Pontificia Universidad Católica de Valparaíso

Facultad de Ingeniería

Escuela de Ingeniería Informática

Ingeniería Civil en Informática

MUSICALIDAD DEL MOVIMIENTO

Autor:

Ivonne Elizabeth Gatica Vivar

Informe final del Proyecto para optar al Título profesional de

Ingeniero Civil en Informática

Profesor Guía:

Silvana Roncagliolo de la Horra

Profesor Co-referente:

Broderick Crawford Labrín

(Julio, 2007)

Dedicado a mi familia, en especial a mis
Padres, por su apoyo incondicional y
constante, en esta etapa tan importante de mi vida.

Ivonne Gatica Vivar.

Agradecimientos

A Dios, mi familia, profesores y amigos quienes han sido parte fundamental de mi desarrollo profesional y a todos aquellos que me han apoyado en este camino.

Resumen

Este trabajo se centra en generar un sonido a partir del movimiento corporal de una persona, específicamente por el desplazamiento de una de las manos del ejecutante. Está basado en diversos trabajos y proyectos desarrollados a nivel mundial, en donde se relacionan de alguna manera los conceptos de movimiento y sonido.

Aquí se asocian las características físicas del movimiento y del sonido obteniendo una relación entre ellos, la que permite determinar patrones de posición en base a la frecuencia del sonido. Dado esto es posible captar mediante una cámara web los desplazamientos realizados por la mano de una persona y llevarlos al sonido de una nota en específico.

Los resultados de la propuesta han sido satisfactorios, se ha logrado el propósito de encontrar una congruencia entre el desplazamiento realizado y el sonido emitido, incluyendo las diferencias en el volumen de éste.

Abstract

The following work focuses on generating a sound from a person's corporal motion, specifically from the displacement of a hand of the performer. It is based in various works and projects developed worldwide, where the concepts of motion and sound relate to each other somehow.

Here are correlated the physical characteristics of the movement and sound, obtaining a relationship among them, one that allows determining patterns of position on the basis of the frequency of the sound. Once this is established it is possible to perceive by means of a webcam the displacements of a person's hand and translate to the sound of a note in specific.

The results of the proposal have been satisfactory; the main purpose has been to find congruence between the realized displacement and the emitted sound including the differences in the volume of it.

Glosario de Términos

Amplitud: Indica la cantidad de energía que contiene una señal sonora. No hay que confundir amplitud con volumen o potencia acústica.

Ciclo: Distancia entre el principio y el final de una onda completa.

Duración: Está relacionada con el tiempo de vibración del objeto. Por ejemplo, es posible escuchar sonidos largos, cortos, muy cortos, etc.

Fase: La fase de una onda expresa su posición relativa con respecto a otra onda.

Frecuencia: Número de ciclos (ondas completas) que se producen por unidad de tiempo. En el caso del sonido la unidad de tiempo es el segundo y la frecuencia se mide en hertzios (Ciclos/s).

Intensidad: Es la cantidad de energía acústica que contiene un sonido. La intensidad viene determinada por la potencia, que a su vez está determinada por la amplitud.

Longitud de onda: Indica el tamaño de una onda.

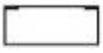
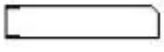
Periodo: Es el tiempo que tarda cada ciclo en repetirse.

Potencia: La potencia acústica es la cantidad de energía radiada en forma de ondas por unidad de tiempo por una fuente determinada. La potencia acústica depende de la amplitud.

Timbre: Es la cualidad que confiere al sonido los armónicos que acompañan a la frecuencia fundamental. Esta cualidad es la que permite distinguir dos sonidos, por ejemplo, entre la misma nota con igual intensidad producida por dos instrumentos musicales distintos.

Tono: Viene determinado por la frecuencia fundamental de las ondas sonoras y es lo que permite distinguir entre sonidos graves, agudos o medios.

Simbología

Símbolo		Descripción
	:	OBJETO
	:	MENSAJE
	:	NUMERO
	:	SIMBOLO
comment	:	TEXTO

I CAPÍTULO
INTRODUCCIÓN

1.1 INTRODUCCIÓN

Movimiento y Sonido están presentes en todo momento de nuestras vidas, los cuales permiten representar y demostrar emociones y estados de ánimo.

El movimiento corporal es una conducta necesaria e imprescindible de todo ser humano. En él se encuentra un lenguaje por medio del cual el hombre se expresa a través de sí mismo. El sonido ha permitido al hombre interpretar situaciones del acontecer diario, puede determinar estados de ánimo o entregar emociones.

Tanto la música en el movimiento como el movimiento corporal en la música responden simultáneamente a principios que nos permiten reflexionar sobre sus bases y consecuencias, en aspectos tan significantes como su forma y construcción, el estímulo y la reacción, el simbolismo y la comunicación o el ánimo y la expresión. Los elementos fundamentales constitutivos de la música "ritmo-melodía-armonía" sin la capacidad de movimiento, de la imaginación motora y de la premonición espacial carecerían de sentido como medio expresivo para el ser vivo consciente. Es más, sin el movimiento el ser humano no podrá percibir ni crear música, donde se configuran las vibraciones que serán transportadas por el medio.

Si se busca, en la música y el movimiento, el medio por el cual se expresan, es posible encontrar que en el caso de la música es el sonido y en el del movimiento es, para el hombre, el cuerpo. Cada percepción promueve alguna sensación. Se puede decir que las impresiones sensoriales influyen en el organismo directamente por la vía de la sensación. Se debería reflexionar sobre sus consecuencias, tanto educativas como terapéuticas.

La masificación de aparatos tecnológicos y sofisticados de producción, manipulación y amplificación audiovisual han acelerado y revivido procesos e inquietudes que surgieron

con las vanguardias modernas y que tienen que ver con la elaboración conceptual y física de nuevos instrumentos de creación. En particular, en la creación musical, durante el siglo XX se vivió una serie de revoluciones y rupturas de paradigmas que llevaron a considerar nuevas formas de componer, interpretar, oír y disfrutar música.

Considerando que tanto movimiento corporal como sonido son expresiones de comunicación utilizados por el hombre dentro de la sociedad, y que es posible, a partir de un sonido, llevarlo a una interpretación en movimiento (como es el caso de una coreografía), se ha planteado la problemática o cuestionamiento de si es posible realizar el proceso inverso, es decir, que a partir de la interpretación de un movimiento corporal sea posible encontrar un sonido congruente, considerando sólo el desplazamiento de una de las manos del ejecutante y en un lugar fijo, es decir, sin desplazamiento del ejecutante dentro de la habitación de trabajo.

Este planteamiento puede verse aplicado en la rehabilitación de niños con algún tipo de discapacidad o deficiencia física motora o no videntes, con el fin de darles la oportunidad de relacionarse, de expresar e interpretar melodías sin la necesidad de contar con un instrumento musical físico, así también puede servir de base en trabajos que impliquen la captura de todo el movimiento corporal, dando así la libertad de desplazamiento del ejecutante y por ende una mayor capacidad de expresión. Además es posible que no sólo sea utilizado en el campo médico de rehabilitación, sino que pueda ser utilizado por ejemplo en el área del cine, arte y educación, entre otras.

1.2 OBJETIVOS

1.2.1 Objetivos General

El objetivo general de este proyecto es realizar un sistema en base a un algoritmo de conversión que permita la interpretación de los movimientos corporales (de una mano) de una persona, por medio de una cámara web, en sonido (notas musicales), es decir, que el cuerpo de la persona (ejecutante) sea el instrumento a interpretar.

1.2.2 Objetivos Específicos

Los objetivos específicos son:

- Determinar un punto de congruencia entre movimiento y sonido por medio de aspectos físicos y técnicos.
- Determinar patrones para el movimiento, según área de acuerdo a la posición.
- Determinar cambios de posición (análisis del movimiento), mediante la captura del movimiento.
- Generar sonido en base a la variación de posición, logrando una sincronización entre movimiento y sonido.

1.3 METODOLOGÍA DE TRABAJO

En el presente trabajo de investigación se utiliza en primera instancia una metodología descriptiva y cualitativa. La obtención de datos de diferentes fuentes (bibliografía, papers, entrevistas, Web, etc.), permiten la descripción y caracterización tanto del sonido como del movimiento, análisis de las investigaciones realizadas, métodos aplicados, los cuales a su vez permiten el análisis de la aplicación en el plano teórico.

El estudio de los trabajos realizados en temas similares y el conocimiento de las aplicaciones existentes como herramientas para el manejo del movimiento y sonido permiten concluir la utilidad de la representación del movimiento en sonido como una herramienta de apoyo a las expresiones de comunicación y artísticas interpretativas.

A partir de estos antecedentes se obtendrá la relación entre los diferentes conceptos considerados para asociarlos con las ecuaciones correspondientes que representen la congruencia entre movimiento y sonido.

Se realizará un estudio de las alternativas existentes tanto de Hardware como de Software para encontrar aquellas que mejor se adapten al desarrollo del sistema planteado.

Por otro lado se utilizará una metodología de desarrollo, en la cual se implementará el sistema descrito con las herramientas mejor catalogadas o evaluadas para dicho efecto. Las

herramientas serán evaluadas en base a sus características y potencialidades, así como un estudio del Hardware, dado que tiene que ser compatible con las herramientas de software seleccionadas.

1.4 ORGANIZACIÓN DEL TEXTO

La organización de este trabajo de título se divide en capítulos y secciones: en el Capítulo II en la sección uno se presentan los antecedentes y definiciones del problema. En la sección dos de este capítulo se presenta el estado del arte, donde se exponen los trabajos en los que se relaciona movimiento y sonido. En la sección tres se presenta un conjunto de características físicas tanto del movimiento como del sonido, se presenta el modo de cómo se relacionan estas características, así como también se dan a conocer diversos tipos de sistemas de captura de movimiento y las características del lenguaje de programación.

En el Capítulo III se presenta el diseño del sistema propuesto, en la sección uno se expone la definición de la propuesta, el diseño de desarrollo, los requerimientos con los que debe cumplir el sistema, los procesos y la arquitectura de procesos. En la sección dos de este capítulo se presenta todo lo que es referente al diseño general del sistema, los pasos que se van a seguir y la interacción existente entre el usuario y el sistema.

En capítulo IV se documenta la implementación computacional. En la sección uno se presenta la estructura o diseño interno del sistema. En la sección dos, la implementación computacional. En la sección tres se explican y detallan las funciones utilizadas así como la relación existente entre ellas.

En el V y último capítulo se realiza la discusión de resultados a partir de las pruebas realizadas. En la sección uno se presentan los tipos de pruebas realizadas mientras que en la sección dos se exponen los resultados obtenidos y se hacen las comparaciones con los datos esperados.

Finalmente se presentan las conclusiones obtenidas a partir del trabajo de título y de los resultados de la implementación del sistema. Al final del trabajo se entrega como anexo los códigos fuentes de las funciones más relevantes utilizadas en el sistema.

II CAPÍTULO
ESTADO DEL ARTE

2.1 DESCRIPCIÓN DEL PROBLEMA

2.1.1 Antecedentes

A través de la historia, el hombre se ha comunicado y expresado mediante sonidos, gestos y movimientos. Con el paso del tiempo esta capacidad de comunicación se ha ido desarrollando con técnicas e implementos de diversas índoles, como es el caso de los instrumentos musicales.

Existe una innumerable cantidad de instrumentos, todos ellos cuentan con características que los hacen especiales y distintos unos de otros, como la forma, material, técnica de ejecución, sonido, tamaño, entre otras, pero todos ellos tienen en común que deben tener algún tipo de contacto físico con el ejecutante o persona para que puedan sonar, en otras palabras para que se escuche el tipo de sonido que pueden emitir.

El tocar un instrumento permite a las personas lograr demostrar algo, una emoción, un sentimiento o simplemente lograr un estado especial como de relajación. Existen personas que por motivos biológicos naturales o por algún tipo de accidente, no cuentan con la capacidad física motora adecuada que les permita tocar, menos interpretar un instrumento musical, siendo así privados de este medio de expresión y comunicación.

El planteamiento o propuesta es aplicable a todo tipo de persona, busca dar la posibilidad de ejecutar un instrumento musical sin la necesidad de contar con el instrumento musical en forma física, es decir, captar mediante una cámara web los movimientos (desplazamiento realizado por una de las manos) del ejecutante y transformarlos en sonidos, notas musicales, dando la oportunidad de lograr una secuencia tal que llegue a ser una melodía.

2.1.2 Definición del Problema

Dado el cambio de posición de un punto dentro de un sistema coordenado, se desea hacer un reconocimiento digital del movimiento el cual pueda ser llevado a sonido armónico.

El sonido generado debe ser lógico y coherente al movimiento realizado, por ejemplo, si se considera el batir de las alas de dos aves, una paloma y un colibrí, el batir de alas de la paloma es lento, pausado, por lo tanto el sonido asociado deberá ser acorde a esto (lento, melodioso) en cambio el batir de las alas del colibrí es rápido, fuerte, no se le puede asociar el mismo tipo de sonido que a la paloma, éste debe ser más ágil.

El problema será acotado considerando sólo movimientos corporales para la creación de las líneas melódicas (música), que serán consideradas como un medio de expresión, trabajando con la detección y análisis del movimiento de una de las manos del ejecutante.

Este proyecto puede ser aplicado en el área de desarrollo de crecimiento personal en personas discapacitadas, como es el caso de las personas mudas, este sistema de reconocimiento de movimientos podría dar la oportunidad de cantar a un mudo, ya que sus propios movimientos pueden ser llevados a sonidos.

Siguiendo en esta área, se podría intentar cambiar los sonidos por palabras con lo cual se puede llegar a otro medio de comunicación a través de los movimientos corporales.

El proyecto es una performance interactiva que simula la sonoridad del cuerpo humano a través de la captación de los movimientos en tiempo real. Este trabajo es generado por toda una línea de búsqueda y tiene que ver con la organización de los movimientos y mecanismos de captura de éstos; relacionado con lo que es el sonido y la expresividad de éste. Mientras que por otro se encuentra lo que es el desarrollo personal y medio de comunicación de niños y adultos, visualizando la utilización a niños con problemas físicos que se encuentran impedidos de realizar actividades cotidianas al igual que el resto de las personas como lo es la interpretación de un instrumento musical como medio de comunicación y expresión.

Con respecto a la propuesta, la resultante del comportamiento motor de la mano será captada por una cámara colocada a una mínima distancia del ejecutante de entre 1,5 mt. a 2 mt. de disancia. Esa información digitalizada ingresa al computador como señales electrónicas que disparan sonidos desde una base de datos ejecutable (parlantes - computador). El parámetro analizado será la zona de luz y sombra que los movimientos articulares proyecten sobre un fondo en contraste, siendo ésta analizada y procesada vía computador. El cuerpo no está intervenido directamente, es decir, no está conectado a cables ni sensores. Pero la imagen capturada del cuerpo funcionaría como parte del discurso sonoro. A diferencia de las estructuras sonoras tradicionales, aquí no se genera melodía, ya que no hay sucesión lógica de sonidos, y tampoco se puede pensar formalmente en armonías ya que tampoco hay progresiones, y todo esto se debe a que todos los sonidos suenan a la vez y dependen directamente de los movimientos generados por el ejecutante.

En la figura 2-1 se puede observar un diagrama con la secuencia o línea conductora entre el movimiento efectuado y la emisión del sonido.

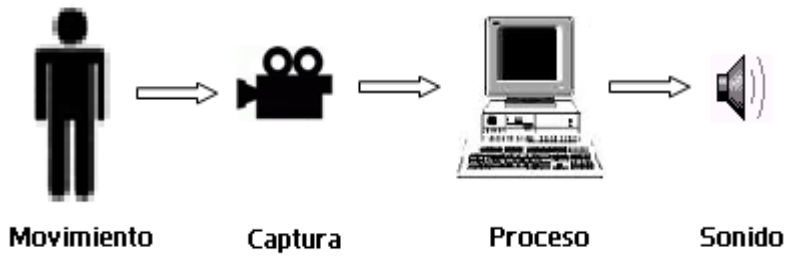


Figura 2-1 Diseño externo del sistema

2.2 ESTADO DEL ARTE

El trabajo realizado utilizando el Sonido y el Movimiento, es realmente variado y está aplicado en diversas áreas, ya sea de la informática como del desarrollo personal o como sistema de expresión de comunicación.

La inquietud de relacionar el movimiento con el sonido lleva bastante tiempo, la mayoría de los trabajos realizados están basados o inspirados en el trabajo de Lev Sergeievitch Thermen (1896 - 1993), científico nacido en Leningrado (actual San Petersburgo). Thermen

desde joven estudió con gran pasión el mundo de la música y de la electricidad. Su experiencia con el violoncelo le hizo querer inventar un instrumento de similar registro pero eliminando ciertas barreras físicas como podía ser la necesidad de frotar un arco contra las cuerdas y tener que desplazar los dedos por el mástil [1].

Este instrumento tiene un aspecto extraordinariamente peculiar, la mayor parte de los Theremines consisten en un gabinete del cual salen dos antenas, una hacia arriba y una hacia la izquierda (desde el punto de vista del ejecutante). El sonido se produce cuando el intérprete mueve las manos en las proximidades de las antenas, pero sin tocarlas. La antena izquierda (desde el punto de vista de un ejecutante diestro) es horizontal y con forma de bucle, y sirve para controlar el volumen: cuanto más cerca de la misma esté la mano izquierda, más baja el volumen, y viceversa. La antena derecha suele ser recta y en vertical, y sirve para controlar la frecuencia: cuanto más cerca esté la mano derecha de la misma, más agudo será el sonido producido. En la figura 2-2 se puede apreciar a Thermen tocando el Theremin.

Figura 2-2 Thermen tocando el Theremin

El funcionamiento del Theremin es simple, se obtienen sonidos de tono variable utilizando una corriente alterna de frecuencia también variable. Se coloca una pequeña varilla vertical

a modo de antena, así como se muestra en la figura 2-3 correspondiente al plano del funcionamiento del Theremin, lo que genera ondas electromagnéticas de muy débil intensidad alrededor de la misma. Estas ondas tienen una longitud y una frecuencia definida, la aproximación de una mano, que es un conductor eléctrico, altera la configuración del campo electromagnético que rodea a la antena, cambia su capacitancia y, por lo tanto, afecta la frecuencia de la corriente alterna generada en el instrumento. De esta manera, se produce una suerte de “toque invisible” en el espacio que circunda la antena, y, de la misma manera en que en un cello la presión de un dedo sobre la cuerda produce un sonido tanto más agudo cuanto más se acerca el dedo al puente, en el Theremin el tono se incrementa cuanto más se acerca la mano a la antena. Del mismo modo, la intensidad del tono cambia al aproximar una mano a la otra antena, de forma circular, alrededor de la cual se forman ondas electromagnéticas siguiendo el mismo principio. La aproximación de una mano produce un cambio en la intensidad de la corriente alterna que produce el tono. Así, levantando la mano por encima de la antena horizontal en forma de anillo la nota crece en volumen, mientras que bajando la mano por debajo de ella se hace más débil, hasta morir en el medio del más suave pianissimo [2].

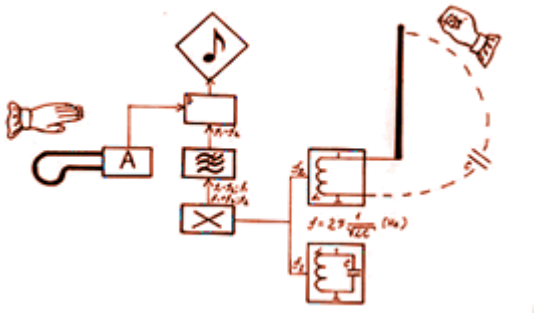


Figura 2-3 Plano de Thermen de su Theremin

El sonido del Theremin se produce según el principio del heterodino, con un oscilador de frecuencia variable superponiendo y afectando la oscilación de un oscilador de frecuencia fija. La frecuencia resultante cae dentro del rango audible para el oído humano, y esto es lo que se escucha a través de un parlante. La frecuencia del oscilador fijo es de 170 Khz. y la del variable oscila entre 168 y 170 KHz. Ello deja frecuencias audibles de 0 a 2 Khz. para ejecutar el instrumento [2].

Originalmente, su versión más primitiva fue llamada Aetherophone (se podría traducir como Eterófono), y constaba sólo de la antena de tono. Dicho diseño fue tempranamente mejorado por el inventor, añadiendo posteriormente una antena para controlar el volumen [1].

Thermen perfeccionó su instrumento de muchas maneras: construyó modelos que, en lugar de comandarse mediante el movimiento de las manos, "leían" microrreflejos en la superficie de los ojos y eran controlados por los movimientos oculares, implementó una batería de cuatro Theremines controlados por los movimientos del cuerpo de una compañía de ballet, con la cual la propia danza creaba la música (sonido) entre otras [2].

Actualmente, algunos de los modelos caseros y comercializados de Theremin disponen tan sólo de la antena que controla el tono, lo cual siendo rigurosos les convierte en realidad en un "Eterófono", y su uso frecuentemente es el de un aparato para efectos especiales más que un instrumento musical, al no poder acentuar ni separar las notas producidas. También se han llegado a producir Theremines de forma más o menos artesanal con formas de interactuar muy distintas, como por ejemplo, Theremines ópticos que miden la cantidad de luz que les llega a un sensor. También la casa Roland, en algunas de sus sucursales y vía internet, comercializa un sensor de infrarrojos llamado D-Beam, con el cual se puede controlar no sólo el tono, sino alternativamente el parámetro que se elija [1].

Para quienes disponen de dinero, hay muchos modelos de Theremines a la venta, y, para los más habilidosos, están disponibles los circuitos (bastante simples en general, como se puede apreciar en las figuras 2-4 y 2-5) para armarse su propio Theremin, también se venden en kit por correo, listos para soldar y utilizar [2].

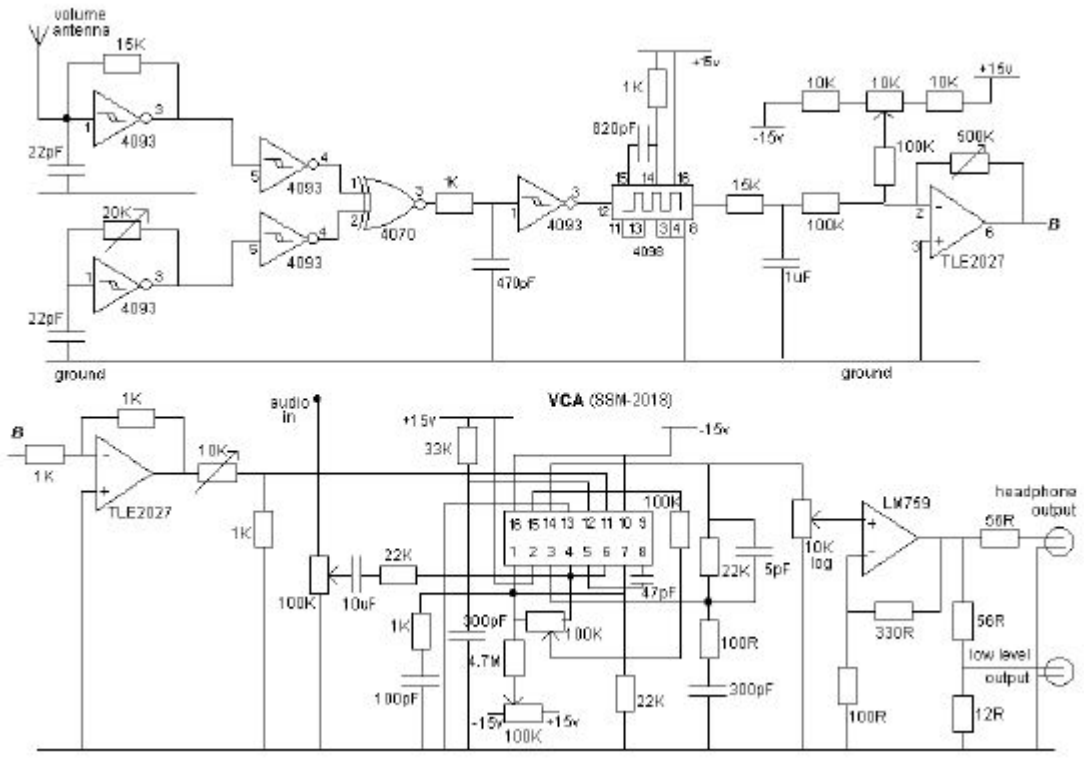


Figura 2-4 Control de circuito del volumen de un Theremin

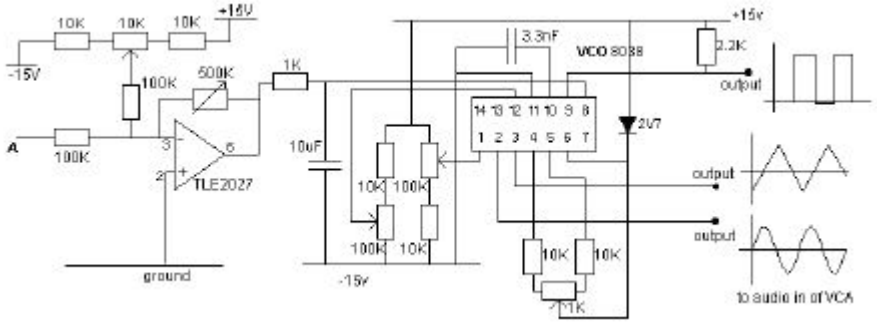


Figura 2-5 Tabla control de circuitos de un Theremin

El inquietante sonido del Theremin se usó en muchas bandas sonoras de películas durante los '50s y '60s. Prestó música de ambiente a muchas películas clásicas de Ciencia Ficción como "Ultimátum a la Tierra", "La guerra de los Mundos" o a películas de suspenso como "Recuerda". En los '60s y '70s, bandas como "Lothar and the Hand People", "The Bonzo Doo Dah Dog Band" y "Led Zeppelin" rescataron el Theremin para el público durante un

corto período de tiempo. Hoy en día, muchas bandas usan Theremines aunque son pocos los que lo usan "musicalmente".

En la actualidad es posible encontrar diversos estudios y trabajos orientados al manejo, captura y relación entre movimiento y sonido, con diversas técnicas, herramientas y finalidades.

En el 2002, en el Museo Universitario de Ciencias y Arte MUCA Campus ubicado en costado sur de la Torre de Rectoría, Ciudad Universitaria, México, se ha desarrollado "Primer Movimiento", una instalación en un cuarto oscuro iluminado por un foco central, donde el usuario se introduce y toma una placa de madera, e inmediatamente recibe un sonido producto del reflejo de la luz. Es un instrumento que traduce los movimientos en sonidos o música. Si el individuo se mueve o se le ocurre ponerse, por ejemplo, a bailar, la placa sostenida en su mano podría provocar una canción u otros sonidos pregrabados que son emitidos como notas musicales.

Primer Movimiento fue creado por Pedro Cervantes, ingeniero encargado de la programación computacional de la pieza, el músico Eduardo Meléndez y el antropólogo Cuauhtémoc Sentíes, con el fin de sorprender al usuario y explorar su capacidad expresiva al convertir sus movimientos en resonancias. Con Primer Movimiento se quiere generar en el usuario el sentimiento de arrojarlo en una alberca sonora para que se sienta rodeado y absorbido por la música, que se muestre abrumado, como si estuviera flotando en medio del sonido.

El mecanismo de la instalación funciona así: el usuario-visitante elige una o varias de las 15 placas que tienen impresas símbolos en terciopelo que son captados por una cámara de video colocada en el techo del cuarto. Este video se transfiere a un computador que analiza las imágenes cuadro por cuadro para identificar de que tarjeta se trata y en que coordenada de la habitación se encuentra. Posteriormente esa información se envía a un segundo computador que convierte los datos recopilados en los sonidos que el usuario escucha de manera tridimensional. Cada una de las placas de madera que el usuario puede elegir tiene impreso un código de audio, sin embargo los tonos, el timbre, la velocidad y la amplitud del sonido dependen de las coordenadas tridimensionales en las que se ubique la placa. La

participación del usuario es fundamental. La pieza per se es un instrumento inanimado que sólo se activa con los movimientos del visitante [3].

Otra visión de la relación y aplicación que es posible dar a estos conceptos de Música y Sonido es la planteada por Alejandra Ceriani [4], quien propone un proyecto denominado “Proyecto Hóseo”.

Hace años que la señorita Ceriani realiza trabajos en danza multimedia y en video - danza. Actualmente indaga en una propuesta interactiva: movimiento - captación de imagen y sonido en tiempo real, una performance interactiva que surge a partir del estudio articular de los movimientos.

Proyecto Hóseo es una Performance interactiva que simula la sonoridad del cuerpo humano a través de la captación de movimientos corporales en tiempo real. Tiene que ver con la organización de series generadas a partir del estudio articular de los movimientos; mínimas estructuras sumatorias que dan como resultado un acontecer perceptible de la estructuración interna del cuerpo en esa cadena de movimientos.

A la par, de la asociación con músicos electroacústicos en la búsqueda de composiciones aleatorias, es de interés la construcción de técnicas de escucha integradas al espectáculo. La disposición de los puntos de sincronización, por ejemplo, entre un movimiento o una figura y un momento sonoro, así también entre un corte del movimiento y un corte del sonido. Luego se apela a la disociación y se origina una serie de efectos en la percepción, un juego activo de superposiciones y sustituciones que se expande al espectador.

Con respecto a su propuesta, luego de realizar un seminario de postgrado (en la FBA, UNLP) vinculado a las nuevas tecnologías interactivas, encauzó la búsqueda anterior. La resultante del comportamiento articular del cuerpo es captado por una cámara colocada a una mínima distancia del performer. Esa información digitalizada ingresa al computador como señales electrónicas que disparan sonidos desde una base de datos sonoro ejecutable (teclado - computador).

El parámetro analizado es la zona de luz y sombra que los movimientos articulares proyectan sobre la piel, siendo ésta analizada y procesada vía computador. El cuerpo no

está intervenido directamente, pero la imagen del gesto de ese cuerpo funciona como parte del discurso sonoro, un continuo audible de alternancias y superposiciones de diferentes estímulos para la captación y repercusión sensorial que compromete tanto al espectador como al performer [4].

Esta tecnología aplicada abre el juego a nuevos elementos de lenguaje escénico, que interactúan sobre el cuerpo y el espacio, descubriendo otros campos de generación de movimiento, dando como resultado nuevos diseños corporales y espaciales, coreográficos y escénicos. También definen nuevos modos de experiencia del cuerpo promovidas por los sistemas de captación de movimiento corporal (sensores de captación) en tiempo real, lo cual innova en la mediatez público-espectáculo. Con estas nuevas tecnologías es posible explorar más allá del propósito estético del consumo visual de lo escénico y permiten la experimentación entre lenguajes y medios promoviendo formatos originales de exhibición.

Recientemente, se ha realizado un trabajo por Tomas Thayer [5] y Mirko Petrovich, profesores de la Universidad ARCIS para la nueva sección “Los sonidos de la música” del Museo Interactivo Mirador (MIM) de Santiago, Chile. Este es un espacio virtual denominado “Instrumento invisible”, figura 2-6, que genera respuestas multimediales por los movimientos del cuerpo. El Instrumento Invisible es un moderno espacio de experiencia virtual. A medida que los niños y niñas se mueven, producen sonidos de cuerdas, percusión e, incluso, sonidos de animales. El trabajo consiste en la captura del movimiento corporal por medio de dos cámaras de video dispuestas en una sala de 2 x 1.5 mt^2, con esto es posible identificar dos dimensiones, para saber el sonido correspondiente al movimiento ejecutado, trabajan con intensidad de luz y 8 cuadros (4 por lado), es decir, que son capaces de identificar 4 posiciones diferentes por lado del cuerpo (derecho e izquierdo), al realizar un movimiento se genera una diferencia de luz, en el espacio que se encuentre con mayor sombra se reconocerá la nueva posición y se ejecutará el sonido correspondiente a esa posición. Los sonidos generados son en base a Samplers, es decir, sonidos predeterminados ya guardados en el computador, utilizan tres tipos de sonidos: animales, batería y piano; los que van cambiando cada 30 segundos. En esta sala sólo se trabaja con los conceptos de altura y frecuencia, no es posible encontrar la relación de intensidad ni duración (tiempo).

Figura 2-6 “Instrumento invisible”

El Dr. Tom Chau, Ingeniero Biomédico, descrito por sus pares como “estrella naciente en la comunidad de investigación”, ha realzado la calidad de la vida de niños con discapacidad con su trabajo innovador en el instituto de investigación de Bloorview MacMillan. Su área de investigación es sistemas inteligentes para la rehabilitación pediátrica (sistemas automáticos del descubrimiento y reconocimiento del patrón), tecnologías que incluyen interfaces hombre-máquina, desarrollo útil de dispositivos, análisis fisiológico de señales, estrategias prostéticas de control, dinámica de la rehabilitación del movimiento patológico. Las poblaciones de interés incluyen niños y la juventud con discapacidades físicas y de desarrollo [6].

Actualmente, el Dr. Chau es el coordinador de la investigación de los sistemas inteligentes en el centro de los niños de Bloorview MacMillan, el centro pediátrico más grande de la rehabilitación de Ontario. Él recibió el grado de PhD de la universidad de Waterloo en Ingeniería de diseño de los sistemas, MASc en la ingeniería biomédica y eléctrica, y BASc en la ciencia de la ingeniería, ambas de la universidad de Toronto. Trabajó en la industria por varios años antes de ensamblar el centro en 1999 y lleva a cabo concesiones profesionales de IBM Canadá. En Bloorview MacMillan, él trabaja con un equipo interdisciplinario incluyendo terapeutas ocupacionales, terapeutas de la música, un especialista de la educación de la niñez, y un neuropsicólogo. Sus intereses actuales de la investigación giran alrededor del diseño y de la evaluación de los sistemas informáticos

inteligentes para mejorar la calidad de la vida total de niños con discapacidades físicas y de desarrollo [7].

Cabe destacar que el Dr. Chau y su equipo han creado un plan de estudios de la música que utiliza una cámara de vídeo simple para seguir el movimiento de niños con limitaciones físicas severas y para traducir ese movimiento a música. Esto permite la creación de la música por los niños que, en el pasado, podrían soñar solamente con tocar un instrumento musical.

El Instrumento Virtual de la música fue diseñado para dar a los niños que tienen deficiencias físicas una oportunidad de aprender y de tocar música. La tecnología consiste en un software, un telón y un computador. La cámara fotográfica captura los movimientos del niño y el software en el computador transforma los movimientos en notas musicales. Además el niño ve una imagen con formas circulares coloreadas, como se muestra en la figura 2-7. Obrando recíprocamente con estas formas, el niño puede tocar notas musicales y crear música que suena agradable simplemente moviendo su cuerpo en el espacio. Se ha diseñado el instrumento tal que un niño tiene la oportunidad de aprender a tocar melodías cada vez más complejas a tiempo y en la semejanza de un instrumento musical deseado, tal como un piano o una flauta. En este sentido el instrumento proporciona la motivación de la maestría [8].

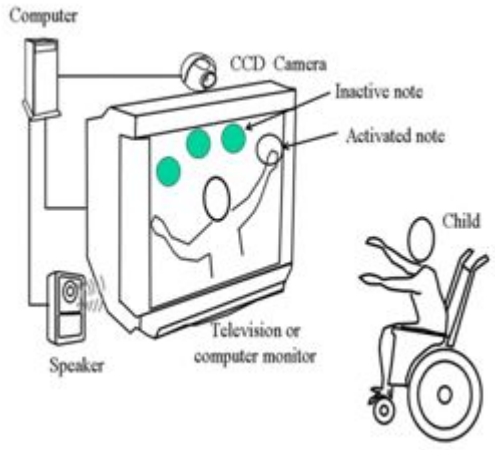


Figura 2-7 Diseño del “Instrumento Virtual”

Hace cinco años que se viene realizando una conferencia internacional sobre las nuevas interfaces para la expresión musical. Investigadores y músicos de todo el mundo se reúnen para compartir su conocimiento y trabajo basado en el nuevo diseño de interfaz musical.

La conferencia comenzó como un taller en Conference on Human Factors in Computing Systems (CHI) en 2001. Desde entonces, las conferencias internacionales se han llevado a cabo alrededor del mundo, dirigida a los grupos dedicados a la investigación sobre las nuevas interfaces para la expresión musical.

La información de todas las conferencias NIME hasta la fecha convenientemente pueden ser referenciadas usando el recurso DBLP de las páginas de la Universidad de Trier. El DBLP también proporciona entradas BibTeX para los papers publicados en NIME [9].

En su versión 2007, realizada en entre el 6 al 10 de Junio en la Universidad de Nueva York, el programa se presenta en la figura 2-8.

APPLICATIONS

Matching Parts: Inner Voice Led Control for Symbolic and Audio Accompaniment

Rage in Conjunction with the Machine

Introducing Pitch, Melody and Harmony into Robotic Musicianship

B-Keeper: A Beat-Tracker for Live Performance

Integrating HyperInstruments, Musical Robots & Machine Musicianship for North Indian...

The Wrist-Conductor

Design Issues in Interaction Modeling for Free Improvisation

fMRI-Compatible Electronic Controllers

GHI project and "Cyber Kendang"

Sensillum : an improvisational approach to composition

The Samchillian Tip Tip Tip Cheeepeeeee: A Relativistic Keyboard Instrument

Interfacing Audiences Into the Compositional Process

The ColorDex DJ Systems: A New Interface for Live Music Mixing

The WaveSaw: A Flexible Instrument for Direct Timbral Manipulation

Visual Feedback in Performer-Machine Interaction for Musical Improvisation

Effortful Interaction: Playing the Edge-trimmer

INTERFACES

Tactophonics - Your Favourite Thing Would Like to Sing

Diamair, Composing for Choir and Integral Music Controller

Interactive Spatialization and Sound Design using an Evolutionary System

EyeMusic: Performing Live Music and Multimedia Compositions with Eye Movements

The FrankenPipe: A Novel Bagpipe Controller

Developing multimodal interactive systems with EyesWeb XMI

Real-Time Feature-Based Synthesis for Live Musical Performance

jPop-E: An Assistant System for Performance Rendering of Ensemble Music

Recognition and Prediction in a Networked Music Performance System for Indian Percussion

JamiOki -PureJoy: A New Direction in Electronically-Mediated Vocal Improvisation

Overview of the design and creation of a Digital Musical Instrument controlled using...

The MIDI Pick

TOOLS

Improved Position Tracking of a 3-D Gesture-Based Musical Controller Using a Kalman Filter

The Lambent Reactive - An Audiovisual Environment for Kinesthetic Playforms

Real-Time Beat-Synchronous Audio Effects
Automatic Notation Generators
A Web-Accessible, Searchable Bowstroke Database
Gesture Control of Sounds in 3D Space
Adaptive Tuning using Theremin as Gestural Controller
VR-RoBoser: Real-Time Adaptive Sonification of Virtual Environments Based on Avatar...
A Unified Toolkit for Accessing Human Interface Devices in Pure Data and Max/MSP
Control Strategies for Navigation of Complex Sonic Spaces

Figura 2-8 Programa presentado por el NIME versión 2007

Básicamente estos trabajos y otros propuestos por diversas Universidades de Europa, han llevado a la inquietud de trabajar con Sonido y Movimiento, pero desde una perspectiva no vista hasta ahora, que es la de llevar el movimiento corporal a una representación en sonido utilizando tres dimensiones: tiempo, frecuencia y volumen.

2.3 CONTEXTO DEL PROBLEMA

2.3.1 Antecedentes Físicos

» Características y componentes del sonido

La acústica es la parte de la física y de la técnica que estudia los fenómenos que percibe el sentido del oído humano en toda su plenitud, los que se denominan ruidos o sonidos, ocupándose así de su producción y propagación, de su registro y reproducción, de la naturaleza del proceso de audición, de los instrumentos y aparatos para su medida [10].

El sonido desde el punto de vista físico es la vibración que se propaga en un medio elástico. El sonido audible por el oído humano, se refiere a la sensación detectada por el oído que producen las rápidas variaciones de presión en el aire por encima y por debajo de un valor estático. Este valor estático lo da la presión atmosférica (alrededor de 100.000 pascals), el

cual tiene pequeñas variaciones y de forma muy lenta, tal y como se puede comprobar en un barómetro [11].

La función del medio transmisor es fundamental, ya que el sonido no se propaga en el vacío. Por ello, para que exista el sonido, es necesaria una fuente de vibración mecánica y también un medio elástico (sólido, líquido o gaseoso) a través del cual se propague la perturbación. El aire es el medio transmisor más común del sonido [12].

Los instrumentos musicales ilustran perfectamente la variedad de cuerpos cuya vibración puede dar origen a un sonido. Esencialmente, en los instrumentos de viento, lo que vibra es la columna de aire contenida en el instrumento; en los instrumentos de cuerda, lo que vibran son las cuerdas del instrumento; en los instrumentos de percusión lo que vibra es un diafragma o bien un objeto metálico (unos platillos, por ejemplo).

Generalmente se utilizan cuatro cualidades subjetivas para describir un sonido musical: intensidad, tono, timbre y duración. Cada uno de estos atributos depende de uno o más parámetros físicos que pueden ser medidos, pero las cualidades que realmente caracterizan el sonido son la intensidad, su altura o tono y su timbre.

Desde el punto de vista de la intensidad, los sonidos pueden dividirse en fuertes y débiles. La intensidad depende principalmente de la presión sonora, pero también del espectro de parciales y de la duración. La intensidad de un sonido viene determinada por la amplitud del movimiento oscilatorio, subjetivamente la intensidad de un sonido corresponde a la percepción que se tiene del mismo como más o menos fuerte.

Cuanto más grande es la amplitud de la onda, más intensamente golpean las moléculas en el tímpano y más fuerte es el sonido percibido. Por ejemplo, al elevar el volumen de la radio o del televisor, lo que se hace es aumentar la intensidad del sonido.

El tono o altura de un sonido depende únicamente de su frecuencia, es decir, del número de oscilaciones por segundo. La altura de un sonido corresponde a la percepción del mismo como más grave o más agudo. Cuando mayor sea la frecuencia, más agudo será el sonido. Para un sonido puro el tono viene determinado principalmente por la frecuencia, aunque también puede cambiar con la presión y la envolvente. Los humanos somos sensibles a las

vibraciones con frecuencia comprendida entre 16 Hz y 20.000 Hz. Por debajo de 16 Hz se llaman infrasonidos y por encima, ultrasonidos. La frecuencia fundamental es la primera frecuencia de vibración de un cuerpo, es decir, hablando de una cuerda de guitarra, ésta vibra en su frecuencia más baja. A la vez, se producen armónicos, que son frecuencias más altas, es decir, vibraciones de divisiones de la cuerda (1/2, 1/4...).

El timbre es la cualidad del sonido que permite distinguir entre dos sonidos de la misma intensidad y frecuencia emitidos por dos focos sonoros diferentes. Se puede así distinguir si una nota ha sido tocada por una trompeta o un violín. Esto se debe a que todo sonido musical es un sonido complejo que puede ser considerado como una superposición de sonidos simples. De esos sonidos simples, el sonido fundamental de frecuencia v es el de mayor intensidad y va acompañado de otros sonidos de intensidad menor y de frecuencia 2v, 3v, 4v, etc. Los sonidos que acompañan al fundamental constituyen sus armónicos y de sus intensidades relativas depende el timbre.

Los armónicos generan el timbre característico de una fuente de sonido (ya sea una voz humana, un instrumento musical, etc.). Permiten diferenciar un tipo de instrumento de otro, o reconocer el timbre de la voz de una persona. Los armónicos más altos son inaudibles, y lo que da diferentes timbres a diferentes instrumentos es la amplitud y la ubicación de los primeros armónicos y los parciales. Y las diferentes trayectorias de las ondas sonoras de dos instrumentos tocando al unísono es lo que permite al oyente percibirlos como dos instrumentos separados. Por ejemplo, si dos instrumentos ejecutaran la nota Do4 (la tecla blanca central de un piano), la onda fundamental de ambos poseería la misma frecuencia (en este ejemplo 264 Hz o ciclos por segundo); pero sus timbres son diferentes porque cada uno produce una altura de armónicos diferentes.

En acústica y telecomunicaciones, un armónico de una onda es un componente sinusoidal de una señal. Su frecuencia es un múltiplo de la fundamental. La amplitud de los armónicos más altos es mucho menor que la amplitud de la fundamental y tiende a cero (por eso los armónicos por encima del quinto o sexto generalmente son inaudibles). El concepto y la existencia de armónicos tienen su fundamento matemático en la teoría de las series de Fourier. En la figura 2-9 se presenta la ecuación que representa la serie de Fourier.

$$y(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(nx) + b_n \mathrm{sen}(nx)]$$

Figura 2-9 Ecuación de forma de la serie de Fourier

» Señales digitales

Formalmente una señal senoidal se define como f(t) = cos(w · (t + p)). Siendo t el tiempo en segundos, *w* la frecuencia angular en radianes por segundo (*w* = 2 · *pi* · frecuencia en hercios) y *p* la fase de la señal en segundos. Dentro del mundo digital hay que introducir varios conceptos: como la resolución en bits de la señal a generar y la frecuencia de muestreo.

La resolución en bits de la señal no afecta a la ecuación anterior y tan solo atañe al valor devuelto por la función *cos()*. En ANSI C tanto *sin()* como *cos()* devuelven un tipo double que, dependiendo de la máquina y el sistema operativo, será un número real en coma flotante de 32, 64 ó más bits, pero, en cualquiera de los casos, acotado entre los valores -1.0 y +1.0. Por lo tanto, independientemente de la máquina siempre será posible hacer un escalado de la señal conociendo los valores límites de ésta.

En el mundo digital el "instante de tiempo" pasa a ser "muestra". La variable tiempo es una variable continua mientras que la muestra viene determinada por una variable discreta entera no negativa (0, 1, 2, 3, 4 ...). Para sustituir la variable tiempo *(t)* en la ecuación inicial simplemente se hace *t* = *n* / fm. Siendo *n* el índice de la muestra (0, 1, 2, 3 ...) y fm la frecuencia de muestreo para el dispositivo, en muestras por segundo (la calidad CD corresponde a 44100 muestras por segundo). Así, cuando *n* = 0, *t* = 0 segundos mientras que si n = fm, t = 1 segundo. La ecuación inicial queda así: f(n) = cos(w * ((n / fm) + p)), donde *w* es la frecuencia en radianes por segundo de la señal a generar, *n* es la muestra a generar (0, 1, 2 ...), fm es la frecuencia de muestreo en muestras por segundo y *p* es la fase de la señal en segundos.

La figura 2-10 muestra el código en lenguaje C que genera un tono audible de 440 Hz con una frecuencia de muestreo de 44.1 KHz y muestras de 16 bits con signo (calidad CD).

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define F_SENIAL 440   /* un LA en la 4ª octava del piano */
#define F_MUEST  44100 /* frecuencia de muetreo: 44.1 KHz */
#define TIEMPO   2     /* segundos de reproducción */
#define MUESTRAS_TOTALES (TIEMPO * F_MUEST)
signed short int salida[MUESTRAS_TOTALES];
int main(void) {
   int i;
   float v;
   for (i = 0; i < MUESTRAS_TOTALES; i++) {
      v = cos(2 * M_PI * ((float)i / F_MUEST) * F_SENIAL);
      salida[i] = (signed short int) rint(v * 32767);
   }
   return 0;
}
```

Figura 2-10 Código sencillo en C para generar un coseno audible

Otro aspecto importante a la hora de abordar la síntesis es el de obtener la frecuencia en hercios (Hz) de una señal a partir de su nota musical. Según está internacionalmente dispuesto, la frecuencia de una nota La en la cuarta octava de un piano debe ser de 440 Hz. Al tomar en cuenta, además, que por cada octava que se sube, se dobla la frecuencia y por cada octava que se baja, se divide la frecuencia a la mitad, es posible calcular fácilmente la frecuencia en hercios que debe tener una nota cualquiera con sólo conocer su nombre (Do, #Fa, etc) y su octava.

Según Fourier, cualquier señal puede ser descrita perfectamente mediante una suma de cosenos de diferente fase, amplitud y frecuencia. Para sintetizar un sonido mediante síntesis aditiva se realiza una suma algebraica de varias señales senoidales (cosenos o senos) de diferente frecuencia, fase y amplitud. Normalmente, un sonido musical se caracteriza por una frecuencia fundamental y una serie de frecuencias denominadas armónicos. La frecuencia de cada armónico se define como un múltiplo de la frecuencia fundamental. De esta manera se obtienen todas las frecuencias de los cosenos que componen una señal con sólo dar su frecuencia fundamental (el tono que se percibe). En la realidad, a la hora de percibir el sonido de un instrumento musical entran en juego otros factores, como las resonancias entre cuerdas (en el caso de los instrumentos de cuerda) y otros sonidos no musicales (por ejemplo, en un piano, el ruido que producen los martillos al golpear las cuerdas o la presencia de frecuencias no múltiplos de la frecuencia fundamental).

En la figura 2-11 se presenta el código en lenguaje C que genera un timbre mediante síntesis aditiva. El número de componentes frecuenciales está limitado a 3 aunque puede ser variado fácilmente.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define F_MUEST      44100  /* frecuencia de muestreo: 44.1 KHz */
#define F_SENIAL     440    /* un LA en la 4ª octava del piano */
#define TIEMPO       2      /* 2 segundos de reproducción */
#define NUM_ARMONICOS 3     /* nÃºmero de componentes de la seÃ±al a generar */
#define MUESTRAS_TOTALES  (TIEMPO * F_MUEST)
int armonico[NUM_ARMONICOS] = {1, 2, 4};
float amplitud[NUM_ARMONICOS] = {1.0, 0.4, 0.3};
float fase[NUM_ARMONICOS] = {0.0, 0.0, 0.1};
signed short int salida[MUESTRAS_TOTALES];
int main(void) {
   int i, a;
   float v;
   for (i = 0; i < MUESTRAS_TOTALES; i++) {
      v = 0;
         for (a = 0; a < NUM_ARMONICOS; a++) {
         v += amplitud[a] * cos(2 * M_PI * ((float)i / F_MUEST + fase[a]) *
                (F_SENIAL * armonico[a]));
      }
      /* limitamos la señal por si hay desbordamiento */
      if (v > 1.0)
         v = 1.0;
      else if (v < -1.0)
         v = -1.0;
      /* escribimos en la salida */
      salida[i] = (signed short int) rint(v * 32767);    }
   return 0;}
```

Figura 2-11 Ejemplo de síntesis aditiva en lenguaje C

Este tipo de síntesis se utiliza más a menudo junto con métodos de análisis para emular sonidos previamente grabados. Para el análisis de las componentes frecuenciales de una señal se utiliza la transformada de Fourier; dicha transformada permite calcular a partir de una señal cualquiera la magnitud y fase de cada una de las componentes frecuenciales que posee (permite obtener una imagen de la señal en el dominio de la frecuencia, lo que se denomina "ecuación de análisis"). Se trata de una transformada integral reversible ya que permite mediante otra transformación pasar del dominio de la frecuencia al dominio del tiempo (restaurar la señal original) mediante lo que se denomina antitransformada o transformada inversa (ecuación de síntesis). Es esta antitransformada la que realiza una síntesis aditiva como se puede apreciar en la figura 2-12.

$$X(\omega) = \int_{-\infty}^{+\infty} x(t) e^{-j\omega t} dt \qquad \text{Ecuación de análisis}$$

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} X(\omega) e^{j\omega t} d\omega \qquad \text{Ecuación de síntesis}$$

$$e^{-j\omega t} = \cos(\omega t) - j\,\text{sen}(\omega t)$$

$$e^{j\omega t} = \cos(\omega t) + j\,\text{sen}(\omega t)$$

Figura 2-12 Ecuación de la transformada de Fourier

» Características y componentes del movimiento

Cinemática es la parte de la mecánica que estudia los movimientos de los cuerpos sin importar las causas que los producen. El movimiento es un fenómeno físico que se define como todo cambio de posición, que experimentan los cuerpos de un sistema, o conjunto, en el espacio con respecto a ellos mismos o a otro cuerpo que sirve de referencia. En un movimiento intervienen básicamente tres elementos: la distancia o longitud que recorre el móvil, la trayectoria que sigue y el tiempo que tarda en recorrerla.

Las distancias se miden a partir del punto en que el móvil inicia el movimiento o a partir de algún punto de referencia considerado arbitrariamente como origen de las distancias. Si al medir una distancia se especifica tanto la magnitud como su dirección, entonces se está especificando una distancia vectorial medida en una dirección particular entre dos puntos y se le conoce como desplazamiento.

Todo cuerpo en movimiento describe una trayectoria la que puede ser línea recta o curva. La trayectoria es el conjunto de todas las posiciones por las que pasa un cuerpo en movimiento, la trayectoria de un cuerpo siempre será una línea continua [13].

En el lenguaje ordinario los términos distancia y desplazamiento se utilizan como sinónimos, aunque en realidad tienen un significado diferente. La distancia recorrida por un móvil es la longitud de su trayectoria y se trata de una magnitud escalar. En cambio el desplazamiento efectuado es una magnitud vectorial. El vector de desplazamiento representado en la figura 2-13 tiene su origen en la posición inicial, su extremo en la posición final y su módulo es la distancia en línea recta entre la posición inicial y la final.

Figura 2-13 Diferencia entre desplazamiento y trayectoria

Otros conceptos son rapidez y velocidad, estas son dos magnitudes relacionadas con el movimiento que tienen significados y definiciones diferentes. La rapidez, magnitud escalar, es la relación entre la distancia recorrida y el tiempo empleado. La rapidez no tiene en cuenta la dirección. La velocidad es una magnitud vectorial que sí tiene en cuenta la dirección, relaciona el desplazamiento o cambio de la posición con el tiempo [14].

2.3.2 Relación de conceptos a utilizar

Dados los conceptos y descripciones presentados tanto para las características del movimiento como del sonido es que se considerarán los siguientes términos para el desarrollo del proyecto:

Frecuencia: será utilizada para dar la nota en cada posición. Se considerarán rangos de acuerdo a la frecuencia respectiva de cada nota. Las frecuencias serán representadas en un plano de coordenadas cartesianas teniendo un punto de referencia (0,0), donde se encuentra ausencia de sonido y movimiento. La frecuencia será representada por la coordenada “y” del plano cartesiano, mientras mayor sea el valor de la coordenada “y”, mayor será el valor de la frecuencia por ende se emitirá una nota más aguda.

Tiempo de duración: el tiempo que dure la emisión del sonido estará dada por el tiempo que se encuentre el punto observado en una posición en particular.

Intensidad: esta característica estará representada por una similitud existente con lo que es el volumen. Estará representada por la coordenada “z”, es decir, mientras se vaya acercando o alejando el punto observado del punto observador (cámara web), se irá aumentando o disminuyendo de volumen, según corresponda.

Timbre: se considerará inicialmente la tímbrica del piano, para acotar la cantidad de sonidos a emitir se utilizará un máximo de 8 notas correspondientes a una octava, estas notas están dentro de los 259,5 y 546,01 Hz.

Rapidez: la rapidez con la que cambiará la tonalidad o frecuencia de una nota estará asociada con la rapidez del movimiento realizado, así como del tiempo que permanece el punto observado en una posición en particular.

2.3.3 Sistemas de captura

La captura de movimiento es la creación de una representación 3D a partir de una actuación en vivo, en contraste con la animación que es creada mediante el proceso de keyframing*.

La captura de movimiento puede suponer un ahorro de tiempo sustancial en un proyecto de animación, puede facilitar enormemente el proceso de animación, especialmente cuando se trata de crear de manera realista el comportamiento de un personaje, además con la captura de movimiento se pueden captar los detalles y matices del movimiento de un actor y trasladarlos a un personaje 3D.

La captura de movimiento tiene dos aplicaciones principales, la más usual es la captura de datos para su uso en programas de animación 3D o de análisis de movimiento. El término “tiempo real” describe la otra aplicación, en la cual un sistema de captura de movimiento se usa para trasladar los movimientos de un actor humano a un personaje animado en tiempo real. De este modo puede usarse para programas de televisión en vivo, para actuaciones en eventos, ferias, etc. El tiempo real por tanto no solo permite el uso del sistema frente al público, sino que da más flexibilidad a las capturas que serán posteriormente tratadas en una aplicación 3D [15].

La captura de movimiento es utilizada intensivamente en el sector de creación de video juegos, debido a que se está buscando mayor calidad y semejanza a la realidad. La captura de movimiento permite un ahorro de tiempo y por ende una optimización de éste.

Generalmente se usan dos tipos principales de animación de personajes 3D: acción en tiempo real dentro del propio video juego y secuencias de introducción y de transición

* Técnica utilizada en animaciones para permitir la definición de escenas de transición entre secuencias de iniciales y finales.

La captura de movimiento se está haciendo cada día más común entre los programas de TV en vivo. Puede usarse para situar un personaje virtual en un escenario real, o para situar actores reales en un escenario virtual con actores virtuales, o personajes virtuales en una escena virtual.

El cine es otro sector que frecuentemente utiliza la captura de movimiento. La animación basada en captura es esencial para crear personajes que se muevan de manera realista en situaciones que sería imposible o peligroso usar actores reales [15].

También es posible encontrar la utilización de la captura de movimiento en análisis biomecánicos con fines de rehabilitación y para ayudar a diseñar prótesis de manera eficaz. En educación, la captura de movimiento puede suponer una diferencia cualitativa de un animador permitiendo al alumno ampliar su espectro de posibilidades.

La captura de movimiento es indispensable para las aplicaciones de entrenamiento basadas en realidad produciendo una inmersión mucho mejor que si se usara un joystick u otro dispositivo.

Para realizar la captura de movimiento existen distintos tipos de sistemas de hardware, así como software para su procesamiento.

Dentro de lo que son los Softwares, es posible destacar: Shape Animador, Star Trak, Humodan, Natural motion, Mocap, Motion Captor, entre de otros.

En lo que respecta a Hardware, hay distintas técnicas o implementos, como son los sistemas ópticos, de LEDs infrarrojos, mecánicos y magnéticos, los más utilizados son los sistemas de captura magnéticos, basados en campos magnéticos y antenas como sensores y los sistemas de captura ópticos que utilizan cámaras para realizar el seguimiento de unas marcas situadas en el cuerpo del actor. En la figura 2-14 se muestra un diagrama del funcionamiento de un sistema de captura óptico de tres cámaras y sensores.

Los sistemas de captura magnéticos presentan distorsiones en presencia de campos magnéticos y trabajan con un número limitado de marcas. Sin embargo, presentan menos problemas a la hora de identificar las marcas y pueden ser utilizados casi sin procesamiento posterior, por lo que son utilizables en aplicaciones interactivas. Los sistemas de captura ópticos no presentan limitación en cuanto al número de marcas y suelen emplearse en la captura de movimientos complejos.

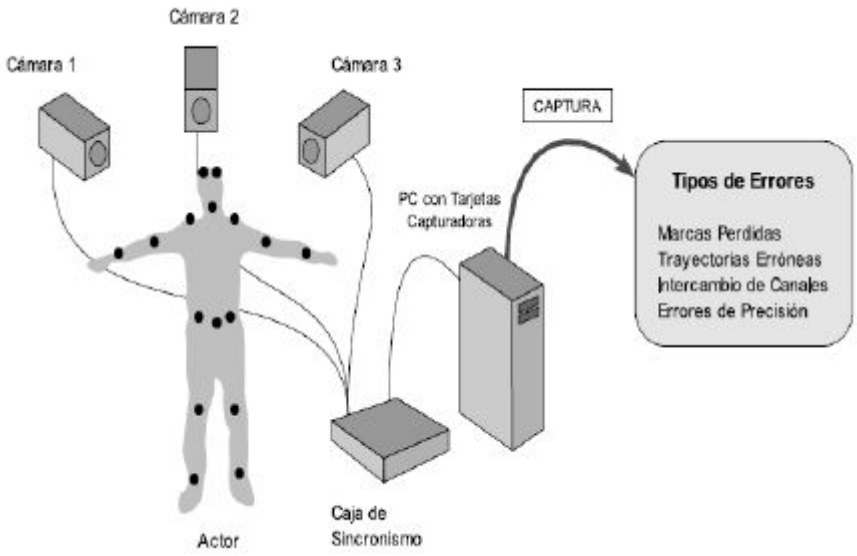


Figura 2-14 Sistema de captura óptico

Existen dos tecnologías principales en los sistemas de captura ópticos: reflectivos (sistema pasivo) y sistemas basados en LEDs (sistemas activos, basados en diodos que emiten luz).

Los sistemas de captura ópticos reflectivos utilizan filtros de infrarrojos montados en el lente de la cámara para captar la luz infrarroja reflejada por los marcadores. Por el contrario los sistemas ópticos activos usan LEDs que se activan a intervalos de tiempo regulares y miden la intensidad de la luz emitida por los LEDs.

La captura del movimiento realizada con sistemas ópticos presenta una serie de problemas. Un tipo de problema se debe a la oclusión de marcas por el propio cuerpo del actor u otros elementos de la captura. Si el actor durante la captura oculta alguna marca en zonas donde al menos dos cámaras puedan verla simultáneamente, el sistema generará errores puntuales

en los frames donde la marca no esté visible. Otro tipo de error muy común es la mala identificación de marcas por parte del sistema de reconstrucción 3D.

Partiendo del conjunto de imágenes 2D capturadas, el sistema en algunas ocasiones no asigna las coordenadas 3D reales a una marca, volviendo a tomar de nuevo la trayectoria correcta unos frames más adelante. Esto provoca un efecto de trayectorias erróneas en ciertos fragmentos de la captura.

Dos marcas que estén próximas en un momento determinado, pueden intercambiar sus valores en un intervalo de tiempo, dándose lugar el tercer tipo de errores de intercambio de canales. Estos errores del sistema se deben a una identificación incorrecta de las marcas, y son difíciles de localizar automáticamente de forma general.

Por último, existe un tipo de error por falta de precisión que provoca un efecto de trayectorias temblorosas en la captura de forma global. Es, en parte, debido a la curvatura del lente, la mala fijación de las marcas reflectantes a las articulaciones del actor y a la pérdida de exactitud en la identificación de puntos 3D.

Estos tipos de errores es posible apreciarlos en las figuras 2-15 a 2-17 en donde se muestra la pérdida de puntos en las secuencias captadas.

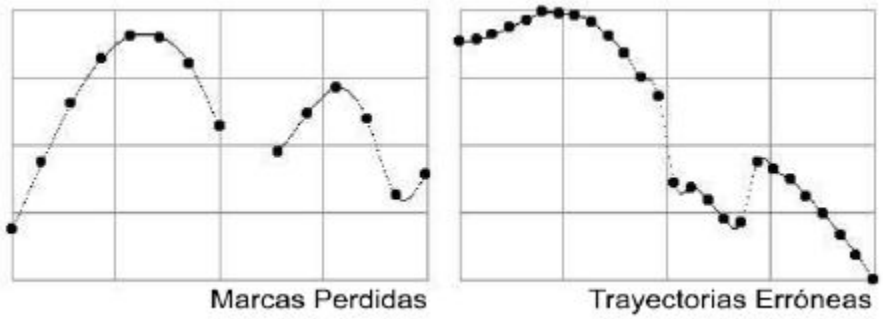


Figura 2-15 Marcas Perdidas y Trayectorias Erróneas

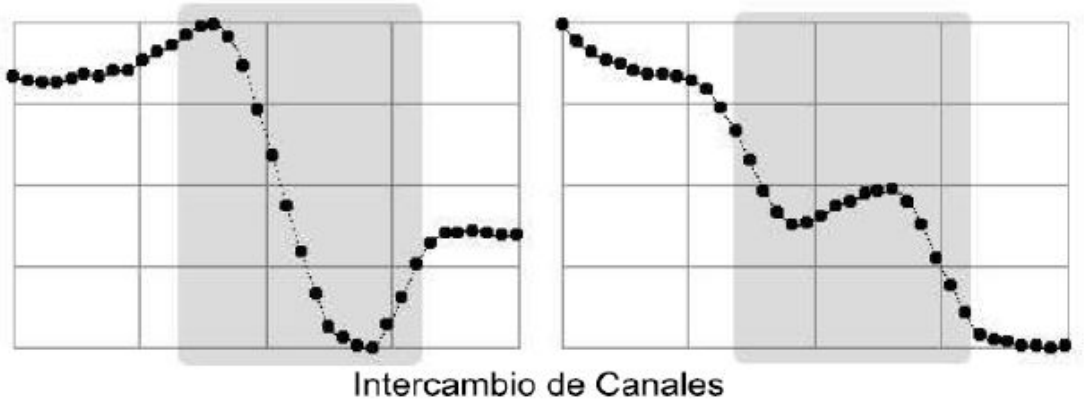


Figura 2-16 Intercambio de Canales

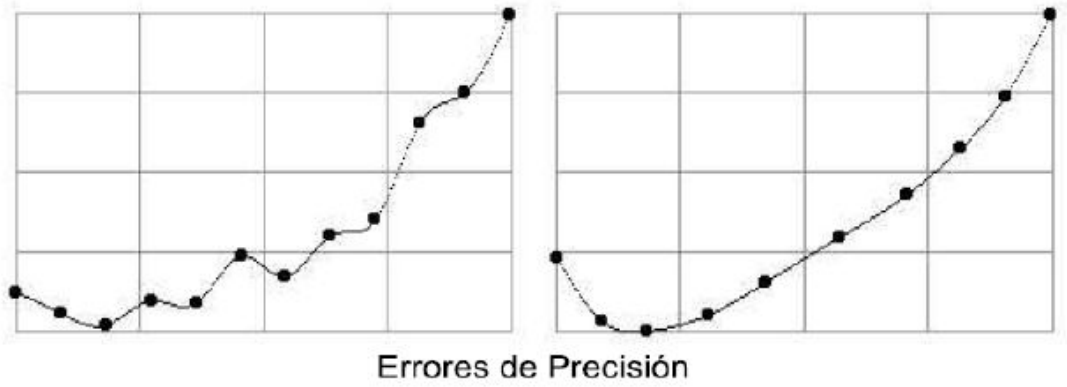


Figura 2-17 Errores de Precisión

La detección de estos tipos de errores suele realizarse de forma manual. Un experto reproduce los datos captados y analiza los diferentes canales de entrada, identificando los problemas que han ocurrido en la fase de captura. Hecho esto, se procede a una fase de limpieza de los datos.

La limpieza de los datos suele realizarse bien de forma semiautomática, empleando interpolación mediante curvas [16], o bien manualmente, situando las marcas erróneas en una posición válida para cada frame.

Esta identificación de errores y su posterior limpieza es una labor muy costosa en tiempo si se realiza de forma manual o con procesos semiautomáticos, llegando a ser el verdadero cuello de botella en los sistemas de captura del movimiento.

2.3.4 Lenguajes de programación

Pure Data, es un entorno de programación visual orientada a objetos que permite ser extendido de manera muy flexible. Pure Data fue utilizado originalmente para el modelado de audio a tiempo real. Programadores alrededor del mundo, Europa principalmente, incorporan o realizan sus aportes al desarrollo de esta herramienta y a la fecha es capaz de tratar diversos formatos de audio y video inclusive a través de Internet y redes de comunicaciones en general. De manera gráfica se sintetiza en objetos que representan funciones capaces de recibir diferentes argumentos como entrada (input) de control o datos a ser procesados. Se pueden agrupar e interconectar objetos (patches) para crear diferentes tipos de interfaces que gestionen la data/media a tratar. En la figura 2-18 es posible observar estos objetos representados por “cajas” siendo las líneas conectoras los argumentos de entrada o salida.

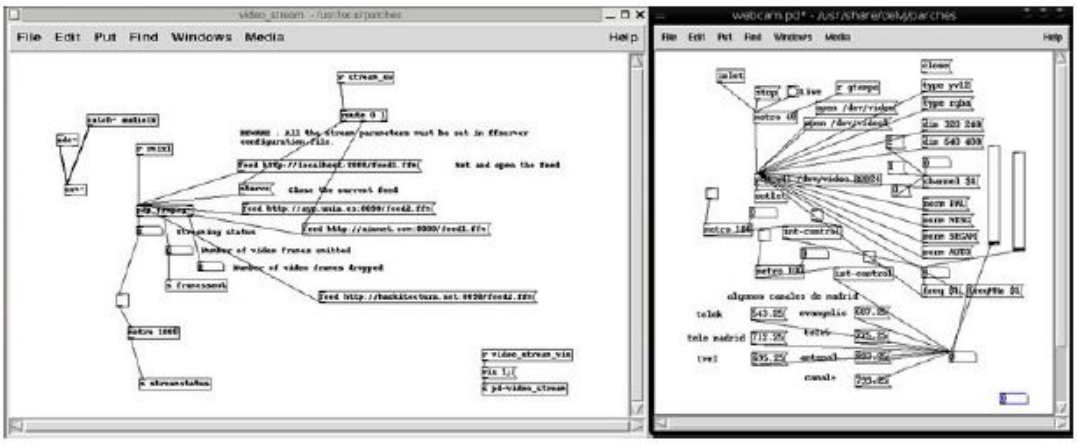


Figura 2-18 Programa en Pure Data

Los objetos encierran (aunque disponibles en el código fuente) al usuario o programador de Pure Data, quizás decenas o centenas de miles de líneas de código, escritas por otros desarrolladores de software libre cumpliendo con requisitos de alguna otra aplicación. Entra también el concepto de las librerías, colección de funciones llamadas ante requerimientos de una o más aplicaciones [17].

La arquitectura de Pure Data está basada en un corazón y una serie de módulos externos, con distintas funcionalidades, siendo éstos los siguientes:

* PDP- Sistema de Gráficos para el Pure Data.
* PIDIP - Objetos de Video adicionales para el PDP.
* GEM- Módulo de 3D (openGL).
* 3DP- Módulo de 3D (openGL), basado en PDP.

La modalidad que propone esta metodología de desarrollo resulta de interés a variados perfiles de programadores o artistas, dada la versatilidad que permite a fin de crear, procesar, tratar data/media. El crecimiento por tanto está siendo exponencial [18].

En octubre del 2003 se celebró la primera convención internacional de Pure Data, en el IEM de Graz, la institución que alberga la lista de correo de este software. En palabras de varios asistentes, la mejor de las presentaciones fue la de Tom Schouten, autor del PDP y que actualmente trabaja en una extraña posible mutación del sistema.

El título de esta charla "Pure Data: the languaje to hack the media", sintetiza brillantemente de lo que se está hablando: una herramienta libre de la era cibernética con la que cualquiera, sólo con algo de hardware doméstico y conocimiento, puede por ejemplo capturar la señal de televisión, manipularla con toda clase de efectos y máscaras; introducirle texto desde cualquier base de datos dinámica de Internet y emitir el resultado de vuelta.

Pure Data es una herramienta que permite control total de los píxeles, y su operado remoto. Así también se permite la implementación de la entrada de datos desde sensores en una mira de robots ecológicos. El límite es la imaginación y el tiempo disponible de cada uno [18].

III CAPÍTULO
DISEÑO DEL SISTEMA

3.1 SISTEMA PROPUESTO

3.1.1 Definición

El sistema propuesto consiste en la observación del desplazamiento de un punto con respecto a un punto de referencia y captura del movimiento efectuado por una de las manos del ejecutante. Este movimiento será capturado por una cámara web, la que se ubicará frente al ejecutante, determinando así la trayectoria descrita por éste hasta una nueva coordenada.

Al tener estos datos (valores), de desplazamiento, trayectoria y velocidad, se propone llevarlo a su equivalente en datos correspondientes al sonido; datos como duración, timbre, tono y frecuencia.

Con esta relación numérica determinada, es posible asignarle un cierto sonido a un movimiento, logrando así una congruencia entre el movimiento y el sonido.

Al realizar esto, hay que considerar que el sonido debe tener una coherencia con el movimiento ejecutado, es decir, el sonido debiera escucharse inmediatamente realizado el movimiento y debiera durar aproximadamente lo que demoró el movimiento en llegar a su fin.

Una vez que se hayan capturado los movimientos, se analizará cuál fue el cambio de posición, es decir, cuál es la nueva ubicación de las manos con respecto a un punto de origen y cuál fue la trayectoria descrita por éste.

Al tener la nueva ubicación y trayectoria se determinará cuál es el sonido que corresponde, asignando su respectivo tono, frecuencia, volumen y duración (tiempo sonoro).

El sonido emitido debe ser congruente con el movimiento realizado, respetando los tiempos e intensidades. El sistema debe funcionar en tiempo real, es decir, que el sonido debe escucharse a la par se realiza el movimiento.

También es necesario determinar patrones de posición debido a que se trabajará con sonidos puros, es decir, frecuencias determinadas, correspondientes a la tímbrica del piano, estos patrones representarán un área de dispersión y a cada una de éstas le corresponderá un sonido (nota musical) en particular, ya que hay que considerar que no necesariamente se va a poder ejecutar un movimiento exactamente igual al anterior.

La propuesta es tomar como base el trabajo solicitado por el MIM y el realizado por el Dr. Chau en Toronto, Canadá. Cambiando los samplers por frecuencias, aumentando los cuadros de posición, agregando una tercera dimensión correspondiente al volumen y trabajando con más de un tipo de instrumento.

En la figura 3-1 se representan las fases de la propuesta, básicamente consta de cuatro que son:

1.- Captura de movimiento realizado.

2.- Reconocimiento del movimiento.

3.- Asociación del movimiento con el sonido.

4.- Escucha del sonido correspondiente al movimiento realizado.

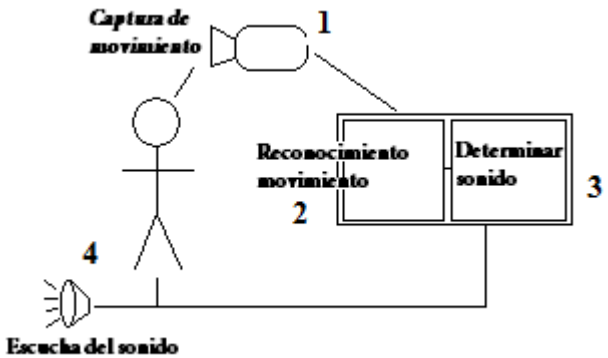


Figura 3-1 Fases de la propuesta

3.1.2 Diseño de desarrollo

Como diseño de desarrollo para obtener los resultados deseados se han planteado una serie de pasos o etapas a seguir:

Organización de la información recopilada, determinar herramientas a utilizar y relacionar conceptos de sonido y movimiento.

Después de analizar todos los conceptos recopilados y llegar a una conclusión de cómo es posible relacionar mediante el computador características del movimiento con características del sonido, (tomando las cuatro características más representativas de éste que son: intensidad, tono, timbre y duración), es que se ha tenido que tener una serie de consideraciones para ponerlas en práctica.

Determinar el área de trabajo para seccionar en base a ella la cantidad de notas a utilizar. Se utilizará 1 octava de la escala musical, es decir, 8 notas (con tímbrica del piano, las que podrían ser eventualmente modificadas a las de otro instrumento).

Otra de las razones por las que se escogió utilizar esta escala de notas es que a diferencia de otros instrumentos, en el piano es posible captar solo un sonido, esto quiere decir que, si bien conceptualmente o visualmente un Sol sostenido es igual a un La bemol (misma tecla en el piano) en algunos instrumentos esta nota sufre una leve variación, por lo tanto para no entrar en mayor complejidad, se ha decidido acotarlo a un mismo sonido.

En la práctica la cantidad de notas utilizadas es de 8, dado el área de trabajo utilizada y considerando que la persona al estar quieta (sin desplazamiento corporal, sólo el de las manos) no podrá abarcar un área superior a la mencionada, la persona o ejecutante, se encontrará sentada, el lugar donde se encuentre el ejecutante debe contar con un fondo de una sola tonalidad que haga contraste con la persona, y la cantidad de luz debe ser pareja, es decir, que en toda la habitación o lugar donde se esté utilizando el sistema debe contar con la misma intensidad de luz para no afectar los datos capturados. Cada nota tiene una frecuencia inicial (por denominarlo de cierto modo) a la que se le denomina “nota

fundamental", por cada aumento de octava, existe la duplicación en el valor de la frecuencia de la nota.

Como se ha especificado que se tomarán sonidos puros, es que se ha debido hacer una nueva tabla de valores con las frecuencias (Figura 3-2 muestra una tabla con los valores de las frecuencias de las notas de un piano) ya que se ha decidido trabajar con rangos de valores a modo de patrones de posición. Si la frecuencia de un La_4 es 440 Hz, la frecuencia de un $\#La_4$ es de 466 Hz y la de un Si_4 es 494 Hz, entonces, todos los puntos que se encuentren entre los 453 y 480 Hz., serán representados por el sonido correspondiente al $\#La_4$. Esto se ha determinado aplicando una fórmula muy sencilla que es N = (N1+N2)/2, donde N es la frecuencia de la nota, y N1 con N2 son notas continuas.

	Oc. 0	**Oc. 1**	**Oc. 2**	**Oc. 3**	**Oc. 4**	**Oc. 5**	**Oc. 6**	**Oc. 7**	**Oc. 8**
Do		32,70	65,41	130,81	261,63	523,25	1046,50	2093,00	4186,01
Do#		34,65	69,30	138,59	277,18	554,37	1108,73	2217,46	
Re		36,71	73,42	146,83	293,66	587,33	1174,66	2349,32	
Re#		38,89	77,78	155,56	311,13	622,25	1244,51	2489,02	
Mi		41,20	82,41	164,81	329,63	659,26	1318,51	2637,02	
Fa		43,65	87,31	174,61	349,23	698,46	1396,91	2793,83	
Fa#		46,25	92,50	185,00	369,99	739,99	1479,98	2959,96	
Sol		49,00	98,00	196,00	392,00	783,99	1567,98	3135,96	
Sol#		51,91	103,83	207,65	415,30	830,61	1661,22	3322,44	
La	27,50	55,00	110,00	220,00	440,00	880,00	1760,00	3520,00	

La#	29,14	58,27	116,54	233,08	466,16	932,33	1864,66	3729,31	
Si	30,87	61,74	123,47	246,94	493,88	987,77	1975,53	3951,07	

Figura 3-2 Tabla de frecuencias

Captura del movimiento con una cámara de video. Esta cámara será ubicada delante del ejecutante, a una distancia de entre 1,5 a 2 mts. Será posible captar los cambios de posición realizado por las manos en los sentidos horizontal y vertical. Una variación de posición en el sentido vertical, determinará el cambio de tono, es decir la frecuencia de la nota o sonido a emitir, mientras que un cambio en el sentido horizontal representará la mantención del tono emitido.

Cada vez que el desplazamiento se realiza en el eje “y” del plano, se obtiene una variación en el sonido que se escucha, a medida que mayor es el valor de la coordenada “y”, mayor es la tonalidad de la nota, es decir, produce un sonido más agudo. En cambio si el movimiento se realiza en la coordenada “x” del plano, sin sufrir variación en el eje “y”, lo que se obtiene es la continuidad de la nota.

Vale destacar que a lo que se llama punto centro de referencia, no necesariamente es el centro tal como se conoce (centro de los cuatro cuadrantes), para el desarrollo del proyecto sólo se utiliza el primer cuadrante, debido a que no se tienen frecuencias ni tiempos negativos, pero el centro sigue siendo el punto (0,0).

Reconocimiento del movimiento efectuado, es decir, codificar la información extraída de la cámara de video en el computador, para luego estos datos poder asociarlos con el sonido que se quiere representar. Hay que preocuparse del tipo de cámara que se va a utilizar, ya que hay que tener en cuenta que existe un cierto retardo en el traspaso de la imagen desde la cámara hacia el computador.

Una vez que se cuenta ya con la información de la imagen, hay que relacionarla con el sonido, para que este pueda ser emitido por los parlantes del computador o por algún otro dispositivo externo.

Reconocimiento del movimiento efectuado, traspasar la información de la cámara al computador, analizar el tipo de movimiento para asociarle la intensidad correspondiente. Este movimiento será considerado como la coordenada en el eje “z”.

Si un punto va en movimiento en el eje “x” (sonando una nota mantenida) y cambia su posición en la coordenada “z”, el sonido se sentirá más o menos fuerte dependiendo de la proximidad que tenga al punto observador (cámara web), mientras más cerca del punto centro se encuentre o más alejado del punto observador, menor será el volumen de la nota emitida.

Sincronización del movimiento captado por la cámara, es decir, relacionar las tres dimensiones que se han logrado capturar, pero tomando en cuenta el volumen que debería tener el tono que se ha determinado. Hay que hacer esta sincronización ya que se quiere dar una intención al sonido emitido en base al movimiento realizado, es decir, que se espera escuchar al mismo tiempo tono y volumen de éste.

Relacionar el movimiento obtenido con el sonido a emitir. Escucha del sonido.

3.1.3 Requerimientos

El programa debe ser capaz de detectar el cambio de posición, un movimiento corporal, específicamente el cambio de posición de las manos del ejecutante, (eventualmente niños con algún tipo de discapacidad física motora o no videntes).

Luego este movimiento debe ser procesado y analizado, identificando la zona en la que se encuentra la mano, sabiendo esta zona, asignarle un sonido. Este sonido debe ser el correspondiente a una nota musical, no necesariamente a un instrumento musical en específico.

El análisis del movimiento debe reflejar tanto el movimiento vertical como horizontal además de reflejar el de profundidad, es decir, si se considera trabajar en un eje coordenado, sería determinar los ejes “x”, “y” y “z”.

En el sonido emitido se debe ver reflejado el concepto de intensidad, es decir, que de acuerdo al movimiento realizado se escuche más o menos fuerte el sonido emitido.

3.1.4 Procesos

Con los requerimientos, estudios y antecedentes que se han presentado es posible establecer un conjunto de procesos principales que el sistema deberá considerar, estos son: *captura de imagen*, proceso mediante el cual es capturada la imagen del ejecutante, *determinar centro de gravedad*, proceso que permite detectar un movimiento por parte del ejecutante, *detectar cambio de posición*, proceso encargado de la identificación de la nueva posición del punto observado, *asignar sonido*, proceso en el cual se asocia un sonido al movimiento detectado, *generación de sonido*, proceso encargado de la emisión del sonido correspondiente.

3.1.5 Arquitectura de procesos

En esta instancia se obtiene un diagrama de flujo tentativo para los procesos definidos anteriormente y la estructura interna de cada proceso dividida en diferentes tareas. Las que son presentadas en las figuras desde la 3-3 a 3-8.

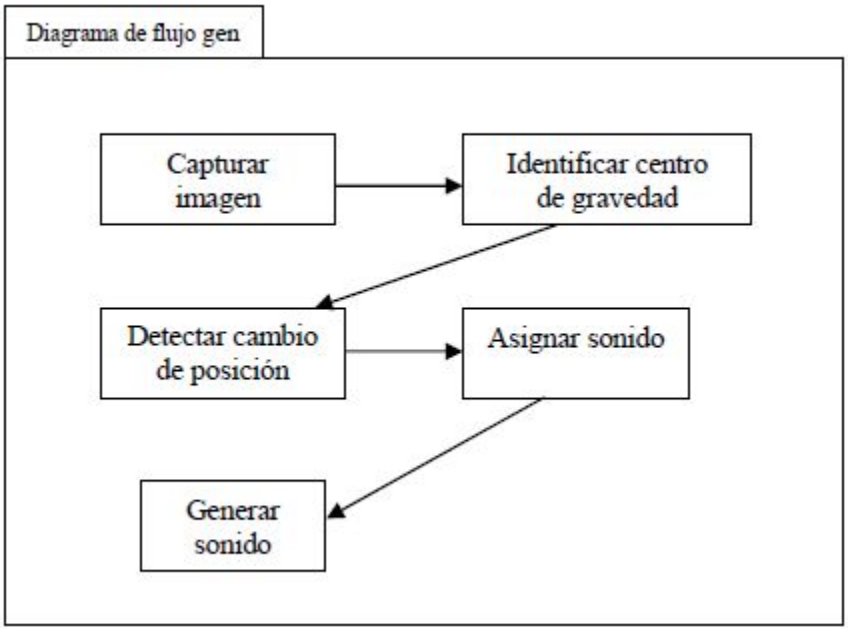


Figura 3-3 Diagrama de Flujo General

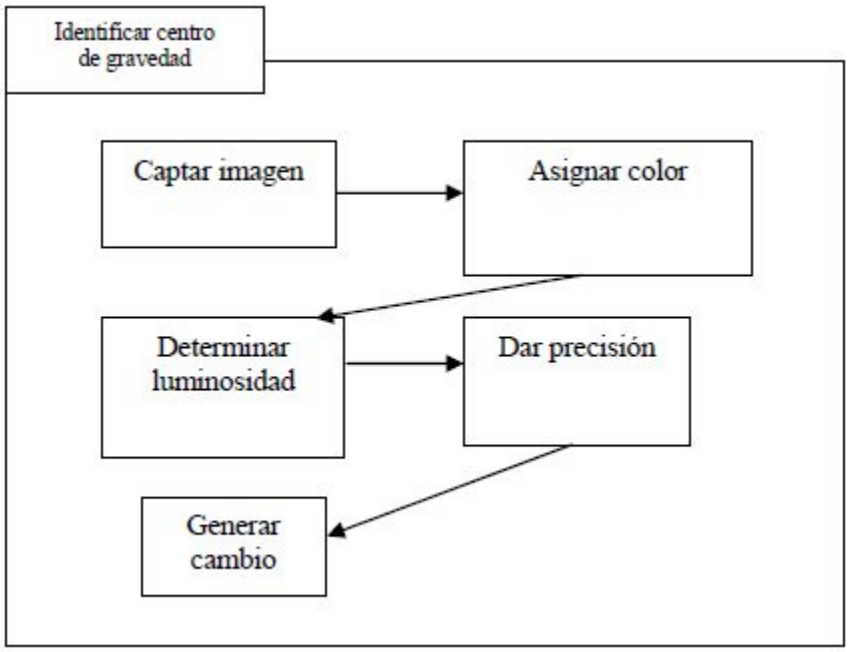


Figura 3-4 Identificar centro de gravedad

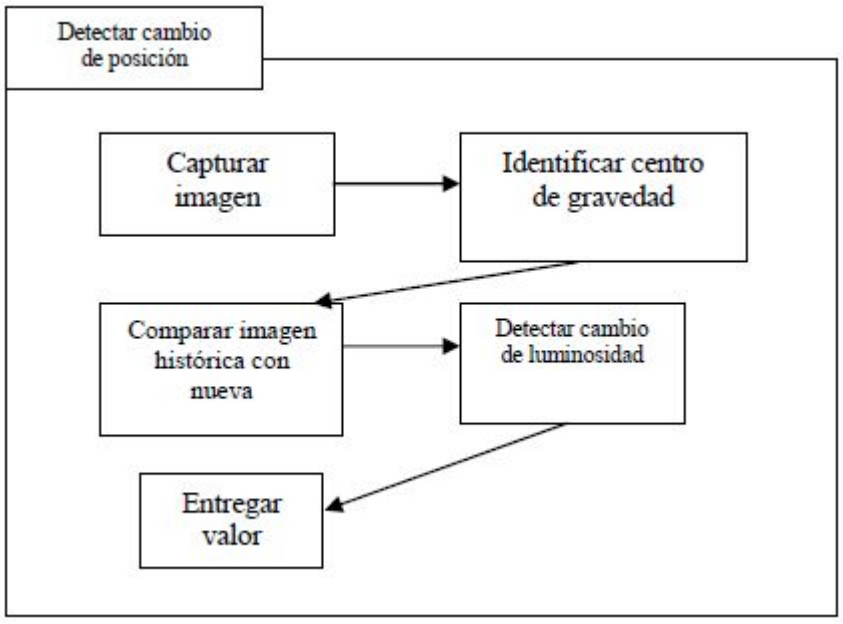


Figura 3-5 Detectar cambio de posición

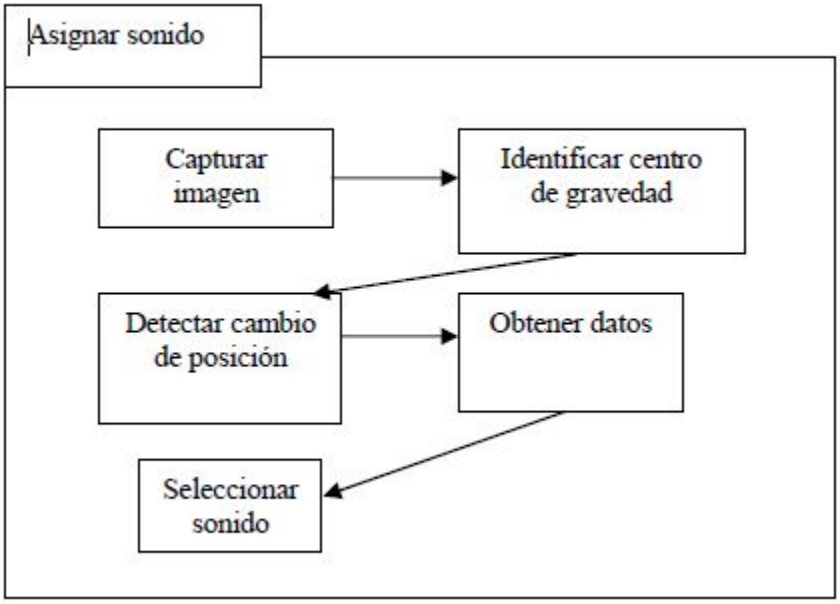


Figura 3-6 Asignar sonido

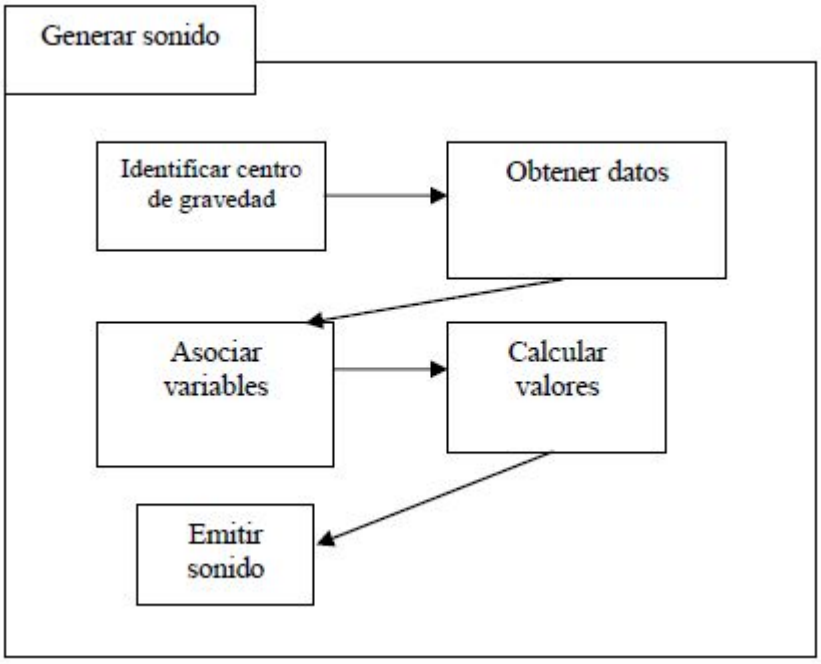


Figura 3-7 Generar sonido

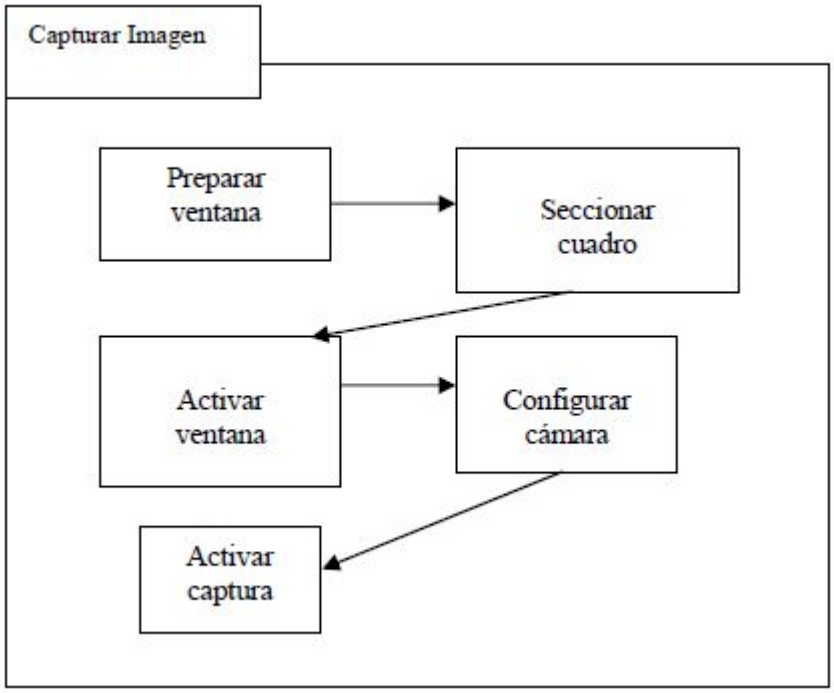


Figura 3-8 Capturar Imagen

3.2 DISEÑO GENERAL DEL SISTEMA

3.2.1 Pasos a seguir

Para el diseño externo del sistema se han propuesto los siguientes pasos:

1.- Crear área de trabajo

- Ventana de captura; tamaño, título, con o sin borde

2.- Preparar condiciones de captura

- Frames a utilizar

- device (detectar la entrada correspondiente a la cámara web)

3.- Tratamiento de la imagen (video capturado)

- Pasar a color de trabajo (rojo, azul, verde o B/N)

- Posicionar para captar distancias

4.- Asociar la distancia con el volumen del sonido

- Pasar como parámetro el valor de la distancia, siendo que mientras más cerca se encuentre el punto observado del ojo de la cámara mayor será el volumen de la nota, el rango en el que se manejará el sonido va de 0 a 127.

Para la captura de movimiento se utilizará una cámara web de alta resolución, dado la calidad de la imagen otorgada; al no tener que hacer una captura de movimiento para ser enviada por la web, el retardo de la imagen es mínimo. Como sistema de captura, no se utilizará un programa especializado, ya que estos estudios no están disponibles o son softwares comerciales, además que todos ellos utilizan algún medio externo ya sea de cableado o sensores, lo que no cumple con los objetivos planteados.

Como herramienta de programación se utiliza el lenguaje Pure Data ya que es un entorno de programación visual orientado a objetos que permite ser extendido de manera muy flexible. La arquitectura de Pure Data está basada en un corazón y una serie de módulos externos, con distintas capacidades.

Como sistema operativo se utiliza Windows XP debido a que las herramientas utilizadas con Pure Data son compatibles y por ser el sistema operativo más común. Es posible utilizar otros sistemas operativos, como es el caso de Linux o Mac, pero para esto se deben utilizar las versiones de Pure Data adecuadas al caso que corresponda.

Finalmente como requerimientos técnicos se tiene:

SW Pure Data más las librerías necesaria tales como GEM, Zexy
Una cámara web
Computador con los requerimientos para soportar Windows XP
Windows XP

No es necesario contar con tarjetas de video o sonido especiales, dado que con las características propias del computador, es suficiente, no se requiere de calidad de resolución de imagen ya que se analiza intensidad de luz.

3.2.2 Interacción usuario - sistema

A continuación se presenta un cuadro en el cual se muestran los distintos eventos junto a su descripción en los que se puede encontrar el usuario con el sistema.

Evento	**Descripción**
1.- Conectar la cámara	El usuario debe cerciorarse de que la cámara esté bien ubicada a la distancia requerida y conectada.
2.- Iniciar el sistema	El usuario debe poner en marcha el sistema.
3.- Crear y configurar el área de trabajo	El usuario debe configurar el área de trabajo, debe seleccionar la opción crear para abrir la ventana donde se verá la imagen que será capturada y procesada.
4.- Enfocar la cámara	El usuario se debe cerciorar de que la cámara está enfocando lo que se desea capturar y que se encuentra a la distancia apropiada.
5.- Cambiar colores de la captura	El usuario debe pasar del estado normal de la imagen al estado del color que se va a procesar, este puede ser rojo, azul, verde o alfa (duotonal).
6.- Identificar punto a seguir	El usuario debe especificar cuál será el punto que se va a seguir para identificar el

	movimiento efectuado.
7.- Comenzar captura	El usuario debe ejecutar la opción de captura de imagen.
8.- Cambiar tipo de instrumento (opcional)	Si el usuario o el ejecutante lo desea, cambiar el timbre de las notas emitidas, es decir, cambiar el instrumento del sonido que se está escuchando.

IV CAPÍTULO

IMPLEMENTACIÓN DEL SISTEMA

4.1 DISEÑO INTERNO DEL SISTEMA

En la figura 4-1 se presenta el diagrama de flujo del funcionamiento del sistema interno por funciones.

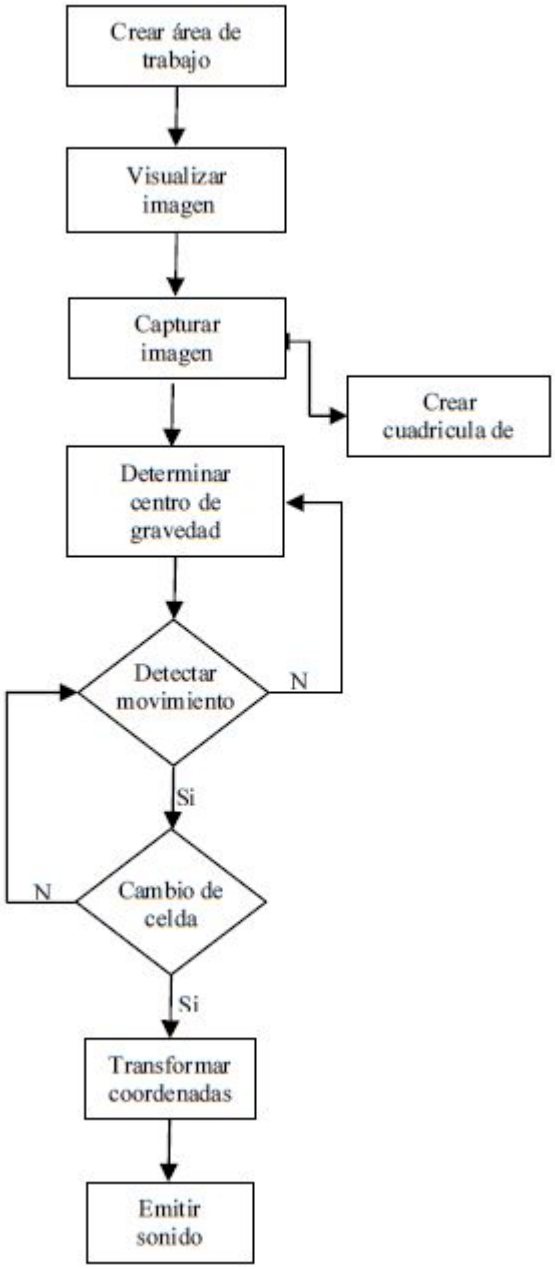


Figura 4-1 Diagrama de flujo del sistema interno

Durante el desarrollo del sistema surgieron una serie de dificultades. Uno de los problemas presentados es que Pure Data o por lo menos las herramientas de GEM trabajan sólo con una ventana, es decir, ejecuta sólo una ventana de trabajo. Es posible crear varias ventanas, pero cualquier cambio que se efectúe sobre [gemwin] se verá reflejado en la ventana activa. Es por esto que no se podría trabajar con dos cámaras o se debería utilizar las cámaras en forma independiente, ejecutando la captura del movimiento como acciones separadas.

Existen herramientas como [pd_mgrid] que reconoce el cambio de posición en una película, en tiempo real, el problema es que esta herramienta sólo está disponible para Linux. Por lo tanto es necesario buscar una herramienta similar en GEM, compilar la función existente de forma tal que funcione en Windows o crear una nueva función que permita realizar esta tarea, esto es determinar el cambio de posición. Al crear esta función es necesario programarla en C o C++, crear una nueva DLL y agregarla como librería a Pure Data. Toda modificación o nueva función que se desee crear, debe ser codificada en C o C++ y ser agregada como Librería o DLL, ya que en Pure Data no existe una programación directa mediante líneas de código.

Existe otra herramienta que es [pix_blob], con la que es posible capturar cercanía con respecto al punto observador (ojo de la cámara), esto teniendo un fondo parejo y un punto a reconocer que se distinga, tenga brillo o contraste con respecto al fondo. Al considerar la cercanía con el ojo de la cámara, se podría tomar esto como el volumen de la nota que se está emitiendo.

Otro de los problemas es cómo asignarle el valor de duración a la nota, ya que para hacer sonar una nota es necesario considerar tres parámetros que son: frecuencia, duración y volumen. La duración que se quiere trabajar es la cantidad de tiempo que se encuentre afectada la zona que fue asignada a cada nota, pero no es posible determinarlo con anterioridad. Al trabajar con la coordenada “x” como se venía planteando, se ha encontrado que esta variable es poco estable, es decir, que los valores que se pueden captar cambian muy rápido por lo que no es posible apreciar la mantención o duración de una nota.

Dependiendo del alcance de la cámara que se utilizará, corresponderá a la cantidad de cuadros en los que se verá dividida el área de trabajo y claro que esto también llevará a tener definida cuál será la dimensión de esta área de trabajo, aunque inicialmente se está trabajando con una ventana de 600 x 600 pixeles.

El problema más grande que se ha presentado es la poca documentación de las funciones con las que cuentan las librerías utilizadas en Pure Data, y el manejo y parámetros aceptados por cada una de estas funciones. Otra cosa novedosa es la presentación de la programación en Pure Data y la supuesta codificación de ésta.

La gráfica del programa presentado es muy sencilla, dada las limitaciones de programación que Pure Data tiene para estos fines, no cuenta con una interfaz gráfica “amigable” para el usuario, debido al tiempo requerido para desarrollar ésta y porque no está dentro de los objetivos planteados, además que para desarrollar una interfaz, es necesario agregar nuevas librerías al programa Pure Data, librerías que manejen GUI.

Inicialmente el objetivo es obtener el sonido de un punto en una posición específica dependiendo del cambio efectuado en el área de trabajo. Además de obtener un sonido, se desea obtener un grado de intensión en ello, es decir lograr realizar una interpretación musical a base de movimientos, logrando determinar rapidez e intensidad.

Para que el sistema funcione, lo primero es crear el área de trabajo. Se debe crear una ventana donde se mostrará la imagen captada por la cámara, a esta ventana se le aplican las divisiones o cuadros que representarán a cada nota y en ésta además se determinará el movimiento de cercanía con respecto a la cámara.

Los parámetros que acepta esta ventana de inicio son: tamaño, bordes (si tiene o no bordes), barra de título, nombre de la ventana, color, crear/destruir ventana, reset (limpiar los valores entregados al crear la ventana para ser modificados en una creación posterior).

La programación de la ventana de inicio está dada por la figura 4-2, correspondiendo al código mostrado en la figura 4-3.

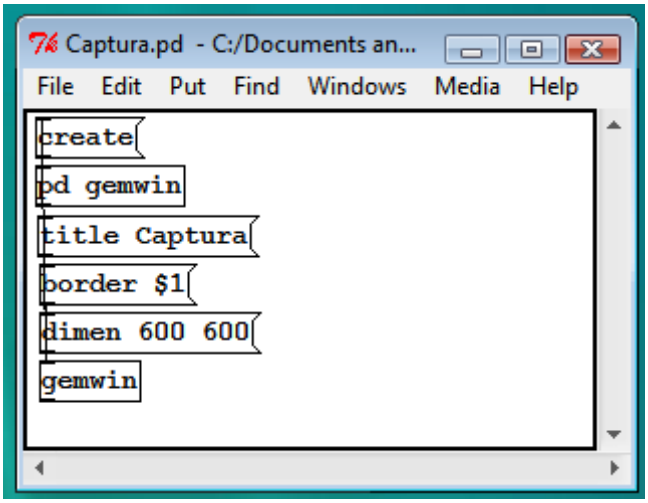


Figura 4-2 Cabeza del sistema propuesto

```
#N canvas 158 37 760 327 12;
#X floatatom 106 72 5 1 127 0 - - -;
#X floatatom 148 135 5 0 5000 0 - - -;
#X floatatom 55 15 5 0 127 0 - - -;
#X obj 53 250 noteout 1;
#X floatatom 598 11 5 0 0 0 - - -;
#X obj 587 41 pgmout 1;
#X obj 55 209 makenote 100;
#X text 661 10 asigno el instrumento;
#X text 109 11 numero de nota 0-127;
#X text 160 59 volumen 1-127;
#X text 200 130 duracion en ms;
#X text 140 250 salida;
#X obj 13 -2 bng 15 250 50 0 empty empty empty 0 -6 0 8 -262144 -1-1;
#X connect 0 0 6 1;
#X connect 1 0 6 2;
```

Figura 4-3 Código en Pure Data de Cabeza del sistema propuesto

Para la generación del sonido, es necesario contar con tres parámetros de entrada que son: el valor de la nota, que va entre 0 a 127, valores utilizados por Pure Data para definir los sonidos según su frecuencia, son 127 ya que esta es la cantidad máxima de sonidos audibles que se pueden generar por el computador, se utilizan valores enteros. Si se ha programado el trabajo con el tono de una nota en específico, es posible modificar este tono en tiempo de ejecución.

Otro de los parámetros requeridos es el volumen, que está comprendido entre 1 y 127. Finalmente es necesario asignar la duración del tono audible, que está dado en ms. (milisegundos). Para que esto tenga sentido y se escuche, se hace mediante la función [makenote] (figura 4-4 función de sonido [makenote]), que al asignarle un valor lo utiliza como tiempo en la duración del tono emitido, si no se le entran parámetros, asume todo en 0 o 1, es decir que suena el tono de un piano, que es el tono predeterminado y al mínimo de volumen.

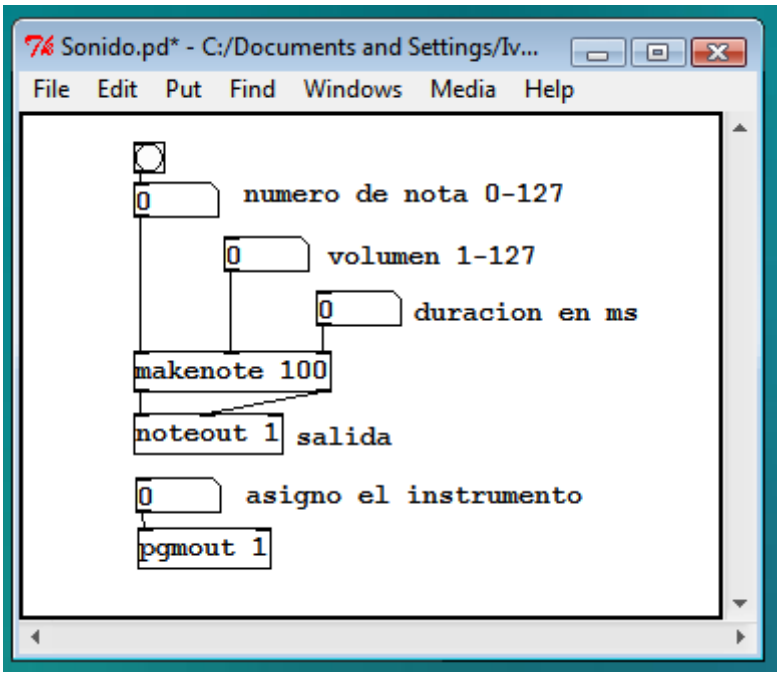


Figura 4-4 Emisión de sonido

Al utilizar la variable [noteout], se está utilizando la salida de audio propia del computador, es decir con los parlantes comunes o integrados, no un dispositivo en especial y además utiliza los tonos que se pueden generar por el computador y no unos predeterminados o grabados, como es el caso de los Samplers.

Para la captura del movimiento y reconocimiento de la cámara se utiliza como herramienta fundamental [pix_video] (figura 4-5 presenta la función [pixvideo] para la captura de movimiento), a esta herramienta se le pueden agregar varias funciones con las que se puede regular la intensidad de luz, color, cantidad de frames a los que corre la imagen, es decir, se puede ver la película en velocidad real, adelantar o retrasar. Es posible trabajar sólo en dos tonos o en tonos de grises, entre otras cosas.

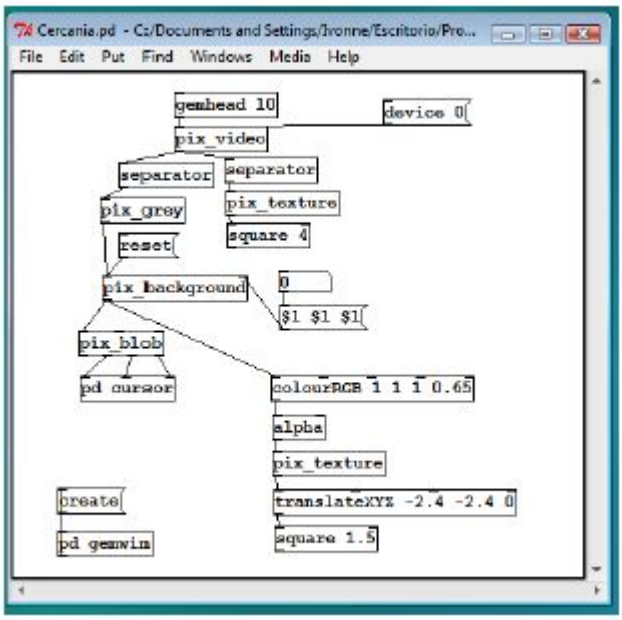


Figura 4-5 Captura de movimiento con [pixvideo]

También es posible rotar las imágenes, modificar tamaño o la forma. Al trabajar con esta función, es posible dimensionar o cuadricular la ventana de trabajo con el fin de contar con los cuadros que identificarán cada sonido.

Para lograr el volumen del sonido, se determina la cercanía del punto observado con respecto al foco de la cámara. Aquí se presenta la captura del movimiento de una mano, figura 4-6, nótese que esta captura es a modo de prueba, ya que no se está considerando la imagen del cuerpo del ejecutante. Con este tipo de captura es posible determinar mediante el centro de gravedad cómo ha ido variando la imagen considerando las diferencias de luminosidad, con lo que es posible asignar el volumen de la nota que se está emitiendo. Esto es posible mediante la función [pix_blob] y los parámetros adecuados para lo que se desea obtener.

Figura 4-6 Determinación de volumen

4.2 IMPLEMENTACIÓN COMPUTACIONAL

A continuación, en la figura 4-7, se presenta diseño del programa propuesto con la relación de todos sus módulos.

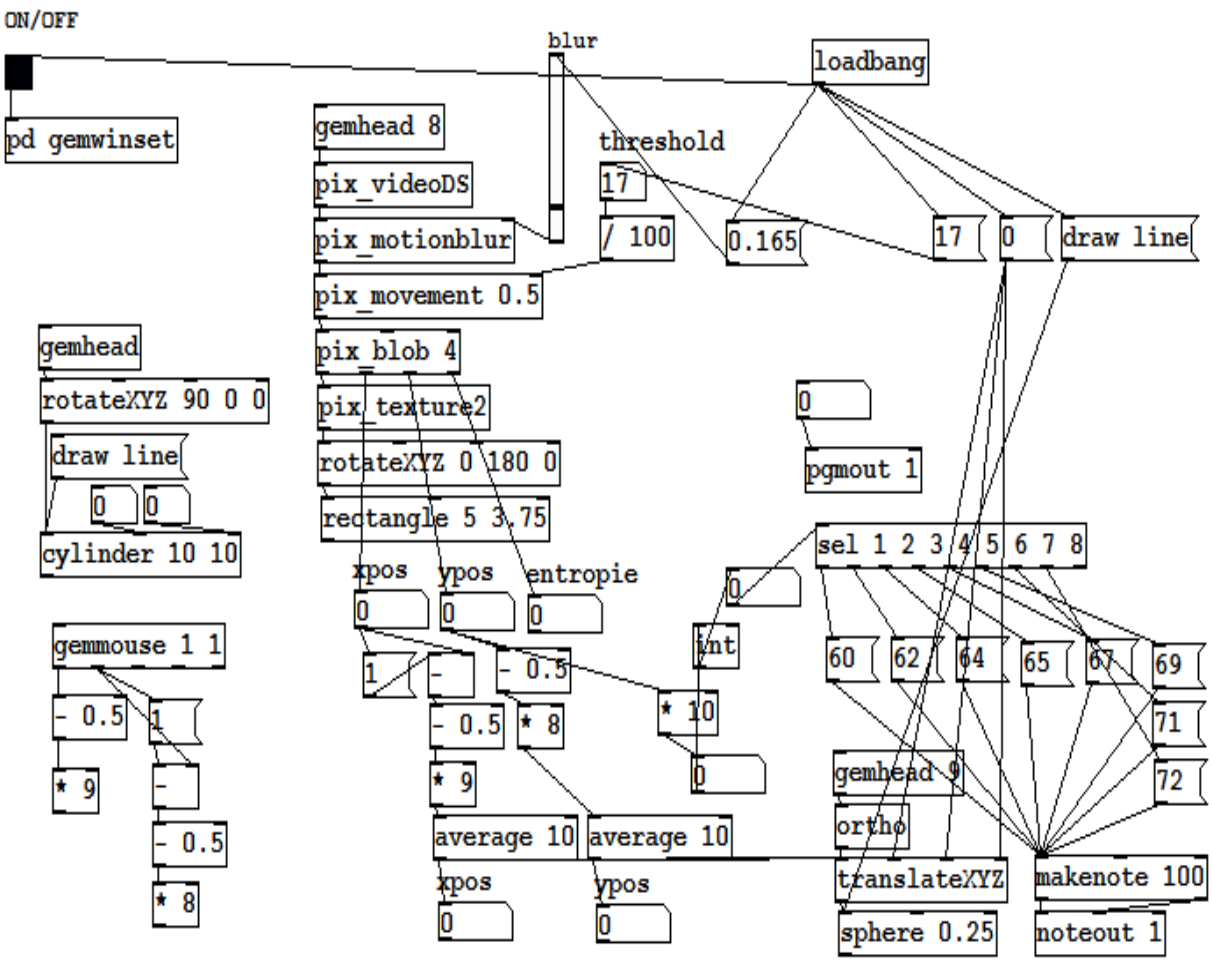


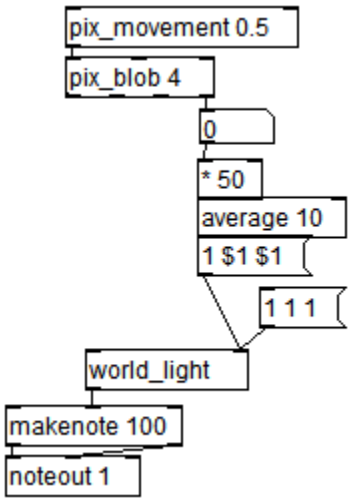
Figura 4-7 Diseño del sistema

4.3 PROGRAMA DIVIDIDO POR FUNCIONES

En esta sección se presenta el comportamiento de las distintas funciones utilizadas en la implementación del sistema así como una breve descripción de cada componente de dichas funciones.

Cada componente tiene su propio código, el que puede ser modificado para crear una nueva utilidad. En la programación tanto de las funciones como de las componentes, no se ve explícitamente el código fuente, sólo se trabaja de forma visual gráfica.

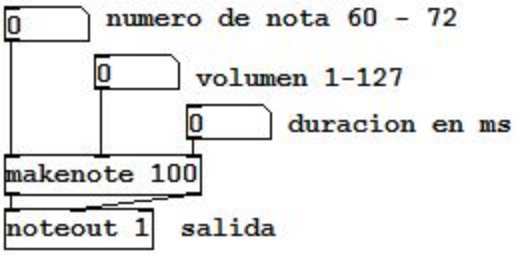
Asignar volumen de la nota



Funcionamiento

Al obtenet el valor de la entropia con [pix_blob], es posible trabajar el valor del volumen de la nota. Si al tener el punto de mayor luminosidad, se tiene el valor del tamaño de ese punto o su variación, mientras mas grande el valor significa que el punto es grande lo que implica que está lejos del foco de captura y mientras mas pequeño este valor, implica que está más cercano al foco de captura. Este valor es en coma flotante y menor a 1, por lo que es necesario multiplicarlo por 1000 (mil) y obtener valores que sean aceptados por [makenote], estos valores van entre 1 a 127. valores admitidos por el computador siendo 0 sin audio perceptible y 127el máximo perceptible.ya que los valores que se obtienen mediante la entropia son inversos a lo requerido, es decir, mientras mas alejado del foco mayor el valor y lo postulado es que matyor valor, mayor volumen, es que resulta necesario hacer una conversión de estos

valores, para que los valores de entrada sean coherentes a los valores de salida.
Generar Sonido

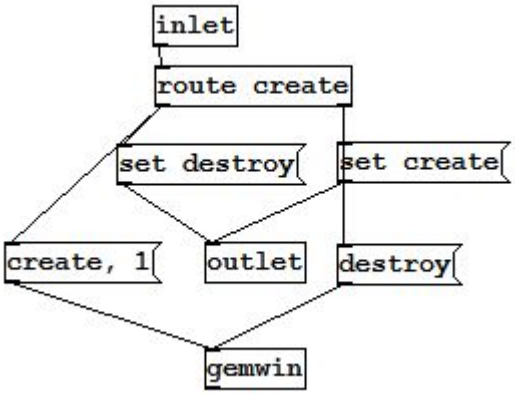


Componente	**Descripción**
[makenote]	Envía un mensaje de prendido (note-on) y retrasa el apagado (note-off). Se puede usar las salidas MIDI o procesos de notas externos dentro de Pure Data. [makenote] acepta tres parámetros de entrada que son: número de nota, volumen y duración. El número de nota va entre 1 y 127, estos valores corresponden a la representación MIDI de las frecuencias de las notas, este rango de notas es imitado a 127 valores debido a que es el rango audible de posible emisión por el computador.

[noteout]	Permite la salida del sonido MIDI generado por la tarjeta del sonido, seleccionando la salida disponible (parlantes).

Funcionamiento
Permite escuchar los sonidos mediante [makenote], es posible cambiar manualmente los valores que recibe como parámetros como lo son el número de nota que va entre el 1 al 127, pero se está trabajando solo con 8 notas que van entre 60 y 72 que corresponden a una escala (escala central del piano) DO – RE – MI – FA – SOL – LA – SI – DO. Volumen de la nota escuchado y la duración de la nota, la duración que se está utilizando es un valor constante de 500 ms, que equivale a la duración del sonido emitido por una cuerda pulsada del piano. Mediante [noteout] se está seleccionando el medio por el cual se está emitiendo el sonido, es decir, por notas MIDI y por los parlantes incorporados en el computador.

Crear área de trabajo

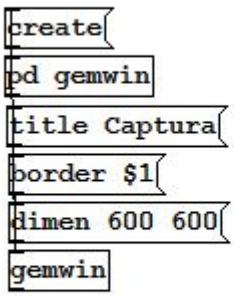


Componente	**Descripción**
[inlet]	Recibe un mensaje externo. Parámetro de entrada
[outlet]	Devuelve un valor. Parámetro de salida
[route create]	[route_create] comprueba el primer elemento de un mensaje contra cada uno de sus argumentos de creación, que pueden ser números o símbolos (pero no una mezcla de los dos a no ser que los tipos de datos sean definidos explícitamente), luego envía los mensajes por las salidas apropiadas. Si se encuentra un bang, y el mensaje contiene sólo UN elemento, entonces envían un aviso a la salida correspondiente. Si un bang es encontrado, y el mensaje contiene múltiples elementos (una lista), entonces envían todos los elementos de la lista excepto el primer elemento a la salida correspondiente. Si ningún bang es encontrado, entonces envían el mensaje entero la salida de derecha – la salida "de

	rechazamiento”. El número de salidas es el número de argumentos más uno.

Funcionamiento
Esta función permite abrir y cerrar la ventana de trabajo, además de activar esta ventana, dado que solo creando la ventana no es suficiente para trabajar sobre ella.

Activar atributos de la ventana

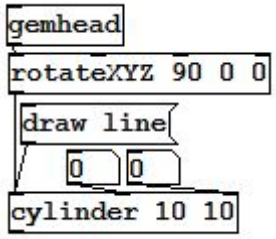


Componente	**Descripción**
[gemwin]	Es la ventana de control. Recibe varios parámetros controlando los atributos de la ventana de trabajo. El argumento inicial es el número de frames por segundos para utilizar, el valor por defecto es de 20 frames por segundo. Es extremadamente importante activar la interpretación antes de dibujar algo debido a que editar al mismo tiempo no es recomendado ya que es una buena manera de generar un segmento defectuoso. Cuando la interpretación es conectada, los objetos de Gem establecen una red de interpretación. Cuando la interpretación es apagada, la red es desactivada.

Funcionamiento
Es posible abrir la ventana donde se trabajará mediante el mensaje [create[, con este se abre la ventana de trabajo con los valores asignados. Es posible dar valores a tamaño de la ventana, modificar en tiempo de ejecución estos valores, pero para que se vean reflejados es necesario cerrar la ventana y abrirla nuevamente. Otro de los valores a asignar es el nombre de la ventana, si no se le asigna un nombre en particular, la ventana adopta por defecto el nombre “GEM”. La ventana puede o no tener borde, un mensaje de [border 1[crea todo un marco y

espacio para el título o nombre de la ventana. Es posible asignar la ubicación de la ventana dentro de la pantalla, de no dar valor a esto, toma por defecto la esquina izquierda del monitor.

Segmentar espacio de trabajo



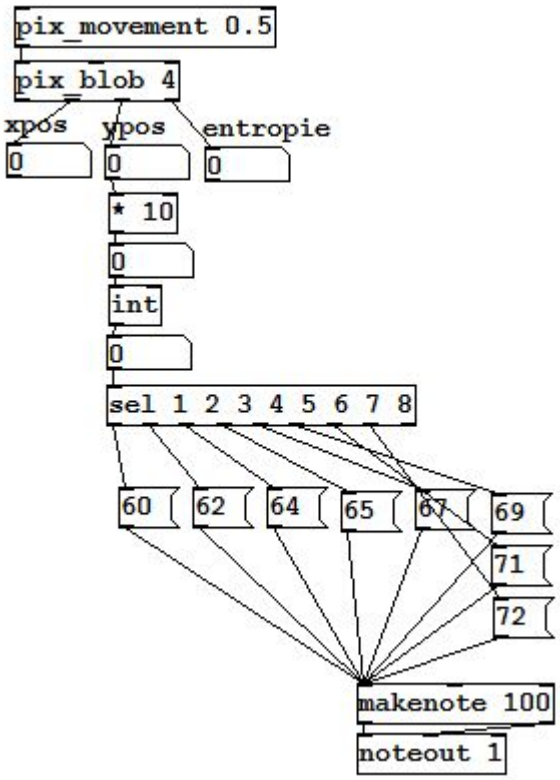
Componente	**Descripción**
[gemhead]	[gemhead] conecta los objetos de la GEM para el manejador de la ventana. El principio de cualquier gemList comienza con el gemhead. Sin el gemhead, los objetos de GEM no recibirán el mando (la orden). Cualquier objeto de GEM puede ser conectado a gemhead y estos recibirán la oreden de reconocimiento. Si gemhead recive un bang, ejecutará lo que ha reconocido. [gemhead] también acepta 1 o 0 para permitir e incapacitar la interpretación para la cadena.

	gemhead toma un argumento para determinar cuando recibe el mando. El valor por defecto es 50, mientras menor sea el valor, mas pronto [gemhead] recibirá el mando, el menor valor que reconoce es 1. Este valor es importante cuando se desea hacer una mezcla con el canal alfa y para ciertos objetos como la luz, este valor puede ser modificado dinámicamente mediante un mensaje “set”. El valor de orden(pedido) de interpretación también puede ser negativo. Valores inferiores (-10) serán dados antes de valores más altos (-3). [gemhead] con números negativos no será afectado por cambios de punto de vista.
[cylinder]	El objeto [cylinder] entrega un cilindro en una posición y color predeterminado. La apariencia del cilindro puede ser modificada con el mensaje [draw[, el tamaño puede ser modificado por medio de la segunda entrada.
[rotateXYZ]	[rotateXYZ] acepta un gemList y cambia la matriz de transformación corriente por la rotación especificada. La rotación alrededor del X, Y y el Eje de altura (en este orden) puede ser especificada separadamente por argumentos y cambiada vía entradas.
Funcionamiento	
Una vez creada y activada la ventana de trabajo, se crea una figura para dividir esta ventana, esta figura corresponde a un cilindro, del cual solo se ven las líneas resultantes de la unión de sus vértices y es rotado en 90º para que quede en forma	

vertical y logre el efecto divisor que se requiere.

De esta forma la ventana queda dividida en 10 secciones horizontales de las cuales 8 serán reconocidas con sonidos (notas musicales).

Convertir posiciones en sonido



Componente	**Descripción**

[pix_movement]	Detecta el movimiento entre dos marcos subsecuentes y lo almacena en el canal alfa para IMÁGENES- RGBA y en los canales de luminancia para YUV-*GRAYSCALE-IMAGES. Debido a que el ruido en la imagen puede afectar este tratamiento, cambios entre dos pixeles correspondientes que son más pequeños que <el umbral> no son considerados.
[pix_blob]	Calculará el centro de gravedad de una cierta combinación de canales, es posible escoger esta combinación para los cálculos cambiando el <mode>. Modos válidos son: 0 (color gris) (falta), 1 (rojo), 2 (verde), 3 (azul), 4 (alfa). Por ejemplo, el seleccionar "rojo" (1) va a la posición de peso de cada pixel con su valor rojo. Los pixeles con más rojo serán ponderados más fuertes, así moviendo " el centro de gravedad " - la posición de gota - más cerca a si mismo. La cantidad total "de los rojos" en la imagen define el tamaño de la gota. Es posible especificar una combinación de canales <con pesos en color>. pE: " 1 0 2 " dará más peso al canal azul que al canal rojo, verde - y los valores alfa no serán tomados en cuenta.
[select]	En su forma más simple, [select] chequea la constante de entrada, que es definida al crear el argumento. Si coinciden, la primera salida da el “bang”, de otra manera la entrada simplemente es enviada por a la segunda salida. Varios argumentos de creación pueden ser definidos. En efecto se puede usar [selec] para probar la entrada para muchos valores diferentes. Existirá una salida para cada valor

	de prueba y finalmente una salida para los valores que no coincide con ninguno de ellos (la salida de derecha).

Funcionamiento

Con [pix_movement] es posible detectar un cambio entre una imagen y otra, detecta variaciones entre cuadros de cada imagen, esta opción junto a [pix_blob] hacen posible una detección total de movimiento entre imágenes.

[pix_blob] detecta el centro de gravedad de la imagen capturada, esto quiere decir, que identifica el punto con mayor luminosidad dentro de la imagen con lo que cualquier movimiento detectado, este punto variará, si no es en posición puede ser en tamaño. Además con esta herramienta es posible obtener como respuesta tres valores que son: posición en coordenadas X e Y y entropía (tamaño). Tomando el valor Y, que es el que se está trabajando, se tiene el movimiento en la coordenada Y, este valor que se obtiene está en coma flotante y es menor a 0 (cero) por lo que es necesario hacer una conversión. La variación de este valor va entre 0.0 a 0.9 lo que se adecua a los valores que se están trabajando, 1 – 8 (notas utilizadas) los valores se multiplican por 10 y luego se transforman a enteros por lo que se obtienen valores entre 0 a 9, luego se hace una selección de estos valores y se le asigna el valor correspondiente a la nota que se espera que se escuche. Esto es equivalente a un “case” donde:

case 1:

nota = 60;

case 2:

nota = 62;

:

case 8:

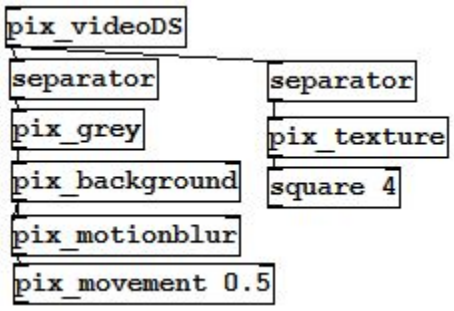
nota = 72;

Cada escala va separada por doce valores, escala con accidentes incluidos sostenidos (#) y bemoles (b).

Como se está trabajando con notas naturales es que se toman los valores de tonos completos a excepción de los valores entre MI y FA que solo tienen medio tono de separación.

Estos valores entran en [makenote] identificando el sonido que debe ser emitido y luego pasa a [noteout] que finalmente hace que suene.

Asignar color a la ventana de trabajo

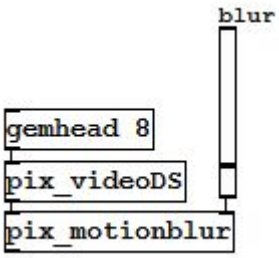


Componente	**Descripción**

[separator]	[separator] aísla todo debajo de ello (y todo lo demás en la cadena) de todos los efectos de transformación.
[pix_grey]	Convierte el espacio en color de una imagen a GRIS. Las imágenes pueden ser almacenadas en varios formatos color-spaces. Mientras YUV le da las imágenes coloreadas y RGBA agrega un canal alfa, GRIS es muy práctico si no es relevante el color y se tiene que ahorrar memoria de CPU. Se puede usar [pix_grey] para convertir las imágenes de cualquier formato en el ESPACIO GRIS. Si la imagen ya está en el ESPACIO GRIS, esto no hará nada.
[pix_background]	[pix_background] remueve los nuevos movimientos del fondo de una imagen, comparando una imagen estática en memoria con una corriente entrante de vídeo. Todos los valores sin una gama dada son cambiados a negro.
Funcionamiento	
Color asignado a la ventana de trabajo, por defecto trabaja con fondo negro, este valor puede ser modificado en tiempo de ejecución, para ver el cambio reflejado es necesario cerrar y volver a abrir la ventana de trabajo.	

Pero para este caso no es necesario considerar un valor en especial ya que no se trabaja con este fondo directamente.

Determinar precisión



Componente	**Descripción**
[pix_videoDS]	Solo funciona en Windows. Captura de video en tiempo real usando DirectShow.
[pix_motionblur]	Aplica un aspecto borroso de movimiento muy simple y rápido a una secuencia de imágenes. El método usado implica la mezcla

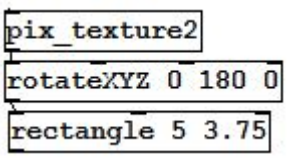
	de la imagen actual con una imagen histórica y guarda el resultado seguida de la histórica. La mezcla es: output = (stream * gain) + (history * 1 - gain), la aplicación de un factor de aspecto borroso más alto se mezclará en más de la imagen histórica y así más de la historia será guardada causando enturbiar más notoria.

Funcionamiento

Es posible determinar la precisión del movimiento, es decir, que se pueden identificar la más leve diferencia entre los cuadros o considerar grandes rangos de espacio [pix_motionblur] permite esta detección mientras menos sea el valor que se le entrega, mayor será la precisión que considere, este valor puede ser modificado en tiempo de ejecución.

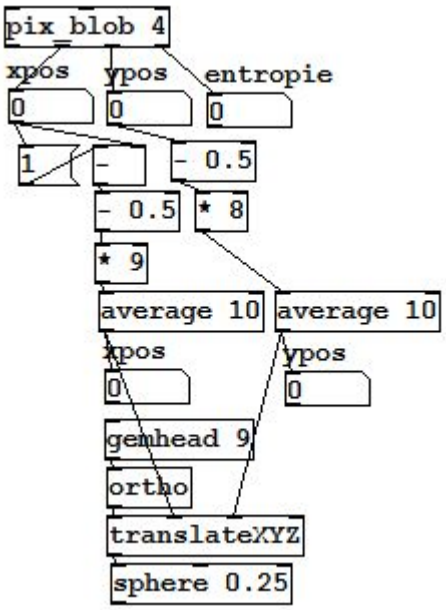
[pix_videoDS] permite la captura de la imagen (video) que será el valor de entrada que recibe [pix_motionblur] para el análisis.

Segmentar Ventana de trabajo



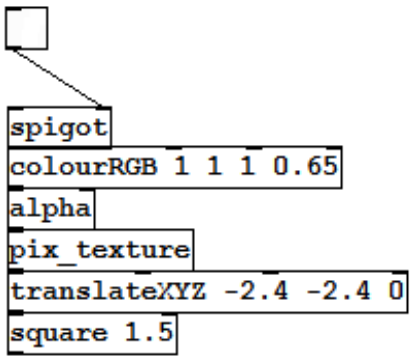
Componente	**Descripción**
[pix_texture]	Permite trazar un mapa de textura con la corriente pix. Independiente del valor que se encuentre en la red de pix, este será utilizado. Enviando un mensaje de calidad se puede cambiar la calidad del trazar un mapa de textura. Sin embargo, sobre muchas máquinas (sobre todo SGIs), no hay ninguna diferencia de velocidad. [pix_texture] es capaz de habilitar las texturas de imágenes de cualquier tamaño. [pix_texture] trata de usar el modo más rápido de conseguir un pix en una textura. Esto implica la utilización "el rectángle"-texturing de ser disponible. Esto, a su turno, puede conducir a algunos problemas con varios geos.
[rectangle]	El objeto [rectangle] entrega un rectángulo en una posición y color predeterminado. El ancho y alto del rectángulo pueden ser seleccionados por argumentos y modificados por medio de la segunda y tercera entrada.
Funcionamiento	
Este es el cuadro donde se ve la imagen capturada por la cámara de video o web, es un rectángulo el que debe ser rotado en 180º para que tome el aspecto de figura plana.	

Identificación del centro de gravedad

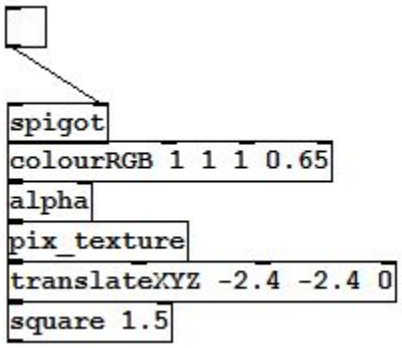


Componente	**Descripción**
[translateXYZ]	Acepta un gemList y cambia la matriz de transformación corriente por la traducción especificada. La traducción es determinada por un vector (X Y Z).
[ortho]	Cambia el modo de inspección corriente. "perspectiva" (los objetos que están lejos aparece más pequeña que los objetos que están cerca), "ortográfico" (los paralelos aparecen paralelos) para todas las formas subsecuentes.

[sphere]	El objeto [sphere] da una esfera segmentada en la posición predeterminada y con el color activo. La mirada de la esfera puede ser cambiada con el mensaje <draw>, su tamaño puede ser cambiado vía la segunda entrada.
Funcionamiento	
El determinar el centro de gravedad significa encontrar el punto de mayor luminosidad de la imagen capturada, cuando este valor cambia, es posible obtener los valores de la nueva posición. Para que visiblemente esto se detecte es que se crea una esfera que sigue este punto, con [translateXYZ] es posible entregarle los valores de X e Y para que la esfera se mueva al punto correspondiente, estos valores son entregados por [pix_blob]. A esta esfera es posible darle ciertas características tales como tamaño, color y estilo. El estilo quiere decir, forma de visualización como esfera sólida, sólo los puntos que la componen (vértices) o líneas (unión de vértices).	



Visualizar entorno de trabajo



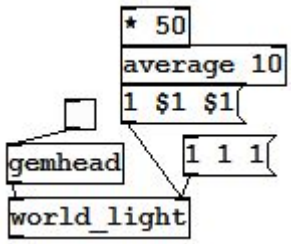
Componente	**Descripción**
[spigot]	Pasa mensajes de su entrada izquierda a su salida, mientras un número no nulo es enviado a su entrada derecha. Cuando su entrada derecha es cero, el mensaje de entrada es ignorado. [spigot] trabaja esencialmente como una entrada. Cuando la puerta está abierta, los mensajes pueden pasar. Cuando la puerta está cerrada, los mensajes no son considerados. Por defecto la puerta siempre está cerrada. [spigot] sólo acepta un argumento numérico de creación: un uno o un cero. En el tiempo de creación [spigot] está "cerrado". Con un uno por defecto sirve para abrir.

[colourRGB]	Pone el color de toda la forma subsecuente y operaciones de vértice mientras el objeto no sea reinicializado por otro [color] / [colorRGB]. Si se pone el valor alfa, se tendrá un objeto [alfa] que permite la mezcla de alfa.
[alpha]	Conecta y desconecta la mezcla alpha. [alpha] automáticamente usa el glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA) la función de OpenGL. En una cáscara de nuez, cualquier pixel que tiene un componente alfa será mezclado con el pixel que está en el buffer, produciendo el efecto habitual coloreado de cristal.
[square]	El objeto [square] da un cuadrado en la posición predeterminada con el color activo. El tamaño del cuadrado puede ser cambiado vía la segunda entrada.

Visualizar entorno de trabajo
Funcionamiento
Ventana adicional en la que es posible ver la imagen o película sin modificaciones. Esta ventana es usada o visualizada en tonos grises y es de menor tamaño que la ventana de trabajo, en esta no se aprecia la cuadrícula con que se trabaja, color

seleccionado, posición del cursor ni punto de luminosidad. Sirve para tener una referencia de lo que va sucediendo sin interferencias.

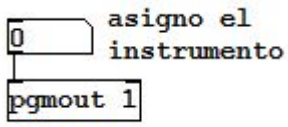
Asignar luminosidad



Componente	**Descripción**
[world_light]	Produce una luz que es a una distancia infinita de la escena. Esto quiere decir que todos los rayos ligeros (de luz) son paralelos, que reduce el cómputo algo. Se puede adaptar la luz con [rotate]. La segunda entrada pone el color de la fuente luminosa. Los atributos tienen que ser puestos antes de la interpretación de los vértices.
Funcionamiento	

Luminosidad que se le asignará al área de trabajo, esto es que tan brillante u opaca es la imagen con la que se está trabajando, esto incide directamente con el punto de mayor luminosidad y la precisión que se utilizará en la determinación de este.

Seleccionar Instrumento



Componente	**Descripción**
[pgmout]	selecciona el instrumento que se va a escuchar, por defecto es seleccionado el timbre del piano que corresponde al valor 1, se puede modificar fácilmente con un Number
Funcionamiento	
Ya se ha mencionado que se utilizará por defecto la tímbrica del piano, pero es posible mediante [pgmout] cambiar manualmente la tímbrica y en tiempo de ejecución.	

El valor utilizado por defecto es 1.

V CAPÍTULO

PRUEBAS Y RESULTADOS

5.1 PRUEBAS

Para la detección de fallas o errores, se realizaron diversas pruebas. Estas pruebas se realizaron en un cuarto en el cual se dispuso un telón blanco en una de las paredes, apegado a éste se puso una silla en la que se ubica el ejecutante. Frente a este conjunto se instaló a dos metros de distancia aproximadamente una cámara web con la cual se capturan los movimientos del ejecutante (movimientos de la mano), los que serán procesados por el sistema. El cuarto cuenta con iluminación pareja (igual intensidad de luz).

Al realizar las pruebas es posible encontrar errores en los datos de salida o respuesta por parte del sistema. Los posibles errores que pudieran producirse son principalmente que la frecuencia de la nota emitida no sea la que corresponda a la posición en la que se encuentra el punto en observación (mano del ejecutante). Esto puede deberse a que el punto a observar no sea bien enfocado, que se capte más de un punto en movimiento o que la iluminación de la habitación no sea la adecuada.

Para comprobar que el sonido emitido sea el correcto y que se están respetando los cuadros de posición se utilizan dos métodos:

El primer método utilizado es un medio visual, se cuenta con una tabla de valores con las posiciones, es decir, dependiendo de la posición, ubicación dentro del área de trabajo del punto observado, se tiene asignada una frecuencia específica, esta tabla se presenta en la figura 5-1.

Posición	Nota	Rango Frecuencia
1	Do	254,285 - 277,645
2	Re	277,646 - 311,645
3	Mi	311,646 - 339,43
4	Fa	339,44 - 370,615
5	Sol	370,616 - 416
6	La	416,001 - 466,94
7	Si	466,95 - 508,565
8	Do	508,566 - 555,29

Figura 5-1 Tabla Posición/Nota/Frecuencia

Como método de validación se le han incorporado al programa tres campo, los que entregan: posición en coordenadas X e Y, y la frecuencia asociada. Para ver los cuadros de posición se creó un módulo especial en el cual se pueden visualizar la frecuencia de la nota emitida y la coordenada del cuadro que se ha reconocido como nueva posición, como se muestra en la figura 5-2.

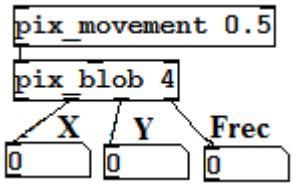


Figura 5-2 Módulo que entrega coordenadas X e Y, y frecuencia

El segundo método empleado consiste en utilizar un medidor de frecuencias, algo similar a un afinador electrónico de instrumentos, ese valor es comparado con los valores de la tabla de Frecuencias/Posición este dispositivo capta el sonido que se ha emitido y devuelve el valor de la frecuencia de éste. Estos valores son comparados con los valores que especifica el sistema con respecto a las frecuencias de las notas emitidas. Como se muestra en las figuras 5-3 y 5-4.

Figura 5-3 Medidor de Frecuencias

Figura 5-4 Medición de Frecuencia

5.2 RESULTADOS

Se realizaron pruebas con niños de entre 10 a 16 años sin problemas físicos, se captaron los movimientos realizados por estos y los datos fueron procesados por el sistema. La respuesta por parte del programa fue la emisión de un sonido correspondiente a una nota musical dependiendo de la ubicación.

Las pruebas fueron realizadas en condiciones normales, con varios movimientos por parte del ejecutante y con diferencias en la intensidad de luz dentro de la habitación, los datos de respuesta fueron puestos en tablas, los que se compararon con los datos de la tabla de referencia.

Los sonidos emitidos por el sistema como respuesta en condiciones normales, se puede decir que son coherentes con lo ejecutado y propuesto. Pero se pudo detectar que mientras el movimiento se produce dentro del mismo cuadro, se repite la nota la cantidad de veces que se ha detectado una variación de posición o movimiento.

Los sonidos que fueron emitidos por el sistema en condiciones no normales, es decir, con diferencias en la intensidad de luz, con la ejecución de varios movimientos a la vez, con la cámara mal enfocada, no fueron los que se esperaban, estos arrojaron errores en las frecuencias y en ocasiones no se alcanzó a tener una nota específica, hubo una variación muy rápida en la frecuencia emitida, por lo que no fue posible escuchar notas constantes.

Los valores obtenidos en las pruebas en las distintas condiciones, se muestran junto a los esperados en la figura 5-5.

Figura 5-5 Tabla comparación valores de respuesta

	Condiciones Normales	**Diferencias de luz**	**Sin enfoque**	**ESPERADO**	**Nota**	**Posición**
Niño 1	325	327	310	330	Mi	3
	490	496	490	494	Si	7
	433	450	462	440	La	6
	527	516	506	523	Do	8
	271	254	270	262	Do	1
Niño 2	490	506	490	494	Si	7
	387	386	365	392	Sol	5
	295	290	307	294	Re	2
	270	253	270	262	Do	1

	343	327	324	350	Fa	4
Niño 3	443	421	421	440	La	6
	386	376	403	392	Sol	5
	536	547	500	523	Do	8
	301	277	306	294	Re	2
	387	378	384	392	Sol	5

Según los datos de prueba que se pudo obtener es posible detectar que para que el sistema cumpla con lo esperado y propuesto debe encontrarse en las condiciones mínimas, es decir, con la luz adecuada, sin perturbaciones mientras se realizan los movimientos por parte el ejecutante y la cámara debe encontrarse bien enfocada. Es posible realizar cambios o mejoras en el sistema para que funcione de forma esperada en condiciones adversas.

CONCLUSIONES

El trabajo realizado cumple con las expectativas expuestas desde un comienzo. Se ha logrado el propósito de encontrar una congruencia entre el desplazamiento realizado y el sonido emitido, escuchar las diferencias en el volumen de una nota determinada, y visualizar la frecuencia de ésta con respecto a la posición.

Aún cuando se podrían utilizar los sistemas operativos Linux, Mac o Windows, para la ejecución de este trabajo se empleó Windows XP, por ser el sistema más accesible, mientras que el lenguaje de programación más apropiado para desarrollar el sistema fue Pure Data, este lenguaje de programación fue diseñado para trabajar con imágenes en movimiento y sonido, permite directamente la captura de movimiento y tiene la facultad de ser portable a otros sistemas operativos, mientras las librerías utilizadas sean compatibles con éste o tengan su equivalente.

El programa fue diseñado para la captura del movimiento (desplazamiento) de una de las manos del ejecutante, considerando la diferencia de luz y sombra para determinar las distintas posiciones.

El área de trabajo fue dividida en 8 secciones, a cada una de éstas se le asignó el sonido de una nota musical, correspondiente a la escala central del piano o frecuencias que van entre 254 y 555 Hz. Es posible ampliar el número de secciones, por ende el número de notas a utilizar dependiendo del área de trabajo con la que se va a trabajar, incluso se podría capturar no sólo el movimiento de una mano, sino que el desplazamiento de todo el cuerpo del ejecutante.

Para le ejecución del programa se debe emplear una sala con luz homogénea, debido a que la diferencia de luminosidad afecta la detección del punto observado (mano del ejecutante). La distancia a considerar entre la cámara y el ejecutante, podría variar entre 1,5 y 2 m de distancia.

Esta propuesta podría ser desarrollada con mayor cantidad de recursos y utilizarse como base o pilar para el desarrollo de programas que puedan ser utilizados en el área de desarrollo personal, intelectual, entretenimiento, así como en áreas de educación, arte, cine, etc.

REFERENCIAS

[1] http://es.wikipedia.org/wiki/Theremin

[2] http://axxon.com.ar/zap/223/c-Zapping0223.htm

[3] http://www.universia.net.mx/index.php/news_user/content/view/full/10927/

[4] Proyecto Hóseo, Alejandra Cerani. Coreógrafa, Directora e Intérprete. Profesora de Comunicación Audiovisual en la FBA, UNLP. http://www.alternativateatral.com/ver_nota.asp?codigo_nota=82

[5] Licenciado en Música de la Universidad de Chile. Especialista en Informática Músical. Profesor de la Universidad Arcis (Escuela de Música y Laboratorio Arcis de Informática Musical). Profesor de Informática Musical en la Universidad Metropolitana, UMCE. tomasthayer@gmail.com

[6] http://peo.on.ca/events/awards/OPEA/2005/Chau.htm

[7] http://www.utoronto.ca/gdrs/faculty/Chau.htm

[8] tom.chau@utoronto.ca
http://myhero.com/myhero/hero.asp?hero=tom_chau

[9] http://itp.nyu.edu/nime/2007/

[10] Física Clásica y Moderna. W.E. Gettys. Editorial McGraw-Hill. 1991.

[11] Introducción al sonido por Alejandro Pérez. Ingeniero en Comunicación Audiovisual especializado en Psicoacústica. http://www.videoedicion.org/manuales/audio/intro_al_sonido_.htm

[12] http://es.wikipedia.org/wiki/Sonido

[13] Arturo García González. Profesor Universidad Autónoma de Ciudad Juárez.

[14] Física Básica. Fernández-Rañada, Antonio. Ed. Alianza, 1997.

[15] Hernando Ortega Carrillo; Victor Hugo Godoy Aguirre; Carmen Ramos Nava http://www.dpye.iimas.unam.mx/mocap/MocapSystemSP.html

[16] M Música y Movimiento. Leonardo Riveiro Holgado. Profesor de Formación Rítmica y Danza. Escuela Universitaria de Profesorado (Segovia). Universidad Autónoma de Madrid. 1992.

[17] http://es.wikipedia.org/wiki/Pure_data

[18] http://mcs.hackitectura.net/tiki-read_article.php?articleId=26

APENDICE A – CÓDIGOS FUENTES

CÓDIGOS FUENTES

Algoritmos relevantes de funciones utilizadas.

pix_background.cpp

```
#include "pix_background.h"
#include <string.h>

CPPEXTERN_NEW(pix_background)

pix_background :: pix_background() :
 m_Yrange(0), m_Urange(0), m_Vrange(0), m_Arange(0), m_reset(1)
{
 long size,src,i;
 inletRange = inlet_new(this->x_obj, &this->x_obj->ob_pd, &s_float, gensym("range_n"));
 m_savedImage.xsize=320;
 m_savedImage.ysize=240;
 m_savedImage.setCsizeByFormat(GL_RGBA);
 m_savedImage.reallocate(); }

pix_background :: ~pix_background()
{
 if(inletRange)inlet_free(inletRange); }

void pix_background :: processRGBAImage(imageStruct &image)
{
 int h,w,hlength;
 long src,pixsize;

 src = 0;
 pixsize = image.xsize * image.ysize * image.csize;

 if(m_savedImage.xsize!=image.xsize ||
 m_savedImage.ysize!=image.ysize ||
 m_savedImage.format!=image.format)m_reset=1;

m_savedImage.xsize=image.xsize;
 m_savedImage.ysize=image.ysize;
 m_savedImage.setCsizeByFormat(image.format);
 m_savedImage.reallocate();

 if (m_reset){
 memcpy(m_savedImage.data,image.data,pixsize);
 m_reset = 0;  }

 hlength = image.xsize;

 unsigned char*data =image.data;
 unsigned char*saved=m_savedImage.data;

 for (h=0; h<image.ysize; h++){
 for(w=0; w<hlength; w++){
 if (((data[src+chRed ] > saved[src+chRed ] - m_Yrange)&&
 (data[src+chRed ] < saved[src+chRed ] + m_Yrange))&&
 ((data[src+chGreen] > saved[src+chGreen] - m_Urange)&&
 (data[src+chGreen] < saved[src+chGreen] + m_Urange))&&
 ((data[src+chBlue ] > saved[src+chBlue ] - m_Vrange)&&
 (data[src+chBlue ] < saved[src+chBlue ] + m_Vrange))&&



 ((data[src+chAlpha] > saved[src+chAlpha] - m_Arange)&&
 (data[src+chAlpha] < saved[src+chAlpha] + m_Arange)))
 {
data[src+chRed] = 0;
```

```
 data[src+chGreen] = 0;
 data[src+chBlue] = 0;
 data[src+chAlpha] = 0;  }
 src+=4; } }
 m_reset = 0;
}

void pix_background :: processGrayImage(imageStruct &image)
{
 int i;// h,w,hlength;
long src,pixsize;
unsigned char newpix, oldpix, *npixes, *opixes;

 src = 0;
 pixsize = image.xsize * image.ysize * image.csize;
 if(m_savedImage.xsize!=image.xsize ||
 m_savedImage.ysize!=image.ysize ||
 m_savedImage.format!=image.format)m_reset=1;

 m_savedImage.xsize=image.xsize;
 m_savedImage.ysize=image.ysize;
 m_savedImage.setCsizeByFormat(image.format);
 m_savedImage.reallocate();

 if (m_reset){
 memcpy(m_savedImage.data,image.data,pixsize);
 m_reset = 0;
 }

 npixes=image.data;
 opixes=m_savedImage.data;
 const unsigned char thresh=m_Urange;
 i=pixsize;
 while(i--){
 newpix=*npixes++;
 oldpix=*opixes++;
 if((newpix>oldpix-thresh)&&(newpix<oldpix+thresh))npixes[-1]=0;
 }
 m_reset = 0;
}
#ifdef __MMX__
void pix_background :: processRGBAMMX(imageStruct &image)
{
long i,pixsize;
 pixsize = image.xsize * image.ysize * image.csize;

 if(m_savedImage.xsize!=image.xsize ||
 m_savedImage.ysize!=image.ysize ||
 m_savedImage.format!=image.format)m_reset=1;

 m_savedImage.xsize=image.xsize;
 m_savedImage.ysize=image.ysize;
 m_savedImage.setCsizeByFormat(image.format);
 m_savedImage.reallocate();
if (m_reset){
memcpy(m_savedImage.data,image.data,pixsize);
 }
 m_reset=0;

 i=pixsize/sizeof(__m64)+(pixsize%sizeof(__m64)!=0);

 __m64*data =(__m64*)image.data;



 __m64*saved=(__m64*)m_savedImage.data;

const __m64 tresh=_mm_set_pi8(m_Yrange, m_Urange, m_Vrange, m_Arange,
 m_Yrange, m_Urange, m_Vrange, m_Arange);
 const __m64 offset=_mm_set_pi8(1, 1, 1, 1, 1, 1, 1, 1);
```

```
 __m64 newpix, oldpix, m1;

 while(i--){
newpix=*data;
 oldpix=*saved++;
 m1 = newpix;
 m1 = _mm_subs_pu8 (m1, oldpix);
oldpix= _mm_subs_pu8 (oldpix, newpix);
 m1 = _mm_or_si64 (m1, oldpix); // |oldpix-newpix|
 m1 = _mm_adds_pu8 (m1, offset);
 m1 = _mm_subs_pu8 (m1, tresh);
 m1 = _mm_cmpeq_pi32 (m1, _mm_setzero_si64()); // |oldpix-newpix|>tresh
 m1 = _mm_andnot_si64(m1, newpix);

 *data++ = m1;
 }
 _mm_empty();

void pix_background :: processYUVMMX(imageStruct &image)
{
 long pixsize;

 pixsize = image.xsize * image.ysize * image.csize;

 if(m_savedImage.xsize!=image.xsize ||
 m_savedImage.ysize!=image.ysize ||
 m_savedImage.format!=image.format)m_reset=1;

 m_savedImage.xsize=image.xsize;
 m_savedImage.ysize=image.ysize;
 m_savedImage.setCsizeByFormat(image.format);
 m_savedImage.reallocate();

 if (m_reset){
 memcpy(m_savedImage.data,image.data,pixsize);
 }
 m_reset=0;

 vec_dss(3);
 #endif
 }
}
#endif //ALTIVEC
```

pix_blob

```
#include "pix_blob.h"
pix_blob :: pix_blob(int argc, t_atom *argv)
{
if (argc) {
 if (argc==1) this->ChannelMess(atom_getint(argv));
 else this->GainMess(argc, argv);
 } else m_method = 0;

 m_xOut = outlet_new(this->x_obj, &s_float);
 m_yOut = outlet_new(this->x_obj, &s_float);
 m_zOut = outlet_new(this->x_obj, &s_float);
}


pix_blob :: ~pix_blob()
{
 outlet_free(m_xOut);
 outlet_free(m_yOut);
 outlet_free(m_zOut);
}
```

```
void pix_blob :: processRGBAImage(imageStruct &image)
{
 unsigned char *pixels = image.data;
 int rows = image.ysize;

char channel = -1;
float gain_r = 0.3, gain_g = 0.3, gain_b = 0.3, gain_a = 0.1;
float sum = 0.0, sum_x = 0.0, sum_y = 0.0;
 float /*blob_x = 0., blob_y = 0.,*/ blob_z = 0.;

 switch (m_method) {
 case 1:
 channel = chRed;
 break;
 case 2:
 channel = chGreen;
 break;
 case 3:
 channel = chBlue;
 break;
 case 4:
 channel = chAlpha;
 break;
 default:
 error("pix_blob: no method %d:: using GREY", m_method);
 case 0:
 gain_r = 0.3086; gain_g = 0.6094; gain_b = 0.082; gain_a = 0.0;
 break;
 case -1:
 gain_r = m_gain[chRed];
 gain_g = m_gain[chGreen];
 gain_b = m_gain[chBlue];
 gain_a = m_gain[chAlpha];
 }

 if (channel==-1){
 while (rows--) {
 int cols = image.xsize;
 while (cols--) {
 float val = gain_r * pixels[chRed] + gain_g * pixels[chGreen] +
 gain_b * pixels[chBlue] + gain_a * pixels[chAlpha];
 sum += val;
 sum_y += rows * val;
 sum_x += cols * val;
 pixels+=4;
 }
 }
 } else
 while (rows--) {
 int cols = image.xsize;
while (cols--) {
 int val = pixels[channel];
 sum += val;
 sum_y += rows * val;
 sum_x += cols * val;
 pixels+=4;
 }
 }



blob_z = sum;
 outlet_float(m_zOut, );
 if (sum) {
 outlet_float(m_yOut, 1 - sum_y/(image.ysize*sum));
 outlet_float(m_xOut, 1 - sum_x/(image.xsize*sum));
 }
}
void pix_blob :: processGrayImage(imageStruct &image)
```

```
{
 unsigned char *pixels = image.data;
 int rows = image.ysize;

 float sum = 0.0, sum_x = 0.0, sum_y = 0.0;
 float /*blob_x = 0., blob_y = 0.,*/ blob_z = 0.;
 while (rows--) {
 int cols = image.xsize;
 while (cols--) {
 int val = *pixels++;
 sum += val;
 sum_y += rows * val;
 sum_x += cols * val;
 }
 }

 blob_z = sum;
 outlet_float(m_zOut, sum/(image.xsize*image.ysize*255));
 if (sum) {
 outlet_float(m_yOut, 1 - sum_y/(image.ysize*sum));
 outlet_float(m_xOut, 1 - sum_x/(image.xsize*sum));
 }
}
void pix_blob :: processYUVImage(imageStruct &image)
{
 unsigned char *pixels = image.data;
 int rows = image.ysize;

 int sum = 0, sum_x = 0,sum_y = 0;
 while (rows--) {
 int cols = image.xsize;
 while (cols--) {
 int val = pixels[chY0];
 sum += val;
 sum_y += rows * val;
 sum_x += cols * val;
 pixels+=2;
 }
 }

outlet_float(m_zOut, (float)sum/(float)(image.xsize*image.ysize*255));
if (sum) {
 outlet_float(m_yOut, 1 - (float)sum_y/(float)(image.ysize*sum));
 outlet_float(m_xOut, 1 - (float)sum_x/(float)(image.xsize*sum));
 }
}

void pix_blob :: ChannelMess(int channel)
{
if (channel<0 || channel>4) {
 error("pix_blob: channel out of range");
 return;
 }
 m_method = channel;
}
void pix_blob :: GainMess(int argc, t_atom *argv)
{
 t_float gain = 0.0;
 m_gain[chAlpha]=0.0;
 int i;

switch (argc) {
 case 1:
 gain = atom_getfloat(argv);
 if (gain<=0) gain=1.0;
 m_gain[chRed]=m_gain[chGreen]=m_gain[chBlue]=gain;
 break;
 case 3:
 case 4:
 for (i=0; i<argc; i++){
```

```
 gain = atom_getfloat(argv++);
 m_gain[i]=(gain<0.)?0.:gain;
 }
 break;
 default:
 error("pix_blob: only 1, 3 or 4 gains are allowed");
 return;
 }

 m_method = -1;
}
```

pix_motionblur.cpp

```
#include "pix_motionblur.h"
/
pix_motionblur :: pix_motionblur()
{
 long size,src,i;

 inletmotionblur = inlet_new(this->x_obj, &this->x_obj->ob_pd,
 &s_float,
 gensym("motionblur"));

 m_blur0 = 256;
 m_blur1 = 0;

 m_savedImage.xsize=320;
 m_savedImage.ysize=240;
 m_savedImage.setCsizeByFormat(GL_RGBA);
 m_savedImage.reallocate();
 m_savedImage.setBlack();
}

pix_motionblur :: ~pix_motionblur()
{}

void pix_motionblur :: processRGBAImage(imageStruct &image)
{
 int h,w,height,width;
 long src=0;
 register int R,R1,G,G1,B,B1; //too many for x86? i really don't know or care
 int rightGain,imageGain;
 unsigned char *pixels=image.data;
 unsigned char *saved = m_savedImage.data;

 m_savedImage.xsize=image.xsize;
 m_savedImage.ysize=image.ysize;
 m_savedImage.setCsizeByFormat(image.format);
 m_savedImage.reallocate();
 if(saved!=m_savedImage.data)m_savedImage.setBlack();saved=m_savedImage.data;

 rightGain = m_blur1;
 imageGain = m_blur0;
 height = image.ysize;


 width = image.xsize;

 for (h=0; h<height; h++){
 for(w=0; w<width; w++){
 R = pixels[src+chRed];
 R1 = saved [src+chRed];
 G = pixels[src+chGreen];
 G1 = saved [src+chGreen];
 B = pixels[src+chBlue];
 B1 = saved [src+chBlue];
```

```
 R = R * imageGain;
 R1 = R1 * rightGain;
 G = G * imageGain;
 G1 = G1 * rightGain;
 B = B * imageGain;
 B1 = B1 * rightGain;

 R = R + R1;
 G = G + G1;
 B = B + B1;

 R1 = R>>8;
 G1 = G>>8;
 B1 = B>>8;

 saved[src+chRed] = (unsigned char)R1;
 saved[src+chGreen] = (unsigned char)G1;
 saved[src+chBlue] = (unsigned char)B1;

 pixels[src+chRed] = (unsigned char)R1;
 pixels[src+chGreen] = (unsigned char)G1;
 pixels[src+chBlue] = (unsigned char)B1;

 src += 4;
 }
 }
}
void pix_motionblur :: processGrayImage(imageStruct &image)
{
 int h,w,height,width;
 long src;
 register int G, G1; //too many for x86? i really don't know or care
 int rightGain,imageGain;
 unsigned char *pixels=image.data;
 int Gray;

 src = 0;
 Gray=chGray;

 unsigned char *saved = m_savedImage.data;

 m_savedImage.xsize=image.xsize;
 m_savedImage.ysize=image.ysize;
 m_savedImage.setCsizeByFormat(image.format);
 m_savedImage.reallocate();
 if(saved!=m_savedImage.data)m_savedImage.setBlack();saved=m_savedImage.data;

 rightGain = m_blur1;
 imageGain = m_blur0;
 height = image.ysize;
 width = image.xsize;

 for (h=0; h<height; h++){
 for(w=0; w<width; w++){
 G = pixels[src+chGray];



 G1 = saved[src+chGray];
 G = G * imageGain;
 G1 = G1 * rightGain;

 G = G + G1;
 G1 = G>>8;
 saved[src+chGray] = (unsigned char)G1;
 pixels[src+chGray] = (unsigned char)G1;
 src ++;
 }
 }
}
```

```
void pix_motionblur :: obj_setupCallback(t_class *classPtr)
{

class_addmethod(classPtr, (t_method)&pix_motionblur::motionblurCallback,
 gensym("motionblur"), A_GIMME, A_NULL);
}

void pix_motionblur :: motionblurCallback(void *data, t_symbol*, int argc, t_atom*argv)
{
 GetMyClass(data)->motionblurMessage(argc, argv);
}
```

pix_movement

```
#include "pix_movement.h"
#include <string.h>
#include <math.h>
#include <time.h>
#ifdef __APPLE__
# include <Carbon/Carbon.h>
#endif

CPPEXTERN_NEW_WITH_ONE_ARG(pix_movement,t_floatarg, A_DEFFLOAT)

pix_movement :: pix_movement(t_floatarg f)
{
 buffer.xsize = buffer.ysize = 64;
 buffer.format = GL_LUMINANCE;
buffer.csize = 1;
 buffer.reallocate();
 buffer2.xsize = buffer2.ysize = 64;
buffer2.format = GL_LUMINANCE;
 buffer2.csize = 1;
 buffer2.reallocate();

 if(f<=0.)f=0.5;
 if(f>1.f)f=1.0;
 treshold = (unsigned char)(255*f);
 inlet_new(this->x_obj, &this->x_obj->ob_pd, gensym("float"), gensym("tresh"));

 index = 0;
 averageTime = 0;
}

pix_movement :: ~pix_movement()
{
void pix_movement :: processRGBAImage(imageStruct &image)
{
bool doclear=(image.xsize*image.ysize != buffer.xsize*buffer.ysize);
 buffer.xsize = image.xsize;



 buffer.ysize = image.ysize;
 buffer.reallocate();
 if(doclear) buffer.setWhite();

 int pixsize = image.ysize * image.xsize;

 unsigned char *rp = image.data; // read pointer
 unsigned char *wp=buffer.data; // write pointer

1 while(pixsize--) {
 *wp++=(unsigned char)grey;
 rp+=4;
}
```

```
}
void pix_movement :: processYUVImage(imageStruct &image)
{
 int pixsize = image.ysize * image.xsize;

 int Y1, Y0;
 unsigned char thresh;

1 = chY1;
 Y0 = chY0;
thresh = treshold;

 unsigned char *rp = image.data; // read pointer
 unsigned char *wp=buffer.data; // write pointer to the copy
 unsigned char grey,grey1;

 grey = 0;
 grey1 = 0;
 pixsize/=2;
while(pixsize--) {
 grey = rp[Y0];
 rp[Y0]=255*(abs(grey-*wp)>thresh);
 *wp++=grey;

 grey1 = rp[Y1];
 rp[Y1]=255*(abs(grey1-*wp)>thresh);
 *wp++=grey1;

 rp[chU]=128;
 rp[chV]=128;
rp+=4;
 }
}

#ifdef __VEC__
oid pix_movement :: processYUVAltivec(imageStruct &image)
{
 if (image.xsize*image.ysize != buffer.xsize*buffer.ysize){
 buffer.xsize = image.xsize;
 buffer.ysize = image.ysize;
 buffer.data = new unsigned char [buffer.xsize*buffer.ysize*2];
 }

union{
 signed short c[8];
 vector signed short v;
 }shortBuffer;

 int pixsize = image.ysize * image.xsize/8;

 int i;
 vector signed short thresh;



 shortBuffer.c[0] = treshold;
 thresh = shortBuffer.v;
 thresh = (vector signed short)vec_splat(thresh,0);

vector unsigned char *rp = (vector unsigned char *) image.data; // read pointer
vector unsigned char *wp = (vector unsigned char *) buffer.data; // write pointer to the copy
vector unsigned char grey,grey1;
 vector unsigned char one = vec_splat_u8(1);
 vector unsigned short Y,UV,Ywp,UVwp,hiImage,loImage;
 vector unsigned short Y1,UV1,Ywp1,UVwp1,hiImage1,loImage1;
 vector signed short temp,temp1;
 vec_dst( rp, prefetchSize, 0 );
 vec_dst( wp, prefetchSize, 1 );
 #endif
```

pix_videoDS

```
#ifndef INCLUDE_PIX_VIDEO_H_
#define INCLUDE_PIX_VIDEO_H_
#include "Base/config.h"

#if defined VIDEO_NEW || defined HAVE_DIRECTSHOW
# ifndef DO_AUTO_REGISTER_CLASS
# define NO_AUTO_REGISTER_CLASS
# endif
#endif

#include "Base/GemBase.h"
#include "Base/GemPixUtil.h"

class GEM_EXTERN pix_video : public GemBase
{
 CPPEXTERN_HEADER(pix_video, GemBase)

 public:
 pix_video(t_floatarg num = 0);

 protected:

 virtual ~pix_video();


 virtual void render(GemState *state);

 virtual void postrender(GemState *state);


 virtual void stopRendering();

 void cleanPixBlock();

 virtual void dimenMess(int x, int y, int leftmargin = 0, int rightmargin = 0
 int topmargin = 0 , int bottommargin = 0) {}

virtual void offsetMess(int x, int y);


 virtual int startTransfer();

virtual int stopTransfer();

virtual void swapMess(int state);

virtual void enumerateMess();
virtual void csMess(int format);
virtual void dialogMess(int,t_atom*);

nt m_haveVideo;
 pixBlock m_pixBlock;
 imageStruct m_imageStruct;
int m_swap;
 int m_colorSwap;

 private:
static void dimenMessCallback(void *data, t_symbol *s, int ac, t_atom *av);
static void offsetMessCallback(void *data, t_floatarg x, t_floatarg y);
static void swapMessCallback(void *data, t_floatarg state);

static void dialogMessCallback(void *data, t_symbol*,int,t_atom*);
 static void enumerateMessCallback(void *data);
static void csMessCallback(void *data, t_symbol*colorspace);
};

#endif // for header file
```