UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Programa de Maestría y Doctorado en Música

Escuela Nacional de Música

Centro de Ciencias Aplicadas y Desarrollo Tecnológico

Instituto de Investigaciones Antropológicas

El gesto en control del gesto

Diseño e implementación de un framework para la interacción gestual natural.

TESIS QUE PARA OPTAR POR EL GRADO DE **MAESTRÍA EN MÚSICA**

(TECNOLOGÍA MUSICAL)

Presenta:

Mauro Herrera Machuca

Tutor principal:

Roberto Morales Manzanares
(Universidad de Guanajuato)

MÉXICO, D.F. MAYO 2013

Índice general

Índice de figuras

Índice de cuadros

"El diseño de software no puede evitar afectar como suena la música por computadora, pero nosotros, programadores de software, debemos intentar no proyectar nuestras propias ideas a través del software "

Miller Puckette [1].

"SOCRATES: ¿Y quién utilizará la obra del fabricante de liras? ¿No será él quien mejor sepa dirigir al que la fabrica y que, una vez terminada, sepa si la lira está bien fabricada o no?"

Platón [2, (Cratilo 390b)].

1. Introducción

Se presenta en este trabajo un desarrollo tecnológico, producto de una inquietud personal que a su vez responde a una necesidad actual de explorar las posibilidades que la emergencia y acceso a nuevas tecnologías ofrecen para la expresión artística. Parte de esta inquietud se extiende brevemente hacia la teoría con el objetivo de plantear algunos conceptos que permitan articular formalmente una discusión sobre los procesos que toman lugar cuando la exploración se pone en marcha.

En específico, presento el desarrollo de un conjunto de aplicaciones que se dedican a capturar video con información 3D, y mediante el análisis de imagen conforman una representación del gesto a partir de lo que aquí llamo descriptores. Estos descriptores son utilizados como información de control para manipular procesos de composición, sonorización en tiemporreal e improvisación.

Con el propósito de acotar las posibilidades se redujo la extensión de la captura gestual al movimiento de la mano y la presencia y posición de los dedos. Con esto se logra una representación consistente de un gesto bien seccionado y altamente descrito. Esta simplificación permite que su implementación en la exploración sonora sea cognitivamente coherente, de manera que la relación entre el gesto sonoro y el de las manos sea evidente para el receptor.

La presentación de la aplicación desarrollada es precedida por la exposición de algunos conceptos básicos necesarios para entender el contexto dentro del cual se posiciona este trabajo, ademas de una revisión del gesto como concepto central y su relevancia en el campo de la música.

Finalmente, la exploración se pone en marcha con la implementación de una composición sonora que demuestra como ejemplo las posibilidades que se abren con este sistema.

Como producto de la exploración se obtiene a su vez una composición materializada en una aplicación que en su diseño y desarrollo ofrece un espacio multidimensional delimitado.

En este sentido la herramienta se presenta con una obra explorativa, sin embargo queda disponible para seguir el proceso de experimentación con compositores, artistas escénicos y artistas sonoros.

1.1. Justificación

El gesto es un principio básico de comunicación del cuerpo que escapa a las palabras, sirve para generar contextos emotivos y transmite intenciones. En su gran mayoría de expresiones encontramos una carga cultural que le da forma y sentido interpretativo dentro de un contexto social, es decir, los gestos son aprendidos a través de la historia de cada individuo y son específicos de su entorno. Sin embargo también existen gestos cuya carga de significado son inmanentes a la naturaleza del ser humano y se relacionan con la comunicación de los afectos primarios: felicidad, sorpresa, miedo, tristeza, enojo, asco ó repulsión e interés [3].

Así mismo la música se vale del cuerpo para actuar sobre un instrumento y producir

ondas sonoras que en su estructura tímbrica y temporal llevan codificado su mensaje. El gesto es movimiento y como parte operativa en la ejecución musical tiene una gran influencia en como se codifica el mensaje sonoro.

Desde el punto de vista de la interacción entre humanos, el gesto y la música comparten la característica de ser considerados sistemas semióticos capaces de producir significado, lenguajes no verbales que comunican más allá de la palabra y el pensamiento propositivo, como lo expone Jackendoff en [4].

En este sentido las ciencias de la cognición y la semiótica ofrecen campos de investigación que nos permiten indagar sobre la naturaleza de los lenguajes no verbales desde dos puntos de vista diferentes.

Por un lado la semiótica y las ciencias sociales nos ofrecen una visión desde las humanidades, considerando desde la cultura (característica que nos identifica como humanos) hasta el impacto que las interacciones humanas tienen en la conformación del tejido social, como plantean los interaccionistas simbólicos[5].

Por otro lado, el estudio de los procesos cognitivos y de percepción humana establecen una fundamentación biológica más propia de las ciencias naturales, constituyendo un modelo biológico del ser humano mediante el cual darle sentido y una explicación a los procesos de comunicación no verbal, como se puede ver en el trabajo de David Huron [6].

Para establecer un modelo de comunicación humana dentro del cual colocar a la música y el gesto como elementos de un lenguaje se presenta el trabajo de Sheizaf Rafaeli, ver sección 2.3, donde concibe a la interacción como un proceso complejo que solo puede tomar lugar plenamente entre humanos. Con el modelo de Rafaeli es posible identificar la importancia de los procesos de significación no verbal como elementos de transmisión de información y las limitaciones de la tecnología como medio de comunicación para transmitirlos. Otra ventaja de usar este modelo es que hace un cuestionamiento importante acerca del uso del concepto de interacción en los nuevos medios y permite observar los procesos que toman lugar en esta exploración desde una perspectiva más objetiva con respecto a la comunicación.

Las interfaces humano-máquina permiten alimentar a la computadora con coman-

dos emitidos por un usuario a través de un gesto, para controlar su funcionamiento, estableciendo una relación de amo y esclavo donde la máquina responde, con un muy limitado grado de interacción, de acuerdo a lo que está programada para hacer. Un ejemplo cotidiano de un dispositivo que lleva a cabo esta tarea es el ratón, que toma información de su desplazamiento físico para apuntar y navegar en un espacio digital representado a través de la visualización de la pantalla.

El impacto del desarrollo de interfaces en la historia de la computadora es de suma importancia, pues marca un cambio radical en el tipo de usuario, haciendo accesible la tecnología a no especialistas [7, Overture pg. xviii]. Como dicen Bettetini y Colombo: *"Los «universos» preparados e introducidos en el ordenador se abren al usuario, que entra simbólicamente en ellos, pero con un elevadísimo efecto de realidad, en virtud de una materialización (virtual) protésica de todo su cuerpo"*[8].

De la misma manera, el movimiento del cuerpo humano halla en los instrumentos musicales un objeto para materializar su expresión en sonido, sin embargo en ambos casos su idioma queda sujeto a las posibilidades del instrumento, posibilidades que inmediatamente delimitan la motricidad a la manera de accionar sobre él.

En su trabajo "Spectromorphology: explaining sound-shapes", Denis Smalley [9] plantea que la relación entre el sonido y el gesto se establece como una *subrogación gestual*, que a partir del aprendizaje de la experiencia de como suena algo, se puede relacionar un sonido nuevo con el movimiento que lo causó.

En la herramienta que aquí presento, esta cualidad protésica que mencionan Bettetini y Colombo y que es común para el instrumento musical y la interfaz humano máquina, se evade con la integración de tecnología de visión computarizada y obtención de mapas de profundidad, buscando añadir un mayor efecto de realidad. Es una cualidad que se busca en una nueva clase de interfaces llamadas NUIs (por sus siglas en ingles: Natural User Interface) que grupos como The Natural User Interface group[10], Responsive Environments Group del MIT Media Lab[11] y The OpenNI Organization[12] se han dado la tarea de estudiar y desarrollar.

Las NUIs son interfaces que por su diseño permiten emular una interacción con la computadora que se asemeja a la forma en que cotidianamente interactuamos con el

mundo. Por ejemplo, la Interfaz de Interacción Natural que aquí presento no delimita el movimiento del cuerpo para ser accionada y permite la creación de instrumentos que capturen el carácter del gesto para controlar objetos sonoros. Dos elementos de un instrumento que propone un idioma propio, delimitado por la conjunción de los sets de expresiones posibles con el gesto y la creación sonora, dos lenguajes con fuertes cargas estéticas cuya relación será explorada en una experimentación práctica.

Es en este punto que sostengo la relevancia y la contribución de este proyecto, ya que con el desarrollo de una herramienta con la cual se pueda experimentar con la relación gesto-música se puede proponer la experimentación sonora con una doble connotación: artística y metodológica. Propongo que la experimentación se puede desarrollar como una práctica artística que con expresiones estéticas comunica desde los gestos y la música simultáneamente. Vista desde la perspectiva metodológica, la herramienta permite un experimento donde se pone a prueba la posibilidad de vincularlos cognitivamente en un sistema en el cual, ambos idiomas se encuentran activos en el proceso de generación y transmisión de significado. Se podría considerar como una experimentación de exploración semiótica que se vale de los conceptos presentados en los capítulos 2 y 3 para darle sentido a los procesos que toman lugar. Con este instrumento la generación sonora mantiene en sus espectromorfologías coherencia con el gesto que las genera, que a pesar de ser sonidos sintetizados sin referencias a posibles fuentes conocidas, al verse generados por un sujeto que se mueve, se vinculan cognitivamente con la trayectoria, el esfuerzo y la forma del gesto, ofreciendo un enlace al movimiento como causante de nuevas experiencias sonoras.

1.2. Antecedentes

En 1965 se estrenó la obra *"Variations V"*, una colaboración entre John Cage y la compañía de danza Merce Cunnigham en la que se le entregaba a los bailarines control del sonido mediante una serie de foto-celdas apuntando hacia reflectores, que al ser intervenidos por la presencia de un bailarín activaban sonidos grabados en cintas, también se implementó una serie de antenas que al tener a un bailarín a menos de 4 pies de distancia disparaban sonido. Esta obra ademas de implementar control por

movimiento implementaba imágenes de video diseñadas por Nam June Paik [13].

En esta obra se establece un paradigma de la creación en el cual eventos de la danza son determinantes para el desarrollo sonoro de la pieza, en ella el tipo de control que ejercen los bailarines es de tipo botón, es decir accionan un sonido con un evento. Es destacable que siendo la primera obra de este tipo implementa una interacción que no requiere de un contacto físico con un dispositivo o sensor. Sin embargo la información obtenida del movimiento es mínima o nula, pues el activar foto-celdas o acercarse a una antena no dice nada sobre las cualidades del movimiento que provocaron el evento activador. Es en si una interacción que se asemeja a la percusión pero que carece de información sobre la dinámica musical. Sin embargo deposita en el bailarín la responsabilidad de la ejecución y participación en la creación sonora, le da un papel de intérprete.

Actualmente el paradigma de interacción estrenado en *"Variations V"* es recurrente y nuevas tecnologías se han explorado para integrar al movimiento como elemento de control para la síntesis de sonido y video. En el proyecto que aquí presento identifico cinco trabajos como los antecedentes que me motivaron a integrar mi propio sistema para producir obras interactivas. Considero importante mencionar que en todos los trabajos que se revisan en esta sección la composición de una obra interactiva implica la constitución de un sistema que permita capturar el movimiento para posteriormente ser utilizado como señal de control. Esto significa que parte de las posibilidades de desarrollo de una obra están determinadas por el sistema de captura y deben ser consideradas como elementos de la obra misma.

Para hablar de estos trabajos he decidido dividirlos en dos categorías, los que se valen de sensores montados sobre el cuerpo del bailarín y los que utilizan sensores externos al bailarín.

Dentro de los primeros destaca el Sistema Interactivo de Composición e Improvisación para Bailarines SICIB de Roberto Morales-Manzanares[14]. Este implementa un sistema de sensores "A Flock of Birds" de Ascension Tecnology Corporation que provee el seguimiento de objetos en el espacio con 6 grados de libertad: posición en 3D y orientación 3D. Es un sistema robusto que sin embargo tiene la inconveniencia de depender

de cables para transmitir la información del bailarín a la computadora, reduciendo la libertad de acción del usuario. Su aplicación principal es la captura de movimiento para la animación de personajes 3D, por lo cual no está optimizado para funcionar en tiempo real y rinde una latencia de hasta 0.5 segundos. La obra de Roberto utiliza la ventaja de conocer la posición 3D de las extremidades de los bailarines para desarrollar una composición de acuerdo a una arquitectura virtual que determina el espacio dentro del cual se mueve un bailarín. Más aún utiliza un algoritmo para describir y reconocer trayectorias de movimiento que le sirven para tomar decisiones.

Wayne Siegel en[15] implementa sensores de flexión montados en tobilleras, rodilleras y coderas que se comunican de manera inalámbrica desde un transmisor a un receptor que codifica la información en protocolo MIDI y la alimenta al ambiente de programación gráfica, MAX/MSP. En el trabajo de Wayne la información del movimiento es independiente de la posición del bailarín y en cambio ocupa la más detallada información del ángulo y la velocidad angular de las articulaciones del cuerpo para controlar sus sonidos. Este tipo de información hace que las obras de Siegel tengan un carácter particular y distintivo cuando se comparan con el resto, ver *"Sisters"* para dos bailarinas y sonido interactivo [16].

Dentro de la segunda categoría, la mayoría de trabajos implementan procesos de visión computarizada para extraer información del movimiento. Es decir, *"...la transformación de datos provenientes de imágenes fijas o una videocámara en una decisión o una nueva representación"*[17].

Un trabajo pionero en su aplicación es el de Camurri [18, 19] quien además de implementar un sistema de visión computarizada implementa un lenguaje de programación gráfica *"EyesWeb,"* (ver apéndice J) que permite aplicar procesos de análisis de movimiento de manera modular. La versatilidad de su lenguaje tiene como consecuencia que mucha gente a adoptado su programa para llevar a cabo implementaciones propias. Estrenada en el 2009, la obra *Mexicano* de Gamultimedia en la que participé con el diseño sonoro, utiliza EyesWeb en algunas escenas para extraer información del movimiento de los bailarines, utilizando elementos del movimiento como indice de contracción y cantidad de movimiento para controlar elementos de la síntesis de audio.

El objetivo del trabajo de Camurri y EyesWeb es la extracción de rasgos emotivos del movimiento, lo que implica una diferencia esencial en el acercamiento a su análisis. Camurri no solo obtiene información del movimiento sino que hace un ejercicio de interpretación, clasificando los resultados en estados emotivos y analizando la importancia de los gestos corporales en elementos de la música como el estilo.

En contraste con Camurri, Mark Coniglio desarrolla *"Isadora"* (ver apéndice G), un software para el rastreo de movimiento y lo echa a andar con el proyecto de Troika Ranch, una compañía de danza que incorpora ampliamente procesos multimedia y la colaboración de varios autores. Isadora en comparación con EyesWeb se enfoca más en las múltiples posibilidades expresivas del multimedia. Es un sistema que ademas del análisis implementa bibliotecas de control de video y sonido.

Finalmente el *"Very Nervous System"*[20] diseñado por el artista sonoro David Rokeby es un sistema que implementa dos cámaras y hace rastreo de movimiento, ha sido muy utilizado como interfaz para añadir interacción a instalaciones, para la creación de obras sonoras interactivas y obras para danza interactiva.

Con la disposición al público de programas de visión computarizada para el rastreo de movimiento, como EyesWeb, Isadora o bibliotecas de Jitter para Max/MSP, la danza interactiva y las instalaciones interactivas se han colocado como un área de exploración importante en la escena del arte actual, y la cantidad de trabajos en este sentido se ha incrementado y diversificado significativamente. Hoy en día podemos encontrar que la interactividad está involucrada desde compañías de danza (ie. Troika Ranch), campañas publicitarias y hasta programas de acercamiento a las prácticas musicales para personas con capacidades diferentes, como se ve en [21].

1.3. Objetivo

El objetivo general de este trabajo fue la creación de una herramienta que extrae información gestual del movimiento de las manos y la hace accesible para su interpretación como gesto sonoro sin invadir el cuerpo del ejecutante con sensores que disminuyan su movilidad. Esta herramienta fue implementada como una interfaz de interacción natural que permite una cartografía del gesto motriz al gesto sonoro.

Metas

- Selección de parámetros a medir: dado que se midieron parámetros del movimiento con procesos de visión computarizada, fue importante evaluar que parámetros del movimiento se pueden obtener y de éstos cuales contienen la información más relevante para la descripción del gesto. A estos parámetros les llamé descriptores del movimiento.

- La extracción de parámetros del movimiento debe de ocurrir en tiempo real, entendiendo *tiemporreal* como el desarrollo de procesos que se alimentan con información no predeterminada que se produce al momento, y que conforme aparece es considerada para la toma de decisiones, el control y la generación de información nueva. El tiempo que pase entre la captura de movimiento y la reproducción sonora debe de ser preferiblemente menor a 0.2s para evitar que el retraso afecte la percepción de que la respuesta es instantánea [8, pg. 17].

- Implementación de una interfaz que hace que los datos obtenidos del análisis del movimiento estén disponibles para que otros programas los utilicen.

- Implementación de una interfaz específica de SuperCollider que recibe los datos obtenidos del gesto y mantentiene la coherencia de estos.

- Implementación de sintetizadores que sirven de ejemplos de implementación.

2. Conceptos básicos

En esta sección se revisan los conceptos básicos que constituyen la taxonomía categórica dentro de la cual se puede contextualizar este trabajo con respecto a otro tipo de obras. En un principio se introduce la categoría de *"nuevos medios,"* que sirve para colocar el primer set de disciplinas del arte dentro de la cual se puede colocar el uso de la herramienta que presento.

A continuación se presenta la categoría de *"multimedia,"* que al ser considerada tiene un área de intersección con *"el arte con nuevos medios"*. Dentro de ésta intersección se

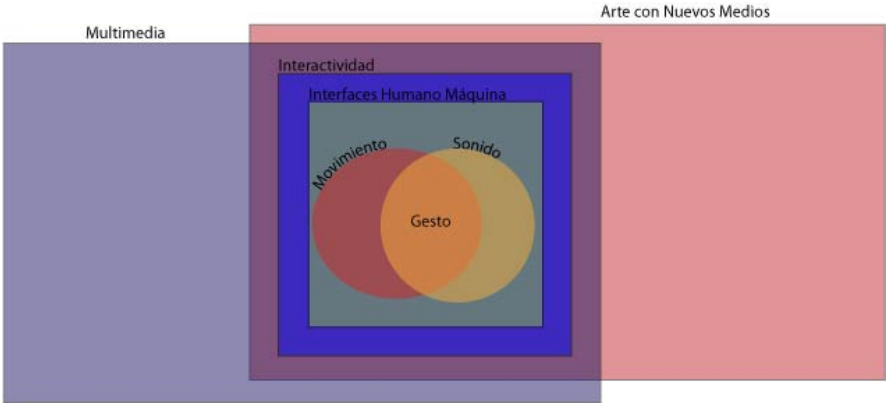

Figura 1: Contexto de las disciplinas del arte dentro del cual se encuentra la aplicación de este trabajo.

pueden colocar a *"las artes interactivas con nuevos medios"*, que a su vez abarcan *"el diseño de interfaces humano-máquina."* Para finalizar, dentro de *"el diseño de interfaces humano-máquina,"* esta herramienta permite trabajar con la concepción del *"gesto"* como un elemento que se comparte entre *"el movimiento"* y *"el sonido"*.

Recapitulando lo anterior presento en la figura 1 de manera gráfica y como diagrama de Ven el contexto en el cual se pueden considerar las aplicaciones de esta herramienta, tomando *"el gesto"* el lugar del objeto de estudio como un concepto que existe en la intersección del sonido con el movimiento.

2.1. Nuevos medios

Según M. Tribe y R. Jana [22] el termino *"nuevos medios"* fue acuñado por las grandes empresas de la comunicación, los "Mass Media," en los años noventas cuando el internet comenzó a popularizarse como un nuevo medio para transmitir información directamente al usuario. La referencia era directa al uso de la red como habilitador de un nuevo medio de comunicación.

Así, el termino nuevos medios se ha relacionado con el uso de la computadora desde sus inicios, y por frecuencia de uso y claridad de argumentación es conveniente mantener el convenio.

Ahora, considerando que el desarrollo de la computadora comienza desde principio

del siglo XX nos podríamos cuestionar sobre que tan novedoso es utilizarla.

Regresando a los inicios, en 1833 cuando Charles Babbage propone su máquina analítica, precursora de la computadora moderna, utiliza un sistema de tarjetas horadadas para representar instrucciones. Para el diseño de las tarjetas, Babbage se inspira en el telar de J.M. Jacquard que, al rededor de los años 1800, tejía telas con patrones programados en el mismo tipo de tarjetas[23].

Sin embargo en los primeros capítulos de la historia de la computadora, su objetivo era llevar a cabo cálculos matemáticos para fines científicos y militares, su acceso era restringido y su operación muy laboriosa. Conforme la computadora se fue desarrollando, el acceso a ellas se fue abriendo hasta que finalmente surgió la computadora personal. Para entonces el desarrollo de interfaces y su implementación en los sistemas operativos hacían posible que cualquiera pudiera utilizar aplicaciones como hojas de cálculo y procesador de textos sin la necesidad de conocimientos en computación. Más tarde, con la conformación del internet y el *world wide web*, fue posible navegar un espacio virtual de información libre que a través de una red de hipervínculos permitía la asociación de datos relevantes a un tema, o una búsqueda.

Esta "democratización" de la computadora y "liberación" de la información creó un nuevo mercado para aplicaciones de todo tipo, desde agendas organizadoras, correo electrónico, videojuegos en red y hasta simuladores de vuelo. Al llegar a las manos de artistas, estas nuevas tecnologías despertaron su curiosidad y sus posibilidades para la expresión comenzaron a explorarse.

Las tecnologías de la información se han desarrollado a velocidades vertiginosas ofreciendo siempre nuevas posibilidades de exploración. En este sentido, la informática digital no ha dejado de ser un nuevo medio, y las posibilidades de creación se extienden con cada avance tecnológico y el establecimiento de nuevas formas de comunicación. Manovich [23] propone que en la actualidad nos encontramos en una revolución de los "new media," ya que han venido a cambiar nuestra cultura con formas de producción, distribución y comunicación mediadas por la computadora.

Para Manovich, la estructura de los nuevos medios se separa en dos capas, una en la que la información digital desarrolla todo un lenguaje con códigos propios como:

estructura de datos, estructuras de control, memoria y apuntadores. Por otro lado se encuentra la representación humana, que por si misma tiene un lenguaje propio con todo un bagaje cultural para interpretarla, como la forma, el color, las palabras y el sonido.

La integración de los *nuevos medios* a nuestra vida cotidiana provoca que estas dos capas de comunicación se permeen entre si, influenciandose continuamente y resultando en una nueva cultura de la computadora:

> "una mezcla de significados humanos y de computadora, de formas tradicionales en que la cultura humana modelaba el mundo y el propio modo de la computadora de representarlo" [23, pg. 64].

Las tecnologías de los nuevos medios se renuevan continuamente ofreciendo condiciones diferentes para las prácticas artísticas, donde el espacio y tiempo cobran otro significado y la técnica esta estrechamente relacionada con las matemáticas y la ciencia.

2.2. Multimedia

Una de las características más importantes de la tecnología digital es que con un mismo principio de codificación se puede representar una variedad infinita de datos, ya sean imágenes, sonido, textos, luz, etc. Se comienza a digitalizar la realidad y la computadora se convierte en un medio donde la información contenida representa segmentos de ella.

> "La digitalización de los diversos tipos de señales determina, ademas, una separación de cada tipo de información de su soporte específico, permitiendo la realización de sistemas multimedia en condiciones de contener y procesar varios tipos de datos" [8, pg. 22].

Los viejos medios -la fotografía, la música, el cine, la pintura- se unifican en un mismo sistema que además de representarlos con un mismo código, puede transformarlos, sintetizarlos y extraer información de ellos a través de procesos de análisis cuantitativos.

Es en esta característica de los nuevos medios que se abre un abanico de posibilidades para la creación. Los procesos a los que se sujetan las representaciones en la computadora pueden ser diseñados para la expresión, con ellos se pueden modificar representaciones y empaparlas de nuevos significados, además se pueden combinar y explorar los vínculos que se dan en las intersecciones entre los idiomas específicos de cada medio.

La multimedia encuentra en los nuevos medios un espacio de desenvolvimiento natural, su alojamiento en un mismo dispositivo permite combinarlos y transformarlos para corresponder a propósitos expresivos y presentarlos como una obra integrada.

No propongo que la práctica de la multimedia solo sea posible con los nuevos medios, ya en el cine, el teatro y la opera se hace presente. Sin embargo los nuevos medios vienen a ofrecer posibilidades de correlacionar cualidades de representación de una manera cuasi-sinestética, como reproducir la información de una fotografía digitalizada como señal de audio o desarrollar una narrativa no lineal a través de hipervínculos y páginas web.

En resumen, se pueden procesar los medios en un espacio de representación numérica común donde pueden ser sometidos a gramáticas de procesos formales que afectan directamente su representación análoga en relación directa a otros medios. Los resultados de estos procesos son restaurados en representaciones multimedia cuyo contenido se somete al juicio estético del receptor, quién no necesariamente está al tanto de todo el desarrollo tecnológico detrás de la obra. Consecuentemente es responsabilidad del artista diseñar los procesos con que modificará o creará su información, los vínculos entre los medios y las cualidades que quiere lograr para transmitir un discurso en su obra.

2.3. Interactividad

"...cada nuevo instrumento que utiliza el hombre, por un lado, responde a exigencias ya presentes y, por el otro, transforma el contexto y el entorno."

Gianfranco Bettetini [8].

La interactividad ha tomado un papel importante en el desarrollo de las artes con nuevos medios, es un fenómeno que despierta mucha curiosidad y abre la posibilidad de que las obras incluyan en su discurso la participación activa del espectador, así como el control en tiempo real de su desarrollo. Sin embargo, el concepto mismo de interactividad no es de fácil definición, y su uso indiscriminado llega a hacerlo confuso.

La interacción es un fenómeno de la comunicación que experimentamos de forma cotidiana en las relaciones interpersonales, teóricos de la corriente del interaccionismo simbólico de las ciencias sociales como Mead, Blummer y Goffman [5], quienes se ocuparon de estudiar los procesos de interacción así como su importancia en la generación de un orden social, sin embargo, cuando se concidera la interacción con la computadora, una definición concreta se vuelve indispensable para evaluar su impacto, además de permitirnos reconocer cuando está o no presente.

En esta sección presento una definición de interactividad con la cual trabajaré por el resto de este documento y la que servirá para colocar el trabajo en un espacio donde los procesos de comunicación pueden ser ordenados según su nivel de interactividad.

En su modelo, Rafaeli presenta la idea de que interactividad es un fenómeno que puede ocurrir a través de un medio de comunicación aunque no siempre lo hace. Para que la interacción pueda suceder, el canal de comunicación debe de contar con ciertas propiedades, tomando en cuenta que según la teoría de la comunicación un canal de información no tiene que ser necesariamente digital, sino una vía de intercambio de mensajes de cualquier tipo, gestuales, palabras o cambios de voltaje en una linea, por mencionar algunos.

Para Rafaeli la interacción es

> "...una expresión de que tanto, en una serie dada de intercambios de comunicación, una tercera (o posterior) transmisión (o mensaje) se relaciona al grado al cual previas transmisiones se relacionan con transmisiones todavía más anteriores" [24, pg. 111].

La definición de Rafaeli presenta una espiral referencial recursiva, ya que para que exista interacción, debe existir un historial de los mensajes intercambiados con anterioridad y la consecuencia de éstos prevalece en los mensajes subsecuentes.

El modelo de Rafaeli es de particular utilidad cuando consideramos sistemas computacionales, pues los podemos evaluar como un flujo de mensajes discretos que se desenvuelven en el tiempo y que según su relación con mensajes anteriores determina el grado de interactividad presente. Argumenta que la interacción es una propiedad de la comunicación y no del medio, es decir, el medio puede o no permitir la interacción, pero el hecho de que esta exista depende plenamente del intercambio de mensajes y sus referencias con mensajes anteriores. Consecuentemente propone tres niveles de interacción: no interactivo, reactivo e interactivo[24, pg. 119]; diferenciados por las características de los mensajes intercambiados entre dos sujetos.

- No interactivo o comunicación bilateral: existe en canales de comunicación duplex tan pronto haya un flujo de mensajes en ambas direcciones.
- Reactivo o cuasi-interactivo: requiere, ademas de una comunicación duplex, que los mensajes posteriores hagan referencia a mensajes anteriores coherentemente.
- Interactivo: difiere de reactivo en que incorpora referencias al contenido, naturaleza, forma o presencia de referencias previas.

Rafaeli presenta gráficamente su modelo expresado en sujetos y mensajes, ver figura 2, lo que facilita su aplicación al análisis.

Para entender el modelo de Rafaeli se define P como a una persona,O como a otra y los tipos de mensajes intercambiados se distinguen de la siguiente manera:

- Mj describe mensajes intercambiados entre P y O cuya característica es que son independientes de mensajes anteriores, es decir no hacen ninguna referencia a ellos.
- O[Mj] es la situación en la cual la otra persona considera el mensaje Mj para emitir el siguiente mensaje Mj+1.

  De la misma manera, P[Mj] es la situación en la cual la persona considera el mensaje Mj para emitir el siguiente mensaje Mj+1.

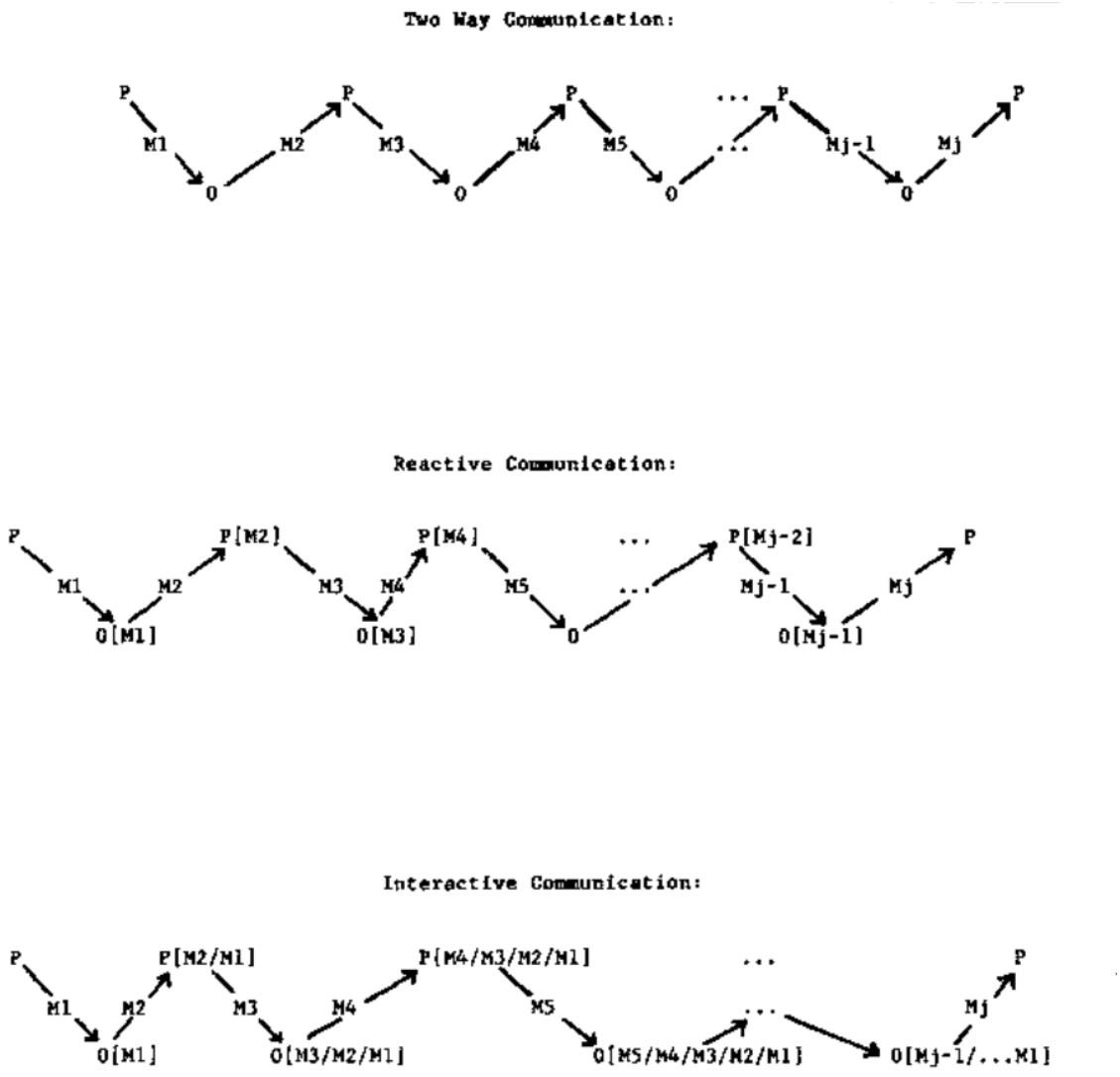

Figura 2: Modelo de Interactividad de Rafaeli. Se muestra en este diagrama los tres niveles de interacción propuestos por Rafaeli [24, pg.120].

- O[Mj-1/...M1] es la situación en la cual la otra persona considera los mensajes antes recibidos hasta el primero M1 para emitir el siguiente mensaje Mj+1.

  De la misma manera, P[Mj-1/...M1] es la situación en la cual la persona considera los mensajes antes recibidos hasta el primero M1 para emitir el siguiente mensaje Mj+1.

En este trabajo tomo el modelo de Rafaeli como punto de partida para analizar el sistema de comunicación implementado y las posibilidades que provee, tomando en

cuenta al interprete, el diseñador, el desarrollo de una obra y la percepción de las audiencias.

2.4. Interfaz humano máquina

Las Interfaces Humano-Máquina son dispositivos y tecnologías que nos permiten la comunicación con la computadora.

Es a través de las "Interfaces Humano Máquina" que es posible un constante movimiento de un modo de representación a otro. Con ellas podemos acercarnos a la computadora desde nuestros propios códigos culturales y transmitirle instrucciones sin la necesidad de preocuparnos por el proceso de traducción a *"lenguaje de máquina."* Ha sido el desarrollo de interfaces lo que ha hecho posible que la computadora se convierta en la herramienta que es hoy. Ya en 1945 Vannevar Bush, fundador del "Advanced Research Projects Agency" (ARPA), se daba cuenta de la necesidad de crear una computadora cuyo funcionamiento fuera compatible con la forma de pensar del ser humano, sin embargo fue hasta finales de los años sesentas que Duglas Engelbart presentó su "oN Line System" (NLS) con avances en interfaces como el ratón y un sistema gráfico de ventanas para editar texto, con lo que la computadora tomó su curso en el desarrollo que la transformaría en una herramienta de uso común. [25, Overture, pg. xvii].

Si en las primeras etapas de su desarrollo existió el temor de que la máquina sustituyera al ser humano en sus funciones, este fue fugaz. En cambio, la computadora tomó el papel de un dispositivo que amplía la inteligencia del hombre, que le sirve de sustento para desarrollarse más allá de lo permitido por su condición humana, una herramienta para el desarrollo de conocimiento y la creatividad. Es gracias a las interfaces que es posible que cualquier persona, sea o no experta, haga uso de ella y que sea un producto comercial de consumo común. Más, con la integración de servicios como el internet y el correo electrónico, la computadora comienza a volverse una necesidad de la era actual y toma importancia en el desarrollo de la vida cotidiana.

Las interfaces gráficas como las ventanas, los botones y los menús le permiten al usuario comunicarse con la máquina y saber en que estado se encuentra. Con los teclados y el ratón se puede ingresar información para almacenarla, transformarla y comunicarla.

Actualmente las computadoras son accesibles y transportables, se redujo su tamaño al de un cuaderno de notas, la información esta disponible desde cualquier lugar y el internet nos permite comunicarnos a grandes distancias. Los paradigmas de los medios de comunicación han cambiado, hoy en día la maquina es un medio interactivo que nos da acceso a un espacio virtual de información, conectarse ahora no implica encender un aparato receptor como la televisión, o el radio, que proveen la información en una solo dirección. Hoy en día la información se accede como un servicio, está siempre disponible y por su naturaleza duplex permite que el usuario contribuya, da lugar a la creación de una nueva forma de red social y libera las fuentes de información de las grandes corporaciones de los medios de comunicación masiva. Todo a través de un esquema de solicitudes protocolizadas inscritas en un código del que el usuario nunca se entera, simplemente da click a un botón.

Las interfaces han cambiado la forma de relacionarnos con la máquina y los otros, son un elemento primordial que determina el uso y las posibilidades de la tecnología, conforme sus diseños van siendo más intuitivos las posibilidades de explotar los nuevos medios incrementan.

Sin embargo, cuando nos enfrentamos a la computadora como un instrumento sonoro, articular el sonido de manera gestual es sin duda uno de los grandes retos. Las interfaces humano máquina que comúnmente se usan para este fin, controladores con botones y perillas, joysticks y teclados, solo nos permiten cambiar parámetros del sonido transmitiendo un grado de intencionalidad limitado y considerando la estrecha relación del gesto con el movimiento, es notable que la transmisión de intención y emotividad se reduce considerablemente con la restricción del rango de movimiento impuesto por el dispositivo.

En [26, pg. 416] se hace un breve recuento de algunos acercamiento que se han elaborado para integrar al gesto como medio de control durante la segunda mitad de los años ochenta. Desde entonces un gran cúmulo de trabajo, como demuestra J. Paradiso en [27], ha ido convergiendo hacia una nueva disciplina que responde a la necesidad de nuevas interfaces que remuevan los elementos protésicos de control.

Las NUI's o Natural User Interface implementan métodos que permiten simular una

interacción similar a la que se tiene con el mundo real, una *interacción natural*. Ejemplos de métodos implementados para este tipo de interfaces son: el reconocimiento de voz, visión computarizada, reconocimiento de patrones, superficies táctiles y realidad virtual. Actualmente hay varios grupos de investigación comercial y científica que exploran este tipo de interfaces humano máquina, entre ellos algunos destacados son: The Natural User Interface group[10], Responsive Environments Group del MIT Media Lab[11] y The OpenNI Organization.[12]

3. El gesto

3.1. Un idioma del cuerpo

El gesto como operador de la comunicación no verbal se caracteriza porque su esencia implica al comportamiento físico del ser humano, como se puede apreciar en un recuento que hace Claude Cadoz en [28] sobre diferentes definiciones del termino.

En [5, pg.169] Jeffrey Alexander expone que para George Hebert Mead, precursor del interaccionismo simbólico, la importancia del gesto en las interacciones humanas es incuestionable y lo asume como un lenguaje, definiendo interacción como una "conversación de gestos." En el mismo texto, Alexander explica que Mead considera al gesto como "los múltiples movimientos y expresiones que realiza la gente, incluyendo el lenguaje," y que es mediante los gestos que el individuo puede "simbolizar la experiencia." Estos acercamientos al estudio del gesto marcan la necesidad de comprender los mecanismos de comunicación empleados por el ser humano que escapan al lenguaje formal donde el modelo triádico de Peirce no es suficiente para resolver los procesos de significación.

> "...existe una triple conexión de signo, objeto significado, cognición producida en la mente" Peirce [29, CP 1.372].

Se puede afirmar entonces, que existe un idioma del cuerpo que se desdobla continuamente a través de un sistema de múltiples canales de comunicación, de los cuales destacan las manos, el rostro y la cabeza.

A esto, Patrizia Magli añade que:

> "... se trata de un idioma afectado a una especie de insomnio comunicativo: transmite ininterrumpidamente informaciones aún cuando su emisor esté callado o inmóvil. El cuerpo no puede no comunicar"[30, pg.38].

De acuerdo con Magli, el gesto se desarrolla como múltiples movimientos del cuerpo que se gestan desde la misma intencionalidad como regulador del discurso. El gesto apoya a la palabra proveyendo un contexto anímico, claves para su descodificación o bien mensajes concretos, como el pulgar levantado señalando "bien".

En [3] Ekman y Friesen proponen en uno de los primeros estudios estructurados del gesto, un esquema categórico mediante el cual analizar los mecanismos que emplea el ser humano para comunicar a través de lo que ellos llaman lenguajes no verbales. En sus indagatorias una de las preguntas que se proponen contestar es la existencia de un lenguaje del cuerpo y sus características.

En un esfuerzo de formalizar el estudio de este lenguaje, Ekman y Frisen presentan cinco categorías para organizarlo según sus funciones en la comunicación [3].

Las categorías son: emblemas, ilustradores, expresiones afectivas, reguladores y adaptadores.

1. Emblemas: signos cuyo significado está culturalmente establecido y puede ser directamente traducido a una o un par de palabras. Algunos ejemplos son el movimiento afirmativo de cabeza o el pulgar levantado.

2. Ilustradores: apoyan la expresión verbal, imprimen un sentido de intención al discurso matizando según el contexto, como la acentuación en ciertos aspectos que deban resaltarse o disimularse. No tienen un significado concreto por si mismos, sino que sirve de apoyo al discurso en desarrollo.

3. Expresiones afectivas: estos gestos sirven para establecer estados emotivos relacionados con los afectos primarios, se concentran principalmente en el rostro y son considerados pan-culturales, es decir, comunes a todas las culturas o universales.

4. Reguladores: no transmiten un mensaje con alto contenido de información, más bien regulan la interactividad y mantienen el canal de comunicación sincronizado,

marcando el relevo entre el papel de emisor y receptor. De esta manera se mantiene la consistencia del sistema de comunicación y el flujo en la dirección correcta. En general los reguladores son utilizados sin reconocerlos ni tener gran conciencia de ellos, sin embargo son cruciales para mantener una comunicación fluida. Por ejemplo la mirada que espera una respuesta a un mensaje previo o tender la mano para iniciar un saludo.

5. Adaptadores: se utiliza este termino debido a la creencia de que son movimientos aprendidos, se postula que a temprana edad y con el propósito de satisfacer necesidades corporales, efectuar tareas corporales, manejo de emociones, cumplir con estándares sociales o aprendizaje de actividades instrumentales. Desde un punto de vista ontogénico estos gestos nos permiten adaptarnos a un medio y aquí Ekman y Frisen proponen tres niveles de adaptación.

    - Adaptadores a uno mismo: el cuerpo en si es considerado como un medio al cual adaptarnos, el medio a través de cual se hace presente nuestra voluntad demandada por la conciencia y las necesidades del cuerpo, estamos sujetos a él, es una condición humana. Consecuentemente adaptarnos a él implica el movimiento, gestos que van siendo aprendidos y regulados. Estos gestos cobran un significado común a un grupo de personas y por lo general son muy específicos entre culturas. Algunos ejemplos serían cubrirse los ojos o limpiarse los labios con la lengua.
    - Adaptadores al medio: estos gestos son aprendidos y sirven para mantener interacciones interpersonales y prototipos de conducta social. Como adaptador nos integra en un tejido social bajo ciertas normas de comunicación. En este tipo de gesto podríamos incluir gestos típicos del coqueteo, la seducción, dar y tomar de otra persona, movimientos impulsados a atacar o bien protegerse de un ataque. Muchos de estos gestos pueden no ser desplegados plenamente, sin embargo quedan de ellos pequeñas contracciones musculares que se efectúan involuntariamente.
    - Adaptadores a objetos: son aprendidos mediante la utilización de una he-

rramienta y se pueden invocar en una conversación como un evento que lo detona y que lleva al sujeto a ponerse en la posición de uso de dichas herramientas. Algunos ejemplos de estas actividades son: manejar un automóvil, fumar y tocar un instrumento musical

Estas categorías son importantes para comprender la vasta riqueza de expresión del gesto y asimilar las acotaciones a las que el mismo diseño del proyecto nos somete.

Con la herramienta que presento se digitaliza una porción de la información emitida por el emisor y se mapea como señal de control a parámetros de síntesis y composición sonora, sin embargo el resto de la información gestual no desaparece, sino que sigue su proceso de comunicación dentro del dominio de su propio medio, con lo cual la gestualidad del gesto se mantiene íntegra como factor importante de expresión. En este sentido el publico recibe dos tipos de señales simultáneas. Por un lado se enfrenta al gesto mismo con todas las funciones comunicativas que plantean Ekman y Friesen, y por otro recibe un estímulo sonoro que responde al movimiento y que por su diseño guarda relación con las trayectorias propioceptivas que lo desatan, promoviendo una subrogación remota, que como plantea Smalley (ver sección 3.2), permite asociar el movimiento con el resultado sonoro.

3.2. El gesto en la música

La música, como el gesto, también echa a andar procesos que producen significado fuera del dominio de las palabras, en *Parallels and Nonparallels between Language and Music* [4] Jackendoff en respuesta a la pregunta: ¿Cómo son la música y el lenguaje diferentes? Propone que según su función ecológica, la principal diferencia es que la música exalta los afectos y que el lenguaje comunica un pensamiento propositivo. El gesto por su lado tiene un papel importante en la interacción no verbal que se da entre humanos. Para Mead [5] y los representantes del interaccionismo simbólico, el gesto constituye un elemento primordial de comunicación para la estructuración social, a tal punto que plantea que la actitud con que se expresa el individuo influye fuertemente la forma en que se relaciona con los demás.

En este punto es ineludible plantear el paralelo que se devela al comprobar que la música al igual que el gesto transmite algo, ya sea un estado emotivo, una señal o una intención.

Otro elemento en común que tiene la música con el gesto es que ambos están fuertemente vinculados con el movimiento del cuerpo como elemento creador, en [31, pg.16] Philip Tagg propone la siguiente definición de música:

> Música es esa forma de comunicación interhumana la cual se distingue de otras en que los estados y procesos afectivos / gestuales (corporales) experimentados individualmente y colectivamente son concebidos y transmitidos como estructuras de sonidos no verbales humanamente organizados a los mismos que generan estos sonidos y / o a otros que han adquirido la habilidad cultural, principalmente intuitiva, de descodificar significado de estos sonidos en la forma de respuestas afectivas / gestuales adecuadas.

Es de notar que en su definición, Tagg nos presenta la noción de una organización sonora no verbal que no solo impacta los procesos afectivos, sino que también involucra al cuerpo mediante los gestos, exponiendo su cercana relación con el movimiento. Toda creación musical se vale del cuerpo para generar y organizar el sonido, sin embargo cuando la creación sonora es mediada por un elemento como la computadora hay que tomar en cuenta que al proceso se le añade un nuevo elemento.

En [26, pg. 411] Jeff Pressing presenta un modelo, ver figura 3, para el proceso de tocar un instrumento, donde el movimiento humano lo controla a través de una interfaz y el instrumento suena mediante un mecanismo actuador. Entre la interfaz de control y el mecanismo actuador se lleva a cabo un proceso que convierte la información de control al formato del mecanismo actuador. Considerando al piano como ejemplo, el teclado es la interfaz de control y el mecanismo de percusión, los pedales, las cuerdas y los elementos de resonancia son el mecanismo actuador. Este proceso es válido para todo tipo de instrumentos, sin embargo, en el caso de los sistemas digitales, la transmisión de gestualidad a través de la computadora no es a través de un sistema mecánico, sino que se implica en el proceso la traducción del movimiento físico a una representación

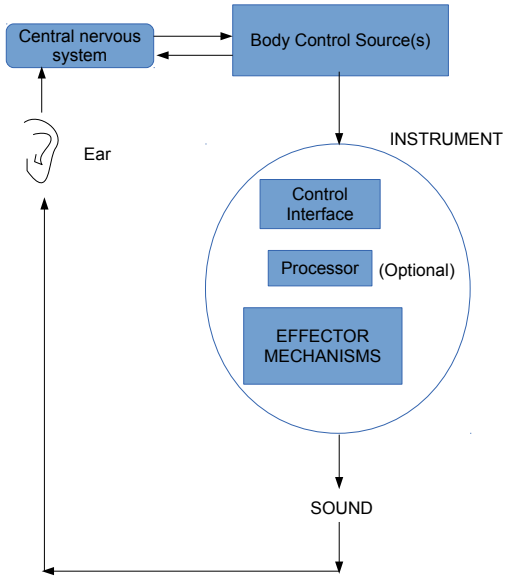

Figura 3: Boceto cibernético de una persona tocando un instrumento.

digital, que posteriormente tiene que ser interpretada y reinsertada en el mundo de lo físico. Este proceso de transducción al mundo digital ofrece la ventaja de desplegarse en un código común en el que convergen muchos tipos de información y medios de representación, como se ha mencionado anteriormente en la sección 2.2. Codificado de forma digital el gesto se puede someter a procesos numéricos que no solo lo interpreten o lo transformen, sino que también lo relacionen con otros tipos de información antes de ser regresado a una representación análoga. Sin embargo en el proceso de transducción la información digitalizable es limitada y requiere ser interpretada para que sea significativa, reduciendo dramáticamente lo transmisible, si se compara con el proceso que transfiere mecánicamente la información gestual impresa en la acción de tocar el piano.

En [19] Camurri y sus asociados hacen un ejercicio experimental para medir el impacto del gesto en la intencionalidad musical percibida en diferentes sujetos con resultados interesantes. Por ejemplo, la intensidad emotiva de distintos pasajes tiene

una alta correlación con las variaciones dinámicas y el tempo con la demarcación de los limites entre frases, sin embargo se encontró que poca relación existe con la intensidad de movimiento del torso del ejecutante.

Es interesante notar que Camurri utiliza en su marco teórico el modelo de los afectos básicos - enojo, miedo, dolor y alegría - para analizar sus resultados experimentales, concuerda muy bien con la propuesta que hace Jackendoff con respecto al papel de la música como lenguaje no verbal: *exalta los afectos*. En el mismo texto, Jackendoff plantea que entre las siete capacidades cognitivas compartidas entre la música y el lenguaje, el control fino del tracto vocal es una de ellas. Esta capacidad la plantea como una extensión del desarrollo del control fino del movimiento de la mano y su importancia para el uso de herramientas. Desde este punto de vista se puede considerar también el uso de instrumentos musicales como una extensión de esta habilidad.

Hasta aquí queda claro que con el movimiento se habilita la creación sonora, sin embargo parece ser que no es una cuestión causal unidireccional, Anthony Storr presenta en [32] como resultado de un experimento que al medir con un electromiógrafo los potenciales de actuación muscular de las piernas mientras se escucha música, estos reflejan un considerable aumento, incluso cuando el sujeto a recibido la orden de permanecer quieto.

Es un hecho que la música nos incita a movernos, seguir el ritmo con el cuerpo, mover el pie, la cabeza o simular que tocamos un instrumento cuando escuchamos música es prueba de ello y ha servido como argumento para sustentar la necesidad de una *cognición corporeizada* desde donde estudiar el papel del cuerpo en nuestro entendimiento del mundo como proponen Godøy y Leman en [33].

El vinculo tan cercano que hay entre la música y el cuerpo se devela y queda claro cuando la voz, al ser gesticulada por el cuerpo conlleva inevitablemente en su emisión mensajes con información sobre una intención o el estado del emisor, Alfred Tomatis lo expresa muy bien en el último capítulo de su libro “El oído y el lenguaje.”

> “Cantar con nuestro cuerpo es transmitir a otro nuestras sensaciones propioceptivas. Si el canto es amplio y armonioso, si es ágil y fácil, transmitirá el calor de un ensanchamiento torácico tranquilo y poderoso; si por el con-

> trario, es penoso y apretado, si es estrecho y sostenido con esfuerzo, nos aprisionará en las mismas angustias que bloquean su emisión" [34, pg. 180].

En este mismo sentido, Denis Smalley [9] sostiene que un agente humano produce espectromorfologías a través del gesto en un proceso propioceptivo que al imprimir energía en un objeto lo hace sonar de acuerdo a una trayectoria de energía y movimiento. Pero más importante para este trabajo es su propuesta de que este proceso no se lleva a cabo en una sola dirección, causa-fuente-espectromorfología, sino que es un proceso que se revierte y que nos permite a través de nuestra experiencia y formación cultural, relacionar espectromorfologías con posibles fuentes y causas, a este proceso le llama *subrogación gestual.*

Smalley propone cuatro niveles de subrogación gestual y las clasifica de acuerdo a la relación que guarda la fuente sonora con el sonido mismo y una organización musical [9, pg. 111-112].

- La de primer orden vincula cognitivamente a un sonido con su causante sin que este llegue a ser considerado música, permitiendo al escucha reconocer claramente la fuente.

- La de segundo orden implica un ordenamiento de eventos del primer orden en una forma musical en el tiempo, que devela las habilidades y estado emotivo de un ejecutante.

- La de tercer orden se aplica cuando un gesto se infiere o imagina a partir de una espectromorfología que no es plenamente reconocible y apenas se puede vincular con una causa conocida.

- Finalmente la subrogación remota se relaciona con vestigios gestuales en sonidos cuya causa o fuente son desconocidos, pero que mantienen una espectromorfología con referencias a propiedades propioceptivas, a características del esfuerzo y la resistencia percibida en las trayectorias del gesto. Esta ultima forma de subrogación es común en la música electroacústica y es un mecanismo mediante el cual

podemos relacionarnos con estímulos sonoros no antes conocidos e imposibles de encontrar en la naturaleza.

Otro esfuerzo por comprender el papel del gesto en la música lo encontramos en el trabajo de Guerino Mazzola y Moreno Andreatta [35], donde se propone una base de estructuras comunes entre matemáticas y música que permite establecer una relación entre las dos disciplinas.

Mediante la aplicación de gráficas dirigidas ó digráficas, se plantea que dos recorridos diferentes que desemboquen en el mismo lugar se pueden considerar como los dos lados de una ecuación, donde el recorrido puede ser visto como el movimiento gestual, incluyendo así el movimiento corporal de la ejecución como un elemento formalizado de la estructura musical.

4. Herramientas para la implementación

Para seleccionar las herramientas con las que se desarrollé este proyecto fue necesario conocer y evaluar los diferentes programas y lenguajes disponibles y tomar algunas consideraciones tanto técnicas como éticas para su selección.

Dentro de las consideraciones técnicas para seleccionar las herramientas busqué que fueran lo suficientemente maleables para implementar sistemas modulares de diseño propio y que pudieran comunicarse entre si. Las herramientas deben permitir desarrollar programas cuyo rendimiento de C.P.U sea suficiente para evaluar respuestas a señales de entrada en un tiempo mínimo, suficiente para poder apreciar inmediatez o tiempo real.

Preferí el software de código abierto porque en su filosofía implica la posibilidad de utilizarlo libremente para desarrollar diseños de sistemas propios con la posibilidad de intervenir el programa mismo. Los programas que seleccioné tienen la ventaja de que cuentan con un lenguaje desarrollado para programar aplicaciones que satisfagan la necesidad de proyectos específicos sobre una plataforma que facilita cierto tipo de tareas. Por ejemplo, SuperCollider implementa una estructura que optimiza el trabajo con audio, OpenFrameworks cuenta con un framework que optimiza el trabajo con

visión computarizada y generación de gráficos, y OSC es ideal para la comunicación entre programas.

Dentro del software para audio se consideraron varias opciones: Max/MPS, PureData,Chuck, Csound y SuperCollider. Para visión computarizada y generación de gráficos: vvvv, Isadora, Quartz Composer, Processing, EyesWeb y OpenFrameworks.

Es importante aclarar que cada uno de los programas que consideré cuenta con una amplia gama de posibilidades y herramientas que no siempre existen en otros, así que no descartaría trabajar con ellos en otros proyectos.

Así también considero que por empatía a las implicaciones de la filosofía de código abierto era responsable preferir aplicaciones de este tipo en cada ocasión.

A continuación presento algunos de los conceptos que caracterizan a las diferentes herramientas y que tuve que tomar en cuenta, seguido de una breve presentación y argumentación para su selección. Una descripción de las herramientas seleccionadas también se incluye en los apéndices, invito al lector a revisarlas si se quiere conocer más detalle sobre ellas.

4.1. Código abierto

Hoy en día existen muchas herramientas disponibles para la creación digital interactiva en computadora, muchas de ellas son desarrolladas por compañías para su comercialización y otras, las que usaremos y presentaremos aquí, son desarrolladas por un conjunto de colaboradores que conjugan su trabajo a través de sistemas CVS (Current Version System) que se alojan en la red y permite su integración en un sólo código.

La inherente forma abierta de estos programas permite la participación de quien quiera contribuir y tenga la facultad para hacerlo, además de permitir su distribución gratuita y fomentar la generación de un conocimiento grupal, abierto y de acceso a cualquiera que tenga el interés en buscarlo: Código Abierto (OpenSource).

De igual importancia es el hecho de que este tipo de herramientas (conocimiento) no tienen un dueño, lo que las convierte en publicas y de acceso fácil, los programas se pueden bajar de la red listos para ser ejecutados. Una simple mención de las personas que colaboraron con el programa es suficiente para tener el derecho moral de utilizarlo

para una aplicación propia: Creative Commons.

Éstos valores se contrastan con los principios que se deducen de los programas comerciales, los cuales no comparten su código y venden su producto bajo licencias que sólo permiten el uso de su versión corriente y cobran extra por cada actualización.

Sin embargo el uso de programas de código abierto no siempre es fácil y su aprendizaje requiere de mucho interés y práctica. Muchos grupos de desarrollo tienen foros abiertos en internet que invitan a hacer preguntas y que son contestadas por usuarios expertos y desarrolladores, lo que genera una base de conocimiento que va creciendo conforme la comunidad de usuarios se va extendiendo. No está de más mencionar que la popularidad de cada proyecto y el nivel de participación de la comunidad tiene implicaciones directas en su nivel de desarrollo. Mientras mas grande y activa sea la comunidad hay mayores posibilidades de que el programa sea mejorado en poco tiempo.

Se ha decidido trabajar con programas de código abierto por ser de acceso gratuito, la mayoría son multi-plataforma (son ejecutables en la mayoría de los sistemas operativos actuales) y cuentan con una comunidad de apoyo accesible y confiable, lo que permite que durante la implementación de un programa se cuente con asesoría técnica constante. Además, trabajar con código abierto permite expandir el sistema y añadir nuevas funcionalidades que le podrían ser útiles a otros proyectos, contribuyendo así a la comunidad.

La construcción de conocimiento y las posibilidades de crear se multiplican cuando se cuenta con la voluntad de aprender y compartir ideas, los programas de código abierto nos permiten esto y contribuyen con una alternativa loable a la forma en que se lucra con el conocimiento en el momento en que se hace privado e impone una cuota para tener acceso a él. Por esta misma razón el código abierto ha tomado un lugar muy importante en el desarrollo de proyectos creativos, la posibilidad de acceder a herramientas programables se presta para el desarrollo de aplicaciones específicas cuya única función se justifica dentro del discurso de una obra. Con ellas el artista puede acceder a todos los rincones de la programación y explorar libremente sus necesidades expresivas, abriendose camino a través de los retos que impone la conformación de una obra.

4.2. Estrategias de programación

Por lo general los diferentes sistemas para programar se han establecido con el propósito de satisfacer las necesidades que surgen para soportar un tipo de información dado, como el sonido o la imagen.

Esto se debe a que las características de cada medio imponen una gramática diferente según las lógicas de su representación digital. Por lo que un ambiente para programar aplicaciones sonoras debe ser muy eficiente y confiable para su ejecución en tiempo real, y permitir un esquema de calendarización de procesos que permita su ejecución en el orden correcto, ya que de ello depende el flujo de la información. En contraste, ambientes enfocados en la imagen deben ser muy eficientes, y permitir acceso a los chips de gráficos para optimizar el proceso de calcular imágenes y actualizar los pixeles en la pantalla.

Además, cada lenguaje de programación implementa un vocabulario pertinente al medio que mejor soporta, de tal manera que si su enfoque es visual encontraremos herramientas para dibujar primitivos geométricos o estructuras de datos que faciliten las operaciones entre imágenes. Por otro lado, lenguajes enfocados al sonido tendrán sintetizadores de forma de onda o filtros dentro de su vocabulario. Comúnmente los proyectos de multimedia interactivos desarrollan módulos independientes que se comunican entre si conformando un sistema integral.

Los sistemas de desarrollo que se discuten pueden ser considerados como lenguajes de programación, pues tienen un vocabulario y una gramática que les permite acoplarse a una cantidad infinita de aplicaciones por medio de construcciones lingüísticas que describen un comportamiento o programa. Una diferencia importante entre los programas que se tomaron en cuenta son los tipos de interfaz que implementan para desarrollar proyectos, a continuación se hace una distinción entre el tipo de herramientas según su presentación al usuario. Cabe destacar que el tipo de interfaz implicada en el desarrollo de un proyecto tiene implicaciones directas en el proceso creativo de la obra.

4.2.1. Interfaces gráficas

Hay programas que entregan el control de parámetros al usuario mediante una serie de elementos gráficos como ventanas y botones que permiten cambiar sus estados afectando directamente las variables que controlan. Este tipo de Interfaces Gráficas limitan al usuario a trabajar con las posibilidades del programa, volviendo la herramienta un tanto rígida y limitada para aplicaciones que requieran otro tipo de implementación. La popularidad de estos programas reside en su facilidad de uso ya que no demandan un conocimiento extenso de la tecnología en cuestión y su tiempo de aprendizaje es rápido.

Para aplicaciones interactivas novedosas este tipo de programas se vuelven limitantes y la mayoría carece de métodos para comunicarse con otras aplicaciones. Consecuentemente en este trabajo no se discute este tipo de software.

4.2.2. Programación gráfica

Como ya mencioné, para lograr aplicaciones interactivas novedosas es necesario un control de software más allá de lo provisto en una interfaz gráfica. De hecho, el grado de control necesario ya implica la programación de nuevas aplicaciones diseñadas específicamente para cada obra.

Una de las formas en que se ha buscado facilitar la programación al creciente número de artistas digitales es mediante la programación gráfica.

La programación gráfica implementa una serie de objetos (vocabulario) representados gráficamente como cajas con el nombre del objeto dentro o una figura representativa de la función que lleva a cabo el objeto. Así cada caja representa un proceso que se lleva a cabo sobre una señal o un grupo de datos. Estos datos son entregados al objeto como una entrada y el resultado es devuelto como salida.

Al grupo de objetos disponibles para hacer una programación se le llama librería. Las librerías pueden ser categorizadas según el tipo de aplicación para la cual están diseñadas.

Al interconectar objetos se logra la programación de un algoritmo complejo (programa) que consiste de varios procesos y responde a las necesidades especificas del programador. En este forma de programación el tipo de objetos disponibles en las librerías es

crucial para el proceso de creación, ya que si uno pretende implementar una aplicación que implique un proceso que no existe en la librería se encuentra en problemas.

La gran mayoría de paquetes de programación gráfica permite al usuario participar con la creación de objetos propios, sin embargo su programación requiere del manejo de algún lenguaje de programación de alto nivel como C++ o Java.

4.2.3. Programación con lenguajes de alto nivel

Los lenguajes de programación de alto nivel como C++, Java y Lisp adquieren su nombre por la relación que mantienen con los comandos que son ejecutados por la computadora a nivel de máquina.

Un lenguaje de programación de alto nivel se compone de comandos definidos por palabras de lenguaje natural, facilitando su comprensión y aprendizaje.

Para poder ejecutar un programa escrito en uno de estos lenguajes es necesario traducirlo a lenguaje de máquina, comúnmente llamado *ensamblador*.

Ensamblador es una serie de comandos definidos a nivel de bits que permiten llevar a cabo las operaciones básicas que conforman todos los programas. Es por esto que se dice que el lenguaje de máquina es un lenguaje de bajo nivel y lenguajes como C++, SmallTalk, Lisp, Prolog y Java son de alto nivel.

En este texto se distingue entre dos tipos diferentes de lenguajes de alto nivel:

- Los que son interpretados

- Los que son compilados

Los primeros son programas que permiten la ejecución inmediata de su código. Cuando se escribe un comando este puede ser traducido a lenguaje de máquina y ejecutado sin la necesidad de conocer el resto del programa. Este proceso es similar a lo que hacemos cuando resolvemos un problema utilizando una calculadora. Se van ingresando los números a la maquina y el resultado es utilizado para el siguiente cálculo.

En comparación, los lenguajes compilados toman todo el código de un programa, lo traducen a lenguaje de máquina y nos entregan un archivo ejecutable. Este archivo

contiene el programa que al ser ejecutado echa a andar los procesos para los que fue creado.

Los lenguajes de alto nivel producen programas independientes cuya ejecución es más rápida, lo que los hace ideales para aplicaciones en tiempo real.

Para una buena revisión del funcionamiento de una computadora y sus lenguajes de programación se puede consultar el primer capítulo de [36] y el segundo de [37].

4.3. Selección de herramientas para el proyecto

4.3.1. Programación musical: SuperCollider

La selección de SuperCollider como plataforma para el desarrollo de toda la programación sonora se basa en la posibilidad que ofrece como un lenguaje de programación de alto nivel para aplicaciones musicales. Permite implementar estructuras complejas y algoritmos de composición que en lenguajes de programación como *Max* es difícil lograr. Con respecto a esto Miller Puckette comenta lo siguiente:

> "Toda la noción de control de flujo, con ciclos, condicionales y subrutinas, es fácil de expresar en lenguajes de texto, pero hasta ahora, los lenguajes de programación gráfica no han encontrado la misma fluidez o economía de expresión que tienen los lenguajes de texto." [1]

Las estructuras de control como *if, while, do...* son fáciles de codificar, cuenta con varias estructuras de datos para alojar información en arreglos de fácil acceso, su diseño orientado a objetos permite crear clases y organizar la información en grupos de datos complejos. Cuenta con sistemas para agendar eventos sincronizados por relojes implementados sobre el reloj del CPU y ejecutados en hilos diferentes, y métodos para compartir un reloj en la red, con lo que se pueden sincronizar eventos con máquinas remotas.

Sumado a todo esto, la eficiencia con que se ejecuta la síntesis de sonido y el creciente número de librerías que la comunidad aporta, permite arriesgarse a decir que SuperCollider es una herramienta tan completa que es difícil llegar a los límites de sus

capacidades. En caso de sentirse limitado por lo que puede hacer por nosotros, cuenta con una API [1] para programar UGens en C++ y expandir el lenguaje.

4.3.2. Programación visual: OpenFrameworks

La gran ventaja que tiene OpenFrameworks sobre ambientes de programación como Processing y Max/MSP es que se salta la necesidad de una Máquina Virtual. Se programa en C++, que se ha convertido en un lenguaje estándar con compiladores disponibles en prácticamente cualquier sistema operativo. La traducción del código a lenguaje de máquina es de un solo paso, rindiendo mucho más rápido la ejecución, lo cual es una ventaja si se quiere poner en marcha varios procesos de alto consumo de CPU cómo Visión Computarizada, Simulación de Fluidos, Interacción y Multimedia en *tiemporreal*.

OpenFrameworks funciona mediante un bucle infinito, es decir, una función que está constantemente siendo ejecutada y dentro de la cuál se llevan a cabo todos los cálculos de actualización del estado del programa. Al ser muy eficiente, OpenFrameworks ha crecido rápidamente en la comunidad de artistas de nuevos medios e interactividad. Sin embargo OpenFramewoks tiene una desventaja, implica programar en C++, un lenguaje de programación que podría considerarse difícil, sin embargo OpenFramewoks simplifica considerablemente el proceso de creación implementando funciones especificas para la creación de gráficos, manejo de video, interactividad con dispositivos periféricos y comunicación, entre otras.

Al estar escrito en C++, OpenframeWorks comparte lenguaje con muchas de las bibliotecas de programación más eficientes. Entre ellas, para procesos de gráficos se encuentra OpenGL y para visión computarizada OpenCV que, como se puede intuir por sus nombres también son de código abierto. Presentamos una breve descripción de estas dos bibliotecas como parte de los apéndices.

OpenFrameworks tiene ya en su versión 0.7 una gran comunidad de usuarios y un foro muy activo donde es posible interactuar con otros usuarios, intercambiar ideas,

[1] Application Programming Interface: es una interfaz desarrollada para permitir la comunicación entre lenguajes de programación durante la implementación y compilación de una aplicación[38]

ayudar y ser ayudado [39].

4.3.3. Open Sound Control (OSC)

El intercambio de información entre los módulos del sistema se lleva a cabo mediante mensajes en protocolo Open Sound Control (OSC), ver apéndice M para una definición de protocolo. La comunicación OSC es ideal para la transmisión de mensajes entre programas dentro de una misma computadora, a través de sistemas de redes locales LAN e Internet. A demás su implementación ya se encuentra integrada en OpenFrameworks y SuperCollider. Una descripción completa de los estándares del protocolo, más una serie de áreas de aplicación donde se ha utilizado puede consultarse en [40]. En el apéndice N hago una breve descripción del protocolo y en la sección 4.2. expongo su aplicación en éste proyecto. La selección de OSC ante MIDI se justifica principalmente por su mayor definición de mensajes, MIDI comunica con 8bits mientras que OSC utiliza los 32bits disponibles por la computadora, más aún OSC cuenta con la posibilidad de transmitir diferentes tipos de datos y se conjuga con protocolos de comunicación standard de internet como es TCP/IP (ver apéndice M), mientras que MIDI solo transmite números enteros y tiene que ser adaptado para comunicarse por redes. En el proyecto que presento los tipos de datos que se transmiten son variados y no todos describen propiedades del sonido, también se transmite información visual y descriptores de movimiento.

5. Diseño y desarrollo

Introduzco en esta sección un esquema de las partes que integran el sistema desarrollado en este trabajo. Comprende tres subsistemas que comparten entre si información sobre el movimiento. Ésta es obtenida a partir de estrategias de visión computarizada para la *percepción*[2] del espacio. Los descriptores obtenidos son empacados en mensajes OSC y se envían al resto de los subsistemas para ser utilizados como señal de control.

[2]Considerar que una computadora puede percibir, es una aseveración que se usa como una metáfora que pone énfasis en uno de los objetivos más importantes de las ciencias de la computación: la inteligencia artificial

Los subsistemas que comprenden el diseño son:

- Imagen
  - Captura y análisis de movimiento.
  - Despliegue visual y síntesis de gráficos.
- Audio
  - Captura, síntesis, análisis y despliegue de audio.
- Intercomunicación entre subsistemas OSC.

El primer subsistema esta compuesto de dos módulos. El primero se dedica a la captura y análisis de imágenes en tiemporreal y el segundo toma la información obtenida para visualizarla.

El primer modulo integra las bibliotecas ofxOpenCv y ofxKinect en OpenFrameworks. En concreto ofxKinect nos permite acceder a las funciones integradas en el dispositivo $Kinect^{tm}$, como la captura de imágenes de profundidad, rgb y control del motor interno del dispositivo. Con ofxOpenCv se puede acceder a la biblioteca OpenCV que nos permite hacer análisis de imágenes secuenciales en tiemporreal e implementar algoritmos de visión computarizada con una latencia mínima.[3] En este proyecto implemento un algoritmo de rastreo de blobs para analizar el movimiento de las manos y un algoritmo para obtener posiciones en el blob donde es plausible que haya dedos.

El segundo módulo implementa una visualización que ilustra los resultados del análisis, esto es un buen método de retroalimentación que le permite al usuario tener acceso visual a los datos que se obtienen. En la implementación de este segundo módulo propongo una presentación estética de los datos que demuestra una sencilla forma de abordar este medio. Este subsistema se programó en C++ sobre la plataforma que proporciona OpenFrameworks.

[3]Latencia es un concepto muy importante en el ámbito de cómputo en *tiemporreal* e interacción. Se refiere al tiempo que transcurre desde que la máquina recibe un paquete de datos, los procesa y nos entrega el resultado. Esto es de particular importancia cuando trabajamos con interacción pues de eso depende que el sistema de la impresión de que reacciona inmediatamente al estímulo.

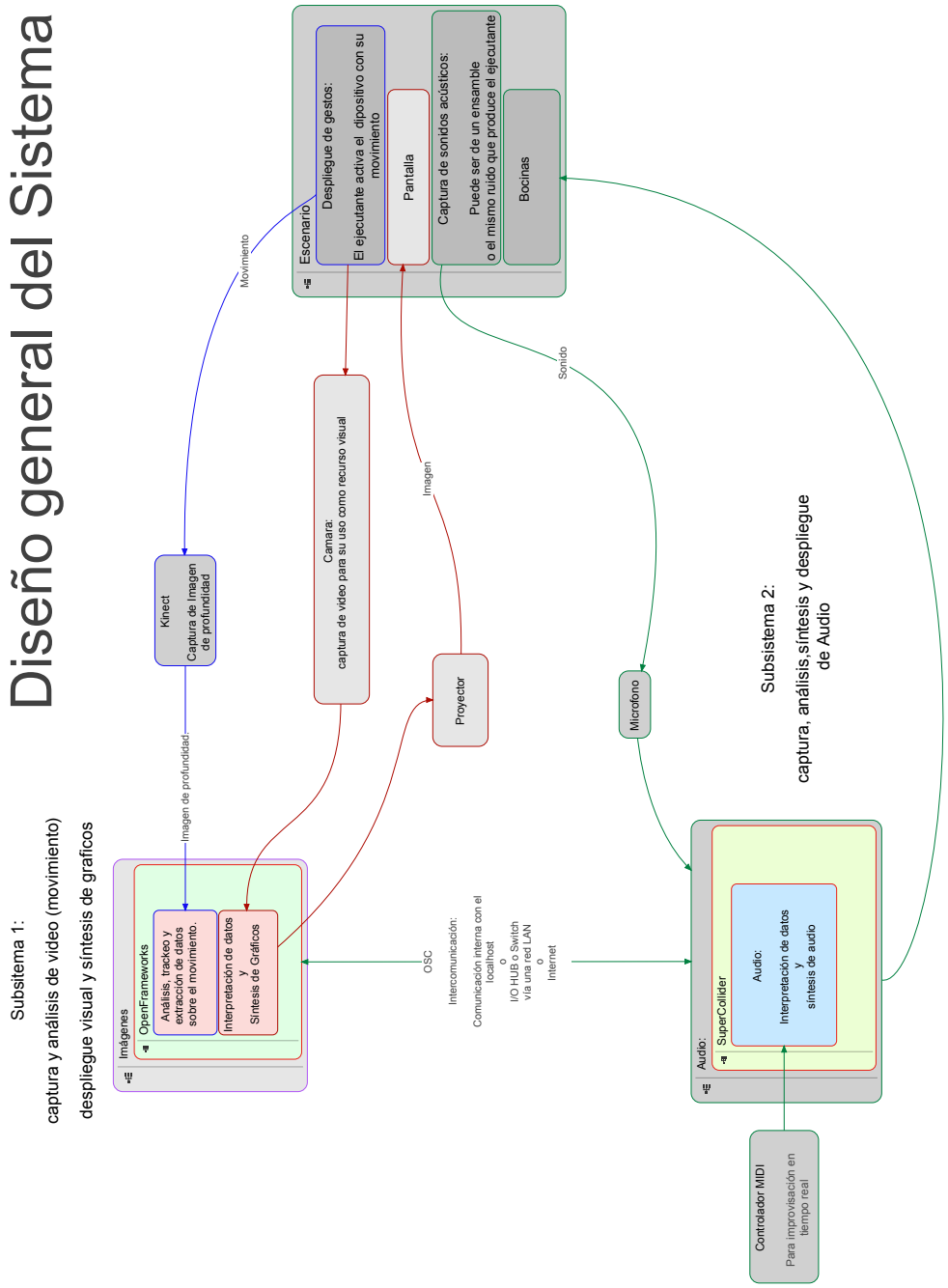

Figura 4: Diseño modular del sistema interactivo con sus subsistemas definidos.

El segundo subsistema se implementó en SuperCollider y lleva a cabo todos los procesos de audio, así tengan que ver con entradas, salidas, análisis o síntesis.

Para la representación de los datos obtenidos del primer subsistema definí y programé un objeto tipo Blob. Éste estructura la información en un símbolo, integrando los descriptores del movimiento y un tag que lo diferencia de otros objetos. También implementé un objeto *ReceiveBlobs*, que recibe los datos y los asigna a su blob correspondiente. De esta manera fue posible mantener una consistencia entre los datos que produce el rastreo del primer subsistema con los datos que representan cada blob en SuperCollider.

La comunicación entre estos dos subsistemas se efectúa mediante mensajes OSC que además de transmitirse dentro de la computadora podrían transmitirse a través de internet, como se puede apreciar en el apéndice N. Esto abre las posibilidades a que cada módulo sea ejecutado en una computadora diferente, reduciendo la exigencia de cada procesador. O que la comunicación ocurra entre dos localidades remotas, y ejecutar una improvisación telemática.

La comunicación es el tercer subsistema. Implementa un lenguaje común entre programas cuya forma de codificación permite que los mensajes puedan ser interpretados por el receptor. De esta manera se define un vocabulario de interacción entre los subsistemas, de forma que para cada mensaje hay al menos un receptor y un descodificador que lo puede interpretar, ya sea como un evento o una variable de control.

5.1. Subsistema 1: imagen

5.1.1. Captura de mapa de profundidad

El dispositivo para obtener el mapa de profundidad o $DepthMap$ es el $Kinect^{tm}$, que por estar disponible abiertamente al mercado es de fácil acceso. Para su uso en desarrollos independientes existen bibliotecas de código abierto que implementan controladores del dispositivo y nos ofrecen acceso a la información que captura.

En el cuadro 1 presentamos algunas de la especificaciones del $Kinect^{tm}$.

Los detalles del funcionamiento del dispositivo no están disponibles al público ya

Cuadro 1: Algunas especificaciones del $Kinect^{tm}$ [41][42].

Campo visual	58º H, 45º V, 70º D
Camara con filtro pasa IR	VGA(640x480) a 11bits y 30fps
Resolución espacial a 2m de distancia del sensor	3mm
Resolución de profundidad a 2m de distancia	10mm
Camara RGB	VGA(640x480) a 8 bits y 30fps
Proyector IR	LED Láser tipo 1
Rango de operación	0.8m a 3.5m

Figura 5: Patrón Moteado, Kinect[45]

que a la fecha son protegidos por patentes, sin embargo hay información en el internet que permite indagar sobre su funcionamiento. Con tecnología desarrollada por la compañía Israelí PrimeSense[43] John MacCormick del Dickinson College presume que la generación de una imagen de profundidad 2D se logra con la proyección de un patrón moteado que posteriormente se analiza con algoritmos de luz estructurada, paralaje de profundidad de visión estereoscópica y profundidad por diferencia de enfoque[44].

Los sistemas de luz estructurada proyectan patrones conocidos sobre objetos y deducen su estructura 3D a partir de la deformación del patrón sobre el objeto.

El análisis de paralaje obtiene información de profundidad al comparar la diferencia de ángulos entre el patrón moteado proyectado y la imagen capturada por la cámara con filtro pasa IR. La diferencia de ángulos se debe a la diferencia de posiciones en que se encuentran ambos elementos. Finalmente la diferencia de enfoque se basa en el hecho

Figura 6: Ejemplo del uso de Luz Estructurada[44]

de que los puntos del patrón moteado se enfocan diferente según la distancia a la cual se proyecten. Mientras más lejos se proyecte, más difuminado se verá el punto[44].

Si bien no tenemos mucha información sobre el dispositivo, los mapas de profundidad que entrega son de mucha utilidad para procesos de visión computarizada. Con este tipo de imágenes podemos llevar a cabo procesos de discriminación entre fondo y figura sin necesidad de aplicar algoritmos complicados.

El dispositivo se conecta a la computadora vía USB de manera que integrarlo en proyectos de este tipo es muy viable.

5.1.2. Análisis de movimiento

En esta sección se describe la implementación de un algoritmo para discriminar los objetos y rastrear su movimiento.

- Selección de datos a medir

  El rastreo de los blobs es esencial pues permite etiquetarlos y así calcular su velocidad como un vector 3D. Al considerar información 3D de los blobs se amplía considerablemente la cantidad de información contenida en los datos sin que el aumento de procesamiento sea demasiado. Así, los vectores no solo ofrecen información de la posición de los blobs en un momento determinado, también puede accederse a información sobre la dirección y forma de su trayectoria.

  Considerando la posición anterior del blob podemos calcular un vector de velocidad. Manteniendo un registro de velocidades en el tiempo de puede calcular la aceleración. Como se puede ver, rastreando una sola variable se tiene acceso a 3 diferentes tipos de información:

  - posición (x, y, z)

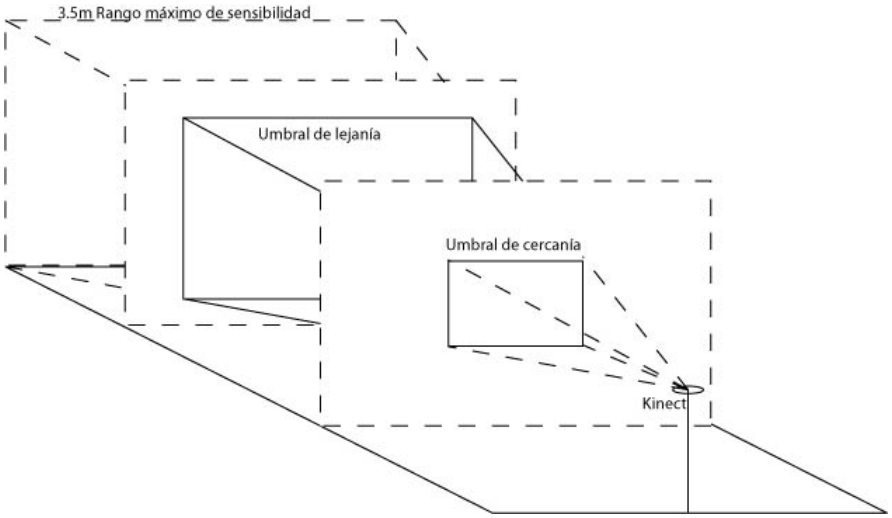

Figura 7: Aplicación de umbrales para la segmentación de los objetos que podrán ser rastreados

- velocidad (x, y, z)
- aceleración (x, y, z)

Este tipo de información es rica en significados, por ejemplo, con la velocidad no solo sabemos que tanto se mueve en x, y y z, también sabemos la dirección del movimiento, su amplitud, la dirección en la qué cambia y su rapidez de cambio.

Finalmente, con el objetivo de tener más definición en la descripción de gestos de la mano decidí implementar un algoritmo para medir la curvatura interna de los blobs rastreados y determinar en que posiciones es plausible la presencia de un dedo. Hasta el momento la posición de los dedos es considerada en 2D y para su rastreo se implementa el mismo algoritmo implementado para el rastreo de blobs.

- Segmentación con umbrales

  La segmentación es importante pues nos permite distinguir los objetos que vamos a rastrear del resto de la escena. Cuando una segmentación es adecuada facilita considerablemente la tarea de rastreo y evita tener que confrontar problemas de eliminación de ruido en la información. Es muy importante el papel del $Kinect^{tm}$ que, al entregarnos una imagen donde cada pixel indica la distancia de la cámara

al punto que representa se puede delimitar el espacio que vamos a analizar a una especie de ortoedro o caja que discrimina todo lo que no se encuentra dentro de él. La asignación del espacio a analizar se define con la aplicación de umbrales a la información de distancia. Un umbral es un límite al rango dinámico de una variable, de tal manera que toda la información que se encuentre por encima o por abajo del umbral, según la definición, no es considerada. En éste caso se aplican dos umbrales a la información, uno que delimite la cercanía al sensor y otro que delimite la lejanía. Así el valor del umbral de cercanía delimitará la distancia después de la cuál se tomará en cuenta la información y el umbral de lejanía marcará la distancia después de la cual ya no se tomará en cuenta la información.

En la figura7 se muestra con lineas solidas el espacio que se analizaría después de aplicar los umbrales. La calibración de los umbrales se lleva a cabo *in situ* con el teclado de la computadora, de tal manera que con las teclas [>] y [<] se controla el umbral de lejanía y con las teclas [-] y [+] se controla el umbral de cercanía. Esto es muy útil, pues permite que el programa se adecue a las circunstancias de cada lugar.

- Detección de blobs. Los blobs son conjuntos de pixeles del mismo color (blancos en este caso) que están conectados entre si por una regla de adyacencia-8, de tal forma que para un pixel p que se encuentra en la posición (x, y) los pixeles en las posiciones $(x-1, y), (x-1, y-1), (x, y-1), (x+1, y-1), (x+1, y), (x+1, y+1), (x, y+1), (x-1, y+1)$ son adyacentes, esto se contrasta con la regla de adyacencia-4 en la cual los pixeles diagonales $(x-1, y+1), (x-1, y-1), (x+1, y-1), (x+1, y+1)$ no se consideran [46, pg. 66 y 536].

  La detección de blobs se lleva a cabo con la biblioteca de ofxOpenCv que implementa la detección de blobs en una clase llamada ofxCvContourFinder y representa los blobs con una clase propia llamada ofxBlob.h que permite acceder a todos los datos que lo describen: posición (x,y) de su centroide[4], área ocupada por el blob, posición y dimensiones del rectángulo que contiene al blob, contorno

[4]El centroide de un blob se define como su centro de gravedad

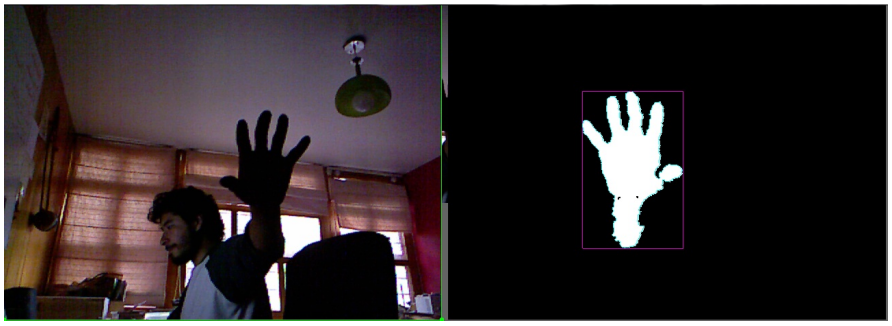

Figura 8: Ejemplo de detección de blobs en una imagen. A la izquierda se encuentra una imagen de la escena y a la derecha la imagen que se obtiene después de aplicar umbrales y reconocimiento de blobs.

que delimita al blob y si detecta huecos dentro de un blob.

- Rastreo

  El algoritmo de detección de blobs de OpenCV los ordena en un arreglo de acuerdo a su tamaño, colocando el más grande en la posición cero, de tal forma que si se reduce su tamaño cambia de posición y no hay forma de saber en que lugar se encontrará al siguiente tick del reloj ni de dónde venía, es decir, la detección de blobs no realiza ningún rastreo del movimiento de los blobs en el tiempo. Este es en si un problema que no se relaciona a la detección de blobs, tenemos dos imágenes consecutivas con un grupo de blobs y nos gustaría saber que blobs en ambas imágenes representan el mismo objeto. Este tipo de análisis se conoce como rastreo y es un problema común de visión computarizada.

  Hay dos acercamientos comunes para resolver este tipo de problemas, uno tiene que ver con la extracción de rasgos específicos del objeto a rastrear y buscar su posición en imágenes consecutivas, el otro toma ventaja del hecho de que en imágenes consecutivas los objetos a rastrear no cambian tan abruptamente, por lo tanto es viable primero detectar la posición del objeto y luego buscarlo en las imágenes subsecuentes[47].

  Para este proyecto implementé un algoritmo muy sencillo que opera bajo dos

presunciones y entrega resultados consistentes.

- Primera presunción: la posición de blobs que representan el mismo objeto no cambia tan rápidamente en imágenes consecutivas, por lo tanto se puede considerar la distancia entre blobs de dos momentos consecutivos como un indicador para asignar correspondencia. Consecuentemente, se compara la posición (x, y) del centroide de un blob en una imagen al tiempo t con la posición de los centroides del grupo de blobs en la imagen al tiempo $t + 1$ y asumimos que le corresponde el que produzca la menor distancia.
- Segunda presunción: el área del rectángulo que contiene al blob que representa un objeto, no cambia tan rápidamente de un momento a otro, por lo tanto se puede presumir que el área de un blob al tiempo t va a ser similar al área que el blob generado por el mismo objeto tenga al tiempo $t + 1$. Así se puede utilizar esta información para reducir ambigüedades que se presenten con la primera presunción en caso de que dos objetos se crucen o que sus centroides se acerquen mucho.

El código para implementar el rastreo está escrito en C++ sobre la plataforma de OpenFrameworks, a continuación se describe el algoritmo tal como fue implementado.

La meta del código que aquí se presenta es poder etiquetar los blobs detectados adecuadamente después de la aplicación de umbrales por el objeto $contourFinder$, conforme se van moviendo. Puede saberse que se ha logrado el objetivo porque se pueden desplegar los blobs detectados y mostrar que su número de etiqueta a través del tiempo apunta al mismo blob sin importar su tamaño.

El algoritmo se implementó en la clase Detect.h

Detect.h En la clase Detect se llevan a cabo el proceso de captura de mapas de profundidad, aplicación de umbrales, detección de blobs y rastreo. Para poder

llevar a cabo el rastreo se creó una estructura de datos de tipo *struct*[5] llamada *taggedBlob*. Esta estructura se implementa para contener los datos que describen las propiedades de un blob, tales como:

- Número de etiqueta asignado por el rastreador: tag.
- Objeto del blob a tiempo t: blob.
- Objeto del blob a tiempo $t - 1$: prevBlob.
- Objeto del blob a tiempo $t - 2$: pprevBlob.
- Velocidad (x, y, z) del blob: velX, velY, velZ.
- Aceleración (x, y, z) del blob: accX, accY, accZ.
- dedos[n](x, y): vector que contiene la posición y etiqueta de los posibles dedos.

El algoritmo de rastreo ejecuta en la función *sortBlobs*() cuando en el mapa de profundidad se detecta que uno o más blobs aparecen. El objeto *contourFinder* es utilizado con este propósito y entrega un vector llamado *blobs* ordenado de más grande a más pequeño, según su área.

En este primer momento se crea una nueva instancia de *taggedBlob* para cada blob detectado por el *contourFinder* y se ordenan en un vector llamado *sortedBlobs*, utilizando su posición para asignarle un número de etiqueta, evitando así etiquetas repetidas. En este momento se cuenta sólo con la posición de los blobs y su etiqueta asignada, es decir, todavía no se aplica ningún tipo de rastreo.

El siguiente momento ocurre cuando el dispositivo $Kinect^{tm}$ captura el siguiente mapa de profundidad, se buscan los nuevos blobs y se comparan con los blobs previamente detectados.

La comparación se lleva a cabo creando dos arreglos 2D, uno que representa en cada columna la distancia entre un blob en el mapa actual con cada *taggedBlob*

[5] "struct" Estructura de datos que C++ ofrece para describir una variable con un tipo de dato personalizado que puede contener en si variables de cualquier tipo de datos, con lo cual es posible asignar a una variable una serie de propiedades.

en el vector de *sortedBlobs*; y otro que representa en cada columna la diferencia de áreas entre un blob en el mapa actual con cada *taggedBlob* en el vector de *sortedBlobs*. Los arreglos se normalizan con respecto a su valor máximo y se mezclan en un arreglo de costos de la siguiente manera.

$$\mathbf{C} = \mathbf{D} * \alpha + \mathbf{A}(1 - \alpha) \tag{1}$$

Donde C representa el arreglo de costos, D el arreglo de distancias, A el arreglo de diferencias de áreas y α es un valor de ponderación en el rango $[0, 1]$ que indica qué influencia tendrá D y A en el arreglo de costos. Actualmente estoy trabajando con valores de α de entre 0.6 y 0.8, otorgando mayor peso al arreglo de distancias que al arreglo de diferencia de áreas. Esta decisión se tomó considerando el hecho de que una comparación por áreas es más propensa a errores y su uso se implementa como un elemento que puede ayudar a desambiguar situaciones donde considerar solo la distancia puede generar un rastreo equívoco.

Para asignar a cada *taggedBlob* uno de los blobs actuales comenzamos asignando el blob que tiene el valor mínimo en el arreglo de costos. Por ejemplo, en el siguiente arreglo el valor mínimo sería 0,22 que se encuentra en la posición $[1, 1]$.

$$\mathbf{C} = \begin{pmatrix} 0{,}22 & 0{,}53 & 0{,}87 \\ 0{,}31 & 0{,}57 & 0{,}65 \\ 0{,}63 & 1{,}0 & 0{,}48 \end{pmatrix}$$

En este caso se asignaría al *taggedBlob* en la primera posición del vector *sortedBlobs* el blob en la primera posición del vector de blobs del $contourFinder$. Como solo podemos asignar a cada *taggedBlob* un blob del $contourFinder$, para evitar repeticiones saturamos los valores correspondientes al blob ya asignado, resultando en un arreglo de la siguiente forma.

$$\mathbf{C} = \begin{pmatrix} 2{,}0 & 2{,}0 & 2{,}0 \\ 2{,}0 & 0{,}57 & 0{,}65 \\ 2{,}0 & 1{,}0 & 0{,}48 \end{pmatrix}$$

En la asignación al *taggedBlob* se guarda el blob previo en la variable *prevBlob* y el anterior a la variable *pprevBlob*, de esta manera se tiene un historial de dos momentos anteriores. Si el blob es nuevo, se crea un objeto *taggedBlob* para contenerlo, se checan las etiquetas que existen para elegir uno que no esté ocupado y se añade a la cola del vector *sortedBlobs*.

Una vez asignados todos los blobs detectados en la imagen actual se checa si hay blobs en el vector *sortedBlobs* a los que no se les haya asignado un nuevo tag. Si es así, esto significa que han desaparecido y por lo tanto hay que borrarlos y liberar la etiqueta que lo enumeraba. Finalmente se actualizan los valores de velocidad y aceleración de cada *taggedBlob*.

En caso de que no haya blobs en la imagen actual, se borran todos los blob y se espera que aparezcan.

Detección de dedos Una vez aislados y rastreados los blobs de las manos se le aplica un algoritmo de curvas prominentes al contorno de los blobs con el objetivo de reconocer puntos donde es plausible que haya la presencia de dedos. Los puntos detectados son almacenados en una estructura de datos de tipo *struct* llamada $finger$ que sirve para representar un dedo con sus descriptores:

- Número de etiqueta asignado por el rastreador: tag.
- Posición a tiempo t: fingerPos(x, y).
- Posición a tiempo $t - 1$: prevFingerPos(x, y).
- Posición a tiempo $t - 2$: pprevFingerPos(x, y).
- Velocidad (x, y) del dedo: velX, velY.
- Aceleración (x, y) del dedo: accX, accY.

El algoritmo utilizado para la detección de los dedos puede consultarse en [48] y consiste en ajustar un triángulo con un ángulo α mínimo dentro del contorno del blob de la mano como se muestra en la figura 9. El algoritmo consta de dos partes, la primera donde a cada punto **p** en la curva del perímetro del blob se le

intenta ajustar un triángulo definido por los puntos $(\mathbf{p}^-, \mathbf{p}, \mathbf{p}^+)$ con las siguientes restricciones:

- $d_{min}^2 \leq |\mathbf{p} - \mathbf{p}^+|^2 \leq d_{max}^2$
- $d_{min}^2 \leq |\mathbf{p} - \mathbf{p}^-|^2 \leq d_{max}^2$
- $\alpha \leq \alpha_{max}$

Donde $|\mathbf{p} - \mathbf{p}^+| = |\mathbf{a}| = \mathbf{a}$ denota la distancia entre $\mathbf{p}$ y $\mathbf{p}^+$, $|\mathbf{p} - \mathbf{p}^-| = |\mathbf{b}| = \mathbf{b}$ la distancia entre $\mathbf{p}$ y $\mathbf{p}^-$ y $\alpha \in [-\pi, \pi]$ la apertura del ángulo del triángulo a ajustar. Se calcula α de acuerdo a la siguiente formula:

$\alpha = arccos(\frac{a^2+b^2-c^2}{2ab})$

Todos los puntos $\mathbf{p}$ que cumplan con las restricciones son considerados como admisibles, los que no, son desechados y no serán considerados en el siguiente paso del algoritmo. Para el caso específico de esta implementación también se desechan los puntos cuya curvatura sea convexa ya que se buscad dedos y de estos se sabe que su curvatura es cóncava. Un punto se encuentra en una curva cóncava si:

$b_x c_y - b_y c_x \leq 0$

En la segunda parte del algoritmo se suprimen múltiples puntos que indiquen una misma curvatura y se selecciona el que cuente con un α más pequeño, esto se logra comparando puntos vecinos que se encuentren a una distancia menor que d_{max}, como se puede apreciar en la figura 9.

El buen funcionamiento del algoritmo depende de la cuidadosa selección de los parámetros d_{min}, d_{max} y α_{max}, por lo cual en la implementación se permite el ajuste de estas variables mediante el teclado para su calibración según las condiciones en que se echará a andar la aplicación.

Se puede apreciar en la figura 10 que el resultado de la detección de curvas prominentes es bastante acertado para detectar dedos, lo cual pone a nuestra disposición una interesante gama de posibilidades para jugar con elementos gestuales del movimiento de la mano.

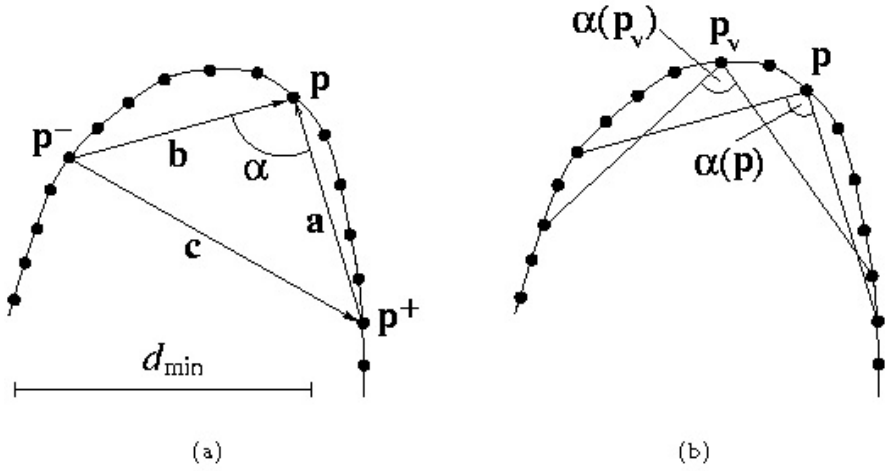

Figura 9: Detección de curvas prominentes. (a) Determina si **p** es un punto candidato. (b) Prueba la prominencia de **p** para suprimir repeticiones cercanas que no son máximas [48]

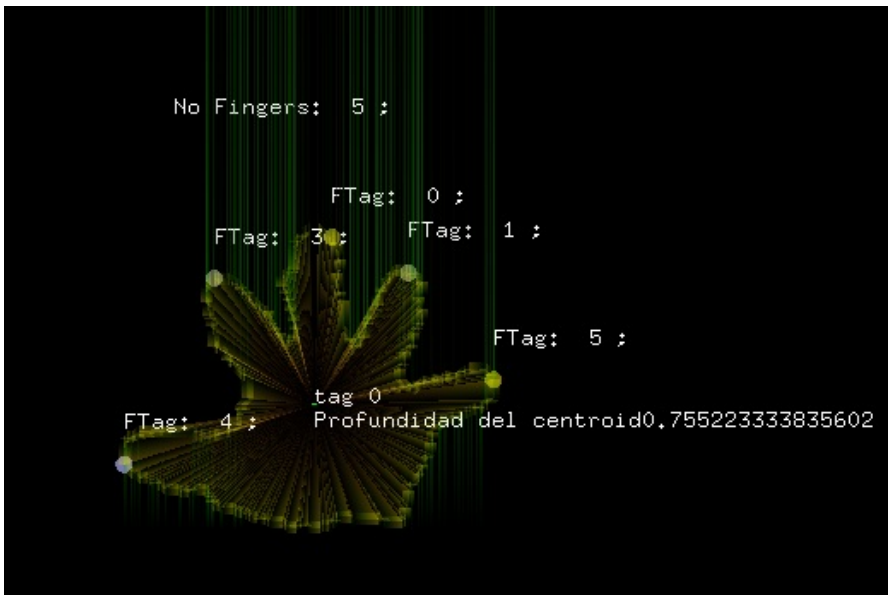

Figura 10: Resultado de detección de curvas prominentes.

El algoritmo de rastreo aplicado para hacer un seguimiento de los dedos es el mismo que se aplicó para rastrear blobs, con la pequeña diferencia de que el arreglo de costos se construyó únicamente tomando en cuenta la posición (x, y).

5.1.3. Despliegue visual y síntesis de gráficos

Este subsistema se encarga de llevar a cabo los cálculos numéricos necesarios para la generación de contenidos gráficos e imprimirlos a la pantalla.

El objetivo principal de visualizar la información que esta siendo capturada y analizada por el programa es proveer de retroalimentación al ejecutante. Esta información es un apoyo cognitivo importante para el control del instrumento, pues cierra un ciclo de comunicación donde el interprete puede acceder a lo que la computadora está viendo y relacionar directamente el resultado de sus movimientos con lo que "interpreta" de ellos. La retroalimentación en este tipo de sistemas es importante para el aprendizaje del instrumento, ver explícitamente el modelo de control que implementa transparenta la interactividad y revela el estado de los parámetros que mide. Al ser explícita la información se representa a si misma como lo que es: movimiento.

En este sentido, los gráficos que se presentan como interfaz se apegan a los principios de lo que Jef Raskin[49] llamara "Cognetics:" una disciplina práctica que trabaja con hechos empíricos de la cognición con el propósito de crear un modelo humano para el diseño de avances tecnológicos, en vez de crear modelos metafóricos de mecanismos cognitivos. En este trabajo el despliegue visual se implementó en OpenFrameworks, de manera que el acceso a la información capturada y analizada es inmediato y no requiere transmisión entre aplicaciones.

Utiliza principalmente la información que describe el contorno del objeto detectado, la posición de su centroide, su velocidad y aceleración.

La información del contorno la se despliega de diversas maneras, como lineas tangentes que tocan la silueta en un solo punto, lineas verticales y horizontales que emergen de cada punto en el contorno hacia los bordes de la pantalla y lineas que nacen del centroide del blob hacia cada punto del contorno. Este ultimo set de lineas sirven también para rellenar el blob y representar el centroide como punto de convergencia.

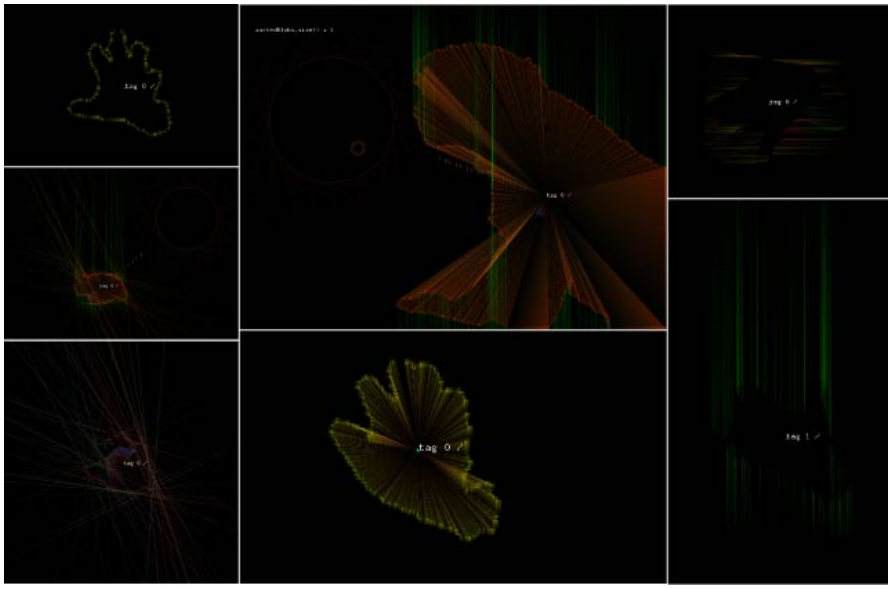

Figura 11: Muestra de los visuales generados para la retroalimentación.

En la figura 11 se muestra una variedad del tipo de visualizaciones que se proponen.

El centroide se representa con una figura que se conforma de un número de triángulos sólidos que va cambiando rápidamente con el tiempo.

Los gráficos se generaron con la biblioteca de OpenGL, ya que permite un control fino del color, de tal manera que se pueden dibujar lineas cuyo color cambia en un barrido desde su punto de inicio a su punto final. También el color de las figuras sólidas puede ser controlado definiendo un color para cada vértice de la figura, el resultado es una figura coloreada con una gradiente de color entre los vértices.

La velocidad y aceleración se representan como lineas que tienen la magnitud y dirección de sus vectores, ofreciendo una visualización en magnitudes de numero de pixeles por tiempo de ejecución.

La figura 12 muestra como se visualizan los vectores de velocidad y aceleración. El vector de velocidad se representa en azul y el de aceleración en verde.

Finalmente, debe de ser mencionado que al hacerse todos los análisis en imágenes de $640 * 480$ las magnitudes deben de ser escaladas al tamaño de la pantalla para visualizarse correctamente.

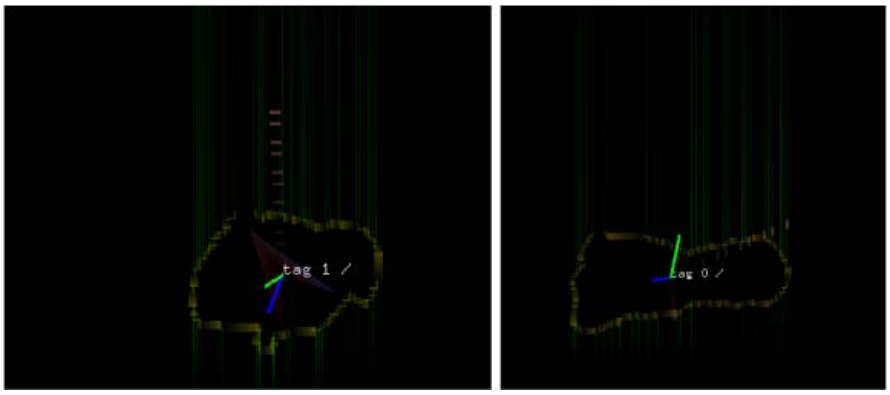

Figura 12: Vectores que muestran la velocidad y la aceleración. El vector de velocidad se representa en azul y el de aceleración en verde

5.2. Subsistema 2: comunicación OSC

La comunicación OSC entre programas se lleva acabo a través de la conexión interna del localhost. El localhost es el nombre que se le asigna a la computadora local para comunicarse a través de la interfaz de red en retroalimentación consigo misma. Esta conexión es muy útil para entablar comunicación entre programas mediante protocolos de internet, como TCP/IP y UDP. Además, permite acceso a los puertos de la máquina, lo que implica que puede haber múltiples conexiones en diversas direcciones operando simultáneamente.

El *localhost* tiene asignada por definición la dirección 127.0.0.1 y para conectarse con los diferentes programas requiere que se le asigne un puerto. El servidor de SuperCollider opera por definición en el puerto 57110 y el cliente, con el cual nos comunicamos opera en el puerto 57120.

El primer paso para entablar una comunicación entre programas, ya sea dentro de una misma computadora o a través del internet es definir la dirección IP hacia la cual se va a transmitir y el puerto por el cual el programa receptor espera la información.

Como en cualquier protocolo es necesario que el programa receptor y el programa emisor conozcan de antemano el formato en que la información será enviada. En el caso de OSC la gran ventaja del protocolo es que se define según las necesidades de los datos a transmitir. En el contexto del modelo de comunicación standard OSI (ver apéndice

Cuadro 2: Formato del mensaje OSC diseñado para enviar información del movimiento a otros programas. El primer mensaje esta diseñado para transferir la información de los blobs rastreados, mientras que el segundo se encarga de la información sobre los dedos de cada blob.

/blobs	número de blobs	tag	posición X	posición Y	velocidad X	velocidad Y	aceleración X	aceleración Y	...

/fingersBlob	número de fingers	posición X	posición Y	...

M), a OSC le corresponde la capa de presentación, o capa 6[50], lo que implica que su formato depende del diseño que el programa implemente. La gran ventaja de OSC es que el formato se define por el usuario de acuerdo a las necesidades de comunicación especificas de cada aplicación.

En este caso el diseño implementado es el que se presenta en el cuadro 2, como podemos ver es un mensaje cuyo tamaño varía según la cantidad de blobs. El primer valor del mensaje es la dirección que sirve para comunicar, a partir de un consenso, el tipo de paquete que se envía. El segundo indica cuantos blobs van a ser enviados y los siguientes 7 valores son los parámetros que describen a cada blob. Con excepción del número de blobs que es un número entero, todos los demás valores se envían como números de punto flotante.

Para el caso de los dedos, la dirección es $/fingersBlob$, el segundo número indica la cantidad de dedos que serán enviados y los siguientes valores, la posición (x, y) de cada dedo.

Es importante recalcar que el formato de la información requiere de un consenso entre emisor y receptor, de esta manera el receptor puede descodificar el mensaje en un proceso llamado *parsing*, extrayendo los valores de cada uno de los datos y asignandolos a sus variables correspondientes.

5.3. Subsistema 3: audio

La sección encargada de la interpretación sonora del movimiento está programada en SuperCollider y recibe los datos del programa de análisis y despliegue visual. Esta

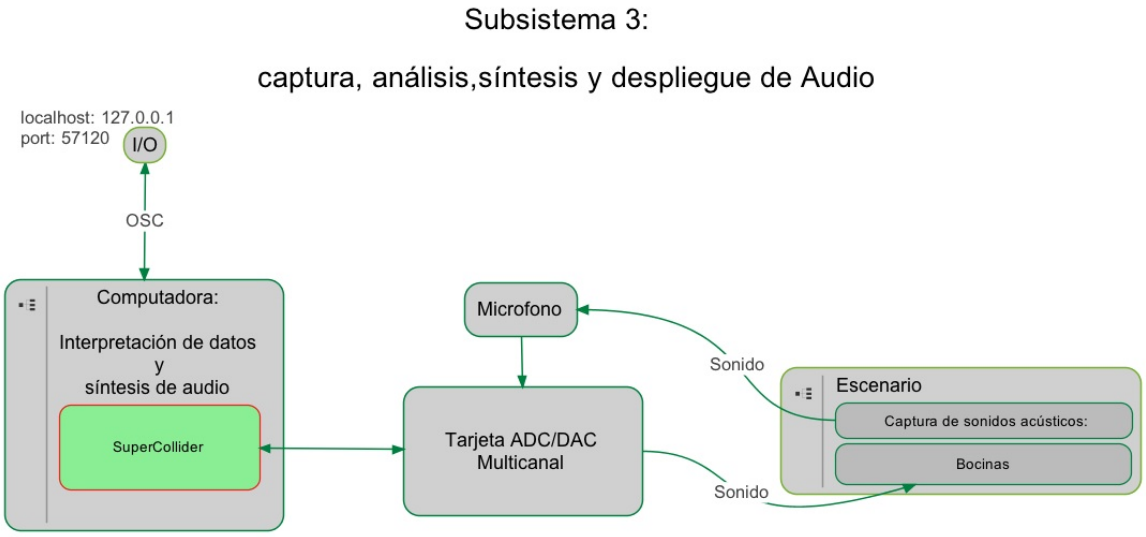

Figura 13: Conexión y comunicación con el sistema de audio.

información se transmite mediante mensajes OSC a través del *localhost*. En la figura 13 se muestran las conexiones de comunicación del subsistema. El servidor de SuperCollider responde en el puerto 57110, sin embargo, para recibir los datos desde OpenFrameworks no se comunica directamente con el servidor, sino que envía los datos al cliente o *sclang* que responde en el puerto 57120. Esto se debe a que los datos deben ser desempacados de su protocolo y después interpretados antes de ser sintetizados como sonido.

Con este propósito creé una clase llamada *"ReceiveBlobs"* que se encarga de responder a mensajes OSC con un patrón de dirección específica, en este caso */blobs* y */fingersBlob*.

El objetivo de la clase *ReceiveBlobs* es desempaquetar los mensajes OSC mediante dos objetos de tipo OSCFunc(): oscR y oscF. Estos se encargan de mandar llamar las funciones *parse*{}, si el mensaje llega al la dirección con patrón */blobs*, o *parseFingers*{} si llega a */fingersBlob*.

Dentro de las funciones *parse* se van asignando los valores recibidos en el mensaje a sus respectivos objetos y variables. Por ejemplo, *parse*{} crea, actualiza o destruye objetos de la clase *Blob* mientras que *parseFingers*{} llena una lista 2D conteniendo la posición de los posibles dedos para cada *Blob*, ver figura 14 para una representación

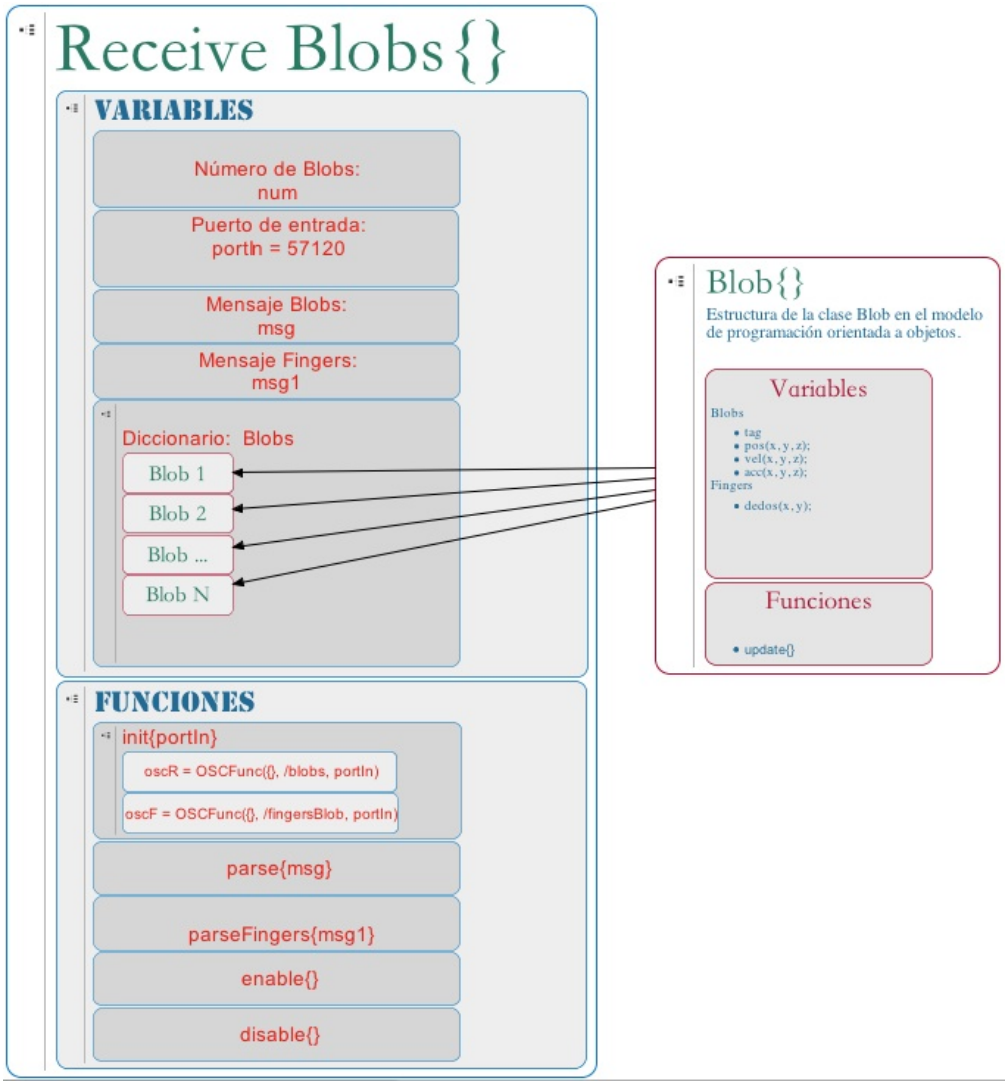

Figura 14: Definición de los objetos de tipo Blob y ReceiveBlobs.

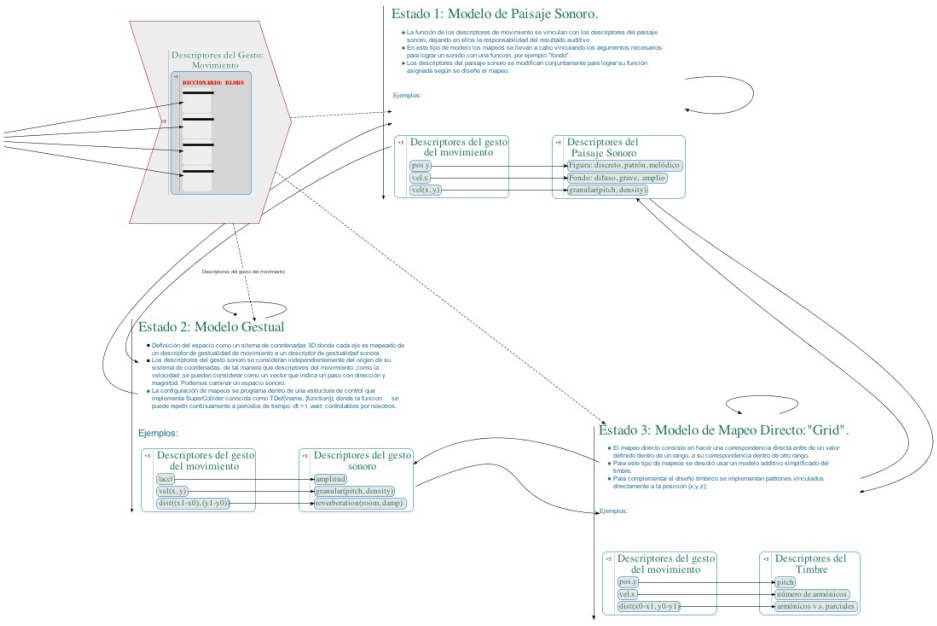

Figura 15: Diseño de la máquina de estados del algoritmo sonoro

gráfica de las clases implementadas con sus principales variables y funciones.

La clase *Blob* describe cada blob con los mismos parámetros que se obtienen con visión computarizada, incluyendo la etiqueta que lo representa. La implementación de esta última clase nos ofrece la gran ventaja de que podemos asignar diferentes respuestas para cada etiqueta, como un sintetizador o el control de alguna tarea de composición.

ReceiveBlobs{} agrega los Blobs recibidos a una estructura de datos tipo *Dictionary* que los aloja en una alberca y provee una llave para acceder a cada uno. La llave asignada a los blobs es su etiqueta de rastreo.

Una vez resuelta la comunicación y representación de datos se tiene una instancia de la clase *ReceiveBlobs* con los descriptores del movimiento accesibles y listos para ser usados en la producción de sonido.

5.3.1. Diseño del algoritmo sonoro

El proceso para la producción sonora se define a partir de una máquina de estados finitos, su diseño se puede apreciar en la figura 15

En mi diseño cada estado definido tiene acceso a los descriptores del gesto de mo-

vimiento analizados, estos conforman el vocabulario compartido mediante el cual se definen los eventos que disparen las transiciones de un estado a otro. En este proyecto decidí que el comportamiento y los mapeos que se llevarán a cabo en cada estado reflejarán en su diseño alguno de los modelos de objeto sonoro que escogí, estos son:

- Paisaje Sonoro
- Gestualidad
- Modelo aditivo del timbre

En el tercer estado el modelo aditivo del timbre se toma como principio para también explorar la más sencilla forma de mapeo, el mapeo lineal. En la gráfica se menciona como mapeo directo, pues en este tipo de mapeo los valores son tomados tal cual y transportados a una escala de rangos diferentes sin hacer mayor interpretación de la información que conllevan.

Consideré que el vocabulario que se desenvuelve con estos tres modelos debería de ser suficiente para desarrollar una pieza electrónica interactiva.

Mapeo El mapeo es un proceso mediante el cual una variable x de un sistema dinámico definido por una serie de variables que conforman un espacio A de N dimensiones, es trasladada o "*mapeada*" mediante una función de transferencia h a una variable y del espacio B definido por otro set de variables y que se relaciona con x por h. Un proceso similar al que se usa para describir los filtros digitales, donde la señal de entrada y salida se relacionan por una función de transferencia como se muestra en el texto de F. Richard Moore [51]. Otro ejemplo de función de transferencia se puede apreciar en la técnica de síntesis llamada *waveshaping* donde el valor de salida de una función en determinado momento depende de una función de transferencia, ver pp. 50 de [37].

En el mundo de la música es común que se tengan que hacer mapeos entre variables bien definidas, como en el caso de este proyecto es *posición(x, y)*, a variables con alto grado de ambigüedad, como lo puede ser el concepto de fondo, que solo se entiende por su función en un contexto sonoro, pero que no es medible objetivamente.

> "Por lo general los músicos no están interesados en medidas o análisis objetivos. Lo que les importa es la función del timbre en relación a la composición, e incluso más todavía, la afectividad creada por la percepción del timbre en el contexto de la obra" [52].

Por lo tanto algunos mapeos implementados en este trabajo no pueden hacerse directamente de una variable a otra, sino que se ajustan a un grupo de variables que en conjunto provocan un efecto, o afecto deseado. Este tipo de situaciones corroboran lo difícil que puede llegar ser el medir la eficacia de un sistema de mapeo, pues esto solo se materializa en el oído del receptor y su forma de juzgar la interacción con el sistema. Sin embargo, basando los mapeos en modelos del timbre y del paisaje sonoro es posible relacionar la intencionalidad de ciertos tipos de gestos con la función sonora de los elementos que conforman estos modelos.

A continuación hago una presentación de cada estado y las características de su comportamiento.

Estado 1: modelo de paisaje sonoro Para este modelo me apoyo en el texto de Murray Shafer [53] que describe muy bien la función de *figura y fondo* en el contexto del paisaje sonoro. Cabe destacar que el mismo Murray Shafer sostiene que la percepción de un sonido como figura o fondo depende claramente de elementos culturales, añadiendo al discurso de la inconmensurabilidad contra la cual nos enfrentamos cuando se trabaja el mapeo como una parte medular de una obra interactiva.

En este caso consideramos la breve definición que Shafer hace:

> "...la figura corresponde a la señal o marca sonora, el fondo a los sonidos ambientales al rededor- que seguido pueden ser notas sonoras- y el campo al lugar donde todo ocurre, el paisaje sonoro" [53].

En la figura 16 se ve una descripción del "$Estado1$," el mapeo en este estado será de variables del movimiento a parámetros del Paisaje Sonoro, la descripción que hace Murray Schafer de Figura y Fondo son aplicadas para lograr que ciertos sonidos tomen

una función, sean dinámicos y respondan a los valores de entrada del sistema, principalmente, el movimiento de las manos.

Se muestra en la figura algunos de los mapeos implementados, donde la posición se mapea al fondo y la velocidad a la figura. Este tipo de descripciones de mapeo pueden significar una variedad de cosas durante el proceso de desarrollo de instrumentos, donde las diferentes variables van cambiando sus valores de acuerdo a la necesidad final y no como un proceso directo donde se ajustan los rangos de una a otra.

En el caso particular de la obra que presento como ejemplo de aplicación de la herramienta aquí desarrollada, la posición X se utiliza para cambiar la posición en un buffer desde donde un sintetizador granula el sonido. La posición Y afecta la velocidad con que se lee el buffer, afectando la altura percibida del sonido. La magnitud de la velocidad se utiliza para controlar la amplitud del sonido, relacionando directamente la cantidad de movimiento con el volumen. La presencia de dedos en la imagen dispara eventos que se despliegan en patrones rítmicos que aumentan en densidad mientras más dedos haya.

Otro tipo de información que tomamos en cuenta para el control del sonido es la distancia entre las dos manos. Mientras mayor sea la distancia en x entre las manos mayor es la mezcla de reverberación en el sonido y mientras mayor sea en el eje de y, mayor será el tiempo de reverberación.

Estado 2: modelo gestual Este modelo representa uno de los objetivos principales de la tesis, la transferencia de información entre el espacio gestual del cuerpo y el espacio gestual sonoro. Propongo un modelo en el que los parámetros del gesto del cuerpo se relacionen con parámetros del gesto sonoro que puedan tener significaciones similares, por ejemplo puede ser la magnitud de la velocidad relacionada con la amplitud o densidad de un gesto sonoro. Para poder definir una relación entre ambos espacios se considera al gesto sonoro como la parte audible de los eventos gestuales del cuerpo. Cuando se analiza, por ejemplo el sonido del gis en fricción con el pizarrón cuando se dibuja una linea, si se considera esta acción en el ámbito del cuerpo podemos medir su velocidad, dirección, aceleración, posición, si tiene dedos, cuantos dedos tiene y sus

posiciones.

Estos parámetros que se logran medir tienen a su vez una repercusión en la parte sonora del gesto analizado, como por ejemplo, se nota que mientras más rápido sea el movimiento, más agudo es el sonido y su amplitud aumenta. Añado otro ejemplo para mostrar las diversas posibilidades que este tipo de acercamiento ofrecen. Si se considera la acción de revolver un vaso con agua para disolver azúcar, cada vez que la cuchara golpea el cristal se emite un evento sonoro, mientras más rápido se agite, más frecuente será el sonido. En el análisis visual se puede medir cada que la dirección de movimiento en un eje cambia repentinamente de positivo a negativo. A este tipo de eventos gestuales del cuerpo se le puede relacionar con un evento gestual sonoro.

Este tipo de eventos discretos son perfectos para relacionarlos con métodos de síntesis granular, donde el grano puede ser considerado como un evento discreto. Así se puede relacionar la frecuencia del cambio de dirección de los eventos gestuales con la densidad con que se generan los granos de un sintetizador, por ejemplo.

Ademas la síntesis granular tiene la ventaja de modelar eventos sonoros como la resultante de la suma de muchos micro eventos, que en conjunto evocan un significado diferente que cuando son escuchados por separado, así como una gota en el agua cayendo

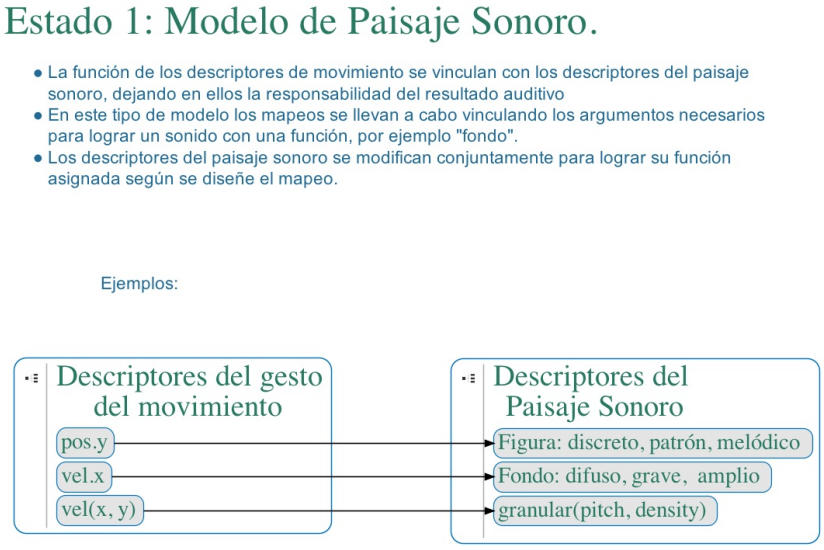

Figura 16: Estado 1

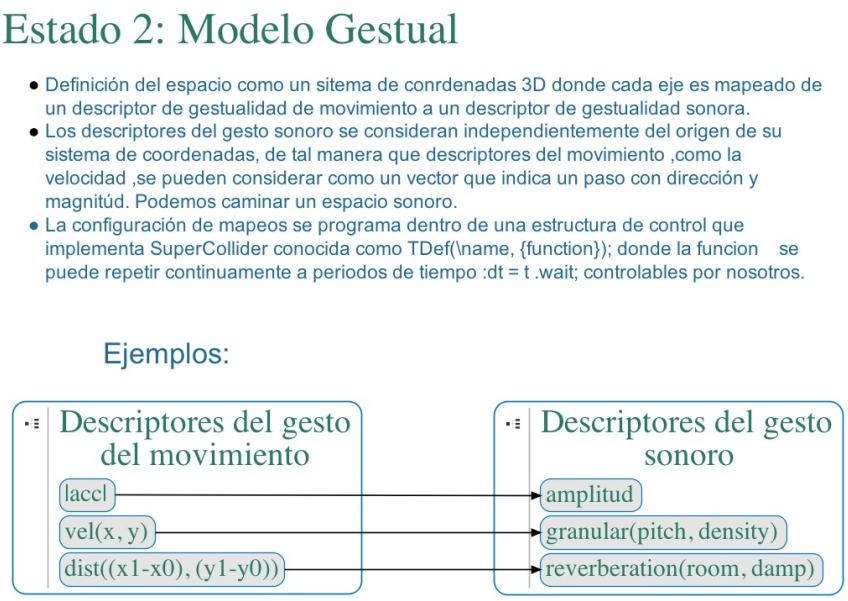

Figura 17: Estado 2.

periódicamente puede evocar una gotera en la bañera y en contraste muchas gotas cayendo densamente sin orden específico pueden evocar lluvia. En [54], Barry Truax y Damian Keller, resaltan la importancia de la estructura temporal de la síntesis granular para evocar objetos sonoros con alto contenido de información y basan sus premisas en un análisis orientado hacia la ecología acústica. En este sentido se propone una cartografía que dote de relevancia ecológica a la estructura temporal de los clusters de granos que se generen en el tiempo.

Para sustentar mejor las decisiones de control tomamos en cuenta algunos aspectos de la espectromorfología de Denis Smalley, quien en [9] relaciona directamente el gesto sonoro con el movimiento:

> "Un agente humano produce espectromorfologías mediante el movimiento del gesto, utilizando el sentido del tacto o un implemento para aplicar energía a un cuerpo sonoro. Un gesto es, por lo tanto, una trayectoria de energía-movimiento que excita un cuerpo sonoro, creando vida espectromorfológica. Desde el punto de vista de ambos el agente y el escucha que mira, el proceso musical del gesto es táctil, y visual así como aural" [9, pg. 111]

Smalley contrasta la función del gesto con la textura, habla de trayectorias como un elemento que define el gesto y le da un contexto para su descodificación de significado, pues argumenta que el gesto no solo nos remite a un evento sonoro, sino que también nos coloca en una experiencia psicológica en general.

> "No solo escuchamos la música, también descodificamos la actividad humana detrás de las espectromorfologías a través de las cuales automáticamente obtenemos abundante información psico-física" [9, pg. 111]

El tipo de sintetizadores creados con esta estrategia consideran las diferencias de energía contenida en el gesto como diferencias de densidades de granos, asumiendo que a mayor energía mayor producción de micro eventos sonoros.

Queda claro con estos ejemplos como la metáfora sirve de enlace para relacionar la gestualidad del cuerpo con la gestualidad sonora. Más aún esta es una cualidad que se comparte con la gran mayoría de las *interfaces de usuario naturales* o NUI's[6] por sus siglas en Ingles.

La gran particularidad de las NUI's es que al promover una interacción similar a la que se usa para interactuar con el mundo real, permiten un control intuitivo a través de la computadora del instrumento sonoro que se diseña.

Este tipo de control abre las posibilidades de relacionar una infinita cantidad de métodos de síntesis sonora con el movimiento, manteniendo una relación intuitiva.

Manuel Rocha en[55] y Curtis Roads en[56][7] exponen algunas de las funciones y metáforas que se pueden manejar con la síntesis granular. En la tabla 3, vemos una serie de Oposiciones estéticas propuestas por Curtis Roads que sirven de referencia para implementar mapeos.

En el caso particular del estado 2, uso la metáfora de caminar el espacio como una forma de liberar a los parámetros sonoros del origen del sistema de coordenadas de la cámara. Se define la velocidad como un vector que expresa la dirección y magnitud de un desplazamiento. Es decir los parámetros sonoros definen su sistema de coordenadas con

[6] Natural User Interface

[7] el capitulo de Aesthetics of Composing with Microsound, especialmente la sección de Aesthetics Opositions

Cuadro 3: [Oposiciones Estéticas en la Síntesis Granular. Curtis Roads[56].]

Formalism v.s. Intuituinism
Coherence v.s. Invention
Spontaneity v.s. Reflection
Intervals v.s. Morphlogies
Smoothness v.s. Roughness
Attraction v.s. Repulsion in the Time-Domain
Parameter Variation v.s. Strategy Variation
Smoothness v.s. Roughness
Simplicity v.s. Complexity in MicroSound Shynthesis
Code v.s. Gramar
Sansation v.s. Comunitation

el origen en su valor actual y al ser actualizados avanzan según el vector de velocidad. Esto permite que los centroides controlen el movimiento de los parámetros de síntesis despegados del plano cartesiano que captura la cámara y que al desaparecer y aparecer en posiciones diferentes su valor se mantiene, pues depende de la velocidad y no de la posición.

Para los sonidos con tono, las alturas son seleccionadas con información local del movimiento, es decir, no hay una retícula fija que indique una nota o frecuencia según la posición, por el contrario se definen reglas de crecimiento o reducción de valores según la dirección, magnitud y velocidad del desplazamiento. Esto nos lleva a instrumentos cuyo estado cambia continuamente y depende de los estados anteriores que lo llevaron ahí.

Implementé los sintetizadores granulares con una envolvente aplicada a la amplitud y otra a la frecuencia, alterando el desarrollo de cada grano con el mismo patrón, pero sobre alturas diferentes. Esto permite diseñar el desarrollo de cada grano para evocar micro-eventos, como una gota de agua o el impacto de un trozo de vidrio en el concreto. También es posible diseñar micro-eventos abstractos que cuando se reproducen en conjunto cambian completamente las propiedades tímbricas del sonido. Otro acercamiento a este mismo tipo de sintetizadores es trabajar un grano como la grabación de un micro-evento que se reproduce cambiando las propiedades de la grabación para

ajustarlas a la duración y altura requerida para diversificar el evento. En este caso el modelo del micro evento no es abstracto sino concreto.

Finalmente, este tipo de sintetizador es activado de manera concurrente según la velocidad del movimiento, así mientras mayor sea el movimiento más sintetizadores se activarán en un menor lapso de tiempo, aumentando la densidad sonora.

Para poder situar este tipo de sonidos en un espacio resonante es importante no aplicar una reverberación a cada grano, sino que se supone que los granos excitan el espacio y por lo tanto se dirige el sonido de todos los granos a un mismo bus de audio al cual se le aplica una reverberación general. Esto no solo es consistente con el modelo ecológico del sonido, sino que también hace eficaz su aplicación.

Estado 3: mapeo directo y modelo aditivo del timbre En contraste con el diseño del estado anterior, el comportamiento de los sintetizadores se definió con respecto al sistema de coordenadas de la cámara, manteniendo la posición como un set de coordenadas con respecto un origen fijo. Siguiendo este sistema, se aplicó un mapeo directo de los descriptores del movimiento a parámetros de síntesis aditiva. Este tipo de mapeo define claramente el tipo de control dentro de un rango y mantiene los puntos de referencia claros y en coherencia con el espacio. Esto es ideal para desarrollar trayectorias cuya repetición resulte en eventos sonoros similares. Un ejemplo de mapeo de este tipo se implementó entre la coordenada y de la posición de un centroide, con la altura percibida. Si el blob se mueve hacia arriba la altura incrementa y si se mueve hacia abajo disminuye. Estos cambios se pueden implementar como un barrido continuo en la dimensión de las alturas o a través de un patrón discreto incremental preestablecido, mismo que se repite en reversa si el movimiento es contrario.

Otra estrategia es definir patrones melódicos que se desdoblan con la posición de un centroide. Este acercamiento se implementó con la coordenada x de la posición. Los patrones melódicos se aplican con una retícula 2D de valores que son leídos por la posición x y de acuerdo al valor ahí representado se actualizan los valores del sintetizador. Otro acercamiento que exploré es la implementación de reglas de aumento, donde el tono o nota musical se actualiza de acuerdo a una regla con un cierto grado

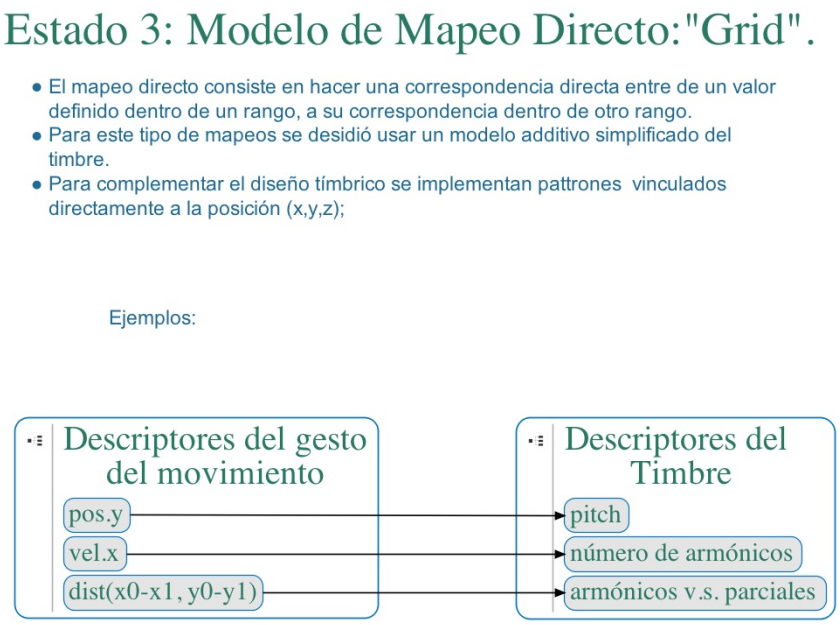

Figura 18: Estado 3

de azar. Por ejemplo, si se considera un tono con pitch class 0, se puede decidir que con cada incremento en x se incremente el pitch class por un valor escogido al azar entre los siguientes tres $[7, 5, 3]$. En total hay 12 pitch class, lo que implica que es suficiente con mapear 12 diferentes posiciones para tener 12 reglas diferentes. Esto promueve un desarrollo dinámico y continuo.

Otro parámetro sonoro que exploré es la relación armónica entre la frecuencia de las componentes con los que se construye la síntesis, esta relación se puede favorecer o no, cambiando claramente el carácter sonoro del timbre. Este parámetro se relacionó con la magnitud de la velocidad del centroide, con lo que se mantiene una relación con la gestualidad del movimiento. A mayor velocidad menor relación armónica entre componentes.

5.3.2. Integración del algoritmo sonoro

Para llevar a cabo los procesos necesarios para el diseño de cada estado fue necesario implementar una serie de estructuras tipo *Tdef*.

Cuadro 4: Diseño de la máquina de estados finitos.

Estado 2	Estado 3	Estado 1	Estado 2
Estado 1	Estado 2	Estado 3	Estado 1
Estado 3	Estado 1	Estado 2	Estado 3
Estado 2	Estado 3	Estado 1	Estado 2

Tdefs Este tipo de estructuras definen hilos de ejecución con un desarrollo temporal interno propio controlado por el método *int.wait*, de esta manera se puede utilizar *Tdefs* a velocidades de ejecución diferentes. Uno de los *Tdef* mas importantes del proyecto, la máquina de estados, mantiene actualizados los descriptores del movimiento, ejecutandose cada 0.05 segundos. Otros se ejecutan con un tiempo declarado en un patrón de duraciones, muy útiles para implementar ritmos.

Transiciones La máquina de estados controla el comportamiento del algoritmo. Ésta activa e implementa cada estado, actualiza los parámetros de síntesis y lleva a cabo las transiciones entre estados cuando una de las reglas de transición se cumple. Las reglas de transición fueron definidas con respecto a la velocidad y las condiciones de frontera a las que nos sujeta el sistema de coordenadas de la cámara, de tal manera que el contacto de un blob con una frontera a una velocidad con un componente ortogonal mayor a *transitonThreshold* dispara la transición al siguiente estado. De esta forma se definen cuatro reglas de transición, una para cada frontera, con un vocabulario definido por el evento de colisión y la velocidad.

Topología de la máquina de estados finitos El diseño implementado en este proyecto conforma un anillo de transiciones en ambas direcciones, de tal manera que comenzando en Estado 1 una transición hacia la derecha lleva al Estado 2 y una transición a la izquierda a Estado 3.

En la tabla 4 se muestran las transiciones en un arreglo en el cual el elemento en cada casilla puede tener una transición a uno de sus vecinos de arriba, abajo y laterales, pero no los diagonales.

Esta configuración permite que al interpretarse el instrumento exista siempre la

posibilidad de moverse a uno de los otros dos estados.

Funciones Para la definición programática de los estados hubo que auxiliarse de métodos o funciones. Éstas se utilizaron para efectos como generar listas, generar patrones rítmicos, retículas de valores y control de sintetizadores.

Los sintetizadores Los sintetizadores se definen en un documento del lado del cliente de SuperCollider, estos proveen argumentos para cambiar su estado interno y pueden ser alterados desde cualquier otro proceso o Tdef que esté en marcha. Una vez definidos se envían al motor de audio de SuperCollider o *scsynth*, desde donde son instanciados y agendados para su ejecución en el árbol de nodos.

6. Aspectos de Diseño

A continuación presento algunos temas que fueron considerados para el diseño de la herramienta. La gran mayoría se relacionan con la preocupación de diseñar un instrumento que permita la mayor expresividad posible.

6.1. Foco de atención

En su libro *"The Humane Interface"*[49] Raskin propone el termino *"Locus of attention"*, que aquí traducimos como *"Foco de atención"*, para definir el elemento en el cual se enfoca nuestra atención en un determinado momento. Según Rasking, es comúnmente aceptado que tenemos un solo foco de atención, sin embargo dentro del ámbito de la música el foco de atención puede extenderse a varios niveles. El ejemplo más obvio es el de un pianista que debe poner atención a lo que tocan ambas manos. Sin embargo en la creación de este instrumento he cuidado que los parámetros de control del sonido no sean demasiados, pues esto podría resultar en un instrumento incontrolable y el propósito de transmitir una intención se perdería.

En este sentido la implementación de una visualización para la retroalimentación sirve no solo para proveerle al intérprete una referencia de lo que sucede, sino que

amplía los elementos disponibles para que el público pueda descifrar lo que ocurre en el escenario.

La idea de hacer del gesto el elemento principal de preocupación en este trabajo esta estrechamente vinculada con la exposición de obras mediante un lenguaje cotidiano que no requiere de muchos elementos nuevos por parte del receptor para interpretar lo que sucede.

6.2. Comprensión de procesos simultáneos

En el diseño y ejecución de este instrumento llegan a ocurrir múltiples procesos que no son obvios para quien observa, sin embargo para que una ejecución sea efectiva es necesario la comprensión de los mecanismos mediante los cuales el instrumento opera. Este conocimiento es importante para que el ejecutante pueda desenvolverse de la manera más efectiva posible, tomando en cuenta las posibilidades del instrumento.

Como ya ha sido mencionado, la visualización del análisis de datos juega un papel muy importante en este punto, pues es a través de ella que se hace obvia la reacción de la computadora a los estímulos del ejecutante. Este recurso también le otorga una herramienta al público para descifrar lo que realmente ocurre en el escenario.

Este punto no solo es importante para una mejor comunicación, sino que le ofrece a los actores la posibilidad de proponer nuevas metáforas de interacción que se añadan al discurso de la obra o la simple posibilidad de resaltar intenciones al momento de improvisar.

Hay procesos que pueden ser muy obvios y otros que se desarrollan en un plano menos visible, como los procesos de composición, que operan a un nivel más abstracto y cuyos mecanismos pueden no siempre ser tan evidentes. Es entonces importante mantener un nivel de abstracción en algunos aspectos que sorprendan al observador, sin necesidad de develar completamente lo que sucede, pero no mantenerlo completamente en la penumbra. Un buen diseño de sintetizadores hace que la correspondencia entre el gesto sonoro y el gesto del movimiento sean fácilmente comprensibles sin volverse aburridos.

Consideré entonces que la transparencia es importante en el caso de los gestos,

que por su naturaleza toman un papel frontal en el diseño sonoro, pero que se puede contrastar y apoyar con aspectos de composición que agreguen texturas y movimientos lentos en la obra cuyos mecanismos de acción no sean obvios pero que si provean de un buen fondo sobre el cual desarrollar figuras sin que se vuelva monótono.

Es en estos procesos de construcción y diseño sonoro que es importante tomar en cuenta trabajos como el de la espectromorfología de Smalley[9] y el soundscape de Murray Schafer[53] para crear obras que sean consistentes con la forma en que escuchamos y que transmitan contenido.

6.3. Latencia en interacción y el problema de *tiemporreal*

Una consideración muy importante durante el diseño y la programación de esta herramienta fue el mantener la eficacia de ejecución de los procesos ya que planteo ejecutar todo dentro de una misma computadora, esto se hace notar en la implementación del algoritmo de rastreo, cuya implementación es sencilla y busca ser eficiente sin perder eficacia.

Una buena eficiencia en la ejecución de la herramienta implica, que el tiempo que transcurre desde que se captura una imagen por el dispositivo $Kinect^{tm}$ hasta que su imagen es desplegada y el sonido afectado por los análisis efectuados, sea el mínimo posible. Esto permite que haya una sensación de inmediatez que mantiene el enlace temporal entre el movimiento y el sonido, manteniendo una coherencia entre los ámbitos en que se desarrollan los gestos como una relación causal.

6.4. Percepción de interacción en audiencias y en ejecutante

Durante el diseño de los sintetizadores y el tipo de imágenes creadas fue importante mantener la transparencia de la interfaz, de tal manera que la interacción fuera fácilmente comprensible por audiencias no especializadas. De tal manera que debe de poderse establecer una causalidad que nos indique que hay una intencionalidad que está siendo transferida a través de los mapeos implementados.

Esta transferencia de información es considerada como un facilitador de la expresión,

promoviendo que haya desarrollo de discursos y argumentos artísticos con el uso de esta herramienta. Así, no solo las audiencias son beneficiadas, sino que se provee de una herramienta intuitiva para que el ejecutante o los ejecutantes puedan desenvolverse a través de la computadora sin necesidad de conocimientos especiales para activarla, ni restricciones físicas al movimiento.

7. Discusión y conclusiones

Al comenzar esta tesis mi objetivo principal era simplemente hacer un programa que lograra hacer que las cosas sonaran con el movimiento, conforme fui aprendiendo sobre métodos para aplicar mapeos y hacer análisis con visión computarizada me di cuenta que el gran problema no era que las cosas sonaran nada más, había que considerar que la expresión del ejecutante se transmitiera del movimiento al sonido con coherencia. Pensar en como trascender la rigidez de los sistemas formalizados e implementar una aplicación que deje permear a través de sus algoritmos las intenciones del artista, fue lo que me llevó a considerar el gesto como el elemento central que debía investigar.

El gesto ofrece como concepto un vínculo entre el sonido y el movimiento que además, se presenta como un elemento importante de la expresión no verbal en ambos ámbitos.

Como lenguajes no verbales, la transmisión de significado del gesto de la mano al sonido no es una variable que pueda medirse objetivamente, sin embargo si es posible percibir de forma subjetiva que en el diseño sonoro hay una relación importante entre el movimiento de la mano y las características tímbricas temporales que éste genera. En este sentido la subrogación gestual que propone Smalley en [9] es un elemento clave para entender el proceso que toma lugar cuando suena el instrumento, ya que nos ofrece un marco teórico para vincular una fuente sonora con el movimiento que la causó. El hecho de que una subrogación gestual tome lugar, apunta hacia nuestra experiencia como generadora de una base de conocimiento a la cual se hace referencia para significar nuevas experiencias, esto se puede relacionar con el esquema de Kant y los procesos del *juicio preceptivo*, que mediante la experiencia cognitiva permite categorizar el mundo en

un esquema al que posteriormente nos referimos para significar nuevas experiencias, en [57, pg. 57-120] Umberto Eco hace una buena presentación de estos conceptos y expone un ejemplo de cómo estos procesos toman lugar con la anécdota de cuando Marco Polo vio por primera vez un rinoceronte en la isla de Java:

> "A pesar de que nunca había visto a esos animales antes, por analogía con otros animales conocidos pudo distinguir el cuerpo, las cuatro patas, y el cuerno. Dado que su cultura le proveyó con la noción de un unicornio - un cuadrúpedo con un cuerno en su frente, para ser preciso- designó aquellos animales como unicornios" [57, pg. 57].

Así entonces la subrogación remota que propone Smalley, ver sección 3.2, y que toma lugar cuando escuchamos sonidos nuevos quizá tenga que ver con un juicio perceptivo impulsado por una necesidad de darle sentido al mundo, aplicando una categorización esquematizada para juzgar nuevas experiencias sonoras. Aquí la imaginación toma un papel importante, pues es a través de ella que aplicamos nuestro esquema categórico para juzgar nuevas experiencias. Me gustaría poner énfasis en que el grado en que los modelos del mundo imaginables se asemejen a la realidad no es una preocupación cuando el objetivo es decir algo y no interpretar algo, aquí utilizo el verbo interpretar de la misma manera en que podría usar juicio perceptivo en el sentido de que ambos connotan un proceso de significación del mundo. Significar el mundo nos transporta inmediatamente al lenguaje, por un lado, y a las ciencias de la cognición, por otro. Para fundamentar el lado del lenguaje se presentó el modelo de interacción de Rafaeli, ver sección 2.3, como una visión desde la comunicación, donde flujos de información actúan a través de interfaces que la regulan. Los interaccionistas simbólicos, ver sección 3.1, se percataron de la importancia de las comunicaciones interindividuales y pusieron énfasis en ellas, de aquí surge el enfoque en el gesto y los lenguajes no verbales que sin duda comunican y generan modelos del mundo sin necesidad de formalizarlos en palabras sujetas al trinomio del signo de Pierce: símbolo, significado y significante. Este tipo de comunicación es esencial para las artes, pues puede re estructurar la interacción de categorías en relaciones diferentes, un proceso muy similar al que en este trabajo me

refiero como mapeo o cartografía.

La relación entre gesto y música se establece en la sección 3.2, y con el concepto de subrogación remota de Smalley se describe una traslación semántica, donde los signos se trasladan de ámbito para significar en otro campo semántico, como cuando el movimiento transmite información a través del sonido.

La representación digital del mundo permite este tipo de relaciones como una operación nativa, convirtiendo al código binario en un lenguaje que privilegia y promueve la intercomunicación multimedia, ver sección 2.2. En la sección 3.2, se presentó la noción de que el gesto tiene un papel primordial en la producción de expresiones sonoras y musicales. Incluso, Mazzola y Andreatta [35], hacen un esfuerzo implacable para sustentar matemáticamente el papel fundamental que el gesto tiene en la producción musical.

Por otro lado, se presentaron algunas referencias que evidencian el papel tan importante de los procesos cognitivos corporales que se desatan con el sonido y lo vinculan con el movimiento, haciendo alusión a que un modelo biológico de las interacciones humanas también es posible y que incluso se podría comenzar a vincular con el interaccionismo simbólico a través de modelos de la comunicación no verbal surgidos de la psicología como el de Frissen y Ekman, ver sección 3.

Con esto sustento la relevancia de esta tesis como una interfaz de interacción natural que mediante el gesto permite una interacción entre el movimiento y el sonido de manera transparente, es decir, sin hacer evidente el papel que juega la computadora en la traslación semántica, simulando la forma en que interactuamos naturalmente con la realidad. Una simulación de subrogación gestual que valiendose de la imaginación relaciona experiencias remotas como el movimiento y la música para expresar algo.

Al volver al modelo de interacción de Rafaeli, se vió que una de las condiciones necesarias de los canales de información para permitir la interacción es que el flujo de datos sea duplex, es decir, que vaya en ambas direcciones de manera asíncrona. Como acabo mencionar, la afectación del movimiento y el sonido es mutua, los datos fluyen para los dos lados, lo que permite que una interacción natural tome lugar. Aquí me gustaría recalcar que la interacción natural la opera el performer como un control sonoro que responde de manera analógica al movimiento, es decir, no está interactuando

Figura 19: Esquema de comunicación.

directamente con la computadora, sino que interactua de manera *"transparente"* con el sonido, a esto se superpone la interacción del performer con el público como ocurre en la más común situación del improvisador. En la figura 19 vemos un modelo que muestra los canales de comunicación activos cuando se improvisa con la herramienta. Se puede pensar en este esquema como una extensión al modelo de interfaz que presenta Jeff Pressing en [26, pg, 411], ver sección 3.2.

La magnitud en que el sonido será gestual depende en gran medida de las estrategias de mapeo, en este sentido es importante recoger el modelo del gesto de Ekman y Frissen presentado en la sección 3.1 y relacionarlo de alguna manera con los modelos del campo sonoro para establecer dinámicas de control que permitan la transferencia de información de un campo semántico a otro.

Se puede notar que las acotaciones hechas en la captura del movimiento permiten una descripción cinemática enfocada al movimiento de la mano y la presencia de dedos, lo cual ofrece un espacio reducido en dimensiones, que definidas como variables deberán controlar atributos del sonido según una de las tres visiones seleccionadas del campo

sonoro: la espectromorfología, el paisaje sonoro y el modelo aditivo del timbre.

Esto ha sido de gran ayuda para proveer coherencia entre la información emitida con movimiento y el sonido causado, por ejemplo, un mapeo lineal basado únicamente en la posición como el que toma lugar en el tercer estado del algoritmo sonoro, no deja que el gesto se transfiera tan fielmente como se logró con las estrategias adoptadas en el segundo estado.

Durante la improvisación con este instrumento se desarrollan dos discursos paralelos, el movimiento y el sonido (ignorando la generación de imagen, pues esta no ha sido abordada con el suficiente detalle como para ofrecer un buen análisis). Con esto las posibilidades de expresión que se abren abarcan una amplia gama que deberá ser explorada y madurada como un idioma más de la totalidad de expresiones artísticas. Con esto me refiero a que a pesar de reducir la representación del gesto en su forma digital, este no deja de desplegarse a plenitud dentro de su ámbito y será sometido a un juicio perceptivo basado en categorías propias del gesto. Recordando a Ekman y Frissen, en su modelo de la comunicación no verbal, eso que se transmite gestualmente se relaciona fuertemente con el establecimiento de estados emotivos, entre otras características.

Por otro lado, el sonido va poblando el medio acústico con una carga importante de esta misma gestualidad que se despliega a la par, en este trabajo me propuse llevar a cabo una coherencia entre los discursos, pues ambos lenguajes comparten la característica de relacionarse con estados emotivos. En [6] David Huron menciona que la música puede causar respuestas biológicas en el cuerpo humano, como la reducción de producción de testosterona, hormona que se asocia con comportamientos de violencia. Biológicamente las hormonas regulan entre otras cosas los estados emotivos, si la música puede transmitir estos estados, entonces se puede crear una coherencia entre los estados emotivos primarios que proponen Ekman y Frissen y las representaciones sonoras diseñadas.

La importancia de tener modelos del objeto sonoro como la espectromorfología de Smalley es que pude formalizarse en categorías o variables del campo semántico del sonido, estados emotivos que nos gustaría representar en conjunto con los que se representan en el campo semántico del movimiento, dando lugar a una verdadera interacción

entre el sonido y el movimiento. De esta manera la expresión artística puede tomar lugar como representación no verbal, trascendiendo la rigidez del lenguaje de programación y transparentando la interfaz, manteniendo un alto grado de naturalidad en su ejecución.

Presenté en esta tesis la implementación de una composición, un espacio sonoro que se puede navegar, y que al navegarse se despliega en el tiempo. Cada vez que se ejecuta el programa, el interprete desdobla un paisaje sonoro posible, uno de una infinita cantidad de paisajes posibles, pues difícilmente las trayectorias gestuales serán idénticas. Por otro lado cada interprete se moverá diferente, generando un discurso gestual diferente. La improvisación llevará a cada interprete a resultados diferentes.

En la composición del espacio sonoro, fue determinante el concepto de navegar, pues es un concepto que se puede relacionar directamente con el movimiento, promoviendo los principios de *cognetics* mencionados en la sección 5.1.3. En este caso el modelo humano sería el desplazamiento en el espacio.

Sin embargo no es necesario que toda la información se traduzca de gestos a gestos, se pueden utilizar elementos como la posición para controlar sintetizadores con otro tipo de función, como un sonido de fondo o disparo de eventos incidentales, aumentando las posibilidades dinámicas del sonido.

También se abren canales de comunicación que no quedan totalmente explorados, como una representación visual del gesto, control de parámetros de la imagen en movimiento por el audio, sincronización de eventos incidentales con eventos visuales, la definición de objetos virtuales con los cuales interactuar, como: botones, elementos que se puedan arrastrar o empujar; partículas y sistemas de parvadas. La retroalimentación visual en el uso del instrumento es importante, sin embargo las posibilidades para desarrollar un discurso paralelo o de apoyo al gesto todavía queda por explorarse.

En otra linea de análisis y recuperando la definición de interactividad expuesta en la sección 2.3, la obra no implementa elementos suficientes para considerar que la interactividad con la computadora se desarrolle más allá de un nivel reactivo. Todavía queda lugar para la implementación de reconocimiento de patrones gestuales, como movimiento circular con las manecillas del reloj o en contra, detección de posturas de la mano y reconocimiento de trayectorias, por mencionar algunos. La implementación

de algoritmos de inteligencia artificial podría llevar a aplicaciones donde el programa reconociera usuarios, reaccionando diferente a cada uno o hiciera un estimado del estado emotivo del interprete y respondiera de acuerdo.

Finalmente, el papel de las audiencias en el diseño de una obra con esta herramienta es importante, pues son ellas quienes descifran los signos y le dan sentido a lo que ocurre en escena. En este sentido hay que tomar en cuenta factores como la transparencia de la interacción deseada para cada motivo, la independencia del ejecutante del instrumento, la libertad para improvisar y la correlación de todos los elementos involucrados: visualización, sonido y movimiento. Elementos que cuentan cada uno con un poder discursivo que puede hacer que una obra se salga de control y pierda inteligibilidad. Por otro lado, el papel que éste instrumento puede tomar en una obra puede extenderse desde un instrumento sonoro que genera figuras gestuales integrado en un ensamble, hasta el centro de una obra que con el gesto genera suficiente información para desarrollar una obra transdisciplinaria con una visualización y diseño del movimiento que se involucren con un contexto discursivo.

Queda entonces como trabajo a futuro la creación de obras para la exploración del potencial que tiene esta herramienta. La conformación de un colectivo que con la participación de diferentes artistas experimenten con las posibilidades de interacción y propongan obras que den un contexto para el diseño de composiciones sonoras y visuales que se conjuguen para crear un repertorio de obra interactiva.

Bibliografía

[1] M. Puckette, "Max at seventeen.," *Computer Music Journal*, vol. 26, no. 4, pp. 31–43, 2002.

[2] Platón, *Cratilo o Del lenguaje*. Editorial Trotta, 2002.

[3] P. Ekman and W. V. Friesen, "The repertoire of nonverbal behaviour: Categories, origins, usage, and coding.," *Semiotica*, vol. 1, no. 1, pp. 49–98, 1996.

[4] R. Jackendoff, "Parallels and nonparallels between language and music.," *Music Perception*, vol. 26, pp. 195–204, 2009. http://www.mva-org.jp/Proceedings/2011CD/papers/09-31.pdf.

[5] J. C. Alexander, *Las teorías sociológicas desde la Segunda Guerra Mundial*. Gedisa, 2000.

[6] D. Huron, "Lecture 2. an instinct for music: Is music an evolutionary adaptation?," *The 1999 Ernest Bloch Lectures*, 2002.

[7] R. Packer and K. Jordan, *Multimedia: From Wagner to Virtual Reality*. W. W. Norton and Company, 1992.

[8] G. Bettetini and F. Colombo, *Las Nuevas Tecnologías de la Comunicación*. Paidos, 1995.

[9] D. Smalley, "Spectromorphology: explaining sound-shapes," *Organized Sound*, vol. 2, no. 2, pp. 107–126, 1997.

[10] T. O. Organization, "The openni organization.," 2011. http://www.openni.org/.

[11] R. E. G. del MIT Media Lab., "Responsive environments group del mit media lab.." http://resenv.media.mit.edu/.

[12] T. N. U. I. Group., "The natural user interface group.." http://nuigroup.com/.

[13] L. E. Miller, "Sicib: An interactive music composition system using body movements.," *The Musical Quarterly*, vol. 85, no. 3, pp. 545–567, 2001.

[14] R. Morales-Manzanares, E. F. Morales, R. Dannenberg, and J. Berger, "Sicib: An interactive music composition system using body movements.," *Computer Music Journal*, vol. 25, no. 2, pp. 25–36, 2001.

[15] W. Siegel and J. Jacobsen, "The challenges of interactive dance: an overview and case study.," *Computer Music Journal.*, vol. 22, no. 4, pp. 29–43, 1988.

[16] W. Siegel, "Waine siegel: Composer." http://www.waynesiegel.dk/?pid=30.

[17] G. Bradski and A. Kaehler, *Learning OpenCV.* O'Reilly Media, 2008.

[18] A. Camurri, P. Coletta, M. Peri, M. Ricchetti, A. Ricci, R. Trocca, and G. Volpe, "A real-time platform for interactive dance and music systems," *proc. International Computer Music Conference*, 2000.

[19] A. Camurri, B. Mazzarino, and G. Volpe, "Analysis of expressive gesture in human movement: The eyesweb expressive gesture processing library," *proc. XIV Colloquium on Musical Informatics*, 2003.

[20] T. Winkler, "Motion-sensing music: Artistic and technical challenges in two works for dance.," *proc. International Computer Music Conference*, pp. 471–474, 1998.

[21] A. Peñalba Acitores, "Jornadas de danza interactiva con personas con capacidad motriz reducida," *Revista Transcultural de Música*, vol. 14, 2010.

[22] M. Tribe and R. Jana, *Arte y nuevas tecnologías.* Taschen, 2006.

[23] L. Manovich, *The Language of New Media.* The MIT Pess, 2002.

[24] S. Rafaeli, "Interactivity: From new media to communication.," *Sage Annual Review of Communication Research: Advancing Communication Science.*, vol. 16, pp. 110–134, 1988.

[25] R. Packer and K. Jordan, *Multimedia: from Wagner to virtual reality.* Norton, 2002.

[26] J. Pressing, *Synthesizer performance and real-time techniques.* Computer music and digital audio series, A-R Editions, 1992.

[27] J. Paradiso, "Electronic music interfaces.," Marzo 1998. http://web.media.mit.edu/ joep/SpectrumWeb/SpectrumX.html.

[28] C. Cadoz and M. M. Wanderley, "Gesture - music," *Trends in Gestural Control of Music*, pp. 71–93, 2000.

[29] P. Charles Sanders, *Collected Papers of CHARLES SANDERS PEIRCE.* Harvard University Press, 1958.

[30] P. Magli, "Para una semiótica del lenguaje gestual.," *deSignis*, vol. 3: Los Gestos, sentidos y prácticas., pp. 37–51, Octubre 2002.

[31] P. Tagg, "Introductory notes on the semiotics of music.," Julio 1999. http://tagg.org/xpdfs/semiotug.pdf.

[32] A. Storr, *La música y la mente: El fenómeno auditivo y el porqué de las pasiones.* Paidos de música, 2002.

[33] R. I. Godøy and M. Leman, *Musical Gestures: Sound, Movement, Meaning.* Routledge, 2010.

[34] A. Tomatis, *El oído y el lenguaje.* Ediciones Orbis, S.A., 1969.

[35] G. Mazzola and M. Andreatta, "Diagrams, gestures and formulae in music," *Journal of Mathematics and Music*, vol. 1, no. 1, pp. 23–46, 2007.

[36] C. Doge and A. T. Jerse, *Computer Music: Synthesis, Composition and Performance*. Schirmer - Thomson Learning, 2nd ed., 1997.

[37] C. Roads, *The Computer Music Tutorial*. The MIT Press, 1996.

[38] F. F. O. L. D. O. Computing., "Foldoc: Free on line dictionary of computing.." http://foldoc.org/Application+Program+Interface.

[39] OpenFrameworks., "Of- about." http://www.openframeworks.cc/about.

[40] U. B. CNMAT, "opensoundcontrol.org," Diciembre 2011. www.opensoundcontrol.org.

[41] PrimeSense., "Resources." http://www.primesense.com/press-room/resources/file/4-primesense-3d-sensor-data-sheet.

[42] J. Stowers, M. Hayes, and A. Bainbridge-Smith, "Quadrotor helicopter flight control using hough transform and depth map from a microsoft kinect sensor.," *proc. Conference on Computer Vision Applications*, pp. 9–31, June 2011. http://www.mva-org.jp/Proceedings/2011CD/papers/09-31.pdf.

[43] PrimeSense., "Primesense: Natural interaction." http://www.primesense.com.

[44] J. MacCormick, "Selected talks: How does the kinct work?." http://users.dickinson.edu/ jmac/selected-talks/kinect.pdf.

[45] R. Vision, "Razor vision: Making computer vision a little easier.," Diciembre 2011. http://razorvision.tumblr.com/post/15039827747/how-kinect-and-kinect-fusion-kinfu-work.

[46] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Prentice-Hall, 2002.

[47] R. Szeliski, *Computer Vision: Algorithms and Applications.* Springer, 2010.

[48] D. Chetverikov and Z. Szabó, "A simple and efficient algorithm for detection of high curvature points in planar curves," *proc. 23rd Workshop of the Austrian Pattern Recognition group*, 1999.

[49] J. Raskin, *The Humane Interface - New Directions for Designing Interactive Systems.* Addison-Wesley Longman, 2000.

[50] A. Shmeder, A. Freed, and D. Wessel, "Best practices for open sound control," *porc. of the Linux Audio Conference*, pp. 131–140, 2010.

[51] F. R. Moore, *Elements of Computer Music.* P T R Ptrentice Hall, 1990.

[52] P. Boulez, "Timbre and composition -timbre and language.," *Contemporary Music Review*, vol. 2, pp. 161–171, 1987.

[53] R. Schafer, *The Soundscape: Our Sonic Environment and the Tuning of the World.* Destiny Books, 1994.

[54] D. Keller and T. Barry, "Ecologically-based granular synthesis," *proc. International Computer Music Conference*, pp. 117–120, 1998.

[55] M. Rocha Iturbide, G. Eckel, and B. Becker, "The development of gist, a granular synthesis toolkit based on an extension of the fof generator.," *proc. International Computer Music Conference*, pp. 296–302, 1995.

[56] C. Roads, *Micro Sound.* The MIT Press, 2001.

[57] U. Eco, *Kant and the platypus.* VINTAGE U.K. Random House, 2000.

[58] J. McCartney, "Supercollider: a new real time synthesis language.," *proc. International Computer Music Conference*, pp. 157–258, 1996.

[59] J. McCartney, "Rethinking the computer music language: Supercollider.," *Computer Music Journal*, vol. 26, no. 4, pp. 61–68, 2002.

[60] M. V. Mathews and J. Pasquale, "Rtsked: A scheduled performance language for the crumar general development system.," *proc. International Computer Music Conference*, p. 286, 1981.

[61] C. Scaletti, "Computer music languages, kyma and the future," *Computer Music Journal*, vol. 26, no. 4, pp. 69–82, 2002.

[62] G. Wang, P. R. Cook, and A. Misra, "Designing and implementing the chuck programming language," *proc. International Computer Music Conference*, pp. 17–20, 2005.

[63] G. Wang, R. Fiebrink, and P. R. Cook, "Combining analysis and synthesis in the chuck programming language," *proc. International Computer Music Conference*, pp. 17–20, 2007.

[64] R. Boulanger, *The Csound Book.* The MIT Press, 2000.

[65] MESO, "vvvv - a multipurpose toolkit.." http://www.meso.net/vvvv.

[66] vvvv, "vvvv- a multipurpose toolkit.." http://vvvv.org.

[67] FreeFrame., "Freeframe.." hhttp://freeframe.sourceforge.net.

[68] TroikaTronix., "Troikatronix.." http://www.troikatronix.com/.

[69] A. Inc., "Quartz composer user guide," 2004-2007.

[70] I. Greenberg, *Processing: Creative Coding and Computational Art.* Firendsof ED, 2007.

[71] I. Corporation, "Open source computer vision: Reference manual," 1999-2001.

[72] P. Martz, *OpenGL Distilled.* Addison Wesley Profesional, 2006.

[73] L. Dostálek and A. Kabelová, *Understanding TCP/IP: A clear and comprehensive guide to TCP/IP protocols.* Packt Publishing, 2006.

[74] D. Wessel and M. Wright, "Problems and prospects for intimate musical control of computers.," *Computer Music Journal*, vol. 26, no. 3, pp. 11–22, 2002.

[75] F. R. Moore, "The dysfunctions of midi," *Computer Music Journal*, vol. 12, no. 1, pp. 19–28, 1988.

[76] M. Wright and A. Freed, "Open sound control: A new protocol for communicating with sound synthesizers.," *proc. International Computer Music Conference*, pp. 101–104, 1997.

Apéndices

Apéndice A

SuperCollider

Creado por James McCartney su primera versión salió en 1996 como un ambiente para síntesis de audio en tiempo real[58]. Implementaba algoritmos novedosos, muchos de los cuales vieron su camino hasta la versión actual, la tercera.

Desde su primera versión, SuperCollider fue diseñado para la síntesis de sonido en tiempo real utilizando un lenguaje de programación de alto nivel tanto para la programación de sintetizadores como para la creación de algoritmos de composición.

Fue diseñado con un método de *"garbage collection"* dinámico que libera y maneja la memoria en tiempo real, permitiendo a los sintetizadores ser instanciados dinámicamente, lo que da lugar a que no haya que definir un número máximo de voces estático.

A diferencia de *Max*, SuperCollider es un lenguaje de programación interpretado.

Implementó un modelo de síntesis a partir de *"Closures"*, que representan una función y el estado (*environment*)en que se encuentran sus variables dentro de un mismo dato. Este sistema permite calcular las muestras (samples) dentro de un buffer (contenedor) de audio a un tiempo de control, mientras que el buffer se esta ejecutando a tiempo de audio. De esta manera es posible calcular señales generadas por procesos complejos para su reproducción en tiempo real.

Para agilizar el procesamiento de señales y siguiendo el paradigma de programación orientada a objetos, McCartney creó un tipo de dato de señal que representa un buffer de determinado número de muestras (samples). Al diseñar operadores que funcionaran sobre estos buffers es posible agilizar el trabajo del interprete para manipular señales

en tiempo real.

La segunda versión de SuperCollider buscó mantener el lenguaje de programación de alto nivel con el sistema de síntesis, sin embargo esto implicaba que el proceso de síntesis tenía que ser interrumpido para generar eventos de control.

Finalmente, en la tercera versión de SuperCollider [59] se separó el sistema de síntesis en un servidor que se ejecuta a parte. Así las instrucciones de composición y control pueden ser ejecutadas a parte y calculadas con anticipación y sin interrumpir el proceso de síntesis.

SuperCollider3 implementa un modelo de cliente y servidor en el cual el cliente se encarga de procesar e interpretar la parte del lenguaje de alto nivel y el servidor procura la creación de datos para la síntesis en tiempo real.

La comunicación entre las dos partes se logra a través de mensajes enviados vía alguno de los protocolos de red UDP o TCP, manteniendo la información en un protocolo modificado de Open Sound Control (OSC).

SuperCollider es modular y define sus componentes siguiendo la tradición de la programación orientada a objetos a tal punto que todo en SuperCollider es un objeto. Las unidades generadoras de señales son todas implementadas como objetos de una misma clase: UGens.

Como en todos los lenguajes de programación musical, su orden de ejecución es crucial para el buen funcionamiento del sistema. Para garantizar el orden correcto SuperCollider organiza los Sintetizadores en grupos estructurados en forma de gráficas de árbol, de tal manera que cada nodo puede ser un Synth o un grupo.

Cada nodo en el árbol es identificado por un número que nos permite referenciarlo y el orden de ejecución es definido al recorrer el árbol nodo por nodo.

Finalmente, un par de arreglos globales se encargan de transportar la información entre sintetizadores para permitir que se interconecten. Los arreglos tienen la función de buses, uno se utiliza para flujo de información a velocidad de audio y el otro a velocidad de control, es decir a $\frac{1}{64}$ veces la velocidad de audio.

Dado que el mismo diseño de SuperCollider implica un servidor y un cliente en comunicación vía los mismos protocolos que son utilizados para transmitir mensajes por

internet (TCP/UDP), la posibilidad de controlar el motor de síntesis en un servidor desde varios clientes remotos es inherente al diseño. Ahora realizar conciertos simultáneos en lugares geográficamente lejanos es relativamente fácil.

Con SuperCollider podemos definir sintetizadores en servidores remotos y controlarlos de manera local, lo que implica que la información a transmitir no es información de audio, si no información de control. La gran diferencia entre transmitir audio o información de control vía internet esta en la cantidad. Transmitir comandos de control para cambiar los parámetros de un sintetizador alojado en el servidor remoto requiere mucho menos información que el trasmitir una señal sonora de un sintetizador local a una máquina remota.

Tenemos con SuperCollider la posibilidad de interactuar en *tiemporeal* con una o más personas que se encuentren a kilómetros de distancia en lugares diferentes y todos escuchar los resultados sonoros a la vez. Lo único que necesitamos es una conexión de red y el número *ip*[1] de los participantes con quien vamos a tocar.

Desde 2002 SuperCollider ha sido distribuido bajo la licencia GNU y su código está disponible para ser bajado de la red vía un servidor CVS. SuperCollier sigue siendo un sistema en crecimiento, cuenta con su propio repositorio de librarías para extender el lenguaje y un sistema para manejarlas desde el mismo ambiente llamado *"Quarks."*

[1]el número ip es asignado por el proveedor del servicio de internet, es único para cada conexión y sirve como dirección postal para enviar mensajes por la inmensa red informática del internet

Apéndice B

Max/MSP y Pure Data

A pesar de ser dos lenguajes distintos de programación, Max/MSP y Pd implementan en su núcleo el mismo paradigma.

Diferenciando del nombre de los lenguajes que lo implementan, su creador, Miller Puckette, llama a su paradigma *"Max"* en honor a Max Mathews, cuyo programa de música por computadora RTSKED [60] es su influencia más importante.

El paradigma *Max* implementa bloques constructores prefabricados listos para llevar a cabo configuraciones útiles para la ejecución de música por computadora, lo cual implica [1]:

- Un protocolo de control de calendarización de eventos: scheduling control.
- Cómputo a velocidad de audio (audio rate).
- Un acercamiento vía modularización y componentes
- Intercomunicación entre módulos y componentes.
- Una representación gráfica
- Un editor de patches

Max/MSP y Pd son ambientes de programación musical gráficos, que proveen al usuario de una serie de bloques constructores, cada uno con una función diferente y diseñados para la producción musical y su ejecución en *tiemporreal*. Cada uno de estos

bloques es representado por un rectángulo que aloja su nombre. Están provisto con entradas y salidas de tal manera que pueden ser interconectados de manera gráfica vía lineas que se trazan entre ellos y permiten así crear configuraciones complejas de procesos de señales de audio.

Max es un lenguaje de programación compilado, una característica importante que implementa es su capacidad de estructurar programas en módulos, donde cada módulo esta programado para llevar a cabo una tarea especifica y puede estar escondido en un contenedor dando la apariencia de ser un componente más. De esta manera *Max* permite organizar programas complejos en varios módulos que también pueden ser reutilizados posteriormente.

A cada programa hecho en *Max* se le llama *"patch"*, haciendo referencia a los antiguos sistemas análogos de síntesis, como el Buckla y el Moog, que utilizaban cables para interconectar módulos entre si e implementaban el control por voltaje. A cada configuración de estos sistemas se le llamaba un *patch* o *parche* como comúnmente se traduce al español.

Uno de los grandes logros de *Max* es su protocolo para agendar (schedule) y sincronizar eventos para su control y ejecución en *tiemporeal*. Como el orden de ejecución no es conocido previo a la programación y el flujo de información entre componentes es crucial, un protocolo para controlar cuándo debe ejecutarse cada cosa es necesario. Cada componente necesita información sobre la cual llevar a cabo el proceso para el cual esta programado. Si la información no esta presente porque los bloques anteriores no se han ejecutado tenemos un error. El protocolo que *Max* implementa tanto en Max/MSP como en Pd (a pesar de ser diferentes[1]) cumple con la misma función: mantener el orden de ejecución de los bloques y procurar una sincronía en el flujo de datos.

Sin embargo creo que el punto más importante del paradigma de programación musical *Max*, además del gran trabajo de scheduling, es que busca ser un programa neutral, es decir un programa que no impone una idea personal de lo que es la música (de esta forma Miller Pueckette nos entrega un laudero, más que un instrumento más).

Miller lo expresa muy bien en las siguientes lineas:

> "El diseño de software no puede evitar afectar como suena la música por

> computadora, pero nosotros, programadores de software, debemos intentar no proyectar nuestras propias ideas a través del software " [1]

Este objetivo remarca la importancia de la diferencia de lo que es un lenguaje de programación musical y lo que es un programa musical.

Creo que es fácil confundir programas musicales y lenguajes de programación musical, estando la diferencia en que un lenguaje tiene un vocabulario y una gramática, con lo cuál nos habilita a expresar gran variedad de ideas y resolver un rango muy amplio de problemas, generalmente un programa musical esta hecho para un propósito especifico con lo cual su diseño se reduce a resolver cierto rango de problemas.

Carla Scaletti lo expresa muy bien en las siguientes lineas:

> "Un lenguaje lo provee a uno con un set finito de "palabras" y una "gramática" para combinar estas palabras en frases, enunciados y párrafos para expresar una variedad infinita de ideas. Un lenguaje no hace nada por si mismo; uno utiliza un lenguaje para expresar sus propias ideas. Esto es lo que hace que este paquete particular de softwares sea tan abierto, extensible y capaz de ser utilizado de formas no anticipadas por el autor"[61]

A pesar de que Miller piensa que falta mucho para que *Max* logre su objetivo como un lenguaje neutral, *Max* tiene ya 25 años desde que comenzó a gestarse en el IRCAM en 1985 y su desarrollo a través de los años ha sido constante. La importancia de *Max* en la música electroacústica de los 90's hasta la actualidad es innegable.

Apéndice C

ChucK

Este sistema de programación musical sale en su primera versión en el año 2004 como un proyecto de la universidad de Princeton con el propósito de implementar un lenguaje de alto nivel que permita tanto la programación de síntesis y composición, como el desarrollo de procesos y algoritmos a nivel de muestras, manteniendo un control preciso de ocurrencia de eventos en el tiempo. Además, ChucK propone un lenguaje cuya estructura permita una legibilidad comprensible y representativa del proceso u algoritmo programado, así como la posibilidad de programar e implementar cambios durante su ejecución en *tiemporreal*.[62]

A diferencia de SuperCollider, ChucK implementa un sistema integrado desde el cual se controla la síntesis de audio y los eventos de composición. Cuenta con un compilador que interpreta al vuelo las lineas de código para una maquina virtual (VM) que reparte sus recursos de computo en hilos (shreds) de procesos que son ejecutados en paralelo y sus resultados entrelazados por un sistema coordinador (shreduler) que se apoya en la noción de un tiempo común cuyo estado es accesible a todos los procesos. El sistema cuenta además, con un interpretador que se encarga de actualizar los argumentos de entrada y pasarlos a las unidades generadoras en el motor de síntesis para controlarlas, ya sea por interfaces humano-máquina o señales recibidas por un servidor que constantemente espera información de la red. Finalmente, las unidades generadoras son ordenadas en una configuración de gráfica y ejecutadas según un protocolo controlado estrictamente por una variable de tiempo *"now"*. Esta variable mantiene el tiempo

actual de acuerdo a las muestras computadas y permite a cada unidad conocerlo. Al computar una muestra cada unidad agenda el tiempo en que calculará su próxima muestra y el shreduler se encarga de avanzar el tiempo una vez todas las muestras del tiempo actual han sido calculadas.[62] Es por ésta razón que Chuck es presentado como un sistema de programación *"Strongly-Timed"* (fuertemente sincronizado[1]). Actualmente Chuck sigue desarrollándose, implementando un nuevo tipo de "Unidades Analizadoras." El objetivos es poder implementar algoritmos de análisis que puedan compartir información con Unidades Generadoras y permitir síntesis por análisis en un mismo sistema que tiene acceso directo desde un lenguaje de alto nivel a los cálculos muestra por muestra del motor de síntesis[63]. Chuck es un proyecto joven que sigue en desarrollo, tal vez en el futuro su uso sea más común. Sin embargo creo que su particular diseño aporta al proceso de desarrollo de ambientes y lenguajes de programación musical.

[1]traducción propuesta por el autor

Apéndice D

Csound

Desarrollado en el MIT por Barry Vercoe en 1985, Csound es un programa para la creación musical que implementa un motor de síntesis manejado por dos tipos de archivos: el de orquesta y el de partitura.[64]

Csound se echa a andar de una forma análoga a una función con argumentos en un lenguaje de programación de alto nivel.

Los argumentos de la función serían los archivos de orquesta y partitura.

El archivo de orquesta describe los instrumentos que serán utilizados y el archivo de partitura describe los momentos en que los instrumentos serán tocados y la duración del evento.

Csound resuelve el problema de agendar eventos al pedirle al usuario en la partitura una descripción exacta (en forma de lista) de los momentos en que ocurrirán.

El archivo de orquesta describe al programa instrumentos cómo interconexiones lógicas entre generadores de señales, los cuales pueden ser accionados según las instrucciones escritas en la partitura.

Al echar a andar el programa, la partitura es reordenada en orden cronológico y argumentos necesarios para el control de eventos en el tiempo como, pulsos por segundo y aceleración son tomados en cuenta para la ejecución. Los comandos de la orquesta son traducidos a código de máquina, aparta memoria suficiente para instanciar los generadores de señales necesarios, los arregla en instrumentos según la programación del usuario y genera una estructura para controlarlos con la información de la partitura.[64]

(pg. 99-105)

En Csound, como en otros programas hay dos velocidades para la ejecución del programa, una para la generación de señales de audio y otra para el control de cómo se están generando las señales.

Csound es un programa con el cual se pueden lograr gran variedad de instrumentos de síntesis, incluyendo instrumentos que implementan análisis de audio en tiempo real, sin embargo su diseño no es para ejecución en tiempo real. Csound es un programa que toma la programación de usuario y la imprime en un archivo de audio. Esto no es en detrimento de la importancia de Csound ya que como *Max* ha influenciado fuertemente el desarrollo de la música electroacústica.

Apéndice E

OpenFrameworks

OpenFramewoks es un proyecto todavía en estado de desarrollo (su actual versión es 0.6) que se inspira en Processing para crear una serie de librerías que se incorporan en un sistema contenido escrito en C++ llamado *Framework*.

OpenFrameworks se distribuye con el Framework montado en proyectos (archivo en que la IDE guarda una configuración específica) para IDE's que funcionan en Linux, Mac OS X y Windows, haciendolo bastante portable. En realidad OpenFrameworks es my portable, cualquier máquina que contenga un compilador de C++ puede echar a andar un programa escrito con OpenFrameworks, solo es necesario hacerle accesible todas las librerías en que se basa, tarea no siempre fácil si no se cuenta con una IDE.

Una IDE es un *"Ambiente Integrado de Desarrollo"*[1] que permite mantener en un proyecto la configuración necesaria para que el compilador tenga accesible toda la información que requiere para compilar un programa, eso implica enlazar las librerías y los archivos que se requieren para la aplicación programada. Distribuciones de *frameworks* de este tipo nos evitan tener que configurar un proyecto, una tarea que puede ser difícil si no se tiene la suficiente experiencia programando y nos permite comenzar a codificar inmediatamente.

OpenFrameworks se distribuye con todo su código sin compilar y accesible, esto hace que sea muy fácil extender sus librerías y deducir su forma de operación.

OpenFrameworks tiene ya en su versión 0.7 una gran comunidad de usuarios y un

[1] "Integrated Development Environment"

foro muy activo donde es posible interactuar con otros usuarios, intercambiar ideas, ayudar y ser ayudado.[39]

Apéndice F

vvvv

Similar a *Max/MSP, vvvv* es un ambiente de programación gráfico que permite crear aplicaciones para la síntesis de video principalmente. Fue desarrollado por el grupo MESO en Frankfurt Alemania en 1998 como un herramienta para uso interno de la compañía que sirviera para llevar a cabo prototipos de instalaciones interactiva de manera rápida y eficaz. En el año 2002 *vvvv* es lanzado al publico de manera gratuita para uso en aplicaciones no comerciales.[65]

vvvv es un programa diseñado para funcionar en tiempo real, trabaja bajo el principio de representación visual de objetos mediante rectángulos que pueden ser interconectados para crear flujo de datos a través de diversos procesos. Sus principales aplicaciones son la generación de imágenes en movimiento, visuales y audio interactivo en ambientes multiusuario así como el manejo de ambientes con un amplio espectro de materiales multimedia. Utiliza las herramientas del API DirectX de Microsoft para el manejo de recursos de las tarjetas gráficas lo cuál le permite ejecutar con la eficiencia de los videojuegos, sin embargo esto implica que su uso esta restringido a sistema operativo Windows. Una de las características de *vvvv* es el uso de *spreads*, una técnica mediante la cual es posible manipular muchos objetos de manera indistinta a como se manipularía uno solo.[66] Este ambiente de programación es expansible mediante la programación de procesos en la plataforma de plugins de video "FreeFrame," también disponible de manera gratuita y con código abierto en la red. [67]

Apéndice G

Isadora

Isadora también es un ambiente de programación gráfico. Surge de la compañía de danza *TroikaRanch* desarrollado por Mark Coniglio para satisfacer las necesidades del grupo de tener una herramienta para desarrollar aplicaciones interactivas y multimedia para las artes escénicas. Isadora permite recibir información desde sensores y procesarla para controlar sonido y video. Su creador afirma que es fácil de aprender, desafortunadamente no es un programa de acceso gratuito ni de código abierto por lo tanto no hay mucha información acerca de su diseño. Corre en su versión más avanzada en el sistema operativo de Mac OSX y en una versión anterior en Windows de Microsoft.[68]

Apéndice H

Quartz Composer

Al igual que Isadora y *vvvv*, Quartz Composer (QC) es un ambiente de programación gráfico. QC viene incluido con el sistema operativo OS X de Mac, provee un editor gráfico para desarrollar programas interconectando objetos como en los ambientes mencionados anteriormente. En QC a cada programa generado con su editor gráfico se le llama composición y tiene la ventaja de poder acceder a cada composición de manera programática además del editor gráfico. QC esta pensado para permitir el uso de la tecnología disponible en el Sistema operativo OS X sin requerir que el usuario sea experto en ella.

Cuenta con un framework para desarrollar objetos propios y desarrollar aplicaciones independientes, es decir, auto contenidas que no necesitan del editor de QC para ser ejecutadas.

Quartz Composer viene con una serie de herramientas para trabajar con 3D usando OpenGL, MIDI, QuickTime y varios recursos nativos del sistema operativo OS X, lo que permite crear aplicaciones visuales para el acceso general del sistema operativo, como vídeos QuickTime y *"Screen Savers"*. [69]

Como es de esperarse QC es exclusivo para Mac y requiere de licencia de propietario, sin embargo ésta viene incluida con el sistema operativo y provee su código de manera abierta en el paquete de desarrollador incluido en la distribución del OS X.

Apéndice I

Processing

Processing es más que un ambiente de programación, es un lenguaje dentro de otro. Basado en Java, Processing necesita del interprete de la *Máquina Virtual (JVM)* de Java para poder poder funcionar. La JVM es una especie de sistema operativo, un interpretador que compila el Lenguaje Java y lo traduce a lenguaje de máquina. El lenguaje de máquina puede ser específico para cada máquina y el sistema operativo sobre el cual están corriendo. Así una Mac OS X tiene un lenguaje de máquina distinto al de una PC corriendo Windows o una corriendo Linux. Entonces la JVM tiene que ser especifica para cada sistema, pero el lenguaje Java puede ser igual indistintamente del sistema operativo, esto permite que podamos escribir programas en lenguaje Java y poderlo correr en cualquier sistema operativo. Al final es la JVM la que se encarga de interpretar el programa y convertirlo en lenguaje de máquina. La JVM viene instalada por omisión en casi todos los sistemas operativos, por lo cual es de fácil acceso.

Processing es una *IDE*, que en español significa Ambiente de Desarrollo Integrado, escrito en lenguaje Java que nos permite escribir y compilar en código Java. Processing es un IDE pensado para poder ser de acceso a un grupo de usuarios muy general, no solo expertos en programación, también a diseñadores, artistas y principiantes. Es por esa razón que Processing se presenta como una ventana con muy pocos botones, con alto contenido de significado que puede fácilmente descodificado.

[Imagen del IDE de Processing]

Hay un marco de edición de código, un botón de play que lo compila y un marco de

mensajes que nos permite saber el resultado de la compilación. El IDE de Processing incluye a su vez una serie de librerías con una serie de funciones que hacen más fácil programar aplicaciones con contenidos gráficos y multimedia en movimiento (tiemporreal). Por esta razón Processing es un Lenguaje de programación de multimedia en un IDE que lo puede compilar (son tan solo funciones y objetos de Java) y ademas nos permite programar en Java, situandonos en una situación en que tenemos un lenguaje en otro lenguaje, Processing en Java. El compilador de Java que utiliza la IDE de Processing se llama *jikes*, un proyecto de código abierto de IBM.

Processing contiene funciones para trabajar con geometría, 3D (mediante P3D y OpenGL), video, sonido, redes, MIDI, OSC, OpenCV (VisiónComputarizada) y muchas cosas más que colaboradores van añadiendo al lenuaje.

La gran ventaja de Processing es que es fácil de programar y lo más importante: fácil de aprender.

Processing se ha expandido rápidamente y cuenta actualmente con una gran cantidad de usuarios, por lo cual es fácil compartir trabajo con mucha gente, siendo de especial interés (por la naturaleza de nuestro trabajo) gente que se dedica al arte.[70] [pgs. 143- 170]

Apéndice J

EyesWeb

EyesWeb es una plataforma para sistemas interactivos de danza y música.

Su objetivo, según su creador (A.Camurri 2000), es ofrecer un sistema integrado que permita al usuario experimentar con modelos de comunicación expresiva y estrategias de mapeo gestual, además de ser fácil de aprender y utilizar.[18]

EyesWeb se comprende de un software de programación gráfica (Como *vvvv, MAX* y Pd) con una serie de bloques basados en OpenCV que llevan a cabo operaciones básicas (suma, multiplicación, resta, dilatación, erosión) en *tiemporreal* sobre un tipo de variable que representa imágenes, habilitandonos a trabajar con video e imágenes capturadas continuamente por una cámara, es decir Visión Computarizada.

Los bloques de EyesWeb, así como en Pd y SuperCollider se dividen en librerías, siendo de particular interés la librería llamada *Exspressive Gesture Processing Library* que fue diseñada para extraer información de la danza.

La información que extrae fue escogida a partir de un estudio del movimiento siguiendo los principios de descripción del movimiento de Laban (¿Quién es Lavan? las doñas en los lavaderos).

Algunos de los datos que EyesWeb es capaz de extraer son: cantidad de movimiento, índice de contracción e índice de estabilidad.

Pero es posible acceder a otros de igual importancia como: posición, área, rectángulo de frontera del bailarín y trayectorias

EyesWeb es un proyecto muy activo que incluye además la creación de hardware

específico para la transducción de información gestual y captura de video desde dos cámaras con una misma tarjeta de digitalización.

Actualmente el software se encuentra es su quinta generación con la versión XMI (eXtended Multimodal Interaction), se distribuye de manera gratuita y abierta con una serie de condiciones de distribución mínimas. [18]

EyesWeb es extensible, permite que uno programe sus propios objetos.

Apéndice K

OpenCV

OpenCV es una herramienta para el desarrollo de proyectos que implementen Visión Computarizada. Desarrollado originalmente por Intel, OpenCV es un conjunto de librerías escritas en C++ que fácilmente pueden ser integradas en proyectos mediante API's [1]. De hecho, OpenCV se incluye en muchas herramientas de programación y frameworks. Su amplia funcionalidad para trabajar con imágenes y realizar operaciones en ellas, permite implementar algoritmos de Visión Computarizada de manera más ágil, a demás de estar optimizado para funcionar en *tiemporreal.*[17]

Para conocer todas las funciones de OpenCV el *"Open Source Computer Vision: Reference Manual"*[71] las presenta junto con ejemplos de uso.

En OpenFrameworks OpenCV se incluye envuelto en un API que ofrece acceso a las funciones más comunes, sin embargo se puede acceder a toda la potencia de OpenCV llamando directamente a sus funciones sin problemas de compatibilidad.

[1] ver pg. 34 para definición de API

Apéndice L

OpenGL

Open Graphics Library es una API que le permite al programador llamar funciones que se implementan directamente en el hardware para llevar a cabo el *rendering* de objetos en dos y tres dimensiones.

Al tener acceso directo a las tarjetas de gráficos, OpenGL es un sistema muy efectivo para la creación de gráficos en tiempo real y es independiente del sistema operativo. Sin embargo OpenGL no cuenta con soporte para integrar Interfaces de usuario ni sistemas de ventanas. Para poderlas implementar, OpenGL requiere combinarse con otras librerías que si lo hagan. Desafortunadamente dichas librerías no suelen ser independientes del sistema operativo, por lo cual OpenGL comúnmente se combina con GUL y GLUT. La primera le permite tener acceso a las interfaces de usuario y la segunda implementa un sistema de ventanas independiente del sistema operativo.[72][Sección 1.1-1.3]

Actualmente OpenGL es casi un estándar para la programación de modelos en tres dimensiones y sus librerías vienen incluidas en casi todos los sistemas de síntesis de video.

En OpenFramewoks OpenGL viene integrado como el sistema principal de gráficos, por lo que no hay que integrarlo con bibliotecas extra.

Apéndice M

Protocolos y TCP/IP

Un protocolo en telecomunicaciones es la especificación y estandarización de una serie de procedimientos que se llevan a cabo para asegurar que un mensaje transmitido llegue a su destinatario y pueda ser interpretado. Un protocolo tiene que ser estandarizado ya que se emplea tanto por el transmisor como por el receptor. Si el procedimiento no es conocido por ambas partes el éxito de la transmisión no puede ser garantizado. Son los protocolos que permiten que se logre el intercambio de información entre dos o más computadoras conectadas en una red o a través del internet. Para lograr que la información llegue de un punto a otro se empaqueta en datagramas cuya estructura esta determinada por la especificación del protocolo. Por lo general en la estructura se incluye información sobre la procedencia y el destino del paquete.

El protocolo más común utilizado en internet es el TCP/IP, su estructura consiste en una serie de capas, como se muestra en la figura M.1. Cada capa es responsable de que la transmisión sea exitosa a través de un dispositivo o tecnología presente en el camino de la información. En el protocolo TCP/IP, TCP se encarga de la capa de transporte e IP de la capa de red (*network*).

Siguiendo las especificaciones del protocolo IP (Internet Protocol), las computadoras en una red, ya sean locales o de alcance más amplio, se identifican con un número o dirección IP único que le permite a los demás usuarios de la red saber de donde proviene la información y hacia donde mandarla. El número IP de una conexión es comúnmente comparado con una dirección postal, en el sentido de que es único y que mediante él es

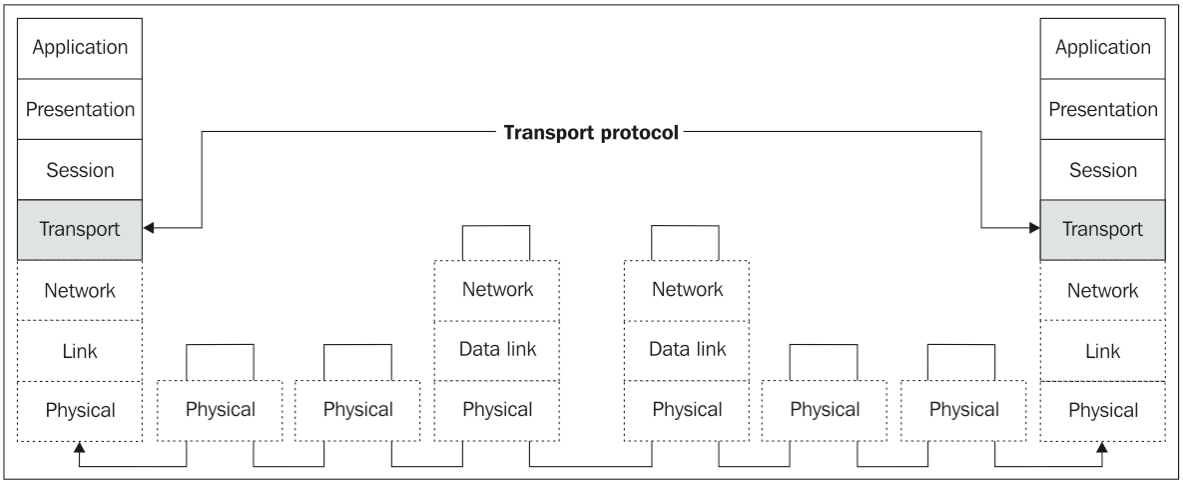

Figura M.1: Diagrama del Protocolo TCP/IP: las capas del protocolo tienen que ser implementadas por el transmisor y descodificadas por el destinatario para que la información llegue de forma exitosa a la aplicación que hará uso de ella. En este ejemplo se considera una transmisión en una red de amplio alcance [73][pg. 12]

posible localizar su procedencia geográfica.

En una red local sin conexión a internet el número IP es asignado por el Switch o Hub evitando repeticiones. Si la red es más amplia, el número IP es comúnmente asignado por el proveedor del servicio de internet.

Hoy en día es posible crear una red local sin la necesidad de involucrar cables ni Switches ni Hubs. Con la tecnología actual se puede crear una red local inalámbrica y transmitir información a través de ella. Esta solución no cambia en absoluto la manera en que nosotros programamos la transmisión de datos, pues la tecnología WIFI se encarga de resolver la transmisión inalámbrica sin alterar la forma en que está constituido el paquete de datos TCP o UDP.

El puerto es la dirección interna en la computadora en la que se va a entregar el mensaje. Los puertos nos sirven para distribuir la información a los programas que la requieren. De esta forma podemos recibir información en la misma dirección IP destinada para diferentes programas.[73][pg. 14]

Los protocolos TCP y UDP están estandarizados y su anatomía o datagrama describen cómo están constituidos. Dentro del datagrama hay un espacio para los datos

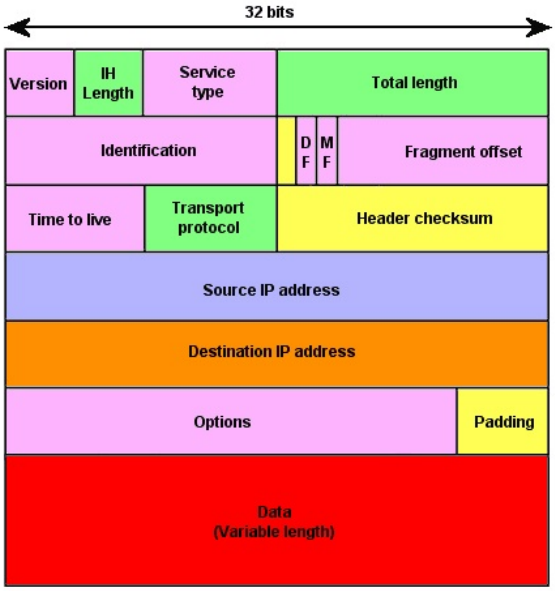

Figura M.2: Diagrama del datagrama IP

que son transmitidos, en este espacio es que se aloja el protocolo OSC, ver figuraM.2.

La diferencia entre TCP y UDP se encuentra a grandes rasgos en que TCP es un protocolo que se asegura de que la información llegue a su destino mediante un sistema que establece un dialogo con el destinatario, el cual le manda una confirmación cada que un paquete de datos es recibido. En cambio el protocolo UDP emite sus paquetes de datos sin importarle si llegan a su destinatario. Es el destinatario el que se tiene que encargar de recibirlos cuando estos son transmitidos.

Como ya se mencionó anteriormente, OSC es independiente del protocolo de transporte. Se transmite como dato (*payload*) dentro de un paquete TCP o UDP y es desenvuelto en el programa que lo recibe.

Como es de esperarse, en el sistema que aquí se propone, es necesario que todos los módulos tengan conocimiento de la dirección a la cual enviarán información, de otra manera es imposible compartir los datos.

Las direcciones IP de cada módulo pueden ser consultadas en las propiedades de conexión.

Apéndice N

OSC

El protocolo OSC surge como propuesta para hacer frente a las deficiencias presentes en MIDI, como se menciona en [74] y se pone en evidencia en [75]. OSC busca mecanismos de control abiertos y que, en contraste con MIDI, permitan un control continuo y concurrente de variables sonoras[76].

La unidad básica del protocolo es el mensaje. Un mensaje consiste de un patrón de dirección (ej. \dir) OSC seguido por un rótulo de tipo, indicando si los argumentos transferidos son *'f' floats, 'i' integers* ó *'s' strings* y finalmente los argumentos o datos.

El hecho de que OSC no tenga mensajes definidos (como en un lenguaje) nos permite generar un lenguaje específico para la interacción. OSC es otro ejemplo de lo que Miller Puckette pretende cuando declara que un programa no debe de imponer una idiosincrasia al usuario, ver pg 89. De hecho es tan abierto que puede ser utilizado para muchos más propósitos que para los que fue creado. [40][Application Areas]

El protocolo OSC es particularmente robusto en su implementación ya que permite la creación de *bundles* o grupos de mensajes que son recibidos en un mismo paquete y toman efecto simultáneamente, lo que nos permite sincronización entre procesos reduciendo al mínimo la latencia entre ellos.

Además cuenta con una estampa de tiempo, de forma que permite planear eventos que dependen altamente del tiempo para su realización.

OSC es un proyecto de código abierto que se mantiene en desarrollo. Consecuentemente podemos esperar que OSC nos siga proporcionando nuevas posibilidades para

interconectar módulos de interacción.

Es independiente del protocolo de la capa de transporte [1] por lo tanto puede incrustarse sobre TCP o UDP indistintamente, ver pg. 105 para más información sobre los protocolos.

El protocolo OSC se ha vuelto muy popular en el ambiente artístico pues es fácil de utilizar y permite compartir información a través del internet permitiendo que haya flujo de información entre varios lugares remotos, abriendo la posibilidad de explorar temas como la telepresencia y la creación colaborativa.

Otra posibilidad que OSC nos ofrece es la separación de procesos de alto consumo de CPU en computadoras diferentes, haciendo más eficiente la ejecución de procesos paralelos pero manteniendo la comunicación entre ellos. En aplicaciones interactivas es un paradigma muy útil pues separa los procesos de captura, análisis y despliegue, sin des habilitar la posibilidad de compartir información sobre los eventos que ocurren y ante los cuales hay que responder.

[1] Los protocolos para transmisión de información en la red se dividen en capas, cada capa se asegura de que la información pase a través de algún dispositivo o tecnología salvaguardando la información y verificando remitentes y destinatarios. Así, las capas del protocolo TCP/IP en orden ascendente son: *"physical, data link, network, transport, session, presentation application."* OSC es una implementación de la capa de presentación. Cada capa depende de la anterior, con excepción de la capa de presentación y aplicación que pueden funcionar directamente sobre la capa de transporte,[73]

Apéndice Ñ

MIDI

Musical Instrument Digital Interface es un protocolo estandarizado para la intercomunicación de datos musicales que implica tanto la parte de programación como la parte de diseño de dispositivo. MIDI implementa un vocabulario de eventos sonoros y una gramática que buscan describir el sonido. Para lograrlo utiliza un tamaño de dato de 10 bits de los cuales mostramos su significado a continuación:

- 1 bit de inicio (start bit): representa el inicio de un mensaje y siempre será 0
- 1bit de estatus (status bit): comunica si el mensaje es un comando a un argumento para un comando.
- 7 bits para el mensaje MIDI, esto nos ofrece un rango de valores desde 0 a 127.
- 1 bit de finalización (stop bit): comunica que el mensaje ha terminado.

Fue diseñado para el control de dispositivos musicales en tiempo real, lo que permite acceder al control del sonido de varios sintetizadores desde un mismo aparato.

MIDI nos proporciona la posibilidad de controlar sintetizadores con equipo tangible, es decir: botones, perillas, sliders y cualquier artefacto que implemente el protocolo MIDI en su construcción.

La gran ventaja del control con artefactos MIDI es que la posibilidad de lograr expresarse gestualmente incrementa.

Desafortunadamente, MIDI reducen su control a 128 niveles diferentes, lo cual se torna reducido para ciertas aplicaciones y para una descripción completa del universo sonoro.

Una buena referencia de los detalles inmersos en el protocolo MIDI es [37][pgs. 969-1016]