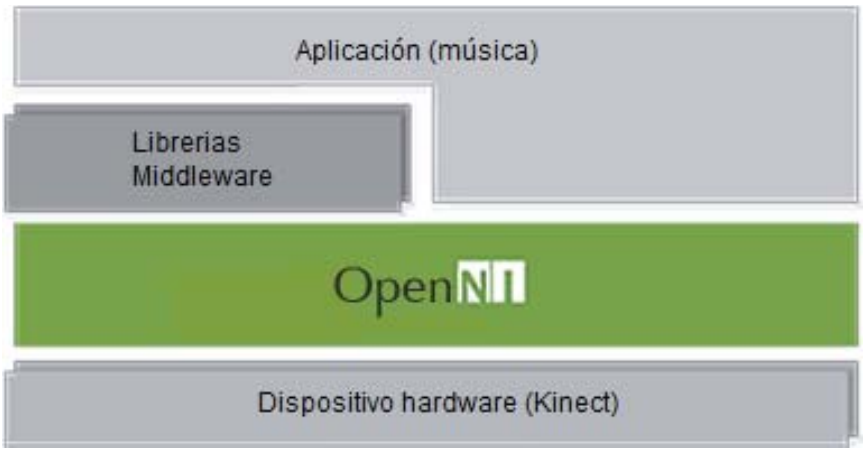

Figura 2-3: Arquitectura de OpenNI (fuente: http://openni.ru/)

2.2.2 NiTE

Es la librería Middleware de visión 3D por computador más robusta y avanzada, desarrollada por PrimeSense. Su última actualización data del año 2013, sin embargo sigue siendo útil para la ejecución de programas de desarrolladores de Kinect. Ocupa recursos mínimos de la CPU y mantiene soporte en la medida de lo posible.

Esta Middleware proporciona la aplicación con una API[9] de control de usuario clara (si se trata de control basado en la mano o en todo el cuerpo). Los algoritmos utilizan la profundidad, color y la información de audio recibida desde el dispositivo de hardware, que les permiten realizar funciones tales como la localización y seguimiento de la extremidad. Además cuenta con un analizador de escena (separación de los usuarios de fondo), que precisa las articulaciones en el seguimiento del esqueleto, reconocimiento de los gestos del usuario y más. En la Figura 2-4 es posible apreciar el funcionamiento de un programa del controlador de OpenNI con Kinect.

[9] Application Programming Interface (Interfaz de Programación de Aplicaciones).

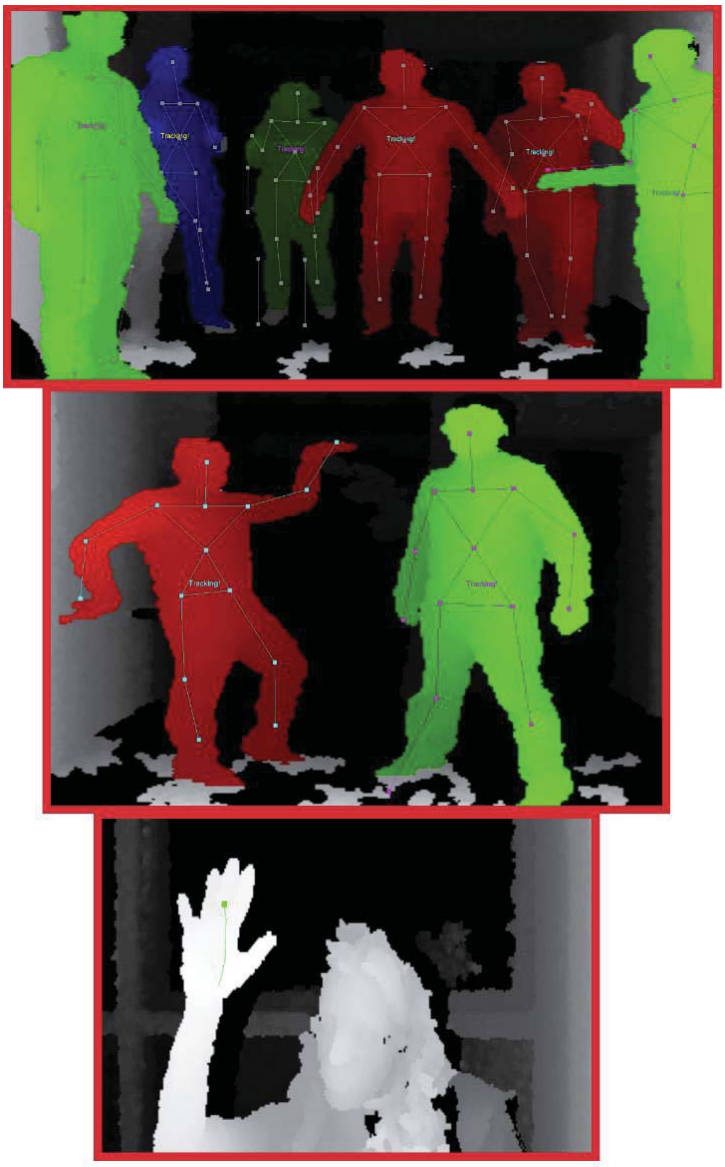

Figura 2-4: Ejemplos de OpenNI/NITE (fuente: http://openni.ru/)

2.2.3 Kit de desarrollo de software de Windows (SDK) [9]

Kinect estuvo en la mira de muchos científicos, aficionados e inventores al ser una herramienta muy potente. Esto motivó a programadores a lograr el acceso a este dispositivo, consiguiendo un SDK no oficial. Por medio de la manipulación sobre el sensor lograron construir nuevas aplicaciones aprovechando el potencial que dispone este periférico. Debido a esto, Microsoft se vio vulnerado con el hackeo de su sensor, lo que motivó a la empresa a liberar en el año 2010 de forma oficial el software SDK para Kinect en entornos de Windows, el que permitía acceder a todas las funciones del sensor (seguimiento de usuario, cámara RGB, acceso a micrófonos) [13].

El SDK oficial de Microsoft es muy utilizado en la actualidad por distintos desarrolladores, su plataforma de uso es un software de descarga gratis que presenta ejemplos, documentos, formas de operar el SDK, herramientas de lenguajes de programación, entre otros; con el objetivo de facilitar la tarea de crear proyectos propios, tal como se muestra en la Figura 2-5.

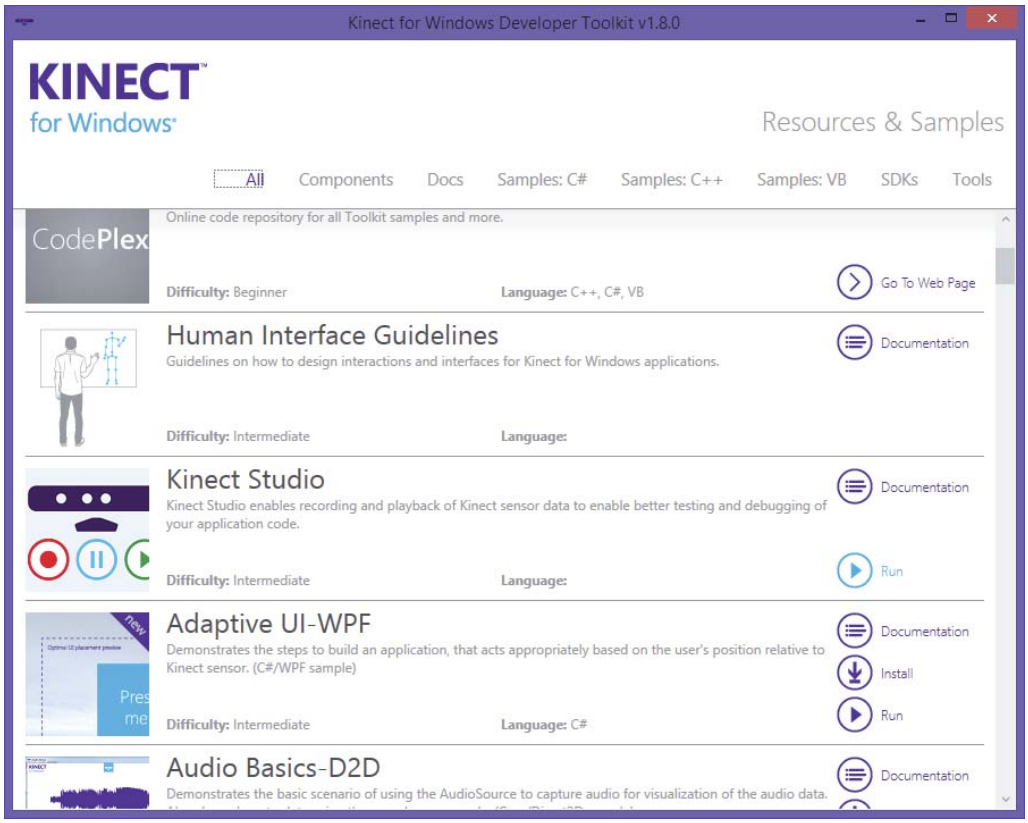

Figura 2-5: Plataforma del SDK de Windows (fuente: http://www.mcs.csueastbay.edu/)

3 Desarrollo

3.1 Herramientas computacionales

Para llevar a cabo el proyecto, se necesitaron variadas herramientas computacionales. Por ahora, es indiferente el orden de uso, sólo importa clarificar sus funcionalidades y formas de operar. En el transcurso del informe se irá detallando cómo funciona cada una por si sola y el orden de ejecución para crear contenido musical y manipularlo con Kinect.

3.1.1 Kinectar

Kinectar es un set de herramientas que convierten a Kinect en un controlador MIDI. Requiere de un seguimiento del esqueleto, por lo que es necesaria la utilización de otros softwares, que pueden ser Synapse u OSCeleton (ambas aplicaciones construidas por desarrolladores independientes). Estos softwares envían datos OSC a Kinectar, y luego Kinectar se configura para los valores de control de salida MIDI y notas basadas en el movimiento del cuerpo.

Características:

- Salidas MIDI y OSC escalables.
- Cuatro generadores de notas MIDI basados en la condición.
- Compatible con ReWire[10].
- MIDI controlables en conmutación configurada.
- Sistema global de “banderas” para cambiar configuraciones en ON y OFF.

Kinectar ha sido descontinuada, por tanto no tendrá más actualizaciones [14]. Sin embargo permanece disponible para quien la desee ocupar. Ha sido reemplazada por un software de características mejoradas, pero esto no quiere decir que es inservible, puesto que cumple con muchas de las funcionalidades que se necesitan para llevar a cabo este proyecto. Pese a que este nuevo software es más actualizado y tiene más capacidades, también posee particularidades que no interesan; es decir, para este proyecto sólo es necesaria la transformación del cuerpo en un controlador MIDI. Por lo tanto, por simplicidad, se trabajará solamente con Kinectar.

[10] Protocolo de software que permite el control remoto y transferencia de datos entre ediciones de audio digital y el software relacionado.

En lo sucesivo se mostrará una serie de imágenes que hará más fácil la tarea de familiarización con la interfaz de trabajo. Primero una captura del software Kinectar completo (se puede observar en la Figura 3-1) y luego un desglose de este, como se muestra desde la Figura 3-2 hasta la Figura 3-7. Con la intención de evaluar cada componente de trabajo de Kinectar.

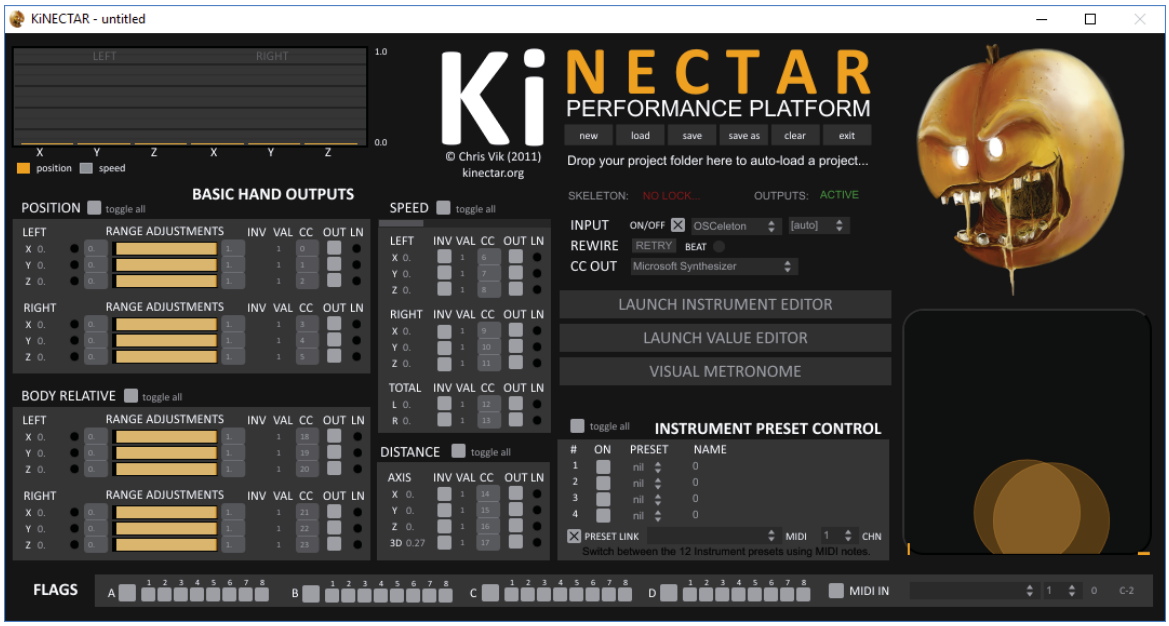

Figura 3-1: Plataforma de Kinectar (fuente: elaboración propia)

La primera parte que se obtiene del software es la lectura de las manos y su seguimiento, como se muestra en la Figura 3-2, posteriormente, el reconocimiento de la magnitud del movimiento de las manos (qué tan rápido y brusco es el movimiento) para así poder manejar los datos recibidos desde Kinect, como se observa en la Figura 3-3.

Figura 3-2: Lectura y seguimiento de las manos (fuente: elaboración propia)

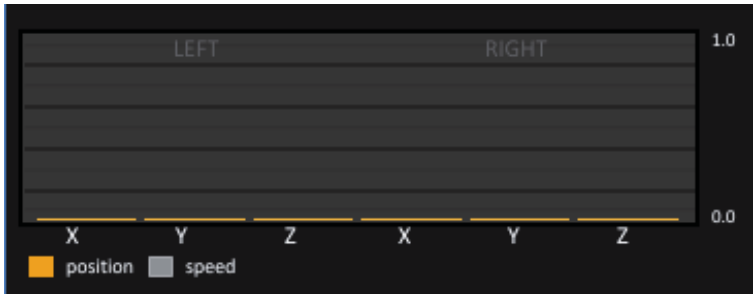

Figura 3-3: Reconocimiento de la magnitud del movimiento de las manos (fuente: elaboración propia)

Luego, se obtiene lo más importante del software: el sistema de control para configurar los rangos en que las manos pueden desplazarse por la pantalla; por ejemplo, de 0 [-] a 0.5 [-] en un rango de 0 [-] a 1 [-], es decir, la mano es leída sólo hasta la mitad de la pantalla. También, en qué eje (x, y, z) su desplazamiento efectúa alguna orden (subir o bajar el volumen desplazando la mano izquierda en el eje y, por ejemplo). Además, configurar su velocidad de movimiento para ejecutar cualquier otra orden; por ejemplo, el movimiento brusco de la mano derecha en el eje x detiene el sonido de la pista de audio. Finalmente, especificar que la distancia entre las manos, en cualquiera de los tres ejes, genere alguna orden; producir el efecto de un filtro, por ejemplo. Todo lo anterior se puede verificar en la Figura 3-4.

Figura 3-4: Configuración de la posición y velocidad de movimiento (fuente: elaboración propia)

Por otro lado, existen interferencias entre el mapeo de CC's[11] a MIDI si se encuentra Kinectar activado, por lo que se deben silenciar todas las entradas y salidas que puedan afectar a la correcta transferencia de datos. En la Figura 3-5 se muestra el ícono que realiza esta función.

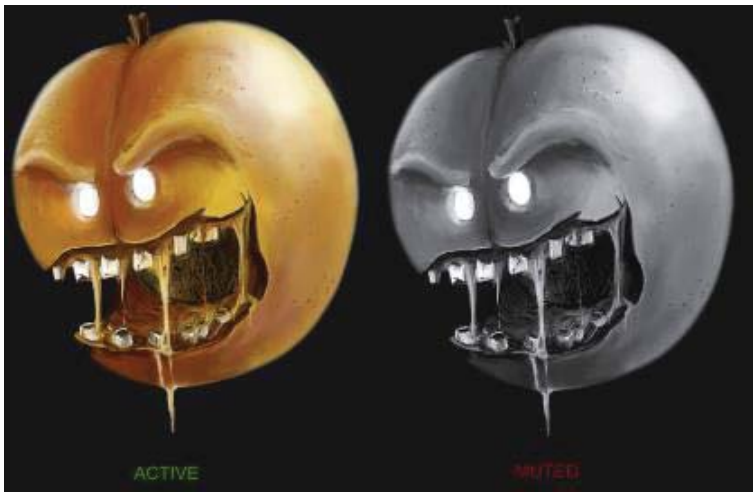

Figura 3-5: No silenciado/Silenciado para entradas MIDI (fuente: elaboración propia)

Asimismo, en la Figura 3-6, se pueden observar los botones típicos de todo software: crear proyectos, abrirlos, guardarlos, etc.; información acerca de si se está leyendo efectivamente el cuerpo o de si las salidas están silenciadas o activas. Además de la ejecución de otras funcionalidades de Kinectar que en este proyecto no se utilizan, por lo que quedan sólo como referencia para el lector (funciones que también pueden advertirse en la Figura 3-7).

[11] Control Change: son números que van desde 0 a 127 y corresponden a parámetros de control que se pueden asociar a alguna variable MIDI. Por ejemplo, la asignación del parámetro 50 del CC de Kinectar para el control de una perilla de volumen de un software capaz de enviar datos MIDI.

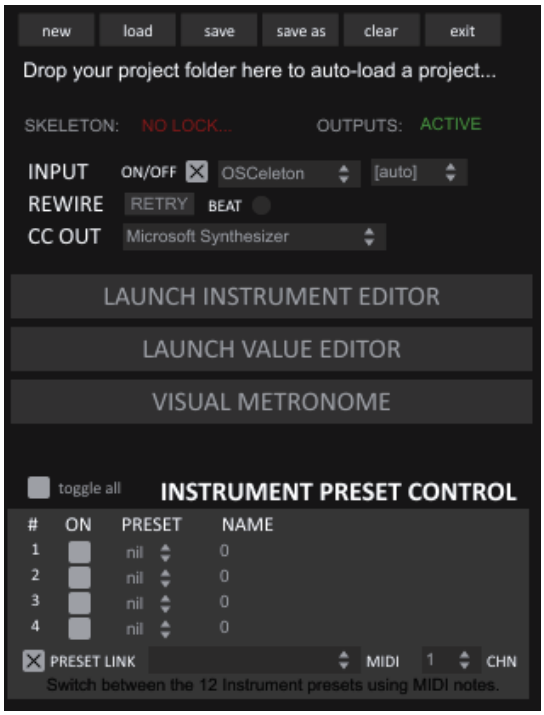

Figura 3-6: Herramientas de trabajo de configuración de Kinectar (fuente: elaboración propia)

Figura 3-7: Sistema global de banderas para configuración (fuente: elaboración propia)

3.1.2 Pure Data

Pure Data es un lenguaje de programación visual, de código abierto, que funciona con múltiples dispositivos (desde computadores hasta Smartphones). Permite a músicos, artistas visuales, intérpretes, investigadores y desarrolladores crear sus propios proyectos gráficamente; es decir, sin escribir ninguna línea de código. Es utilizado para generar sonidos, videos, gráficos 2D/3D y procesar sensores de interfaz y dispositivos de entrada MIDI [15].

Para efectos de este proyecto, Pure Data tendrá la utilidad de cargar contenido musical y lograr controlarlo mediante el movimiento de las manos. Es decir, conocer valores de CC's que se interconecten entre Kinectar y Pure Data mientras se efectúan los movimientos; como ejemplo, con el movimiento de las manos se puede controlar el volumen, los efectos de filtro, el loop (o repetición continua) de la pista, entre otros. En la Figura 3-8 se puede observar y la plataforma de trabajo de Pure Data.

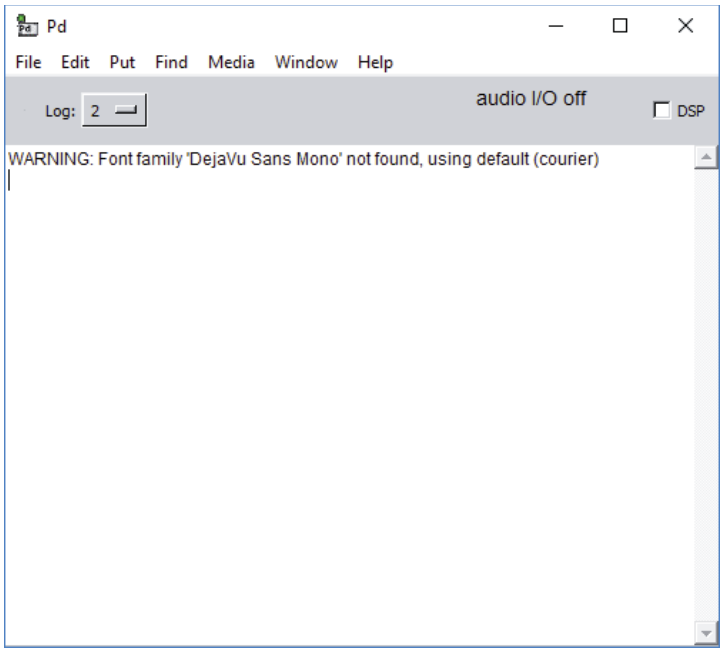

Figura 3-8: Plataforma de Pure Data (fuente: elaboración propia)

3.1.3 OSCeleton

Desarrollado por SenseBloom –una asociación con sede en Coímbra, Portugal– en el año 2011. Es un pequeño programa que toma los datos del esqueleto de Kinect desde la librería de OpenNI, y lee y envía las coordenadas de las articulaciones del esqueleto a través de mensajes OSC. Finalmente estos mensajes pueden ser enviados a otros protocolos de comunicación mediante su compatibilización [16]. OSCeleton se encuentra actualmente desactualizado por dos motivos:

1. Su librería, OpenNI/NiTE, fue comprada por Apple, con lo que dejó de mantener su código abierto; por tanto, desarrolladores independientes dejaron de actualizarla y mejorarla. Hoy está compatibilizada como máximo para Windows 8. A pesar de lo anterior, su código se mantiene en GitHub (véase referencia [15]) para quien desee modificar su contenido o actualizarlo.

2. SenseBloom dejó de dar soporte a OSCeleton desde el año 2012. Aunque existe un nuevo OSCeleton[12] que trabaja mediante la librería del SDK de Windows. Es parcialmente incompatible con la versión original (aunque siempre puede ser una buena herramienta de trabajo para quien desee trabajar desde el SDK de Windows).

En la Figura 3-9 se puede observar el funcionamiento de OSCeleton en Windows 7 de 32 bits, mediante la operación de los ejemplos del controlador OpenNI.

[12] https://www.github.com/Zillode/OSCeleton-KinectSDK/

Figura 3-9: OSCeleton en ejecución (fuente: elaboración propia)

3.1.4 loopMIDI

loopMIDI es un software creado por Tobias Erichsen. Es útil para la manipulación de contenido musical y puede utilizarse para crear puertos MIDI de bucle de retorno virtual. Esto sirve de ayuda para interconectar aplicaciones en Windows que requieran abrir puertos de hardware y MIDI para la comunicación. Los puertos creados son únicos para cada usuario y sólo existen mientras la aplicación loopMIDI se está ejecutando. Por lo tanto, si se cierra la sesión, los puertos creados dejarán de existir. Al cerrar la configuración, la aplicación no finalizará, sino que se minimizará a la barra de bandeja [17]. En la Figura 3-10 se puede observar la interfaz de loopMIDI y la creación de un puerto virtual listo para ser utilizado por otro software que lo requiera.

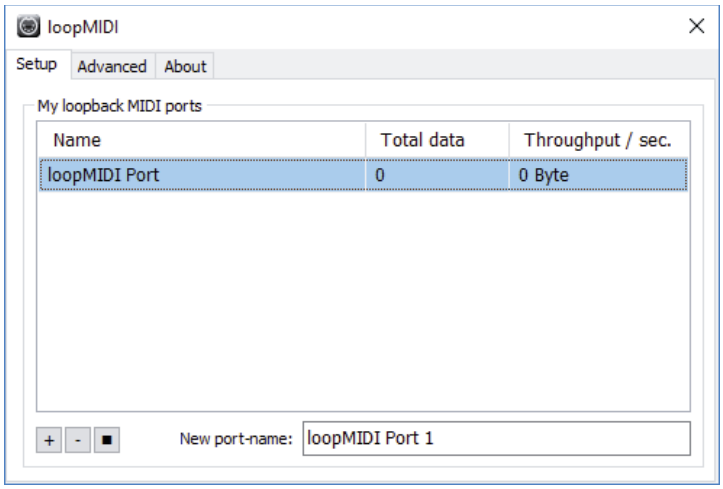

Figura 3-10: loopMIDI en ejecución (fuente: elaboración propia)

3.1.5 Embarcadero Delphi [18]

Embarcadero Delphi es un entorno de desarrollo de software diseñado para la programación visual orientada a objetos. Se instauró con el propósito de agilizar la creación de programas. En Delphi se utiliza una versión más actual del lenguaje Pascal, conocida como Object Pascal. Por otro lado, es un lenguaje muy versátil y se usa para casi cualquier proyecto, como por ejemplo servicios del sistema operativo, establecer comunicación entre un servidor web y un programa, aplicaciones de consola, conectividad con bases de datos, para realizar aplicaciones visuales, etc.

Este lenguaje produce aplicaciones en código máquina, por lo que el computador las dilucida inmediatamente y no precisa de un lenguaje interprete como es necesario en otros lenguajes de programación. En la Figura 3-11 se observa la última actualización de la plataforma de desarrollo de Embarcadero Delphi (versión 10.1) y sus múltiples aplicaciones con la tecnología en ordenadores y Smartphones.

Para efectos de mayor facilidad de trabajo, en este proyecto se utilizará la versión Embarcadero Delphi 7.

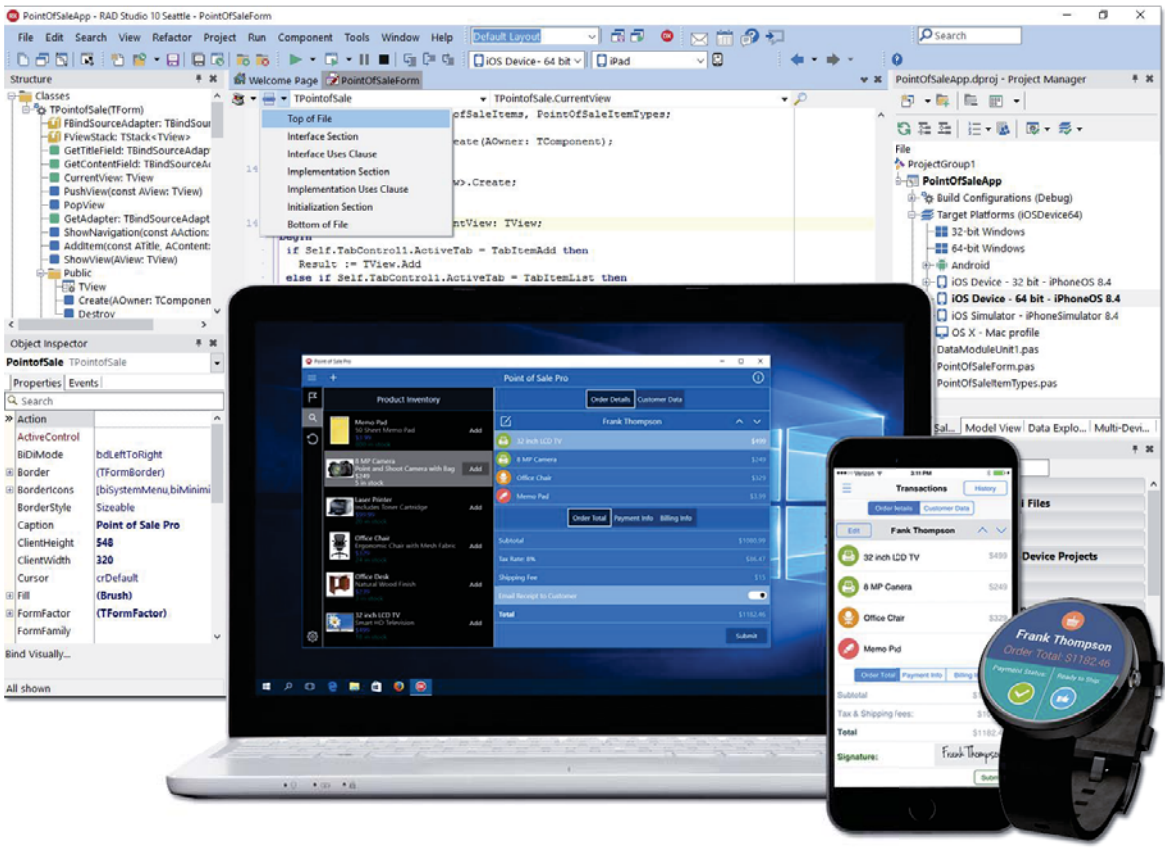

Figura 3-11: Entorno de programación de Embarcadero Delphi en su última versión (fuente: https://www.embarcadero.com/)

3.1.6 Adobe Photoshop

Adobe Photoshop es el programa de imágenes digitales más avanzado del mundo. Lo utilizan fotógrafos, diseñadores, profesionales web y profesionales de vídeo. La aplicación le ofrece el control creativo y potente más exhaustivo de la composición y la manipulación de imágenes en 2D y 3D, la edición de vídeo y el análisis de imágenes. Ya que Photoshop forma parte de Adobe Creative Cloud, se puede acceder a las actualizaciones más recientes y versiones futuras en el momento en que estén disponibles, puesto que sus versiones se almacenan y actualizan en la nube [19]. En la Figura 3-12 se puede observar la plataforma de trabajo en ejecución de Adobe Photoshop.

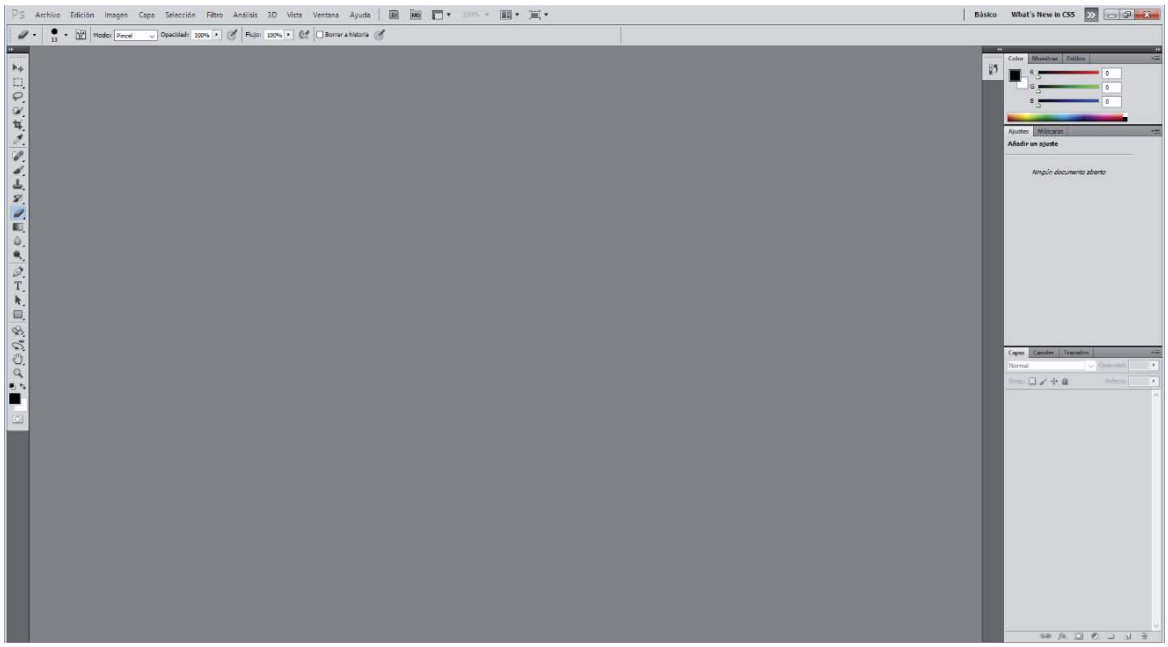

Figura 3-12: Entorno de programación de Adobe Photoshop (fuente: elaboración propia)

3.1.7 Adobe Illustrator

Adobe Illustrator es el software de gráficos vectoriales estándar, utilizado en todo el mundo por diseñadores de todo tipo que desean crear gráficos digitales, ilustraciones y tipografía para toda clase de medios: impresión, la Web, medios interactivos, videos y dispositivos móviles. Ya que Adobe Illustrator forma parte de Adobe Creative Cloud, se puede acceder a las actualizaciones más recientes y a las versiones futuras desde el momento en el que estén disponibles, puesto que sus versiones se almacenan y actualizan en la nube [20]. En la Figura 3-13 se puede observar la plataforma de trabajo en ejecución de Adobe Illustrator.

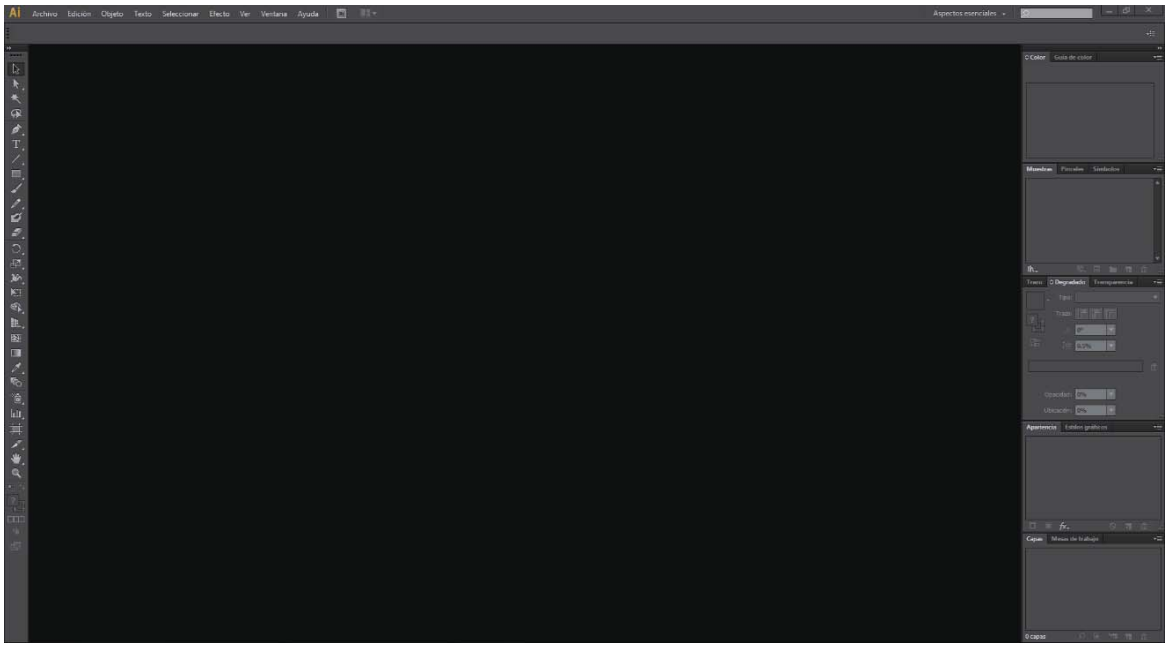

Figura 3-13: Entorno de desarrollo de Adobe Illustrator (fuente: elaboración propia)

3.2 Puesta en marcha

A partir de este momento se empieza a desarrollar la solución propuesta. Para empezar, se tiene que pensar en qué softwares de los presentados en el subcapítulo 3.1 serán de utilidad para manipular la música y lograr controlarla con Kinect. Para ello se ordenan cuatro de ellos en la siguiente disposición:

1. Pure Data, para manipular contenido musical.
2. Kinectar, para transformar el cuerpo en un controlador MIDI y conectarlo a Pure Data.
3. OSCeleton, para leer el cuerpo y conectarlo a Kinectar.
4. loopMIDI, para crear puertos MIDI virtuales y vincular los tres softwares anteriores.

Para la puesta en marcha se presenta la problemática de trabajar con los cuatro softwares simultáneamente. Esto genera dificultad y confusión en la manipulación, ya que se deben tener muchas ventanas abiertas en el ordenador. Para corregir esta problemática, se desarrollará una plataforma denominada GUI[13] que juntará estas cuatro ventanas y las mezclará en una sola. La forma de desarrollarla será por medio de la programación en el software Embarcadero Delphi 7 (descrito en el subcapítulo 3.1). Por otro lado, la GUI tendrá integrado un set de instrucciones que clarificarán al usuario su forma de uso. Para la creación de este set de instrucciones se trabajará con Adobe Photoshop y Adobe Illustrator (descritos en el subcapítulo 3.1). Una vez que la GUI

[13] Interfaz gráfica de usuario: Utiliza un conjunto de imágenes y objetos gráficos para representar información y acciones disponibles en la interfaz.

esté conformada sólo resta crear el sistema de control de las pistas de audio, con el objetivo de conseguir un ejemplo para que el usuario pueda interactuar por primera vez con la plataforma. Todo lo anterior se detalla en las secciones 3.2.1 y 3.2.2 respectivamente.

3.2.1 Creación de GUI

Se lleva a cabo, mediante dos pasos: primero, usando Embarcadero Delphi 7 para programar la plataforma; y segundo, usando Adobe Photoshop y Adobe Illustrator para mejorar la interfaz haciéndola más intuitiva y agradable a la vista, además de proporcionar las pestañas del menú principal y las ventanas de instrucciones. No se dará mayor detalle en esta sección más que la imagen referida en la Figura 3-14, que muestra parte del código de programación de la GUI, y la imagen presentada en la Figura 3-15 que muestra la creación de la ventana de instrucciones. Todo esto, porque se considera que lo más importante es la familiarización del usuario con Kinect mediante el sistema de control de la música y su manipulación con el movimiento de las manos (descrito con profundidad en la sección 3.2.2).

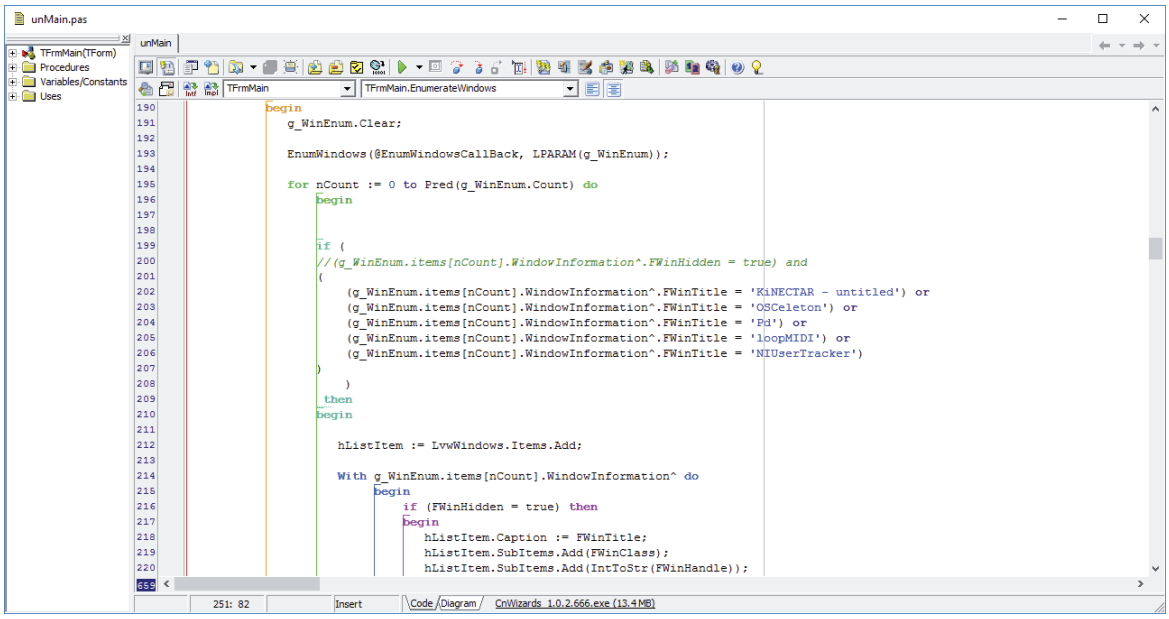

Figura 3-14: Parte del código de programación de la interfaz gráfica en Embarcadero Delphi 7 (fuente: elaboración propia)

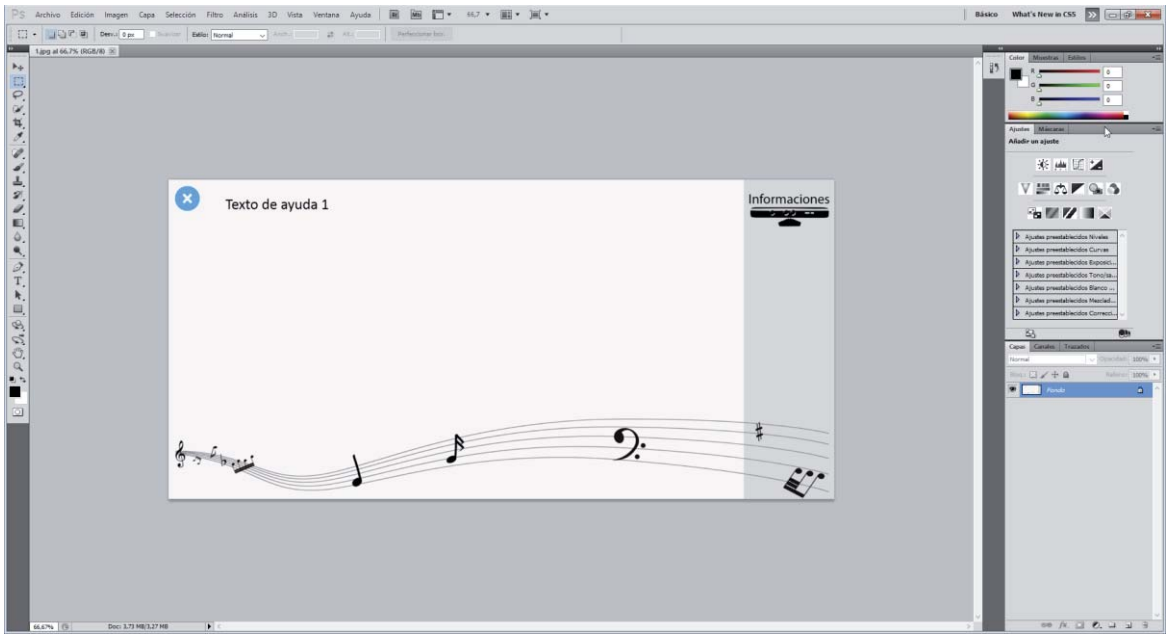

Figura 3-15: Creación de ventana de ayuda en Adobe Photoshop (fuente: elaboración propia)

3.2.2 Creación del sistema de control de música

Pure Data

Una vez que la interfaz de usuario se ha desarrollado, es necesario crear un archivo de ejemplo para que quien desee ocupar la plataforma sepa de qué manera puede ser operada. En primer lugar se elaborará un patch[14] en Pure Data. La forma de interactuar con este patch será cargando un archivo de audio pregrabado en formato .wav, y luego controlar su reproducción (play, stop, pause, volumen, efectos de filtros y velocidad de reproducción) mediante el movimiento del cuerpo. En la Figura 3-16 se muestra parte de la iniciación del patch creado en Pure Data para el control de la pista de audio.

[14] Unidad o archivo en donde se programan proyectos en Pure Data.

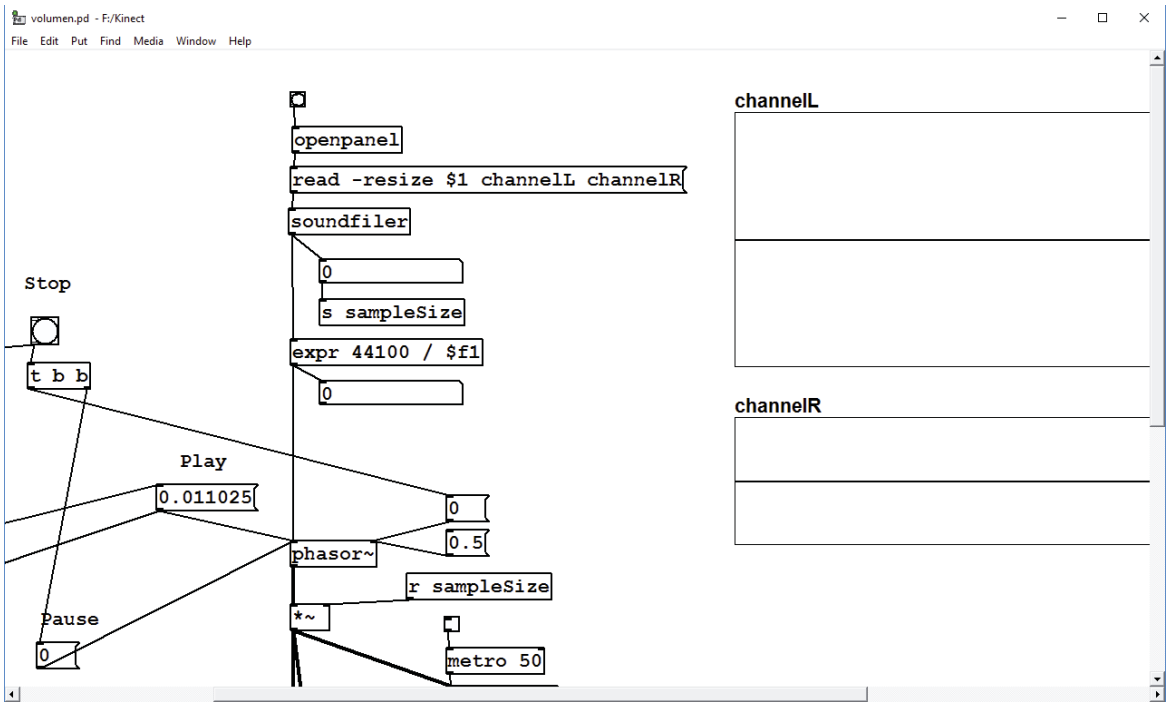

Figura 3-16: Parte del código de programación del patch (fuente: elaboración propia)

El patch permite controlar cuatro pistas de audio en forma simultánea, además posibilita el grabado y la lectura de los últimos movimientos ejecutados (presets). A continuación se repasarán los pasos fundamentales para la creación del patch:

Control de pistas

En la Figura A-1 del Apéndice se muestra la iniciación completa del patch, que cada vez se fue mejorando y adaptando a las necesidades del usuario. De esta figura se desprenden cinco actividades:

1. Control principal [21]: Se muestra en la Figura 3-17. Todo lo que está dentro del recuadro de color plomo es lo que se ve finalmente en el patch, el resto es parte de la programación. Este fragmento del patch permite la carga de la pista de audio, divide el número de muestras totales para obtener la porción que cada pista necesita para ser leída (marco de color rojo), brinda información acerca de la reproducción de la pista y de sus canales de transmisión (izquierdo y derecho) y finalmente transfiere información al resto del patch, mediante los comandos “send” y “receive”. El único problema que afecta a la pista es que sólo se permiten 01:30 minutos de reproducción continua para luego volver al inicio y entrar en loop.

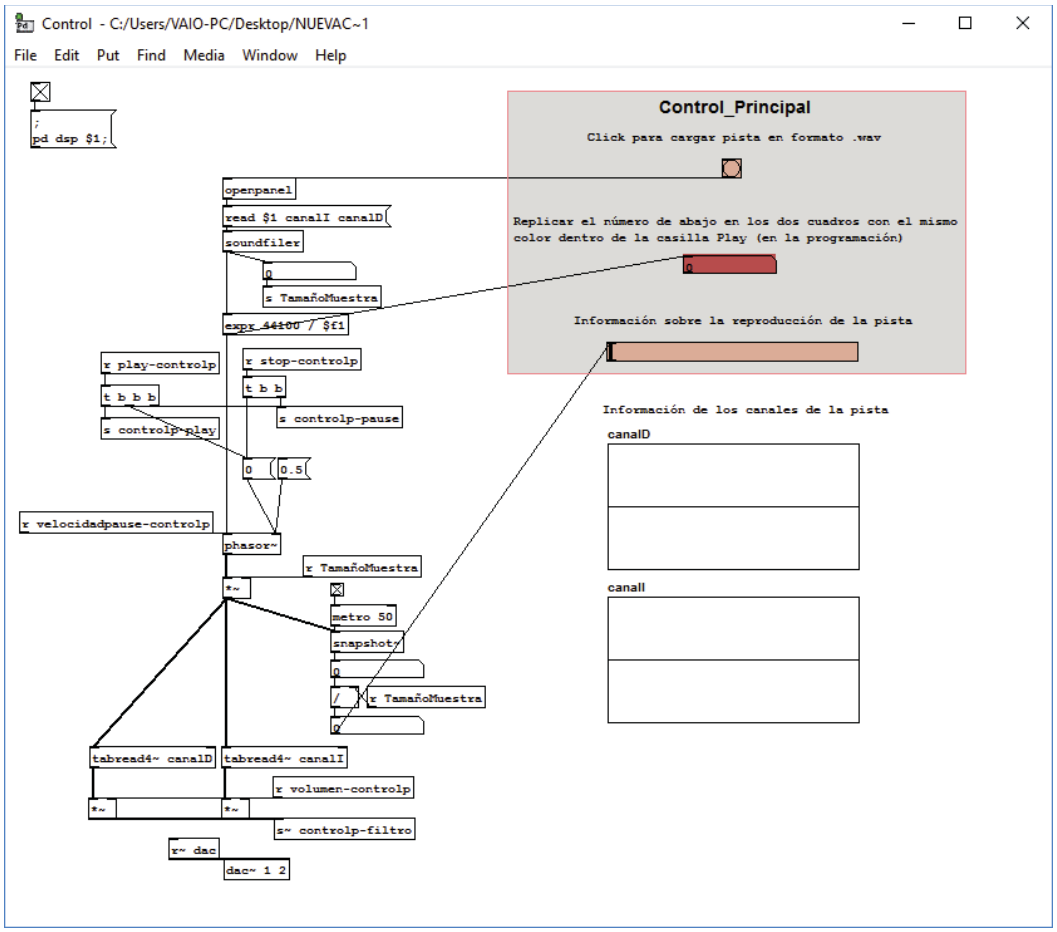

Figura 3-17: Programación del control principal del patch (fuente: elaboración propia)

2. Control del botón play, pause y stop: El control del botón play se muestra en la Figuras 3-18. En los recuadros de color rojo se debe replicar el valor obtenido en el control principal (marco de color rojo en la Figura 3-17), ya que son quienes controlan la velocidad de reproducción de la pista, es decir, un valor distinto ocasionará que la reproducción cambie de rapidez. Los comandos ctlin-ctlout[15] logran conectar el patch de Pure Data con Kinectar. Y a partir de este enlace se consigue controlar el botón play de la pista con el movimiento de ambos brazos simultáneamente hacia arriba (eje y), mediante el CC 1 y CC 22 de Kinectar (quienes forman la conexión son los comandos: $\boxed{ctlin\ 1\ 1}$ – $\boxed{ctlin\ 22\ 1}$). Los otros dos comandos (CC 6 y CC 7 de Kinectar) cumplen la misma funcionalidad, solo que mapeados hacia otras variables.

[15] Comandos de Pure Data destinados para la asignación a un número de controlador específico, desde el cual puede enviar-recibir valores de control MIDI. Son usados en este proyecto para el control de todas las variables con Kinect.

Por otra parte, debajo de los comandos ctlin, se realiza la transformación de los valores de CC's arrojados por Kinectar –que van desde 0 a 127– a un valor adecuado para la lectura de Pure Data.

En la Figura 3-19 y Figura 3-20 se obtienen los mismos resultados anteriores; sin embargo, en estos casos se controlan los botones pause y stop, respectivamente. Además de ser controlados mediante otros movimientos del cuerpo.

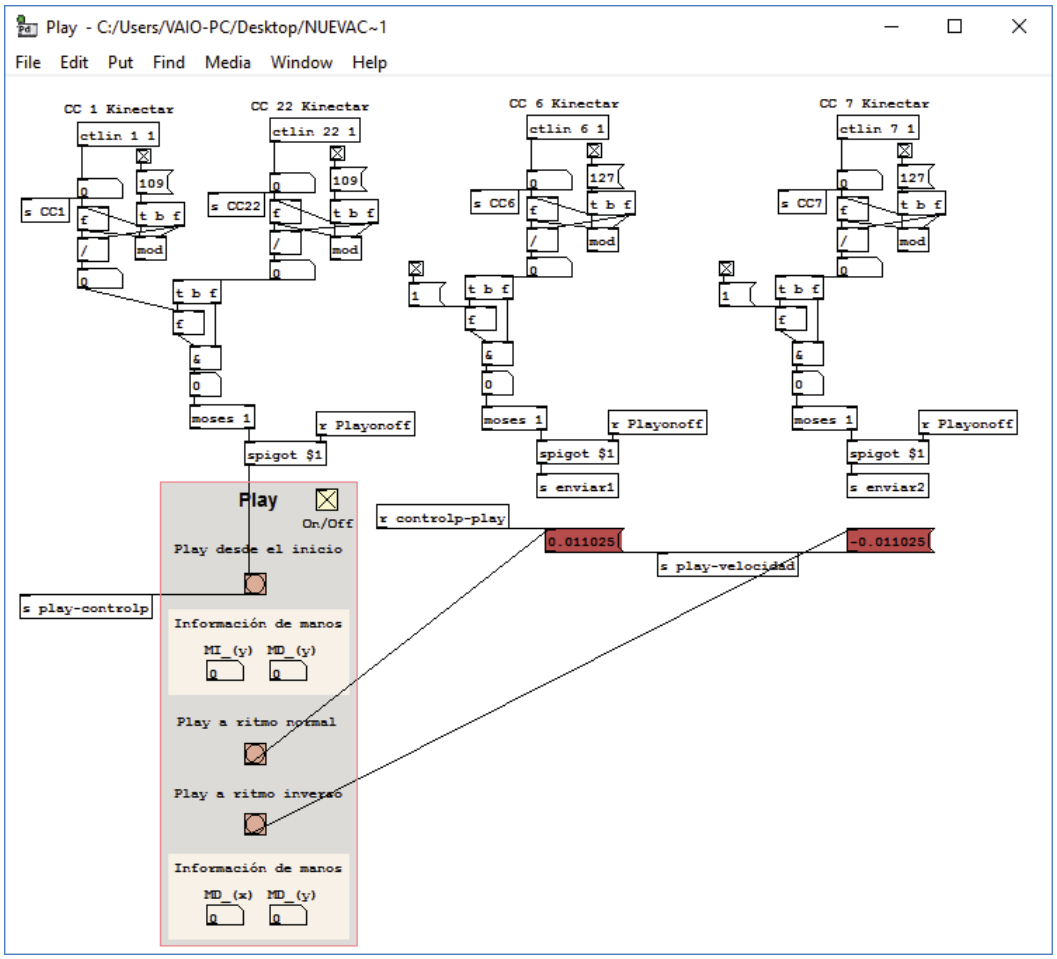

Figura 3-18: Programación del control del botón play del patch (fuente: elaboración propia)

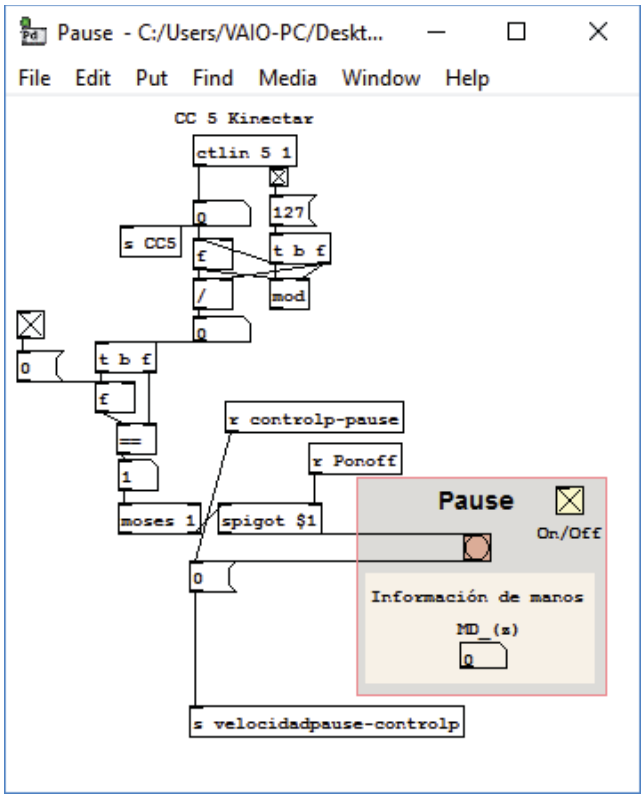

Figura 3-19: Programación del control del botón pause del patch (fuente: elaboración propia)

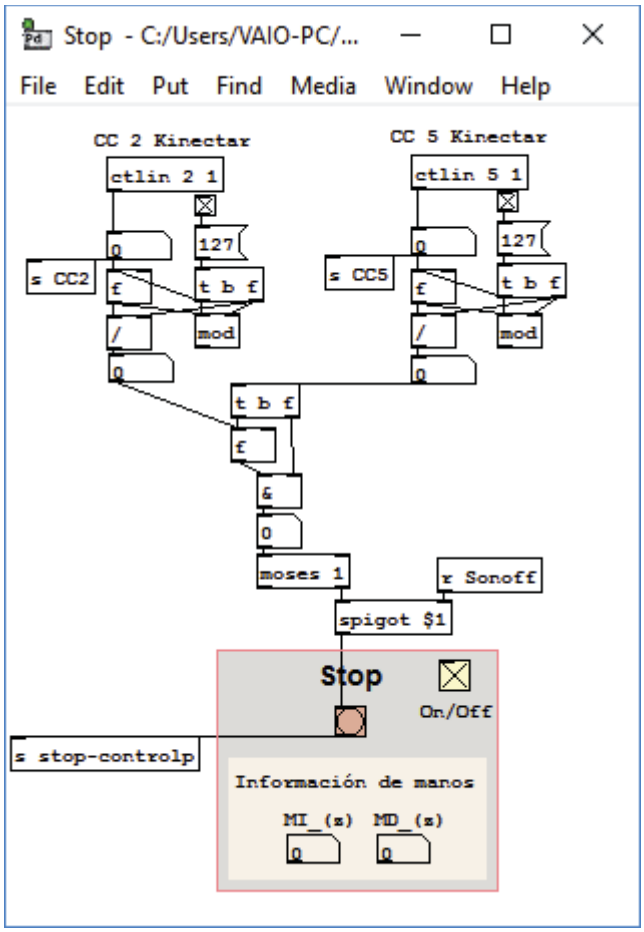

Figura 3-20: Programación del control del botón stop del patch (fuente: elaboración propia)

3. Control del volumen y velocidad de reproducción: En la Figura 3-21 se muestra el control del volumen y su funcionalidad es la misma que en los casos anteriores; con la salvedad de que, en este caso, su valor es controlado por una variable más visual (como lo es la barra que proporciona intuitivamente el aumento o disminución del parámetro).

   Similar a lo anterior, el control de la velocidad de reproducción se muestra en la Figura 3-22, y permite el control de la pista de audio en ritmo normal o invirtiendo el sonido.

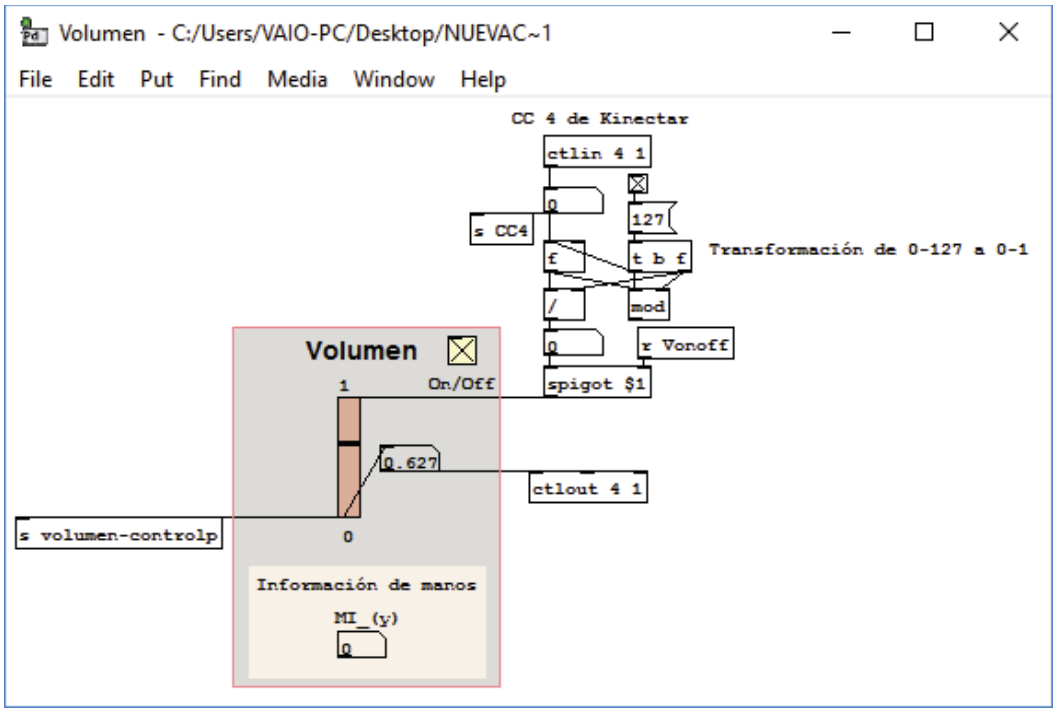

Figura 3-21: Programación del control del volumen del patch (fuente: elaboración propia)

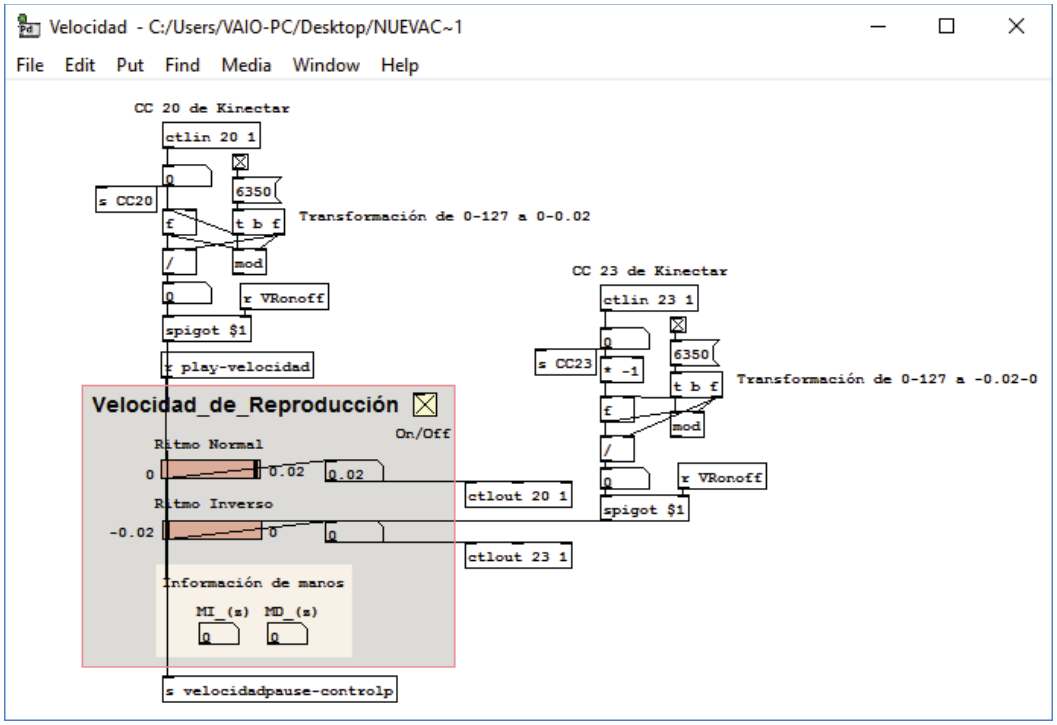

Figura 3-22: Programación del control de la velocidad de reproducción del patch (fuente: elaboración propia)

4. Control del filtro pasabanda: Se muestra en la Figura 3-23 y funciona de la misma manera que todos los parámetros anteriores, es decir, se obtienen los valores de Kinectar y se transforman a los que se requieren en Pure Data. Su diferenciación radica en que el "Control de batidos" –que es la cantidad de beats que genera el sonido– se maneja en forma manual (con el mouse u otro controlador MIDI), de manera de hacer más sencilla la tarea de trabajar con el filtro.

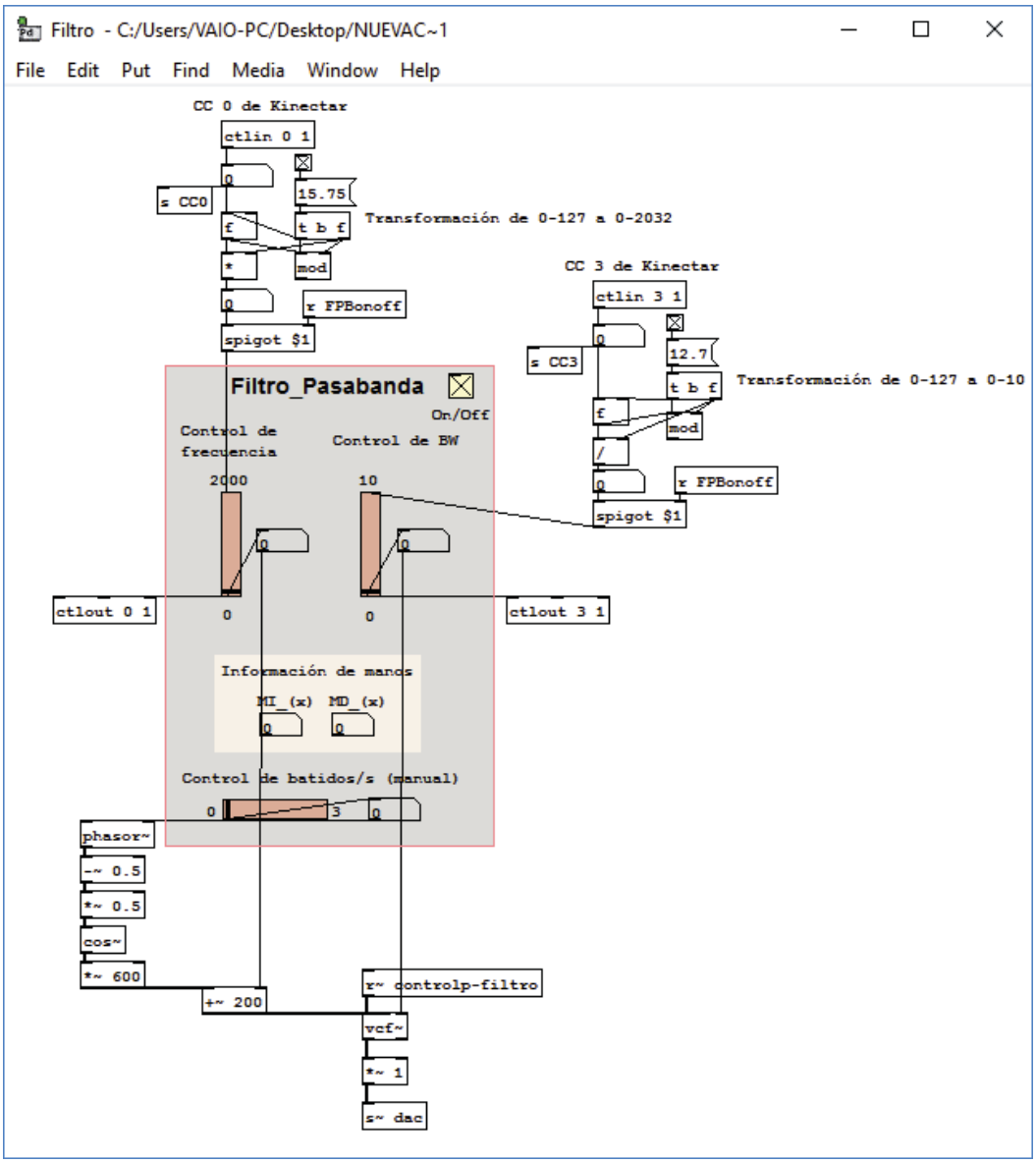

Figura 3-23: Programación del control del filtro pasabanda del patch (fuente: elaboración propia)

5. Presets [22]: Mostrados en la Figura 3-24. Su funcionalidad se basa principalmente en facilitar al usuario su performance en las presentaciones, debido a que permite grabar/leer estructuras del control de las pistas. Se conforma por cuatro variables: la primera (cuadro con el número 0) envía todos los valores de una pista a cero. Los siguientes tres (cuadros con los

números 1, 2 y 3) permiten la grabación de parámetros y su posterior lectura. Finalmente las casillas cuadradas cambian de modo (grabación o lectura), todo muy sencillo e intuitivo. Por defecto el modo de lectura viene incluido, es decir, el cuadro sin marcar. El manejo de los presets es de forma manual (mediante un mouse u otro controlador MIDI).

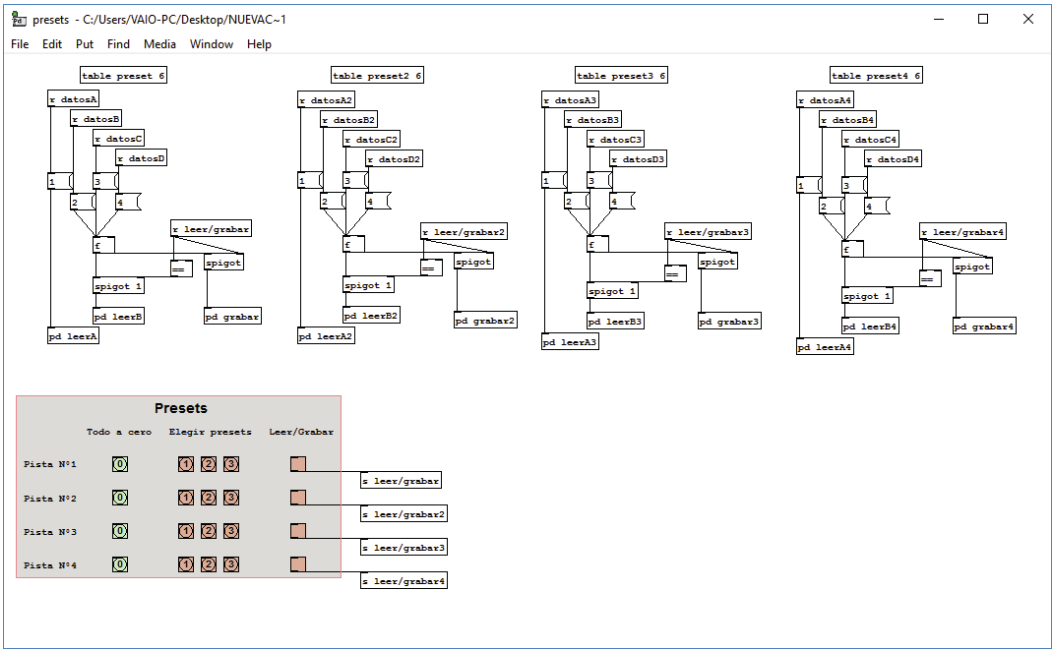

Figura 3-24: Programación de los presets del patch (fuente: elaboración propia)

Los cinco pasos antes descritos lograron la creación de la lectura y control de una de las pistas de audio, sin embargo, el método se replicó para tres pistas más. En la Figura 3-25 se muestra el progreso final. Por comodidad de visibilidad sólo se muestran el control de dos pistas, pero el método funciona para cuatro con simultaneidad.

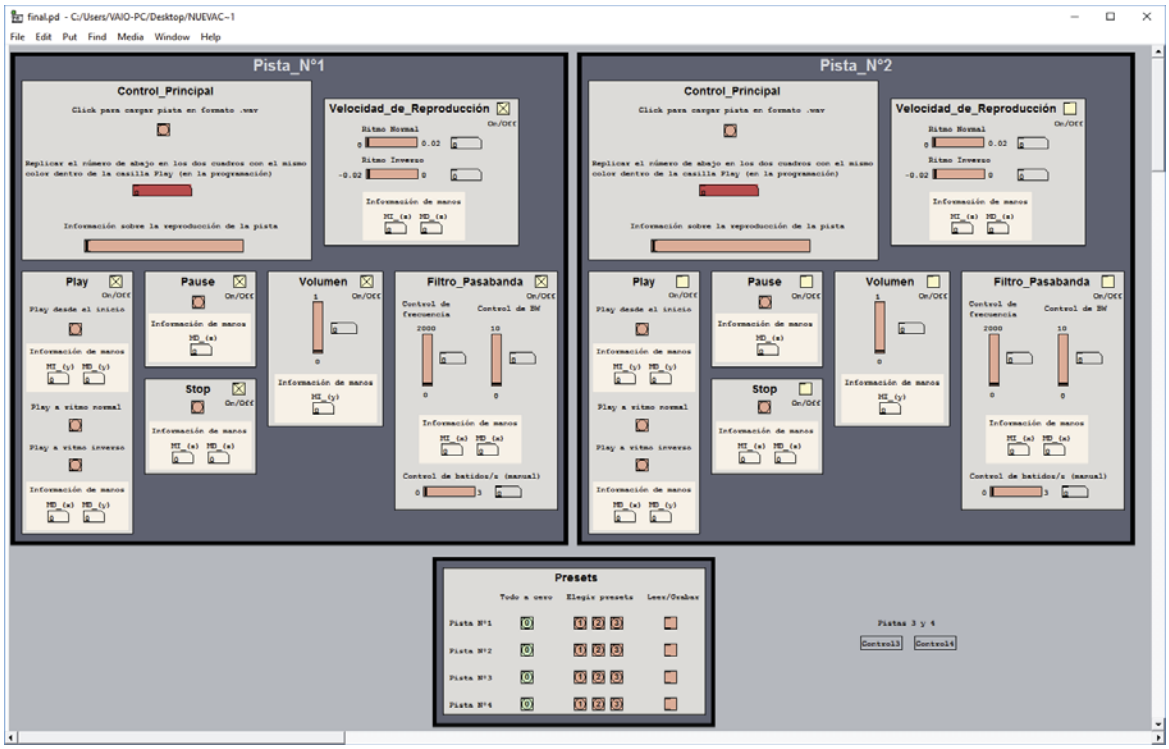

Figura 3-25: Progreso final del control de pistas (fuente: elaboración propia)

Kinectar

En este momento, se debe configurar Kinectar para que se vincule con el patch y se pueda controlar la reproducción de la pista. Por ahora, sólo interesa la forma de conexión de estos dos softwares, puesto que en el Capítulo 4 se fundamentará cómo interactuar con todos los programas que están inmersos en la GUI. A parte de explicar detalladamente como instalar lo necesario, comenzar a manipular la plataforma y ocupar el ejemplo dado.

En la Figura 3-26 se presenta un ejemplo de la manipulación de Kinectar para conectar los CC's hacia una salida MIDI de Pure Data. Los cuadros que están marcados con una "X" (mostrados en una casilla roja) son los canales desde los cuales Kinectar recibe datos MIDI de Pure Data para controlar algún parámetro. Por ejemplo, el primer ajuste (CC 0) indica que el brazo izquierdo efectuará una acción en el patch de Pure Data (supongamos el control de un filtro). Pero sólo responderá si el brazo se mueve a lo largo del eje x del plano cartesiano (horizontalmente). El resto de los ajustes (velocidad del movimiento o distancia entre las manos) se configura de manera similar.

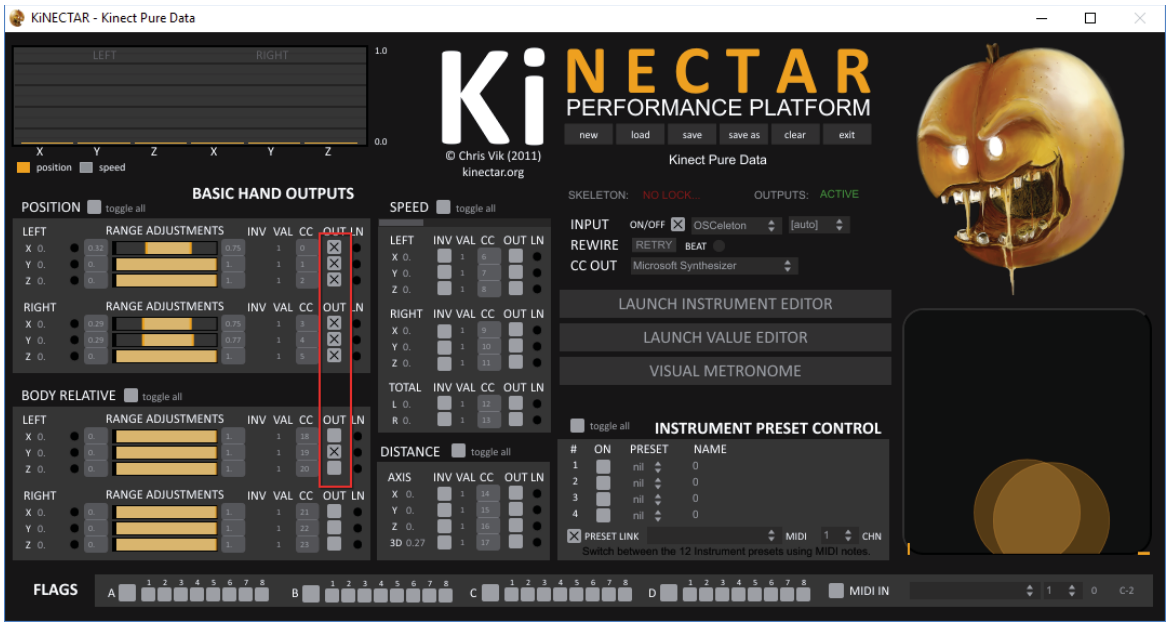

Figura 3-26: Configuración de Kinectar para manipular audio de Pure Data (fuente: elaboración propia)

Para comodidad del usuario se muestra un listado de valores mapeados desde Kinectar hacia Pure Data, de manera de facilitar la lectura o aprendizaje para quienes no conocen bien estas herramientas.

Tabla 3-1: Listado de los valores mapeados desde Kinectar hacia Pure Data

Parámetro de Pure Data	Comando Pure Data	CC de Kinectar
Play desde el inicio	ctlin 1 – ctlin 22	CC 1 – CC 22
Play a ritmo normal	ctlin 6	CC 6
Play a ritmo inverso	ctlin 7	CC 7
Pause	ctlin 2 – ctlin 5	CC 2 – CC 5
Stop	ctlin 2 – ctlin 5	CC 2 – CC 5
Volumen	ctlin 4 – ctlout 4	CC 4

Velocidad de reproducción a ritmo normal	ctlin 20 – ctlout 20	CC 20
Velocidad de reproducción a ritmo inverso	ctlin 23 – ctlout 23	CC 23

4 Resultados

Luego de haber creado la GUI y haber definido todos los parámetros necesarios para su entendimiento se debe ejecutar y explicar la plataforma desarrollada. Sin embargo, es importante tener en cuenta que algunos softwares son dependientes de librerías y/o aplicaciones, por lo tanto, antes de iniciar la GUI se deberá instalar todo lo obligatorio para que se ejecute de manera correcta.

4.1 Instalación de librerías y controladores

Ingresar el CD (que viene adjunto) a la unidad de discos y copiar la carpeta GUI directamente en el disco C. Es decir, en la ruta → ***C:\.*** Tal como se muestra en la Figura 4-1.

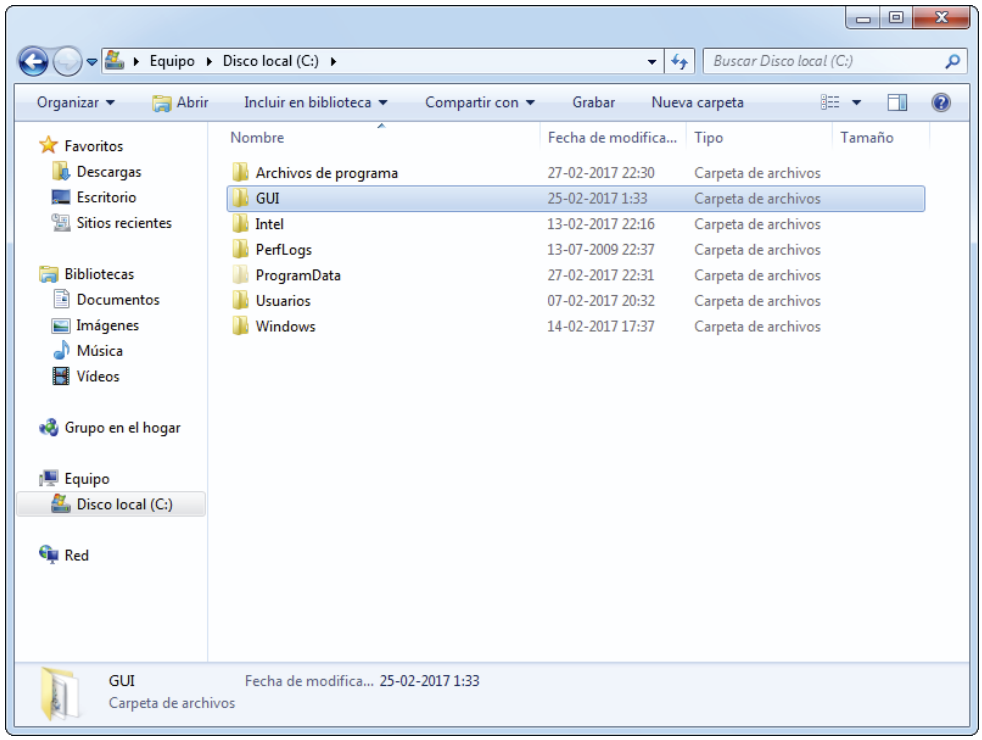

Figura 4-1: Ruta de destino de la carpeta GUI (fuente: elaboración propia)

Luego, ingresar a la carpeta Instalación → ***C:\GUI\Instalación*** y ejecutar el archivo ***Drivers.exe***, marcar la casilla seleccionada como se muestra en la Figura 4-2 y hacer clic en ***Install.*** Esto instalará los controladores de OpenNI necesarios para que el ordenador sea capaz de percibir a la Kinect.

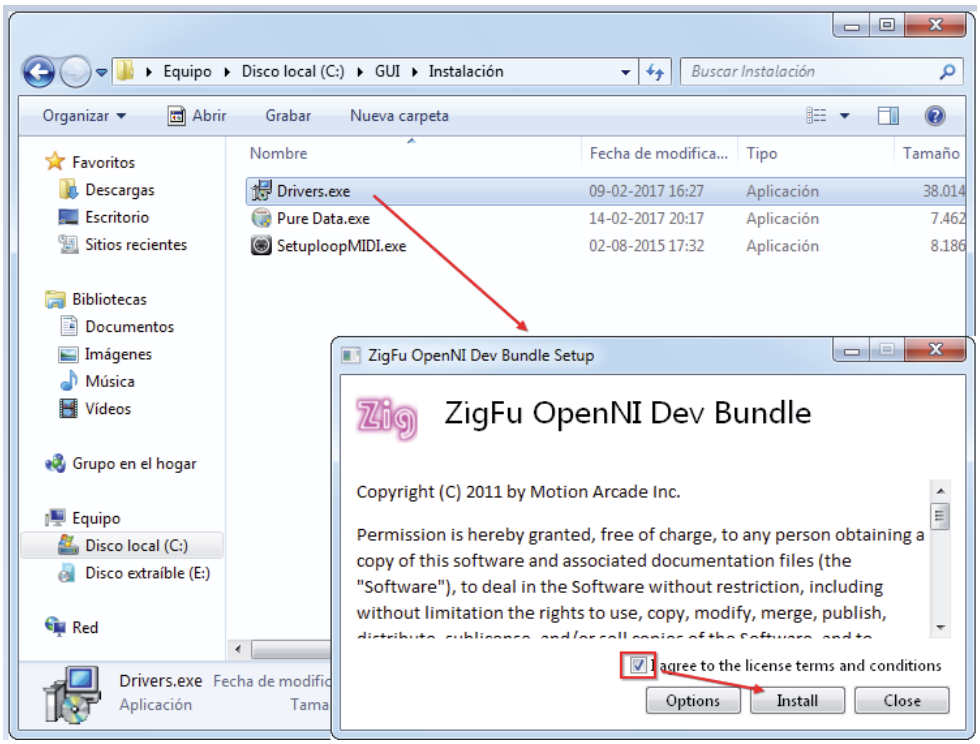

Figura 4-2: Instalación de los controladores de Kinect (fuente: elaboración propia)

Para comprobar si todo resultó correctamente, se debe conectar mediante cable USB la Kinect al computador e ir a la ruta → ***Panel de control\Sistema y seguridad\Sistema***. Hacer clic en ***Administrador de dispositivos*** y verificar que se encuentren instalados los controladores ***PrimeSense*** como se ve en el recuadro en rojo de la Figura 4-3. La acción anterior se puede contrastar revisando que encienda y parpadee una luz verde en la Kinect, lo que indica que está siendo correctamente leída.

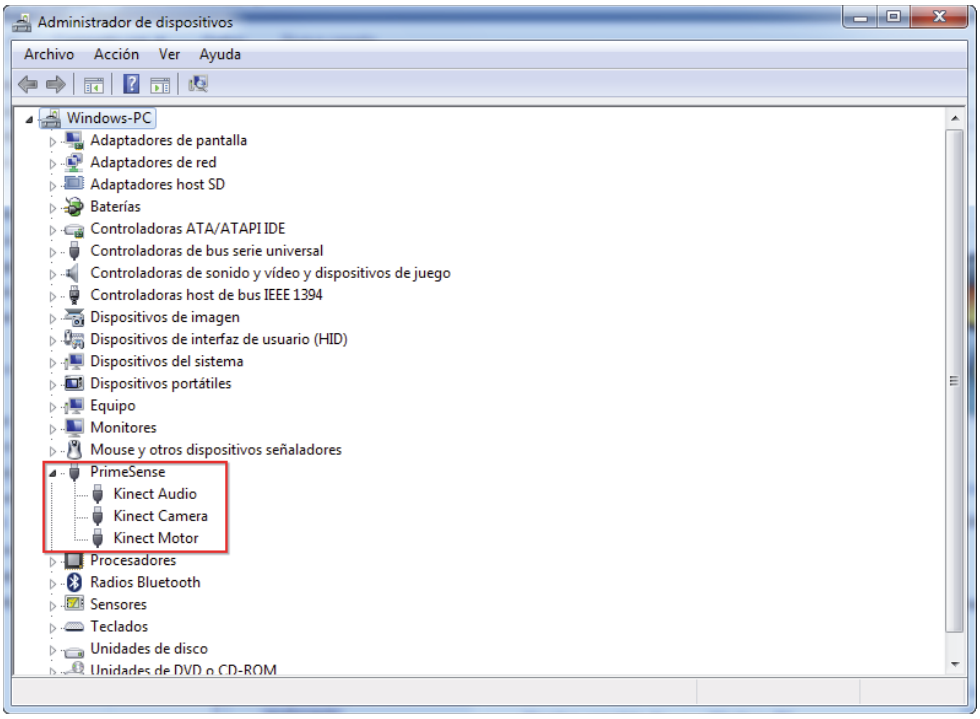

Figura 4-3: Comprobación de la correcta instalación de los controladores (fuente: elaboración propia)

4.2 Instalación de softwares

De la misma forma que se instaló el archivo *Drivers.exe* en la sección 4.1, se deben ejecutar e instalar los archivos ***Pure Data.exe*** y ***SetuploopMIDI.exe***, ubicados en la misma ruta → ***C:\GUI\Instalación***, como se aprecia en la Figura 4-4.

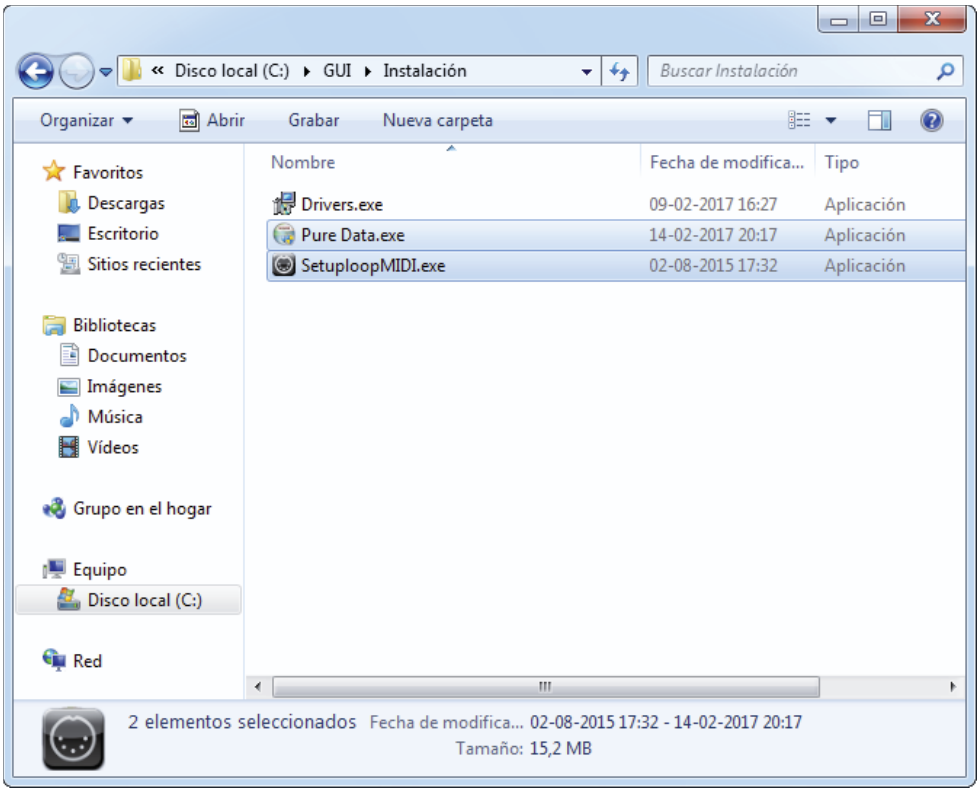

Figura 4-4: Instalación de los programas requeridos (fuente: elaboración propia)

4.3 Montaje y ejecución

Luego de que la base del funcionamiento de la plataforma ha sido correctamente instalada, sólo resta ejecutar la interfaz ***GUI.exe***, ubicada en la ruta → ***C:\GUI***. Si se quiere, se puede cambiar el nombre de *GUI.exe* al que se desee. De igual manera cambiar la ruta de ejecución (copiar y pegar el archivo ejecutable en el escritorio, por ejemplo).

Una vez que se ejecutó el archivo *GUI.exe*, se debe esperar un par de segundos para que se lancen todos los programas que conforman la GUI. Se sabrá que está todo cargado en cuanto se cierren todos los archivos que se auto ejecutaron y la plataforma se muestre como una ventana en solitario. Tal como se muestra en la Figura 4-5.

Figura 4-5: Interfaz ejecutada (fuente: elaboración propia)

Lo primero que aparecerá en la GUI será la pestaña Instrucciones. Desde esta pestaña se tendrá información acerca de cómo cargar el ejemplo dado, además de algunos consejos para desarrollar programas propios con las herramientas que ofrece la interfaz.

Una vez cargada la interfaz, se debe ir a la pestaña LoopMIDI. Se debe hacer clic en el recuadro "+" para agregar un nuevo puerto MIDI virtual (como se muestra en la Figura 4-6 dentro de la casilla marcada con rojo), y si se desean agregar más se debe seguir presionando hasta agregar los puertos deseados. Análogamente cliqueando en el recuadro "-" se eliminan los puertos no queridos.

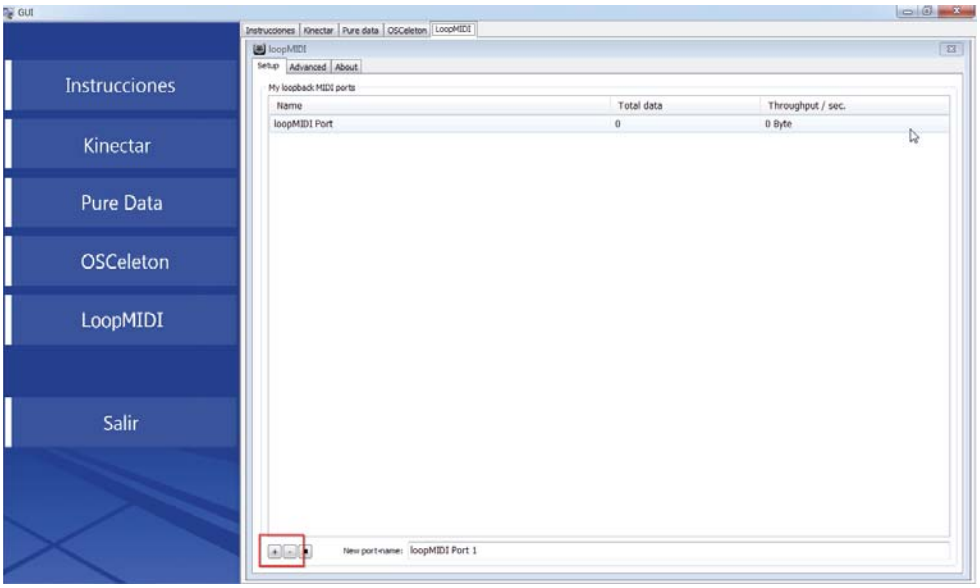

Figura 4-6: Pestaña LoopMIDI en ejecución (fuente: elaboración propia)

Luego de saber que el puerto MIDI está en ejecución se debe ir a la pestaña Pure Data y presionar las teclas "Control + O" para cargar el archivo de Ejemplo ***Kinect.pd***, ubicado en la ruta → ***C:\GUI\Ejemplo*** como se muestra en la Figura 4-7.

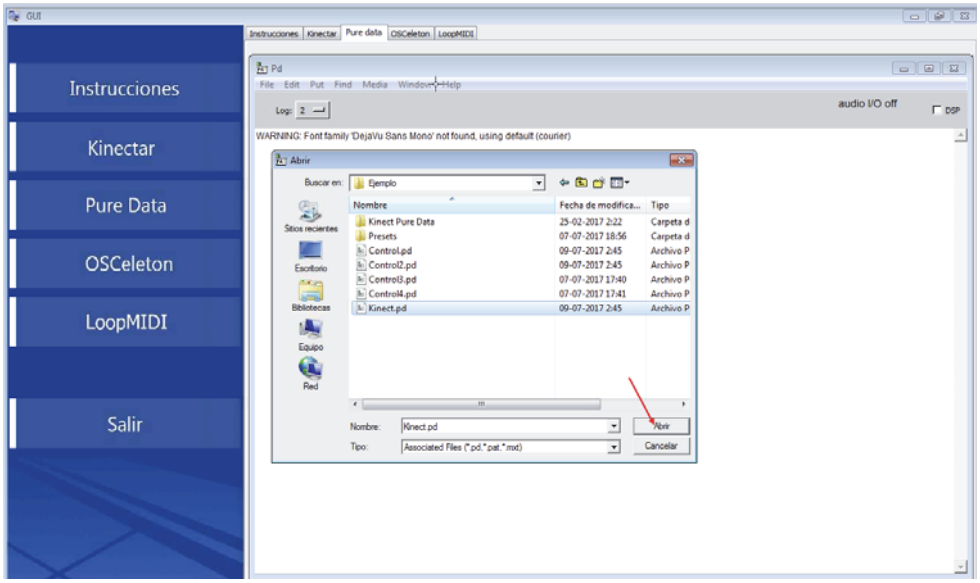

Figura 4-7: Carga del archivo de ejemplo (fuente: elaboración propia)

Esta acción abrirá una nueva ventana que contiene un patch creado como ejemplo, muy intuitivo y sencillo de operar. Para cargar una pista de audio sólo se debe guiar por las instrucciones del patch y empezar a utilizarlo. En este caso, la pista de ejemplo se ubica en la ruta →

C:\GUI\Ejemplo y corresponde al archivo ***Pista de audio.wav***, como se muestra en la Figura 4-8.

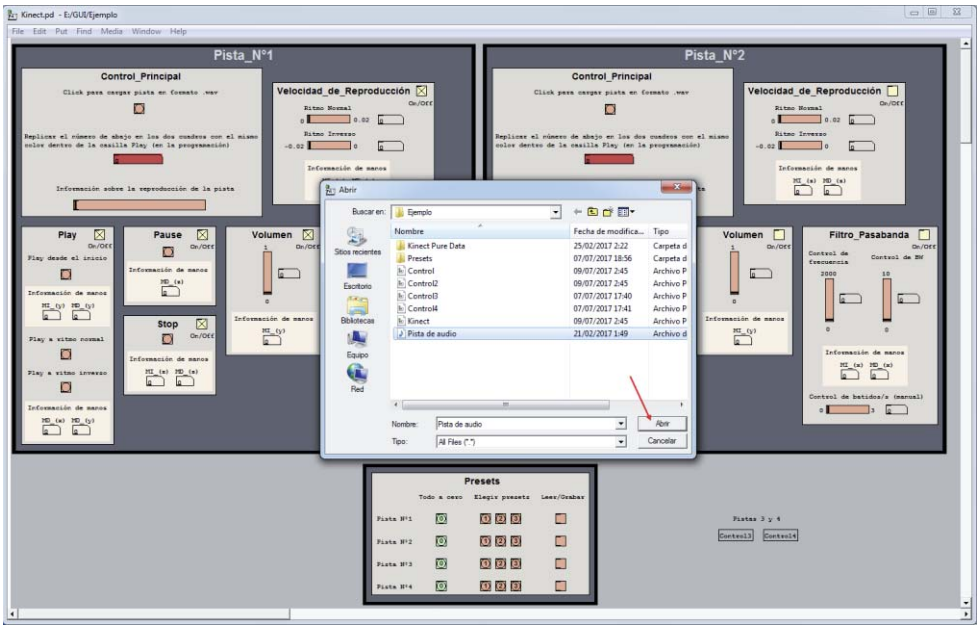

Figura 4-8: Carga del archivo ejemplo Pista de audio.wav (fuente: elaboración propia)

Finalmente, se debe ir a la pestaña Pure Data en la GUI para establecer la entrada del puerto MIDI virtual "loopMIDI Port" desde la opción ***Media/MIDI Settings...***. Notar que sólo se debe seleccionar en la casilla de ***Input Devices***, tal como se muestra en la Figura 4-9. Esta acción interconectará los softwares para manipular contenido musical. En definitiva, se sintoniza a Pure Data para establecer la conexión a Kinectar y controlar la pista de audio del ejemplo.

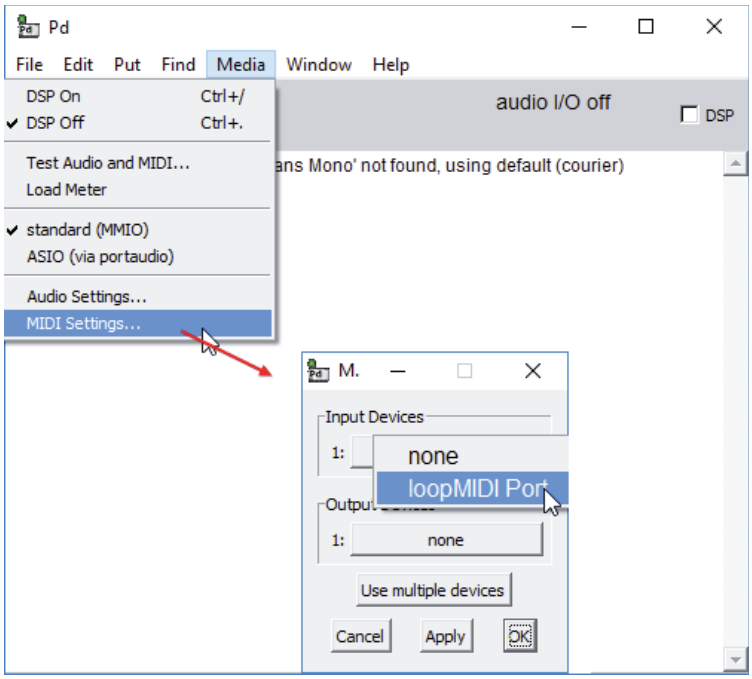

Figura 4-9: Establecimiento de “loopMIDI Port” como entrada (fuente: elaboración propia)

Por otro lado, se debe cargar otro archivo de ejemplo. Esta vez en Kinectar, por tanto se debe hacer clic en la pestaña correspondiente al programa. En este software no hay problemas con la carga del puerto MIDI virtual “loopMIDI Port”, debido a viene cargado por defecto cuando se ejecuta la interfaz, por lo que no es necesario modificar nada. No obstante, igual se debe revisar que en ***CC OUT*** esté configurado ***“loopMIDI Port”*** (mostrado con un número 1 en la Figura 4-10).

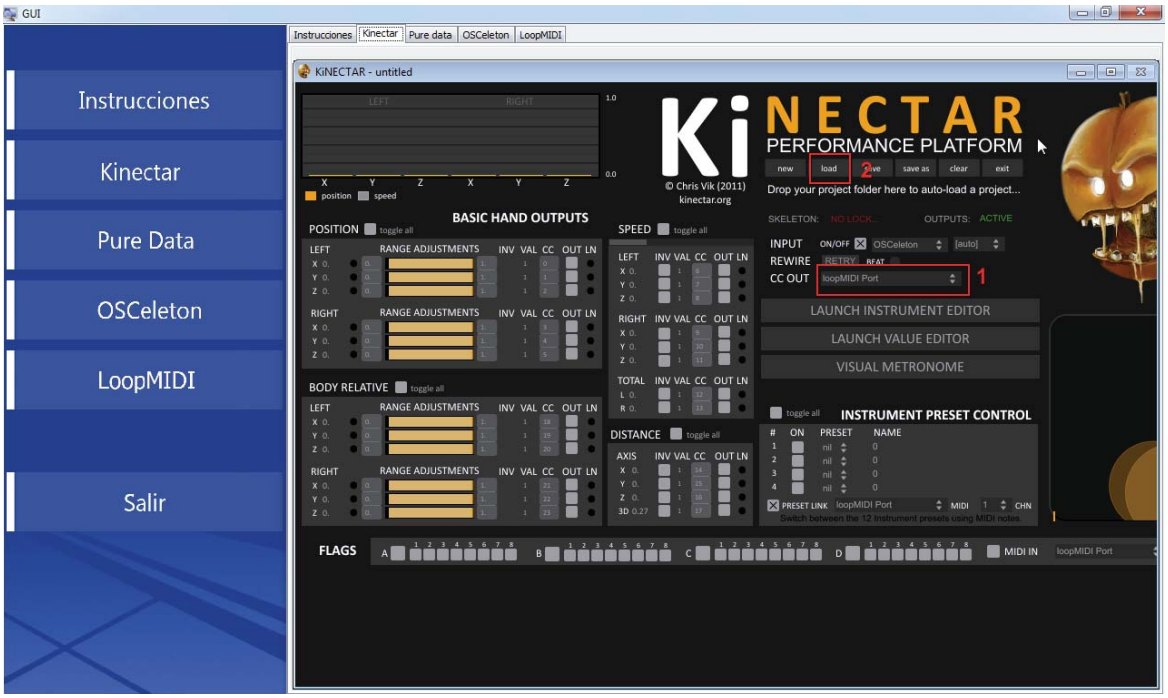

Figura 4-10: Revisión de CC OUT y carga del archivo de configuración de ejemplo (fuente: elaboración propia)

Posteriormente se debe cargar la información del ejemplo, ubicada en la carpeta Ejemplo en la ruta → ***C:\GUI\Ejemplo*** la forma de hacerlo es cliqueando el botón ***load*** y cargando la carpeta ***Kinect Pure Data***. Como se muestra en la Figura 4-11.

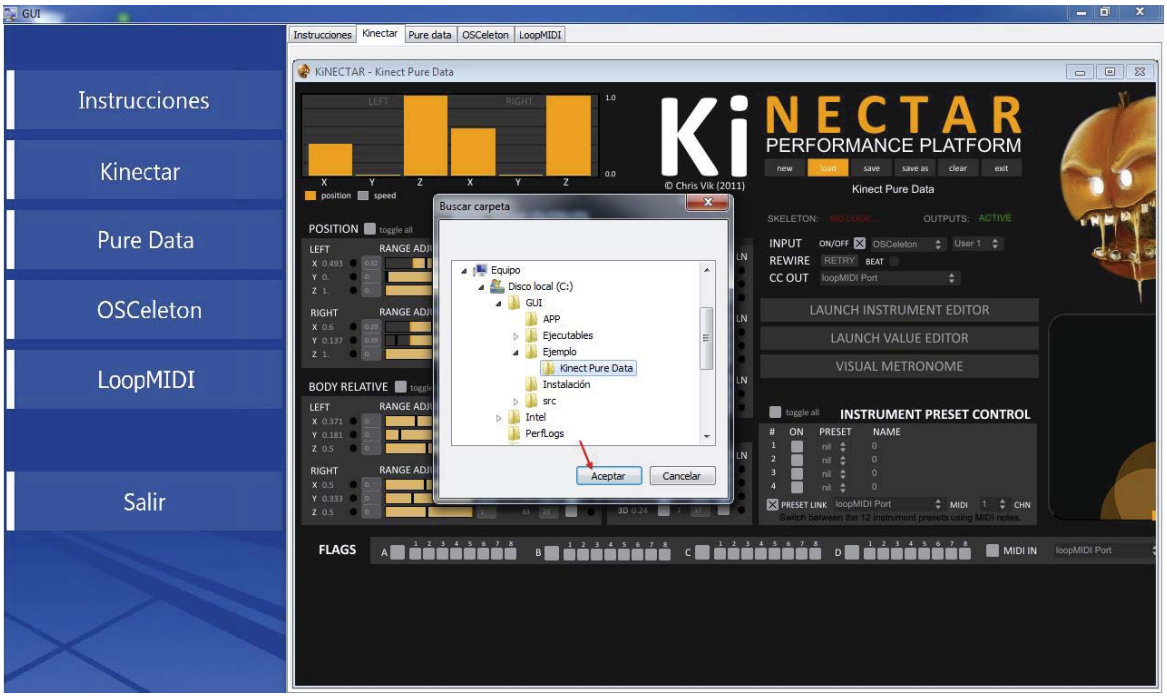

Figura 4-11: Carga del archivo de configuración de ejemplo (fuente: elaboración propia)

Una vez cargado el archivo de ejemplo solo resta establecer la conexión entre Kinectar y OSCeleton. Si se analiza Kinectar con detención, en la casilla ***INPUT*** se puede ver que aparece configurado OSCeleton por defecto, mientras que en ***SKELETON*** aparece ***"NO LOCK..."*** (O sin conexión). La forma de conectarlo es adoptando la pose "psi"[16], como se ve en la silueta de la Figura 4-12. Se puede revisar en la pestaña OSCeleton si se realiza correctamente, ya que se formarán líneas a través de la silueta del cuerpo o se puede repetir la pose hasta que en Kinectar aparezca ***"TRACKING"*** en la casilla ***SKELETON***, como se observa en la Figura 4-13.

16 Denominada así por su parecido con el símbolo matemático psi.

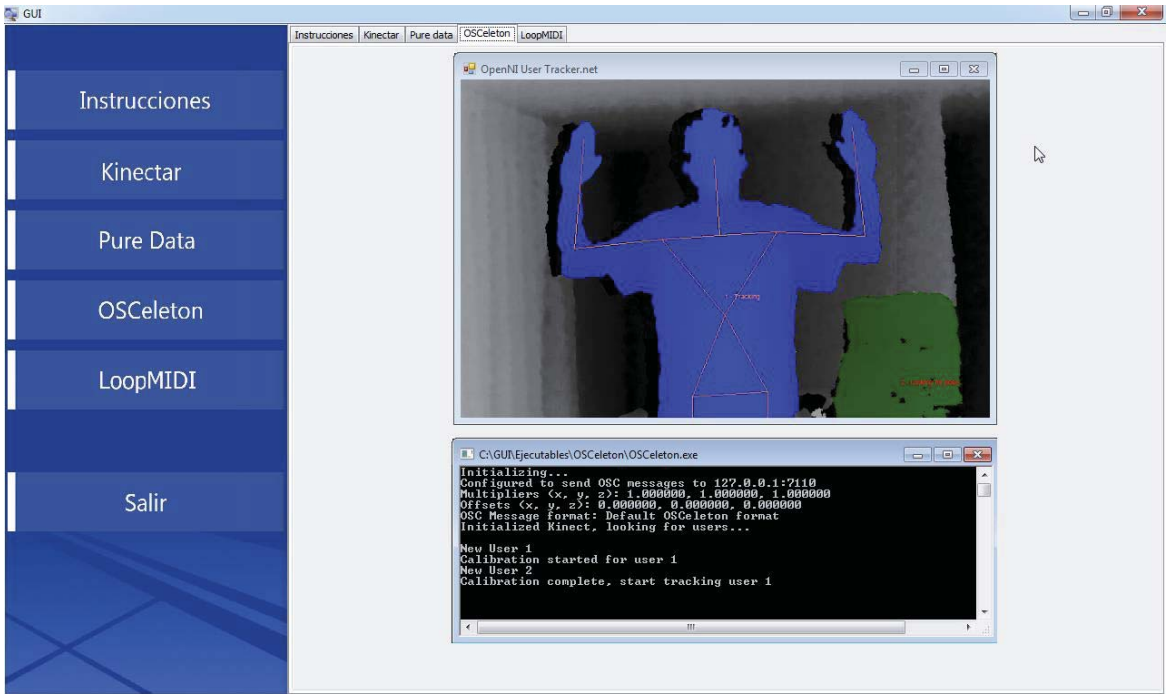

Figura 4-12: Pose "psi" mostrada por OSCeleton (fuente: elaboración propia)

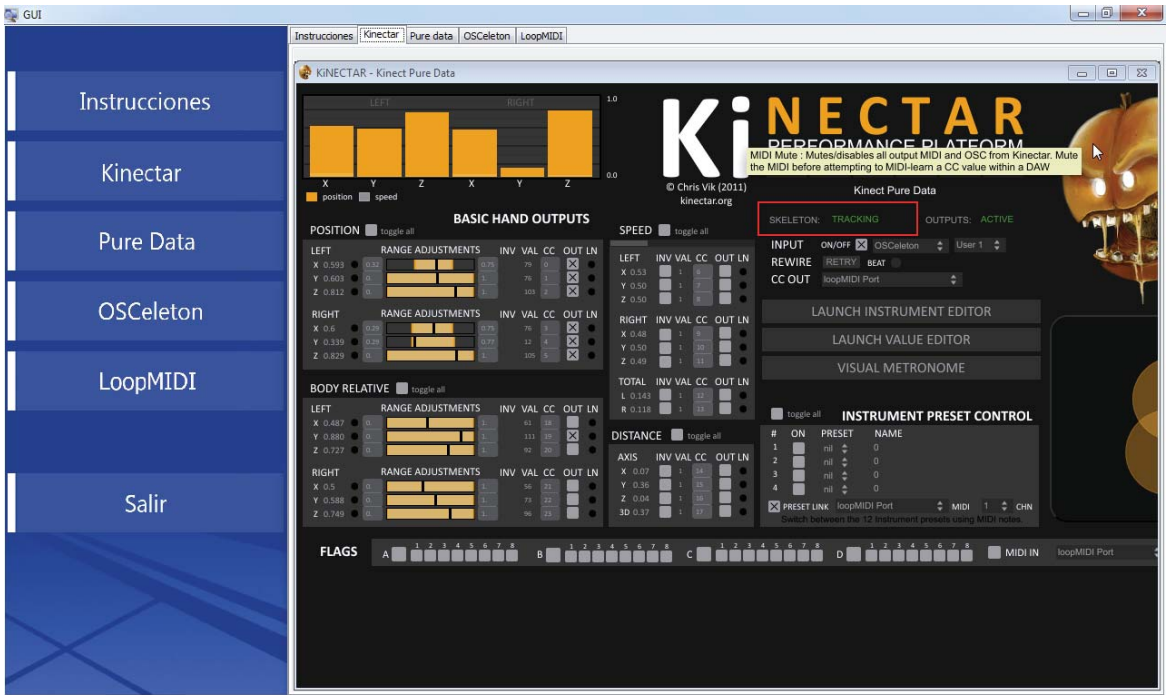

Figura 4-13: Kinectar leyendo el movimiento del cuerpo mediante "TRACKING" (fuente: elaboración propia)

En este punto, se encuentran todos los softwares interconectados, por lo que es momento de comenzar a modificar el sonido. A modo de aclaración:

1. Botón play: Se activa al levantar los dos brazos simultáneamente. Mostrado en la Figura 4-14.

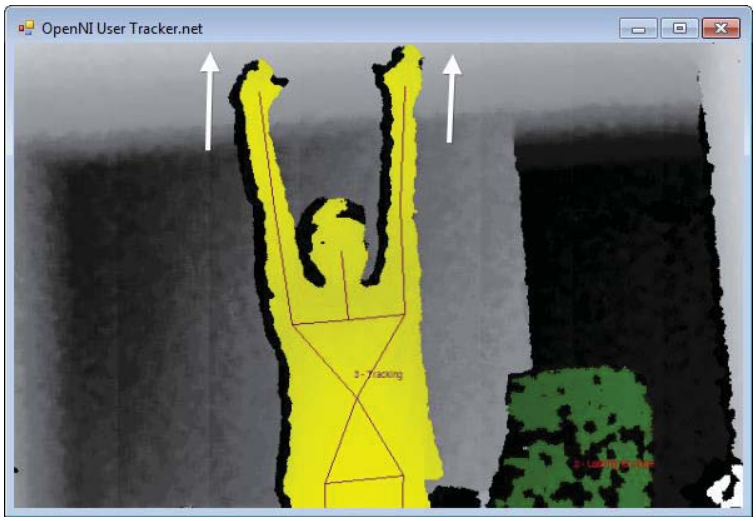

Figura 4-14: Ejemplo de activación del botón play (fuente: elaboración propia)

2. Botón stop: Se activa al enviar los dos brazos simultáneamente hacia adelante. Mostrado en la Figura 4-15.

Figura 4-15: Ejemplo de activación del botón stop (fuente: elaboración propia)

3. Botón pause: Se activa al enviar los dos brazos simultáneamente hacia atrás. Mostrado en la Figura 4-16.

Figura 4-16: Ejemplo de activación del botón pause (fuente: elaboración propia)

4. Control de filtro pasabanda: Manejado por ambos brazos moviéndose a lo largo del eje "x" cartesiano (horizontalmente). Mostrado en la Figura 4-17.

   - Mano izquierda: ancho de banda.
   - Mano derecha: frecuencia.

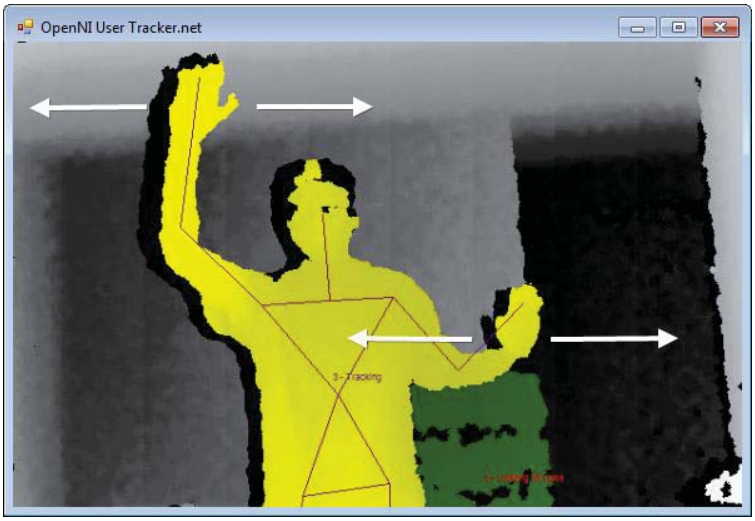

Figura 4-17: Ejemplo de control de filtro (fuente: elaboración propia)

5. Control de volumen: Manejado por el brazo izquierdo en el eje “y” cartesiano (verticalmente). Mostrado en la Figura 4-18.

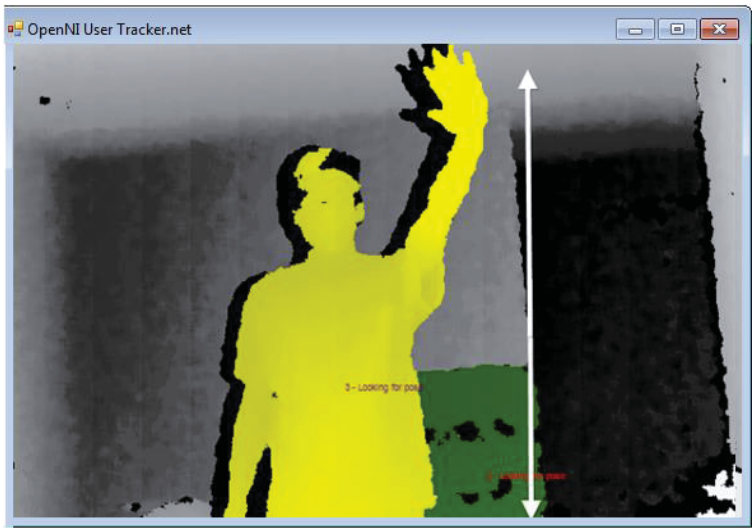

Figura 4-18: Ejemplo de control de volumen (fuente: elaboración propia)

6. Control de velocidad de reproducción (VR): Manejado por ambos brazos moviéndose a lo largo del eje “z” cartesiano (horizontalmente hacia adelante y atrás). Mostrado en la Figura 4-19.

    - Mano izquierda: Aumento y disminución de velocidad de reproducción en sentido clásico.
    - Mano derecha: Aumento y disminución de velocidad de reproducción en sentido inverso.

Figura 4-19: Ejemplo de control de la VR (fuente: elaboración propia)

Discusión y conclusiones

Es fácil darse cuenta que Kinect abre un mundo de alternativas de creación de nuevas ideas innovadoras, educativas, comerciales, gubernamentales, sociales, entre otras. Las múltiples plataformas de programación la hacen ser el futuro en tecnología, al estar en continuo desarrollo y evolucionando (por ejemplo en su segunda versión). Esto hace pensar que quizá sea uno de los dispositivos electrónicos más importantes a nivel mundial, ya que con su presencia en distintas ferias de tecnología, con una variedad de proyectos cada año, ayuda a reafirmar esta tendencia.

Muchos desarrolladores aficionados, científicos e inventores, se han visto motivados en la creación de nuevas ideas que mejoran alguna problemática existente en torno a la música y que además aportan con nuevas alternativas para innovar en este aspecto. Esto es de gran utilidad, debido a que forman parte de la base que abre nuevos caminos para la proyección de opciones diferentes y modernas de desempeño musical en los escenarios, lo que, hasta este momento, se ha visto disminuido.

Un aspecto positivo que se puede rescatar de la unión música-tecnología (de Kinect específicamente) es que, en ambos sectores, existe un sinfín de softwares que hacen que la compatibilidad sea aún con mayor facilidad. Es decir, una gran cantidad de plataformas de programación de Kinect, mezcladas con otra considerable cuantía de softwares musicales, y además agregando su conexión a la interfaz –que en este caso es el hardware de Kinect– logran que esta idea relativamente actual de música-tecnología escénica sea una inspiración por mucho tiempo más.

Analizando con mayor profundidad, Pure Data es una potente herramienta para la creación de proyectos musicales creativos. Su principal ventaja es que es de código abierto, entonces posee muchos desarrolladores que le brindan soporte y tutoriales ante cualquier problemática, además de crear continuamente nuevas librerías para mejorar aún más este software. Sin embargo, para contrastar su uso con Kinect, de una manera más precisa, es necesario que trabaje con dos herramientas que también son fundamentales y es ahí donde se debe empezar un proceso de desarrollo para actualizar softwares que están pronto a perder compatibilidad.

La interfaz de usuario desarrollada termina siendo una herramienta muy útil para quienes deseen empezar a trabajar con Kinect y un entorno musical sin tener idea alguna sobre cómo empezar. Su función es la de facilitar la tarea de tener que organizar por cuenta propia cada uno de los

programas y evitar abrir tantas ventanas. Mejora el orden y la visualización, evitando que el usuario se pierda intentando utilizar los programas necesarios para sus metas propuestas. Incluso se pueden seguir agregando pestañas para ampliar el concepto de trabajo con Kinect y música. Como mostrar una visualización que se contraste con el sonido y se proyecte escénicamente. Se puede pensar en múltiples situaciones que abarquen el uso de las herramientas descritas, puesto que Kinect sigue en constante desarrollo y se actualiza por desarrolladores independientes.

Dos de las herramientas computacionales usadas no son manejadas en el campo de la ingeniería electrónica (Adobe Photoshop y Adobe Illustrator específicamente), sin embargo para desarrollar esta plataforma fueron muy útiles para lograr que la interfaz sea más amigable e intuitiva, ya que permitieron añadir colores, diseños e implementar casillas y otras funcionalidades que mejoran considerablemente la tarea de bosquejar una GUI sin que su vista sea como la de un programa que no es dirigida a un usuario que desconoce de cómo utilizar nuevas herramientas.

Se pretende seguir mejorando esta primera versión, hasta llegar a un acabado prolijo y de mejor calidad, de manera de lograr que su entorno se visualice en alta resolución y que el usuario no tenga problemas de ningún tipo para su utilización.

Por otra parte no se puede desconocer que los comandos ctlin-ctlout de Pure Data son de suma importancia: primero, para lograr la conexión de este software con Kinect; y segundo, para realizar un seguimiento preciso de los datos MIDI enviados desde otro software a Pure Data. Cumple funciones muy variadas, y que son necesarias para interconectar protocolos MIDI no sólo para los objetivos de este proyecto, sino para otros propósitos que tengan que ver con la interconexión de este interesante protocolo. En este documento fue posible evaluar distintas alternativas que generarán en el corto plazo una ayuda a la precisión de Kinect con respecto a los movimientos, y a la creación de estilos musicales.

Se espera seguir trabajando con este proyecto en el futuro para mejorar y hacer más simple su uso para nuevos usuarios. Por otro lado, se desea plantear como desafío (o propuesta) –a desarrolladores interesados en retomar este proyecto– actualizar las librerías de OpenNI para la última versión de Windows y concientizar acerca del proceso de crear nuevos planes utilizando el SDK de Windows, puesto que se mantiene más activo y actualizado. Además de que es el Kit de desarrollo del futuro por la reciente actualización de la versión de Microsoft Kinect (v2). Finalmente, combinar la tecnología con el despliegue escénico es un factor muy interesante y que podría generar bastante expectativa en un proyecto como este, por lo que también se puede seguir ampliando agregándole herramientas que permitan la proyección de los movimientos artísticamente hacia el público, mediante la programación con softwares aptos para este tipo de trabajos (como openFrameworks o Processing), de manera de aportar para que cada vez pueda ser una propuesta más cautivadora y quizá en algún momento inclusiva.

Bibliografía

[1] J. Basalo Gonzalez, A. Blanco Blanco, J. Cambeiro Formoso, V. García G. del Barbo y J. M. Rey Martinez, «Usos alternativos de Kinect,» [En línea]. Available: http://sabia.tic.udc.es/. [Último acceso: 21 abril 2016].

[2] J. Enterprises Ltd., «Guitarra, Arte Pulsado,» [En línea]. Available: http://guitarra.artepulsado.com/foros/showthread.php?6630-Macintosh-el-ordenador-ideal-para-los-m%FAsicos&s=45eab7a117f5606da8e2b3462e497669/. [Último acceso: 15 abril 2017].

[3] A. Rivera Fernández, Interviewee, *Problemáticas de un músico escénicamente.* [Entrevista]. 24 abril 2016.

[4] C. Vik, B. Hammond y S. Burns, «Ethno Tekh,» [En línea]. Available: http://ethnotekh.com/. [Último acceso: 21 abril 2016].

[5] Fragment.in, «Computer Orchestra,» [En línea]. Available: http://computer-orchestra.com/. [Último acceso: 22 abril 2016].

[6] T. Thompson, «Space Palette,» [En línea]. Available: http://spacepalette.com/. [Último acceso: 24 abril 2016].

[7] J. Nusz, «Custom Logic Web Blog,» [En línea]. Available: http://custom-logic.com/. [Último acceso: 24 abril 2016].

[8] «MOTIV,» [En línea]. Available: http://musicwithmotiv.com/. [Último acceso: 27 abril 2016].

[9] F. J. Levancini Garces, «Sensor Kinect,» de *Diseño de aplicaciones Kinect para educación: Un sistema virtual del cuerpo humano*, Valparaíso, 2014, pp. 6, 9, 15, 16.

[10] B. Widenhofer, «EETimes,» Austin, 2010.

[11] R. Bogdan Rusu, E. Cohen, T. Shimizu Washio, M. Bell, E. Berger y R. Wang, «OpenNI, The standart framework for 3D sensing,» [En línea]. Available: http://openni.ru/. [Último acceso: 13 junio 2016].

[12] «OpenNI: la alternativa open source para interaccionar con Kinect,» [En línea]. Available: http://animusproject.wix.com/web/apps/blog/openni-la-alternativa-open-source-para-hablar. [Último acceso: 14 junio 2016].

[13] J. Jared St., «How the Kinect Was Hacked,» de *Kinect Hacks*, O'Reilly, 2013, pp. 3-4.

[14] C. Vik, «Ethno Tekh,» [En línea]. Available: http://www.ethnotekh.com/software/kinectar/. [Último acceso: 11 abril 2017].

[15] M. Puckette, «Pure Data,» [En línea]. Available: https://www.puredata.info/. [Último acceso: 24 noviembre 2016].

[16] Sensebloom, «GitHub Repository,» [En línea]. Available: https://www.github.com/Sensebloom/OSCeleton. [Último acceso: 12 febrero 2017].

[17] T. Erichsen, «Tobias Erichsen,» [En línea]. Available: http://www.tobias-erichsen.de/software/loopmidi.html. [Último acceso: 12 febrero 2017].

[18] I. Embarcadero Technologies, «Embarcadero Delphi,» [En línea]. Available: https://www.embarcadero.com/es/products/delphi. [Último acceso: 15 febrero 2017].

[19] A. Photoshop, «Preguntas frecuentes,» [En línea]. Available: https://helpx.adobe.com/la/photoshop/faq.html. [Último acceso: 09 enero 2017].

[20] A. Illustrator, «Preguntas frecuentes,» [En línea]. Available: https://helpx.adobe.com/es/illustrator/faq.html. [Último acceso: 09 enero 2017].

[21] R. Hernández, «YouTube,» [En línea]. Available: https://www.youtube.com/channel/UC-RatzHn1ukuuINLqnbBYeg/. [Último acceso: 14 julio 2017].

[22] C. Nervi, «YouTube,» [En línea]. Available: https://www.youtube.com/channel/UCqGJ1KPAosIDvzVV-jyRI8Q/. [Último acceso: 14 julio 2017].

A Apéndice

A.1 Índice de figuras

A.2 Índice de tablas

A.3 Códigos de programación

A.3.1 Embarcadero Delphi 7

Listado A-1: Código de programación Embarcadero Delphi 7

```
{*-------------------------------------------------------------------------------
  @Autor     : ViMX
  @Sitio web : www.voodoo-studios.com
  @Editor    : Cristian Tapia Macaya
  @Version   : Alpha
  -------------------------------------------------------------------------------
*}
{$IFDEF DEBUG}
  {$OPTIMIZATION OFF}
{$ENDIF}
unit unMain;
{*=============================================================================*}
interface
uses
Tlhelp32,
  Windows,
  Messages,
  SysUtils,
  Variants,
  Classes,
  Graphics,
  Controls,
  Forms,
  Dialogs,
  ComCtrls,
  StdCtrls,
  ExtCtrls,
  ImgList, jpeg;

Type
  TFrmMain = class(TForm)
    gpxWindows  : TGroupBox;
    LvwWindows  : TListView;
    imgListMain : TImageList;
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    TabSheet2: TTabSheet;
    GroupBox3: TGroupBox;
    Panel1: TPanel;
    GroupBox1: TGroupBox;
    Panel2: TPanel;
    txtSearch: TEdit;
    btnRefresh: TButton;
    btnSave: TButton;
    btnClear: TButton;
    Button3: TButton;
    TabSheet3: TTabSheet;
    Panel3: TPanel;
    Image1: TImage;
```

```
50      Image2: TImage;
51      Image3: TImage;
52      Image4: TImage;
53      Image5: TImage;
54      TimerAdelante: TTimer;
55      TimerActualizaID: TTimer;
56      Panel4: TPanel;
57      HWNDTxtKinectar: TEdit;
58      HWNDTxtPd: TEdit;
59      Label2: TLabel;
60      Label1: TLabel;
61      Button1: TButton;
62      Button2: TButton;
63      Image6: TImage;
64      btn1: TImage;
65      btn2: TImage;
66      btn3: TImage;
67      btn4: TImage;
68      Image7: TImage;
69      TabSheet4: TTabSheet;
70      GroupBox2: TGroupBox;
71      Panel5: TPanel;
72      Label3: TLabel;
73      HWNDTxtOSC: TEdit;
74      TabSheet5: TTabSheet;
75      GroupBox5: TGroupBox;
76      Panel6: TPanel;
77      Image8: TImage;
78      HWNDTxtLoopMIDI: TEdit;
79      HWNDTxtNI: TEdit;
80      Panel7: TPanel;
81
82
83      Procedure CapturarEnPanelKinectar; // hWnd of Slave App
84      Procedure CapturarEnPanelPd;
85      Procedure CapturarEnPanelOSC;
86      Procedure CapturarEnPanelLoopMIDI;
87      procedure CapturarEnPanelNI;
88
89      function KillTask(ExeFileName: string): Integer;
90      Function  ExecCmd( Const cmdline:String):HWND;
91      procedure IniciarAPPs;
92      Procedure ActualizarProcesos;
93
94  {*=============================================================================*}
95      Procedure EnumerateWindows;
96  {*=============================================================================*}
97      procedure FormCreate(Sender: TObject);
98      procedure EventbtnClick(Sender: TObject);
99      procedure FormClose(Sender: TObject; var Action: TCloseAction);
100     procedure btxSearchChange(Sender: TObject);
101     procedure Button1Click(Sender: TObject);
102     procedure Button2Click(Sender: TObject);
103     procedure TimerAdelanteTimer(Sender: TObject);
104     procedure TimerActualizaIDTimer(Sender: TObject);
105     procedure Image2Click(Sender: TObject);
106     procedure Image3Click(Sender: TObject);
107     procedure Image4Click(Sender: TObject);
108     procedure Image6Click(Sender: TObject);
109     procedure btn1Click(Sender: TObject);
110     procedure btn2Click(Sender: TObject);
111     procedure btn3Click(Sender: TObject);
112     procedure btn4Click(Sender: TObject);
113     procedure Image7Click(Sender: TObject);
114     procedure Image8Click(Sender: TObject);
115
116 {*=============================================================================*}
117 private
118     Function GetListCount: Integer;
119     property ListCount   : Integer read GetListCount;
120 public
```

```
121         mHWnd :HWND; // hWnd of Slave App
122         APP1: string;
123         APP2: string;
124         APP3: string;
125         APP4: string;
126         APP5: string;
127 end;
128
129 var
130   FrmMain: TFrmMain;
131
132 implementation
133 uses
134   unWindowEnum, unMensaje;
135
136 var
137   g_WinEnum : TWinEnumeration;
138
139 const
140   CRLF = sLineBreak;
141
142 {$R *.dfm}
143
144 {*=============================================================================*}
145 Function EnumWindowsCallBack(AHandle: HWND; AWinInfo: TWinEnumeration): BOOL;
146 stdcall;
147 var     lpWinRec : PWinInfoRec;
148 begin
149
150         lpWinRec := New(PWinInfoRec);
151
152         Try
153            FillChar(lpWinRec^, SizeOf(lpWinRec) * 4, $0);
154
155            if(AHandle > 0) then
156               begin
157                  With lpWinRec^ do
158                        begin
159                           FWinHandle := AHandle;
160                           FWinHidden := IsWindowVisible(AHandle);
161
162                           if(GetWindowText(AHandle, FWinTitle, MAX_TITLE) = 0)
163 then
164                               begin
165                                  StrCopy(FWinTitle, PChar('Untitled Window'));
166                               end;
167
168                            if(GetClassName(AHandle, FWinClass, MAX_CLASS) = 0)
169 then
170                               begin
171                                  StrCopy(FWinClass, PChar('Untitled ClassName'));
172                               end;
173                        end;
174
175                  AWinInfo.Add(lpWinRec);
176               end;
177         finally
178             Dispose(lpWinRec);
179         end;
180
181         Result := True;
182
183 end;
184
185 {*=============================================================================*}
186 Procedure TFrmMain.EnumerateWindows;
187 var     hListItem : TListItem;
188         nCount    : Integer;
189 begin
190
191         if(Assigned(g_WinEnum)) then
```

```
192             begin
193                g_WinEnum.Clear;
194
195                EnumWindows(@EnumWindowsCallBack, LPARAM(g_WinEnum));
196
197                for nCount := 0 to Pred(g_WinEnum.Count) do
198                    begin
199
200
201                    if (
202                    //(g_WinEnum.items[nCount].WindowInformation^.FWinHidden     =
203 true) and
204                    (
205                        (g_WinEnum.items[nCount].WindowInformation^.FWinTitle    =
206 'KiNECTAR - untitled') or
207                        (g_WinEnum.items[nCount].WindowInformation^.FWinTitle    =
208 'C:\GUI\Ejecutables\OSCeleton\OSCeleton.exe') or
209                        (g_WinEnum.items[nCount].WindowInformation^.FWinTitle    =
210 'Pd') or
211                        (g_WinEnum.items[nCount].WindowInformation^.FWinTitle    =
212 'loopMIDI') or
213                        (g_WinEnum.items[nCount].WindowInformation^.FWinTitle    =
214 'OpenNI User Tracker.net')
215                    )
216                        )
217                     then
218                    begin
219
220                       hListItem := LvwWindows.Items.Add;
221
222                       With g_WinEnum.items[nCount].WindowInformation^ do
223                            begin
224                                if (FWinHidden = true) then
225                                begin
226                                   hListItem.Caption := FWinTitle;
227                                   hListItem.SubItems.Add(FWinClass);
228                                   hListItem.SubItems.Add(IntToStr(FWinHandle));
229                                   hListItem.SubItems.Add(BoolToStr(FWinHidden,
230 True));
231
232                                   if (FWinTitle = 'KiNECTAR - untitled') then
233                                   begin
234                                          HWNDTxtKinectar.Text                  :=
235 IntToStr(FWinHandle);
236                                   end;
237
238                                   if (FWinTitle = 'Pd') then
239                                   begin
240                                          HWNDTxtPd.Text := IntToStr(FWinHandle);
241                                   end;
242
243                                   if                 (FWinTitle                 =
244 'C:\GUI\Ejecutables\OSCeleton\OSCeleton.exe') then
245                                   begin
246                                          HWNDTxtOSC.Text := IntToStr(FWinHandle);
247                                   end;
248
249                                   if (FWinTitle = 'loopMIDI') then
250                                   begin
251                                          HWNDTxtLoopMIDI.Text                  :=
252 IntToStr(FWinHandle);
253                                   end;
254
255                                   if (FWinTitle = 'OpenNI User Tracker.net') then
256                                   begin
257                                          HWNDTxtNI.Text := IntToStr(FWinHandle);
258                                   end;
259
260
261                               end;
262                           end;
```

```
263                     end;
264                 end;
265
266             gpxWindows.Caption    :=    Format('Enumerated    Windows:    %d',
267 [g_WinEnum.Count]);
268           end;
269
270 end;
271
272
273 procedure TFrmMain.IniciarAPPs;
274 begin
275       APP1 := 'C:\GUI\Ejecutables\Kinectar\Kinectar_v0.8.3.exe';
276       APP2 := 'C:\GUI\Ejecutables\Puredata\bin\pd.exe';
277       APP3 := 'C:\GUI\Ejecutables\OSCeleton\OSCeleton.exe';
278       APP4 := 'C:\GUI\Ejecutables\loopMIDI.exe';
279       APP5                              :=                          'C:\Program
280 Files\OpenNI\Samples\Bin\Release\UserTracker.net.exe';
281
282        KillTask('Kinectar_v0.8.3.exe');
283        KillTask('Pd.exe');
284        KillTask('OSCeleton.exe');
285        KillTask('loopMIDI.exe');
286        KillTask('UserTracker.net.exe');
287
288        ExecCmd(APP1);
289        ExecCmd(APP2);
290        ExecCmd(APP3);
291        ExecCmd(APP4);
292        ExecCmd(APP5);
293
294        TimerAdelante.Enabled := true;
295
296 end;
297
298
299 {*=============================================================================*}
300 procedure TFrmMain.FormCreate(Sender: TObject);
301 begin
302          IniciarAPPs;
303
304 //        SetWindowText(Handle, Application.Title);
305
306        g_WinEnum := TWinEnumeration.Create(TWinEnumerationItem);
307
308        Try
309            EnumerateWindows;
310        except
311            On E:Exception do
312               begin
313                  if(Not(ExceptObject is EAbort)) then
314                     begin
315                        raise E.CreateFmt('An Exception Has Occured: %s [%s]',
316 [E.Message, e.ClassName]);
317                     end;
318               end;
319        end;
320
321 end;
322
323 {*=============================================================================*}
324 procedure TFrmMain.FormClose(Sender: TObject; var Action: TCloseAction);
325 var     nCount : Integer;
326 begin
327
328        if(Assigned(g_WinEnum)) then
329           begin
330              if(g_WinEnum.Count >= 0) then
331                 begin
332                    for nCount := 0 to Pred(g_WinEnum.Count) do
333                        begin
```

```
334                             Dispose(g_WinEnum.items[nCount].WindowInformation);
335                           end;
336                      end;
337
338                     KillTask('Kinectar_v0.8.3.exe');
339                     KillTask('Pd.exe');
340                     KillTask('OSCeleton.exe');
341                     KillTask('loopMIDI.exe');
342                     KillTask('UserTracker.net.exe');
343                   g_WinEnum.Free;
344              end;
345
346            Action := caFree;
347
348  end;
349
350
351  {*=============================================================================*}
352  procedure TFrmMain.EventbtnClick(Sender: TObject);
353  begin
354        ActualizarProcesos;
355  end;
356
357
358  Procedure TFrmMain.ActualizarProcesos;
359
360    Procedure ClearListView;
361      begin
362         With LvwWindows do
363              begin
364                 LockWindowUpdate(Handle);
365                 items.Clear;
366                 gpxWindows.Caption := Format('Enumerated Windows: %d', [0]);
367                 LockWindowUpdate(0);
368              end;
369      end;
370
371  var     hSaveDlg : TSaveDialog;
372          hFile    : TextFile;
373          nCount   : Integer;
374
375  const
376          FMT_FILE_LINE = 'Title: %s' + CRLF + 'ClassName: %s' + CRLF + 'Handle:
377  %d' + CRLF + 'Visible: %s' + CRLF + '%s' + CRLF;
378  begin
379
380         ClearListView;
381         EnumerateWindows;
382
383  end;
384
385
386  {*=============================================================================*}
387  procedure TFrmMain.btxSearchChange(Sender: TObject);
388  var     hListItem : TListItem;
389  begin
390
391          hListItem  :=  LvwWindows.FindCaption(0,  txtSearch.Text,  True,  False,
392  True);
393
394          if(hListItem <> nil) then
395             begin
396                LvwWindows.Selected := hListItem;
397                hListItem.MakeVisible(True);
398             end;
399
400  end;
401
402
403  {*=============================================================================*}
404  Function TFrmMain.GetListCount: Integer;
```

```
405 begin
406
407         Result := LvwWindows.Items.Count;
408
409 end;
410
411 Procedure TFrmMain.CapturarEnPanelKinectar;
412 begin
413         if (HWNDTxtKinectar.text <> '') then
414         begin
415
416                   mHWnd := StrToint(HWNDTxtKinectar.text);
417                   Windows.SetParent( mHWnd, Panel1.Handle );
418                   Windows.MoveWindow(mHWnd,     0,     0,     Panel1.ClientWidth,
419 Panel1.ClientHeight, True);
420
421                   // Put the focus on notepad
422                  // Windows.SetFocus( mHWnd );
423                  // ShowWindow(mHWnd,3);
424         end;
425
426 end;
427
428 Procedure TFrmMain.CapturarEnPanelPd;
429 begin
430         if (HWNDTxtPd.text <> '') then
431         begin
432
433                   mHWnd := StrToint(HWNDTxtPd.text);
434                   Windows.SetParent( mHWnd, Panel2.Handle );
435                   Windows.MoveWindow(mHWnd,     0,     0,     Panel2.ClientWidth,
436 Panel2.ClientHeight, True);
437
438                   // Put the focus on notepad
439                  // Windows.SetFocus( mHWnd );
440                  // ShowWindow(mHWnd,3);
441         end;
442
443 end;
444
445 Procedure TFrmMain.CapturarEnPanelOSC;
446 begin
447         if (HWNDTxtOSC.text <> '') then
448         begin
449
450                   mHWnd := StrToint(HWNDTxtOSC.text);
451                   Windows.SetParent( mHWnd, Panel7.Handle );
452                   Windows.MoveWindow(mHWnd,     0,     0,     Panel7.ClientWidth,
453 Panel7.ClientHeight, True);
454
455                   // Put the focus on notepad
456                  // Windows.SetFocus( mHWnd );
457                  // ShowWindow(mHWnd,3);
458         end;
459
460 end;
461
462 Procedure TFrmMain.CapturarEnPanelLoopMIDI;
463 begin
464         if (HWNDTxtLoopMIDI.text <> '') then
465         begin
466
467                   mHWnd := StrToint(HWNDTxtLoopMIDI.text);
468                   Windows.SetParent( mHWnd, Panel6.Handle );
469                   Windows.MoveWindow(mHWnd,     0,     0,     Panel6.ClientWidth,
470 Panel6.ClientHeight, True);
471
472                   // Put the focus on notepad
473                  // Windows.SetFocus( mHWnd );
474                  // ShowWindow(mHWnd,3);
475         end;
```

```
476
477 end;
478
479 Procedure TFrmMain.CapturarEnPanelNI;
480 begin
481         if (HWNDTxtNI.text <> '') then
482         begin
483
484                   mHWnd := StrToint(HWNDTxtNI.text);
485                   Windows.SetParent( mHWnd, Panel5.Handle );
486                   Windows.MoveWindow(mHWnd,     0,     0,     Panel5.ClientWidth,
487 Panel5.ClientHeight, True);
488
489                   // Put the focus on notepad
490                  // Windows.SetFocus( mHWnd );
491                  // ShowWindow(mHWnd,3);
492         end;
493
494 end;
495
496
497 procedure TFrmMain.Button1Click(Sender: TObject);
498 begin
499 //        EventbtnClick(Sender);
500          CapturarEnPanelKinectar;
501          CapturarEnPanelPd;
502          CapturarEnPanelOSC;
503          CapturarEnPanelLoopMIDI;
504          CapturarEnPanelNI;
505 end;
506
507 function TFrmMain.KillTask(ExeFileName: string): Integer;
508 const
509   PROCESS_TERMINATE = $0001;
510 var
511   ContinueLoop: BOOL;
512   FSnapshotHandle: THandle;
513   FProcessEntry32: TProcessEntry32;
514 begin
515   Result := 0;
516   FSnapshotHandle := CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
517   FProcessEntry32.dwSize := SizeOf(FProcessEntry32);
518   ContinueLoop := Process32First(FSnapshotHandle, FProcessEntry32);
519   while Integer(ContinueLoop) <> 0 do
520   begin
521     if ((UpperCase(ExtractFileName(FProcessEntry32.szExeFile)) =
522       UpperCase(ExeFileName)) or (UpperCase(FProcessEntry32.szExeFile) =
523       UpperCase(ExeFileName))) then
524       Result := Integer(TerminateProcess(
525                         OpenProcess(PROCESS_TERMINATE,
526                                     BOOL(0),
527                                     FProcessEntry32.th32ProcessID),
528                                     0));
529      ContinueLoop := Process32Next(FSnapshotHandle, FProcessEntry32);
530   end;
531   CloseHandle(FSnapshotHandle);
532 end;
533
534
535 Function  TFrmMain.ExecCmd( Const cmdline:String):HWND;
536 var       PI :PROCESS_INFORMATION;
537       SI :STARTUPINFO;
538       Ret :LONGBOOL;
539       lsRuta: string;
540 Begin
541                Result := 0;
542
543
544     lsRuta := ExtractFilePath(cmdline);
545
546
```

```
547                 ZeroMemory( Addr(PI), SizeOf(PI) );
548                 ZeroMemory( Addr(SI), SizeOf(SI) );
549
550                 // Initialize the STARTUPINFO structure
551                 SI.cb := SizeOf(SI);
552
553     Si.dwFlags := STARTF_USESHOWWINDOW;
554     Si.wShowWindow := SW_MINIMIZE; //SW_SHOW;    SW_HIDE;//
555 //    si.dwX := 2000;
556
557                 // Start the shelled application:
558                 Ret   :=   CreateProcessA(   Nil,   PChar(   cmdline   ),Nil,   Nil,
559 True,NORMAL_PRIORITY_CLASS,   Nil, PChar(lsRuta),    SI, PI);
560
561                 // --- let it start - this seems important
562                 WaitForSingleObject( PI.hProcess, 50 );
563
564 //    form1.memo1.lines.Add( InttoStr(PI.dwProcessID));
565
566
567                 If Ret Then
568                 Begin
569       //showmessage('pare');
570 //                    Result := InstanceToWnd( PI.dwProcessID );
571   //    form1.memo1.lines.Add('ExecCmd ->' + InttoStr(Result));
572
573                       CloseHandle( PI.hThread );
574                       CloseHandle( PI.hProcess );
575                 End;
576     {
577     hProcess: THandle;
578     hThread: THandle;
579     dwProcessId: DWORD;
580     dwThreadId: DWORD;
581     }
582
583 End;{ExecCmd}
584
585
586 procedure TFrmMain.Button2Click(Sender: TObject);
587 begin
588   IniciarAPPs;
589 end;
590
591 procedure TFrmMain.TimerAdelanteTimer(Sender: TObject);
592 begin
593         Application.BringToFront;
594         ActualizarProcesos;
595
596         if (HWNDTxtKinectar.Text <> '') then
597         begin
598               if (HWNDTxtPd.Text <> '') then
599               begin
600                     TimerAdelante.Enabled := false;
601                     ActualizarProcesos;
602                     TimerActualizaID.Enabled := true;
603               end;
604
605         end;
606 end;
607
608 procedure TFrmMain.TimerActualizaIDTimer(Sender: TObject);
609 begin
610       TimerActualizaID.Enabled := false;
611       ActualizarProcesos;
612       CapturarEnPanelPd;
613       CapturarEnPanelKinectar;
614       CapturarEnPanelOSC;
615       CapturarEnPanelLoopMIDI;
616       CapturarEnPanelNI;
617       PageControl1.ActivePage := TabSheet3;
```

```
618 end;
619
620 procedure TFrmMain.Image2Click(Sender: TObject);
621 begin
622       PageControl1.ActivePage := TabSheet3;
623 end;
624
625 procedure TFrmMain.Image3Click(Sender: TObject);
626 begin
627       PageControl1.ActivePage := TabSheet1;
628 end;
629
630 procedure TFrmMain.Image4Click(Sender: TObject);
631 begin
632       PageControl1.ActivePage := TabSheet2;
633 end;
634
635 procedure TFrmMain.Image6Click(Sender: TObject);
636 begin
637         KillTask('Kinectar_v0.8.3.exe');
638         KillTask('Pd.exe');
639         KillTask('OSCeleton.exe');
640         KillTask('loopMIDI.exe');
641         KillTask('UserTracker.net.exe');
642
643         Halt;
644 end;
645
646 procedure TFrmMain.btn1Click(Sender: TObject);
647 begin
648       frmMensaje.gMensaje := 1;
649       frmMensaje.ShowModal;
650 end;
651
652 procedure TFrmMain.btn2Click(Sender: TObject);
653 begin
654       frmMensaje.gMensaje := 2;
655       frmMensaje.ShowModal;
656
657 end;
658
659 procedure TFrmMain.btn3Click(Sender: TObject);
660 begin
661       frmMensaje.gMensaje := 3;
662       frmMensaje.ShowModal;
663 end;
664
665 procedure TFrmMain.btn4Click(Sender: TObject);
666 begin
667       frmMensaje.gMensaje := 4;
668       frmMensaje.ShowModal;
669 end;
670
671 procedure TFrmMain.Image7Click(Sender: TObject);
672 begin
673     PageControl1.ActivePage := TabSheet4;
670 end;
675
676 procedure TFrmMain.Image8Click(Sender: TObject);
677 begin
678      PageControl1.ActivePage := TabSheet5;
679 end;
680
681 end.
682
```

A.3.2 Pure Data

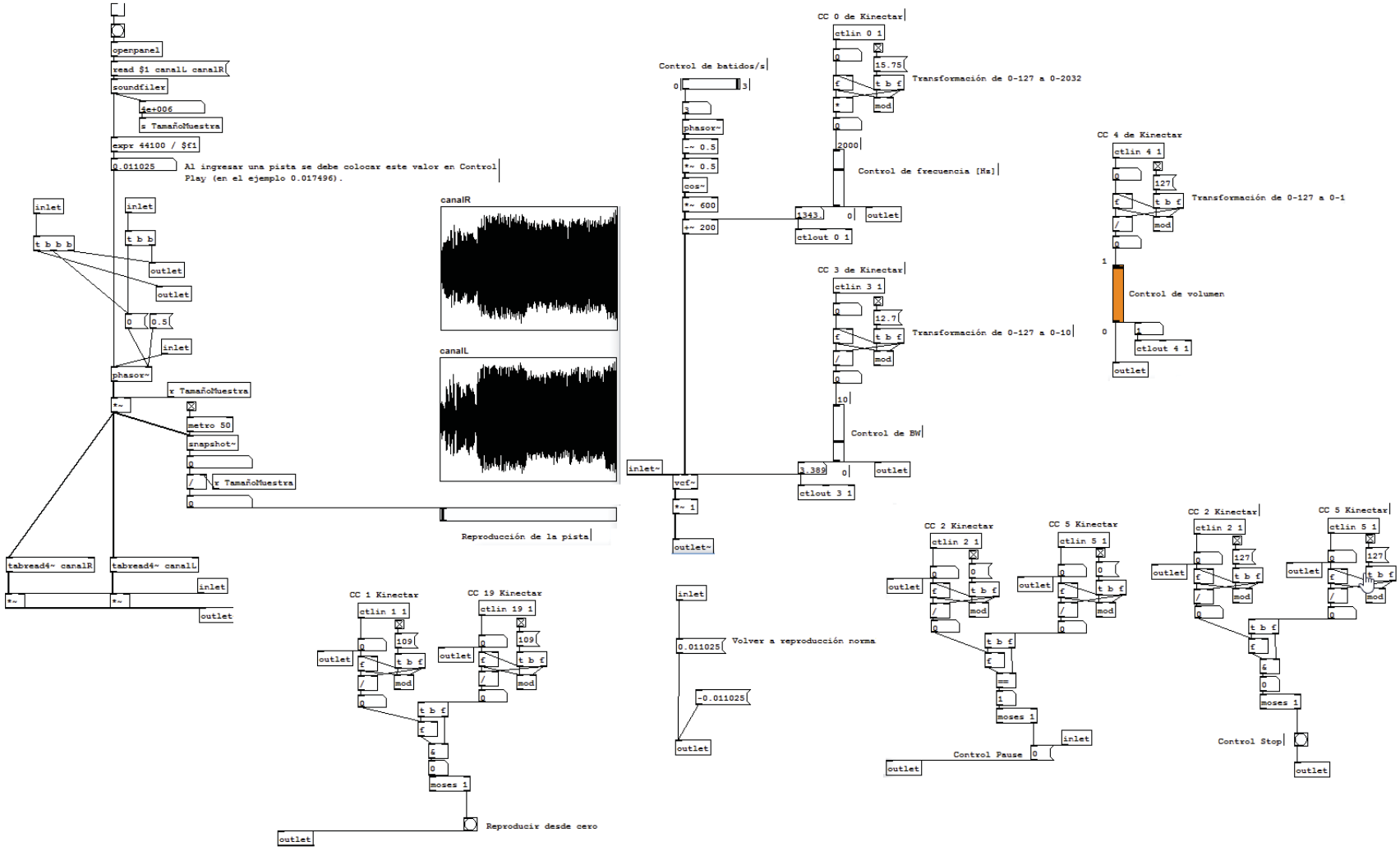

Figura A-1: Código de programación de Pure Data (fuente: elaboración propia)