SURI: CONTROLADOR MIDI POR MEDIO DE UN TECLADO DE COMPUTADOR[1]

Sebastián García Surianu[2]
surianu4@hotmail.com
DOI: 10.17230/ricercare.2016.5.1

1 Trabajo de grado presentado para optar al título de magister en Música de la Universidad EAFIT, que contó con la asesoría del maestro Gustavo Adolfo Yepes Londoño.

2 Egresado de pregrado y posgrado del Departamento de Música de la Universidad EAFIT y docente del CEC de la misma.

Resumen

El presente artículo narra las experiencias del desarrollo de Suri, un programa informático que permite enviar información MIDI a otros programas por medio del teclado del computador y combina las características lingüísticas de un editor de texto con elementos tecnológicos y musicales.

Del mismo modo, relata la forma en la que se conectó el desarrollo de este programa con la búsqueda de nuevas herramientas para la composición musical y describe sus diversas funciones y generalidades técnicas.

Palabras clave: Suri, nuevas tecnologías, Max/MSP, controladores MIDI, aplicaciones informáticas, composición musical.

Abstract

This article reports the experiences of the development of Suri, a computer program that allows sending MIDI information to other programs through the keyboard and combines the linguistic features of a text editor with musical and technological devices.

Besides, this article chronicles the way in which the development of this program was connected to the search of new tools for music composition and describes its various functions and its technical generalities as well.

Key words: *Suri, new technologies, Max/MSP, MIDI controllers, apps, music composition.*

Introducción

El programa *Suri* surgió a partir de una convocatoria promovida por el Departamento de Música de la Universidad EAFIT que invitaba a los alumnos de la misma a participar en los talleres impartidos por diferentes invitados durante el "X Encuentro de Música EAFIT, Música y nuevas tecnologías", realizado en Medellín, Colombia, del 30 de septiembre al 3 de octubre de 2014.

Entre los compositores e intérpretes más destacados estuvieron Adina Izarra y Rubén Riera de Venezuela, el grupo *Lumínico* de México y *Meridian Brothers* de Colombia. Los talleres fueron dictados por algunos de ellos y trataron áreas relativas a las nuevas tecnologías, tales como principios de Max/MSP, que es un entorno de desarrollo gráfico para música y multimedia creado por Miller Puckette a comienzos de 1986 (Colasanto, 2010, p. 20), introducción a *SuperCollider*, que es un lenguaje de programación para la síntesis de audio en tiempo real y la composición algorítmica (SuperCollider, 2016), instrumentación contemporánea, música multimedia y música interdisciplinaria.

La convocatoria consistió en desarrollar un pequeño proyecto en Max/MSP para compartirlo y perfeccionarlo en los talleres, con la asesoría de los invitados al encuentro. Para tal labor, se desarrolló la primera versión de *Suri,* que utilizaba las teclas del computador para generar sonidos del *General* MIDI sin una estructura lingüística aún definida.

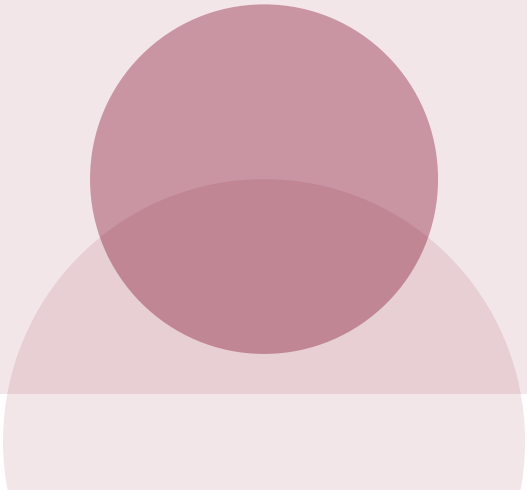
Con posterioridad, para el desarrollo del programa *Suri*, fue necesario emprender un recorrido por diferentes etapas de aprendizaje, desarrollo y ejecución, a partir de las experiencias obtenidas con músicos de México y Venezuela y los estudios realizados en el CMMAS (Centro Mexicano para la Música y las Artes Sonoras, ubicada en la ciudad de Morelia, en el Estado de Michoacán, en México) hasta su implementación práctica en proyectos artísticos de la Universidad EAFIT. Se divide en tres etapas: proyecto de investigación *Música narrativa*, que encabeza el maestro Andrés Posada; el concierto del primer semestre de 2016 del ensamble *Periscopio*, dirigido por el doctor Víctor Agudelo y el *Taller de Suri*, que dirige el autor de este artículo.

Durante el diplomado en el CMMAS, el proyecto evolucionó a gran escala gracias a las asesorías del compositor Francisco Colasanto, quien orientó el desarrollo de la aplicación durante la estadía en México. Por otra parte, el compositor Rodrigo Sigal, director del CMMAS y miembro del grupo *Lumínico,* en su estadía en Colombia, dio a conocer la labor que realiza dicha institución en el campo de las nuevas prácticas musicales y brindó siempre su apoyo para la realización y publicación del proyecto.

1. Acerca de *Suri*

Suri es una aplicación informática que permite enviar información MIDI a otros programas por medio del teclado del computador y combina las características de un editor de texto con funciones de un controlador MIDI. Permite visualizar la entrada de caracteres en un cuadro de diálogo y manipula los sonidos –nativos de otros programas– a través de los parámetros de su interfaz. Los sonidos mencionados se emiten en tiempo real al hacer contacto con las teclas y el resultado musical depende de la preparación del material sonoro y la destreza de cada ejecutante.

A pesar de que *Suri* no emite sonidos por medio de un sistema de resonancia propio y no posee un timbre particular generado por vibraciones acústicas, es verdad que logra ser una herramienta que cumple las características necesarias para producir música en forma similar a los instrumentos tradicionales.



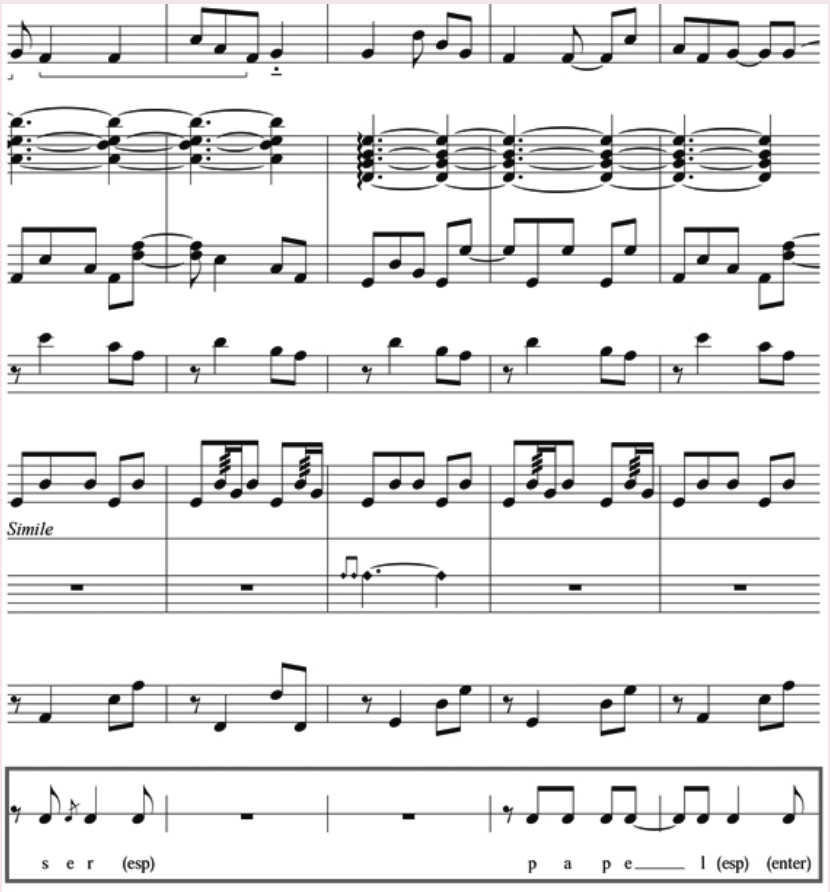
Von Hornbostel y Sachs (1914) publicaron una clasificación de los instrumentos musicales en la que incluyen los electrófonos como quinta categoría, al lado de los aerófonos, membranófonos, cordófonos e idiófonos. Aquella categoría hace referencia a todos los instrumentos análogos que generan sonidos a través de señales eléctricas y los instrumentos electrónicos que generan sonidos por medio de procesos digitales. En el último grupo encontramos los computadores, que pueden producir sonidos por síntesis digital o por muestras de audio controladas.

Suri, al estar integrado a un computador, se une a los electrófonos porque, además de producir sonidos a través de un proceso digital, contiene las características propias de un instrumento como, por ejemplo, la capacidad de producir un mismo timbre en alturas distintas, ejecutar una frase con diferentes intenciones, dinámicas y articulaciones, utilizar diferentes escalas y promover un progreso instrumental a partir del estudio de una técnica de ejecución.

La participación de *Suri* en un conjunto instrumental puede ser protagónica, o bien, puede hacer parte de un conjunto de instrumentos acompañantes. En el caso de la obra *Dichos y piropos*, *Suri* participa en un conjunto instrumental conformado por clarinete en Bb, violín I, violín II, viola, violonchelo, piano, dos percusiones y bajo eléctrico.

A continuación, en la figura 1, vemos un fragmento de la obra, en el que *Suri* dialoga rítmicamente con otros instrumentos utilizando partes de la frase: "Me gustaría ser papel para poder envolver ese bombón". La palabra "ser", por ejemplo, se escribe utilizando tres valores métricos, entre ellos, una *acciaccatura* (letra e) y una corchea que marca la tecla espaciadora.

Figura 1. Ejemplo



Fuente: Dichos y piropos. Composición de Felipe Corredor, A. M. Franco, Laura Gutiérrez, Rafael Rivera, Sebastián García Surianu, compases 39-43.

2. Plataforma de desarrollo

Para crear los prototipos previos y la versión actual de *Suri* se implementó el programa Max/MSP, que permitió varias opciones de programación debido a la amplia cantidad de objetos (rutinas del programa que tienen tareas específicas; Colasanto, 2010, p. 29) que permite integrar.

El autor citado explica las relaciones de algunos objetos y sus posibles funciones al conectarlos unos con otros. Su forma de plasmar los métodos para llevar a cabo esas conexiones inspiró la interconectividad de los objetos de *Suri* y brindó una ayuda permanente como guía técnica para el desarrollo de la aplicación. Colasanto expresa en forma explícita dichas funciones y muestra las numerosas opciones que ofrece el programa para desarrollar una idea a través de diferentes rutas.

Para crear *Suri*, por ejemplo, se investigó la ruta más adecuada para desarrollar un prototipo de baja latencia que permitiera una comunicación fluida entre el teclado y el programa. Dicha tarea exigió la experimentación de diferentes modelos que contenían rutas de desarrollo distintas y una combinación de objetos con un flujo de información particular, que permitía encontrar las conexiones oportunas que necesitaba el sistema para crear soluciones efectivas de buen rendimiento.

Se utilizaron, entonces, dos clases generales de objetos: los que están relacionados con el sonido y los que lo están con el texto.

- Los primeros permiten modificar el mapeo MIDI, transportar las alturas, controlar duraciones y manipular cambios de control MIDI.
- Los segundos posibilitan cambiar el tipo de fuente y su contorno, los colores del texto, los colores del fondo y el tamaño de la letra.

Los objetos relacionados con el texto, aunque no conservan una conectividad musical con los que lo están con el sonido, están programados para que puedan manipular, en un futuro, aspectos como la modificación de altura por medio del color o la modificación del volumen mediante el tamaño del texto. Sin embargo, la única relación actual entre ambos grupos de objetos es la de los sonidos generados al escribir en el editor de texto.

3. Prototipos previos

Durante todo el proceso de desarrollo fue una constante utilizar las teclas del computador como medio físico para la producción de sonidos. Esta decisión surgió de la experimentación en los talleres previos a la semana de nuevas tecnologías, en los que se exploró la función del objeto *key* (cuando colocamos un objeto *key* dentro de un *patch* y presionamos cualquier tecla del teclado de la computadora, dicho objeto nos reporta el número de caracter ASCII (Colasanto, 2010, p. 52) para manipular los valores numéricos del código ASCII (*American Standard Code for Information Interchange*, que en español se suele traducir como código estadounidense estándar para el intercambio de información; Colasanto, 2010, p. 52).

Para los fines del artículo se describen a continuación las etapas más importantes del proyecto, divididas en prototipos de desarrollo:

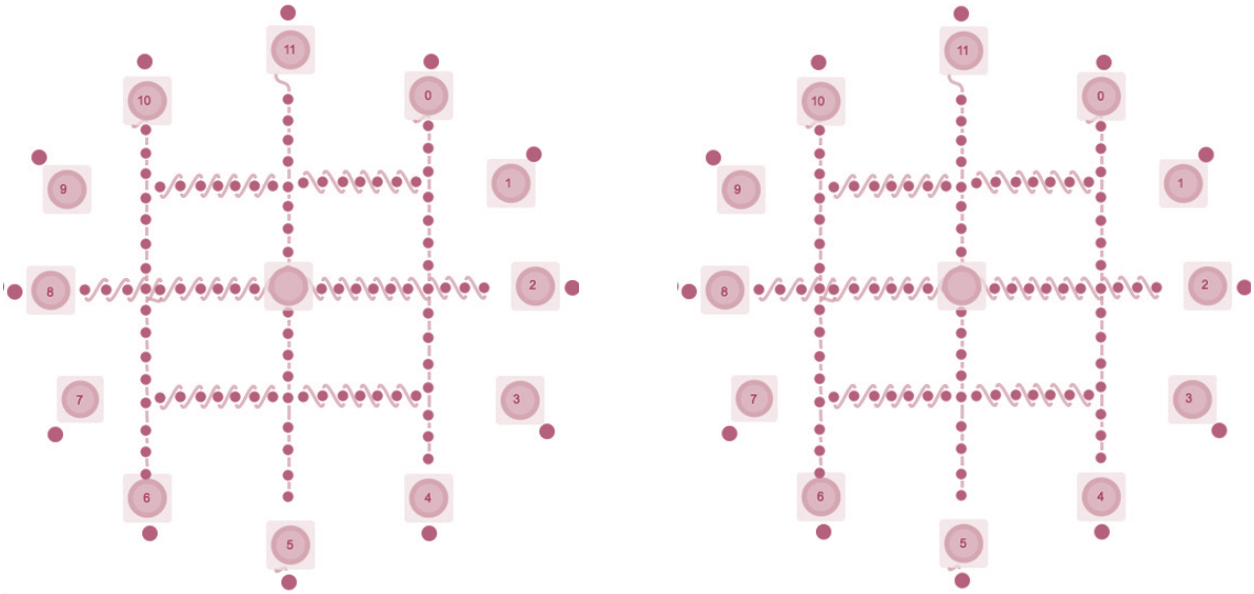
3.1 Primer prototipo: círculos de luces

El primer experimento consistió en un programa que permitía tocar la escala cromática en el teclado y representar los sonidos en forma gráfica en la pantalla. Dichos sonidos se visualizaban en números del 0 al 11 dentro de pequeñas circunferencias de color verde que formaban un círculo grande para cada mano (ver figura 2).

La altura progresiva de la escala ascendía en la misma dirección de las manecillas del reloj y su representación se distinguía por la luminosidad de cada pequeña circunferencia al recibir los impulsos de las teclas. Las luces producidas en cada circunferencia se conectaban en los sentidos horizontal y vertical por unas líneas compuestas de puntos pequeños de luces que atravesaban el círculo. Al tocar las teclas, las líneas producían una sensación de movimiento al dividir el círculo en fracciones distintas.

Figura 2. Círculos de luces (interfaz gráfica)

Mano izquierda **Mano derecha**



Fuente: archivo personal

A continuación, en la tabla 1, podemos ver las características generales del prototipo *Círculo de luces*:

Tabla 1. Características generales del prototipo *Círculo de luces*

Sonidos del *General* MIDI	Si
Selector de cambio de sonidos	No
Interacción gráfica	Si
Funciones de texto	No
Opciones de guardado	No
Transposición de escalas	No
Latencia	No
Conectividad con otros programas	No

Fuente: elaboración propia.

3.2 Segundo prototipo: piano-teclado

El segundo experimento consistió en un programa que permitía tocar la escala cromática en el teclado de una forma práctica por medio de un movimiento diagonal que la misma ubicación de las teclas sugiere. Según las pruebas realizadas, se notó que, al utilizar tres teclas por fila, el movimiento diagonal permitía más versatilidad y ergonomía.

Se asignó, entonces, una columna diagonal para cada mano con tres teclas por fila, lo que determinó cuatro filas para la mano izquierda y otras cuatro para la derecha. Cada fila constaba de una columna que se trasladaba de izquierda a derecha a medida que descendía la posición de las manos. En la figura 3 vemos la representación gráfica de la interfaz y los grupos de teclas por fila y columna:

Figura 3. Piano teclado (interfaz gráfica)

Fuente: Archivo personal.

La digitación recomendada para ambas manos era 2-3-4 (tomando como referencia la digitación para piano). De igual manera, la mano izquierda utilizaba el pulgar para la barra espaciadora (octava), mientras la mano derecha utilizaba el quinto dedo para la tecla "ç" (octava). Las tres teclas de cada grupo por fila sumaban un tono y medio y, al completar el recorrido total, se obtenían los seis tonos de una octava por cada mano. Además, al igual que en el piano tradicional, ambas manos tenían una tesitura contrastante; la mano izquierda ejecutaba los graves y la derecha, los sonidos más agudos.

A continuación, en la tabla 2 podemos ver las características generales del prototipo *Piano teclado*:

Tabla 2. Características generales del prototipo *Piano teclado*

Sonidos del *General* MIDI	Si
Selector de cambio de sonidos	No
Interacción gráfica	Si
Funciones de texto	No
Opciones de guardado	No
Transposición de escalas	No
Latencia	Si
Conectividad con otros programas	No

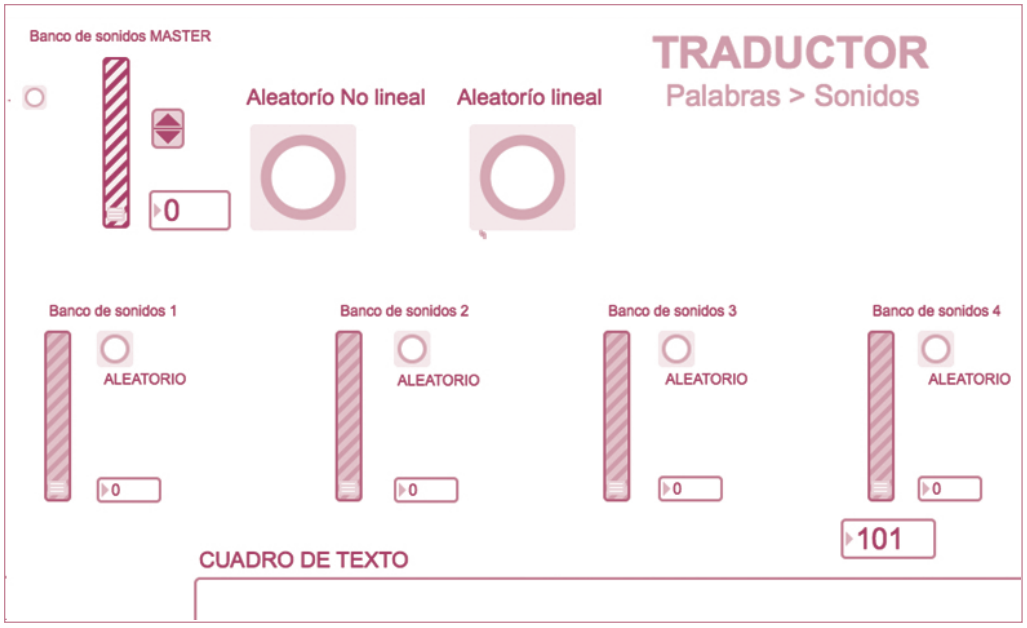
Fuente: elaboración propia.

3.3 Tercer prototipo: traductor de palabras a sonidos –timbres aleatorios–

El tercer experimento consistió en un programa que permitía producir sonidos al escribir en un cuadro de texto e implementaba un sistema que cambiaba, en forma aleatoria, los timbres de siete bancos de sonidos distintos, cada uno de los cuales estaba conectado a un grupo de caracteres (asignados por regiones del teclado) y, a partir de un comando, era posible producir la aleatoriedad. Los sonidos contenidos en cada banco iban de 0 hasta 127 y contenían los 128 sonidos del *General* MIDI que pueden ser manipulados con el objeto *pgmout* de Max/MSP. Este objeto permite enviar un tipo de mensaje MIDI, llamado "cambio de programa" (*program change*), que consiste en cambiar, dentro de algún dispositivo de hardware o software, el número de *preset*, program a, timbre, *patcher* o instrumento que está siendo utilizado (Colasanto, 2010, p. 91).

A continuación, en la figura 4 podemos ver un fragmento de la interfaz con sus nuevas implementaciones.

Figura 4. Traductor de palabras a sonidos –timbres aleatorios– (fragmento de la interfaz gráfica)



Fuente: archivo personal.

Se propusieron dos sistemas de aleatoriedad: aleatorio no lineal y aleatorio lineal. El primero planteaba cambiar, en forma irregular, los valores aleatorios de todos los bancos mediante un comando, lo que daba como resultado una mezcla de grupos de timbres en diferentes regiones del teclado. El segundo sugería cambiar, en forma regular, los valores aleatorios por medio de otro comando, lo que producía como resultado un solo timbre que cambiaba de modo aleatorio en todas las teclas. A continuación, en la tabla 3 podemos ver las características generales del prototipo *Traductor de palabras a sonidos (timbres aleatorios)*:

Tabla 3. Características generales del prototipo. Traductor de palabras a sonidos (timbres aleatorios)

Sonidos del *General* MIDI	Si
Selector de cambio de sonidos	Si
Interacción gráfica	Si
Funciones de texto	Si
Opciones de guardado	No
Transposición de escalas	No
Latencia	Si
Conectividad con otros programas	No

Fuente: elaboración propia.

3.4 Cuarto prototipo: traductor de palabras a sonidos –selector de timbre y escala–

Este prototipo, similar al anterior, pero con opciones más concretas, dio un paso adelante en la delimitación que fue estableciendo la investigación; por una parte, los recursos aleatorios disminuyeron de siete opciones a una (que podía ser activada desde el teclado) y, por otra, fue posible mapear el código ASCII en alturas específicas relativas a una escala diatónica.

Esta nueva implementación agudizó los problemas de latencia que se habían generado desde el primer prototipo, como por ejemplo, retardos continuos en la reproducción del sonido al presionar las teclas, retrasos en el monitoreo del código ASCII en el objeto *number* (una caja de números se utiliza para enviar y mostrar números enteros; Colasanto, 2010, p. 72), retardos en el movimiento de la imagen de la interfaz y problemas de reproducción en computadores con menos capacidad de procesamiento. En la figura 5 vemos el objeto *umenu*, que sirve para poder crear un menú de datos que podrán enviarse a cualquier tipo de objeto en forma de mensaje (Colasanto, 2010, p. 113) y que sirvió para habilitar los cambios de escala en la interfaz mediante un menú.

Figura 5. Traductor de palabras a sonidos –selector de timbre y escala– (fragmento de la interfaz gráfica)

Fuente: archivo personal.

A continuación, en la tabla 4 podemos ver las características generales del prototipo *traductor de palabras a sonidos (selector de timbre y escala)*:

Tabla 4. Características generales del prototipo traductor de palabras a sonidos (selector de timbre y escala)

Sonidos del *General* MIDI	Si
Selector de cambio de sonidos	Si
Interacción gráfica	Si
Funciones de texto	Si
Opciones de guardado	No
Transposición de escalas	No
Latencia	Si
Conectividad con otros programas	No

Fuente: elaboración propia.

3.5 Quinto prototipo: *word music*

Gracias a la colaboración del CMMAS y Colasanto se logra por último un prototipo más estable y que responde con efectividad ante la ejecución de las teclas. De esta forma comenzó un proceso de reorganización de los objetos y una selección más acertada de los selectores dispuestos para el intérprete en la interfaz.

Los primeros en acaparar la atención fueron los objetos que controlan funciones de texto que, con sus opciones de integración, favorecieron el concepto interdisciplinario entre la unión de música y lingüística. Se implementó, entonces, una programación para editar la clase, el tamaño, el contorno, la justificación, los colores de la fuente y los del fondo y el borde del cuadro de texto, mediante el objeto *attrui*, que inspecciona los valores de atributo del objeto que está conectado (Cycling'74/ IRCAM, 2014).

En la figura 6 vemos un fragmento de la interfaz que muestra los objetos de edición de texto y los selectores de eje tonal y escala que se explicarán en el siguiente capítulo.

Figura 6. *Word music* (fragmento de la interfaz gráfica)

Fuente: Archivo personal.

Las otras inserciones realizadas al prototipo fueron: diales para control de parámetros MIDI, diales para cambios de control MIDI, controles independientes para las teclas de *enter, space, backspace*, un menú para el control personalizado del tipo de salida y un banco de cuatro *presets*, a través del que es posible guardar diferentes "memorias" del estado en que se encuentran cualquiera de los objetos gráficos (Colasanto, 2010, p. 239), para guardar sesiones (antes solo funcionaba sin cerrar la aplicación).

A continuación, en la tabla 5, podemos ver las características generales del prototipo *word music:*

Tabla 5. Características generales del prototipo *word music*

Sonidos del *General* MIDI	Si
Selector de cambio de sonidos	Si
Interacción gráfica	Si
Funciones de texto	Si
Opciones de guardado	Si
Transposición de escalas	Si
Latencia	No
Conectividad con otros programas	Si

Fuente: elaboración propia.

4. Versión actual

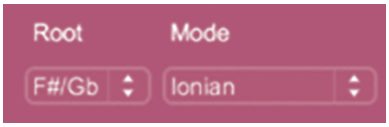
Luego de los prototipos anteriores se llegó por último a un modelo funcional práctico para el presente, con conexiones sólidas y una interfaz con más opciones para el usuario. Los objetos visibles están dispuestos en sentido estratégico para que el ejecutante los manipule en la escena.

A diferencia de las versiones anteriores, la mencionada incluye un sistema de guardado (para almacenar información una vez reiniciado el programa), la integración del *mouse*, casillas para nombrar efectos externos controlados desde *Suri*, un sistema de control para cambiar externamente los *presets*, el mapeo de los modos y otro que independiza todas las alturas MIDI del código ASCII, para una asignación personalizada de sonidos en cada tecla.

A continuación veremos la función de cada uno de los selectores del programa (que tienen alguna repercusión sonora), que muestra la forma en la que se implementaron desde el punto de vista artístico:

4.1 Selectores de escala y de eje tonal (*mode selector* y *root selector*)

Figura 7. Selectores de escala y de eje tonal



Fuente: Archivo personal (interfaz de Suri)

Estos selectores, descritos en la interfaz como *mode* y *root,* son los encargados de cambiar las alturas de los sonidos en el teclado. El primero, se conecta con un objeto llamado *coll*, que permite crear un *array* de valores de hasta 256 elementos por cada índice (Colasanto, 2010, p. 189) y cuya función es enlazar las alturas MIDI de una escala con los valores del código ASCII del teclado, mientras que el segundo permite hacer una transposición de las alturas MIDI de la escala seleccionada. Ambos selectores fueron desarrollados con el objeto *umenu,* cuya función es visualizar líneas de texto en un menú desplegable que envía datos numéricos según la opción elegida.

Si seleccionamos una escala o modo, el objeto se conecta con un *coll* que contiene una configuración MIDI alusiva a dicha escala y envía un mapa de notas específicas para que las teclas reproduzcan dicho modo. Y si cambiamos de nota en el selector de eje tonal, el programa realiza una operación matemática para transponer la misma escala a cualquier otro tono en un rango de F# hasta f, con cubrimiento de todos los semitonos intermedios.

Al escoger una nota en el selector de eje tonal, el programa suma el valor de transposición a cada uno de los valores que conforma la escala que está siendo transportada. El valor de transposición 0 es igual a F#, y así sucesivamente hasta llegar a f, con un valor de transposición 11. Cada valor de transposición se suma a los valores de las escalas de F#, en forma tal que, si se toma como ejemplo una transposición de A, serían todos los valores de F# sumados con el número de transposición 3.

Las carpetas que contienen los códigos numéricos que representan cada escala fueron programados a partir de F# por dos razones: la primera de ellas, debido a los resultados de un proceso de experimentación en el que se probaron diferentes instrumentos virtuales (fabricados por empresas reconocidas), que reaccionaron de manera positiva a las transposiciones propuestas por el programa; y la segunda, por causa del control de las alturas en una zona media, de suerte que no bajara o subiera mucho el registro al hacer las transposiciones. El rango va de un F_1 hasta un A_5, al tomar en cuenta la extensión del rango a través de las transposiciones de octava (que veremos más adelante) y el C central como C_3.

La compositora Ana María Franco aplicó lo antes mencionado en su obra *El país de las nubes* (ver figura 8), que desarrolló para el *Taller de Suri,* al utilizar tres instrumentos virtuales y variar sus propiedades sonoras mediante el cambio de los selectores y la inserción de efectos de audio.

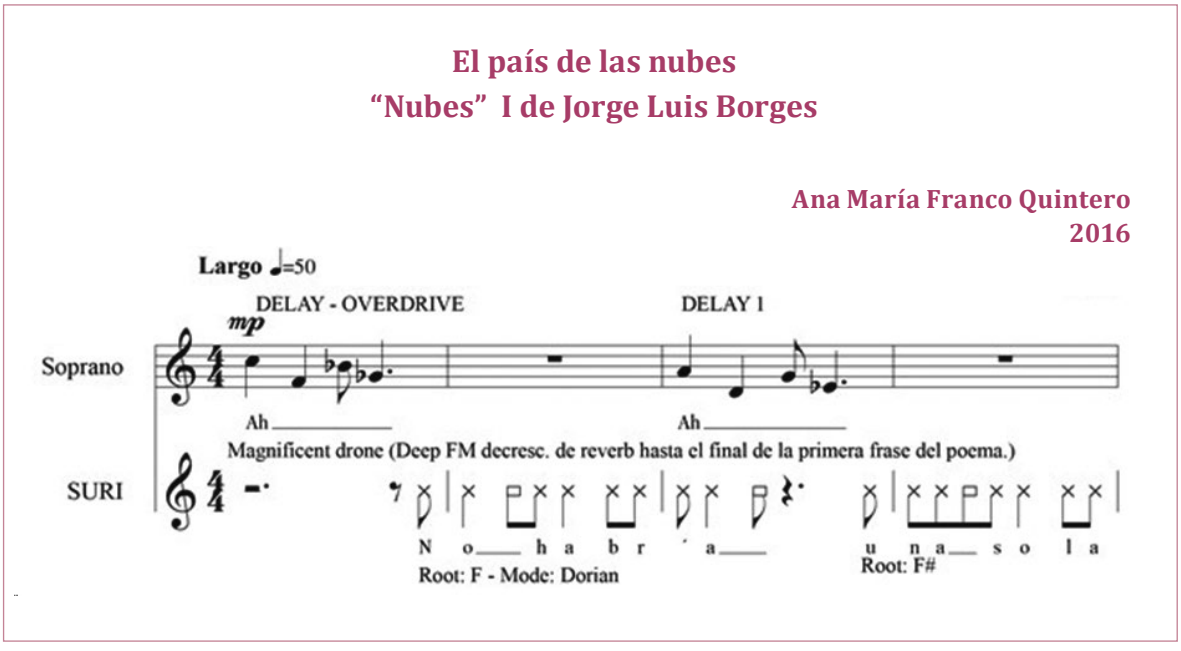
Ella opina que la implementación de dichos selectores brinda diversidad de sonoridades y, a pesar de la aparente tendencia al tonalismo y al modalismo del programa, la sonoridad puede verse deformada por numerosos efectos en el cambio de parámetros del sonido.

Lo anterior se evidenció primero en el uso del instrumento virtual *magnificent drone* (instrumento virtual de LOGICPRO 10), cuya característica primaria desintegra la afinación y desvanece la percepción de la escala implementada en *Suri.* Fuera de ello, el procesamiento de audio integrado al proceso varió las propiedades iniciales del sonido y permitió una identidad más personalizada del instrumento virtual.

Los otros dos instrumentos implementados (*Harp* y *Glass sky* de LOGICPRO 10) dieron en cambio como resultado una sonoridad más tonal, que permite identificar los sonidos producidos por el mapeo del selector de escala en el teclado del computador.

La compositora expresó, por último, que al llevar a cabo una preparación más exhaustiva del sonido e implementar nuevas estructuras tonales en el selector de escala sería posible producir una mayor diversidad sonora y una gama más amplia de posibilidades para la composición.

Figura 8. Ejemplo



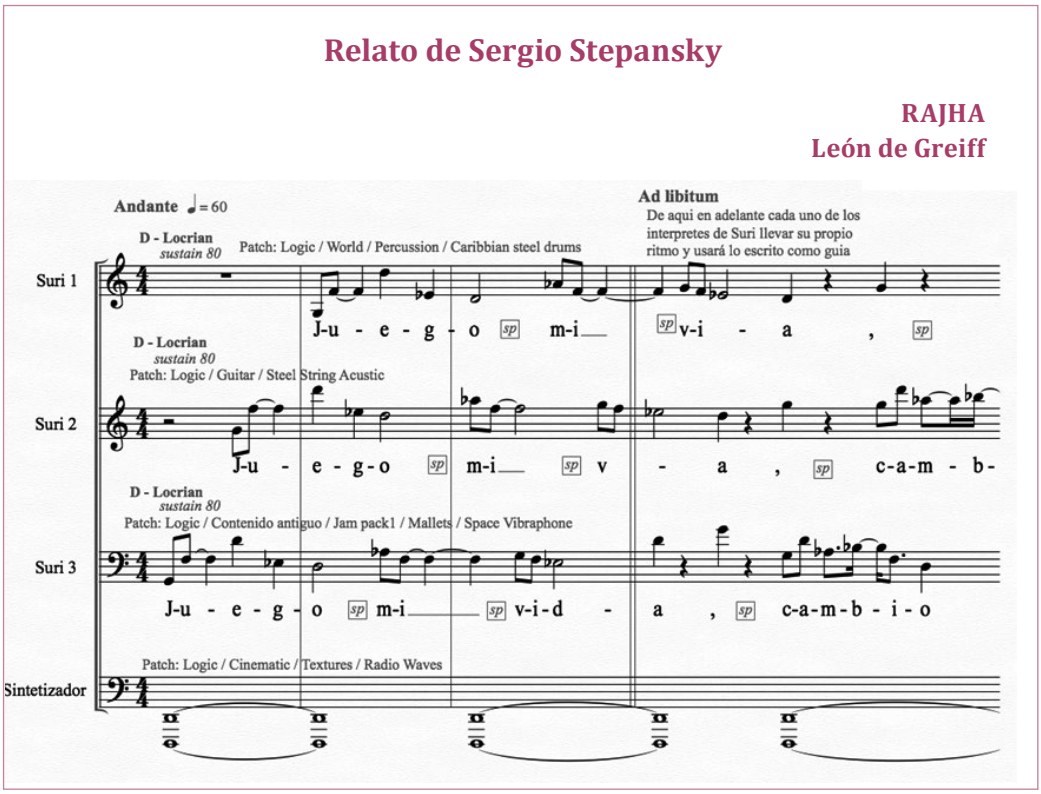
Fuente: *El país de las nubes.* Composición de A. M. Franco, compases 1-4.

En el caso de la obra *Lo confundieron con un psicópata* (compuesta por el autor de este artículo), las alturas resultantes se difuminaron debido a las propiedades de entonación indefinida del sonido utilizado que, al estar mapeado en una región extensa del *sampler*, produjo variaciones ininteligibles, lejanas del concepto tonal que de manera aparente sugiere el selector *mode.* Esto sucedió porque, por no utilizar alturas y timbres definidos, no siempre es fácil percibir la transposición de la muestra de audio mapeada, debido a su posible complejidad de frecuencias y simultaneidad de componentes sonoros.

Por otra parte, el uso del modo eólico y el tipo de sonidos de entonación indefinida usados en la obra produjo una inteligibilidad de las alturas y una organización particular de las muestras de audio que, aunque no reflejaban con fidelidad la sensación de una tonalidad menor, sí mostraban un grupo homogéneo de sonoridades. Además, aunque al mapear una muestra de audio en una región de varias notas los sonidos pueden deformarse (sea por su elongación al disminuir la frecuencia o por su reducción al aumentar la misma), se obtuvo la claridad del sonido, a pesar de las irregularidades en alturas y duraciones.

A diferencia de las percepciones sonoras antes descritas, durante el proceso de creación y experimentación en el *Taller de Suri*, Rodrigo Henao prefirió una tendencia modal en su pieza *Relato de Sergio Stepansky* (poesía de León de Greiff) al implementar sonoridades más definidas sobre un modo locrio. El selector *mode* y *root* cumplieron un papel fijo sin variaciones a lo largo de la pieza y proporcionaron un eje tonal continuo que permitió mantener una misma sonoridad al teclear todas las palabras. Dichas sonoridades, interpretadas por cuatro computadores, sostienen el flujo contrapuntístico que va generando armonía en el paso de letra contra letra. En la figura 9 podemos observar la relación melódica, armónica y lingüística de la pieza sobre la partitura.

Figura 9. Ejemplo



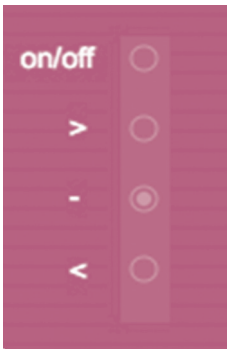
Fuente: Relato de Sergio Stepansky. Composición de Rodrigo Henao, compases 1-4.

Por una parte, se obtuvo un sentido lingüístico en la unión horizontal de cada uno de los caracteres; por otra, se consiguió un sentido armónico en la unión vertical de los mismos. Sin embargo, la conexión de tales elementos se produjo sin intención por la prelación del texto sobre la armonía y la conformación de acordes se dio en forma aleatoria al no pensar en la verticalidad.

El mapeo que hace *Suri* con todas las teclas a través del selector *mode,* asegura que dichos acordes, producidos de manera aleatoria, se agrupen dentro de las alteraciones de la armadura del modo locrio, lo que protege los límites de la tonalidad pero deja al azar la conformación de la armonía.

4.2 Selector de octava (*octave selector*)

Figura 10. Selector de octava



Fuente: Archivo personal (interfaz de Suri).

El selector de octava, visualizado en la interfaz como una hilera vertical de botones pequeños, se desarrolló a través de un objeto de Max/MSP, llamado *radiogroup*, cuya función es proveer una interfaz de usuario para seleccionar diferentes opciones. Con dicha herramienta podemos crear dos tipos de interfaces: una que permite seleccionar un valor de salida determinado y otra que funciona como un *check box* (Colasanto, 2010, p. 261).

En *Suri*, dichas opciones se centran en los cambios de octava por medio de tres selectores: octava inferior, octava central y octava superior. Cada una de estas opciones es asignable mediante los botones que provee el objeto *radiogroup* y el proceso para llevar a cabo la transposición de octava radica en la suma de 12 semitonos (en forma ascendente o descendente) de todos los valores asignados inicialmente en los selectores *root* y *mode.*

Al iniciar el programa, la opción predeterminada del objeto *radiogroup* está asignada a la octava central, que no se modifica con operaciones numéricas, pero al ejecutar un cambio en los otros dos selectores, el objeto desarrolla la operación numérica basada en la adición de seis tonos por encima o por debajo, según el tipo de octavación deseada.

El selector de octava permite a *Suri* trabajar en un registro de tres octavas que pueden ser asignables con libertad por el ejecutante y controladas sobre todo por la interfaz gráfica. Sin embargo, en la programación se instalaron tres *shorcuts* (atajos) para un cambio más ágil dentro del mismo cuadro de diálogo mediante tres teclas. La primera es el símbolo > (mayor que), que hace alusión a una mayor altura y su función es subir todas las alturas una octava. La segunda es el símbolo < (menor que*)* que hace alusión a una menor altura y su función es bajar todas las alturas una octava. Por último, la tercera es el símbolo – (guion) que hace alusión a la línea central y su función es transportar todas las alturas a la región central que trae en forma predeterminada el programa.

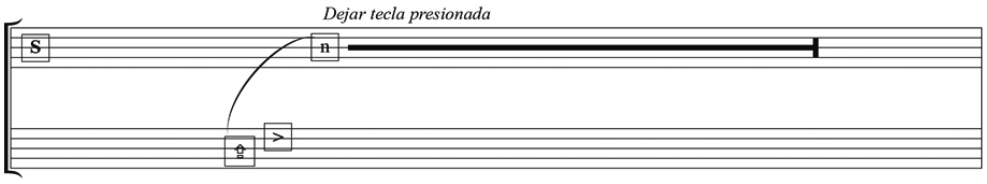
En el proceso de experimentación del *Taller de Suri,* se notaron problemas al usar el selector de octava con los bancos de sonidos del secuenciador si se utilizaba el selector de octava antes de que finalizara completamente el *release* del sonido. El problema resultante fue la prolongación de un sonido que no suspendía su funcionamiento a menos que se presionara la barra espaciadora (que activa y desactiva la línea de tiempo de reproducción del secuenciador). Este problema, que sólo se evidenció en la conexión con el secuenciador, cohibió a los compositores en el desarrollo libre de frases musicales que podían cambiar en medio de los sonidos y redujo las opciones a frases que debían contemplar el uso obligado de silencios en el cambio de octava, para impedir los problemas técnicos antes mencionados.

En la obra "*Sin sentido*", desarrollada por el compositor Felipe Corredor, integrante del *Taller de Suri*, se utiliza el selector de octava como un recurso principal para la variación de una misma frase y, debido a la falta de comprobación del funcionamiento de este recurso en la conexión de sonidos con otros programas, la efectividad del selector de octava requiere pruebas que confirmen su funcionamiento.

Para tal labor, primero será necesario revisar el comportamiento del selector de octava de *Suri* con los sonidos que el compositor seleccione y, si el sonido prolonga su funcionamiento fuera de control, la ejecución de la obra tendría que esperar la solución del problema en las versiones siguientes del programa.

En la figura 11 podemos ver la manera en la que el compositor expresa en la partitura los cambios que debe realizar el ejecutante:

Figura 11. Ejemplo



Fuente: Sin sentido. Composición de Felipe Corredor, fragmento.

De modo adicional al selector de octava es posible también hacer una transposición de octava mediante la asignación de las mayúsculas. Esta asignación funciona únicamente en las letras (no en números u otros caracteres) y su forma de operar es por medio de la asignación de alturas MIDI al código ASCII y no mediante una operación matemática (como es el caso del selector de octava).

Las mayúsculas, al tener una prioridad en el flujo de información que lleva la programación a través de los objetos, tienen el poder de mantener una octava abajo las alturas de las letras, independientemente de los cambios del selector de octava, lo que significa que si las mayúsculas están activas, cualquiera de las tres octavas, inferior, central y superior, sonará una octava por debajo de su valor predeterminado.

La ventaja de la inclusión de este elemento radica en la posibilidad de lograr una nueva octava baja por debajo de la antes mencionada octava inferior, al alcanzar dos octavas por debajo del registro predeterminado que trae el programa, es decir, la octava central.

Por ejemplo, si combinamos las mayúsculas con una octava inferior y una octava superior, el resultado será la octava central debido a que el selector de octava da prioridad siempre a la última selección; pero en este caso, la selección de la mayúscula permanece y modifica lo demás, lo que da como resultado la transposición de 12 semitonos hacia atrás de la octava superior.

Lo mismo sucedería si combinamos las mayúsculas con una octava superior y luego con la octava inferior. La penúltima se cancelaría y la última estaría modificada por las mayúsculas y haría una transposición de 12 semitonos hacia atrás de la octava inferior, con lo que lograría bajar dos octavas a partir de la octava central.

Otro uso importante de las mayúsculas es el resultado sonoro en las relaciones interválicas, que se generan al digitar palabras que empiecen por mayúscula o, simplemente, cuando se quiera alternar mayúsculas con minúsculas. Las distancias entre las notas generan movimientos de grado disjunto, por salto de octava, que pueden ser utilizados por el compositor como un nuevo recurso.

Es importante tener en cuenta que, aunque este sistema del selector de octava y sus posibles relaciones con las mayúsculas funcionan en forma determinada por ahora, es posible que cambie, dependiendo de los arreglos que necesite el programa para solucionar las fallas actuales.

4.3 Botones de guardado (*presets*)

Figura 12. Botones de guardado

Fuente: archivo personal (interfaz de *Suri*).

Botones de guardado es un grupo de botones situados en la parte inferior izquierda de *Suri,* que permiten guardar todos los datos de la interfaz en ocho sesiones diferentes. Para guardar es necesario ajustar toda la sesión con los valores deseados, presionar el botón pequeño *store1...8* (según el número de sesión) y presionar por último el botón de guardado para confirmar. Para borrar el contenido de cada sesión se presiona el botón pequeño *clear1...8* (según el número de sesión) o, para eliminar todas las sesiones guardadas, se presiona el botón *clearall*.

Los botones de guardado se conectan con otro par de objetos: *autopattr* (que expone varios objetos en un sistema de almacenamiento; Cycling'74/ IRCAM, 2014) y *pattrstorage* (que guarda y recupera ajustes prestablecidos; Cycling'74/IRCAM, 2014); ambos permiten guardar las sesiones almacenadas en *presets* para abrir la información una vez que se reinicie el programa. Es fundamental que *presets* y los dos objetos *autopattr* y *pattrstorage* no se entiendan como dos tipos de guardado diferente sino complementario, porque la única forma de conservar todos los datos guardados (en la programación de *Suri*) es por medio de su integración y funcionamiento mutuo. Y, aunque *autopattr* y *pattrstorage* no sean objetos visibles, están representados por medio de dos botones situados en la parte inferior derecha de la interfaz, con el nombre de *write* y *read*, cuya función está relacionada con las frases *Guardar como...* y *Abrir...* (que podemos encontrar en la barra de menús de muchos programas). Hasta ahora no ha sido una necesidad integrar estos dos botones a la barra de menús de *Suri,* aunque es cierto que puede ser una adición importante para versiones futuras.

Para guardar o abrir una sesión completa, el programa exige utilizar dos tipos de archivos; el primero tiene la extensión *.maxpresets* y el segundo, la extensión *.json*. Ambas extensiones hacen parte de la lista de archivos que reconoce Max/MSP y están catalogadas entre los archivos JSON como: *JSON - Preset file - .max presets* y *JSON JSON - .json.*

Para guardar cualquiera de las sesiones es indispensable almacenar la información en las dos extensiones que, en forma automática, el programa solicita y, de la misma manera, para abrir un documento es necesario primero abrir el archivo con extensión *.maxpresets* y luego el archivo con extensión *.json.*

El tipo de guardado almacena únicamente datos que, al abrirlos de nuevo, modifican el estado de aquellos objetos que controlan archivos de audio en otros programas, como ya se advirtió. Puesto que el control de archivos de audio es una tarea más pesada que controlar datos MIDI, el modificar, guardar y abrir los datos de todos los objetos de *Suri* produce menos latencia en la reproducción del sonido al contacto con las teclas.

Jordà Puig (1997) habla de las posibilidades del MIDI en el multimedia y explica cómo un sonido digitalizado de calidad ocupa alrededor de 10mb por minuto, mientras que un fichero MIDI de la misma duración puede ocupar tan solo 10 Kb^2 y explica que esto sucede debido a que, en lugar de contener un sonido digitalizado, incluye las instrucciones necesarias para que otro programa active los sonidos, como ha sido el caso de nuestros ejemplos, en los que hemos explicado la forma en la que *Suri* puede "controlar" los sonidos provenientes del secuenciador y del *sampler*.

En la obra "*Sopa de letras*", desarrollada a través de la tutoría de Andrés Posada, el uso de los botones de guardado es indispensable para almacenar los cambios de parámetros que tienen los instrumentos en distintos puntos de la partitura. Además, el uso de una última versión de *Suri,* que incluye objetos *number* asignados a cada uno de los botones de guardado permite manipular, con un controlador externo, el cambio de sesión a través de cambios de control MIDI (cc).

La nueva implementación favoreció dos puntos importantes para la obra. El primero fue la comodidad de manejo al manipular los efectos sin necesidad de salir del cuadro de texto y el segundo el cambio ágil de parámetros de sonido para una misma palabra o frase. Ambos aspectos constituyeron un recurso importante en la obra, puesto que el permanecer en el cuadro de texto le permitió al intérprete concentrarse más en la interpretación de la partitura y la posibilidad de manipular varias sesiones con rapidez posibilitó mayor libertad de expresión en la composición.

4.4 Controles externos (*external controls*)

Figura 13. Controles externos

Fuente: Archivo personal (interfaz de *Suri*)

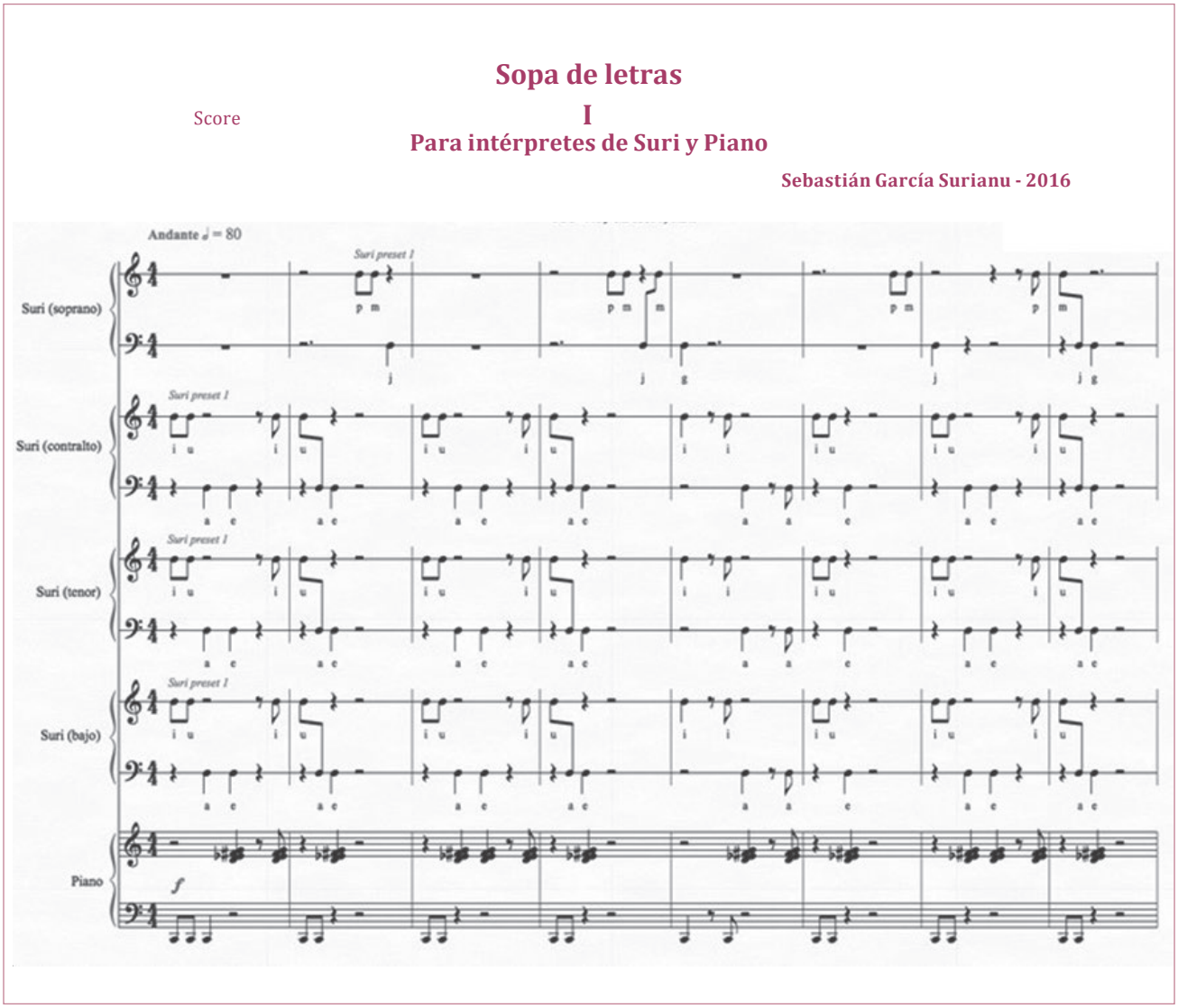
Los controles externos son seis diales que pueden tener una asignación de cambio de control MIDI (cc) para que el intérprete pueda manipular efectos en otros programas desde las opciones de manejo en *Suri*. Se programaron mediante la utilización del objeto *ctlout* de Max/MSP, que permite transmitir mensajes de cambio de control (*control change* o cc) MIDI (Colasanto, 2010, p. 88) y posibilita enviar el nivel de los diales de *Suri* a otros dispositivos para que le faciliten al intérprete asignar automatizaciones MIDI de los parámetros de sus efectos, con lo que se logra que la retroalimentación de un retraso (*delay*), la resonancia de un filtro, el tamaño de la reverberación o cualquier otro parámetro deseado por el ejecutante pueda ser manipulado desde *Suri.*

Huber (2007) hace una relación de este tipo de conectividades y dice que el MIDI OUT, en su condición de maestro, es el que habla y el MIDI IN, en su condición de esclavo, es el que escucha. La funcionalidad de *Suri* ha sido pensada para "hablar" y controlar los sonidos de los programas que "escuchan" y no precisamente para la situación inversa.

En *"Sopa de letras"* se exploran dichas conexiones y recursos en una cadena de flujo que funciona de la siguiente manera: primero, un controlador externo envía datos a *Suri* para asignar los cambios de sesión en los botones de guardado. Luego, dichos cambios incorporan variaciones en los seis diales de los controles externos para que, por último, el cambio de valores emitido desde cada sesión modifique los parámetros de efectos del programa esclavo.

En la figura 14 vemos en la partitura cómo se reflejan los cambios realizados en estas sesiones, con la asignación *suri preset 1*, *suri preset 2* o posteriores, según avanza la pieza.

Figura 14. Ejemplo



Fuente: Sopa de letras. Composición del autor, compases 1-8.

El propósito de la obra es representar los fonemas de cada letra del abecedario para un conjunto coral de cuatro voces por computador: *Suri-soprano, Suri-contralto, Suri-tenor y Suri-bajo,* más la inclusión de un piano como elemento acústico. Cada letra en el teclado emite el sonido característico del fonema y las combinaciones rítmicas entre ellas producen frases musicales variables. El resultado proyectado en el cuadro de texto es un conjunto de letras sin una coherencia semántica, que conectan de manera visual el sonido de cada fonema con su representación escrita.

Para lograr dicha representación fue necesario desarrollar un proceso que consistió en grabar a una mujer que pronunció todos los fonemas para la voz contralto y a un hombre que pronunció los fonemas para la voz tenor. Luego, por medio de una transposición digital, se obtuvieron los fonemas de la voz soprano y la voz baja.

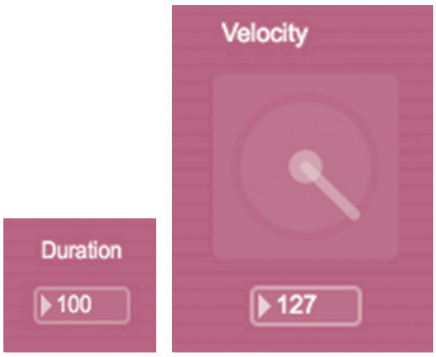
Con posterioridad las muestras de audio se mapearon en cuatro sesiones distintas (una por voz) mediante los códigos de relación MIDI – ASCII que establece el modo *custom*[3] del selector de escala. Este tipo de modo, al establecer cada código ASCII en alturas MIDI distintas, permitió la asignación independiente de cada sonido para cada letra.

En la actualidad es posible la interpretación de la obra por medio de una guía que explica cómo llevar a cabo el proceso de su preparación y montaje. Allí se expone la forma en la que los ejecutantes deben configurar el *sampler* de su computador con los sonidos referentes al tipo de voz que les corresponde y cómo deben configurar los botones de guardado para manipular los programas esclavos a través de los controles externos, que varían en cada una de las sesiones de *Suri*.

Los niveles en los cambios de control MIDI de nuestro programa, cuyos valores deben corresponder a los mismos que controlan el *time* y *feedback* del *delay* en el *sampler,* permiten la variación sonora de los fonemas por medio del cambio de sesión de los botones de guardado. Estas variaciones del sonido se aprovecharon en la obra como un recurso para la creación de texturas y mezclas de colores sonoros que los cuatro músicos producen al concertar las voces.

4.5 Velocidad y duración (*velocity and duration*)

Figura 15. Velocidad y duración



Fuente: Archivo personal (interfaz de *Suri*).

El indicador de duración es un objeto que permite ingresar números enteros entre 0 y 10.000 milisegundos para controlar el tiempo de reproducción que duran los sonidos ejecutados por las teclas y el dial de velocidad es un objeto que permite modificar valores entre 0 y 127 para controlar el volumen de ataque del sonido emitido por la digitación de las teclas. No obstante, la velocidad puede usarse para el cambio de timbre cuando se mapean muestras de audio en diferentes intervalos de velocidad.

Ambos objetos están conectados con otro llamado *makenote* (no visible para el ejecutante), cuya función es conducir la información MIDI a la salida de audio y con modificación previa de los valores de la velocidad y la duración del sonido. Dicho objeto envía, después de transcurrido un tiempo configurable por el usuario, un *note off* por cada *note on* recibido (Colasanto, 2010, p. 86).

Lo anterior se produce porque los objetos que se conectan con el *makenote* envían información que se conduce a través de entradas que permiten cambiar dichos valores en forma constante.

3 Opción del selector de escala que permite personalizar todos los sonidos de las teclas en forma individual porque no tiene asignaciones repetidas entre las alturas MIDI y los códigos ASCII.

Cuando los valores del indicador de duración son pequeños, las muestras utilizadas pueden contemplarse como impulsos de sonidos percutidos porque no revelan la duración completa de su fuente; no obstante, al tener valores más grandes, es posible escuchar hasta diez segundos la reproducción de la fuente. Los valores pequeños permiten que los sonidos ejecutados tengan una relación con la articulación más corta, similar al *stacatto*, y los sonidos más largos posibilitan conectar de modo permanente las notas para producir una articulación similar al *legato*.

Por otra parte, el dial de velocidad permite cambiar los timbres mapeados y genera una escala de volumen entre cada uno de los intervalos de la velocidad mapeada. Por ejemplo, si tenemos cinco timbres que queremos mapear sobre una misma nota, entonces será necesario que cada sonido esté contenido en uno de los cinco intervalos distintos de velocidad y asignados de modo correcto en el editor de mapeo del *sampler*. El primer sonido podría estar asignado, por ejemplo, entre 0 y 25, el segundo entre 26 y 40, el tercero entre 41y 60, el cuarto entre 61 y 90 y el último entre 91 y 127. El volumen actuará para cada timbre en forma independiente en cada uno de los intervalos de velocidad y la cantidad de velocidad no será enviada por la presión de la tecla, sino según el valor asignado al dial de velocidad. Se trata de una dinámica no dependiente del ataque de las teclas, como en la "dinámica por terrazas" de instrumentos como el clavecín o el órgano.

4.6 Selector de ratón (*mouse*)

Figura 16. Selector de ratón

Fuente: Archivo personal (interfaz de *Suri*).

La implementación del ratón (*mouse*) fue fundamental dentro del proceso de desarrollo porque, al igual que el teclado, éste posee la facultad de comunicarse físicamente con el computador y permite al ejecutante tener más recursos para enviar información al *software* por medio de la ejecución del *hardware.*

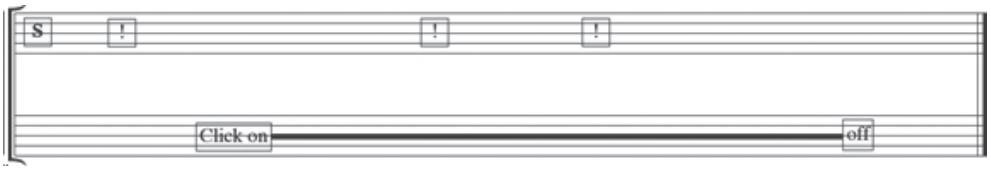
Esta adición al programa surgió del descubrimiento de un objeto, en Max/MSP, llamado *mousestate*, que le permite al programador tener las coordenadas del *mouse* y utilizar sus valores numéricos para cualquier propósito; los botones del *mouse* se muestrean cada 50 milisegundos, mientras que la posición del mouse en cada impulso de entrada (Cycling'74/IRCAM, 2014). Además, posibilita utilizar los botones del mismo *mouse*, los que envían impulsos que pueden interpretarse en forma libre por el sistema.

En *Suri*, la prolongación (*sustain*) del sonido por medio del *mouse* se puede lograr con uno cualquiera de dos tipos de acciones: presionar el botón izquierdo o mover el cursor sobre el eje *x* de la pantalla.

Para la primera acción del selector *mouse* se estableció que el impulso recibido al presionar el botón izquierdo enviaría el máximo valor MIDI (127) al *sustain* para prolongar el sonido y que el impulso recibido al soltar el mismo botón enviaría el mínimo valor MIDI (0) al *sustain* para detener el sonido. La designación de este recurso como control manual produjo opiniones divergentes; por una parte, unos consideraron que la obtención de la prolongación del sonido mediante las manos en el *mouse* podía obstaculizar la fluidez de la escritura mientras que para otros resultó atractivo el hecho de tener el mismo recurso en los límites espaciales y el hardware usual al alcance de las manos en el computador, sin necesidad de un pedal adicional.

Felipe Corredor, en su obra *Sin sentido*, utiliza este recurso del *mouse* en la partitura con la frase *click on* para representar la resonancia del pedal y la palabra *off* para la detención del sonido. En la figura 17 podemos ver un extracto de la partitura en el que se representa lo antes mencionado.

Figura 17. Ejemplo

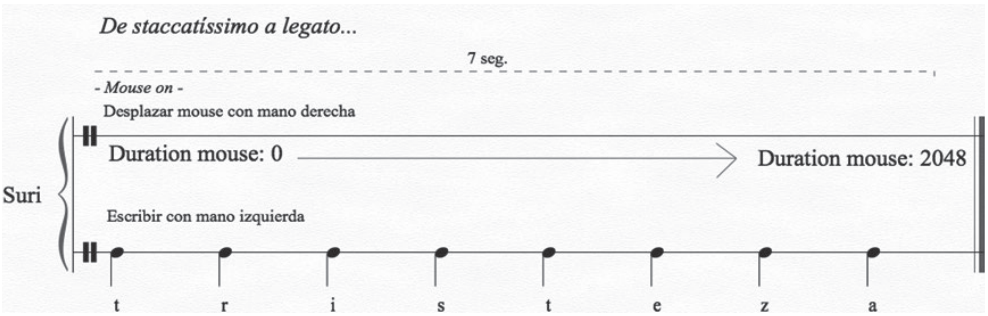


Fuente: *Sin sentido*. Composición de Felipe Corredor, fragmento.

Para la segunda acción del selector *mouse* se estableció que el movimiento del cursor sobre el eje *x* modificaría la duración del indicador de duración según el ancho de la pantalla mediante los datos suministrados por el objeto *mousestate*. La aceptación de este recurso tuvo menos divergencia que la anterior, debido a que es una herramienta que no partió de una referencia existente (a diferencia del pedal de prolongación) y que, a pesar de que inhibe el uso de una mano para la escritura, da más control de las articulaciones al ejecutante.

Como vimos en el capítulo anterior, el indicador de duración puede variar la articulación del sonido al cambiar sus valores y, como en este caso el selector *mouse* cambia de manera progresiva sus valores, podemos decir entonces que el mismo puede cambiar, también de modo gradual, las articulaciones de una frase musical. En la figura 18 vemos un ejemplo que muestra cuando una palabra es escrita con una sola mano y la otra cambia poco a poco su articulación de *staccatíssimo* a *legato.*

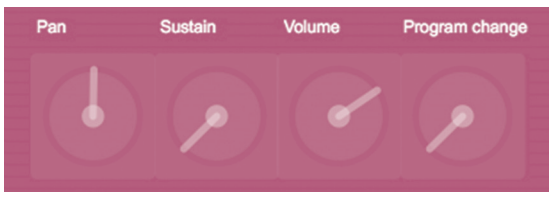
Figura 18. Ejemplo



Fuente: elaboración propia.

4.7 Panorama, pedal de sostenimiento, volumen y cambio de programa (*pan*, *sustain*, *volume* y *program change*)

Figura 19. Panel de cambios de control MIDI



Fuente: Archivo personal (interfaz de *Suri*).

Esta barra de diales controla el nivel de algunos mensajes de cambio de control MIDI que están expuestos en forma fija dentro del programa. Las casillas numéricas representan el valor de cada uno de los diales, que puede oscilar entre 0 y 127.

Es importante tener en cuenta que el volumen o cualquiera de los valores de cambio de control fijos enviarán los datos de su configuración cada vez que se comuniquen con otros programas, lo que puede hacer de su uso una ventaja o desventaja, según la información que se quiera filtrar en el envío.

A continuación se explicará cada uno de ellos:

4.7.1 Dial de panorama *(pan)*

Como dice Jordà Puig (1997), el panorama o *pan*, cuyo valor de cambio de programa es 10, permite definir la posición sonora de un canal en un ámbito de 180 grados, lo que quiere decir que si asignamos el valor cero a este dial en *Suri*, obtendremos la posición del sonido hacia la izquierda; si le asignamos 127, conseguiremos la posición del sonido hacia la derecha, y si le asignamos 64, alcanzaremos la posición central del sonido. En resumen, la dirección del indicador que se balancea dentro del dial está relacionada con la posición a la que está dirigido el sonido.

4.7.2 Dial de sostenimiento o prolongación del sonido *(sustain)*

El dial de sostenimiento, cuyo valor de cambio de programa es 64, produce la función de continuidad o prolongación del sonido, como vimos antes en el capítulo del selector *mouse*. La única diferencia es que, al modificar dicho dial, los movimientos del ratón no producen repercusión en el sonido.

Este control tiene dos posiciones: la primera oscila de 0 a 63 y su estado es apagado. La segunda oscila de 64 a 127 y su estado es encendido. Esto significa que, a pesar de que el dial tiene 128 valores disponibles, para el cambio de control 64 (*sustain*) solo son posibles dos resultados.

4.7.3 Dial de volumen (volume)

Jordà Puig también expresa en su libro que el volumen, cuyo valor de cambio de programa es 7, es uno de los controles más utilizados puesto que tiene la capacidad de controlar todos los volúmenes subalternos y, en el caso de *Suri,* puede controlar el volumen general del *sampler* o del secuenciador.

4.7.4 Dial de cambio de programa (program change)

El dial de cambio de programa se ve en la salida de audio como sintetizador AU DLS Synth, que traen, en forma predeterminada, los computadores APPLE. El dial contiene 128 niveles conectados a 128 instrumentos que el *General* MIDI ofrece, por defecto también, los cuales son posibles de ver en cualquier lista de instrumentos del protocolo MIDI.

4.8 Selector de entrada, espacio y retroceso *(enter selector, space selector y backspace selector)*

Figura 20. Selector de entrada

Fuente: Archivo personal (interfaz de *Suri*).

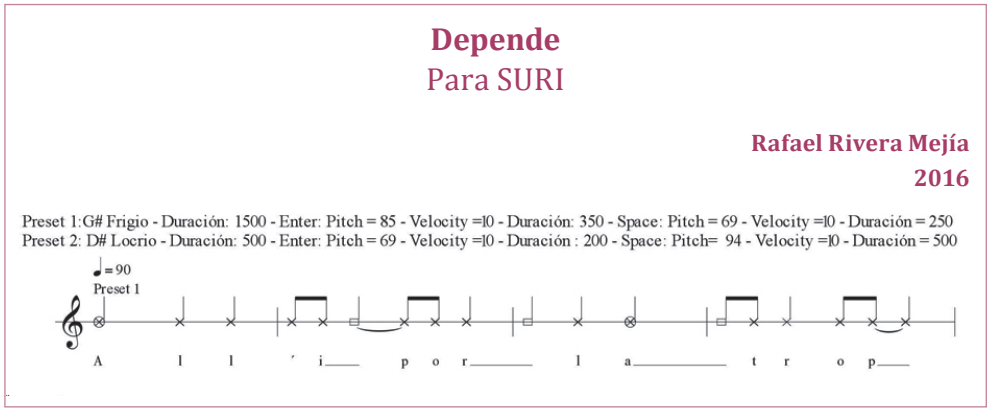
En *Suri* es posible independizar los sonidos de las teclas *enter*, *space* y *backspace* mediante los selectores de entrada, espacio y retroceso. La intención de dicha programación fue dar la posibilidad al ejecutante de personalizar el sonido de dichas tres teclas, o bien, cancelarlo, debido a que las mismas cumplen una función de manejo forzosamente frecuente, lo que podría entorpecer algunas intenciones musicales.

Los mencionados selectores, mediante tres objetos *number,* permiten ingresar datos de altura, velocidad y duración en forma independiente sin afectar las características sonoras de las demás teclas, lo que significa que los datos ASCII incluidos en el objeto *coll* excluyen los números 13 (*enter*), 32 (*space*) y 127 (*backspace*) y su

posible conexión con alguna altura MIDI, debido a que ellos son controlados por dichos selectores y no por las bases de datos que contienen los objetos *coll.*

A continuación, en la figura 21, vemos cómo el compositor Rafael Rivera, en su obra *Depende,* utiliza dos botones de guardado para cambiar la información en los selectores de entrada y espacio:

Figura 21. Ejemplo



Fuente: Depende. Composición de Rafael Rivera, compases 1-4.

Las indicaciones *preset 1 y preset 2* representan las dos sesiones que el ejecutante debe cambiar a lo largo de la obra. Cada una de ellas contiene los valores de altura, velocidad y duración de los selectores de entrada y espacio, con inclusión también de duración, escala y eje tonal de la interfaz general.

Se utilizaron los selectores de entrada y espacio en un rango de 69 a 94 para las alturas, 200 a 500 para la duración y 10 para todas las velocidades. Los valores de altura obedecen a las mismas notas del teclado porque el compositor implementa un instrumento virtual, que conecta de modo predeterminado las alturas MIDI con sus frecuencias relacionadas y el resto de valores para la caracterización del sonido.

Por ejemplo, en el caso del G# frigio, el número MIDI 85 corresponde al cuarto grado de la escala, mientras que el número MIDI 69 del espacio corresponde al segundo grado de la misma. Por otra parte, los momentos requeridos para la ejecución de ambas teclas están detallados en la partitura mediante símbolos específicos, como es el caso de la figura 21, en la que el espacio está representado por neumas o puntos cuadrados sin relleno, que alternan entre los espacios de cada palabra.

Podemos encontrar otro ejemplo de la implementación de estos selectores en la obra *Lo confundieron con un psicópata*, en la que se utiliza la tecla *enter* para emular el sonido que hacían antiguamente las máquinas de escribir al saltar el renglón. Para lograrlo, se mapeó en el *sampler* el sonido de la campana de la máquina de escribir y se sincronizó con el mismo valor MIDI en la casilla numérica *pitch* del selector de entrada (*enter*). De manera adicional a lo anterior, la casilla numérica *velocity* se mantuvo en un nivel inferior del dial de velocidad general para lograr un contraste del volumen

de ataque, mientras que la casilla numérica *duration,* se mantuvo en un nivel superior para reproducir el sonido completo de la campana.

Figura 22. Interfaz de *Suri*

Fuente: archivo personal.

5. Conclusiones

Debido al número de compositores que colaboraron en el proceso de investigación, se ha logrado reunir un material importante que ilustra las numerosas formas de implementar *Suri* en una obra musical. El análisis realizado en los procesos compositivos demuestra la dificultad que exige crear obras con coherencia musical y lingüística pero, a su vez, hace patente el potencial que posee el programa para integrar ambos campos en un trabajo interdisciplinario.

En el proyecto de investigación *Música narrativa* se explora la conexión interdisciplinaria entre la literatura, la música y la tecnología, por medio de la unión de participantes afines a dichos campos y la integración de *Suri* como una herramienta que promueve la comunicación entre ellos. Los resultados, que no pudieron ser descritos en este artículo, debido a que el proyecto no se ha desarrollado en su totalidad, prometen demostrar la efectividad del programa para unir dichas disciplinas en función de una obra artística.

No obstante, la integración del programa en el *Taller de Suri* y el ensamble *Periscopio* produjo resultados tangibles, resultado de una implementación práctica que permitió evaluar los alcances del proyecto. En el *Taller de Suri,* por ejemplo, algunos compositores aportaron ideas que se agregaron en forma oportuna, como el cambio de botones de guardado por medio de un controlador MIDI externo y etiquetas para los controles externos. Las demás ideas, que podrían implementarse en versiones

futuras del programa, fueron: más variedad de opciones para el selector de escala (*mode*), control progresivo de las dinámicas e inclusión de un dispositivo físico que permite más interacción interpretativa entre la máquina y el ejecutante.

Todos estos aportes y aspectos que van marcando una ruta futura de desarrollo se irán llevando a cabo a través del tiempo mediante la participación de *Suri* en proyectos artísticos, lo que permitirá perfeccionar su condición como instrumento musical interdisciplinario para lograr integrar, en forma más conexa, la comunicación entre las opciones de un editor de texto con aspectos concretamente musicales.

La anterior será una tarea que demandará la continuidad de la investigación y permitirá encontrar una mayor conexión entre la lingüística y la música mediante las posibilidades que brinda la tecnología.

Referencias

Colasanto, F. (2010). *Max/MSP guía de programación para artistas.* Morelia: Centro Mexicano para la Música y las Artes Sonoras.

Cycling'74/IRCAM (2014). *Max/MSP software - Max 6 online documentation*. Recuperado el 3 de mayo de 2016, de: https://docs.cycling74.com/max6/dynamic/c74_docs.html

Jordà Puig, S. (1997). *Audio digital y MIDI. Guías monográficas*. Madrid: Anaya Multimedia.

Huber, D. M. (2007). *The MIDI manual. A practical guide to MIDI in the project studio*. Oxford: Elsevier/Focal Press.

SuperCollider (2016). *SuperCollider*. Recuperado el 5 de mayo de 2016, de: http://supercollider.github.io

Von Hornbostel, E. M., & Sachs, C. (1914). *Zeitschrift für Ethnologie.* Recuperado el 5 de mayo de 2016, de: http://www.zeitschrift-fuer-ethnologie.de/