Interacción basada en visión por computador mediante reacTIVision para la generación, control e interpretación de mensajes MIDI y su utilización en la creación audiovisual.

Proyecto de Fin de Carrera.

Autor: Javier Peña Suárez
Alumno de Escuela de Ingeniería Informática de la ULPGC

Tutor: Modesto Castrillón Santana
Profesor de la ULPGC del área de Conocimiento
de Ciencia de la Computación e Inteligencia Artificial.

1 de octubre de 2012

Índice general

Índice de figuras

Agradecimientos

*En primer lugar quiero agradecer a mi tutor **Modesto Castrillón Santana** la ayuda, el apoyo y la confianza que me ha prestado durante la realización de este proyecto. Trabajar en un entorno como el laboratorio de Robótica y Oceanografía Computacional me ha reportado una experiencia muy positiva e inolvidable. Ha sido precisamente en este lugar donde he conocido a otras grandes personas como **Jorge Cabrera Gámez, Antonio Domínguez Brito y José Daniel Hernández Sosa** que gracias a sus conocimientos me han dado la oportunidad de desarrollarme, tanto intelectualmente como personalmente.*

*Tampoco me olvido de mis amigos, esos que me han acompañado y apoyado desde el primer día que me aventuré en este proyecto y con los que he compartido alegrías, discusiones, esperanzas y sueños. **Jose**, muchísimas gracias por todos tus consejos y todo ese trabajo gráfico que has realizado en torno a Logywave, eres una persona muy grande a la que respeto y admiro, nunca dejarás de sorprenderme. **Joaquín** cada día que pasa aprendo más de tí, gracias por todo lo que has compartido conmigo a lo largo de estos años y por confiar en mí y en mi trabajo, te prometo que el día en el que el éxito se presente ante mí no le tendré miedo y le plantaré cara. **Eugenio**, te agradezco todo lo que has hecho por mí y por este trabajo, de tí he aprendido la constancia del día a día y el empeño que le dedicas a las cosas hasta que salen. Juntos formamos algo muy grande y una parte de este proyecto también es de ustedes.*

*Mención especial merece **Jorge Herrera**, sin tí este proyecto nunca hubiese visto la luz, después de todas las discusiones, risas, temores y miedos, nuestro sueño de formar una marca alrededor de todo esto se hará realidad tarde o temprano, muchísimas gracias por tu tiempo y dedicación, eres una persona que vale mucho tanto en el terreno profesional como en el personal.*

*Y, por supuesto, tengo que agradecer, no solo este proyecto, sino todo lo que han hecho por mí, el amor que siempre me han dado y creer en todas mis ideas, a mi familia, mis padres **Fernando** y **María del Pino** y mi hermano **Fernando**. Sin ellos no estaría aquí ni hubiera llegado donde estoy. Gracias por permitir que me haya formado como persona y poder seguir ascendiendo en la vida.*

*Por último me queda alguien muy especial, **Lilia**, gracias por todo el apoyo, la compresión y el cariño que me has dado a lo largo de este trabajo, cuando te conocí esto tan solo era una idea y toda la energía que me transmites ha hecho que sea algo más que eso. Has comprendido, en innumerables ocasiones, mi ausencia por estar desarrollando y te agradezco toda tu ayuda en este viaje, así como enseñarme un mundo que no conocía antes.*

Capítulo 1

Introducción

Uno de los últimos avances en tecnología que más interés está despertando es la tecnología *multitouch*. Esta técnica ofrece una nueva forma de interacción hombre-máquina donde las aplicaciones gráficas son manejadas usando eventos táctiles. Esto abre la puerta a una comunicación más intuitiva y más directa entre el usuario y la máquina. Para lograr esto, son necesarios dos elementos: una superficie multitouch y una interfaz adaptada para el uso con eventos táctiles y objetos físicos que tendrán unos marcadores asignados.

En la actualidad hay varias técnicas para la detección de múltiples eventos táctiles sobre una superficie y los movimientos que realizan, como son los sensores térmicos, de presión, scanner, detección de sombras, etc. Por lo general estas técnicas no están al alcance de todos, dado que es una tecnología muy reciente y ciertos sistemas tienen un coste de momento muy elevado.

Con este proyecto buscamos desarrollar un dispositivo multitouch y un software para la creación de música digital mediante eventos táctiles y objetos. Para ello, es necesario investigar en distintos campos como son el del procesamiento de imágenes, hardware y desarrollo de aplicaciones e interfaces gráficas.

1.1. Descripción

A lo largo de la historia del arte, diversas disciplinas han hecho uso de la representación bidimensional para mostrar sus propuestas creativas. El ordenador es en la actualidad una herramienta de enorme potencial para el arte, no sólo en el contexto de la imagen bi o tridimensional, sino en cualquier situación en la que se haga uso de información.

El proyecto se enmarca dentro del contexto de la aplicación de las nuevas tecnologías para la creación de una interfaz audiovisual capaz de interpretar y controlar mensajes MIDI [C] y señales de audio. Se logrará mostrar las posibilidades que tecnologías como la visión artificial y el control de la señal de audio

pueden aportar en el proceso creativo.

La idea es la de ofrecer la flexibilidad, la riqueza y el potencial de la síntesis digital, de una forma accesible, inmersa y relajada. La interfaz combina un acceso inmediato e intuitivo con unas posibilidades de aprendizaje y dominio casi infinitas, lo que le hace apropiado tanto para usuarios noveles, incluso niños, por ejemplo en una instalación, así como para músicos avanzados o Dj's en concierto.

Bajo estas consideraciones, en los últimos años han surgido no sólo diversas aplicaciones de la tecnología en el proceso creativo sino también diversas herramientas para la producción creativa audiovisual Processing [6], para el uso de la tecnología de interacción en la producción sonora con realimentación visual Reactable [4], su combinación con superficies multitáctiles TUIO [5], así como la aplicación y el uso de técnicas de visión e instrumentación como sensores, p.e. Silent Drum [7], o de análisis del patrón sonoro para el acompañamiento Pandeiro funk.

1.2. Objetivos

Los objetivos de este proyecto se centran en la integración de nuevas tecnologías en la producción creativa audiovisual.

Los objetivos detallados del proyecto son:

- Interpretar mensajes Audio/MIDI a partir de la percepción basada en visión por computador.
- Desarrollar interfaces para facilitar la definición de ficheros de configuración Audio/MIDI.
- Estudiar posibilidades de generación automática de contenidos visuales sincronizados con los comandos OSC [28].
- Desarrollar interfaces basadas en el uso de superficies multitouchs con reconocimiento de Objetos.
- Realizar un prototipo para la iniciación didáctica a la producción musical.

1.3. Metodología

Para las etapas de análisis y desarrollo se hará uso de las herramientas aprendidas en ingeniería del software.

En cuanto al análisis, lo enfocaremos en un entorno orientado a objetos con UML [B], el cual comprende las etapas de análisis de requisitos de usuario y análisis de requisitos de software.

En la etapa de desarrollo se optará por el desarrollo con prototipación que pertenece a los modelos de desarrollo evolutivo, se inicia con la definición de los objetivos globales para el software, luego se identifican los requisitos conocidos y las áreas del esquema en donde es necesaria más definición. Entonces se plantea con rapidez una iteración de construcción de prototipos y se presenta el modelado (en forma de un diseño rápido).

1.4. Recursos Necesarios

Para la realización de este proyecto se hará uso de:

- Un ordenador Personal Multimedia.
- ReacTivision[3].Un framework de visión por computador multiplataforma y de código abierto para el seguimiento rápido y robusto de marcadores fiduciarios agregados a objetos físicos.
- TUIO [5]. Un framework que define un protocolo común y API para superficies multitouch tangibles.
- Un lenguaje de programación. En este caso Processing.
- Una Cámara web.
- Iluminación Infrarroja.
- Metacrilato.
- Capa de Difusión para la Superficie de Proyección.
- Ableton Live!. Secuenciador de Audio/MIDI.
- Pure Data [9]. Un lenguaje de programación gráfico para la creación de música computerizada interactiva y obras multimedia.

1.5. Temporización

El proyecto se divide en las siguientes etapas:

- Etapa de análisis
  - Se llevará a cabo el análisis de los requisitos de usuario y de software así como el de las herramientas de desarrollo necesarias.

    - Duración estimada: 150 horas.

- Etapa de diseño
    - Se llevará a cabo el diseño de la implementación a realizar según los resultados obtenidos en la etapa anterior.
    - Duración estimada: 200 horas.
- Etapa de implementación
    - Se llevará a cabo la implementación del software diseñado en la etapa anterior.
    - Duración estimada: 450 horas.
- Etapa de experimentación y validación
    - Se llevará a cabo la validación y el testeo del software implementado en la etapa anterior así como una documentación que sea útil a nivel de usuario de la aplicación, para así facilitar y favorecer su utilización.
    - Duración estimada: 180 horas.

En total se estima que la duración de todas las etapas sea de 980 horas.

En las siguientes secciones se detallará cada una de estas etapas.

Capítulo 2

Análisis

2.1. Estudio de las herramientas existentes

El proyecto actual parte de la idea de la mesa tangible Reactable [4] desarrollado por el Grupo de Tecnología Musical de la Universidad Pompeu Fabra de Barcelona. Múltiples usuarios simultáneos comparten el control total del instrumento moviendo y rotando objetos físicos sobre la superficie de una mesa circular luminosa. Manipulando dichos objetos, los cuales representan los componentes clásicos de un sintetizador modular, los usuarios pueden crear tipologías sonoras complejas y dinámicas, mediante generadores, filtros y moduladores, en una clase de sintetizador modular tangible.

La Reactable consta de un tablero semi translúcido iluminado directamente con dos cámaras situadas al otro lado del tablero que analiza de vez en cuando la superficie y sigue los movimientos, la naturaleza, la posición y la orientación de los diferentes objetos físicos y lógicos que están situados sobre el tablero por medio de visión artificial. Varios músicos simultáneos comparten control completo sobre el instrumento moviendo y rotando los objetos en el tablero luminoso. Al mover y relacionar los objetos por la superficie del tablero se modifica la estructura y los parámetros del sintetizador de sonido. Estos objetos conforman los típicos módulos de un sintetizador modular. Simultáneamente, el proyector muestra la actividad y las características principales del sonido producido, otorgándole de esta forma la necesaria retroalimentación al ejecutante. De esta forma varios músicos pueden compartir el control desplazando y rotando fichas transparentes sobre la mesa luminosa. Cada uno de los usuarios puede crear una función sonora diferente.

2.1.1. ReacTIVision

ReacTIVision [3], el software detrás de Reactable, es un sistema de visión por computador de código abierto y multiplataforma (no la parte de audio, si la de vídeo) pensado para un rastreo rápido y estable

tanto de marcas asignadas a objetos físicos como de acciones multitouchs.

Fue diseñado principalmente como un conjunto de herramientas para el rápido desarrollo de interfaces tangibles basadas en mesas y superficies interactivas multitouchs.

El sistema fue desarrollado por Martin Kaltenbrunner y Ross Bencina en el Music Technology Group de la Universitat Pompeu Fabra en Barcelona, España, como parte del proyecto Reactable.

ReacTIVision es una aplicación independiente, que envía mensajes TUIO [5] a través del puerto UDP 3333 a cualquier aplicación con un cliente TUIO habilitado.

El protocolo TUIO fue diseñado inicialmente dentro de este proyecto para codificar el estado de objetos tangibles y eventos multitouchs desde una superficie interactiva. Este marco también incluye un conjunto de ejemplos de clientes TUIO para varios lenguajes de programación, que servirá de base para el desarrollo de la interfaz de las aplicaciones de usuario tangibles. Alternativamente ReacTIVision también es capaz de enviar mensajes MIDI para la utilización directa con secuenciadores MIDI.

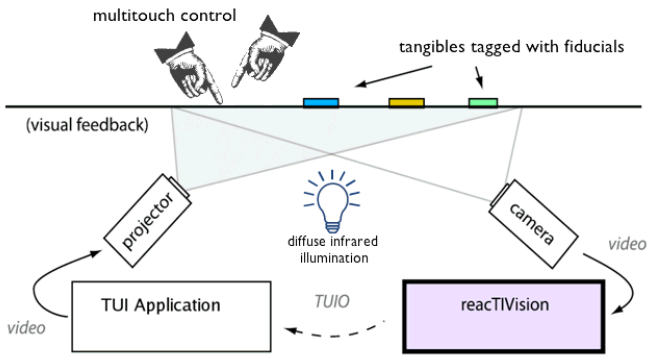


Figura 2.1: Diagrama del funcionamiento de ReacTIVision en un tabletop.
Figura extraída de [3]

La aplicación ReacTIVision está disponible para diversos sistemas operativos: Windows, MacOS X y Linux. Bajo Windows es compatible con cualquier cámara con un controlador adecuado WDM, tales como USB, USB2 y FireWire y cámaras DV. Igualmente bajo MacOS X todas las cámaras FireWire y cualquier cámara compatible con QuickTime trabajará en ReacTIVision. Bajo Linux, las cámaras que soportan FireWire están mejor sustentadas, así como cámaras USB Video4Linux2.

En resumen, ReacTIVision ofrece un mejor rendimiento con el uso de cámaras IEEE1394 en MacOS X o Linux, en los sistemas Win32 WDM la mayoría de las cámaras deben trabajar, incluyendo cámaras IEEE 1394 con un controlador WDM. Algunas cámaras USB en MacOS X se puede utilizar con el

paquete de controladores macam, aunque la captura de vídeo QuickTime es significativamente más lenta que haciendo uso del protocolo IEEE1394.

2.1.2. TUIO

TUIO [5] es un framework que define un protocolo común y API para superficies multitouchs tangibles. El protocolo TUIO permite la transmisión de una descripción abstracta de superficies interactivas, incluyendo el estado de eventos táctiles y objetos tangibles. Este protocolo codifica los datos de control desde una aplicación de seguimiento (por ejemplo, basada en la visión por computador) y lo envía a cualquier aplicación cliente que es capaz de decodificar el protocolo. Existe un número creciente de aplicaciones TUIO y librerías de clientes TUIO para varios entornos de programación, así como las aplicaciones que soportan el protocolo.

Esta combinación de aplicaciones de seguimiento TUIO, protocolos y las implementaciones del cliente permiten el desarrollo rápido de una mesa multitouch. TUIO ha sido principalmente diseñado como una abstracción para superficies interactivas, pero también se ha utilizado en muchas otras áreas de aplicación relacionadas. Técnicamente TUIO se basa en Open Sound Control (OSC) [28] un nuevo estándar para entornos interactivos, no sólo se limita al control de instrumentos musicales y por tanto se puede implementar fácilmente sobre cualquier plataforma que soporte OSC.

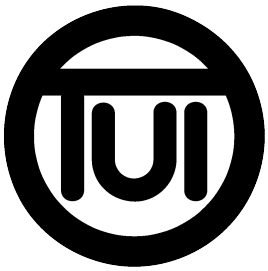


Figura 2.2: TUIO.
Figura extraída de [5]

Desde la publicación inicial de la especificación del protocolo TUIO al dominio público como parte del sintetizador Reactable, después de su puesta en práctica por primera vez en ReacTIVision, el protocolo ha sido adoptado también por muchos proyectos relacionados con materiales e interacción multitouch, como el NUI Group y varias plataformas de interacción tangible. Debido a su adopción generalizada, el protocolo TUIO a día de hoy se puede considerar un standard.

2.1.3. Ableton Live!

Ableton Live! [8] es un secuenciador audio y MIDI, aplicación también conocida como DAW (Digital Audio Workstation) para los sistema operativos Windows y Mac OS X.

Figura 2.3: Ableton live!.
Figura extraída de [8]

Ableton Live! está pensado tanto para la composición musical como para la música en directo. Su interfaz de usuario consiste en una sola ventana con diferentes secciones. La sección principal se divide en dos tipos de vistas. La primera sirve para disparar en cada pista fragmentos de audio o MIDI llamados clips. Su objetivo es realizar sesiones en directo o grabaciones improvisadas.

Figura 2.4: Clips de Audio/MIDI en Ableton Live!.

La segunda vista muestra una secuencia en una regla de tiempo al estilo de un secuenciador tradicional.

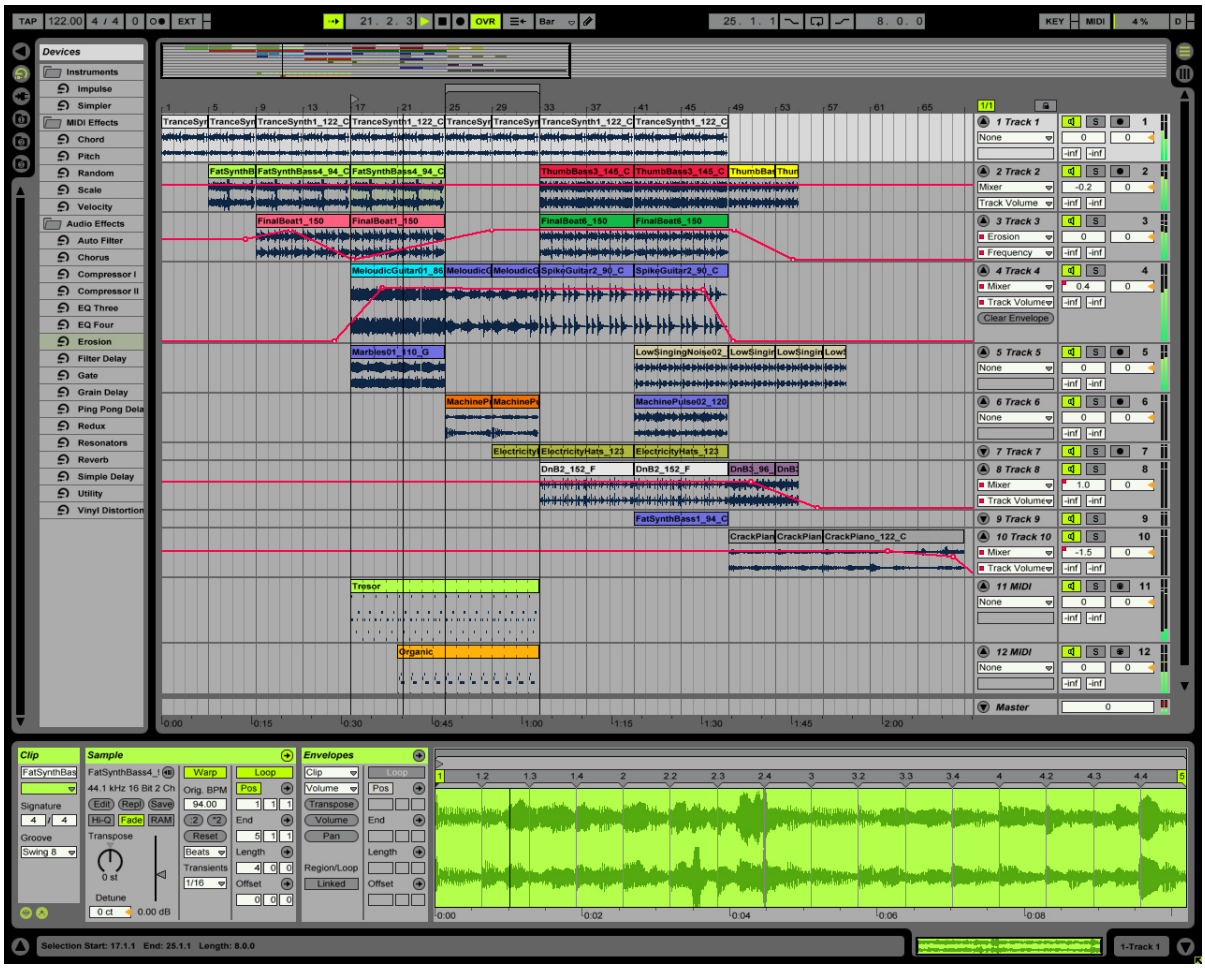


Figura 2.5: Secuenciador en Ableton Live!.

2.1.3.1. Características

- Grabación multipista de hasta 32-bit/192kHz.
- Edición no destructiva con deshacer ilimitado.
- Secuenciación de instrumentos MIDI hardware y software.
- Estiramiento de tiempo de archivos AIFF, WAV, Ogg Vorbis, FLAC y MP3 para improvisación y remezcla.
- Varios efectos de audio incorporados como retardos, filtros, distorsiones, compresores y ecualizadores.

- Incluye instrumentos de software basados en muestras.
- Agrupación de instrumentos, baterías y efectos en una pista para la creación de configuraciones más complejas.
- Soporte para efectos e instrumentos VST y AU con compensación de retardos.
- Soporte para archivos REX.
- Control de parámetros a tiempo real con un controlador MIDI.
- Soporte ReWire.
- Interfaz de usuario basado en una sola ventana.
- Soporte para multiprocesador y multinúcleo.

2.1.4. Processing

Processing [6] es un lenguaje y entorno de programación de código abierto basado en Java, de fácil utilización, y que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital. Fue iniciado por Ben Fry y Casey Reas a partir de reflexiones en el Aesthetics and Computation Group del MIT Media Lab.

Processing es desarrollado por artistas y diseñadores como una herramienta alternativa al software propietario. Puede ser utilizado tanto para aplicaciones locales así como aplicaciones para la web.

Figura 2.6: Processing.
Figura extraída de [6]

A continuación se exponen las librerías utilizadas en Processing para el control del audio y del protocolo MIDI.

2.1.4.1. proMIDI 2.0

La biblioteca proMIDI [10] permite a Processing y cualquier aplicación enviar y recibir comandos MIDI. Mididata puede ser recibido y enviado por los puertos MIDI instalados.

La intención de esta biblioteca fue para sincronizar Processing via MIDI y utilizarlo para efectos visuales en vivo. Otra posibilidad es la generación de datos MIDI para hacer sonido para el procesamiento visual.

Figura 2.7: MIDI.

Existen muchos cambios en la versión proMIDI nueva para el control de los puertos MIDI IN. Ya no es necesario tener cuidado para el envío de la nota off, en lugar de crear notas con una longitud, las notas se almacenan y proMIDI envía automáticamente la nota. También la apertura de entradas y salidas ha cambiado. Ahora se puede abrir sólo un determinado canal de un puerto MIDI.

El cambio más importante sin embargo es que proMIDI incluye ahora un secuenciador así como pistas y patrones para crear estructuras musicales.

En el apéndice C se encuentra toda la información acerca del funcionamiento de los mensajes MIDI, así como varios ejemplos de como se analizan estos mensajes.

2.1.4.2. OscP5

OscP5 [27] es una librería que implementa los comandos del Protocolo OSC para ser utilizados en el entorno de programación Processing. OSC [28] es el acrónimo de Open Sound Control, un protocolo de comunicación entre ordenadores, sintetizadores de sonido y otros dispositivos multimedia optimizado para la tecnología de red actual.

Características:

- **Detección automática de eventos:** oscP5 localiza las funciones en nuestro programa base y las vinculará con los mensajes OSC entrantes gracias al comando oscP5Plug.
- **Protocolos de red:** oscP5 soporta TCP, UDP y multicast. Los comandos de red se manejan con el paquete netP5 que también puede ser utilizado para las operaciones de diferentes comandos de red que no requieran OSC como protocolo.

Este protocolo se ha utilizado en muchas áreas de aplicación. Para más información sobre el funcionamiento del protocolo OSC y los tipos de datos que se pueden enviar, se encuentra detallada toda la información en el apéndice D.

2.1.5. Pure Data

Pure Data (o Pd) [9] es un lenguaje de programación gráfico desarrollado por Miller Puckette durante los años 90 para la creación de música computerizada interactiva y obras de multimedia. Aunque Puckette es el principal autor del software, Pd es un proyecto de código abierto y tiene una gran base de desarrolladores trabajando en nuevas extensiones al programa. Está publicado bajo una licencia similar a la licencia BSD.

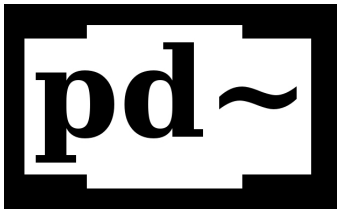


Figura 2.8: Pure Data.
Figura extraída de [9]

Pd es muy similar en alcance y diseño al programa original de Puckette, Max (desarrollado cuando él estaba en IRCAM), y es hasta cierto grado interoperable con Max/MSP, el sucesor comercial del lenguaje Max. Ambos Pd y Max son ejemplos discutibles de lenguajes de programación de flujo de datos. En este tipo de lenguajes, funciones u objetos son conectadas o parchadas unas con otras en un ambiente gráfico que modela el flujo del control y el audio. A diferencia de la versión original de Max, sin embargo, Pd siempre fue diseñado para hacer procesado de señales y tasas de control en la CPU nativa, en vez de descargar la síntesis y el proceso de señales a un tablero de PDS (como el Ariel ISPW que era usado para Max/FTS). El código de Pd es la base de las extensiones MSP de David Zicarelli al lenguaje Max para hacer proceso de audio en software.

Como Max, Pd tiene una base modular de código con externos u objetos que son utilizados como bloques de construcción para programas escritos en el software. Esto hace el programa arbitrariamente extensible a través de una API pública, y alienta a los desarrolladores a añadir sus propias rutinas de audio y control, ya sea en el lenguaje de programación C o, con la ayuda de otros externos, en Python, Javascript, Ruby, y potencialmente otros lenguajes también. Sin embargo, Pd es un lenguaje de programación en sí mismo. Unidades de código modulares y reusables, escritas nativamente en Pd, llamadas parches o abstracciones, son usadas como programas independientes y compartidas libremente entre la comunidad de usuarios de Pd, y ninguna otra habilidad de programación es requerida para usar Pd pero ayuda.

Con la adición del externo Entorno Gráfico para Multimedia (GEM, por su nombre en inglés), y otros externos diseñados para trabajar con él (como Pure Data Packet, PiDiP para Linux, framestein para Windows, GridFlow para proceso de matrices n-dimensionales que integra Pure Data con el lenguaje de programación Ruby, etc.), es posible crear y manipular vídeo, gráficos OpenGL, imágenes, etc, en tiempo real con aparentemente infinitas posibilidades de interactividad con audio, sensores externos, etc.

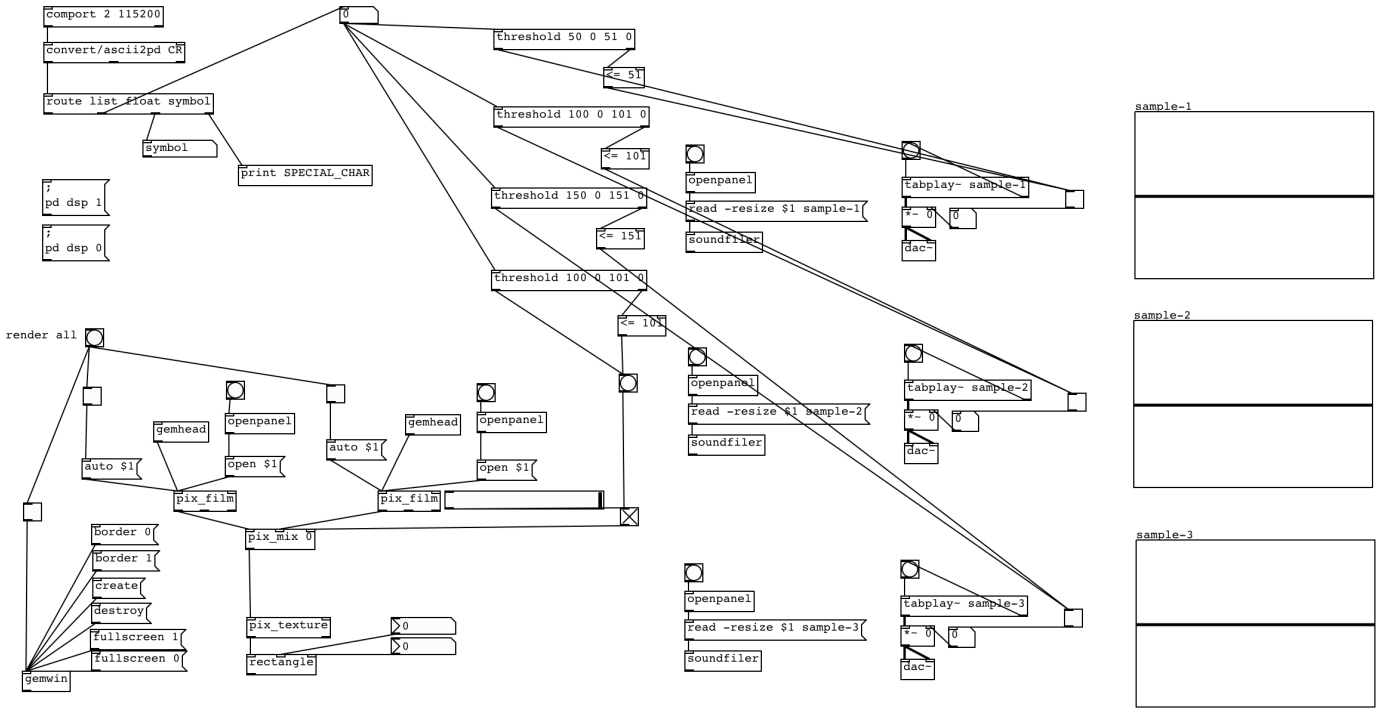


Figura 2.9: Conexiones en Pure Data.

Adicionalmente, Pd está diseñado nativamente para permitir colaboración en vivo a través de redes o de Internet, permitiendo a músicos conectados vía LAN, o incluso en distintas partes del mundo, hacer música juntos en tiempo real.

Las unidades donde se programa el código se llaman patch o abstracciones, son utilizadas como programas independientes y compartidos libremente entre la comunidad de usuarios de Pd. Los patchs constan de diferentes objetos interconectados entre ellos. En su parte superior encontraremos las entradas, donde se les enviaran valores numéricos u otros tipos de datos, y en la posterior la salida de estos.

2.1.6. Jack OS X

Jack [11] (Jack Audio Connection Kit) es un servidor de audio de baja latencia, escrito originalmente para el sistema operativo GNU/Linux actualmente da soporte para el sistema operativo Mac OS X. Permite conectar cualquier aplicación de audio a un dispositivo de audio virtual enrutando una sola salida de sonido en varios canales de audio por separado.

Figura 2.10: Jack OS X.
Figura extraída de [11]

Jack es la mejor opción a utilizar como servidor de audio, ya que ha sido diseñado desde cero para trabajar con audio profesional. Esto significa que se centra en dos aspectos fundamentales:

- Ejecución síncrona de todos los clientes.
- Funcionamiento de baja latencia.

El paquete Jack OS X incluye:

- **Servidor Jack:** La infraestructura necesaria para usar Jack.
- **JackRouter:** Es un driver CoreAudio que permite convertir cualquier aplicación de audio en un cliente Jack.
- **Plugins:** El paquete incluye un plugin VST y un plugin AU para ampliar las posibilidades ilimitadas de enrutamiento de audio cuando se usa Jack.
- **JackPilot:** Es la aplicación que ofrece un entorno gráfico para controlar el servidor Jack y permite enrutar las diferentes conexiones de audio entre las aplicaciones cliente.

2.2. Estudio de las herramientas alternativas

Existen multitud de herramientas alternativas que cumplen la misma finalidad que las anteriores, pero con características algo diferentes.

Sin embargo también hay que considerar la larga trayectoria de estas herramientas alternativas, ya que o bien han sido la base de las herramientas actuales o han incidido en una muy correcta implementación del software en muchos de sus aspectos funcionales.

2.2.1. D-touch

Tanto ReacTIVision como D-touch [25] son sistemas para el seguimiento de la ubicación y la orientación de marcadores (fiducials) en tiempo real.

ReacTIVision surge como una continuación de D-touch principalmente debido a los bajos tiempos de los frame rates alcanzados con D-touch.

D-touch utiliza una topología única para todos los fiduciales en un set. El set consta de 120 fiduciales únicos que se diferencian mediante un código de permutación expresado por el número de hojas negras en el grafo de adyacencia. En el siguiente ejemplo el código de permutación es (1,2,3,6,5,4).

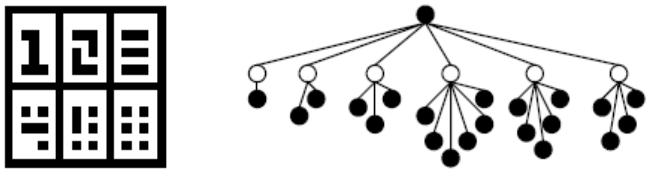


Figura 2.11: Ejemplo código de permutación.
Figura extraída de [25]

D-touch asocia el código de permutación de las regiones de hojas negras a un conjunto específico de fiduciales.En segundo lugar, el D-touch original no prescribe un determinado método para calcular la ubicación y orientación de los fiduciales.

Por último, las geometrías simples de los fiduciales (que por cierto tienen la ventaja de ser fácilmente dibujados a mano) no fueron diseñados para minimizar el tamaño de los mismos.

Por otra parte el sistema de ReacTIVision admite conjuntos de fiduciales de distintos tamaños y topologías sin ningún cambio de código. Consideramos que esta es una mejora significativa respecto a D-touch en nuestro contexto, donde queremos experimentar con fiduciales más pequeños.

El set consta de 128 fiduciales exclusivos, logrando tener una superficie del casi 50 % menos que las del set de 120 fiduciales de D-touch.

A continuación mostramos una comparación de performance entre las diferentes librerías.

Library	CPU Usage	Frame Rate
libfidtrack	18%	30 fps
libdtouch	82%	30 fps
dtouch_old	100%	10 fps

Figura 2.12: Comparativa entre D-Touch y ReacTIVision.
Figura extraída de [15]

La figura 2.12 compara el uso de la CPU y el frame rate obtenido con reactTIVision $libfidtrack$ y dos versiones de d-touch: una versión reciente (libdtouch) y la versión original.

Podemos ver que la actual implementación de ReacTIVision es 4 veces mas rápido que la versión actual de D-touch y mas de 16 veces mas rápido que la versión original.

Podemos resumir entonces las mejoras conseguidas por ReacTIVision frente a D-touch en tres áreas:

- Funcionamiento
- Tamaño de los fiduciales
- Escalabilidad para diferentes topologías de fiduciales

2.2.2. Community Core Vision

Community Core Vision [12] o CCV es un framework de código abierto de visión por computador desarrollado por el NUI Group [19]. Como ReacTIVision toma imágenes con una cámara y las analiza para detectar objetos en la superficie de un display multitouch. A diferencia del framework utilizado en este proyecto, CCV se centra en la detección de eventos táctiles como el tocar la superficie o deslizar los dedos por encima del display multitouch. También implementa el reconocimiento de objetos sin forma definida así como el reconocimiento de objetos identificados con fiduciales, sin embargo este tipo de reconocimiento no está todavía suficientemente desarrollado. Al igual que ReacTIVision puede comunicar los datos extraídos a otras aplicaciones a través de TUIO.

Como se puede ver en la figura 2.13 la ventaja que CCV ofrece frente a ReacTIVision es la cuidada interfaz de usuario de la que viene provista y fácil calibración, siendo esta la única ventaja frente a ReacTIVision.

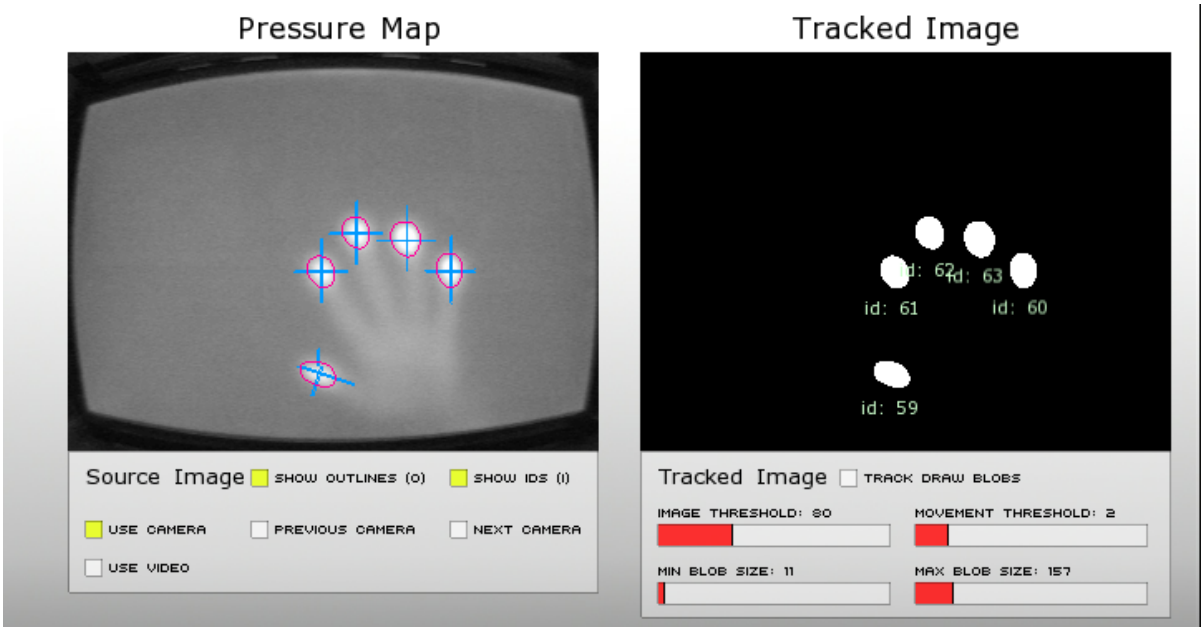


Figura 2.13: Interfaz de usuario de CCV.
Figura extraída de [19]

2.2.3. MAX/MSP

Max [13] es un entorno de desarrollo gráfico para música y multimedia desarrollado y mantenido por Cycling '74, una empresa de programas situada en San Francisco. El programa ha sido usado durante más de quince años por compositores, artistas y diseñadores de programas interesados en la creación de programas interactivos.

Figura 2.14: MAX/MSP.
Figura extraída de [13]

Max es bastante modular, y la mayoría de las rutinas forman parte de una biblioteca compartida. La IPA (Interfaz de Programación de Aplicaciones) permite el desarrollo de nuevas rutinas (llamadas objetos

externos) por terceras personas. Por consecuencia, muchos de los usuarios de Max son programadores no afiliados a Cycling '74 que mejoran el programa, creándole extensiones comerciales y no comerciales. Debido a su diseño extensible e interfaz gráfica (que de manera novedosa representa la estructura del programa y el IGU presentadas simultáneamente al usuario), Max es considerado por muchos como la lengua franca para el desarrollo de programas de música interactiva.

Max tiene numerosas extensiones y encarnaciones; en particular, las extensiones de audio que aparecieron en 1997, trasladadas de Pure Data. Estas fueron llamadas MSP (las iniciales de Miller S. Puckette, autor de Max y Pd). Estas adiciones para Max permitieron que el audio digital sea manipulada en tiempo real, y a la vez, permitía a los usuarios la creación de sus propios sintetizadores y efectos-procesadores. Previamente, Max había sido diseñado con el fin de tener un terreno común con sintetizadores reales, samplers, etcétera, como un lenguaje de control usando MIDI o algún otro protocolo de red.

2.2.4. Soundflower

Soundflower [29] es una extensión del sistema OS X que permite al usuario enrutar el sonido entre aplicaciones OS X Core Audio. También es posible hacer el seguimiento de las conexiones con la aplicación gratuita Soundflowerbed [29].

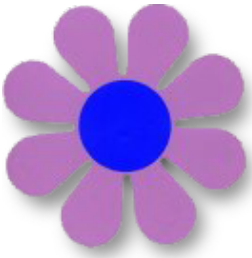


Figura 2.15: Soundflower.
Figura extraída de [29]

Soundflower puede usarse para tareas de audio con 2 ó 16 canales de datos con muestras de 32-bit. Para configurarlo, simplemente hay que entrar en Preferencias del sistema, sonido y elegir la salida virtual y la entrada virtual que queramos y/o en la configuración de la aplicación en cuestión.

EL principal inconveniente es que sólo permite enrutar 16 canales de audio como máximo a diferencia de JACK que nos permite enrutar tantos canales de audio como necesitemos.

2.2.5. LoopBe1

LoopBe1 [24] es un dispositivo virtual MIDI para transferir los datos MIDI entre los programas de ordenador. Básicamente LoopBe1 es un cable invisible para conectar un puerto de salida MIDI de una aplicación hacia un puerto de entrada de cualquier otra aplicación MIDI. Todos los datos MIDI enviados a la salida del programa se canalizan a las solicitudes que reciben las aplicaciones en tiempo real. Se pueden conectar hasta 8 aplicaciones para el puerto de entrada y hasta 8 aplicaciones al puerto de salida.

LoopBe1 tiene un acceso directo de detección de gran alcance para evitar la realimentación MIDI. Si una retroalimentación se detecta, inmediatamente LoopBe1 silenciar su puerto, interrumpir el bucle y avisa con un mensaje emergente.

2.2.6. Sonia Sound Library

Sonia [23] es una librería externa (API) para la plataforma Processing y puede también incluirse en cualquier proyecto Java.

La Biblioteca Sonia proporciona capacidades avanzadas de audio tales como reproducción de la muestra, síntesis de sonido en tiempo real, análisis FFT (frecuencia), anális de la entrada de micrófono en tiempo real, y la escritura de archivos .wav a partir de muestras.

Sonia utiliza el plugin JSyn creado por Phil Burk. Esto significa que los usuarios necesitarán instalar el plugin JSyn en su navegador para poder ver los proyectos de Sonia en línea.

2.3. Pantallas táctiles

Las pantallas táctiles son dispositivos electrónicos que actúan como periféricos de entrada y salida ya que permiten la entrada de datos mediante el contacto de dedos u objetos pasivos (como un lápiz) sobre su superficie y, además, actuúan como pantalla.

Las principales características de una pantalla táctil son:

- Permiten interactuar directamente sobre lo que se muestra al usuario, en vez de hacerlo indirectamente mediante un cursor al utilizar un ratón o touchpad.
- No es necesario que el usuario utilice dispositivos adicionales.

Desde su invención en 1971 por el Dr. Samuel C. Hurst, las pantallas táctiles se han ido haciendo cada vez más populares. Empezaron siendo comunes en TPV's, cajeros automáticos y PDA's y, actualmente,

también las podemos encontrar en los teléfonos inteligentes, vídeo consolas portátiles y navegadores GPS.

Hasta hace relativamente poco, las pantallas eran únicamente capaces de percibir un punto de contacto a la vez, sin embargo esto está cambiando debido a la comercialización de pantallas multitouchs.

En este apartado se describen las principales tecnologías en las que se basan las pantallas táctiles así como ejemplos de dispositivos que las utilizan.

2.3.1. Pantallas táctiles resistivas

Las pantallas resistivas están compuestas por dos capas conductoras y resistivas dispuestas con una pequeña separación. Cuando se ejerce presión sobre la superficie, ambas capas entran en contacto y se produce un cambio en la corriente eléctrica transmitida. Este cambio es detectado por un controlador que se encarga de calcular la posición exacta del contacto.

La pantalla consta de dos capas conductoras resistivas, cada una de ellas con dos barras conductoras ubicadas en lados opuestos. Puesto que las capas son conductoras resistivas, al aplicar una diferencia de potencial entre las barras se genera un gradiente de voltaje sobre la capa.

Al efectuar un toque sobre la pantalla, ambas capas entran en contacto y se crea un divisor de voltaje en el punto de presión.

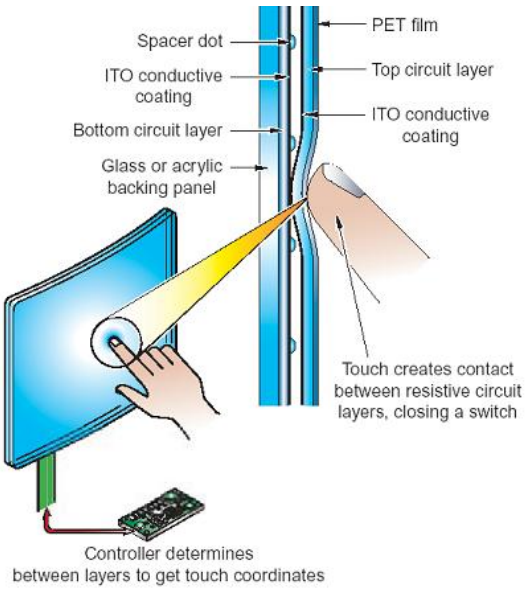


Figura 2.16: Pantalla Resistiva.
Figura extraída de [18]

El controlador determina la posición de la coordenada X aplicando una diferencia de potencial entre las barras conductoras Y+ e Y-. Cuando se produce contacto entre las capas, las barras conductoras de la capa X detectan el voltaje producido y el controlador calcula a partir de dicho valor la coordenada en X. De la misma forma se procede para calcular la coordenada Y: Se aplica una diferencia de potencial entre las barras X+ y X-, se detecta el voltaje producido en la otra capa y, finalmente, se calcula la coordenada en Y.

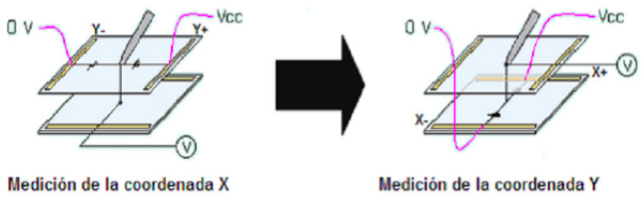


Figura 2.17: Medición de Coordenadas X e Y.
Figura extraída de [17]

2.3.1.1. Ventajas y desventajas

Las pantallas resistivas son menos costosas y tienen una mejor resistencia al agua o al polvo. Sin embargo, el uso de múltiples capas hace que las pantallas sean menos brillantes. Otra de las desventajas que se le ha achacado a las pantallas resistivas es que no permiten detectar múltiples contactos, no obstante, algunas compañías como Stantum[21], ya han desarrollado pantallas multitouchs que incluso son sensibles a la presión.

2.3.1.2. Dispositivos

En la Figura 2.18 se pueden ver distintos dispositivos con pantallas resistivas. Dichas pantallas las podemos encontrar especialmente en TPV's (Terminales Punto de Venta), PDA's y en los últimos terminales móviles.

Figura 2.18: Pantallas resistivas en TPV, PDA y Terminal Móvil.

2.3.2. Pantallas táctiles capacitivas

Estas pantallas están formadas por una membrana de vidrio con una delgada capa metálica colocada sobre la superficie. Al conducir corriente por esta membrana se carga uniformemente. Cuando el usuario toca la pantalla algunas de las cargas se transfieren a él debido a que el cuerpo humano también dispone de capacitancia. Con ello, disminuye la carga de la membrana, es decir, su capacitancia es alterado por el usuario. Este efecto de decrecimiento se mide en los circuitos electrónicos situados en cada esquina de la pantalla. Según la variación de la carga medida en una esquina se puede conocer a qué distancia se encuentra el contacto. Así pues, el punto de toque es calculado por el controlador considerando la variación de carga producida en las cuatro esquinas. En la Figura 2.19 se puede ver cómo se realiza la medición en las esquinas para calcular las coordenadas X e Y del toque.

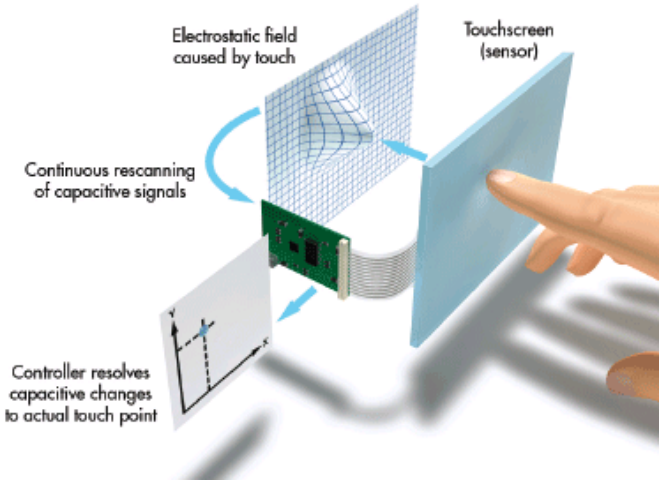


Figura 2.19: Pantalla capacitiva.
Figura extraída de [18]

2.3.2.1. Ventajas y desventajas

Entre las ventajas de las pantallas capacitivas encontramos que responden mejor que las resistivas, además soportan la tecnología multitouch. Entre las desventajas encontramos que estas pantallas son más caras que las resistivas y que únicamente pueden ser utilizadas con objetos con capacitancia por lo que no se pueden utilizar los stylus o lápices convencionales. El uso de este tipo de pantallas es cada vez más popular.

2.3.2.2. Dispositivos

Las pantallas capacitivas se pueden encontrar en muchos de los teléfonos de nueva generación. Como se mencionó anteriormente, este tipo de pantallas es cada vez más popular ya que presentan una mejor respuesta y un mayor brillo que las pantallas resistivas. Entre los dispositivos que han salido al mercado y se han popularizado podemos nombrar el iPhone o el iPod Touch de Apple, que además se caracterizan por ser multitouchs.

Figura 2.20: Samsung galaxy S3, iPhone 4S y HTC One.

2.4. Pantallas multitouchs basadas en tecnología optica

Las tecnologías ópticas, o basadas en el uso de cámaras, requieren de la visión por computador para detectar los toques sobre la superficie de la pantalla. Frustrated Total Internal Reflection (FTIR)[19], Diffused Illumination (DI)[19], Laser Light Panel (LLP)[19], Diffused Surface Illumination (DSI)[19] y Led Light Plane (LED-LP)[19] son algunas de las tecnologías que se basan en el uso de una cámara para construir pantallas multitouchs. Las pantallas táctiles que utilizan estas tecnologías pueden ser construidas fácilmente por cualquier persona y, además, con un bajo presupuesto. También son destacables por su escalabilidad, ya que es muy sencillo realizar pantallas multitouchs de gran tamaño.

Todas estas tecnologías hacen uso de un sensor óptico (cámara), luz infrarroja y una pantalla. Para comprender el funcionamiento de éstas tecnologías es necesario comprender los fundamentos de la luz infrarroja y los sensores ópticos.

- Luz Infrarroja: La luz infrarroja forma parte del espectro de luz no visible. Su rango de longitud de onda se encuentra entre la luz visible y las microondas, es decir, entre los 700 y los 1000 nanómetros. La luz infrarroja será la que permita distinguir los toques que se realicen sobre la pantalla de la imagen proyectada sobre la superficie.

- Cámaras de infrarrojos: Las cámaras comunes suelen bloquear la luz infrarroja dejando pasar únicamente la luz visible. Sin embargo, para construir éstos sistemas basados en la detección de la luz infrarroja, es necesario todo lo contrario.

2.4.1. Frustrated Total Internal Reflection (FTIR)

FTIR hace referencia a una tecnología desarrollada por Jefferson Han. Se basa en la refracción de la luz al cambiar de medio y, particularmente, en la reflexión interna total. La reflexión interna total se produce cuando la luz pasa de un medio a otro con mayor índice de refracción, refractándose de tal modo que no es capaz de atravesar la superficie que separa ambos medios de forma que se refleja totalmente. La reflexión total se produce cuando el haz de luz incide con un ángulo mayor a un cierto valor crítico que se puede calcular mediante la ley de Snell. La tecnología desarrollada por Jefferson Han hace uso de este efecto. Inyectando luz infrarroja en una superficie de metacrilato, ésta queda atrapada en su interior. Cuando el usuario entra en contacto con la superficie se dice que los rayos de luz infrarroja se ven frustrados, ya que ahora son capaces de pasar a través del metacrilato, iluminando la yema del dedo. La luz reflejada por la yema atraviesa el metacrilato y es captada por la cámara infrarroja.

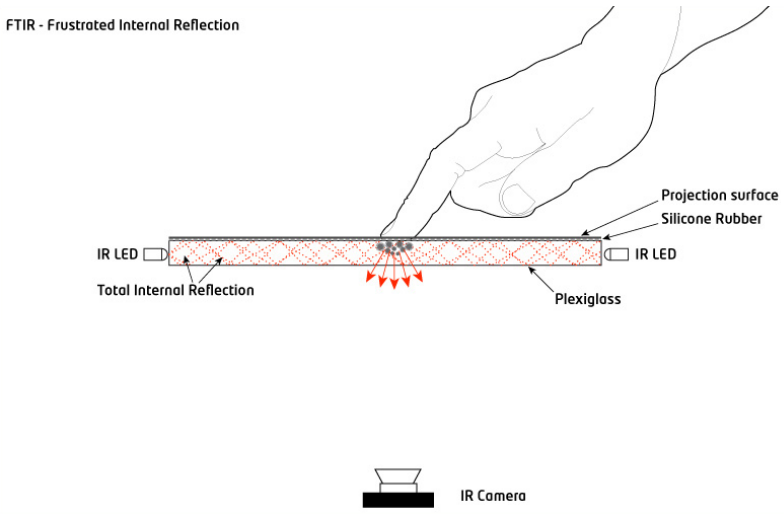


Figura 2.21: Frustrated Total Internal Reflection (FTIR).
Figura extraída de [19]

2.4.1.1. Ventajas

- No es necesario encerrarla en una caja.
- Los eventos táctiles se visualizan con un alto contraste.
- Permite detectar la presión realizada.

2.4.1.2. Desventajas

- No permite reconocer objetos ni marcadores.
- Para un buen funcionamiento requiere una capa de silicona.
- Requiere un marco de leds que deben ser soldados.

2.4.2. Diffused Illumination (DI)

Esta tecnología se puede utilizar de dos formas: Front Diffused Illumination y Rear Diffused Illumination. Ambas se basan en el contraste que se produce en la imagen al tocar la pantalla.

- **Front Diffused Illumination**

  La pantalla se compone de un acrílico y un difusor. La luz ambiental de la habitación ilumina la pantalla, con lo que no son necesarios leds o focos de infrarrojos. Cuando el usuario toca la pantalla interrumpe el paso de la componente infrarroja de la luz, produciendo sombras sobre el difusor. La cámara se encarga de capturar éstas sombras y, a partir de ellas, obtiene las coordenadas del toque.

- **Rear Diffused Illumination**

  Con luces infrarrojas se ilumina uniformemente la pantalla por su parte trasera. La pantalla dispone de un difusor en su parte delantera o trasera, de forma que, al tocar sobre la superficie ésta refleja más luz de lo que hace el difusor. Este incremento de luz es detectado por la cámara.

  En la Figura 2.22 se puede ver una representación del funcionamiento de esta técnica.

  La dificultad de este sistema radica en conseguir una iluminación uniforme sobre la pantalla, ya que si no se consigue habrán partes menos iluminadas en las que resultará necesario presionar más fuerte.

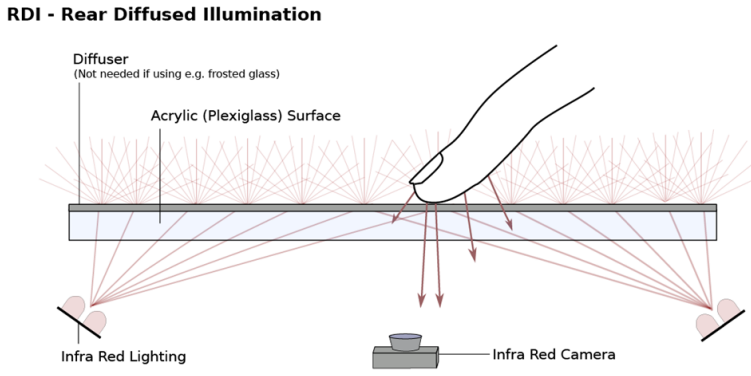


Figura 2.22: Rear Diffused Ilumination (RDI).
Figura extraída de [19]

2.4.2.1. Ventajas

- No necesita una capa de silicona.
- Puede utilizar cualquier material transparente (no solo acrílico).
- Funciona con objetos y marcadores.

2.4.2.2. Desventajas

- Los toques aparecen con bajo contraste.
- Grandes posibilidades de que se detecten falsos positivos.
- Es necesario encerrarla en una caja.
- Difícil de conseguir iluminación uniforme.

2.4.3. Laser Light Panel (LLP)

En esta técnica se utiliza la luz infrarroja proveniente de láseres que se ubican sobre la superficie exterior de la pantalla como se muestra en la figura 2.23. De esta forma se crea sobre la pantalla una superficie de luz infrarroja. Cuando se produce un contacto sobre la pantalla, la luz colisiona con el dedo y se desprende de forma que se detecta por la cámara.

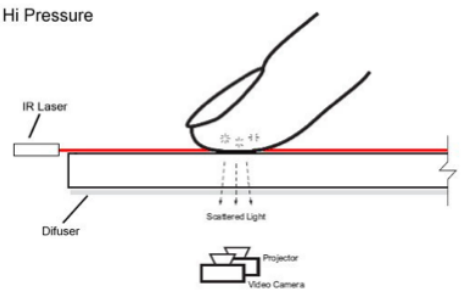


Figura 2.23: Laser Light Panel (LLP).
Figura extraída de [19]

La desventaja de este sistema es el peligro inherente de los láseres ya que pueden resultar dañinos para la vista si se miran directamente.

2.4.3.1. Ventajas

- No necesita una capa de silicona.
- Puede utilizar cualquier material transparente (no solo acrílico).
- No necesita leds.
- No es necesario encerrarla en una caja.
- Más barata que otras técnicas.

2.4.3.2. Desventajas

- No permite reconocer objetos ni marcadores.
- No es sensible a la presión realizada.
- Los eventos táctiles impiden que la luz siga emitiéndose sobre la pantalla (ya que el dedo interrumpe la trayectoria de la luz). Por ello, habrán zonas de la pantalla sin iluminar en las que no será posible detectar toques.

2.4.4. Diffused Surface Illumination (DSI)

Esta tecnología utiliza un acrílico especial para distribuir la luz infrarroja. Básicamente sigue el mismo funcionamiento que FTIR pero se diferencia en el acrílico utilizado, el cual se caracteriza por estar compuesto por pequeñas partículas que actúan como pequeños espejos. Cuando se iluminan los bordes del acrílico con leds, la luz infrarroja se distribuye y dispersa por toda su superficie. Cuando el usuario entra en contacto con el acrílico, la luz es dispersada y por tanto detectada por la cámara. En la Figura 2.24 se representa gráficamente el funcionamiento de esta tecnología.

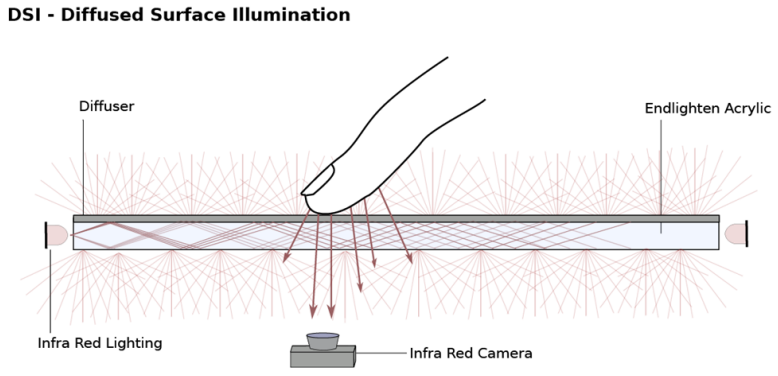


Figura 2.24: Diffused Surface Illumination (DSI).
Figura extraída de [19]

2.4.4.1. Ventajas

- No necesita una capa de silicona.
- Funciona con objetos y marcadores.
- Es sensible a la presión.

2.4.4.2. Desventajas

- El acrílico utilizado es más caro.
- Los eventos táctiles aparecen con bajo contraste.
- Posible restricciones de tamaño debido al grosor del acrílico.

2.4.5. Led Light Plane (LED-LP)

Su construcción es similar a FTIR, sin embargo, como se puede ver en la Figura 2.25, los leds se ubican por encima del acrílico y no en sus bordes. En ese sentido es similar a la tecnología LLP ya que se genera un plano de luz sobre la pantalla. El problema que presenta es que los leds, al producir luz cónicamente no sólo iluminan la superficie si no también los objetos que se encuentran a su alrededor. Para que esto no afecte al funcionamiento se pueden aplicar filtrados software que permitan definir los umbrales de detección.

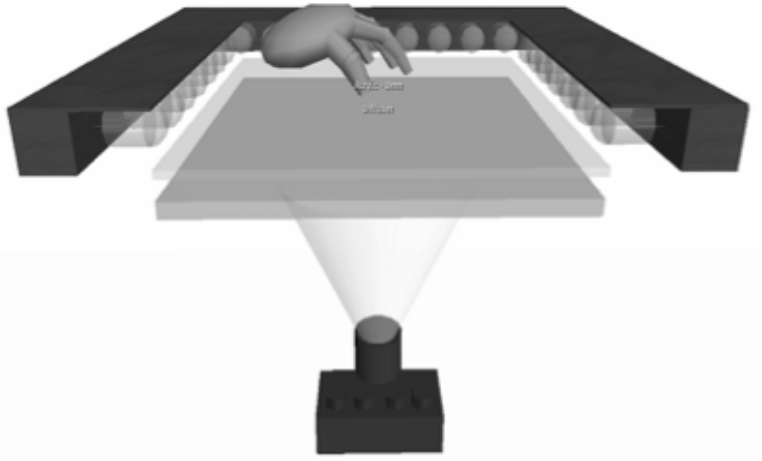


Figura 2.25: Led Light Plane (LED-LP).
Figura extraída de [19]

2.4.5.1. Ventajas

- No necesita una capa de silicona.
- Puede utilizar cualquier material transparente (no solo acrílico).
- No es necesario encerrarla en una caja.
- Más barata que otras técnicas.

2.4.5.2. Desventajas

- No permite reconocer objetos ni marcadores.
- Sólo pueden ser utilizados leds de haz estrecho.
- Requiere el soldado de leds.

2.4.6. Dispositivos

Uno de los dispositivos comerciales que utiliza técnicas de imagen óptica es la mesa multitouch Microsoft Surface. Esta mesa se basa en la técnica Rear Diffused Illumination (Rear DI), por lo que en su interior se puede encontrar un proyector, varias cámaras y focos infrarrojos para iluminar uniformemente la pantalla. Esta mesa se utiliza especialmente en hoteles, tiendas, restaurantes y lugares de entretenimiento.

Figura 2.26: Microsoft Surface.

CUBIT [20] es un sistema multitouch de código abierto que, al igual que Microsoft Surface, basa su funcionamiento en la técnica Rear Diffused Illumination (Rear DI). La intención de su creador es redefinir la interacción con el computador y apartar definitivamente el uso del ratón. CUBIT supone una alternativa más económica que Microsoft Surface. En la Figura 2.27 se pueden ver los diferentes componentes de una mesa CUBIT.

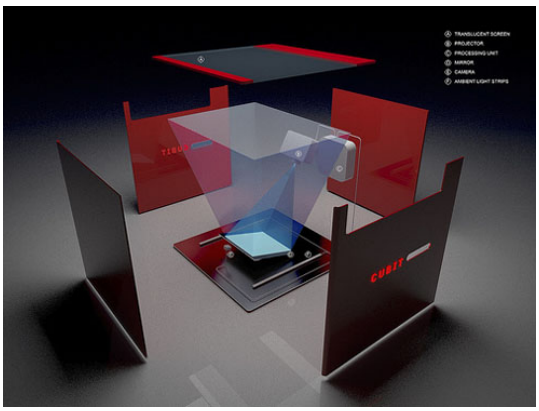


Figura 2.27: Cubit de Nortd Labs.
Figura extraída de [20]

Existen otras empresas que también comercializan mesas multitouchs como es el caso de Ideum[22]. MT-50 es su principal modelo y se caracteriza por tener una pantalla de 50". Esta mesa está destinada principalmente a ser utilizada en museos y exposiciones.

Figura 2.28: MT-50 de Ideum.
Figura extraída de [22]

Pero no todos los dispositivos multitouchs que utilizan esta tecnología siguen la estructura de una mesa (pantalla horizontal). Otras variantes se basan en crear "paredes multitouchs", es decir, la pantalla permanece vertical. Un ejemplo es Microsoft TouchWall, un dispositivo presentado recientemente por Microsoft que sigue la filosofía de Microsoft Surface pero en formato pizarra.

Figura 2.29: Microsoft Touchwall.

2.5. Requisitos software

Este apartado detalla el análisis inicial realizado sobre todas las necesidades y requisitos que finalmente será desarrollado en este proyecto.

Antes de empezar con la implementación del sistema, es muy importante conocer y pensar los requerimientos que queremos que tenga. Una vez establecidos estos requerimientos podremos valorar de una manera más precisa el alcance de los objetivos del proyecto y su viabilidad. Estos requerimientos posteriormente pueden ser modificados durante la etapa de desarrollo del proyecto debido a que no conocemos las limitaciones que podemos llegar a encontrar.

Los requerimientos los analizaremos de dos maneras: requerimientos funcionales y no funcionales. Los requerimientos funcionales son aquellos que describen los servicios que se podrán ofrecer a los usuarios así como las distintas reacciones que tendrá el sistema con diferentes inputs mientras que los requerimientos no funcionales son restricciones sobre los servicios que se ofrecen.

Finalmente se presentará un modelo conceptual de las clases que van a componer el proyecto, para proporcionar una vista general de la estructura de éste, así como los casos de uso.

2.5.1. Requerimientos funcionales del sistema

Los requerimientos funcionales en el sistema se analizarán desde el punto de vista del sistema puesto que los usuarios son totalmente externos a esta sección y por lo tanto no obtienen directamente ninguna funcionalidad.

- Colaborativo, es decir, que pueda ser manipulado por varios ejecutantes en forma local o remota.
- El sistema ha de ser intuitivo, que se pueda interactuar con él sin un manual de instrucciones.
- Poseedor de una sonoridad interesante y desafiante.
- Enseñable y aprendible, incluso por niños.
- Apto para novatos y para músicos electronicos avanzados.
- El sistema tiene que ser capaz de identificar a través de la webcam y haciendo uso de la librería ReacTIVision cada una de las amebas gráficas que se proporcion así como los eventos táctiles.
- El sistema tiene que ser capaz de permitir que los elementos interactivos puedan desplazarse a través de la pantalla interactiva.
- El sistema tiene que ser capaz de calcular la posición de los elementos encontrados.
- El sistema tiene que ser capaz de actualizar los datos de entrada de forma coherente con tal de facilitar la interacción.
- El sistema tiene que ser capaz de representar de manera gráfica en la plataforma Processing los elementos con los que está iteractuando.
- El sistema debe detectar si se está interactuando con Audio, MIDI o con efectos de sonido.
- El sistema debe estar sincronizado con el secuenciador con el que se esté trabajando.
- El sistema representará de manera visual el espectro del sonido que está en la tabla multisensonrial a través de los valores de onda que obtiene Pure Data.

2.5.2. Requerimientos no funcionales del sistema

- Es necesario colocar la webcam en un lugar con iluminación invariante ya que puede afectar en la detección de los elementos.
- La extracción de las características necesarias para la conversión se llevará a cabo utilizando técnicas de análisis espectral.

- Es necesario tener la webcam a una distancia prudencial de la mesa multitouch, sino el sistema no funcionará correctamente.
- Es responsabildad del usuario abrir su proyecto musical en el secuenciador para poder manipularlo en la mesa multitouch.
- La aplicación trabajará con los formatos de audio que soporta Ableton Live!
- El usuario debe seleccionar los archivos de audio con los que trabajará
- El entorno de desarrollo ha sido Processing. Una herramienta muy potente e intuitiva que facilita el desarrollo e implementación del proyecto.

Los requerimientos presentados, constituyen una representación de las operaciones que el usuario podrá llevar a cabo mediante el uso de la aplicación; esto no quiere decir que durante la elaboración y construcción del programa no puedan haber variaciones y matices sobre los requisitos; de hecho, tal como hemos indicado anteriormente, se trata de un desarrollo iterativo en el cual, durante la elaboración del producto, pueden haber cambios o modificaciones en los requisitos que favorezcan una refinación progresiva del sistema.

2.6. Casos de uso

En este apartado se describirán las funcionalidades de las que consta el sistema de manera detallada, es decir, se analizarán en detalle los objetivos funcionales mostrados anteriormente.

Se usará la notación de diagrama de casos de uso, de manera que, se mostrarán casos de uso con sus relaciones y actores implicados. Cada caso de uso representará una funcionalidad y describirá el conjunto de acciones ejecutadas por el sistema tras la orden de un actor.

Se organizará el análisis de funcionalidades en varias secciones: gestión de archivos de audio, gestión de recursos, gestión de efectos, gestión de elementos gráficos y funciones de la vista previa.

2.6.1. Gestión de archivos de audio

Los siguientes casos de uso describen aquellas funcionalidades referentes a la selección de un archivo de audio y su posterior uso en el sistema, hay que tener en cuenta que la aplicación tratará archivos de audio, así como mensajes MIDI u OSC.

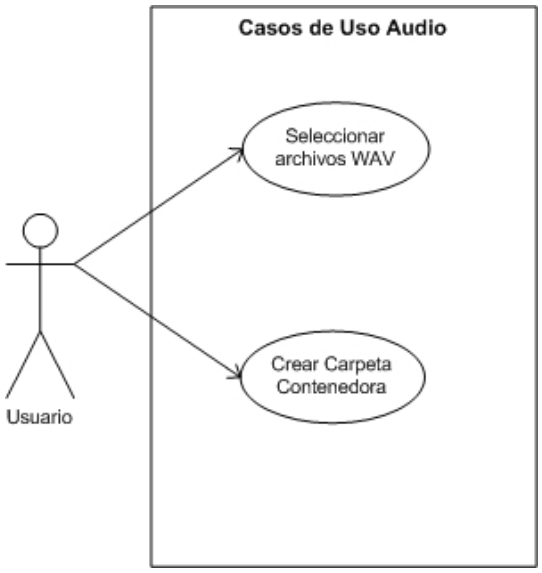


Figura 2.30: Caso de uso para la gestión de archivos de audio.

La aplicación trabajará con archivos de audio en formato WAV, pero el usuario debe ser responsable de la selección de estos archivos de audio, ya que son los que personalizarán la creación musical y los que harán que un proyecto musical difiera de otro.

El usuario deberá selección aquellos archivos de audio en formato WAV que usará a lo largo de la manipulación en sus creaciones musicales.

	Especificación Caso Uso Seleccionar Archivos WAV
Actor Principal	Usuario
Otros Actores	
Precondiciones	Crear un conjunto de muestras de audio.
Postcondiciones	
Evento inicial	Preparar el inicio de un proyecto de audio.
Flujo normal	**1**.- El usuario carga un conjunto de archivos de audio en el ordenador donde está instalado el proyecto. **2**.- El usuario se asegura que dichas muestras siguen el formato de la plantilla dada en Ableton Live!.
Flujo alternativo	

	Especificación Caso Uso Crear Carpeta Contenedora
Actor Principal	Usuario
Otros Actores	
Precondiciones	Se han tenido que seleccionar las muestras de audio con las que el usuario va a trabajar.
Postcondiciones	Proyecto preparado con las muestras de audio que se utilizarán.
Evento inicial	Proyecto de audio preparado para ser utilizado.
Flujo normal	**1**.- Se crea una carpeta contenedora donde se almacenarán los archivos de audio. **2**.- Por último se copian los archivos de audio en la carpeta que se ha creado.
Flujo alternativo	

Cuadro 2.2: Caso de uso seleccionar archivos Wav

2.6.2. Gestión de recursos

En los siguientes casos de uso se describirán aquellas funcionalidades referentes a los recursos usados como inputs por los efectos en el sistema.

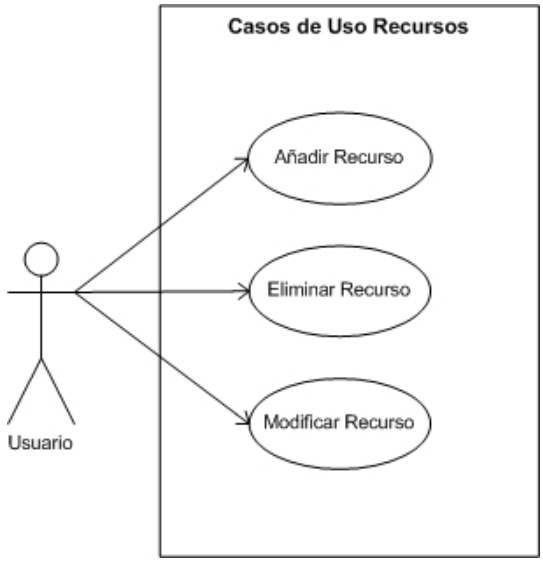


Figura 2.31: Caso de uso para la gestión de recursos.

Debido a que en el presente proyecto se manipulan dos tipos de recursos, existirá un conjunto de amebas gráficas que estarán identificadas con los sonidos de audio WAV previamente cargados, así como otro conjunto de amebas gráficas destinadas al control de instrumentos virtuales MIDI a traves del secuenciador, por tanto la funcionalidad abrir recurso hace referencia al tipo de ameba con la que el usuario va a interactuar. Además el sistema deberá validar la carga, de manera que, si el recurso no ha podido cargarse informará al usuario del fallo y de la no disponibilidad del recurso en el sistema.

La funcionalidad eliminar recurso, permitirá al usuario eliminar un recurso ya cargado, en principio el usuario simplemente tendrá que quitar la ameba utilizada del plano visual en el que se está trabajando.

La funcionalidad Modificar recurso, permite al usuario provocar modificaciones en el recurso con el que está tratando, es decir que ya sea girandolo o moviendo la ameba en dos direcciones se provocarán alteraciones del sonido en el caso de los archivos de audio y se podrán variar las escalas así como crear melodías cuando se está manejando un recurso destinado al control de instrumentos virtuales MIDI.

	Especificación Caso Uso Añadir Recurso
Actor Principal	Usuario
Otros Actores	
Precondiciones	Disponer de un conjunto de archivos de audio con los que trabajar, así como un proyecto en el secuenciador MIDI para trabajar con instrumentos virtuales.
Postcondiciones	Secuenciador MIDI preparado para manipular los instrumentos virtuales.
Evento inicial	
Flujo normal	**1**.- El usuario abre el secuenciador MIDI así como el proyecto sobre el que va a interactuar con las amebas. **2**.- El usuario manipula los recursos con el fin de poder representar su creación musical en directo. **3**.- Se interactua con los recursos hasta que el usuario decida finalizar su sesión musical.
Flujo alternativo	

	Especificación Caso Uso Eliminar Recurso
Actor Principal	Usuario
Otros Actores	
Precondiciones	Elementos añadidos en la mesa multitouch.
Postcondiciones	Mesa multitouch sin uno de los elementos añadidos.
Evento inicial	Elemento añadido o no.

Flujo normal	**1**.- El usuario quita de la mesa multitouch aquel elemento que no quiera seguir usando. **2**.- El sistema deja de reproducir el audio o MIDI relacionado con el elemento que se ha eliminado.
Flujo alternativo	

	Especificación Caso Uso Modificar Recurso
Actor Principal	Usuario
Otros Actores	
Precondiciones	Elementos añadidos en la mesa multitouch
Postcondiciones	Se aplica algún evento al recurso que se está manipulando
Evento inicial	Elemento añadido preparado para ser modificado
Flujo normal	**1**.- El usuario mueve bidireccionalmente o de manera rotacional un recurso. **2**.- El sistema aplica un evento según la elección del usuario. **3**.- El evento deja de reproducirse hasta que el usuario deje de manipularlo.
Flujo alternativo	

2.6.3. Gestión de efectos

En este apartado se describirán las funciones disponibles para la gestión de efectos, así como la manera en que el sistema deberá ejecutar un tipo de efecto de acuerdo a la proximidad de una ameba con otra.

Un efecto será la entidad del sistema encargada de generar una modificación sonora (pitch, echo, reverb...) de acuerdo a un conjunto de amebas gráficas destinadas a dicha modificación.

Cada efecto dispondrá de un conjunto de parámetros configurables que permitirán modificar el comportamiento del mismo, ya se girando la ameba gráfica o moviendola bidireccionalmente. Los distintos tipos de parámetros y su gestión se tratarán en los siguientes apartados.

Cada efecto será capaz de decidir si es aplicable o no en la configuración de hardware que vaya a ejecutarse. Si no lo es, el efecto no se tendrá en cuenta en la proyección musical actual.

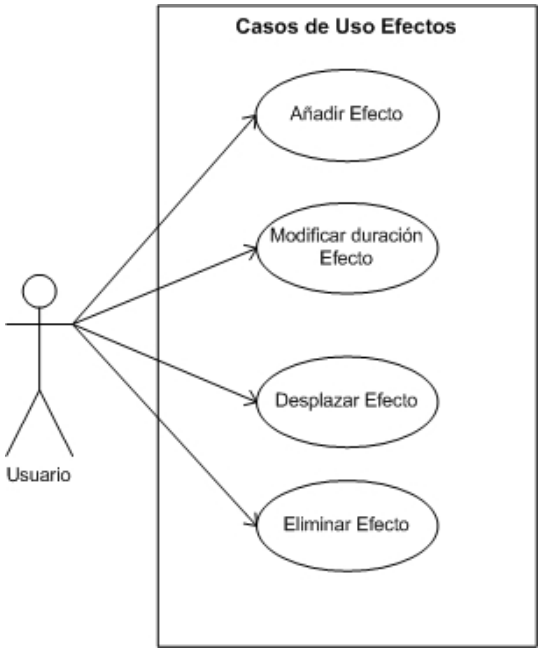


Figura 2.32: Casos de uso para la gestión de efectos.

La funcionalidad añadir efecto, permitirá la inserción de una ameba destinada al control del efecto seleccionado. Si se trata de un archivo de audio, se realizará el efecto de acuerdo a la proximidad de las amebas gráficas, existiendo diferentes amebas según el tipo de efecto que se desea aplicar. Para añadir efectos a los instrumentos virtuales manejados via MIDI se realizarán pequeñas adaptaciones de acuerdo a los parámetros que soporta cada instrumento virtual.

La duración de un efecto podrá variarse, modificando los instantes de inicio y fin, teniendo en cuenta que el sistema sólo permitirá su modificación mientras el instante de inicio sea anterior al de fin, esto se hace debido a que si es solo un usuario el que desea interactuar con la máquina, no tenga que estar pendiente de que se realizar un efecto de manera contínua.

La funcionalidad desplazar efecto permitirá desplazar un efecto hacia otra ameba, ya que se aplican los efectos de acuerdo a la proximidad con una ameba destinada a reproducir audio. El sistema permitirá desplazar el efecto siempre que en la acción, el efecto, no se solape con ningún otro.

Finalmente un efecto podrá ser eliminado del sistama, tan solo habrá que quitar la ameba utilizada del plano visual en el que se está trabajando.

	Especificación Caso Uso Añadir Efecto
Actor Principal	Usuario
Otros Actores	
Precondiciones	Disponer en la mesa multitouch un conjunto de elementos a los que se le pueda aplicar un efecto.
Postcondiciones	Alteración sonora de los recursos disponibles de acuerdo al efecto elegido que se aplicará.
Evento inicial	Elemento disponible o no disponible
Flujo normal	**1**.- El usuario ha añadido elementos a la mesa multitouch. **2**.- El usuario introduce un elemento de efecto en la mesa multitouch. **3**.- El sistema por proximidad detecta a que elemento se le aplicará el efecto.
Flujo alternativo	

	Especificación Caso Uso Modificar Duración Efecto
Actor Principal	Usuario
Otros Actores	
Precondiciones	Disponer en la mesa multitouch un conjunto de elementos y una ameba que represente un efecto.
Postcondiciones	La duración del efecto variará modificando los instantes de inicio y fin.
Evento inicial	Inicio y fin de un efecto
Flujo normal	**1**.- El usuario ha añadido elementos a la mesa multitouch. **2**.- El usuario introduce un elemento de efecto en la mesa multitouch. **3**.- El sistema por proximidad detecta a que elemento se le aplicará el efecto. **4**.- El sistema modificará la duración del efecto de acuardo a los parámetros de incio y fin.
Flujo alternativo	

	Especificación Caso Uso Desplazar Efecto
Actor Principal	Usuario

Otros Actores	
Precondiciones	Disponer en la mesa multitouch un conjunto de elementos con su correspondiente efecto de sonido.
Postcondiciones	Alteración de los parámetros del efecto.
Evento inicial	Elemento disponible o no disponible
Flujo normal	**1**.- El usuario ha añadido elementos a la mesa multitouch. **2**.- El usuario introduce un elemento de efecto en la mesa multitouch. **3**.- El sistema por proximidad detecta a que elemento se le aplicará el efecto. **4**.- El sistema modifica los parametros de los efectos según la interacción del usuario.
Flujo alternativo	

	Especificación Caso Uso Eliminar Efecto
Actor Principal	Usuario
Otros Actores	
Precondiciones	Efecto añadidos en la mesa multitouch.
Postcondiciones	Mesa multitouch sin un efecto añadido.
Evento inicial	Efecto añadido o no.
Flujo normal	**1**.- El usuario quita de la mesa multitouch aquel efecto que no quiera seguir usando. **2**.- El sistema deja de reproducir el efecto.
Flujo alternativo	

2.6.4. Generación de elementos gráficos

En este apartado se mostrará la responsabilidad que tiene el sistema de generar un elemento gráfico una vez sea detectada y reconocida la ameba gráfica a través de la webcam, para su posterior visualización.

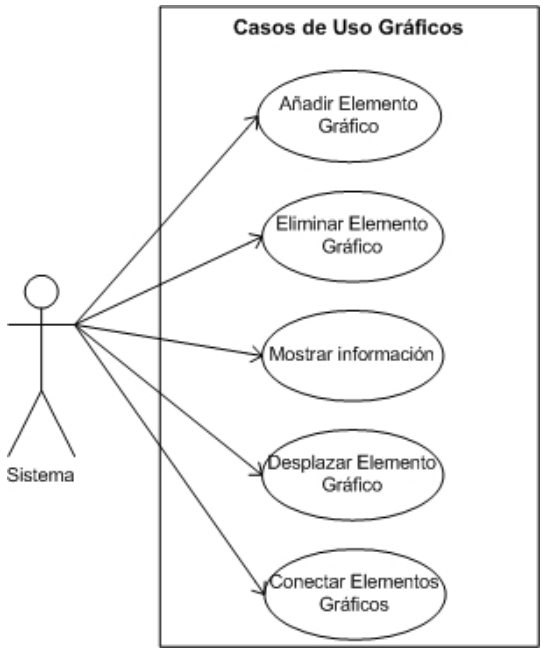


Figura 2.33: Casos de uso para la genración de elementos gráficos.

La funcionalidad añadir Añadir elemento gráfico, mostrará en el panel visual una primitiva básica que representa de manera visual la ameba gráfica y su posición actual de acuerdo al plano visual en el que se está trabajando, así mismo es responsabilidad del sistema eliminar dicho elemento gráfico una vez el usuario deje de trabajar con dicha ameba.

La funcionalidad Mostrar información nos dará al lado de cada elemento gráfico su posición en el eje de coordenadas, así como el id de las amebas que se encuentran sobre el plano visual. Estos parámetros irán cambiando en tiempo real según se mueva la ameba bidireccinalmente.

Es responsabilidad del sistema que si la ameba se desplaza o rota sobre el plano visual, se debe presenciar de manera gráfica y a tiempo real.

Todos los elementos gráficos irán conectados al punto central del plano visual ya sea representado por una línea o dibujando la onda sonora que está generando un determinado sonido en ese momento.

	Especificación Caso Uso Añadir Elementos Gráficos
Actor Principal	Sistema
Otros Actores	
Precondiciones	Sistema preparado para recibir elementos a traves de la webcam
Postcondiciones	Representación gráfica de una primitiva básica en el panel visual.
Evento inicial	
Flujo normal	**1**.- El usuario coloca un elemento en el panel visual. **2**.- La webcam reconoce el elemento que se ha colocado. **3**.- El sistema debe representar graficamente con una primitiva básica el elemento introducido. **4**.- Una vez se deje de usar una ameba en el plano visual, también debería desaparecer de la representación gráfica.
Flujo alternativo	

	Especificación Caso Uso Eliminar Elementos Gráficos
Actor Principal	Sistema
Otros Actores	
Precondiciones	Haber añadido elementos en el plano visual
Postcondiciones	Se deja de representar gráficamente el elemento que ha añadido.
Evento inicial	Elemento añadido o no
Flujo normal	**1**.- El usuario quita un elemento en el panel visual. **2**.- La webcam deja de detectar el elemento que se ha añadido. **3**.- Una vez se deje de usar una ameba en el plano visual, también debería desaparecer de la representación gráfica.
Flujo alternativo	

	Especificación Caso Uso Mostrar información
Actor Principal	Sistema
Otros Actores	
Precondiciones	Haber añadido elementos en el plano visual
Postcondiciones	Se muestra información sobre la posición del elemento en el plano visual.
Evento inicial	Elemento añadido o no
Flujo normal	**1**.- El usuario añade un elemento en el panel visual. **2**.- La webcam reconoce el elemento que se ha colocado. **3**.- El sistema debe representar graficamente con una primitiva básica el elemento introducido. **4**.- El sistema muestra la información sobre las coordenadas en las que se encuentra el elemento. **5**.- El sistema actualiza la información acorde al movimiento bidireccional o rotacional de el elemento que se manipula.
Flujo alternativo	

	Especificación Caso Uso Desplazar Elemento Gráfico
Actor Principal	Sistema
Otros Actores	
Precondiciones	Haber añadido elementos en el plano visual
Postcondiciones	Se muestra el movimiento sincronizado con el plano visual.
Evento inicial	Elemento añadido o no
Flujo normal	**1**.- El sistema debe representar graficamente con una primitiva básica el elemento introducido. **2**.- El sistema actualiza la posición del elemento gráfico mientras el usuario la desplaza. **3**.- El sistema representa gráficamente la nueva posición del elemento.
Flujo alternativo	

	Especificación Caso Uso Conectar Elemento Gráfico
Actor Principal	Sistema
Otros Actores	
Precondiciones	Haber añadido elementos en el plano visual
Postcondiciones	Elementos gráficos conectados con un punto central del plano visual.
Evento inicial	Representación de espectro de ondas o no
Flujo normal	**1**.- El usuario añade un elemento. **2**.- El sistema representa gráficamente con una primitiva básica dicho elemento. **3**.- El sistema conecta con una línea cada elemento con el punto central del plano visual. **4**.- El sistema puede representar el espectro de la onda en caso de tratarse un archivo de audio.
Flujo alternativo	

2.7. Diagrama de clases

El diagrama de clases de un sistema, permite describir su estructura mostrando sus clases y atributos, y relaciones existentes entre ellos. Se utilizan tanto en el análisis como en el diseño con diferentes niveles de abstracción. En este apartado se presenta el diagrama de clases de la fase de análisis. Se trata de la idea base de solución al problema desde un punto de vista conceptual, obteniendo las clases del dominio u objetos de negocio. Servirá como punto de partida para su transformación en el diagrama completo de clases de la fase de diseño.

Al diseñar una clase se debe pensar en cómo se puede identificar un objeto real, como una persona, un transporte, un documento o un paquete. Estos ejemplos de clases de objetos reales, es sobre lo que un sistema se diseña. Durante el proceso del diseño de las clases se toman las propiedades que identifican como único al objeto y otras propiedades adicionales como datos que corresponden al objeto.

El siguiente diagrama presenta estas relaciones, concretamente las establecidas entre el elemento principal, que es la mesa multitouch (llamada Reactable), y cada uno de sus elementos.

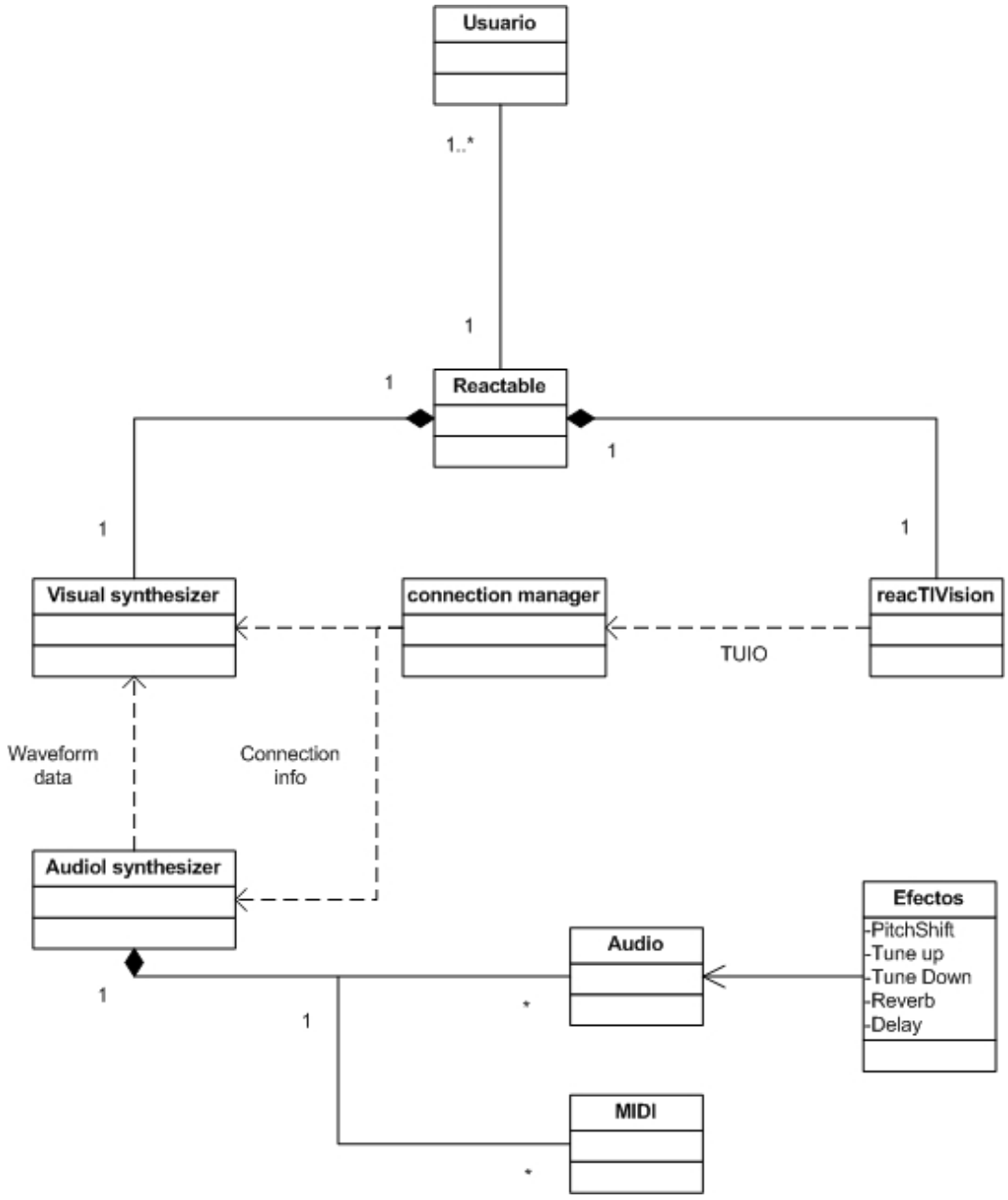


Figura 2.34: Diagrama clases reactable.

Como se observa en el diagrama de es un usuario o muchos de ellos los que estarán interactuando con la mesa multitouch, llamada reactable en nuestro diagrama de clases y como se puede observar, esta estaría compuesta por un modulo de contenido visual y por un modulo de captura de elementos mediante cámara web (ReacTIVision). Una cámara que se situará debajo de la mesa capturará las imagenes que debe procesar el ReacTIVision para reconocer la ubicación, orientación e identid de los fiduciales. Esta información se envía a otros comoponentes del software a traves de un conector de red usando el

protocolo TUIO, un protocolo en la parte superior del OSC (Open Sound Control).

La clase Connection Manager comunicará el modulo de producción de contenidos visuales con el modulo de audio. Como se observa en el diagrama, se pueden distinguir dos clases distintas:

- Audio
- Midi

La clase audio será la encargada de procesar los ficheros de audio que el usuario ha seleccionado y a estos se les podrá aplicar uno de los efectos básicos de las ondas sonoras, como pueden ser, PitchShift, Tune Up, Tune Down, Reverb o Delay.

La clase Midi deberá encargarse del control de aquellos instrumenos virtuales VST que están siendo tratados en el secuenciador, donde los fiduciales podrán controlar los sintetizadores sin necesidad de un teclado maestro, sino con la rotación y movimiento bidireccional de los fiduciales.

Cabe destacar que este diagrama podrá sufrir modificaciones una vez se haya avanzado el diseño de la aplicación, ya que a priori no se puede certificar que las conexiones de los distintos módulos vayan a trabajar tal y como se refleja en el esquema.

También resulta útil crear una vista estática de las definiciones de las clases mediante un diagrama de clases de diseño, que a diferencia del modelo de dominio, no muestra conceptos del mundo real, sino clases software.

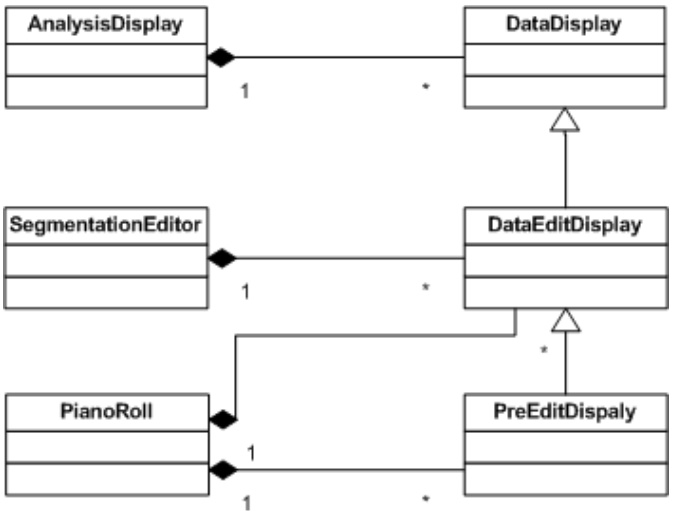


Figura 2.35: Diagrama clases de diseño para el sistema de visualización.

El diagrama de clases de la figura 2.35 muestra de forma esquemática el diseño para la implementación de los distintos modos de visualización que ofrece el Sistema. Este diagrama, así como los anteriores

presentados en este apartado dedicado a la discusión sobre el diseño orientado a objetos, han sido presentados como apoyo a la exposición de este concepto. En los siguientes capítulos, se explicará con un poco más de detalle lo referente al diseño e implementación de las diferentes partes de las que consta el sistema.

Capítulo 3

Diseño

En este apartado se mostrará el diseño de la interfaz gráfica de usuario, así como, la manera en que el usuario interactuará con el motor a través de ella. También se detallará la estructura de clases y se complementará el diseño de cada una de ellas.

El patrón arquitectónico escogido para el diseño de la aplicación ha sido un diseño en tres capas separando, de este modo, la capa de presentación que implementará las vistas, de los elementos del dominio y de la persistencia de los datos (y su gestión).

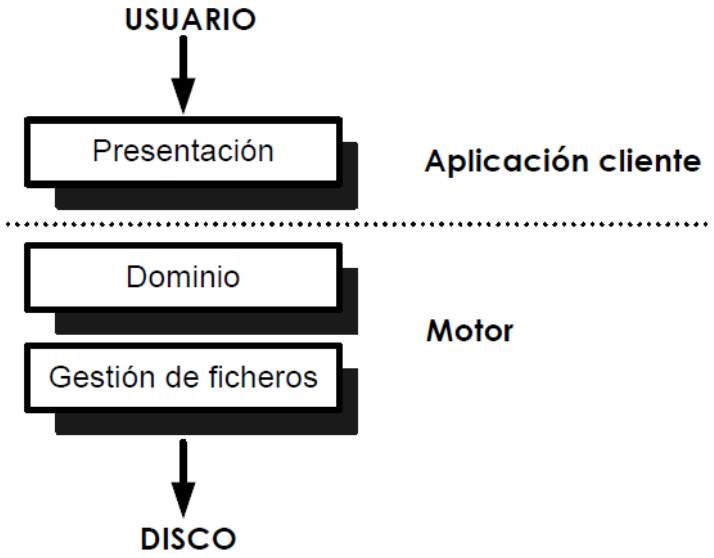


Figura 3.1: Diseño de capas.

Se ha realizado un diseño modular y encapsulado del proyecto, separando y agrupando lo máximo posible las funcionalidades de la aplicación. De este modo se consigue una mayor independencia entre las distintas partes del proyecto, simplificando así, diseño e implementación. Gracias a este enfoque también se simplifican las tareas de ampliación, ya que la modificación de un módulo del proyecto no supone un gran impacto en el resto de módulos.

A continuación y a lo largo de los siguientes apartados, se mostrará, en primer lugar la arquitectura de la aplicación. Para concluir con el diseño de la capa de presentación, el diseño de la capa de dominio y de la gestión de los ficheros de audio.

3.1. Arquitectura de la aplicación

En este apartado se muestra una visión general de los distintos módulos a diseñar. El diagrama de la figura 3.2 no pretende ser exhaustivo sino exponer de manera sencilla el funcionamiento de la aplicación.

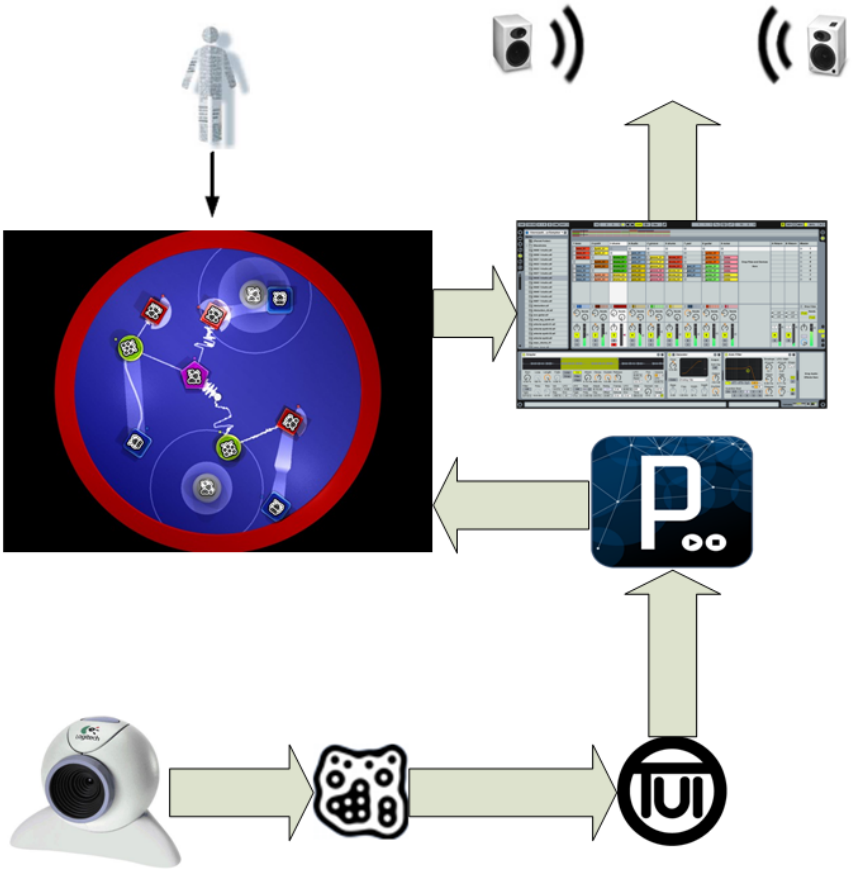


Figura 3.2: Arquitectura de la aplicación.

En el diagrama 3.2 se identifican los diferentes elementos que entran en juego para la realización del proyecto. La cámara web utiliza para el reconocimiento de los fiduciales la librería ReacTIVision, y haciendo uso del protocolo TUIO se envía la información al entorno de desarrollo Processing, donde se implementará el módulo de producción visual. El usuario interactuará con los fiduciales y de acuerdo a su manejo controlará el secuenciador de Audio/MIDI Ableton Live! el cual dará la salida de sonido

3.2. Diseño de la capa de presentación

En este apartado, una vez realizada la especificación, se mostrará el diseño de la capa de presentación, en la cual, se mostrará el diseño funcional y visual del modulo de producción de contenidos visuales. Esta será la capa visible por el usuario y con la que realizará la interacción con el dominio.

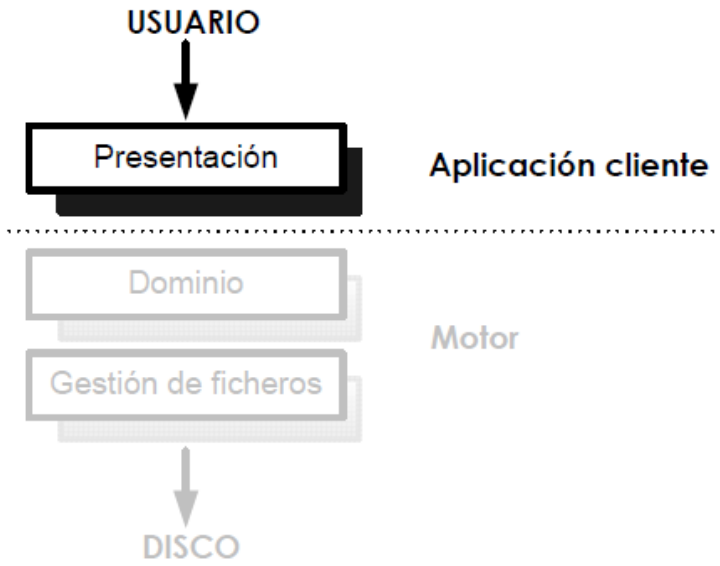


Figura 3.3: Diseño capa de presentación.

Se ha prestado especial importancia a la sencillez y a la usabilidad de la aplicación, ya que, lo que se pretende es que cualquier persona sea capaz de aprender a utilizarla relativamente rápido para que pueda sacarle partido lo antes posible. De modo que, el diseño de nuestra mesa multitouch tiene que ser lo más intuitiva posible para que el usuario pueda moverse con gran facilidad por ella sin necesidad de un largo periodo de aprendizaje.

Uno de los objetivos para que la interfaz sea lo más amigable posible, ha sido que prácticamente la totalidad de las funcionalidades estén disponibles de acuerdo a las formas definidas de los fiduciales. De

modo que el usuario sabrá en todo momento si un fiducial está asignado a los archivos de audio, a controlar un dispositivo MIDI ó aplicar un efecto sencillo sobre los elementos con los que está interactuando.

Una vez fijados los objetivos se procederá al análisis de las vistas, esta será muy sencilla ya que realmente el usuario no estará manejando un programa software, sino que estará interactuando con un grupo de fiduciales como los de la figura 3.4

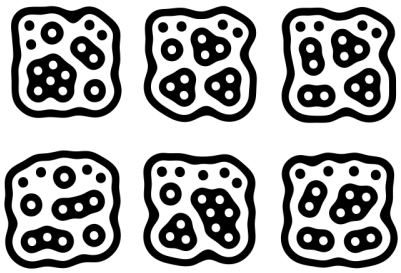


Figura 3.4: Conjunto de fiduciales.

La librería reacTIVision proporciona un archivo pdf con un total de 179 fiduciales que se muestran en el apéndice G. Los fiduciales se separarán por grupos para que el usuario sepa en cada momento que es lo que está manipulando.

La capa de presentación pretende ser lo más intuitiva posible, y al no tratarse el proyecto actual de un software convencional el usuario deberá desenvolverse en la superficie multitouch con un conjunto de fiduciales. La siguiente imagen muestra lo que realmente verá el usuario a la hora de interactucar con este proyecto.

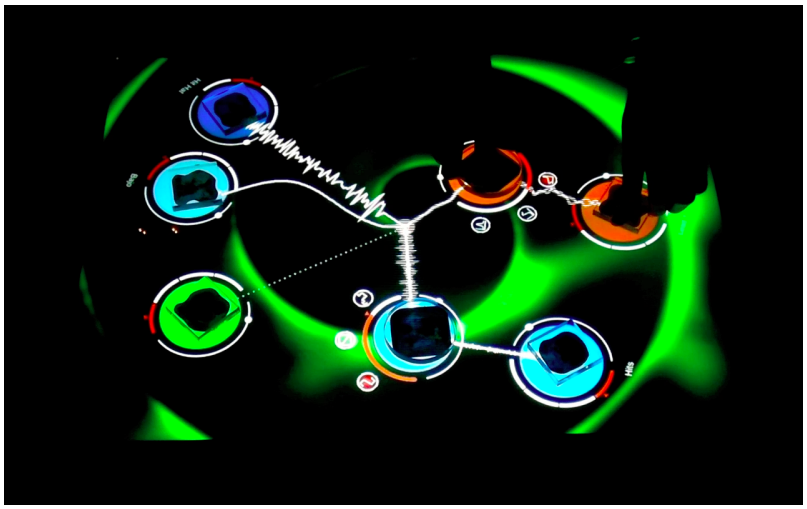


Figura 3.5: Visión realista de la capa de presentación.

3.2.1. La API de TUIO

Las implementaciones de referencia TUIO forman parte del framework reacTIVision y están disponibles para la mayoría de lenguajes de programación común. En esta sección se pretende explicar el funcionamiento de las clases diseñadas en TUIO así como las funciones de las que se harán uso.

El paquete TUIO consta de las siguientes clases:

- TuioClient: Componente decodificador del protocolo TUIO
- TuioCursor: Encapsula /tuio/2Dcur cursores TUIO
- TuioObject: Encapsula /tuio/2Dobj objetos TUIO
- TuioPoint: Es un contenedor, además de una clase para manejar las posiciones TUIO. Es la clase base para las clases TuioObject y TuioCursor
- TuioTime: Es una estructura que se utilizar para representar el tiempo que ha transcurrido desde el inicio de la sesión

A continuación veremos en más detalle las clases más relevantes y de las que se harán uso en el proyecto actual.

3.2.1.1. TuioClient

La clase TuioClient es el componente decodificador del protocolo TUIO. Proporciona una infraestructura de retorno de llamadas utilizando la interfaz TuioListener. Con el fin de recibir y decodificar mensajes TUIO, una instancia de TuioClient se debe crear. La instancia TuioClient genera evento TUIO que se transmiten a todas las clases registradas que implementa la interfaz TuioListener.

```
TuioClient client = new TuioClient();
client.addTuioListener(myTuioListener);
client.connect();
```

Constructor
TuioClient() **El constructor por defecto crea un cliente que escucha por el puerto 3333**
TuioClient(int port) **El contructor crea un cliente que escucha por el puerto especificado en int port**

Las funciones y procedimientos que se implementan en esta clase son las siguientes:

- void acceptMessage(java.util.Date date, com.illposed.osc.OSCMessage message): Método de retorno de llamadas de mensajes OSC donde todos los mensajes TUIO se reciben y se decodifican y donde los retornos de llamada de los eventos TUIO se envían.
- void addTuioListener(TuioListener listener): Agrega TuioListener a la lista de eventos detectores TUIO registrados.
- void connect(): TuioClient comienza a escuchar mensajes TUIO en el puerto UDP que se ha configurado. Todos los mensajes TUIO recibidos se decodifican y los eventos TUIO resultantes se transmiten a todos los TuioListeners registrados.
- void disconnect(): TioClient para de escuchar mensajes TUIO en el puerto UDP que se ha configurado.
- TuioCursor getTuioCursor(long s_id): Devuelve el TuioCursor que corresponde a la ID de la sesión prevista o NULL si el identificador de sesión no se refiere a un TuioCursor activo.
- java.util.Vector<TuioCursor>getTuioCursors(): Devuelve un vector de todos los TuioCursors activos.
- TuioObject getTuioObject(long s_id): Devuelve el TuioObject que corresponde a la ID de la sesión prevista o NULL si el identificador de sesión no se refiere a un TuioObject activo.
- java.util.Vector<TuioObject>getTuioObjects(): Devuelve un vector de todos los TuioObjects activos.
- void removeAllTuioListeners(): Quita todos los TuioListener de la lista de detectores de eventos TUIO registrados.
- void removeTuioListener(TuioListener listener): Quita el TuioListener especificado de la lista de detectores de eventos TUIO registrados.

3.2.1.2. TuioListener

La interfaz TuioListener proporciona una infraestructura de retorno de llamadas que es utilizada por la clase TuioClient para distribuir eventos TUIO a todas las instacias de las clases que implementan la interfaz TuioListener que se define a continuación.

Cualquier clase que implementa la interfaz TuioListener está obligada a implementar todos los métodos de retorno de llamada definidos aquí. TuioClient hace uso de estos métodos de interfaz con el fin de distribuir eventos TUIO a todas las implementaciones de TuioListener registradas.

```
public class MyTuioListener implements TuioListener
...

MyTuioListener listener = new MyTuioListener();
TuioClient client = new TuioClient();
client.addTuioListener(listener);
client.start();
```

Las funciones que se implemntan son las siguientes:

- void addTuioCursor(TuioCursor tcur): Este método de retorno de llamada se invoca por TuioClient cuando un TuioCursor nuevo se agrega a la sesión.
- addTuioObject(TuioObject tobj): Este método de retorno de llamada se invoca por TuioClient cuando un TuioObject nuevo se agrega a la sesión.
- void refresh(TuioTime btime): Este método de retorno de llamada se invoca por TuioClient para marcar el final de un grupo de mensajes TUIO recibidos.
- void disconnect(): TioClient para de escuchar mensajes TUIO en el puerto UDP que se ha configurado.
- void removeTuioCursor(TuioCursor tcur): Este método de retorno de llamada se invoca por TuioClient cuando un TuioCursor se quita de la sesión.
- void removeTuioObject(TuioObject tobj): Este método de retorno de llamada se invoca por TuioClient cuando un TuioObject se quita de la sesión.
- void updateTuioCursor(TuioCursor tcur): Este método de retorno de llamada se invoca por TuioClient cuando un TuioCursor se actualiza durante la sesión.
- void updateTuioObject(TuioObject tobj): Este método de retorno de llamada se invoca por TuioClient cuando un TuioObject se actualiza durante la sesión.

3.2.1.3. TuioPoint

La clase TuioPoint es un contenedor, además de una clase para manejar las posiciones TUIO. Es la clase base para las clases TuioObject y TuioCursor.

Constructores

TuioPoint() **El constructor por defecto no tiene argumentos y define la coordinación de los atributos a cero.**
TuioPoint(float xp, float yp) **Este constructor toma dos argumentos de punto flotante de coordenadas y establece su coordinación de los atributos a estos valores.**
TuioPoint(TuioPoint tpoint) **Este constructor toma un argumento TuioPoint y establece su coordinación de los atributos de acuerdo a tpoint**

Las funciones y procedimientos que se implementan en esta clase son las siguientes:

- float getAngle(float xp, float yp): Devuelve el ángulo de las coordenadas especificadas en xp y xy.
- float getAngle(TuioPoint tpoint): Devuelve el ángulo de TuioPoint.
- float getAngleDegrees(float xp, float yp): Devuelve el ángulo en grados de las coordenadas especificadas en xp y xy.
- float getAngleDegrees(TuioPoint tpoint): Devuelve el ángulo en grados de TuioPoint.
- int getScreenX(int width): Devuelve la coordenada X en píxeles con respecto a la anchura de la pantalla.
- int getScreenY(int height): Devuelve la coordenada Y en píxeles con respecto a la altura de la pantalla.
- TuioTime getTuioTime(): Devuelve la marca de tiempo de este TuioPoint como TuioTime.
- float getX(): Devuelve la coordenada X de este TuioPoint.
- float getY(): Devuelve la coordenada Y de este TuioPoint.
- void update(float xp, float yp): Toma dos argumentos de punto flotante de coordenadas y las actualiza sus atributos de coordenadas a las coordenadas TuioPoint y deja su marca de tiempo sin cambios.
- void update(TuioPoint tpoint): Toma un TuioPoint como argumento y actualiza sus atributos de coordenadas a las coordenadas TuioPoint y deja su marca de tiempo sin cambios.

3.2.1.4. TuioObject

Encapsula /tuio/2Dobj objetos TUIO.

Constructores
TuioObject(long si, int sym, float xp, float yp, float a) **Este constructor toma el identificador de sesión, símbolo de la ID, coordenadas X e Y y el ángulo, y asigna estos valores a la creación de un TuioObject nuevo.**
TuioObject(TuioObject tobj) **Este constructor toma los atributos del TuioObject que se le pasa y assigna estos valores a la creación de un TuioObject nuevos**

Como la la clase TuioObject hereda funciones de TuioPoint, aquí solo se mostrarán aquellas funciones distintas a las de la clase TuioPoint:

- float getMotionAccel(): Devuelve la aceleración del movimiento de TuioContainer.
- float getMotionSpeed(): Devuelve la velocidad del movimiento de TuioContainer.
- java.util.Vector<TuioPoint>getPath(): Devuelve la ruta de TuioContainer.
- TuioPoint getPosition(): Devuelve la posición de TuioContainer.
- float getRotationAccel(): Devuelve la acelaración de rotación de TuioObject.
- float getRotationSpeed(): Devulve la velocidad de rotación de TuioObject.
- long getSessionID(): Devuelve el numero de identificador de sesión de TuioContainer.
- int getSymbolID(): Devuelve el identificador de símbolo de TuioObjec.
- float getXSpeed(): Devuelve la velocidad en X de TuioContainer.
- float getYSpeed(): Devuelve la velocidad en Y de TuioContainer.
- boolean isMoving(): Devuelve true si TuioObject está en movimiento.

Las funciones para TuioCursor son exactamente las mismas que para TuioObject pero cuando se trata del cursor y no de objetos fiduciales.

3.3. Diseño de la capa de dominio

En este apartado, una vez realizada la especificación y el diseño de la capa de presentación, se mostrará el diseño de la capa de dominio, en la cual, se analizará y concretará la estructura interna del proyecto describiendo la estructuración de los diversos módulos del mismo.

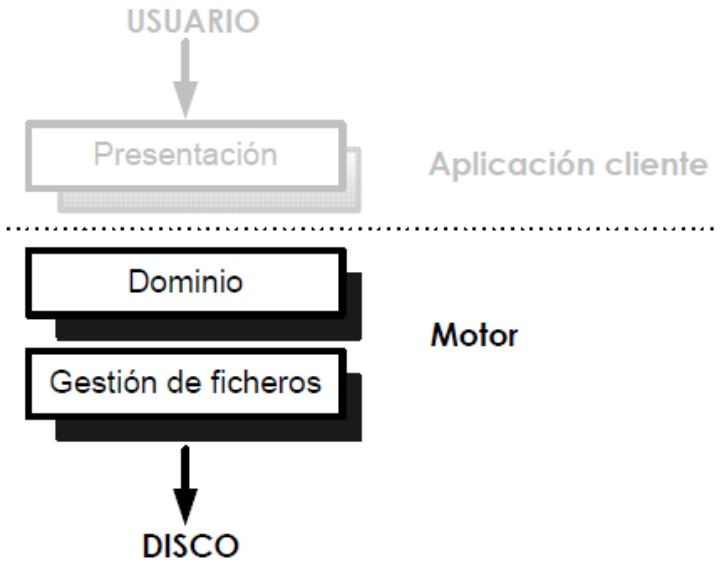


Figura 3.6: Diseño capa de dominio.

El diseño de esta capa se enfocará dividiendo el diseño general en diseños más simples, de modo que, juntos compondrán el dominio de manera completa. Primeramente se analizarán los módulos más específicos, para finalmente, unirlos en el diseño general. Se ha decidido usar este enfoque para mostrar, de la mejor manera posible, el diseño del dominio en la presente memoria.

3.3.1. Módulo de interacción multitouch

En esta sección se explicará el diseño de los módulos que comprenden el sistema reacTIVision, así como las librerías de código abierto de las que hace uso. Este módulo es uno de los más importantes de la capa de dominio, ya que implica el correcto funcionamiento del reconocimiento de los fudiciales y de la captación de movimiento en tiempo real.

El siguiente diagrama constituya el motor interno del sistema reacTIVision, como se puede observar en la figura 3.7.

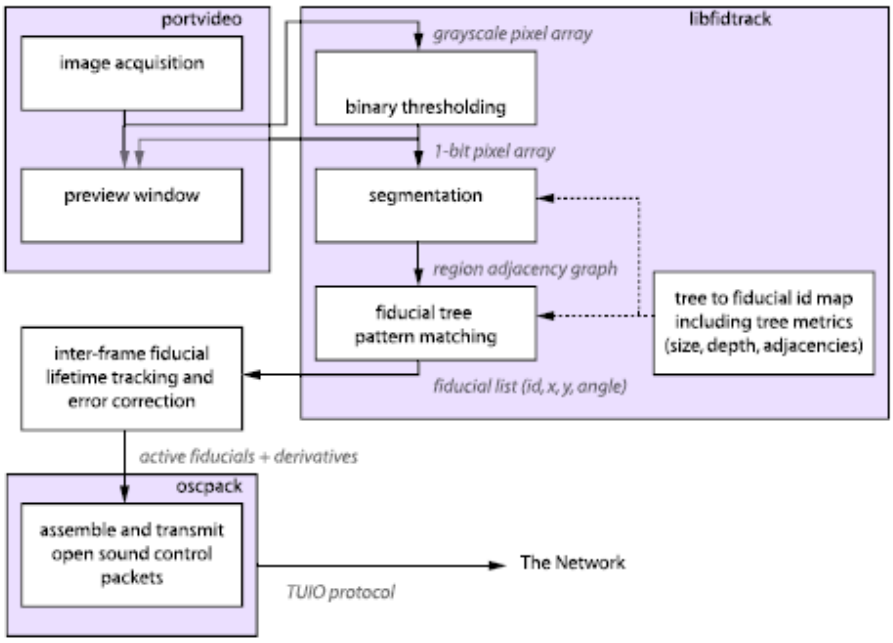


Figura 3.7: Diseño módulo de interacción multitouch.
Figura extraída de [14]

El diagrama muestra el nivel de diseño de módulos del sistema reacTIVision, que incorpora tres bibliotecas de código abierto:

1.- PortVideo: Plataforma de adquisición de vídeo en tiempo real

2.- Oscpack: Librería para el embalaje y desembalaje de los mensajes Open Sound Control.

3.- Libfidtrack: Una librería que implementa algoritmo de reconocimiento fiducial

La Figura 3.7 también ilustra el flujo de datos entre módulos: PortVideo obtiene los frames de video en escala de grises del sistema operativo y los pasa a libfidtrack. Un algoritmo de limitacion (thresholding) produce una imagen binaria que es alimentada al módulo de segmentacion para construir un grafico de proximidad regional. Este gráfico se pasa al modulo de reconocimiento fiducial el cual reconoce subgráficos cuya topología es igual a aquellas que se encuentran en un diccionario cargado de un fichero de descripción de arboles fiduciales.

La información que distingue a un fiducial de otro viene representada por un árbol que contiene la estructura de las zonas en blanco y negro que hay en él. La construcción del árbol se realiza tomando como nodo raíz el color más externo del fiducial, después se saca el número de zonas de color opuesto

que hay dentro de la zona inicial. Para cada nueva zona detectada se vuelve a realizar el paso anterior, y así hasta que todas las zonas del fiducial se han analizado. En la figura 3.8 se observa un fiducial y la representación correspondiente en forma de árbol.

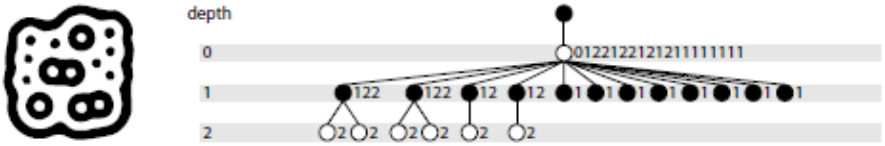


Figura 3.8: Arbol fiducial.
Figura extraída de [14]

Para evitar datos irrelevantes, como por ejemplo el color del fondo, se recurre a un proceso de minimización de los árboles consistente en la eliminación de las partes comunes, dejando únicamente las partes que difieren entre unos y otros. La representación óptima se realiza con 3 o más niveles de profundidad para tener suficiente robustez, ya que, usando menos, se pueden producir falsas identificaciones en la imagen captada, como considerarlo un fiducial distinto o no detectarlo.

Analizando el ejemplo de la figura 3.8 se puede eliminar el nodo raíz de color negro ya que la parte importante se encuentra en los nodos inferiores. El contorno del fiducial no aporta ninguna información sobre la diferenciación de éste con respecto a otro. Las zonas interiores son las que pueden variar y, por lo tanto, dar esa distinción.

Por otra parte el cálculo de la orientación puede ser o no ser necesario dependiendo del uso que se le vaya a dar al fiducial. Este cálculo se realiza a partir de los círculos de menor tamaño del fiducial tal como se describe a continuación: se calcula el centro de la unión de todos los círculos del mismo color, obteniendo dos centros, uno para los blancos y otro para los negros. Cuando se han obtenido ambos centros se calcula el vector orientación que va desde el centro blanco hacia el centro negro, tomando como 0^{o} el vector apuntando hacia arriba y aumentando con el giro en el sentido de las agujas del reloj.

Los identificadores fiduciales reconocidos, junto con su localización y orientación, se pasan a un modulo que trackea fiduciales segun vayan volviéndose visibles y desaparezcan. Este módulo también calcula información de segunda magnitud, tales como los vectores de velocidad y aceleración. Finalmente, esta información se formatea en paquetes de Open Sound Control, que conforman al protocolo TUIO, y se pasan a los clientes por via de un protocolo de red.

Una vez realizado el análisis de las clases de las que se compone el sistema en el apartado 2.7, se representa esquemáticamente cada clase y se muestran las relaciones existentes entre ellas.

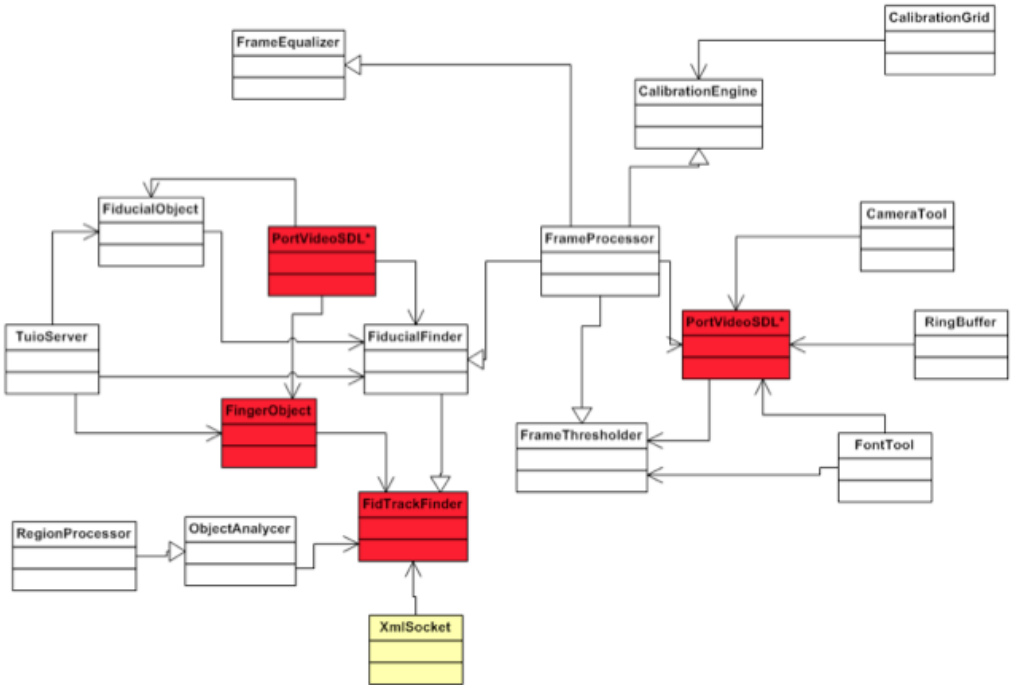


Figura 3.9: Diagrama de clases de Reactivision.

En la figura 3.9 se muestra la relación entre las clases de Reactivision, siendo el origen de la flecha la clase que proporciona información a la clase a la que le llega la punta de la flecha. Las flechas con que tienen las puntas huecas indican herencia, la clase a la que la flecha llega hereda de la flecha de la que sale la flecha. Por otro lado las clases que tienen el fondo de color rojo son las que especificaremos a continuación por ser las más importantes.

3.3.1.1. PortVideoSDL

Clase que actúa como el motor de tratamiento de la información de todo el framework.

Atributos:

- **Camera_:** Dirección de memoria de la cámara.
- **CameraBuffer_:** Buffer donde se almacena lo leído por la cámara
- **Camera_config:** Nombre del fichero de configuración de la cámara.
- **Running_, error_, pause_, calibrate_, help_:** Booleanos que indican si el framework se encuentra ejecutándose, si ha habido un error, si está pausado, calibrando o mostrando la ayuda.

- **Framenumber_:** Número de frame que se está tratando.
- **RingBuffer:** Buffer que almacena hasta 3 frames indicando cual es el siguiente que debe ser leído.
- **Current_fps:** Frames por segundo a los que está trabajando la cámara
- **Display_lock:** Booleano que indica si el acceso al display está siendo usado por algún procesador o si se encuentra libre.

Métodos:

- **PortVideoSDL(const char* name, bool background, const char* camera_config):** Constructor de la clase.
- **Run():** Activa el funcionamiento del framework Reactivision.
- **Stop():** Desactiva el funcionamiento del framework
- **AddFrameProcessor(FrameProcessor *fp):** Añade un procesador de frames al motor.
- **RemoveFrameProcessor(FrameProcessor *fp):** Elimina un procesador de frames del motor.
- **SetMessage(std::string message):** Escribe el mensaje por la consola de comandos.
- **DisplayMessage(const char *message):** Escribe el mensaje en el display del framework.
- **SetDisplayMode(DisplayMode mode):** Establece que está mostrando el display: la imagen original, la imagen umbralizada o nada.
- **GetDisplayMode():** Devuelve el modo en el que se está mostrando el display.
- **CurrentTime():** Devuelve el tiempo que lleva funcionando el framework.
- **SetupWindow():** Inicializa la ventana del display en la que se mostrará la información de la cámara y de la detección.
- **TeardownWindow():** Elimina la ventana del display.
- **SetupCamera():** Inicializa la cámara para que proceda a la captura de imágenes.
- **TeardownCamera():** Deja de usar la cámara.
- **InitFrameProcessors():** Inicializa todos los procesadores de frames que tiene añadidos. Si alguno no se puede inicializar, lo elimina.
- **AllocateBuffers():** Establece el tamaño de los buffers de origen, destino y display y los reserva en memoria.

- **FreeBuffers():** Elimina de memoria el espacio reservado para los buffers.
- **EndLoop():** Finaliza la ejecución de los procesadores de información, mostrando por pantalla un mensaje de error si han finalizado por esa razón.
- **Process_events():** Gestiona el manejo de eventos, que en este caso son las acciones que activan la pulsación de las teclas de opciones.

3.3.1.2. FidtrackFinder

Clase que se encarga de buscar fiduciales y objetos sin identificador en los frames.

Atributos:

- **detect_finger:** Booleano que active o desactiva la detección de dedos.
- **average_leaf_size:** Indica el tamaño de los nodos hoja de los fiduciales.
- **average_fiducial_size:** Indica el tamaño que deben tener los fiduciales.
- **average_finger_size:** Indica el tamaño que deben tener los dedos de las manos para ser reconocidos.
- **finger_sensitivity:** Indica la tolerancia de error del tamaño de los dedos para ser reconocidos.
- **min_handler_size:** Tamaño mínimo que debe tener un objeto sin identificador con un fiducial.
- **max_handler_size:** Tamaño máximo que debe tener un objeto sin identificador con un fiducial.
- **paper_height:** Indica la altura del folio que contenga el dibujo a reconocer.
- **paper_width:** Indica la anchura del folio que contenga el dibujo a reconocer.
- **fid_size:** Tamaño que debe tener el fiducial que identifica al dibujo a reconocer.
- **setFingerSize, setFingerSensitivity, setHandlerSize, setMinimun, setPaperBlockSize, setHeight, setWidth, setFiducialSize:** Booleanos que indican que valor a de aparecer en el display para su configuración.

Métodos:

- **Init(int w ,int h, int sb, int db):** Inicializa el procesador de detección.
- **GetFingerSize():** Devuelve el valor del tamaño de los dedos que se quieren detectar.

- **GetFingerSensitivity():** Devuelve la variación de tamaño que puede tener un dedo para que, aunque no sea del tamaño exacto, se pueda detectar.
- **FidtrackFinder(MessageServer *server, const char* tree_cfg, const char* grid_cfg, int finger_size, int finger_sens):** Constructor de la clase que da valor al servidor que envía los mensajes con lo detectado (server), al fichero de configuración de los árboles de los fiduciales (tree_cfg), a la superficie calibrada (grid_cfg), al tamaño de los dedos que se van a detectar (finger_size) y a la variación de tamaño que se establece para detectar un dedo (finger_sens). Además realiza la llamada a la conexión socket, en el caso de que esta opción esté activada.
- **ToggleFlag(int flag):** Realiza la gestión de teclas para la configuración del detector. Cuando se están modificando los valores de configuración no se puede llamar a ninguna de las otras opciones de modificación de Reactivision hasta que no se termine de darles valores.

3.3.1.3. FingerObject

Clase que representa un dedo de la mano con sus propiedades.

Atributos:

- **Alive:** Indica si el objeto reconocido se encuentra activo en el último frame analizado.
- **Unsent:** Indica si el objeto reconocido está siendo detectado, pero todavía no se ha enviado información sobre él.
- **Session_id:** Identificador de sesión del objeto reconocido.
- **State:** Estado del objeto reconocido: añadido, borrado, expirado o vivo.
- **Smallest_area:** mínima área que el objeto reconocido tiene que tener para ser detectado.
- **Xpos, ypos:** posición en pantalla del objeto reconocido. Incluye un campo que contiene la información sobre la orientación.

Métodos:

- **FingerObject(int width, int height):** Constructor de la clase usando como parámetros su altura y anchura (width, height).
- **RedundantSetMessage(TuioServer *server):** Crea un mensaje para enviarlo mediante protocolo TUIO cuando no se ha modificado ninguno de sus parámetros.
- **GetStatistics():** Devuelve una cadena con toda la información del objeto reconocido.

- **CheckStatus(int s_id):** Devuelve el estado del objeto reconocido indicado (s_id). Este estado puede ser: añadido, borrado, expirado o vivo.
- **Update(float xpos, float ypos, float area, float orientacion):** Actualiza la información del objeto reconocido, en la que también se incluye la orientación.
- **AddSetMessage(TuioServer *tserver):** Crea un mensaje para enviarlo mediante el protocolo TUIO a otra aplicación cada vez que el objeto reconocido se actualiza. En el mensaje también se añade la información sobre la orientación del objeto reconocido.
- **Distance(float x, float y):** Calcula la distancia entre un punto de la pantalla (x, y) y el centro del objeto reconocido.
- **Reset():** Inicializa la información del objeto reconocido.
- **GetX():** Devuelve la posición horizontal del centro del objeto reconocido.
- **GetY():** Devuelve la posición vertical del centro del objeto reconocido.

3.4. Reconocimiento de los fiduciales

Tras capturar la imagen Reactivision la pasa a través de una serie de procesadores que la transforman, para después obtener información de esta. Las seis fases que conforman el framework en orden de procesamiento son: captura, ecualización, umbralización, detección de fiduciales, calibración y envío de mensajes.

1.- **Captura de la imagen:** El primer paso consiste en la captura de una imagen y su almacenamiento en un buffer de imágenes.

2.- **Ecualización:** Es la fase que se encarga de tratar la imagen inicial que le envía la cámara eliminando los elementos erróneos de esta.

3.- **Umbralización:** Se encarga de pasar la imagen a escala de grises para convertirla en una imagen binaria, es decir, solo en dos colores, negro y blanco. Para ello realiza una umbralización de la imagen separando el fondo de los objetos, siendo el fondo negro y los objetos de color blanco.

4.- **Detección de fiduciales:** En esta fase es donde se analiza la imagen para detectar los fiduciales y objetos que se coloquen en la superficie de la pantalla.

5.- **Calibración:** Se encarga de corregir la aberración de la cámara que provoca que la geometría de los objetos captados no sea la correcta. Esto se hace a través de una transformación de coordenadas, de manera que exista una correspondencia entre las coordenadas reales y las captadas por la cámara.

6.- **Envío de mensajes:** Por ultimo toda la información obtenida después del tratamiento se envía a las aplicaciones a través del protocolo TUIO.

En la figura 3.10 se muestra un diagrama del procesamiento de la imagen capturada.

Figura 3.10: Paso de la información en Reactivision.

En cada ejecución y cuando los valores del fichero de configuración han sido cargados, es cuando se comienza a trabajar en la detección de objetos. En primer lugar se comprueba si la cámara está conectada, de lo contrario se informará de ello mediante un mensaje de error. A continuación se muestra por pantalla la resolución de la cámara y el tipo de esta, así como los diferentes comandos de configuración del framework.

Las fases de ecualización, umbralización, detección de fiduciales y calibración se pueden configurar. Esta configuración, como se ha explicado anteriormente se guarda después de cada ejecución, de manera que se pueden volver a recuperarse los parámetros de cada procesador en usos posteriores. La configuración se guarda en un fichero XML, que almacena los valores de las variables de cada procesador, así

como el valor de las variables que se vayan a necesitar. Reactivision es el encargado de cargar este fichero de configuración al iniciar su ejecución.

3.4.1. Fases de Reactivision

A continuación se van a explicar más en detalle cada una de las fases que conforman el framework Reactivision.

3.4.1.1. Captura de imágenes

La captura de imágenes a través de la cámara se realiza de forma paralela al programa principal, ya que se necesita estar tomando imágenes en todo momento para actualizar los valores de posición, orientación, etc de los objetos así como reconocer los nuevos objetos colocados entre captura y captura. Si la captura de imágenes no se hiciese de forma paralela al procesamiento, el programa principal no podría tratar la imagen en ningún momento o por el contrario al no tomar imágenes continuamente se estarían perdiendo parte de las interacciones con la mesa. Este proceso captura frames y los va guardando en un buffer, pudiendo almacenar 3 imágenes de forma simultánea. Si el buffer se encuentra lleno, se realiza una comprobación del estado de la cámara y si ésta sigue en funcionamiento se espera un tiempo de 5 segundos y después se vuelve a intentar guardar una nueva imagen, de esta manera se evita la saturación del buffer. En caso de que la cámara no responda, se aborta la captura de imágenes.

3.4.1.2. Ecualización

Este es la primera fase en la que se modifica la imagen. Sirve para eliminar objetos erróneos o no necesarios en la imagen. Al capturar por primera vez una imagen, ésta contendrá ruido y errores por la mala iluminación o calibración de la cámara. Para evitar en gran medida la captura de ruido, este proceso indica al framework que la primera imagen que tome deberá ser tratada como fondo, de esta manera toda la información que sea errónea será ignorada. Este proceso se puede activar y desactivar pulsando la tecla é'. El fondo se puede poner a negro pulsado la barra espaciadora, de esta manera se evitaran falsos positivos en el reconocimiento ya que esta será la imagen tomada como fondo. En la figura 3.11 se puede ver cómo se elimina el ruido y errores en la imagen pulsando la tecla espaciadora y así indicando al ecualizador que el fondo es todo negro.

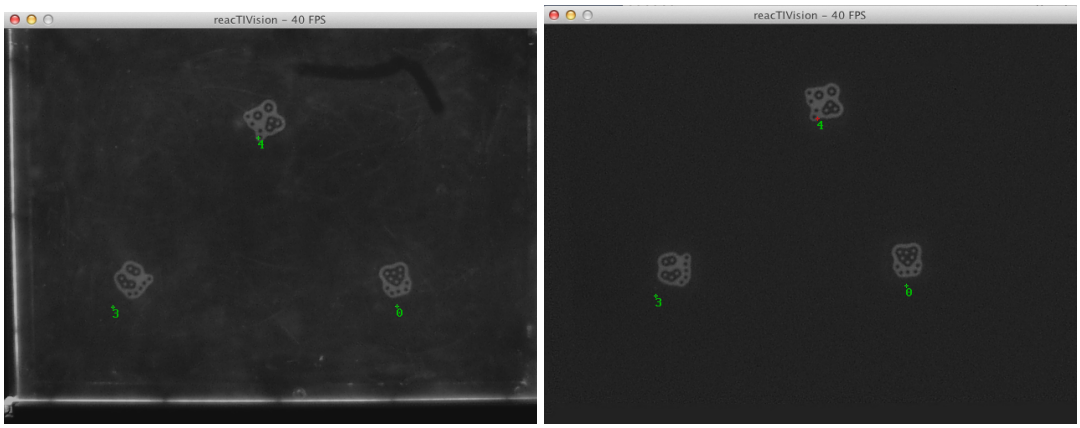


Figura 3.11: ReacTIVision capturando ruido y su posterior ecualización.

3.4.1.3. Umbralización

Este fase consiste en umbralizar la imagen ecualizada para transformarla en una imagen binaria, es decir en blanco y negro. El procedimiento consiste en ir recorriendo la imagen formando regiones de 6x6 píxeles junto a sus 8 vecinos. En cada región se comprueba el mínimo y el máximo valor de los píxeles y después se compara la diferencia de dichos valores con el valor del gradiente. Si esta diferencia es mayor que el gradiente, se considera que la región pertenece al fondo y por tanto se le asigna el color negro. Si por el contrario es menor que el gradiente, se comprueba si esta diferencia es menor que el valor 127, valor medio del total de escala de grises, si es menor pasa a ser blanco y sino a negro. El valor del gradiente puede modificarse pulsando la tecla 'g'. En la figura 3.12 se muestra una imagen ecualizada y el resultado tras ser umbralizada.

3.4.1.4. Detección de fiduciales

Tras la umbralización de la imagen, se pasa a buscar los fiduciales y los objetos sin marcar. Para la identificación, lo primero que se realiza es una segmentación de la imagen, es decir una división de la imagen en regiones. Se comienza recorriendo la primera línea de la imagen pixel a pixel. En el momento que se detecta un pixel, se crea una nueva región si éste es de distinto color al anterior pixel o actualizamos la región del pixel anterior si es del mismo color. Tras analizar la primera línea, se pasa a realizar el mismo proceso con las siguientes líneas, con la salvedad de que ahora también se comprueba el pixel superior al que se está analizando, además del anterior. Cuando un pixel comparte región tanto con su pixel superior como el de su izquierda, se procede a una fusión de sus regiones, dando lugar a una nueva.

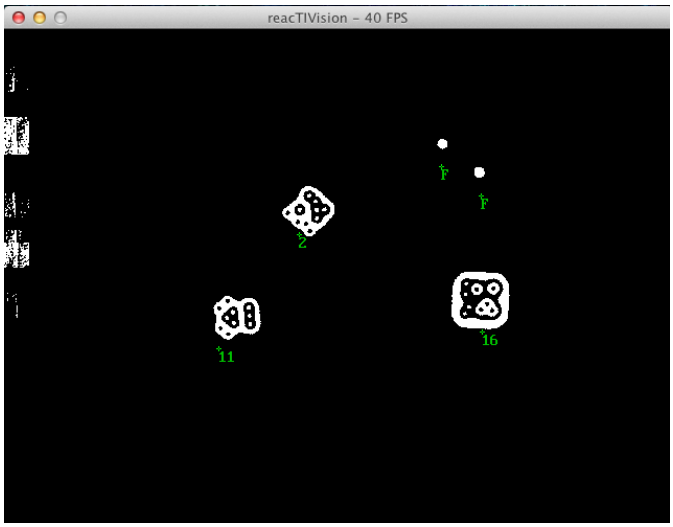


Figura 3.12: Imagen umbralizada en reacTIVision.

Una vez segmentada la imagen, se procede a la búsqueda de los fiduciales que coincida con los almacenados en el fichero de árboles que contiene el framework al ejecutarse. Una vez reconocidos todos los fiduciales colocados sobre la superficie de la mesa, se comprueba que estos fiduciales sean válidos, nuevos o si ya habían sido reconocidos. Para ello lo que el algoritmo hace es comprobar los fiduciales reconocidos con los reconocidos en el frame anterior. Uno a uno comprueba los fiduciales con los ya reconocidos, si se produce una coincidencia se compara la distancia entre los dos y si es menor que el tamaño mínimo de un fiducial, se entiende que se trata del mismo objeto y que simplemente ha sido movido de un frame a otro, por lo que se procede a actualizar sus valores de posición y orientación. Si por el contrario no coincide con ninguno de los fiduciales ya detectados, se añade este a la lista de nuevos fiduciales.

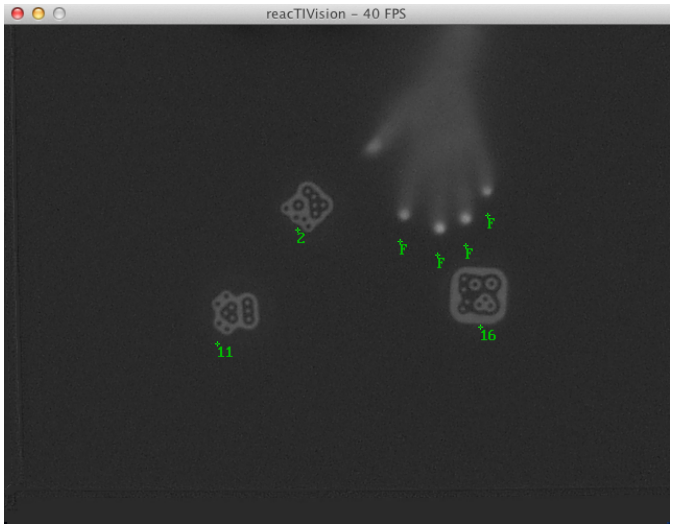


Figura 3.13: Reconocimiento de fiduciales y dedos en reacTIVision.

Para evitar falsos positivos y no confundir los nuevos fiduciales con los ya detectados se procede a realizar una comprobación similar a la explicada en el párrafo anterior.

Por último, se buscan los eventos táctiles. Para ello se examinan las regiones de color blanco y de un determinado tamaño. El tamaño que puede tener un blobs se puede modificar a través de la interfaz de usuario. Una vez detectada las regiones del tamaño establecido, se comprueba que no se trata de objetos que ya se habían detectado anteriormente, para ello se compara la distancia entre cada uno de los nuevos objetos y los ya detectados. Por último y una vez detectados todos los objetos sin fiducial, se procede a extraer las características de estos.

Tras haber reconocido todos los objetos situados en la superficie de la mesa, se procede a identificarlos gráficamente en la pantalla de Reactivision. Los fiduciales serán identificados con un número, según su posición en el fichero de árboles antes mencionado. Los dedos de las manos se mostrarán con la letra 'F'. En la figura 3.13 se muestra un ejemplo de los fiduciales que se han reconocido identificados con su correspondiente número, así como los dedos identificados con la letra 'F'.

3.4.1.5. Calibración

Por último se procede a la calibración de la imagen. Esta fase consiste en una conversión entre las coordenadas reales y las obtenidas por la cámara estableciendo así una relación entre ellas. Esto es necesario ya que las cámaras producen aberraciones ópticas, distorsionando la imagen respecto a la realidad, por lo que a la hora de representar elementos en unas coordenadas determinadas, este objeto puede no quedar situado en la posición esperada. El calibrador puede activarse pulsando la tecla 'C'. Al hacerlo se muestra una rejilla formada por líneas y puntos que se pueden mover por la pantalla. El desplazamiento y modificación de los puntos provoca que las zonas de la imagen dada por la cámara que coincidan con esos puntos, se deformen hasta los nuevos puntos establecidos. En la figura 3.14 se observa una imagen sin calibrar y el efecto que se produce al mover los puntos de la rejilla para ajustarlos a la imagen real.

Las líneas azules y verdes corresponden al calibrador. Las líneas en negro pertenecen a la imagen real y de cómo tendrían que ser las líneas del calibrador. Se puede ver que no coinciden unas con otras.

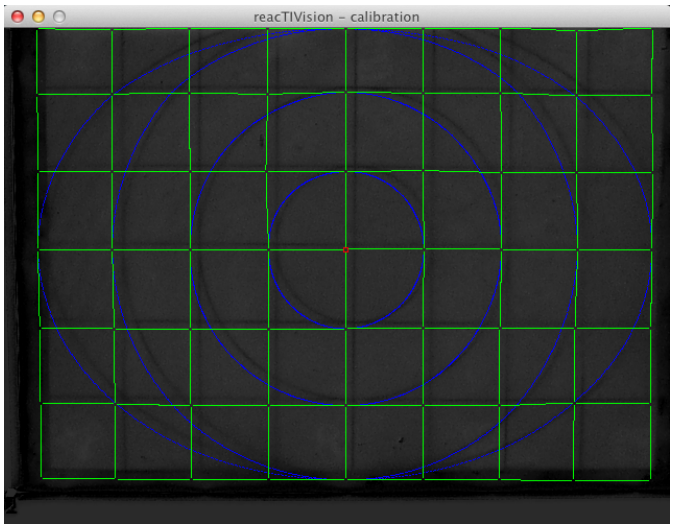


Figura 3.14: Cámara sin calibrar en reacTIVision.

Después de calibrar, se puede comprobar en la figura 3.15 que las líneas negras corresponden con las del calibrador.

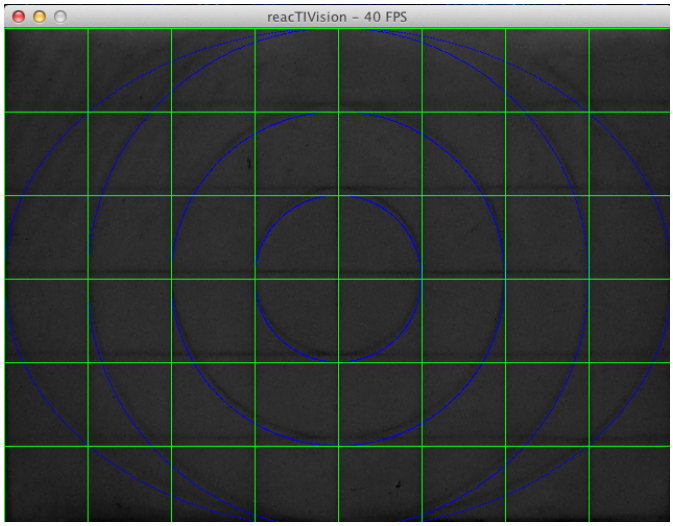


Figura 3.15: Cámara calibrada en reacTIVision.

3.4.1.6. Envío de mensajes

Por último toda la información obtenida después del tratamiento se envía a las aplicaciones a través del protocolo XML y el protocolo TUIO detallado anteriormente.

3.5. Módulo de interpretación de mensajes MIDI y Audio

En este apartado se describen las diferentes partes de las que consta el núcleo del sistema, y a través de las cuales se realizan los procesos principales de interpretación de mensajes MIDI y audio de la aplicación. Podemos hacer una separación bien definida de la arquitectura interna en 2 partes:

- E/S de audio.
- Soporte MIDI.

3.5.1. E/S de audio

El Sistema debe cumplir con los siguientes requisitos de entrada/salida de audio:

- Carga de archivos de audio en formato wave.
- Almacenamiento de archivos de audio en formato wave.
- Reproducción de audio.
  - **1**.- Reproducción del audio original.
  - **2**.- Reproducción de frecuencia fundamental contenida en el audio a lo largo del tiempo.
  - **3**.- Reproducción de la frecuencia fundamental después de la segmentación (melodía).

Observando estos requerimientos, nos damos cuenta de que la aplicación debe ofrecer soporte completo tanto para la entrada como para la salida de audio, debiendo acceder a dispositivos que permitan la reproducción y captura de audio, así como controlar el acceso a disco para lectura y escritura de archivos.

Para llevar a cabo las funcionalidades que satisfacen todos los requisitos de E/S de audio para el Sistema, asignaremos responsabilidades a las siguientes clases, cada una de las cuales se encargará de realizar una tarea concreta tal como se enumera a continuación:

- AudioFileIO: Cargar y almacenar archivos de audio en formato wave.
- AudioPlayer: Reproducir audio original.
- FundPlayer: Reproducir la frecuencia fundamental contenida en el audio original.
- FundSegPlayer: Reproducir la frecuencia fundamental obtenida en el proceso de segmentación; es decir, la serie de notas que forman la melodía.

Asignar una tarea concreta a cada clase favorece un Bajo Acoplamiento, dado que cada objeto se encargará por sí mismo de realizar su trabajo sin depender de la existencia de otros. Además de esta ventaja, este enfoque permite una mayor facilidad en el mantenimiento del código y en la identificación de errores. Por otro lado, este conjunto de clases, las cuales constituyen el motor de entrada/salida de audio de la aplicación, forma un módulo con Alta Cohesión, ya que cada objeto se encarga de realizar una tarea concreta, y entre todos dan soporte completo para la E/S de audio del programa. Estos criterios constituyen principios básicos que deben ser tenidos en cuenta en el desarrollo de todo sistema orientado a objetos y serán continuamente empleados durante todo el diseño del sistema. A continuación, describiremos de forma breve y conceptual el funcionamiento interno de cada clase.

Para satisfacer los requerimientos de reproducción y captura de audio del sistema, emplearemos como parte principal, objetos de la clase AudioOut y AudioIn, ofrecidos por CLAM y derivados de la clase Processing.

La entrada/salida de audio en CLAM está gestionada por la clase AudioManager, la cual también es un proceso y debe iniciarse antes de que entren en acción las clases AudioIn o AudioOut según sea el caso. El manejador de audio se encarga, entre otras cosas, de asignar la velocidad de muestreo y el tamaño de cada frame que trabajará con el dispositivo de entrada o salida.

En el caso de la captura de audio, se inicializa el manejador con una velocidad de muestreo de 11025 Hz y un tamaño de frame de 2048 muestras. El objeto de la clase AudioIn, utilizado para obtener el audio, se configura como una entrada para un sólo canal (mono). Una vez configurados e iniciados el manejador de audio y el canal de entrada, se entra en un bucle donde se obtienen los frames de audio mediante el canal de entrada (objeto de la clase AudioIn), y se van colocando en un objeto de la clase Audio. El procedimiento de grabación continuará hasta que sea detenido por un evento externo; en este caso lo normal es que sea detenido por el usuario.

Todos los métodos y objetos relacionados con la captura de audio se encuentran encapsulados en la clase AudioRecorder, la cual también incorpora un método para servir el audio registrado al cliente que lo demande.

Para la reproducción de audio, deben satisfacerse requisitos que permitan tres modos de reproducción: audio original, frecuencia fundamental y frecuencia fundamental de cada nota de la melodía obtenida tras el proceso de segmentación. El método principal de cada modo de reproducción es diferente. Para el caso de la reproducción de audio original, se inicializa el manejador de audio y un objeto de la clase AudioOut, usado como canal de salida, de forma similar a como se hace en el caso de la captura. A continuación, se entra en un bucle donde se realiza la fragmentación del audio original, y el envío secuencial de los diferentes trozos, para ser reproducidos a través del proceso que está siendo ejecutado por el objeto de la clase AudioOut.

En el caso de la reproducción de la frecuencia fundamental y de la melodía obtenida tras la segmentación, además del manejador de audio y del canal de salida (objeto de la clase AudioOut), se utiliza un oscilador para generar la sinusoide pura correspondiente a la frecuencia deseada en cada momento. CLAM incorpora clases para cubrir estas necesidades; en nuestro caso haremos uso de la clase SimpleOs-

cillator, que al ser un proceso también deriva de Processing, y tiene asociada su correspondiente clase de configuración SimpleOscillatorConfig.

La clase SimpleOscillator es controlada a través de clases control. En este caso, obtendremos el control adecuado con el fin de controlar la frecuencia de oscilación que nos interesa en cada momento durante el curso de la reproducción.

Al igual que en el caso del la reproducción de audio original, se configuran e inicializan el manejador y el canal de salida. Además de esto, se configura e inicializa el oscilador, del cual se obtiene el control adecuado con el fin de controlar la frecuencia de oscilación. A continuación se entra en un bucle en el cual se realiza todo el proceso.

En el caso de la reproducción de la frecuencia fundamental, los datos necesarios son la frecuencia contenida en los frames resultantes del análisis, los cuales se mantienen en un objeto de la clase Frame, la cual deriva de ProcessingData, y que a su vez forma parte del segmento usado como estructura para mantener los datos, es decir, un objeto de la clase Segment que también deriva de ProcessingData.

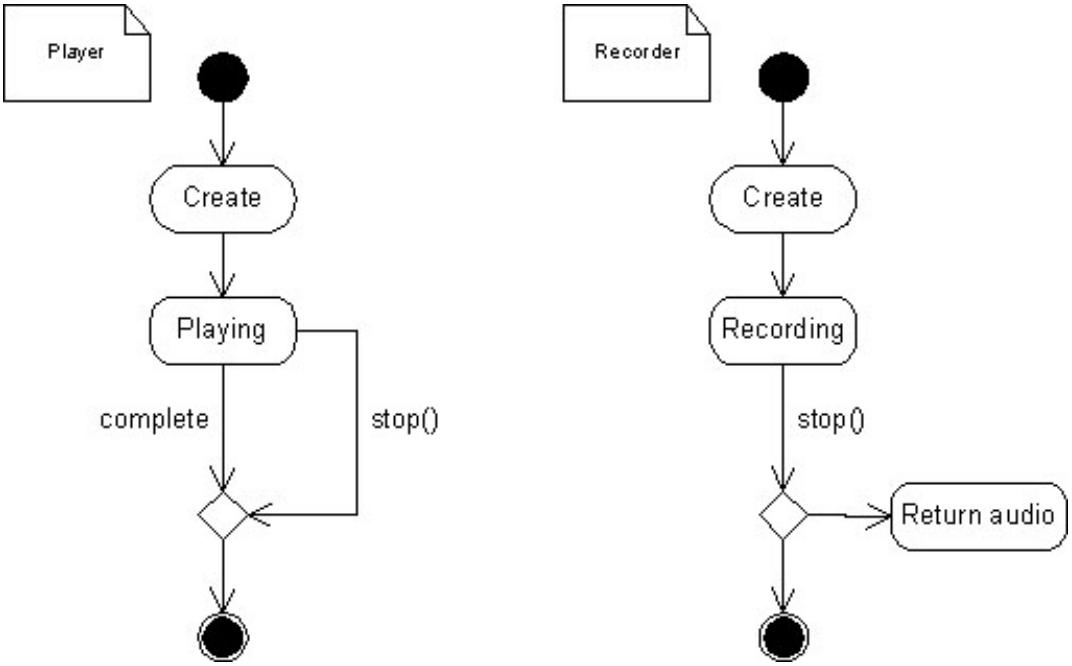


Figura 3.16: Diagrama de estados simplificado para la reproducción y captura de audio.

Los datos utilizados en la reproducción de la melodía obtenida después del proceso de segmentación, son mantenidos en una estructura melodía, para lo cual se utiliza un objeto de la clase Melody, derivada de ProcessingData. La melodía que será reproducida es servida por un objeto de la clase MelodyAnalyzer, la cual se comentará en el siguiente apartado. El objeto de la clase Melody, entre otras cosas, mantiene toda la información que necesitamos para la reproducción, es decir, la lista de notas que forman la melodía. La información de cada una de las notas se encuentra en un objeto de la clase Note, la cual deriva de

ProcessingData. Para el caso de la reproducción, la información que utilizaremos será la relativa a la frecuencia fundamental y a los instantes de tiempo inicial y final de cada nota.

En el diagrama de estados de la figura 3.17, se muestra de forma esquemática el funcionamiento de los procesos de reproducción y captura de audio. Dado que estos procesos son lanzados como hilos de ejecución, tal como se ha comentado anteriormente, se ha optado por iniciar el proceso al instanciar el objeto, es decir, el objeto se crea e inmediatamente después comienza la ejecución del proceso, y cuando el proceso finaliza, se destruye también el objeto. Entonces, los datos necesarios para cada caso se deben pasar como parámetros al constructor de cada objeto.

A continuación se muestra un diagrama estático con el fin de representar el conjunto de clases que formarán el módulo de entrada/salida de audio. Nótese el modo en que las clases activas, es decir, las que tienen su propio hilo de ejecución, pueden ser representadas mediante el lenguaje UML.

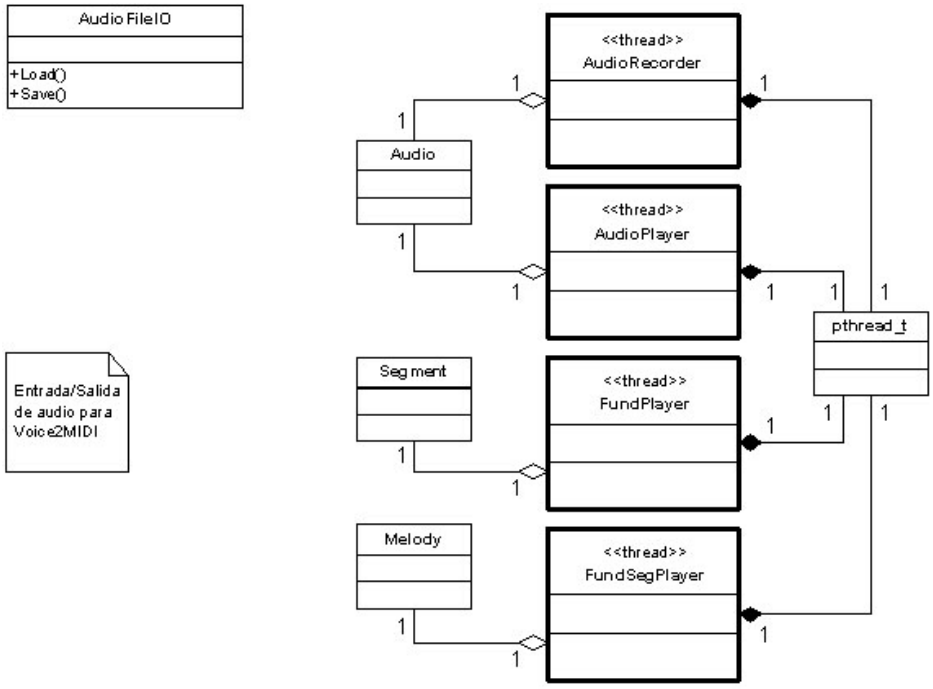


Figura 3.17: Diagrama de clases simplificado para E/S de audio.

En el diagrama de clases de la figura 3.17, se han representado los conceptos más relevantes, con el fin de favorecer la claridad, dado que indicar todos los métodos y atributos de las clases en un solo diagrama, desembocaría en un diagrama complejo y de difícil lectura. En el caso de las clases de este módulo, la implementación ha resultado relativamente sencilla y de código bastante escueto, pero en otros módulos, como por ejemplo los de visualización, entran en juego bastantes métodos y atributos como para representarlos por entero y de forma clara en un espacio reducido; por consiguiente, tanto en este capítulo como en los posteriores se muestra una representación algo simplificada con el fin de favorecer su interpretación.

3.5.2. Soporte MIDI

El proceso de conversión se realiza de una forma realmente sencilla. Tenemos una melodía disponible, fruto del proceso de extracción comentado en el apartado anterior. Entonces a partir de los valores contenidos en dicha melodía, podemos obtener otra melodía equivalente en valores MIDI.

Como se ha indicado anteriormente, los valores obtenidos en el proceso de extracción de la melodía (frecuencia, energía ...) se mantienen en un objeto de la clase Melody. A partir de estos valores puede calcularse fácilmente un equivalente MIDI, y estos nuevos valores calculados pueden ser almacenados en un objeto de la clase MIDIMelody, también disponible en CLAM. La pregunta es: ¿quién debe hacerse cargo de realizar esta tarea? Una buena candidata para ello es la clase MelodyAnalyzer comentada en el apartado anterior, dado que precisamente es en el interior de esta clase donde se obtiene la melodía en el formato guardado en el objeto de la clase Melody; así pues, el mapeo a valores MIDI se hará paralelamente a la obtención de la melodía en su formato inicial.

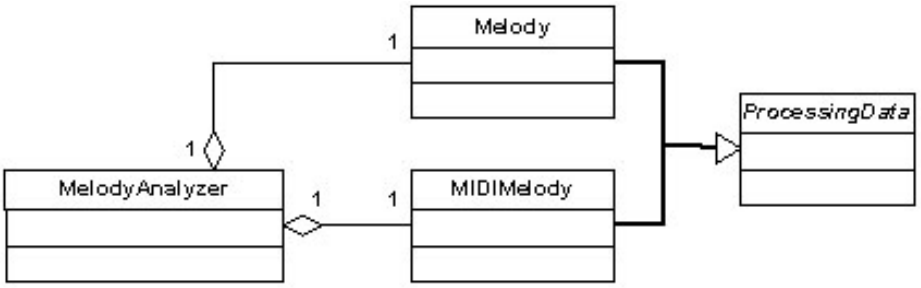


Figura 3.18: Clase MelodyAnalyzer.

En la clase Melody, se guarda una lista con todas las notas que forman la melodía, cuya información se encuentra en objetos de la clase Note. La clase Note, incorpora un método para el cálculo del valor MIDI de la nota a partir del valor de su frecuencia, y otro para obtener el valor de la velocidad de la nota a partir de su energía; entonces lo único que debemos hacer es usar estos métodos para llenar una estructura de la clase MIDIMelody, con tal de obtener el equivalente en representación MIDI. De esta forma ya tenemos disponible una melodía MIDI, a partir de la cual podemos realizar el proceso de reproducción de la misma y el almacenamiento del archivo MIDI en disco; siendo los valores temporales de las notas, los mismos que se mantienen en el objeto de la clase Melody.

Figura 3.19: Conversión MIDI.

Para la reproducción de la melodía MIDI diseñaremos e implementaremos una nueva clase, a la cual

denominaremos MIDIMelodyPlayer. La reproducción MIDI, de igual modo en que se hace en el caso de la reproducción de audio, se realizará lanzando un hilo de ejecución (thread), con el fin de que el usuario pueda parar el proceso si así lo desea; por tanto, el diagrama de estados para este modo de reproducción es el mismo que el presentado para la reproducción de audio en la figura 3.19. Ya que el proceso de reproducción se inicia con la creación del objeto, los datos necesarios son pasados como parámetros al constructor de la clase. En este caso, los datos relevantes son la melodía MIDI, el identificador de dispositivo MIDI empleado y el programa (instrumento) con el cual se interpretará la melodía.

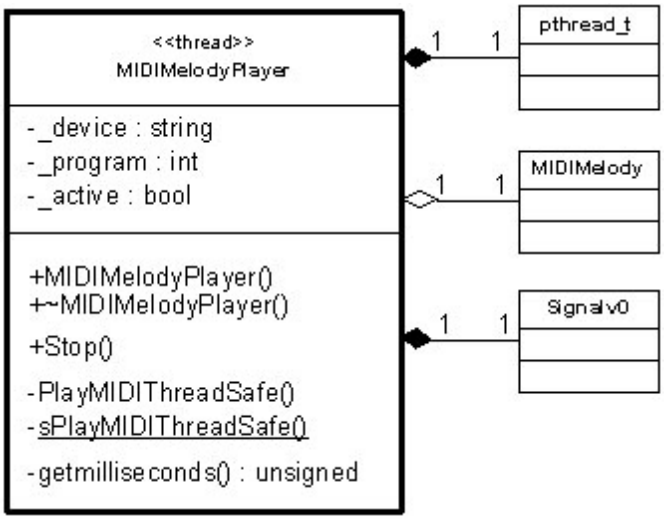


Figura 3.20: Diagrama estático para la clase MIDIMelodyPlayer.

En el diagrama de diseño de la clase MIDIMelodyPlayer, se han incorporado todos los métodos y atributos, tanto públicos como privados. Esto se ha hecho así porque al tratarse de un diagrama con una única clase, hay espacio suficiente y la comprensión resulta rápida y sencilla. Además, la explicación que se hará a continuación, también contribuye a entender de forma completa el funcionamiento de las clases para la reproducción de audio, que han sido presentadas de manera muy simplificada, dado que la idea que subyace en la implementación del reproductor MIDI es muy similar a la que se aplica para la reproducción de audio.

En el constructor de la clase MIDIMelodyPlayer, se crea un hilo de ejecución cuyo código es el contenido en el método estático sPlayMIDIThreadSafe. En este método estático, se hace una llamada al método PlayThreadSafe, en el interior del cual se llevan a cabo todas las acciones necesarias para la reproducción. Igual que sucede con el audio, el MIDI también tiene un manejador encargado de gestionar la E/S. Esta responsabilidad recae sobre la clase MIDIManager de CLAM, y en nuestro caso controlaremos la reproducción haciendo uso de la clase de control MIDIOutControl, la cual se configura por medio de un objeto de la clase MIDIIOConfig. Los objetos MIDIOutControl son configurados y empleados para el envío de mensajes MIDI. Para la reproducción de la melodía usaremos principalmente controles de volumen, programa (instrumento) y note on.

Este método recibe como parámetros: la melodía MIDI con la serie de notas que formarán la pista MIDI del archivo, el entero correspondiente al programa (instrumento) con el cual se interpretará la melodía, y el nombre del archivo a escribir. En nuestro caso, se trata de un archivo que contiene una sola pista, por lo que su estructura estará compuesta por la cabecera seguida de los datos correspondientes a dicha pista.

La cabecera se sitúa al inicio y contiene información básica relativa al archivo (formato, número de pistas y división). El identificador de cabecera (MThd) y la longitud de la misma se codifican como palabras de 32 bits, y para representar la parte de datos se emplean palabras de 16 bits. La parte de datos puede tomar diversos valores, según las necesidades del archivo a escribir; aquí se mencionan los empleados para el caso que nos ocupa.

Se utilizará el formato 0, que indica que el archivo contiene una única pista multicanal, el número de pistas es 1 como ya se ha indicado anteriormente, y la división, a la que asignaremos un valor de 96, indica la resolución en 'ticks' por cuarto de nota. Como no indicamos la signatura temporal ni el tempo de la pista, se le asignarán valores implícitos. Cuando no se define signatura temporal se le asigna el valor por defecto de 4/4, y en el caso del tempo, el valor es 120, expresado en beats por minuto. La cabecera de nuestro archivo tendrá el siguiente aspecto expresado en valores hexadecimales:

```
4D 54 68 64 MThd (palabra de 32 bits)
00 00 00 06 longitud de la cabecera en bytes (palabra de 32 bits)
00 00 formato 0 (palabra de 16 bits)
00 01 una pista (palabra de 16 bits)
00 60 96 ticks por cuarto de nota (palabra de 16 bits)
```

Como se puede deducir, para formar la cabecera necesitamos métodos adecuados para la escritura de palabras de 16 y de 32 bits; también se puede observar que las palabras de 32 bits no se tienen en cuenta para indicar el valor de longitud de la cabecera.

A continuación de la cabecera escribiremos la pista (track), donde estarán representados los mensajes MIDI que serán interpretados por algún reproductor MIDI o secuenciador. Una pista está compuesta de eventos, precedidos por un delta-time. El delta-time representa la cantidad de tiempo que transcurre entre el final de un evento y el inicio del evento siguiente.

Una pista puede contener distintos tipos de eventos:
[event] = [MIDI event] | [sysex event] | [meta-event]

En nuestro caso utilizaremos solamente eventos MIDI (MIDI event). Tenemos una pista en la que usaremos un único canal al cual asignaremos un determinado instrumento (programa), y un valor de volumen. El resto de eventos serán mensajes note on, ya que para el caso del note off usaremos un mensaje note on con valor de velocidad igual a cero.

La melodía MIDI que deseamos reproducir se encuentra almacenada en un objeto de la clase MIDI-Melody, donde los valores de inicio y fin de nota están expresados en segundos en coma flotante, por lo

que implementaremos un método que convierta los segundos a ticks con el fin de escribir el delta-time correspondiente para cada evento. Además, los valores de delta-time se deben representar como valores de longitud variable (en bytes) en el interior del archivo; entendiendo por longitud la cantidad de bytes necesarios para representar el valor; por tanto, también se debe disponer de un método para escribir el delta-time de cada evento como un valor expresado en longitud variable.

La escritura de la pista comienza con el identificador de pista (MTrk), seguido de la longitud en bytes de los datos que la forman. Estos valores son expresados en palabras de 32 bits, y al igual que ocurre en el caso de la cabecera, la cuenta real de bytes comienza a continuación. Al inicio del track todavía no conocemos su longitud, por lo que al término de la escritura de la pista, deberemos regresar a la posición adecuada en el archivo para escribir la longitud correcta.

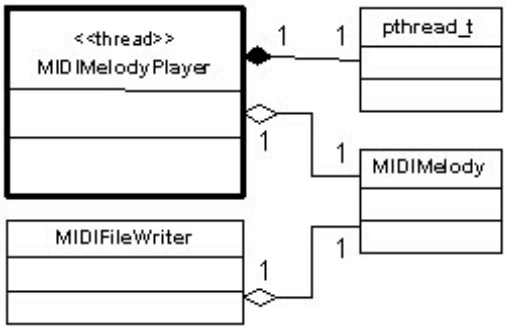


Figura 3.21: Diagrama de clases simplificado para el soporte MIDI.

Capítulo 4

Implementación

En este apartado se analizará la implementación de cada una de los módulos que conforman el diseño del proyecto. La tarea de implementación de un proyecto, consiste en la codificación de las clases y módulos, especificados y diseñados para cada capa, asá como la integración de todas ellas.

El primer módulo que se ha implementado es módulo de producción de contenidos visuales (capa de dominio), ya que de esta manera se puede ver la comunicación y el funcionamiento entre reacTIVision y el protocolo TUIO.

El siguiente módulo es el encargado de la recepción de mensajes OSC en Ableton Live!. Este módulo está basado en el paquete LiveOSC en el cual se han mejorado ciertos aspectos y se le han añadido nuevas funcionalidades para el control MIDI de dispositivos virtuales a través de macros programadas para funcionar con mensajes OSC.

El último módulo es el encargado del reconocimiento de gestos multitáctiles por parte del usuario. Se ha hecho uso de las funciones addTuioCursor, removeTuioCursor y updateTuioCursor explicadas en el capítulo anterior para que el usuario pueda silenciar objetos o controlar parámetros de amplitud.

A continuación se analizará, por separado, cada una de las tareas de implementación mencionadas.

4.1. Módulo de producción de contenidos visuales

4.1.1. Librerías

Como se ha comentado en el análisis de herramientas, para la implementación del módulo de contenidos visuales, usaremos la librería TUIO. En esta apartado también veremos que se han usado librerías

para el control de los archivos de audio WAV, aunque en principio todo esto será un prototipo, ya que realmente lo que se quería demostrar era el correcto funcionamiento de la librería reacTIVision y TUIO.

4.1.1.1. Sonia

La Biblioteca Sonia proporciona capacidades avanzadas de audio tales como reproducción de la muestra, síntesis de sonido en tiempo real, análisis FFT (frecuencia), análisis de la entrada de micrófono en tiempo real, y la escritura de archivos .wav a partir de muestras.

Para este prototipo hemos definido 11 muestras de audio wav, a las cuales hemos llamado mySound, tal y como se observa en el siguiente código

```
1 // Sound
2 import pitaru.sonia_v2_9.*;
3 Sample[] mySound = new Sample[11];
```

4.1.1.2. OscP5

oscP5 es una implementación del protocolo OSC para el entorno de programación Processing. OSC es el acrónimo de Open Sound Control, protocolo de red desarrollado en CNMAT, la Universidad de Berkeley. Open Sound Control es un protocolo para la comunicación entre ordenadores, sintetizadores de sonido y otros dispositivos multimedia que están optimizados para la tecnología de red modernos, y han sido utilizado en muchas áreas de aplicación.

```
1 //Osc Protocol
2 import oscP5.*;
3 import netP5.*;
4 OscP5 oscP5;
5 NetAddress myRemoteLocation;
```

oscP5 se utilizará para enviar datos por un puerto UDP y myRemoteLocation se utilizará para recibir datos mediante el protocolo UDP tal y como se describe en la especificación de la librería OscP5.

4.1.1.3. TUIO

En primer lugar se ha creado una instancia del cliente TUIO. El cliente TuioProcessing inmediantamente comienza a escuchar los mensajes entrantes y TUIO genera eventos de mayor nivel basado en los movimientos de los objetos y cursores.

Se han definido 4 variables:

1.- object_size;

2.- table_size;

3.- scale_factor;

```
25 //Tuio
26 import TUIO.*;
27 TuioProcessing tuioClient;
28 float object_size = 60;
29 float table_size = 760;
30 float scale_factor = 1;
31 PFont font;
```

4.1.1.4. Clase RotateShape

Se ha creado una clase llamada RotateShape, para dibujar alrededor de cada objeto tres curvas de tamaño reducido que irán girando en torno al círculo que envuelve a cada figura, de acuerdo a una velocidad definida. Al tratarse de un primer prototipo solo haremos uso de 11 objetos fiduciales, tal y como se muestra en el siguiente código

```
34 // RotateShape
35 RotateShape[] mybutton = new RotateShape[11];
```

El código desarrollado de dicha clase se encuentra en el fichero RotateShape.pde y la implementación es la siguiente:

```
1  class RotateShape{
2   color c;
3   float speed1,dim1;
4   float speed2,dim2;
5   float speed3,dim3;
6   int xpos,newposX;
7   int ypos,newposY;
8   //float diameter;
9
10
11  RotateShape(){
```

```
12      c = color(100);
13      dim1 = 5;
14      dim2 = 3;
15      dim3 = 1;
16      xpos = width/2;
17      ypos = height/2;
18    }
19
20    RotateShape(color tempC, int PositionX, int PositionY, float speed01,
21                float speed02, float speed03){
22      c = tempC;
23      dim1 = speed01;
24      dim2 = speed02;
25      dim3 = speed03;
26      xpos = PositionX;
27      ypos = PositionY;
28    }
29
30    void move(){
31    strokeWeight(5);
32    stroke(c);
33    fill(c);
34    smooth();
35
36    speed1 += dim1;
37    speed2 += dim2;
38    speed3 += dim3;
39    pushMatrix();
40
41    point(xpos,ypos);
42    //arc(xpos,ypos,100,100,radians(0+frameCount),radians(10+frameCount));
43    noFill();
44    arc(xpos,ypos,100,100,radians(0+speed1),radians(50+speed1));
45    arc(xpos,ypos,90,90,radians(0-speed1),radians(10-speed1));
46    arc(xpos,ypos,80,80,radians(0-speed3),radians(20-speed3));
47    popMatrix();
48
49    }
50  }
```

Como se observa en la figura 4.1 se han enumerado las 3 curvas que va a dibujar la clase RotateShape, estas estarán girando en torno al círculo que envuelve al objeto.

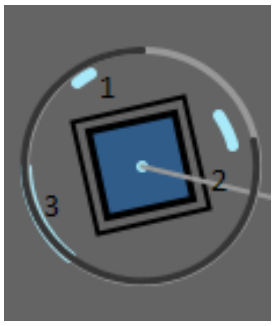


Figura 4.1: Visualización de RotateShape.

4.1.2. Setup()

Si hay una cosa muy específica de Processing son las funciones void setup() y void draw().

Lo importante es que setup y draw son las dos funciones principales de Processing. Las que se ejecutan siempre. Primero setup, una vez, y después draw repetidamente.

En la función setup vamos a configurar la presentación del módulo de producción de contenidos visuales, así que dentro de esta función se ha añadido lo siguiente:

```
47 void setup()
48 {
49   size(640,480);
50   smooth();
51   frameRate(30);
52   ....
```

- size() tiene que ser la primera instrucción que se da cuando programamos en Processing, esta función puede recibir dos o tres parámetros. Si recibe dos, se refieren a la medida de la ventana, que en nuestro caso será 640x480.
- smooth() activa el suavizado.
- framerate(30) indica que trabajaremos 30 frame por segundos.

A continuación se inicializa el protocolo TUIO tal y como se muestra en el siguiente fragmento de código.

```
53   ...
54   //init TUIO
```

```
55    hint(ENABLE_NATIVE_FONTS);
56    font = createFont("Arial", 18);
57    scale_factor = height/table_size;
58    tuioClient  = new TuioProcessing(this);
59    loop();
60    ...
```

- Con la función hint(ENABLE_NATIVE_FONTS) indicamos que se use la versión nativa de las fuentes en lugar de la versión del mapa de bits del fichero .vlw. Esto es útil con la configuración de render por defecto, ya que mejora la velocidad de la representación de las fuentes.
- font = createFont(Arial, 18) indica el tipo de fuente que vamos a usar, en este caso Arial a tamaño 18.
- scale_factor = height/table_size, se utiliza para escalar la información asociada a cada elemento.
- tuioClient = new TuioProcessing(this), crea una instancia del cliente TuioProcessing.

A continuación hacemos uso de la librería oscP5 donde se enviarán paquetes a traves del protocolo UDP por el puerto 9001 y se aceptarán paquetes a través del protocolo UDP por el puerto 9000

```
104 oscP5 = new OscP5(this,9001);
105 myRemoteLocation = new NetAddress("localhost",9000);
```

Una vez se ha inicializado el cliente TUIO, vamos a configurar los parámetros de cada slider que hemos definido haciendo uso de la librería control P5.

```
72    ...
73    controlP5=new ControlP5(this);
74
75    controlP5.addSlider("SR",0,255,128,10,400,5,50);
76    controlP5.addSlider("SG",0,255,128,40,400,5,50);
77    controlP5.addSlider("SB",0,255,128,70,400,5,50);
78
79    myKnobA=controlP5.addKnob("knobA",100,200,128,100,400,40);
80    myKnobB=controlP5.addKnob("knobB",100,200,128,150,400,40);
81    myKnobC=controlP5.addKnob("knobC",100,200,128,200,400,40);
82    ...
```

Como se puede apreciar en el código anterior, hemos creado 3 sliders para modificar los niveles de rojo, verde y azul, así como 3 botones, para controlar la iluminación.

Un control slider se utiliza ya sea de manera horizontal o vertical. Al agregar un control slider, el ancho se compara con la altura. Si la anchura es mayor, se obtiene un control slider horizontal, sin embargo si la altura es mayor, se obtiene un control slider vertical.

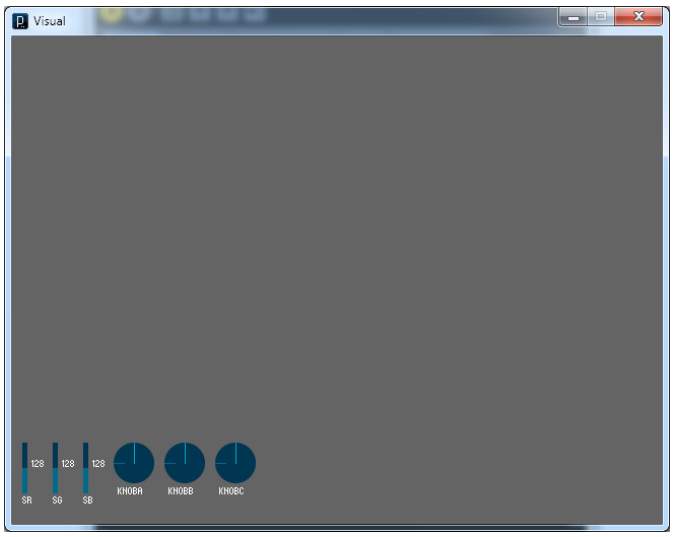


Figura 4.2: Pantalla de presentación.

En la figura 4.2 vemos como está configurada nuestra interfaz gráfica. En la esquina inferior izquierda encontramos los controles usados con la librería control P5, tanto los sliders como los botones.

Una vez tratada la interfaz gráfica nos faltará por inicializar los 11 objetos de la clase RotateShape que definimos al principio.

```
62    ...
63    // RotateShape
64    mybutton[0] = new RotateShape(color(169,234,250),0,0,2,1,0.5);
65    mybutton[1] = new RotateShape(color(32,224,75),0,0,0.5,2,1);
66    mybutton[2] = new RotateShape(color(224,170,32),0,0,0.2,2,3);
67    mybutton[3] = new RotateShape(color(214,32,224),0,0,0.5,2,1);
68    mybutton[4] = new RotateShape(color(56,32,224),0,0,0.5,2,1);
69    mybutton[5] = new RotateShape(color(220,242,54),0,0,0.5,2,1);
70    mybutton[6] = new RotateShape(color(220,242,54),0,0,0.5,2,1);
71    mybutton[7] = new RotateShape(color(220,242,54),0,0,0.5,2,1);
72    mybutton[8] = new RotateShape(color(220,242,54),0,0,0.5,2,1);
73    mybutton[9] = new RotateShape(color(220,242,54),0,0,0.5,2,1);
74    mybutton[10] = new RotateShape(color(220,242,54),0,0,0.5,2,1);
75    ...
```

4.1.3. Draw ()

Dentro de la función Draw() tratamos un vector de TuioObject y TuioCursor desde el cliente Tuio-Processing y se hace uso de un bucle sobre las dos listas para dibujar el feedback gráfico.

```
129 void draw()
130 {
131   background(SR,SB,SG);
132   textFont(font,18*scale_factor);
133   float obj_size = object_size*scale_factor;
134
135   //Marker
136   Vector tuioObjectList = tuioClient.getTuioObjects();
```

Como se ha comentado anteriormente, tuioObjectList es un vector de TuioObjects. La función get-TuioObjects() devuelve el TuioObject correspondiente a la ID de la sesión prevista o NULL si el ID de la sesión no se refiere a un TuioObject activo.

A continuación se realiza un bucle para ir dibujando los objetos, que mediante el uso de fiduciales, reconoce la librería reacTIVision

```
138 for (int i=0;i<tuioObjectList.size();i++) {
139      TuioObject tobj = (TuioObject)tuioObjectList.elementAt(i);
140
141      if(tobj.getSymbolID()>0 || tobj.getSymbolID()<=numFid){
142
143
144      // Basic Rect
145      pushMatrix();
146      stroke(0);
147      fill(randomRGBColor());
148      translate(tobj.getScreenX(width),tobj.getScreenY(height));
149      rotate(tobj.getAngle());
150      rectMode(CORNER);
151      rect(-obj_size/2,-obj_size/2,obj_size,obj_size);
152      popMatrix();
153
154      // RotateShape
155      pushMatrix();
156      noFill();
157      stroke(0);
158      translate(tobj.getScreenX(width),tobj.getScreenY(height));
159      mybutton[i].move();
160      popMatrix();
```

```
161
162       // Draw maximum lines
163       pushMatrix();
164       stroke(randomRGBColor());
165       translate(tobj.getScreenX(width),tobj.getScreenY(height));
166       rotate(tobj.getAngle());
167       rectMode(CENTER);
168       strokeWeight(2);
169       rect(0, 0 ,maxY,maxY);
170
171       // Draw frequency lines
172       stroke(0);
173       strokeWeight(2);
174       rect(0, 0 ,-freY+50,-freY+50);
175       popMatrix();
176
177      }else{
178         stop();
179     }
```

En este bucle vamos a ir recorriendo todos los elementos del vector tuioObjectList. Mientras el ID del objeto tuyo no sea NULL o no se mayor a 11 (el número por defecto de muestras con las que estamos trabajando), se empieza a dibujar en la interfaz gráfica, sino paramos.

Como se observa en el código, primero dibujamos un rectángulo.

```
143    // Basic Rect
144    pushMatrix();
145    stroke(0);
146    fill(randomRGBColor());
147    translate(tobj.getScreenX(width),tobj.getScreenY(height));
148    rotate(tobj.getAngle());
149    rectMode(CORNER);
150    rect(-obj_size/2,-obj_size/2,obj_size,obj_size);
151    popMatrix();
```

En processing para dibujar un objeto dinámico, hay que hacer uso de las funciones translate, pushMatrix y popMatrix. Para mover el origen de lugar es necesario primero poner pushMatrix, después translate con las nuevas coordenadas y popMatrix cuando se desee regresar el origen a su lugar.

- pushMatrix indica que queremos cambiar el origen de las coordenadas, ya que la figura se va a mover.
- translate indica la nueva ubicación del origen.

- popMatrix se utiliza cuando queremos que el origen regrese a su lugar habitual.

El proceso para dibujar el contorno del rectángulo haciendo uso de RotateShape es similar al dibujo del rectángulo, así como el dibujo de la línea que une cada figura con el punto central de la pantalla.

Una vez dibujado los objetos, se incluirá la siguiente información de cada objeto:

- ID: Identificador del objeto.
- LocationX: Posición en el eje X.
- LocationY: Posición en el eje Y.
- Angle: Ángulo del objeto.

En el siguiente código se encuentra la implementación de esta información.

```
199    text("ID: "+tobj.getSymbolID(), tobj.getScreenX(width)+60,
200         tobj.getScreenY(height)-20);
201
202    text("LocationX: "+tobj.getScreenX(width), tobj.getScreenX(width)+60,
203         tobj.getScreenY(height)-10);
204
205    text("LocationY: "+tobj.getScreenY(height), tobj.getScreenX(width)+60,
206         tobj.getScreenY(height));
207
208    text("Angle: "+tobj.getAngle()*360/6.28, tobj.getScreenX(width)+60,
209         tobj.getScreenY(height)+10);
```

Una vez se han detallado los fragmentos de código más importantes, vemos en las siguientes imágenes el correcto funcionamiento del motor gráfico.

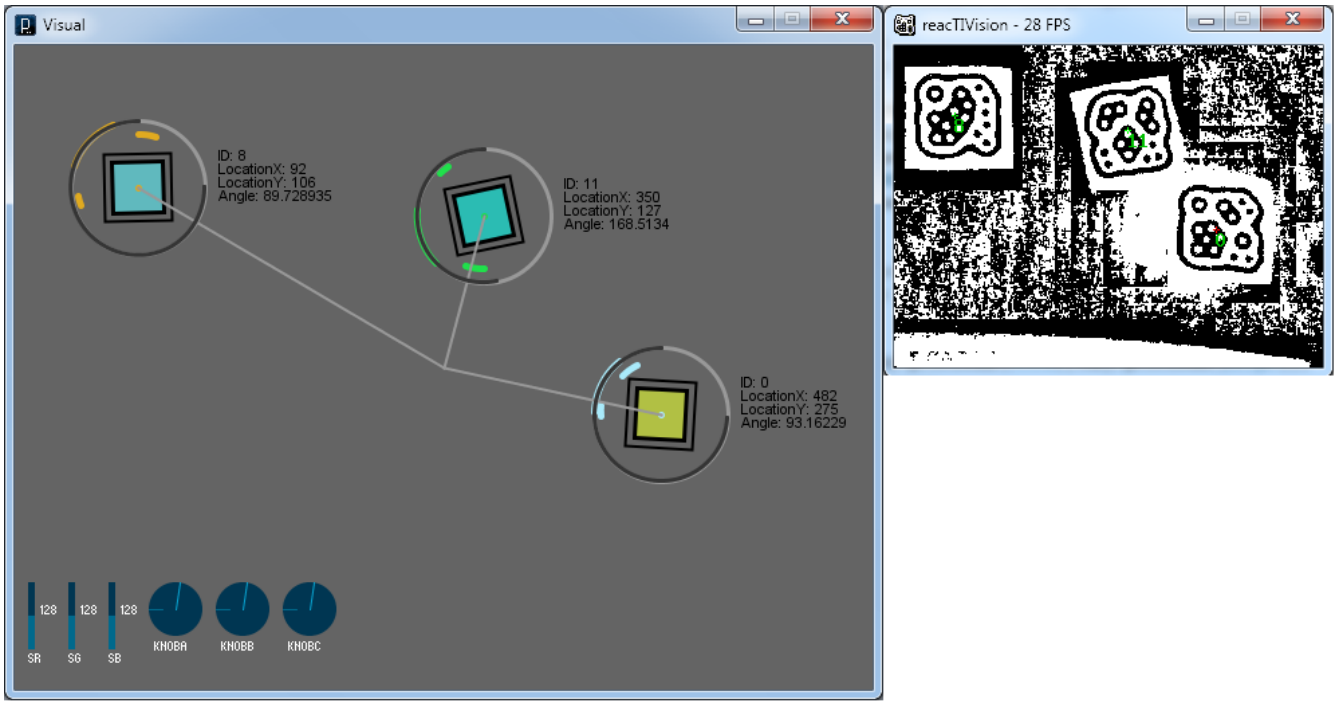


Figura 4.3: Reconocimiento de tres fiduciales en reacTIVision.

4.1.4. Gráficas en Pure Data

A través del protocolo OSC es posible obtener los valores de una onda sonora desde Pure Data, permitiento integrar en nuestro proyecto de Processing los módulos de programación de sonido generados en Pure Data.

Para que processing pueda comunicarse con Pure Data, se debe crear un módulo OSC que enviará paquetes de datos desde Pure Data hasta processing.

Tenemos que indicarle a Pure Data que se conecte al puerto 12000 que es el que escucha por defecto la librería OscP5 de Processig. Para esto incluimos el mensaje çonnectçon los parámetros 127.0.0.1 y 12000. Este mensaje es enviado al objeto üdpsend.ª través de su inlet.

Es imprescindible utilizar el object “packOSC” ya que es el encargado de convertir los mensajes de Pure Data al formato binario de OSC como se puede ver en la figura 4.4.

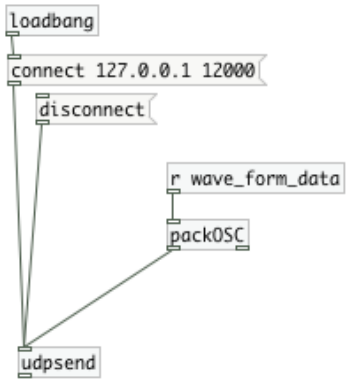


Figura 4.4: Módulo de envío de comandos OSC hacia Proccesing.

Para poder ver el sonido que genera cada fiducial, es necesario tener un módulo de envío de valores gráficos por cada figura. En la figura 4.5 se observa el módulo para el fiducial con id 0 y el fiducial con id 1.

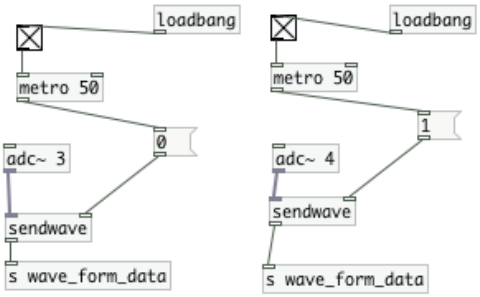


Figura 4.5: Módulo de envío de valores gráficos del fiducial 1 y fiducial 2.

Como se observa en la figura 4.5 cada fiducial tiene asociado un objeto para convertir el audio analógico en digital (adc) este objeto se usa para indicar cual es el canal de entrada de audio, en este caso el fiducial con id 0 tiene la entrada de línea 3 y el fiducial con id 1 tiene la entrada de línea 4, esto es así porque se han reservado los canales 1 y 2 para el master del secuenciador Ableton Live!.

Cada una de estas entradas de audio se analizarán con la función "sendwave", creada para dibujar la onda de cada objeto. Como se observa en la figura 4.6 enviaremos hacia processing 300 puntos de la onda sonara cada 25 milisegundos.

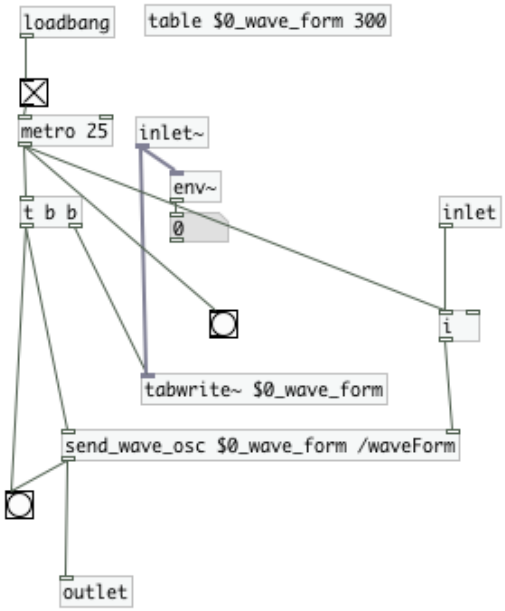


Figura 4.6: Función sendwave.

Una vez tenemos todos los módulos funcionando en Pure Data, Processing tiene que interpretar estos mensajes. Para ello tendremos que crearnos una función que recoja los valores obtenidos en Pure Data. Esta función se ha llamado "getWaveForms¨ y aceptará los valores obtenidos en la tabla "waveForm".

Configuramos las variables globales que tratará la función "getWaveFormsz configuramos los mensajes OSC para indicar que escuchará por el puerto 12000 UDP.

```
182 int numValues = 301;
183 float[] temp_val = new float[numValues];
184 int v_wave;
185 int id_wave;
186 float[][] yvalues = new float[numValues][numValues];
187 int amplitude = 60;
```

```
182   OscProperties properties = new OscProperties();
183   properties.setRemoteAddress("127.0.0.1",12000);
184   properties.setListeningPort(12000);
185   properties.setSRSP(OscProperties.ON);
186   properties.setDatagramSize(4096);
187
188   oscP5_pd = new OscP5(this,properties);
189   oscP5_pd.plug(this, "getWaveForms", "/waveForm");
```

La función getWaveForms analizará cada 25 milisegundos 300 valores de onda y los almacenará en la matriz yvalues[][].

```
1004 public void getWaveForms(float[] theValues) {
1005
1006   int id = int ( theValues[0] ) ;
1007   println("Valor:"+theValues.length);
1008   for (int i=1;i<theValues.length  ;i++) {
1009
1010     // tmp tiene todos los valores de onda
1011     float tmp = (theValues[i] * 10) - 2;
1012     yvalues[id][i-1] =  tmp *amplitude;
1013   }
1014 }
```

4.1.5. Llamadas a TUIO

A continuación se exponen una lista de funciones que sirven para mostrarnos por pantalla información cuando ocurre un evento TUIO para poder interpretar que todo funciona correctamente.

```
254   void addTuioObject(TuioObject tobj) {
255     println("add object "+tobj.getSymbolID()
256             +" ("+tobj.getSessionID()+") "
257             +tobj.getX()+" "
258             +tobj.getY()+" "
259             +tobj.getAngle());
260 }
```

Esta llamada la invoca TuioClient cuando un TuioObject nuevo se añade a la sesión actual. Se mostrará en pantalla que se ha añadido un objecto nuevo así como información relativa al objecto.

A continuación se muestra el código de la llamada a removeTuioObject que realiza lo mismo que addTuioObject pero cuando eliminamos un objeto de la sesión.

```
262   void removeTuioObject(TuioObject tobj) {
263     println("remove object "+tobj.getSymbolID()
264            +" ("+tobj.getSessionID()+")");
265 }
```

Con la función updateTuioObject actualizamos los valores del objeto que se está manipulando, incluyendo su posición en la pantalla, angulo, velocidad de rotación, etc.

```
272   void updateTuioObject (TuioObject tobj) {
273     println("update object "+tobj.getSymbolID()
274             +" ("+tobj.getSessionID()+") "
275                       +tobj.getX()+" "
276                       +tobj.getY()+" "
277                       +tobj.getAngle()
278             +" "+tobj.getMotionSpeed()+" "
279                       +tobj.getRotationSpeed()+" "
280                       +tobj.getMotionAccel()+" "
281                       +tobj.getRotationAccel());
282 }
```

Si añadimos un cursos a la escena, lo actualizamos o lo eliminamos, las funciones que muestran esta información son similares a las de un objeto TUIO pero esta vez tratando el cursor.

```
284   void addTuioCursor(TuioCursor tcur) {
285     println("add cursor "+tcur.getCursorID()
286             +" ("+tcur.getSessionID()+ ") "
287                       +tcur.getX()+" "
288                       +tcur.getY());
289 }
290
291
292   void updateTuioCursor (TuioCursor tcur) {
293     println("update cursor "+tcur.getCursorID()
294             +" ("+tcur.getSessionID()+ ") "
295                       +tcur.getX()+" "+tcur.getY()
296             +" "+tcur.getMotionSpeed()+" "
297                       +tcur.getMotionAccel());
298 }
299
300
301   void removeTuioCursor(TuioCursor tcur) {
302     println("remove cursor "+tcur.getCursorID()
303             +" ("+tcur.getSessionID()+")");
304 }
```

En la siguiente imágen se pueden observar los datos de salida de pantalla cuando manipulamos un objeto, ya sea añadiendolo a la escena, eliminando o actualizando su información.

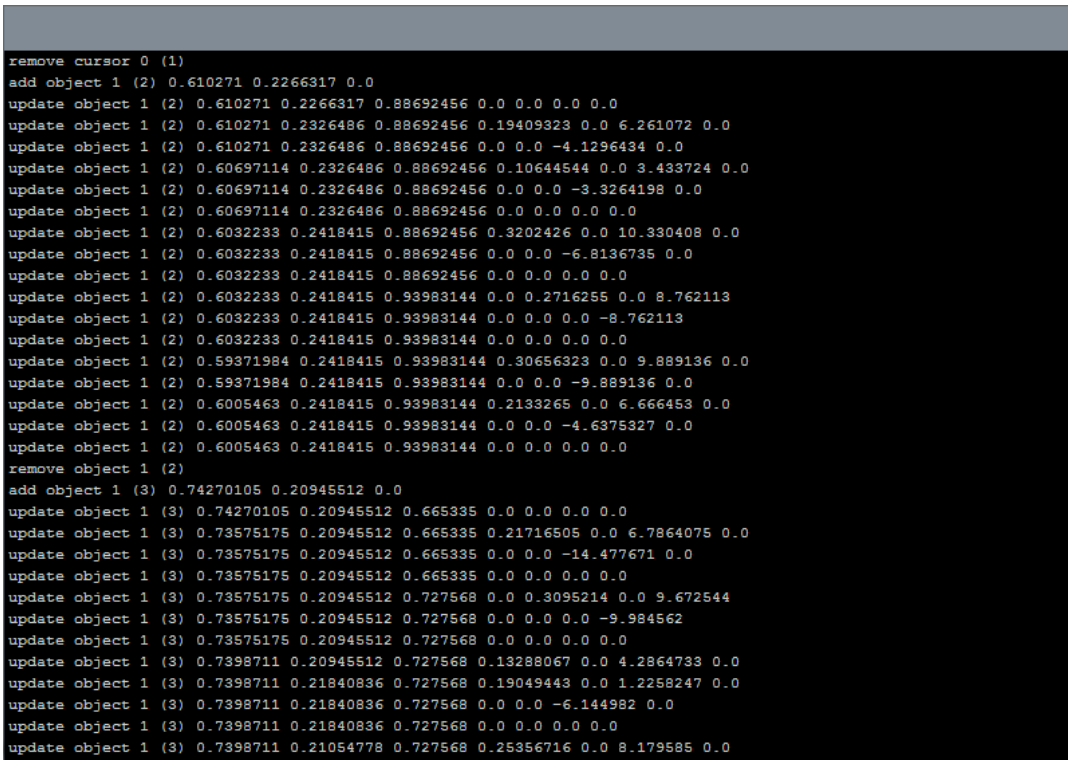


Figura 4.7: Salida de datos en pantalla.

4.2. Módulo de producción de mensajes OSC

Para la implementación de este módulo se ha tenido que modificar una API para comunicar el secuenciador de audio/midi Ableton Live! con el processing, para ello se ha utilizado una API programada en python que nos permite crear un puerto OSC en Ableton Live! comunicándose de esta manera el processing con el secuenciador, transmitiendo paquetes OSC por el puerto UDP 9001 y aceptando paquetes OSC por el puerto UDP 9000.

A continuación profundizaremos en el estudio de los elementos más importantes de esta API y la manera correcta de transmitir y recibir mensajes OSC en processing.

4.2.1. LiveOSC.py

Hemos de obtener todas las propiedades que afectan a un track como son el volumen, mute, envíos, panning, etc y habrá que diferenciar entre track y escena.

A continuación se explican las funciones más relevantes que se han definido para los métodos standard

que Ableton Live! puede realizar.

```
78 def connect_script_instances(self, instanciated_scripts):
79         """
80         """
81         return
```

Esta función es llamada por la aplicación tan pronto como todos los scripts se inicializan. Y permitirá conectar los módulos de extensión necesarios.

```
89 def request_rebuild_midi_map(self):
90         return
```

Esta función se ejecuta cada 100 ms, por lo que la usamos para iniciar nuestra canción (current_song_time) para que podamos procesar comandos entrantes OSC tan rápido como sea posible.

Por defecto se fija basicAPI a 0, así podremos asignarla después de la inicialización. Tratamos de conseguir la canción actual y si podemos vamos a conectar las llamadas a basicAPI por parte de los oyentes, lo que nos permite responder a las entrantes OSC cada 60ms.

```
100 def update_display(self):
101         ######################################################
102         # START OSC LISTENER SETUP
103 
104         if self.basicAPI == 0:
105             try:
106                 doc = self.song()
107             except:
108                 return
109             try:
110                 self.basicAPI = LiveOSCCallbacks.LiveOSCCallbacks(self.
                       _LiveOSC__c_instance, self.oscServer)
111                 # Commented for stability
112                 #doc.add_current_song_time_listener(self.oscServer.
                       processIncomingUDP)
113                 self.oscServer.sendOSC('/remix/echo', 'basicAPI setup
                       complete')
114             except:
115                 return
116         if self.oscServer:
117             try:
118                 self.oscServer.processIncomingUDP()
119             except:
120                 pass
```

```
121
122         # END OSC LISTENER SETUP
123         ########################################################
```

Se utiliza esta función para para enviar eventos MIDI a traves de Ableton Live! a los dispositivos MIDI reales que tengamos conectados, en nuestro caso, processing utilizará la librería MidiBus para interactuar con esta función.

```
144 def send_midi(self, midi_event_bytes):
145         pass
146     def receive_midi(self, midi_bytes):
147         return
```

El cambio de un track a otro viene implementado en la función track_change donde basicamente se incrementa el número de track de acuerdo al comando OSC enviado, que en este caso será /live/-track(index).

```
267 def track_change(self):
268         selected_track = self.song().view.selected_track
269         tracks = self.song().visible_tracks
270         index = 0
271         selected_index = 0
272         for track in tracks:
273             index = index + 1
274             if track == selected_track:
275                 selected_index = index
276 
277         if selected_index != self.track:
278             self.track = selected_index
279             self.oscServer.sendOSC("/live/track", (selected_index))
```

En la función slot_changestate, no se estaban recibiendo mensajes OSC cuando se han eliminado clips o cuando estos son trasladados a otras ranuras, esto ocurre porque la sentencia if-else no envía ninguna notificación del clip eliminado. Por tanto en esta función se ha hecho una modificación para enviar simplemente esto a otra sentencia else.

```
493 def slot_changestate(self, slot, tid, cid):
494         tmptrack = LiveUtils.getTrack(tid)
495         armed = tmptrack.arm and 1 or 0
496 
497         # Added new clip
498         if slot.clip != None:
499             self.add_cliplistener(slot.clip, tid, cid)
```

```
500
501             playing = 1
502             if slot.clip.is_playing == 1:
503                 playing = 2
504
505             if slot.clip.is_triggered == 1:
506                 playing = 3
507
508             length =  slot.clip.loop_end - slot.clip.loop_start
509
510             self.oscServer.sendOSC('/live/track/info', (tid, armed, cid,
                    playing, length))
511             self.oscServer.sendOSC('/live/name/clip', (tid, cid, str(slot.
                    clip.name), slot.clip.color))
512         else:
513             if self.clisten.has_key(slot.clip) == 1:
514                 slot.clip.remove_playing_status_listener(self.clisten[slot.
                        clip])
515
516             if self.pplisten.has_key(slot.clip) == 1:
517                 slot.clip.remove_playing_position_listener(self.pplisten[
                        slot.clip])
518
519             if self.cnlisten.has_key(slot.clip) == 1:
520                 slot.clip.remove_name_listener(self.cnlisten[slot.clip])
521
522             if self.cclisten.has_key(slot.clip) == 1:
523                 slot.clip.remove_color_listener(self.cclisten[slot.clip])
524
525             self.oscServer.sendOSC('/live/track/info', (tid, armed, cid, 0,
                    0.0))
526             self.oscServer.sendOSC('/live/clip/info', (tid, cid, 0))
```

4.2.2. LiveOSCCallbacks.py

Este archivo contiene todas las posibles llamadas OSC que processing puede hacer a Ableton Live!. Contiene la clase LiveOSCCallbacks así como el comportamiento interno de cada una de estas llamadas.

```
29  class LiveOSCCallbacks:
30      def __init__(self, c_instance, oscServer):
31          if oscServer:
32              self.oscServer = oscServer
33              self.callbackManager = oscServer.callbackManager
```

```
34              self.oscClient = oscServer.oscClient
35
36              self.c_instance = c_instance
37          else:
38              return
39
40          self.callbackManager.add(self.tempoCB, "/live/tempo")
41          self.callbackManager.add(self.timeCB, "/live/time")
42          self.callbackManager.add(self.nextCueCB, "/live/next/cue")
43          self.callbackManager.add(self.prevCueCB, "/live/prev/cue")
44          self.callbackManager.add(self.playCB, "/live/play")
45          self.callbackManager.add(self.playContinueCB, "/live/play/continue")
46          self.callbackManager.add(self.playSelectionCB, "/live/play/selection
               ")
47          self.callbackManager.add(self.playClipCB, "/live/play/clip")
48          self.callbackManager.add(self.playSceneCB, "/live/play/scene")
49          self.callbackManager.add(self.stopCB, "/live/stop")
50          self.callbackManager.add(self.stopClipCB, "/live/stop/clip")
51          self.callbackManager.add(self.stopTrackCB, "/live/stop/track")
52          .
53          .
54          .
55          .
```

En el apéndice B tenemos una lista de todas las posibles llamadas que se pueden realizar. A continuación se define la estructura interna de cada llamada, para ello se crean las funciones llamadas por self.

```
133 def tempoCB(self, msg):
134         if len(msg) == 2 or (len(msg) == 3 and msg[2] == "query"):
135             self.oscServer.sendOSC("/live/tempo", LiveUtils.getTempo())
136
137         elif len(msg) == 3:
138             tempo = msg[2]
139             LiveUtils.setTempo(tempo)
```

En el fragmento de código anterior definimos la función tempoCB que es llamada cuando se recibe un mensaje del tipo /live/tempo. Como se indica en el apéndice, este mensaje podría ser de dos tipo:

- /live/tempo: Solicitud del tempo actual, donde se nos responderá con un valor float
- /live/tempo (float tempo): Fijamos el tempo al valor float solicitado

Para ver otro ejemplo del funcionamiento interno de cada función, se describirá también la función stopClipCB, la cual es llamada cuando se recibe un mensaje del tipo /live/stop/clip(int track, int clip).

```
294 def stopClipCB(self, msg):
295         if len(msg) == 4:
296             track = msg[2]
297             clip = msg[3]
298             LiveUtils.stopClip(track, clip)
```

4.2.3. socket_live_8.py

Este módulo proporciona las operaciones relacionada con los sockets. En Unix, es compatible con el protocolo IP y los sockets de Unix En otros sistemas, sólo es compatible con IP. Las funciones específicas para un socket están disponibles como métodos del objeto socket.

Las funciones son las siguientes:

- socket(): Crea un objeto socket nuevo.
- socketpair(): Crea una pareja de objetos sockets nuevos.
- fromfd(): Crea un objeto socket desde un descriptor de fichero abierto.
- gethostname(): Devuelve el hostname actual.
- gethostbyname(): Asigna un hostname a su número de IP.
- gethostbyaddr(): Asigna un número IP o hostname a la información del DNS.
- getservbyname(): Asigna un nombre de servicio o nombre de protocolo a un número de puerto.
- getprotobyname(): Asigna un nombre de protocolo (por ejemplo TCP) a un número.
- inet_aton(): Convierte una dirección IP (por ejemplo 123.45.67.89) en un paquete formado por 32 bits.
- inet_ntoa(): Convierte un paquete formado por 32 bits en una dirección IP.
- ssl(): Soporte del protocolo SSL.
- socket.getdefaulttimeout(): Obtiene el valor de tiempo de espera por defecto.
- socket.setdefaulttimeout(): Fija el valor de tiempo de espera por defecto.

A continuación vemos la programación de la clase socketobject.

```
151 class _socketobject(object):
152
153     __doc__ = _realsocket.__doc__
154
155     __slots__ = ["_sock", "__weakref__"] + list(_delegate_methods)
156
157     def __init__(self, family=AF_INET, type=SOCK_STREAM, proto=0, _sock=None
        ):
158         if _sock is None:
159             _sock = _realsocket(family, type, proto)
160         self._sock = _sock
161         for method in _delegate_methods:
162             setattr(self, method, getattr(_sock, method))
163
164     def close(self):
165         self._sock = _closedsocket()
166         dummy = self._sock._dummy
167         for method in _delegate_methods:
168             setattr(self, method, dummy)
169     close.__doc__ = _realsocket.close.__doc__
170
171     def accept(self):
172         sock, addr = self._sock.accept()
173         return _socketobject(_sock=sock), addr
174     accept.__doc__ = _realsocket.accept.__doc__
175
176     def dup(self):
177         return _socketobject(_sock=self._sock)
178
179     def makefile(self, mode='r', bufsize=-1):
180         """makefile([mode[, bufsize]]) -> file object
181
182         Return a regular file object corresponding to the socket.  The mode
183         and bufsize arguments are as for the built-in open() function."""
184         return _fileobject(self._sock, mode, bufsize)
185
186     family = property(lambda self: self._sock.family, doc="the socket family
        ")
187     type = property(lambda self: self._sock.type, doc="the socket type")
188     proto = property(lambda self: self._sock.proto, doc="the socket protocol
        ")
189
190     _s = ("def %s(self, *args): return self._sock.%s(*args)\n\n"
```

```
191             "%s.__doc__ = _realsocket.%s.__doc__\n")
192     for _m in _socketmethods:
193         exec _s % (_m, _m, _m, _m)
194     del _m, _s
195
196 socket = SocketType = _socketobject
```

4.2.4. struct.py

En este archivo encontramos las funciones para convertir entre valores de Python y estructuras C. Las cadenas de Python se utilizan para mantener los datos que representan las estructuras C y también como cadenas de formato para describir el diseño de los datos en las estructuras C.

El formato del primer carácter opcional indica el orden de bytes, tamaño y alineación. Los caracteres restantes indican los tipos de argumentos y debe coincidir exactamente; estos pueden ser precedidos por un número de repeticiones decimales:

- x: bytes almohadilla (sin datos).
- c: carácter.
- b: bytes con signo.
- B: bytes sin signo.
- h: short.
- H: short sin signo.
- i: entero.
- I: entero sin signo.
- l: long.
- L: long sin signo.
- float: punto flotante.

Casos especiales:

- s: string (array de caracteres).

- p: string en formato pascal.

Casos especiales (sólo disponibles en formato nativo):

- P: Un tipo entero que es lo suficientemente amplio como para contener un puntero.

```
32 _MAXCACHE = 100
33 _cache = {}
34
35 def _compile(fmt):
36     # Internal: compile struct pattern
37     if len(_cache) >= _MAXCACHE:
38         _cache.clear()
39     s = Struct(fmt)
40     _cache[fmt] = s
41     return s
42
43 def calcsize(fmt):
44     """
45     Return size of C struct described by format string fmt.
46     See struct.__doc__ for more on format strings.
47     """
48     try:
49         o = _cache[fmt]
50     except KeyError:
51         o = _compile(fmt)
52     return o.size
53
54 def pack(fmt, *args):
55     """
56     Return string containing values v1, v2, ... packed according to fmt.
57     See struct.__doc__ for more on format strings.
58     """
59     try:
60         o = _cache[fmt]
61     except KeyError:
62         o = _compile(fmt)
63     return o.pack(*args)
64
65 def pack_into(fmt, buf, offset, *args):
66     """
67     Pack the values v1, v2, ... according to fmt, write
68     the packed bytes into the writable buffer buf starting at offset.
69     See struct.__doc__ for more on format strings.
```

```
70      """
71      try:
72          o = _cache[fmt]
73      except KeyError:
74          o = _compile(fmt)
75      return o.pack_into(buf, offset, *args)
76
77  def unpack(fmt, s):
78      """
79      Unpack the string, containing packed C structure data, according
80      to fmt.  Requires len(string)==calcsize(fmt).
81      See struct.__doc__ for more on format strings.
82      """
83      try:
84          o = _cache[fmt]
85      except KeyError:
86          o = _compile(fmt)
87      return o.unpack(s)
88
89  def unpack_from(fmt, buf, offset=0):
90      """
91      Unpack the buffer, containing packed C structure data, according to
92      fmt starting at offset. Requires len(buffer[offset:]) >= calcsize(fmt).
93      See struct.__doc__ for more on format strings.
94      """
95      try:
96          o = _cache[fmt]
97      except KeyError:
98          o = _compile(fmt)
99      return o.unpack_from(buf, offset)
```

4.2.5. Processing usando comandos OSC

Para asociar una orden OSC a un objeto fiducial, debemos indicar la ID de la figura correspondiente, así como el mensaje OSC que queremos enviarle a Ableton Live!, en el siguiente código al objeto número uno se le encarga la orden de que pare el track que se encuentra en la pista número catorce.

```
411 if (tobj.getSymbolID()== 1){
412   OscMessage myMessage2 = new OscMessage("/live/stop/track");
413   myMessage2.add(14);
414   oscP5.send(myMessage2, myRemoteLocation);
415   }
```

Para aquellas órdenes OSC que requieran dos parámetros, debemos añadir un campo extra del tipo ‘add’. Por ejemplo si lo que queremos es reproducir el clip número uno de la pista número once, se hace con el siguiente código:

```
334 if (tobj.getSymbolID()== 8){
335     OscMessage myMessage2 = new OscMessage("/live/play/clip");
336     myMessage2.add(11);
337     myMessage2.add(1);
338     oscP5.send(myMessage2, myRemoteLocation);
339     }
```

Como se puede observar en la siguiente 4.8, en las opciones del secuenciador Ableton Live! nos aparecerá un puerto de salida MIDI que se encargará de recibir y capturar las ordenes OSC de Processing.

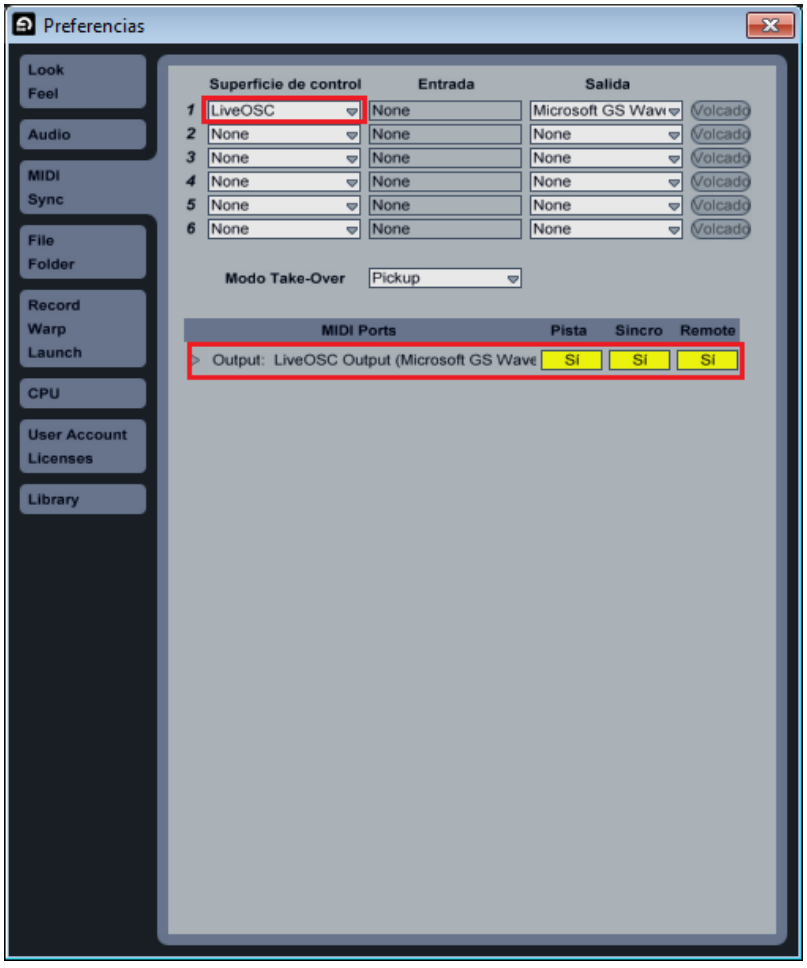


Figura 4.8: Captura del puerto de salida MIDI LiveOSC.

4.3. Módulo de interacción multitouch

Como se comentó en el análisis, el protocolo TUIO también permite el desarrollo rápido de una mesa multitáctil. TUIO ha sido principalmente diseñado como una abstracción para superficies interactivas, pero también se ha utilizado en muchas otras áreas de aplicación relacionadas.

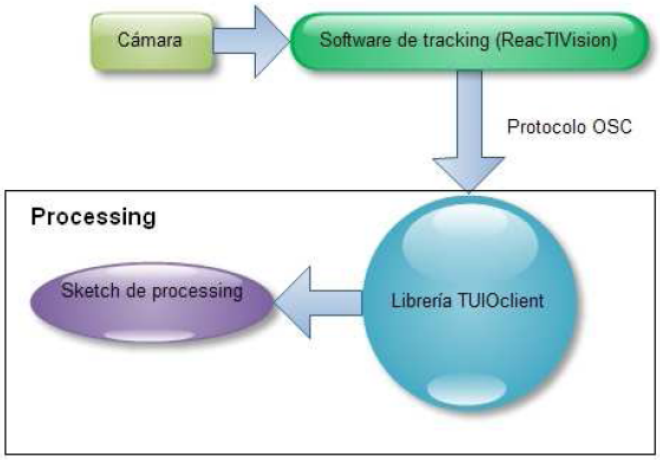


Figura 4.9: Esquema de funcionamiento de la librería.

En este apartado se explicará el desarrollo de la interacción multitouch con el framework ReacTIVision, viendo los fragmentos de código más relevantes.

Hay que tener en cuenta que ReacTIVision por defecto no tiene activado el reconocimiento de los dedos del usuario, así que para ello hay que modificar el archivo de configuración XML que aporta este framework.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<reactivision>
    <!--
these are all the possible configuration tags and their default values
you only need to specify parameters that you want to change from their
   default
please note that the active tags need to be outside of this commented area.

<tuio host="127.0.0.1" port="3333" />
<midi config="midi_demo.xml" />
<camera config="camera.xml" />
<finger size="0" sensitivity="100" />
<image display="dest" equalize="false" gradient="32" />
<fiducial engine="amoeba" tree="default"/>
```

```
<calibration file="default.grid" invert="xa" />

finger tracking is turned OFF by default,
you can enable it by defining the average finger
blob size in pixels.
-->
    <finger size="25" sensitivity="75" />
    <image display="dest" equalize="false" gradient="32" />
    <calibration file="default.grid" invert=" " />
    <tuio host="127.0.0.1" port="3333" />
    <camera config="camera.xml" />
</reactivision>
```

Como se puede observar, se ha incluido un tamaño de 25 y una sensibilidad del 75 % para que el framework ReacTIVision pueda reconocer los dedos como un objeto más, sin embargo el tratamiento y las funciones a utilizar son distintas.

TUIO distingue dos tipos de elementos: objetos y cursores. Los objetos son los fiduciales, mientras que los cursores son los dedos que se apoyan sobre la pantalla (dado que el sistema también es capaz de reconocer el tacto).

Se han definido vectores globales que irán almacenando el número de eventos táctiles que aparecen o desaparecen en nuestra superficie

```
float []fingers_x=new float [tv];
float []fingers_y=new float [tv];
boolean []fingers_active=new boolean[tv];
boolean []modamp=new boolean[tv];
int []modamp_id=new int[tv];
int []modamp_i=new int[tv];
boolean []ismute=new boolean[tv];
```

Como se hizo con los objetos físicos, también se debe inicializar una lista de cursores, que serán los responsables de los eventos táctiles. Cada una de estas funciones, informan un evento de un patrón o de tacto.

Las tres funciones principales para el tratamiento multitouch son:

- void addTuioCursor(TuioCursor tcur): informa sobre la aparición de un dedo en la pantalla
- void updateTuioCursor(TuioCursor tcur): informa que un dedo salió de la pantalla
- void removeTuioCursor(TuioCursor tcur): informa sobre los cambio que sufre un dedo, un cambio de posición.

```
void updateTuioCursor() {
    Vector tuioCursorList = tuioClient.getTuioCursors();
    for (int i=0;i<tuioCursorList.size();i++) {
        TuioCursor tcur = (TuioCursor)tuioCursorList.elementAt(i);
        float vx = tcur.getXSpeed() * tuioCursorSpeedMult;
        float vy = tcur.getYSpeed() * tuioCursorSpeedMult;
        if(vx == 0 && vy == 0) {
            vx = random(-tuioStationaryForce, tuioStationaryForce);
            vy = random(-tuioStationaryForce, tuioStationaryForce);
        }
        addForce(tcur.getX(), tcur.getY(), vx, vy);
    }

    if(tuioDoubleTap) {
        drawFluid ^= true;
        tuioDoubleTap = false;
    }
}
```

Como se comentó en el diseño de la aplicación, se van a implementar dos gestos táctiles.

- Silenciar pistas
- Control de parámetros

Para silenciar una pista, debemos hacer un trazado de izquierda a derecha con el dedo sobre la línea que une un objeto con el punto central de la interfaz visual.

Para implementar dicha opción, añadimos en la función addTuioCursor un mensaje OSC para que el Ableton Live! lo interprete como el bloqueo de sonido de la pista seleccionada.

En el siguiente fragmento de código vemos las acciones que se tendrían que realizar para silenciar el sonido de un objeto

```
if (tobjx.getSymbolID()<=numFid)
        {
          if (muteado[tobjx.getSymbolID()]==false)
          {
            idtemp=idv[i];
            //println("ID del objeto :" +idtemp);
            OscMessage myMessage = new OscMessage("/live/mute");
            myMessage.add(idtemp);
```

```
2528              myMessage.add(0);
2529              println("I sent a UnMute Track Request for "+idtemp);
2530              oscP5.send(myMessage, myRemoteLocation);
2531            }
2532            else
2533            {
2534              idtemp=idv[i];
2535              //println("ID del objeto :" +idtemp);
2536              OscMessage myMessage = new OscMessage("/live/mute");
2537              myMessage.add(idtemp);
2538              myMessage.add(1);
2539              println("I sent a Mute Track Request for "+idtemp);
2540              oscP5.send(myMessage, myRemoteLocation);
2541            }
2542          }
2543  }
```

Para implementar las funcionalidades del control de la amplitud de un objeto físico, o el control de los parámetros de dry/wet de alguna onda específica, los objetos tendrá un parámetro que se puede cambiar tocando la pantalla en el lado derecho del objeto.

En el siguiente fragmento de código se puede observar que se ha definido otro cursor para el control de dichos parámetros, mandando el comando OSC para cambiar la amplitud de un objeto.

```
2335  amplitud[modamp_id[tcur.getCursorID()]]=(anguloatan4/PI);
2336
2337      if (amplitud[modamp_id[tcur.getCursorID()]]<0) {
2338        amplitud[modamp_id[tcur.getCursorID()]]=0;
2339      }
2340      if (amplitud[modamp_id[tcur.getCursorID()]]>0.95) {
2341        amplitud[modamp_id[tcur.getCursorID()]]=0.95;
2342      }
2343      if (modamp_id[tcur.getCursorID()]<=numFid)
2344      {
2345        OscMessage myMessage = new OscMessage("/live/volume");
2346        myMessage.add(modamp_id[tcur.getCursorID()]);
2347        myMessage.add(1-amplitud[modamp_id[tcur.getCursorID()]]);
2348        oscP5.send(myMessage, myRemoteLocation);
2349      }
```

Capítulo 5

Desarrollo del prototipo hardware

Se incluye dentro de este proyecto la construcción y puesta en marcha de un prototipo funcional hardware de superficie multitouch, utilizando la técnica de DSI y una retroproyección, utilizando el software de control implementado anteriormente.

5.1. Fundamentos

Multitud de dispositivos multitouch se basan en soluciones de tipo óptico para conseguir su propósito. La escalabilidad, el bajo coste y la facilidad de construcción son algunos de los motivos de su popularidad. Existen muchas técnicas de tipo "light sensing", como veremos a continuación, para conseguir detectar los toques del usuario sobre una mesa interactiva. Todos ellos consisten en un sensor óptico, normalmente una cámara, una fuente de luz infrarroja, y una forma de mostrar imágenes como una pantalla LCD o un proyector.

Antes de comentar la tecnología óptica utilizada en este proyecto, es necesaria una pequeña introducción hacia la fuente de emisión y la recepción del elemento más importante en estas tecnologías, la luz infrarroja.

5.1.1. Fuentes de luz infrarroja

La luz infrarroja (IR en adelante) es una porción del espectro lumínico que se ubica justamente antes de la luz visible por el ojo humano. Su rango de longitud de onda es mayor que la luz visible, y determina una pequeña banda conocida como NIR (Near InfraRed, Cerca del Infrarrojo) que típicamente se consideran entre 700 nm y 1000 nm (nanómetros). La mayor parte de los sensores de las cámaras digitales son excitables en este rango, y a menudo vienen armados con filtros que lo eliminan, para no interferir con

la luz visible. La simple sustitución de este filtro por otro que elimine la luz visible convierte cualquier cámara digital en una cámara exclusivamente de luz infrarroja.

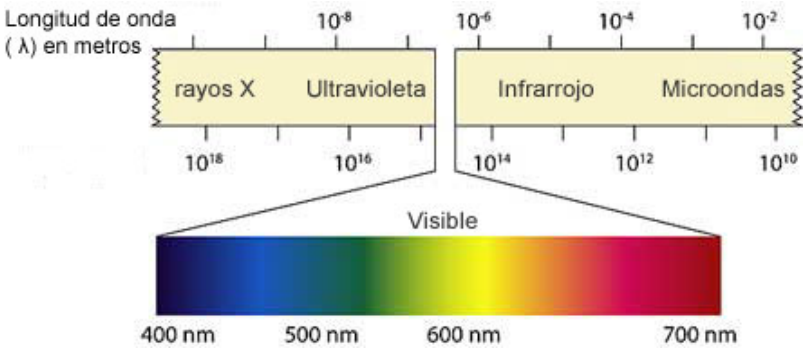


Figura 5.1: Espectro de la luz infrarroja.

En las superficies multitouch se utiliza este tipo de luz para poder separar lo concerniente a la imagen de la pantalla, con la que el usuario interactuará (luz visible), y aquella utilizada únicamente con fines de detección de toques. Aunque ambos haces de luz se mezclen en el mismo dispositivo, es relativamente sencillo trabajar con ellas independientemente utilizando únicamente la banda correspondiente a cada una.

El mayor problema de utilizar esta banda para la detección es evitar la polución lumínica infrarroja, que puede crear falsos positivos a la hora de la detección, y que no suele ser fácil de percibir a priori. Las fuentes de calor, el sol, las lámparas de tungsteno y algunos otros dispositivos producen gran cantidad de luz infrarroja, que debe ser valorada a la hora de construir y ubicar superficies interactivas de este tipo.

Se tendrán en cuenta tres parámetros que resultan de vital importancia para el correcto funcionamiento del sistema:

- Longitud de onda
- Intensidad de radiación
- Ángulo de emisión

5.1.2. Cámaras

Las cámaras generales de consumo están preparadas para captar luz en el espectro visible. Pero esto no es culpa del sensor (normalmente CCD) de la misma que, como se aprecia en la Ilustración 5.2, en realidad abarca muchas más frecuencias. Entre la lente de la cámara y el sensor se suele colocar un filtro que bloquea aquellas frecuencias que están fuera del espectro visible, para evitar artefactos en la imagen.

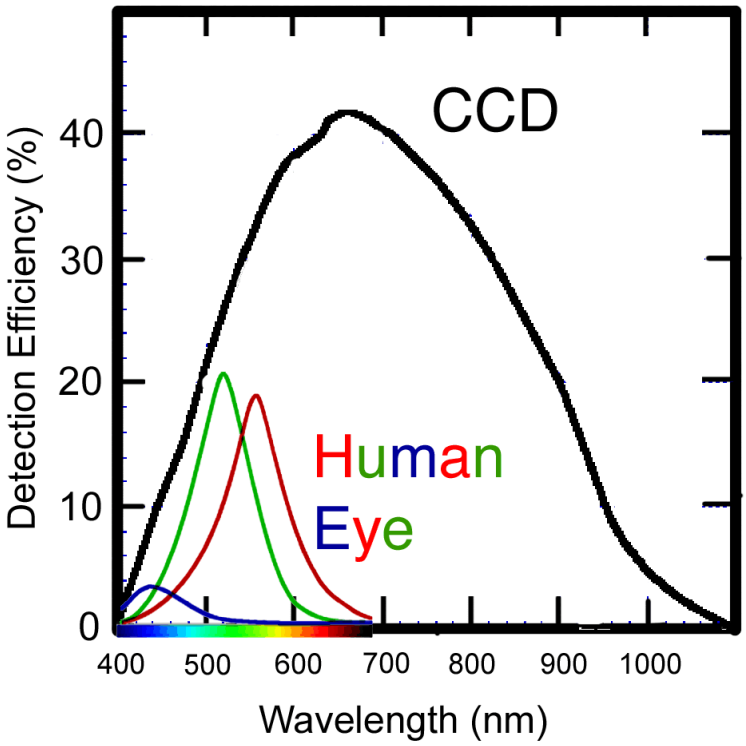


Figura 5.2: Comparación de las frecuencias captadas por el ojo humano y un sensor CCD.

Ya que el fenómeno DSI descrito trabaja con frecuencias en la banda de infrarrojos, por debajo del espectro, debe utilizarse una cámara especial equipada con un filtro que trabaje en dicha banda, o bien sustituir el que trae por defecto.

5.2. Diseño y construcción

5.2.1. Concepto

Inicialmente se desarrolló un boceto conceptual de la mesa. Ésta debía cumplir una serie de especificaciones en sus medidas para cuadrar todos los elementos técnicos que la componen, además de tener en cuenta cuestiones como la facilidad de acceso para su mantenimiento, ventilación, etc. Durante el desarrollo del proyecto se han desarrollado dos estructuras para el prototipo. La primera se elaboró con madera aglomerada como se ve en la figura , sin embargo esta estructura era pesada y difícil de manipular.

El segundo prototipo es más ligero, ya que se construyó con perfiles de aluminio permitiendo montarla y desmontarla rápidamente facilitando el transporte.

Figura 5.3: Prototipo 1.0 y Prototipo 2.0.

El segundo prototipo es más ligero, ya que se construyó con perfiles de aluminio permitiendo montarla y desmontarla rápidamente facilitando el transporte.

5.2.2. Campos visuales: Proyección y captura

Es importante tener en cuenta los campos de proyección y de visión del proyector y la cámara respectivamente. El campo de visión se define matemáticamente como una pirámide truncada; cuanto más distancia haya hasta la cúspide imaginaria, mayor será la base (ver Ilustración 5.4). El ángulo con el que se lanza la proyección (o ángulo de visión de la cámara) se denomina FOV (Field Of View), y se tiene en cuenta tanto el vertical como el horizontal.

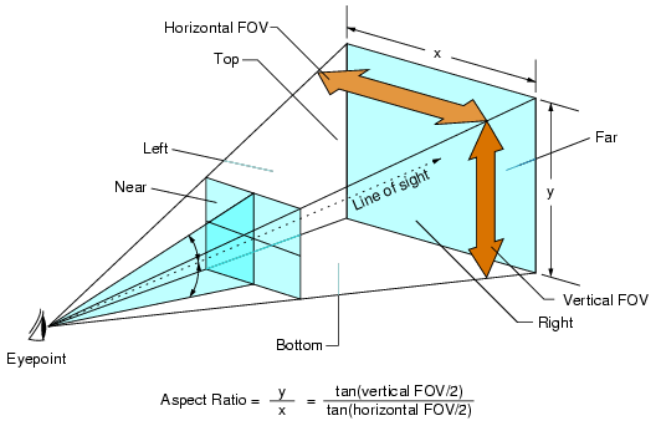


Figura 5.4: Medidas en la proyección.

En cuanto a la cámara, se escogió una cámara USB económica pero de gran potencia, ya que no proporcionaba ninguna deformación debido a la lente usada, aunque el proceso de calibración era más difícil. En la figura 5.5 se observa la proyección del sistema de calibración sobre la estructura usada en las primeras investigaciones con el prototipo funcional.

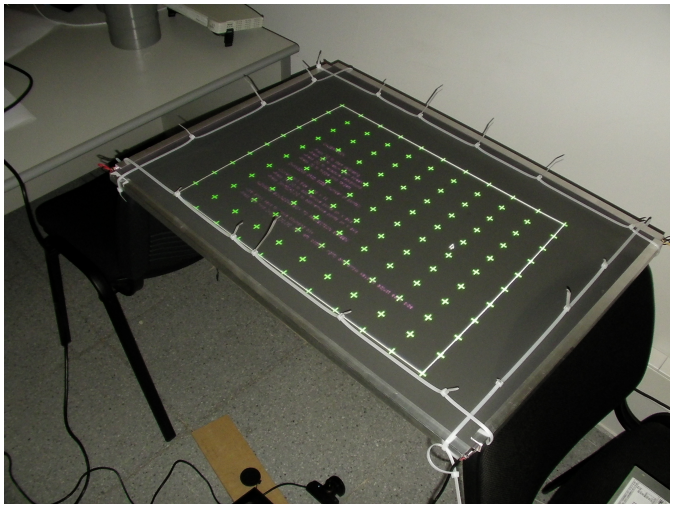


Figura 5.5: Calibración del prototipo inicial.

Respecto al proyector, al no disponer de uno de corta distancia se ha utilizado un espejo para recoger en la superficie de proyección una imagen de mayor tamaño a una distancia más corta.

En el prototipo se decidió hacer la inserción de luz infrarroja utilizando una tira de leds por su sencillez de manipulación, a pesar de ser algo más costoso económicamente de una red de LEDs ovales individuales.

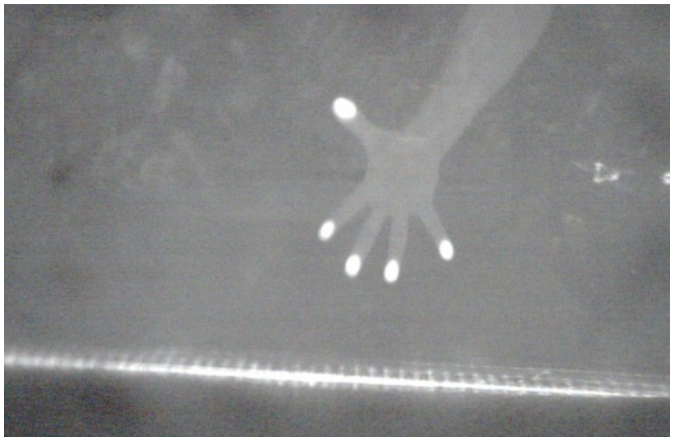


Figura 5.6: Detección de Blobs.

5.2.3. Superficie

La pantalla se plantea como un rectángulo de proporciones 4:3 y medidas 80 x 60 cm. Anteriormente se describió el principio de DSI, muy útil para construir superficies multitáctiles, pero no suficiente. Las mesas interactivas, además de la tecnología necesaria para captar los toques, necesitan una forma de mostrar imágenes, sobre las que el usuario interactuará.

La construcción del prototipo utiliza imágenes retroproyectadas sobre un acrílico. Pero el metacrilato normalmente es transparente, y aunque no lo fuera, no permitiría suficiente definición como para utilizarlo de superficie de proyección. Es necesario añadir una segunda capa a la superficie, que denominamos "Capa Difusora", por encima del acrílico. Esta capa puede ser de diversos materiales, y aunque existen soluciones comerciales óptimas para cumplir este cometido, también lo es que son excesivamente caros. Sin embargo, existen algunos materiales mucho más económicos que pueden cumplir la misma función sin perder prácticamente ninguna definición ni calidad de tacto.

Por último, y en caso de requerirlo, se añade una tercera capa encima de todas las anteriores, denominada "Capa Protectora" con una doble función: por una parte, protege las anteriores capas en el caso de que estas estén fabricadas con materiales frágiles o excesivamente sensibles. Por otra parte, será la que proporcione el tacto final a la superficie interactiva.

En resumidas cuentas, la superficie se compone de tres capas, contando el metacrilato, en el orden mostrado en la figura 5.7.

Figura 5.7: Capas de la superficie del prototipo.

Para su funcionamiento se utiliza el sistema software completo visto en los anteriores capítulos, utilizando Reactivision como servidor TUIO, Processing como motor gráfico y Ableton Live! y Pure Data como servidor de sonido que recibe las ordenes que genera el usuario final en la superficie.

Capítulo 6

Validación y Testeo

6.1. Objetivos de las pruebas

Con las pruebas que se van ejecutar en esta fase del proyecto, se pretende cumplir los siguientes objetivos, establecidos por Glen Myers

- La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Por otro lado, cada una de las pruebas seguirán los siguientes principios:

- Se llevará un seguimiento de cada prueba hasta los requisitos del cliente.
- Se planificarán mucho antes de su ejecución.
- No se realizarán pruebas exhaustivas, ya que el número de permutaciones de caminos para, incluso, un programa de tamaño moderado es excepcionalmente grande.

Otro principio que se debería tener en cuenta pero que no va a ser posible aplicarlo es que, para realizar las pruebas con mayor eficacia, deberán ser llevadas a cabo por un equipo/persona independiente al proyecto. El ingeniero del software que creó el sistema no es el más adecuado para llevar a cabo las pruebas.

6.2. Técnicas de prueba aplicadas

6.2.1. Pruebas de caja negra

Esta técnica se centra en los requisitos funcionales del software, es decir, permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra no es una alternativa a las técnicas de prueba de caja blanca. Más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los métodos de caja blanca.

Las pruebas de caja negra intentan encontrar errores de las siguientes categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Esta técnica se aplicará en el software mediante la partición equivalente de sus clases de datos ejecutando los métodos que las componen como casos de prueba independientes.

Por razones que no están del todo claras, los errores tienden a darse más en los límites del campo de entrada que en el «centro». Por ello, se utilizará el análisis de valores límites(AVL) como técnica de prueba. El análisis de valores límite nos lleva a una elección de casos de prueba que ejerciten los valores límite.

El análisis de valores limite es una técnica de diseño de casos de prueba que complementa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el AVL lleva a la elección de casos de prueba en los «extremos» de la clase. En lugar de centrarse solamente en las condiciones de entrada, el AVL obtiene casos de prueba también para el campo de salida.

6.2.2. Pruebas de caja blanca

Es un método de diseñoo de casos de prueba que usa la estructura de control del diseñoo procedimental para obtener los casos de prueba.

Mediante los métodos de prueba de caja blanca se pueden obtener casos de prueba que:

- Garantizan que se ejercita por lo menos una vez todos los caminos independientes de cada módulo.
- Ejercitan todas las decisiones lógicas en sus vertientes verdadera y falsa.
- Ejecutan todos los bucles en sus límites y con sus límites operacionales.
- Ejercitan las estructuras internas de datos para asegurar su validez.

Esta técnica se utilizará en el software desarrollado sólo en el caso de que surjan errores en las pruebas de caja negra, puesto que tanto su aplicación como diseño son demasiado complejas como para aplicarlas a todos los métodos de todas las clases desarrolladas.

6.2.3. Pruebas de la aplicación

Para llevar a cabo esta técnica sobre el software creado, se planificarán dos pruebas independientes. En la primera se realizará una mesa multitouch virtual en la máquina local en que se está desarrollando el proyecto, probando todas y cada una de las funcionalidades disponibles, intentando encontrar el máximo número de errores posibles.

Esta simulación además constituirá una prueba sobre la interfaz gráfica, la documentación y facilidades de ayuda desarrolladas, y sobre el diseño que se ha adoptado.

Por otro lado se ha intentado probar el dispositivo con personas ajenas al proyecto, así mediante la observación de los movimiento y gestos que realizan se pueden encontrar errores en la aplicación,

6.2.3.1. Prueba realizada por los alumnos de Diseño Gráfico por Computador

Durante el mes de Mayo de 2012, se desarrollaron una serie de demostraciones en el Laboratorio donde se encuentraba la mesa. Gracias a estas pruebas se pudo comprobar la capacidad funcional colaborativa de nuestro dispositivo Multitouch.

A lo largo de la prueba, se pudo comprobar que habían ciertos aspectos en el código que producían incompatibilidades en la funcionalidad colaborativa, ya que si dos o más personas quitaban un objeto de la mesa, a la vez que otra persona realizaba eventos táctiles, los vectores sufrían unas modificaciones no deseadas.

Figura 6.1: Demostración del funcionamiento del dispositivo a los alumnos.

6.2.3.2. Fiesta Freedom Island Tenerife

Durante el mes de Julio de 2012 tuvimos la oportunidad de exponer nuestro proyecto en una fiesta privada organizada por una conocida marca de tabaco.

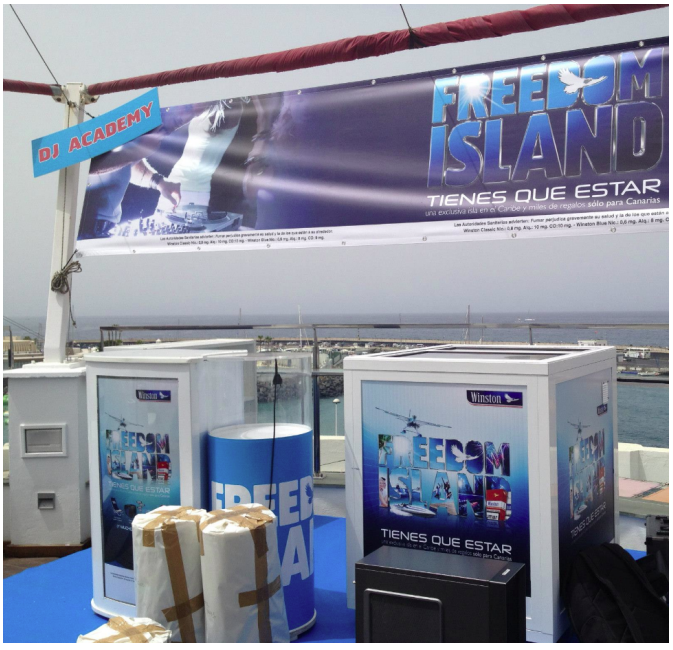


Figura 6.2: Escenario en la fiesta Freedom Island Tenerife.

Este escenario hizo que se pudiera comprobar el funcionamiento de la mesa en un evento real donde varias personas tenían la posibilidad de tocar nuestro dispositivo y así sacar conclusiones mediante la observación y las preguntas realizadas.

Figura 6.3: Fiesta Freedom Island Tenerife.

Gracias a esta fiesta comprobamos que el funcionamiento del dispositivo está limitado a ser utilizado sin la incidencia de la luz solar de manera directa, lo que nos hizo darnos cuenta que para otros eventos esta condición es imprescindible para un correcto funcionamiento.

6.2.3.3. Fiesta Freedom Island Gran Canaria

Dado el éxito que tuvo la fiesta Freedom Island Tenerife, durante el mes de agosto de 2012 se realizó en el Hotel Gloria Palace una fiesta similar a la de Tenerife.

Figura 6.4: Fiesta Freedom Island Gran Canaria.

Ha sido especialmente en esta fiesta donde nuestro dispositivo estuvo funcionando durante más de 5 horas seguidas sin detectar ningún error funcional. Aunque nos hemos dado cuenta que le estructura aún no está diseñada especialmente para ser totalmente transportable, que es algo que se irá corrigiendo con el paso del tiempo y algo de inversión.

Figura 6.5: Demostración en Freedom Island Gran Canaria.

6.2.4. Falsos Positivos y Falsos Negativos

Uno de los problemas típicos durante el manejo de una superficie multitouch es la aparición de falsos positivos y de faltos negativos. Un falso positivo aparece cuando un brillo no provocado por el usuario es reconocido por el software como un toque. Un falso negativo aparece cuando se pierde un toque realizado por el usuario; es decir, cuando un brillo o una zona oscura impide que se reconozca el toque.

Realizamos una rutina consistente en recorrer diez veces primero con uno y luego con dos dedos una diagonal de la mesa y comprobar la cantidad de falsos positivos y de falsos negativos que son detectados por el programa de seguimiento. Con ello dibujamos mediante las siguientes cuatro gráficas:

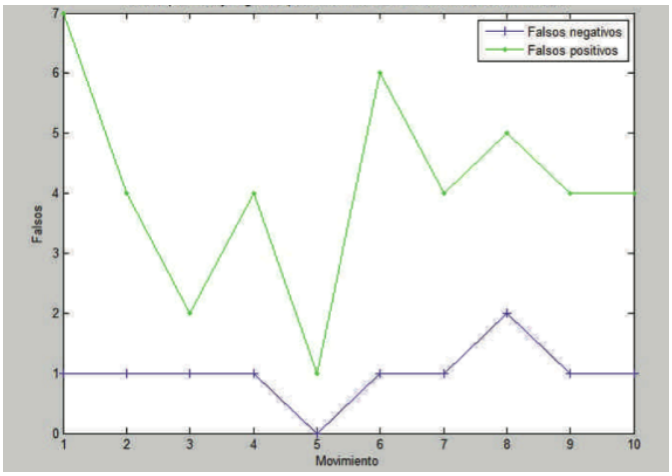


Figura 6.6: Falsos positivos y negativos para una rutina de 10 movimientos con un dedo.

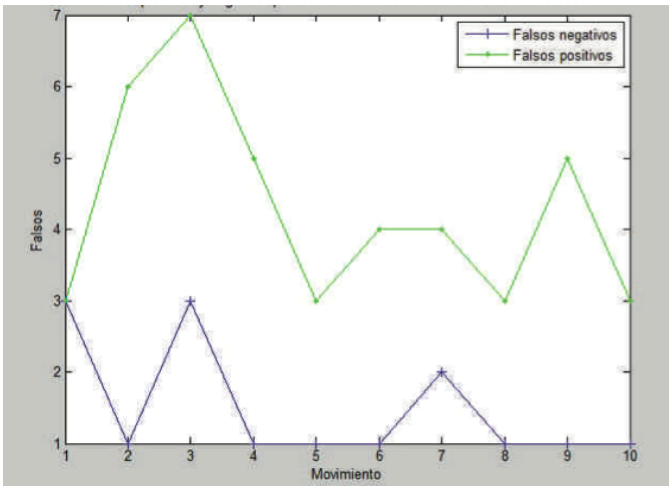


Figura 6.7: Falsos positivos y negativos para una rutina de 10 movimientos con dos dedos.

Observando las Figuras 6.6 y 6.7 comprobamos que en nuestro sistema basado en DSI aparecen pocos falsos negativos (más cuantos más dedos haya) pero que se dispara la aparición de falsos positivos por la cantidad de brillos que se generan por la luz solar, por reflejos dentro de la mesa.

Capítulo 7

Conclusiones y trabajo futuro

Una vez finalizado el proyecto y obtenidos los resultados se procede a realizar un breve análisis del trabajo realizado en el que se incluye: el grado de cumplimiento de los objetivos principales, valoración y posibles mejoras o trabajo futuro en este ámbito.

7.1. Conclusiones

En los requisitos del proyecto se especificaban los objetivos principales que se debían lograr. Como se puede observar a lo largo de la memoria los resultados y los objetivos se han cumplido satisfactoriamente:

1.- Se han controlado tanto los mensajes MIDI como los mensajes de Audio a través de objetos y eventos táctiles que captura una cámara web haciendo uso de la visión por computador.

2.- Se ha desarrollado una interfaz simple e intuitiva para controlar el secuenciador de Audio y MIDI Ableton live!

3.- Se ha estudiado el protocolo OSC para la generación automática de contenidos audiovisuales mediante Processing y Pure Data.

4.- Se ha construido un prototipo funcional capaz de reconocer objetos y eventos táctiles con el fin de generar música de una manera cómoda e intuitiva.

Además de cumplir los objetivos, todo lo que se ha desarrollado se ha incorporado en diferentes aplicaciones para demostrar el correcto funcionamiento de nuestro dispositivo multitouch.

7.2. Valoración personal

En primer lugar, partiendo de un conocimiento muy básico de los sistemas de visión por ordenador y del procesado de imagen, he aprendido y comprendido sus distintas etapas, métodos etc. Esto me ha servido para comprender que el campo de la visión artificial es realmente muy extenso y no se ha podido hablar de todo en este proyecto, por falta de tiempo y porque no era el objetivo principal del proyecto. También he comprendido la dificultad de crear un sistema de visión robusto, que no se deba solo a una escena configurada donde tenemos controlados los colores que aparecen en ella y la iluminación.

En segundo lugar, he aprendido un nuevo lenguaje de programación, Processing, he investigado diferentes librerías escritas para Processing por la comunidad, lo cual me ha servido para comprender la potencia que puede tener una buena comunidad alrededor de un proyecto opensource. He estudiado las clases en Processing y con ello he aprendido conceptos entorno a estructura de las clases en Java, como qué son las Interfaces, clases heredadas, etc. También he conocido diferente software/librerías para la programación multimedia, no solo para Processing, sino para todo tipo de entornos, todas ellas opensource, como pueden ser OpenGL, GStreamer, MidiBus, OscP5, etc.

Al ser Processing un proyecto pensado para gente relacionada con el mundo del arte, he conocido todo un mundo de artistas digitales que se apoyan en proyectos como Processing para hacer sus creaciones, he podido ver muchos proyectos que se están llevando a cabo en el mundo artístico y que tienen una componente técnica muy grande, en mi opinión el artista de creaciones digitales cada vez más tendrá un perfil de programador o una base más técnica, por tanto cada vez más, este tipo de entornos enfocados a la creación de arte digital serán más utilizados y más completos.

He podido comprobar resultados con usuarios reales, lo que ha provocado un feedback del trabajo conseguido. Además el trabajar con músicos y dj's, un tipo de usuario muy exigente, ha facilitado la mejora de las funcionalidades a través de las conclusiones obtenidas de sus reacciones.

Por otro lado la implementación del prototipo funcional me ha permitido estudiar las fortalezas y las debilidades de los dispositivos multitáctiles basados en tecnología óptica y entender el funcionamiento de estos.

Si nos ceñimos al dispositivo DSI, la elección de algunos de los materiales (iluminación infrarroja y filtros) es crítica y la obtención de una luz difusa y uniforme dentro de la mesa, sin brillos,... es tediosa. Por ello, el proceso de configuración y de calibración es extenso y es fácil obtener un prototipo inestable. Debemos sin embargo decir que nuestro sistema DSI funciona adecuadamente en un ambiente controlado y con una luz ambiental tenue.

Esperamos, como escribíamos en el resumen inicial, que este Proyecto de Fin de Carrera pueda servir de base para nuevas investigaciones en la materia. El sistema multitáctil no va a sustituir al ratón, ni a los ordenadores personales convencionales, pero ofrece unas posibilidades de trabajo admirables. El simple hecho de naturalizar la interacción del usuario con el dispositivo, obviando intermediarios, de que podamos comunicarnos mediante gestos con el aparato (el número de gestos posibles sólo depende del

programador), o de que podamos multiplicar por diez la velocidad de determinadas tareas, ya nos indica que sería un error subestimar las posibilidades de estos dispositivos.

7.3. Trabajo futuro

Como trabajo futuro o mejoras que se podrían incorporar a este proyecto han surgido varias ideas que se comentan a continuación:

- Modificar el framework reacTIVision para una calibración automática y no tener que depender del folio de calibración que proporciona la propia librería.
- Ampliar el algoritmo de limpieza de suciedad de las imágenes para limpiar de suciedad el dibujo sin afectar al propio dibujo. Sería necesario un algoritmo más complejo para poder eliminar las zonas de suciedad y distinguirlas de las zonas del dibujo, sin eliminar estas últimas.
- Ampliar el sistema de detección de fiduciales, para que las regiones de búsqueda no sean cuadradas, sino que se puedan adaptar a formas rectangulares.
- Lectura de la plantilla de Ableton Live! a través de processing para que se puedan reconocer todos los clips que existen en un track y no estar limitados solo a cuatro archivos de audio por track.
- Ampliar el control del oscilador y los elementos que interactuan con él para poder cubrir todos los parámetros que permiten modificar las características del sonido entrante.
- Diseñar una mesa plegable y de dimensiones menores a la planteada en este Proyecto de Fin de Carrera.
- Sincronización gráfica del objeto encargado de las repeticiones y del objeto que representa el arpegiador.

Apéndice A

Manual de usuario

A.1. Introducción

LogyWave es un instrumento musical electrónico colaborativo basado en una superficie tangible con reconocimiento de objetos que permite a los músicos y Dj's ser creativos de una forma directa, innovadora y sin precedentes para dar lugar a una composición única y flexible.

LogyWave abre la puerta a una nueva forma de entender la música, adoptando la filosofía de los primeros sintetizadores modulares, y haciendo uso de los programas de software estándar para producción musical y actuaciones en directo.

Figura A.1: Logywave.

Existen dos tipos de objetos, los cuadrados que generan sonido y los objetos redondos que modifican los sonidos. Al colocar estos objetos en la superficie de la mesa, los módulos se activan y su estado se visualiza en la pantalla de la mesa. Los parámetros de los módulos de procesamiento se modifican girando los objetos y a través de gestos táctiles.

A.2. Requisitos

Para poder ejecutar el software detrás de LogyWave es necesario que el equipo utilizado tenga instalado el sistema operativo Windows 7 ó MacOS X y la aplicación ReacTIVision para la detección de los objetos y los gestos táctiles.

A.3. Instalación

Para el correcto funcionamiento de la aplicación se recomienda seguir los siguientes pasos:

1.- Copiar la carpeta LiveOSC en el directorio Remote Scripts de Ableton Live!.

2.- Ejecutar JackOSX y poner el servidor de audio en funcionamiento presionando el botón Start.

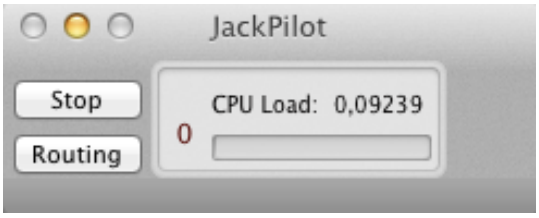


Figura A.2: Servidor JackOSX.

3.- Ejecutar el archivo sendwave.pd en Pure Data y seleccionar las salidas de sonido del servidor JackOSX.

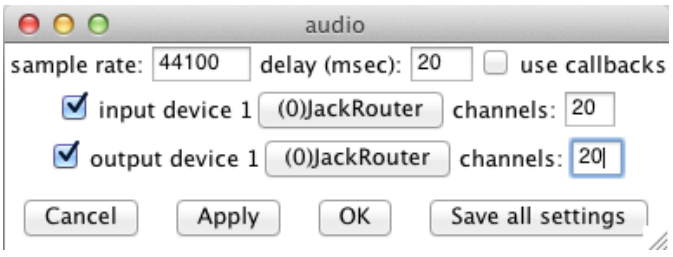


Figura A.3: Configuración del sonido en Pure Data.

4.- Ejecutar el archivo configaudio en el servidor JackOSX para crear las relaciones entre Pure Data y Ableton Live!.

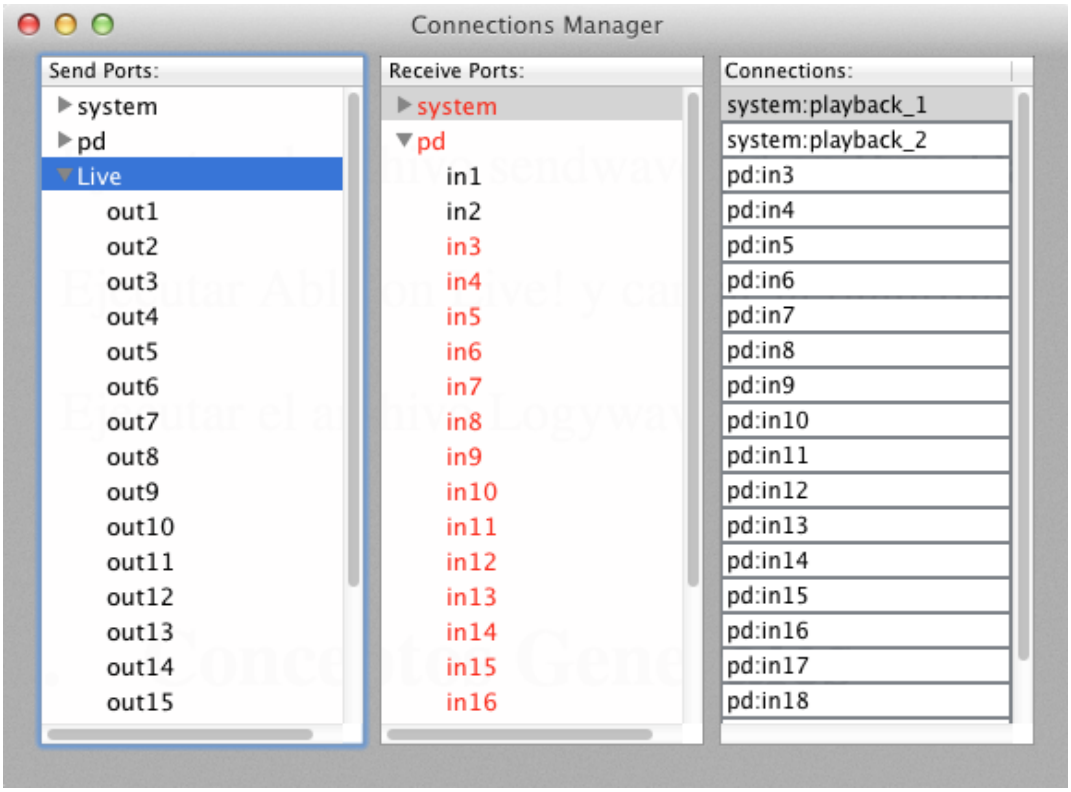


Figura A.4: Configuración del sonido en Pure Data.

5.- Ejecutar Ableton Live! y cargar la plantilla Logywave.als

6.- Ejecutar el archivo Logywave

A.4. Conceptos Generales

Al ejecutar Logywave una esfera que marca el tempo de la plantilla cargada en Ableton Live! se situa la mitad de la pantalla. Esta esfera representa la salida de sonido y cualquier objeto que se una a este punto se puede escuchar.

A.4.1. Tipos de objetos

En logywave existen 3 tipos de objetos.

1.- Generadores de sonido

2.- Efectos de Audio

3.- Controladores Globales

A.4.1.1. Generadores de sonido

Las piezas cuadradas son generadores de sonido. Un generador puede ser un reproductor de clips de Ableton Live! o un oscilador. Los generadores tienen en común que generan sonido por sí mismos y no necesitan estar conectados a otros objeto para escucharlos. Cuando se ponen encima de la superficie se conectarán a la salida de audio o a un efecto de audio.

- **Reproductor de clips de Ableton Live!**

  El objeto de reproducción de bucles, lanzará una de las pistas o una de las ranuras internas en una pista del secuenciador Ableton Live! El usuario podrá controlar más de 20 bucles para que pueda realizar sus mezclas de audio. Los bucles estarán sincronizados al tempo que marque el Ableton Live!

  En el modo de bucle, al girar el objeto podemos seleccionar aquellas ranuras de una pista en concreto. Hay 4 bucles a cada lado del objeto como se puede observar en la figura A.5.

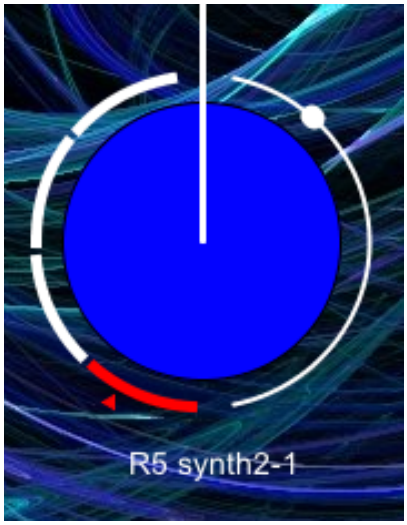


Figura A.5: Reproductor de clips de Ableton Live!.

  El objeto se iluminará del mismo color que el clip correspondiente en Ableton Live! y en su parte inferior nos indicará el nombre del clip que se está reproduciendo.

  La amplitud del reproductor de clips se puede cambiar moviendo con un dedo el punto en el lado derecho del objeto.

- **Oscilador**

  El oscilador es un contenedor de diferentes formas de onda de bajo nivel. Entre las ondas que puede generar se encuentra la onda seno, la onda cuadrada, la onda triangular y el ruido blanco.

  Al rotar el objeto la frequencia del oscilador irá cambiando. Si rotamos el objeto a la derecha se incrementa el valor de la frequencia, mientras que si rotamos el objeto a la izquierda se decrementará el valor de frequencia.

  La amplitud del oscilador se puede cambiar moviendo con un dedo el punto en el lado derecho del objeto.

A.4.1.2. Efectos de audio

Los efectos y filtros tienen una forma redonda. Necesitan una entrada de sonido de un generador con el fin de generar sonido. La mejor manera de utilizar un objeto de efecto es ponerlo en la línea de conexión de un generador. De esta forma el efecto se aplicará automáticamente al sonido que el generador está emitiendo.

- **Filtros**

  El objeto de filtro, filtra el audio procedente de otro objeto. Varios tipos de filtros son accesibles.

  1.- Filtro paso alto:Es un tipo de filtro electrónico caracterizado por permitir el paso de las frecuencias más altas y atenuar las frecuencias más bajas.

  2.- Filtro paso Banda:Es un tipo de filtro electrónico que deja pasar un determinado rango de frecuencias de una señal y atenúa el paso del resto.

  3.- Filtro paso bajo: Es un tipo de filtro electrónico caracterizado por permitir el paso de las frecuencias más bajas y atenuar las frecuencias más altas.

Figura A.6: Efecto de filtrado.

El color del filtro será el mismo que el de la figura a la que se ha unido. La rotación cambia la frecuencia de corte del filtro en caso de un filtro de paso bajo o un filtro de paso alto. En caso de un filtro paso banda se controla la frecuencia central.

La cantidad de resonancia del filtro puede variar moviendo con un dedo el punto en el lado derecho del objeto.

- **Reverb y delay**

  El efecto de retardo (Delay) registra una entrada de audio, la mezcla en su buffer interno con la cantidad feedback y, a continuación se reproduce después de un período de tiempo.

  El color del efecto delay será el mismo que el de la figura a la que se ha unido. La rotación del objeto determina el tiempo de duración del delay. El control es continuo dentro de los valores de tono cuando es cercano a cero y es cuantizado a las duraciones de tempo para valores grandes.

  Varios tipos de delay se pueden aplicar.

  1.- Reverb: Un efecto de reverberación que simula la persistencia del sonido en un espacio particular, utilizando el número de varios ecos.

  2.- Ping-Pong Delay: Cada repeticion se escucha por un canal diferente pasando de un lado a otro.

  3.- Feedback Delay: Efecto de retraso del sonido con retroalimentación

Figura A.7: Efecto de delay.

La cantidad de retroalimentación del efecto delay puede variar moviendo con un dedo el punto en el lado derecho del objeto.

- **Modulador**

  El modulador es un contenedor de varios efectos que utilizan algún tipo de modulación en el sonido entrante. Los efectos son diferentes modulación en anillo, chorus y flanger.

El parámetro principal del efecto se cambia girando el objeto. El parámetro depende del tipo de objetos, para la modulación en anillo es la modulación de tono, para el chorus es la profundidad y para el flanger es la frecuencia.

Varios tipos de moduladores se pueden aplicar.

1.- Modulación en anillo: El tono del efecto de modulación en anillo se controla girando el objeto.

2.- Chorus: Al girar el objeto la profundidad del efecto de coro se cambia.

3.- Flanger: La velocidad del efecto flanger se puede cambiar con la rotación del objeto.

La cantidad de modulación del efecto modulador puede variar moviendo con un dedo el punto en el lado derecho del objeto.

- **Repetidor**

  Es un efecto de audio que se une de manera especial a los generadores de sonido. Producirá repeticiones del sonido seleccionado hasta que se quite el objeto de la superficie.

  La rotación del objeto determinará el tiempo de duración del bucle.

- **Arpegiador**

  El arpegiador es un objeto que sólo modifica el sonido saliente del Oscilador. Es un secuenciador de 16 pasos, que envía eventos de nota reproduciendo una melodía.

  Cada uno de los 16 pasos del arpegiador pueden ser activados o desactivados presionando uno de las ranuras del objeto. Para modificar la frecuencia de cada unas de las 16 notas, existe un recuadro junto a esta figura donde modificaremos los valores de la frecuencia.

A.4.1.3. Controladores globales

Los controladores globales son en forma de estrella y no pueden conectarse a ningún objeto. Cambian el comportamiento de toda la aplicación, como el tempo global o volumen.

El controlador tiene dos subtipos:

- **Ajuste de tempo**

  Se utiliza para cambiar uno de los parámetros globales de la mesa como es el tempo actual.

  Al girar el objeto cambia el tempo de la canción. El número de beats por minutos (BPM) se muestra cada vez que cambia el valor.

- **Ajuste de volumen global**

  Se utiliza para controlar los parámetros que afectan al módulo de salida de la mesa.

  Al girar el objeto cambiará el volumen global de la mesa.

A.4.2. Gestos táctiles

En el software se reconocen dos tipos de eventos táctiles:

1.- Silenciar objetos

Las conexiones entre los objetos pueden dividirse temporalmente a través del gesto de silencio. Esto se realiza trazando una línea con el dedo de un lado de una conexión con el otro lado. Con el fin de activar una conexión silenciada el mismo gesto se repetirá. Si el objeto ha sido silenciado, la línea de unión será discontinua como se puede observar en la figura A.8

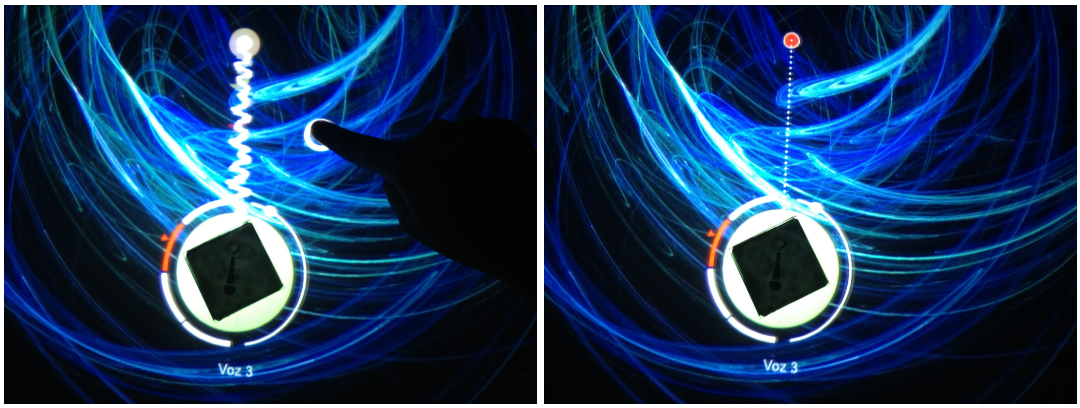


Figura A.8: Objeto silenciado.

2.- Control de parámetros

La mayoría de los objetos tienen un parámetro que se puede cambiar tocando la pantalla en el lado derecho del objeto. Este parámetro se corresponde con la amplitud de los generadores de sonido, y el dry/wet para los efectos.

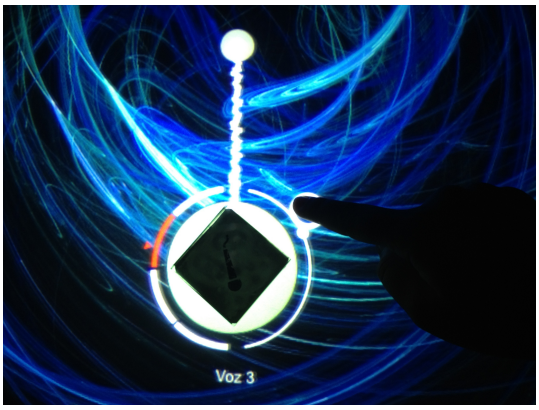


Figura A.9: Control del parámetro amplitud.

Apéndice B

Reglas UML utilizadas

El Lenguaje Unificado de Modelado (UML, Unified Modeling Language) es una lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema con gran cantidad de software. UML proporciona una forma estándar de representar los planos de un sistema, y comprende tanto elementos conceptuales, como los procesos de negocio y las funciones del sistema, en cuanto a elementos concretos, como a las clases escritas en un lenguaje de programación específico, esquemas de bases de datos y componentes software reutilizables.

En el caso del presente proyecto UML se ha utilizado tanto en la fase de análisis, para describir los casos de uso de la aplicación y los diagramas de clases que surgen como consecuencia de las relaciones entre las distintas entidades implicadas, y en la fase de diseño, para especificar el tanto las relaciones entre las distintas clases.

B.1. Componentes

Los componentes que se han utilizado en los distintos diagramas UML en las diferentes secciones son los siguientes:

- Clase: es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica. Una clase implementa una o más interfaces. Gráficamente se representa como un rectángulo que incluyen su nombre, atributos (si los hay) y sus operaciones.
- Atributos: se pueden especificar indicando su clase y quizás su valor inicial por defecto.
- Operaciones: se pueden especificar indicando su signatura, la cual incluye el nombre, tipo, y valores por defecto de todos los parámetros y, en el caso de las funciones el tipo de retorno.

- Alcance estático o dinámico: especifica si cada instancia del clasificador tiene su propio valor de la característica (instance, valor por defecto que no requiere notación adicional) ó si hay un solo valor de la característica para todas las instancias del clasificador (static, se denota subrayando la característica).

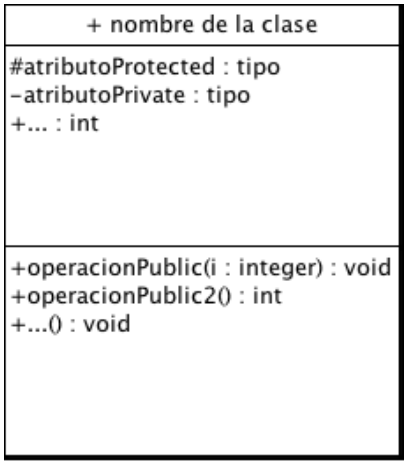


Figura B.1: Representación de clase.

- Interfaz : es una colección de operaciones que especifican un servicio de una clase o componente. Puede representar el comportamiento completo de una clase o sólo una parte. En ella se define un conjunto de especificaciones operacionales, no un conjunto e implementaciones operacionales. Normalmente los atributos de una interfaz no son relevantes, por lo que no se suelen mostrar en la representación gráfica, que es semejante a la de una clase exceptuando que tiene la palabra «interface» sobre el nombre.

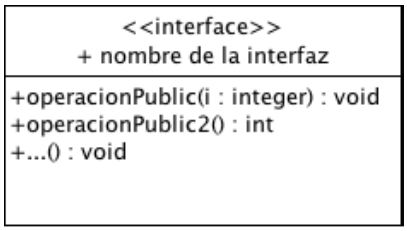


Figura B.2: Representación de interfaz.

- Caso de uso: se trata de una descripción de un conjunto de secuencias de acciones que ejecuta un sistema y que produce un resultado observable de interés para un actor particular. Se utiliza para estructurar los aspectos de comportamiento en un modelo. Gráficamente se representa como una elipse de borde continuo que normalmente sólo incluye su nombre.

Figura B.3: Representación de caso de uso.

- Paquete: es un elemento de agrupación de propósito general para organizar el propio diseño. Un paquete es puramente conceptual, y se visualiza como una carpeta que normalmente sólo incluye el nombre.

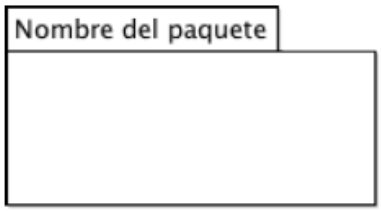


Figura B.4: Representación de paquete.

B.2. Relaciones

Las relaciones utilizadas en los diagramas UML de este proyecto son las siguientes

- Dependencias: es una relación de uso que declara que un elemento utiliza la información y los servicios de otro, pero no necesariamente a la inversa

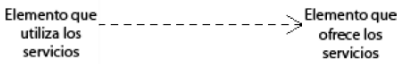


Figura B.5: Representación de dependencias.

- Asociaciones: es una relación estructural que especifica que los objetos de un elemento están conectados con los objetos de otro. Esta relación puede tener en cuenta el concepto de multiplicidad entre los objetos así como la agregación.

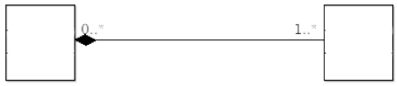


Figura B.6: Representación de asociación.

- Generalizaciones: es una relación entre un elemento general (superclase o padre) y un caso más específico de ese elemento (subclase o hijo).

Figura B.7: Representación de generalización.

- Realizaciones: es una relación semántica entre dos objetos en donde uno de ellos especifica un contrato que otro garantiza que cumplirá. Este tipo de relación se puede encontrar entre interfaces y las clases y componentes que las realizan.

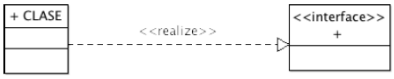


Figura B.8: Representación de realizaciones.

B.3. Tipos de diagramas utilizados

B.3.1. Diagrama de casos de uso

Muestra un conjunto de casos de uso y actores (un tipo especial de clase) y sus relaciones. Los diagramas de casos de uso cubren la vista de casos de uso estática de un sistema. Estos diagramas son especialmente importantes en el modelado y organización del comportamiento de un sistema.

Cada sistema precisa de interacciones, ya sean humanas o con otros sistemas, que usan el sistema con un determinado propósito. Estos actores esperan que el sistema se comporte de una manera predecible. Los casos de uso especifican este comportamiento, describiendo las secuencias de acciones, incluyendo las variaciones que el sistema realiza para obtener el resultado que espera el actor.

El punto fundamental de los casos de uso, es que capturan el comportamiento esperado del sistema, sin necesidad de especificar cómo se implementaría. Además los casos de uso permiten validar y verificar que el sistema se desarrolla correctamente.

Uno de los errores que a menudo se cometen con los diagramas, es pensar que capturan toda la funcionalidad. Si esto fuera así, serían una herramienta que nos descargaría de describir los casos de uso mediante texto. Por el contrario, los diagramas no son más que herramientas de dos dimensiones que nos permiten exponer las relaciones; pero en ningún caso reemplazan el texto.

Un caso de uso bien escrito se puede leer con facilidad. Consiste en una serie de frases escritas con una sola forma gramatical, con acciones simples en las que el actor alcanza un resultado o pasa información a otro.

Por otra parte, la escritura de casos de uso es una tarea complicada, en la que hay que dominar tres cuestiones:

- Entorno, ¿cuál es el sistema bajo discusión?
- El actor principal, ¿quién tiene el objetivo?
- El nivel (de abstracción), ¿cómo de alto o bajo es el nivel del objetivo?

El actor es alguien o algo que tiene un comportamiento. El actor principal es quien inicia la interacción con el sistema, y probablemente quien inicia el caso de uso. En cualquier caso, el actor principal tiene un objetivo, que ha de satisfacer el sistema.

Actores

Por lo general, aunque el actor principal sea X, cualquier otro actor que tenga un nivel de competencias superior a él; también podrá ser un actor principal del caso de uso.

Para aclarar lo que puede hacer cada actor, se definen roles, de forma que cada usuario pueda pertenecer a uno o varios roles. En cuanto a los casos de uso, se nombrará al rol y no las personas concretas que puedan llevarlo a cabo.

Una mera lista de actores no tiene gran utilidad, hay que definir las habilidades que tiene cada uno.

En un caso de uso, son los actores que proveen de un servicio; por ej. una impresora, un servidor web o un grupo de investigadores que nos proveen de datos. Los actores de apoyo permiten identificar las interfaces externas y los protocolos que los rigen. Hay que tener en cuenta que un actor principal, puede ser un actor de apoyo en otro caso de uso.

El propio sistema es un actor, aunque no sea un actor principal ni de apoyo en ninguno de los casos de uso; ya que por lo general no determina el inicio de ninguna acción.

Niveles de objetivos

Hay tres:

- Global: ¿Por qué?
- Usuario: Lo que el actor principal realmente quiere.
- Subvenciones: ¿Cómo?

Los objetivos del usuario, son los que tienen los actores principales. Por ejemplo crear una factura, enviar una factura. Estos objetivos pueden hacer uso de otros sub-objetivos, que estarán en el nivel de subfunciones; por ej. buscar un artículo.

En cuanto a los objetivos globales, incluyen a múltiples objetivos de usuario; y tienen como propósito describir el sistema:

- Muestran el contexto en que operan los casos de uso.
- Muestran el ciclo de vida y la secuencia de objetivos relacionados.

En general, los objetivos de usuario son los importantes. Los objetivos globales dan un contexto a los anteriores. Mientras que las subfunciones sólo hay que usarlas cuando lo requiera un objetivo de usuario; ya sea por clarificar el caso de uso, o porque múltiples casos de uso hagan referencia.

Una de las claves para determinar el nivel de un caso de uso, es preguntarse ¿Es lo que el actor principal realmente quiere?, si la respuesta es afirmativa, se trata de un objetivo de usuario.Otra pregunta que ayuda es, ¿por qué está el actor haciendo eso?.

En general, los casos de uso han de tener en torno a tres u ocho pasos, si se incluyen más probablemente se esté incurriendo en detalles de bajo nivel.

Precondiciones, Garantías y triggers

Las precondiciones establecen lo que debe de cumplirse antes de que se dispare el caso de uso. Hay que tener cuidado, porque lo que generalmente se cumple, no siempre es obligatorio. Las garantías mínimas han de cumplirse siempre, y son el compromiso que se tiene con los interesados, en particular con el actor principal. Han de satisfacerse incluso si el objetivo del caso de uso se abandona. No hay que describir los posibles fallos, pues éstos ya se encuentran en la sección extensions. Una ayuda para detallarlas es preguntarnos, ¿qué haría infelices a los interesados en caso de error?

Por otra parte, las garantías de éxito sólo han de cumplirse si el caso de uso termina con éxito. Son un añadido a las garantías mínimas. Las garantías de éxito satisfacen los intereses de los actores. La pregunta clave es, ¿qué haría infelices a los interesados en caso de éxito?

El trigger es lo que comienza el caso de uso; en ocasiones es el evento que precede al primer paso, y en otras constituye el primer paso.

Escenario principal

Con el objetivo de dar una idea clara del caso de uso se diferencia entre el escenario principal y las extensiones. En el primero, se detalla cuáles son los pasos que se siguen cuando todo ocurre sin problemas; de este modo se facilita la compresión en una primera lectura. Las complicaciones, si las hubiera, quedarán detalladas en las extensiones.

El escenario se compone de pasos, que describen una acción simple.

Hay tres tipos de acciones:

- Interacciones, entre dos actores.
- Validación, llevada a cabo para proteger el interés de alguno de los actores.
- Un cambio interno, para satisfacer a alguno de los interesados.

Extensiones

Las extensiones son las posibles variaciones a partir del escenario principal. Se colocan debajo del escenario principal. Para cada posible variación se comienza exponiendo la condición que la desencadena, y después se detallan los pasos que la manejan. Al final de la extensión se indica si se vuelve al escenario principal o se termina el caso de uso.

Los pasos de la extensión son una continuación del punto del escenario principal del que derivan, por tanto es como si estuvieran allí. Esto facilita la lectura de las extensiones.

Plantilla utilizadas para los casos de uso

	Título
Id	Identificador
Actor Principal	Es quien tiene el objetivo que cubre el caso de uso.
Otros Actores	
Precondiciones	Han de cumplirse antes de que se ejecute el caso de uso.
Evento inicial	Suceso a partir del cual se origina la secuencia de caso de uso.
Garantías mínimas	Han de satisfacerse tras la ejecución del caso de uso.
Garantías de éxito	
Flujo normal	Caso en que todo sale bien.
Flujos alternativos	Lo que puede variar a partir del main success scenario. Tiene asociado el número del paso del main success scenario en que tienen lugar.
Notas	

Tanto en el escenario principal como en las extensiones, pueden hacerse referencia a otros sub-casos de uso, esto se puede indicar subrayándolos.

B.3.2. Diagrama de clases

Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Estos diagramas son los más comunes en el modelado de sistemas orientados a objetos, como es el caso que nos ocupa.

B.3.3. Diagrama de componentes

Representa la encapsulación de una clase, junto con sus interfaces, puertos y estructura interna, la cual está formada por otros componentes anidados y conectores. Este tipo de de diagramas cubren la vista de implementación estática del diseño de un sistema.

B.3.4. Diagrama de despliegue

Muestra la configuración de nodos de procesamiento en tiempo de ejecución y los artefactos que residen en ellos. Abordan la vista de despliegue estática de una arquitectura.

B.3.5. Diagrama de paquetes

Muestra cómo un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Dado que normalmente un paquete está pensado como un directorio, los diagramas de paquetes suministran una descomposición de la jerarquía lógica de un sistema.

Los Paquetes están normalmente organizados para maximizar la coherencia interna dentro de cada paquete y minimizar el acoplamiento externo entre los paquetes. Con estas líneas maestras sobre la mesa, los paquetes son buenos elementos de gestión. Cada paquete puede asignarse a un individuo o a un equipo, y las dependencias entre ellos pueden indicar el orden de desarrollo requerido.

Apéndice C

Mensajes MIDI

Los mensajes MIDI están compuestos por dos o tres bytes, uno de los cuales es el de estado (status byte) y el resto que son de datos (data bytes). Los bytes de estado se caracterizan por tener el bit de mayor peso a 1 mientras que los de datos lo tienen a 0. En algunas ocasiones, según el dispositivo MIDI, puede omitirse el byte status si es el mismo que el emitido en el mensaje anterior. El formato de los mensajes MIDI se puede ver en la siguiente figura.

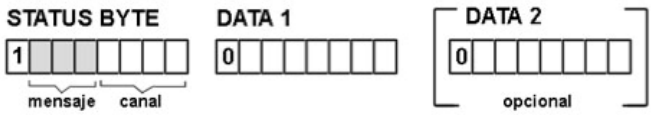


Figura C.1: Estructura de un Mensaje MIDI.

El status byte dispone de 3 bits para definir el tipo de mensaje y 4 bits para el canal. Consecuentemente hay 8 tipos de mensajes diferentes, los cuales vienen descritos en la Figura C.2.

Nombre	Status byte	Data 1	Data 2
Desactivación de nota	1000 nnnn (0x8N)	Tono	Velocidad
Activación de nota	1001 nnnn (0x9N)	Tono	Velocidad
Postpulsación polifónica	1010 nnnn (0xAN)	Tono	Presión
Cambio de control	1011 nnnn (0xBN)	Tipo de control	Valor
Cambio de programa	1100 nnnn (0xCN)	Programa	-
Postpulsación monofónica de canal	1101 nnnn (0xDN)	Presión	-
Pitch	1110 cccc (0xEN)	MSByte	LSByte
Mensaje del sistema	1111 xxxx(0xFX)		

Figura C.2: Tabla de Mensajes MIDI.

- nnn: representa los cuatro bits que indican el canal al que se envía el mensaje
- N: representa el valor nnnn en hexadecimal.
- Todos los valores de datos tienen una resolución de siete bits (el octavo siempre es 0 para indicar que se trata de un byte de datos) por ello pueden representar valores entre 0 y 127 inclusive.
- Hay mensajes como Cambio de programa y Postpulsación monofónica de canal que únicamente utilizan un byte de datos.
- En el mensaje Pitch, los dos datos conforman un único valor de 14 bits cuyo valor puede variar entre -8192 y +8191.
- Los mensajes exclusivos del fabricante siguen su propia estructura y no se aplican a ningún canal en particular.

C.1. Mensajes de canal

Reciben el nombre de mensajes de canal aquellos que se aplican a un canal en particular, es decir, todos menos los mensajes exclusivos del fabricante.

C.1.1. Activación de nota

Estos mensajes indican al dispositivo que debe iniciar una nota.

- El primer byte de datos indica el tono de la nota. En total existen 128 posibles notas siendo la 0 la más grave y la 127 la más aguda. Ya que una octava tiene doce notas, MIDI soporta más de diez octavas. Los múltiplos de 12 corresponden con las notas Do siendo el valor 60 la nota Do central de un piano. En la Figura C.3 se muestra la relación entre los tonos y las notas MIDI.

OCTAVA -2			OCTAVA -1			OCTAVA 0		
NOTA	Nº	HZ	NOTA	Nº	HZ	NOTA	Nº	HZ
DO	0	8.18	DO	12	16.35	DO	24	32.70
DO#	1	8.66	DO#	13	17.32	DO#	25	34.65
RE	2	9.18	RE	14	18.35	RE	26	36.71
RE#	3	9.72	RE#	15	19.45	RE#	27	38.89
MI	4	10.30	MI	16	20.60	MI	28	41.20
FA	5	10.91	FA	17	21.83	FA	29	43.65
FA#	6	11.56	FA#	18	23.12	FA#	30	46.25
SOL	7	12.25	SOL	19	24.50	SOL	31	49.00
SOL#	8	12.98	SOL#	20	25.96	SOL#	32	51.91
LA	9	13.75	LA	21	27.50	LA	33	55.00
LA#	10	14.57	LA#	22	29.14	LA#	34	58.27
SI	11	15.43	SI	23	30.87	SI	35	61.74

OCTAVA 1			OCTAVA 2			OCTAVA 3		
NOTA	Nº	HZ	NOTA	Nº	HZ	NOTA	Nº	HZ
DO	36	65.41	DO	48	130.81	DO	60	261.63
DO#	37	69.30	DO#	49	138.59	DO#	61	277.18
RE	38	73.42	RE	50	146.83	RE	62	293.66
RE#	39	77.78	RE#	51	155.56	RE#	63	311.13
MI	40	82.41	MI	52	164.81	MI	64	329.63
FA	41	87.31	FA	53	174.61	FA	65	349.23
FA#	42	92.50	FA#	54	185.00	FA#	66	369.99
SOL	43	98.00	SOL	55	196.00	SOL	67	392.00
SOL#	44	103.83	SOL#	56	207.65	SOL#	68	415.30
LA	45	110.00	LA	57	220.00	LA	69	440.00
LA#	46	116.54	LA#	58	233.08	LA#	70	466.16
SI	47	123.47	SI	59	246.94	SI	71	493.88

Figura C.3: Relación entre los tonos y las notas MIDI.

- El segundo byte indica la velocidad de ataque, o fuerza con la que se pulsa la tecla. Este valor se asocia con la intensidad sonora aunque algunos sintetizadores son capaces de modificar el timbre de la nota como sucede con los instrumentos reales. La velocidad 0 tiene un significado especial ya que reproduce una nota que no se escucha, por lo que funciona como si se soltase la nota.

C.1.2. desactivación de nota

Su funcionamiento es similar al de activación de nota con una velocidad 0, es decir, indica al dispositivo que debe detener una nota.

- El primer byte indica el tono de la nota.
- El segundo byte indica la velocidad con la que se suelta.

Muchos sintetizadores no tienen en cuenta este mensaje y utilizan el de activación de nota con velocidad 0. La velocidad con la que se suelta la nota podría afectar a la forma en la que el sonido desaparece.

C.1.3. Postpulsación polifónica

Algunos teclados son capaces de detectar de forma permanente los cambios de presión ejercida sobre sus teclas. Cuando se produce un cambio de presión se envía este tipo de mensajes.

- El primer byte indica el tono de la nota.
- El segundo byte indica la presión ejercida sobre la nota.

Algunos sintetizadores, al recibir estos mensajes, son capaces de alterar el timbre y el nivel sonoro de la nota. Otros, sin embargo, los ignoran.

C.1.4. Cambios de control

Este tipo de mensajes es diferente a los demás, ya que permite definir hasta 128 tipos de mensajes diferentes utilizando el primer byte de datos. Todos ellos sirven para modificar la calidad del sonido.

- El primer byte define el tipo de mensaje.
- El segundo byte el valor que se le quiere asignar.

A continuación se describen los tipos de control más utilizados:

- Cambio de control 0: Cambio de banco. Si el sintetizador dispone de varios bancos de sonidos, este mensaje permite alternar entre ellos. Normalmente los sintetizadores tienen hasta 128 instrumentos, pero, pueden tener más haciendo uso de los bancos de sonidos. Tras hacer un cambio de control de este tipo, lo normal es hacer un cambio de programa.
- Cambio de control 7: Volumen. Este es uno de los controles más utilizados ya que permite cambiar el volumen de un canal.
- Cambio de control 10: Panorama. Permite definir la posición sonora de un canal, en un ámbito de 180 grados. Sus valores pueden ser 0, todo el sonido se emite por la izquierda, 64 lo centra y 127 lo sitúa a la derecha. Cualquier valor intermedio es igualmente válido.
- Cambio de control 64: Sostenido. Produce el efecto similar al pedal de un piano, es decir, las notas se mantienen más tiempo. Sus valores pueden estar entre 0 y 63 para desactivarlo y 64-127 para activarlo.
- Cambio de control 91: Reverberación. La reverberación indica la relación entre el sonido directo y el sonido reflejado. Este efecto se utiliza para simular la acústica de salas de concierto.
- Cambio de control 93: Chorus. Este efecto también es muy utilizado y produce un efecto semejante a duplicar los instrumentos, por lo que parece que se aumente el "grueso"del sonido.

C.1.5. Cambio de programa

Programa o patch hace referencia a los diferentes instrumentos disponibles en el sintetizador.

Este mensaje permite cambiar el programa del canal indicado.

- El primer y único byte indica el programa a utilizar.

Algunos sintetizadores disponen de más de 128 instrumentos. En estos casos los programas se agrupan en diferentes bancos. Para poder cambiar de banco es necesario utilizar un mensaje del tipo Cambio de control 0 mencionado anteriormente.

Los 128 instrumentos especificados en MIDI se pueden ver en la Figura C.4.

C.1.6. Postpulsación monofónica de canal

Cumple la misma función que el mensaje Postpulsación polifónica, la diferencia radica en que, la presión enviada, hace referencia a todo el canal en vez de a cada una de las notas. Su valor suele ser la mayor presión de todas las notas pulsadas en ese canal.

C.1.7. Pitch

Sirve para variar el tono de la nota, es decir, desafinar ligeramente el sonido. En general MIDI define el rango de desafinación en +/- 2 semitonos.

- Los dos bytes conforman un único valor de 14 bits cuyo valor puede variar entre - 8192 y +8191.

C.1.8. Mensaje del sistema

Estos mensajes no se aplican a un canal específico si no al dispositivo en general. Los cuatro bits que se utilizan en otros mensajes para indicar el canal, se utilizan en este caso para indicar el subtipo de mensajes. Permiten, por ejemplo, sincronizar y coordinar diferentes dispositivos o posicionar un determinado secuenciador en una determinada posición de una pieza almacenada en memoria.

Patch Number o Programas			
00 - Piano de cola acústico	33 - Bajo eléctrico pulsado	64 - Saxo soprano	97 - Efecto 2 (banda sonora)
01 - Piano acústico brillante	34 - Bajo eléctrico punteado	65 - Saxo alto	98 - Efecto 3 (cristales)
02 - Piano de cola eléctrico	35 - Bajo sin trastes	66 - Saxo tenor	99 - Efecto 4 (atmósfera)
03 - Piano de cantina	36 - Bajo golpeado 1	67 - Saxo barítono	100 - Efecto 5 (brillo)
04 - Piano Rhodes	37 - Bajo golpeado 2	68 - Oboe	101 - Efecto 6 (duendes)
05 - Piano con "chorus"	38 - Bajo sintetizado 1	69 - Corno inglés	102 - Efecto 7 (ecos)
06 - Clavicordio	39 - Bajo sintetizado 2	70 - Fagot	103 - Efecto 8 (ciencia ficción)
07 - Clavinet	40 - Violín	71 - Clarinete	104 - Sitar
08 - Celesta	41 - Viola	72 - Flautín	105 - Banjo
09 - Carillón	42 - Violonchelo	73 - Flauta	106 - Shamisen
10 - Caja de música	43 - Contrabajo	74 - Flauta dulce	107 - Koto
11 - Vibráfono	44 - Cuerdas con trémolo	75 - Flauta de pan	108 - Kalimba
12 - Marimba	45 - Cuerdas con pizzicato	76 - Cuello de botella	109 - Gaita
13 - Xilófono	46 - Arpa	77 - Shakuhachi (flauta japonesa)	110 - Violín celta
14 - Campanas tubulares	47 - Timbales	78 - Silbato	111 - Shanai
15 - Salterio	48 - Conjunto de cuerda 1	79 - Ocarina	112 - Campanillas
16 - Órgano Hammond	49 - Conjunto de cuerda 2	80 - Melodía 1 (onda cuadrada)	113 - Agogó
17 - Órgano percusivo	50 - Cuerdas sintetizadas 1	81 - Melodía 2 (diente de sierra)	114 - Cajas metálicas
18 - Órgano de rock	51 - Cuerdas sintetizadas 2	82 - Melodía 3 (órgano de vapor)	115 - Caja de madera
19 - Órgano de iglesia	52 - Coro Aahs	83 - Melodía 4 (siseo órgano)	116 - Caja Taiko
20 - Armonio	53 - Voz Oohs	84 - Melodía 5 (charanga)	117 - Timbal melódico
21 - Acordeón	54 - Voz sintetizada	85 - Melodía 6 (voz)	118 - Caja sintetizada
22 - Armónica	55 - Conjunto de staccatos	86 - Melodía 7 (quintas)	119 - Platillo invertido
23 - Bandoneón	56 - Trompeta	87 - Melodía 8 (bajo y melodías)	120 - Trasteo de guitarra
24 - Guitarra española	57 - Trombón	88 - Fondo 1 (nueva era)	121 - Sonido de respiración
25 - Guitarra acústica	58 - Tuba	89 - Fondo 2 (cálido)	122 - Playa
26 - Guitarra eléctrica (jazz)	59 - Trompeta con sordina	90 - Fondo 3 (polisintetizador)	123 - Piada de pájaro
27 - Guitarra eléctrica (limpia)	60 - Corno francés (trompa)	91 - Fondo 4 (coro)	124 - Timbre de teléfono
28 - Guitarra eléctrica (tapada o muteada)	61 - Sección de bronces	92 - Fondo 5 (de arco)	125 - Helicóptero
29 - Guitarra saturada (overdrive)	62 - Bronces sintetizados 1	93 - Fondo 6 (metálico)	126 - Aplauso
30 - Guitarra distorsionada	63 - Bronces sintetizados 2	94 - Fondo 7 (celestial)	127 - Disparo de fusil
31 - Armónicos de guitarra		95 - Fondo 8 (escobillas)	
32 - Bajo acústico		96 - Efecto 1 (lluvia)	

Figura C.4: Tabla de programas o patch numbers.

C.2. Ejemplo de archivo MIDI

El siguiente código (FiguraC.5) muestra un archivo MIDI de ejemplo en el que se reproducen 10 notas: 4 de ellas por un piano y las 6 restantes por una viola. Cada par de dígitos en hexadecimal representa un byte.

```
4D 54 68 64 00 00 00 06 00 01 00 02 9E 21 4D 54 72 6B 00 00 00 1B 00 C1 01 8E 35 91 43 7F
87 68 91 43 00 85 61 91 47 7F 87 68 91 47 00 00 FF 2F 00 4D 54 72 6B 00 00 00 25 00 C2 29 9C
10 92 3C 7F 87 68 92 3C 00 89 50 92 3E 7F 87 68 92 3E 00 88 64 92 40 7F 87 68 92 40 00 00 FF
2F 00
```

Figura C.5: Archivo MIDI.

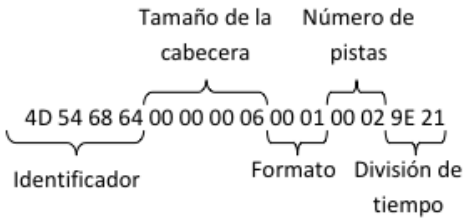
Para encontrar la cabecera del archivo debemos buscar los valores "MThd"(0x4D546864), mientras que las diferentes pistas las podemos encontrar buscando los valores "MTrk"(0x4D54726B). En la figura C.6 se muestra el mismo archivo MIDI separando la cabecera y las diferentes pistas.

```
//Cabecera
4D 54 68 64 00 00 00 06 00 01 00 02 9E 21
//Pista 1
4D 54 72 6B 00 00 00 1B 00 C1 01 8E 35 91 43 7F 87 68 91 43 00 85 61 91 47 7F 87 68 91 47 00
00 FF 2F 00
//Pista 2
4D 54 72 6B 00 00 00 25 00 C2 29 9C 10 92 3C 7F 87 68 92 3C 00 89 50 92 3E 7F 87 68 92 3E
00 88 64 92 40 7F 87 68 92 40 00 00 FF 2F 00
```

Figura C.6: Cabecera y pistas del archivo MIDI.

C.2.1. Análisis de la cabecera

A continuación se explican las diferentes partes de la cabecera:



- Identificador: Como se ha comentado anteriormente, el identificador de la cabecera siempre es 0x4D546864 o "MThd.[en] ASCII.

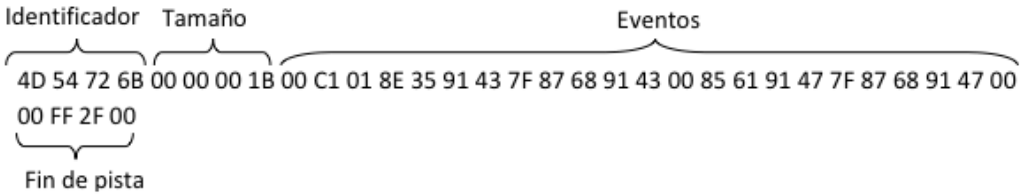
- Tamaño de la cabecera: Puesto que el tamaño del resto de cabecera (Formato + Número de pistas + División de tiempo) no cambia, siempre son 6 bytes.
- Formato: El formato indicado es el 1, por lo que en el archivo se disponen dos o más pistas que contienen los eventos.
- Número de pistas: En este caso, el número de pistas es 2.
- División de tiempo: La división de tiempo es 0x9E21 que en binario es

  1001111000100001

  El primer bit a 1 indica que la división del tiempo está en formato fotogramas por segundo. Así pues, de los 15 bits restantes, los 7 primeros corresponden con el número de frames por segundo, en este caso 30. El resto de bits indica el número de tiempos delta que tendrá cada frame, en este caso 33.

C.2.2. Análisis de la pista

A continuación se realiza un análisis de la primera pista. La segunda pista no se analiza ya que esencialmente es lo mismo que la primera. Los diferentes campos de la pista son:



- Identificador: El identificador de la pista siempre es 0x4D54726B que se corresponde con la cadena ASCII "MTrk".
- Tamaño: Indica la cantidad de bytes de la pista, en este caso 27 bytes.
- Eventos: Todos los eventos MIDI siguiendo el formato delta time – status byte – data 1 – data 2. Estando delta time en formato variable-length.

Ejemplos de eventos:

00 C1 01

El primer byte indica el tiempo delta, en este caso 0. Seguidamente el status byte indica que se trata de un mensaje de cambio de programa destinado al canal uno. El programa al que se cambia es el número 1, que corresponde al piano. El segundo bytes de datos no se utiliza.

8E 35 91 43 7F

En este segundo ejemplo se puede apreciar mejor cómo el tiempo delta está definido como un variable-length. Si tomamos los dos primeros bytes 0x8E35:

10001110 00110101

El primer bit indica que el valor continúa en el segundo byte. De forma que el delta time final es: 00011100110101, o lo que es lo mismo 1845 tiempos delta. Para pasarlos a tiempo real hay que aplicar la división de tiempo de la cabecera.

$$\frac{1845\ Tiempo\Delta}{\frac{33\ Tiempo\Delta\ /fotograma}{30\ fotogramas/s}} = 1.863\ s$$

Es decir, el evento se reproducirá tras transcurrir 1836 milisegundos desde que se produjo el último evento.

Seguidamente, tras la especificación delta time encontramos el status byte y los bytes de datos. El status byte indica que el evento se trata de una activación de notas en el canal 1. El primer dato indica que la nota a reproducir es Sol. Finalmente, el último byte indica que la velocidad de pulsación es la máxima (127).

- Fin de pista: En realidad también es considerado un evento pero con un significado especial: el fin de la pista.

Apéndice D

Protocolo OSC

D.1. Introducción

Open Sound Control (OSC) es un protocolo para la comunicación entre ordenadores, sintetizadores musicales y otros dispositivos multimediales inspirado en la moderna tecnología de las redes. El protocolo tiene algunas ventajas como por ejemplo la independencia del medio de transmisión y la flexibilidad para transportar cualquier tipo de datos. Pensado originalmente para la comunicación de instrumentos musicales (como en el caso del protocolo MIDI), en estos últimos años ha ganando terreno en otros sectores como el multimedia gracias a su potencia y flexibilidad.

El medio de transporte por excelencia son las redes de ordenadores y para obtener velocidad y simplificar las comunicaciones generalmente se usa el protocolo de transporte UDP (User Datagram Protocol) aunque si se pueden usar otros protocolos como por ejemplo el TCP.

D.2. Características principales del protocolo OSC

- Expansible, dinámico.
- Esquema de nombres simbólicos tipo URL.
- Datos simbólicos y numéricos a alta resolución
- Pattern matching que permite de comunicar simultáneamente con varios dispositivos a través de un único mensaje
- Indicadores de tiempo (time tags) de alta resolución

- Posibilidad de empaquetar varios mensajes para aquellos eventos que deben ocurrir simultáneamente.

D.3. Tipos de mensajes

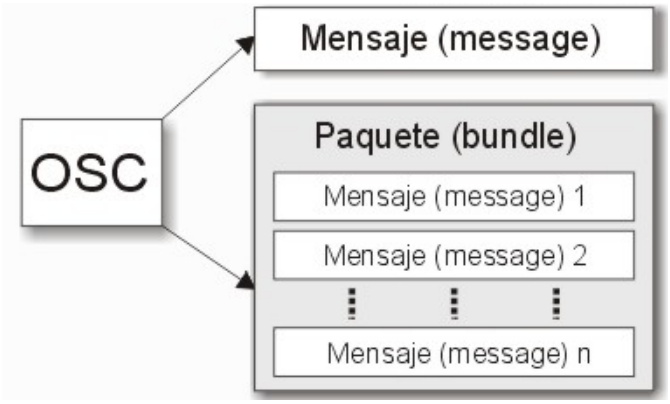


Figura D.1: Mensaje OSC.

La base de la comunicación OSC se encuentra en los mensajes. Los mensajes pueden ser de dos tipos: mensaje único (message) o paquete de mensajes (bundle). El paquete de mensajes es un contenedor que puede alojar uno o varios mensajes únicos.

D.3.1. Mensaje único (message)

Está compuesto por una cadena de bytes (un byte equivale a 8 bit) y se puede dividir en tres partes fundamentales:

Figura D.2: Mensaje único.

La característica común de todas las partes de un mensaje OSC (y como consecuencia del mensaje en si mismo) es que la cantidad de bytes que componen cada una de estas es un valor múltiplo de 4. Esto da la posibilidad de controlar y mantener el alineamiento en la lectura de los distintos datos.

D.3.2. Dirección (address)

La primera parte de un mensaje OSC es la dirección (address) y consiste en una cadena que inicia con el símbolo "/". Se asemeja a un sistema tipo URL de Internet permitiendo la navegación en una estructura jerárquica (o árbol). Generalmente la dirección se usa para identificar el mensaje (como si fuera un código de comando).

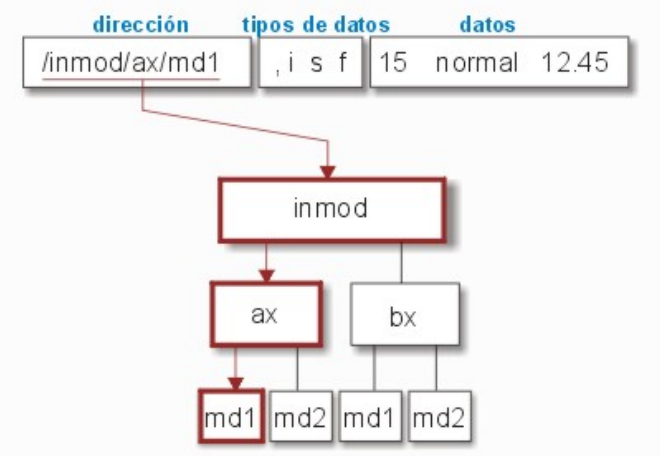


Figura D.3: Mensaje dirección.

En la figura D.3 podemos observar un ejemplo de dirección que permite de alcanzar un hipotético modulo interno "md1.ª través de una estructura tipo árbol invertido.

Ejemplos de direcciones son:

- /status
- /test/light
- /adc/input/3

D.3.3. Cadena de identificación de los datos transportados (tag types)

La segunda parte de un mensaje OSC es una cadena que empieza con el símbolo "," y que está compuesta por letras que sirven para identificar los tipo de datos que el mensaje transporta. Cada letra representa un tipo de dato (cadenas, números enteros, números con coma flotante, etc.

Figura D.4: Cadena de identificación de datos.

En la primera versión del protocolo (la versión 1.0), los tipo de datos posibles eran pocos: números enteros, números con coma flotante, cadenas de caracteres y por último matrices de byte . En la versión sucesiva (la 1.1), se agregaron otros tipos de datos aunque si los más importantes y usados son tres: cadenas de caracteres, números enteros y números con coma flotante.

D.3.4. Datos

La tercera parte del mensaje son los datos, uno después de otro y sin ningún byte de separación entre ellos. Los números generalmente ocupan 4 bytes (con alguna excepción de 8 byte) mientras las cadenas de caracteres y las matrices de bytes pueden tener un largo variable pero siempre un valor múltiplo de 4.

D.3.5. Las cadenas de caracteres OSC

Son secuencias de caracteres ASCII. Se identifica con la letra “s” minúscula del campo Type Tag . El dato tiene que respetar dos condiciones: la primera es que el final de la cadena debe haber por lo menos un byte con valor 0 (/0) indicando el final de ella, la segunda es que el largo de la cadena completa en byte debe ser un múltiplo de 4 (por ejemplo 4,8,12,16,20 bytes, etc.).

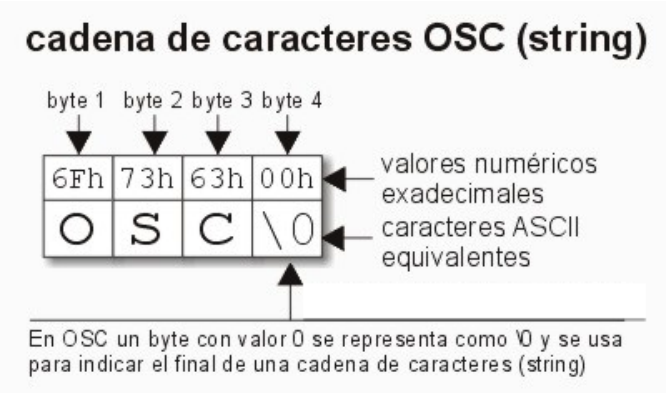


Figura D.5: Cadena de identificación de datos.

D.3.6. Blob (Matriz de bytes)

El tipo de dato Blob permite di transportar una matriz de bytes de cualquier dimensión. Está constituido por dos partes. La primera parte es un número entero que ocupa 4 bytes (int32) y que indica la dimensión de la matriz . La segunda parte es la matriz de byte propiamente dicha. Si la cantidad de elementos de la matriz no es múltiplo de 4 se deben agregar al final de la matriz unos bytes con contenido 0 para alcanzar este valor múltiplo. Los byte agregados no afectan el número de la dimensión de la matriz. El tipo blob es útil cuando se deben transmitir un grupo de bytes que pueden contener como dato 0 como por ejemplo un diseño bitmap y que no sería posible hacerlo con una dato del tipo string.

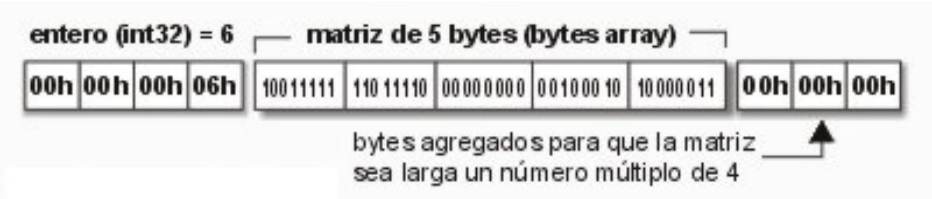


Figura D.6: Matriz de bytes.

En la figura D.6 podemos observar un ejemplo de dato Blob constituido por una matriz de 6 bytes. Los primeros 4 bytes representan un número entero (Int32) que indica la cantidad de byte que componen la matriz, en este caso 6. Los siguientes bytes representan la matriz. Es interesante observar que el tercer byte de la matriz tiene como contenido 0. Si hubiéramos usado un dato del tipo cadena (string) para transportar esta matriz, este byte habría creado un conflicto interpretado como el final de la cadena. Los dos últimos byte con valor 0 sirven para "redondear" el largo del dato a un múltiplo de 4.

D.3.7. Integer (número entero con signo)

El tipo de dato integer ocupa 4 bytes y es con signo (complemento a dos). El dato se transmite con orden "big endian" es decir, el byte más significativo primero. En la figura D.7 se encuentra representado un hipotético mensaje con un dato numérico del tipo integer.

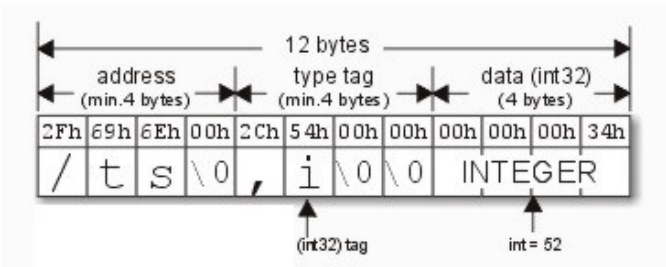


Figura D.7: Mensaje con un dato numérico del tipo integer.

D.3.8. Float (número con coma flotante)

El tipo de dato float ocupa 4 bytes y obedece el estandard IEEE 754. El dato se transmite con orden "big endian" es decir, el byte más significativo primero. En la figura D.8 se encuentra representado un hipotético mensaje con un dato numérico del tipo float.

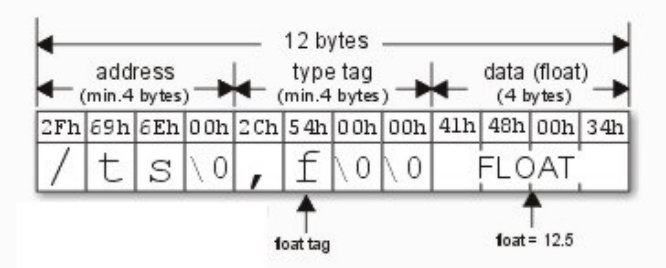


Figura D.8: Mensaje con un dato numérico del tipo float.

D.3.9. Mensajes sin datos

El protocolo OSC especifica distintos tipos de mensajes sin datos: False (F), True (T), Nill (N) e Impulse (I). Los mensajes sin datos son útiles cuando se debe mandar una información del tipo on /off (por ejemplo el estado de un sensor) o un impulso. Las ventajas de estos mensajes son: una descodificación muy simple y mensajes muy cortos, útiles en el caso de canales de transmisión lentos o cuando la potencia de cálculo es reducida (por ejemplo en dispositivos con microcontroladores). Otro uso de los mensajes sin datos es para indicar al sistema que un específico dispositivo es operativo. Este, puede ser programado para transmitir periódicamente un mensaje Nill indicando su presencia.

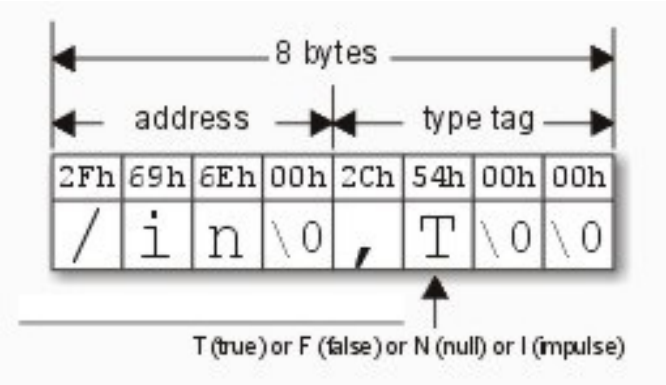


Figura D.9: Mensaje sin datos.

Apéndice E

Comandos OSC

Se ha incluido como apéndice las llamadas que puede realizar Processing al secuenciador Ableton Live! para que este reciba ordenes, así como los envíos de ordenes que Ableton Live! requiere de Processing

E.1. LLamadas OSC de Processing a Ableton Live!

/live/tempo: Solicitud del tempo actual, responde con /live/tempo (float tempo)

/live/tempo (float tempo): Fija el tempo, responde con /live/tempo (float tempo)

/live/time: Solicitud del tiempo actual, responde con /live/time (float time)

/live/time (float time): Fija el tiempo, responde con /live/time (float time)

/live/overdub (int on/off): Activar/Desactivar overdub

/live/state: Devuelve el tempo actual y el estado del overdub

/live/undo: Pide la canción para deshacer la última acción

/live/redo: Pide la canción para rehacer la última acción

/live/next/cue: Salta al punto de señal siguiente

/live/prev/cue: Salta al punto de señal previo

/live/play: Comienza a reproducirse la canción

/live/play/continue Continua reproduciendo la canción desde el punto actual

/live/play/selection: Reproduce la selección actual

/live/play/clip (int track, int clip): Lanza el clip número clip del track número track

/live/play/scene (int scene): Lanza la escena número scene

/live/stop: Detiene la canción actual

/live/stop/clip (int track, int clip): Detiene el clip número clip en el track número track

/live/stop/track (int track): Detiene el track número track

/live/scenes blank or ('query'): Devuelve el número total de escenas

/live/tracks blank or ('query'): Devuelve el número total de tracks

/live/scene: Devuelve el índice de escena seleccionado

/live/scene (int scene): Selecciona la escena con un índice de escena

/live/name/track: Devuelve una serie de todos los nombres de las pistas

/live/name/clip: Devuelve una serie de todos los nombres de los clips

/live/arm (int track): Carga el estado desde el track número track

/live/arm (int track, int armed/disarmed): Carga/Descarga el track número track

/live/mute (int track): Obtener el estado de silencio para el track número track

/live/mute (int track, int mute/unmute): Silencia/Activa el sonido del track número track

/live/solo (int track): Obtiene el estado "solo"para el track número track

/live/solo (int track, int solo/unsolo): Solo/Desactiva solo del track número track

/live/volume (int track, float volume(0.0 to 1.0)): Fija el número del track volumen a volumen

/live/pan (int track, float pan(-1.0 to 1.0)): Fija el número del track panorámica a panorámica

/live/send (int track): Devuelve una lista de todos los envíos y los valores del track

/live/pitch (int track, int clip): Devuelve el valor del pitch de los clips que hay en cada track

/live/return/mute (int track): Obtiene el estado de mute desde el track de retorno número track

/live/return/solo (int track): Obtiene el estado de solo desde el track de retorno número track

/live/return/volume (int track): Devuelve el volumen actual de la pista

/live/return/pan (int track): Devuelve la panorámica del track de retorno número del track

/live/return/send (int track): Devuelve una lista de todos los envíos y los valores en la pista

/live/master/volume (int track): Devuelve el volumen actual del track master

/live/master/pan (int track, float pan(-1.0 to 1.0)): Fija la panorámica del track master

/live/track/jump (int track, float beats): Saltos en pista actualmente en ejecución

/live/track/info (int track): Devuelve clip de Estado de la ranura para todos los clips en la pista

/live/track/view (int track): Selecciona un track para obtener información

/live/return/view (int track): Selecciona un track de retorno para obtener información

/live/master/view: Selecciona el track master

/live/return/device/view (int track, int device): Selecciona un dispositivo de retorno del track

/live/master/device/view (int device): Selecciona dispositivos del track

/live/clip/view (int track, int clip): Selecciona un clip del track para obtener información

/live/detail/view (int): Cambia las vistas de detalle [0 = clip, 1 = track]

/live/clip/info (int track, int clip): Obtiene el estado de un solo clip

/live/devicelist (int track): Devuelve una lista de todos los dispositivos y los nombres de track

/live/clip/loopstart (int track, int clip) Obtiene el loopstart de un clip de un track

/live/clip/loopend (int track, int clip): Obtiene el loopend de un clip de un track

/live/clip/loopstate (int track, int clip): Obtiene el estado de loop para un clip en track

/live/clip/loopstate (int track, int clip, int on/off): Fija el estado de loop para un clip en track

/live/clip/warping (int track, int clip): Obtiene el estado de deformación del clip

/live/clip/warping (int track, int clip, int state): Establece el estado de deformación del clip

/live/clip/signature (int track, int clip): Obtiene el compás de un clip devuelve 4x4 por ejemplo

/live/clip/signature (int track, int clip, int denom, int num): Fija el compás de un clip

/live/master/crossfader: Obtiene la posición actual del crossfader

/live/master/crossfader (float position): Fija la posición del crossfader

E.2. Envíos OSC de Ableton Live! a Processing

Las siguientes funciones devuelven automáticamente un valor cuando en Ableton Live! se produzca un cambio que afecte a Processing. La información es similar al apartado anterior salvo que ahora es Ableton el que se comunica con processing para indicarle ciertos parámetros.

/live/play (2 = playing, 1 = stopped)

/live/track/info

/live/clip/info

/live/clip/position (int track) (int clip) (float position) (float length)

/live/name/return

/live/name/track

/live/name/clip (returns on colour and name changes)

/live/arm

/live/mute

/live/solo

/live/volume

/live/pan

/live/send

/live/master/volume

/live/master/pan

/live/master/crossfader

/live/return/mute

/live/return/solo

/live/return/volume

/live/return/pan

/live/return/send

/live/overdub

/live/tempo

/live/scene

/live/track

/live/master/meter (int 0=left, 1=right) (float value)

/live/return/meter (int track) (int 0=left, 1=right) (float value)

/live/track/meter (int track) (int 0=left, 1=right) (float value)

/live/device/param (int track) (int device) (int param) (int value) (str name)

/live/return/device/param (int track) (int device) (int param) (int value) (str name)

/live/master/device/param (int device) (int param) (int value) (str name)

/live/device/selected (int track) (int deviceid)

/live/return/device/selected (int track) (int device)

/live/master/device/selected (int device)

Apéndice F

Diccionario de términos y conceptos

El objetivo de la creación de un diccionario de datos es dar precisión sobre los datos que se manejan en el sistema, evitando así malas interpretaciones o ambigüedades, a continuación se exponen los conceptos más relevantes que se utilizan en el mundo del audio y MIDI.

- **ADAT**

  Interface de conexión por fibra óptica para transmisión de audio digital entre equipos, desarrollado por Alesis y bien aceptada desde su inicio, actualmente es utilizada también por fabricantes distintos a la marca que lo creó. El término ADAT terminó convirtiéndose en un protocolo y se reconoce más así, que por la original máquina multi-pista Alesis que utilizaba cintas VHS para grabación.

  La interface utiliza conectores Toslink similares a los que usa también el formato S/PDIF (por cierto incompatible). La transmisión puede transportar hasta 8 canales de audio digital no comprimidos a 48 kHz con 24 bits de resolución.

- **AES/EBU**

  Formato profesional diseñado para contener y transmitir datos de dos señales de audio digital a la vez sin compresión PCM. Por su morfología es uno de los formatos más populares utilizándose en el ámbito profesional transportando y trabajando con tipos de señales para equipos DAT, CD, convertidores A/D-D/A en consolas de mezcla, máquinas de video digital, interfaces surround, procesadores de señal digitales, amplificadores, etc. La transmisión consiste en una señal con información de tiempo para la sincronía y es capaz de transportar datos a distintas frecuencias de muestreo. Se cablea con conectores XLR de 3 pines (par trenzado) a 110 ohms o coaxial con conector BNC a 75 ohms, el voltaje de la señal puede ser entre 3 y 10 Volts con una resolución máxima de 24 bits.

- **AIFF**

  El formato estándar para los documentos de audio en el entorno Macintosh. Sus siglas corresponden

a las palabras Audio Interchange File Format. Permite almacenar datos de audio en una amplia variedad de resoluciones, compresiones y formatos. Su equivalente en Windows es el Wav.

- **Amplitud**

  Magnitud de la onda sonora o señal electrónica de audio medida en decibeles

- **Ancho de Banda**

  Rango de frecuencia que un dispositivo o transductor es capaz de captar, reproducir o transmitir.

- **Balance**

  Nivel de volumen relativo entre los canales derecho e izquierdo de una señal estéreo.

- **Balún**

  Convertidor de conexión coaxial (BNC) a XLR con par trenzado .

- **Band, Banda**

  Porción definida de el espectro de frecuencias

- **Bidireccional**

  Patrón de polaridad referente a micrófonos que capta por el lado frontal y trasero del mismo con mínima captación en sus lados.

- **BNC**

  Conector para uso con cable coaxial, creado por los Ingenieros Paul Neill de Bell Labs y el ingeniero de Amphenol Carl Concelman, prácticamente amo y señor de la conexión para video (75 ohms). Hace bastante tiempo que en audio se ha utilizado para la conexión de antenas en sistemas inalámbricos a 50 ohms, sincronía y en transmisión de audio digital para el formato MADI (75 ohms). Actualmente ya ganó terreno en el mundo broadcast por la conexión digital AES/EBU vía coaxial (BNC 75 ohms Desbalanceado) convertido desde/ hacia XLR (110 ohms balanceado) con conectores Balún, resultando más práctica y eficiente la interconexión entre bahías de parcheo digitales, Generadores de House Sync, routers y matrices que en general utilizan BNC como conexión de audio digital.

- **Board**

  Consola o mesa de mezcla

- **BPM**

  Siglas que corresponden a beats per minute. Es decir, el número de golpes rítmicos, beats, que se producen en un tema por minuto. También se conoce como tempo.

- **Bypass**

  Control para la interrupción de un proceso o funcionamiento en dispositivos de audio permitiendo circular la señal sin afectarla.

- **Chorus**

  Un efecto de desdoblamiento que se encuentra normalmente en los sintetizadores y samplers y que permite que un único sonido parezca un grupo de sonidos. La señal inicial se divide y aparece con una afinación ligeramente modificada en relación al original, o con un ligero retardo. Este nivel de afinación y de retardo normalmente puede controlarse mediante un oscilador de baja frecuencia (LFO).

- **Clipping**

  Distorsión audible que ocurre cuando el nivel de señal de un circuito o dispositivo en particular ha sido exedido.

- **Comb Filtering**

  Filtro de peine

- **Compás**

  Conjunto de cuatro tiempos o beats, cuando el compás es de 4/4, el más utilizado en la música dance.

- **Compresor**

  Proceso que reduce el rango dinámico de una señal de acuerdo a una proporción (Ratio) ajustada, por ejemplo, en una relación 2:1 donde el primer número es la cantidad de decibeles que entra y el segundo la cantidad que sale, si el primer valor es 6dB el resultado será 3dB, Otros parámetros del compresor son Threshold /Umbral: Nivel ajustado donde la señal comienza a comprimirse, entre más bajo sea el valor la señal estará comprimida más tiempo. Attack /Ataque: Se refiere al tiempo en que comenzará a actuar el proceso cuando el umbral haya sido rebasado. Release /Relajamiento: tiempo en que el compresor tarda en restaurar la señal a su nivel normal.

- **Compuerta**

  Proceso que cierra la entrada de una señal, activándose cuando el nivel está por debajo de un umbral determinado. Algunos parámetros de las compuertas son: Attack /Ataque: control no muy vital pero útil, se usa para ajustar el tiempo en que la compuerta abrirá completamente después de que el nivel de señal haya excedido un umbral determinado. Ratio /Proporción: Este control permite mantener cierta proporción de la señal original aun cuando la compuerta esté totalmente cerrada y puede ser útil cuando se desea tener un poco de ruido de ambiente en el canal y a la vez reducir la intensidad del mismo. Hold: Ajuste que permite mantener un tiempo mínimo entre el Release ó Liberación cuando el umbral ha sido rebasado y cuando la compuerta comienza a cerrarse, este control ayuda a parar la compuerta cuando la dinámica o amplitud de la señal cambia constantemente. Side chain Filter: Control que fija un rango de frecuencias donde la compuerta responderá mayormente.

- **Cuantización**

  Ver Rango Dinámico, División en intervalos con un valor binario asignado de la amplitud de una señal análoga, así como en la frecuencia de muestreo, entre más valores de cuantización haya más

alta será la resolución, y calidad del sonido. Los parámetros o formatos varían de 8 bits en baja resolución hasta los 20 y 24 bits en alta resolución.

- **Cue**

  Término utilizado para indicar el inicio de alguna acción técnica o instrucción de entrada para grabación, transmisión o espectáculo. Por ejemplo en TV (en radio, proceso similar), el çue"de inicio de transmisión o grabación es la última señal seguida de un conteo regresivo (10 a 2 ó 10 a uno y cue) dada por el floor manager y/o por el director de cámaras mediante micrófono de comunicación para el talento, audiencia en foro y staff en general. En consolas de mezcla principalmente broadcast, Cue es un control de Soleo ó PFL para el operador, aunque el concepto se amplía pues no sólo se verifica la calidad y nivel de señal, sino también se corroboran y monitorean a modo de prevención las fuentes ya sea de audios provenientes de máquinas de video, servidores, señales remotas ó telefónicas, proporcionando también a solicitud u órdenes de terceros el cue (previo) de alguna fuente a distintos destinos como, IFB (apuntador), Fold Back ó cabina de producción.

- **DAC**

  Convertidor Digital a Análogo

- **DCA**

  Amplificador controlado digitalmente. Amplificador cuya ganancia es controlada remotamente por un control de señal digital.

- **De-emphasis**

  Ecualización que consiste en controlar y disminuir alta frecuencia en un proceso de reducción de ruido en segundo orden.

- **De-esser**

  Compresor que reduce sibilancia.

- **Distorsión**

  Diferencia indeseada entre la versión original y final de una señal.

- **Drop Frame**

  Código de tiempo SMPTE usado para compensar error de tiempo acumulado en video a color, Drop Frame salta dos cuadros al comienzo de cada minuto (excepto cada 10 minutos) tal como cuenta los cuadros el video a color. El resultado es que los valores del código de tiempo SMPTE emparejan el tiempo real transcurrido, puesto que el video a color corre más lento (29,97 cuadros por segundo) que el de blanco y negro (30 fps), Drop Frame eis sólo requerido en programas a color en donde los números del código de tiempo SMPTE deben emparejar precisamente el tiempo actual transcurrido.

- **Drop Out**

  Pérdida total de la recepción de señal, debida a interferencias multipath (Reflexión de ondas de radio que causan fluctuación o más de una radio frecuencia llagando a un canal de recepción).

- **Drum box**

  Aparato electrónico con el cual se puede conseguir una batería o una percusión artificial.

- **DSP**

  Digital Signal Proccesing

- **Expander**

  Proceso dinámico que expande el rango dinámico de una señal

- **EQ**

  Ecualización, proceso de señal que busca, atenúa y arrincona frecuencias o banda de frecuencias de una señal o sistema de sonido.

- **Fader** Mando de control para incrementar o disminuir volumen, equalización o balance y panorama de sonido. El fader puede ser deslizante o giratorio.

- **Fase**

  Relación de tiempo entre dos ó más sonidos llegando a un micrófono ó señales a un circuito. Cuando la relación de tiempo es coincidente, los sonidos o señales están en fase y sus amplitudes se suman, cuando la relación de tiempo es distinta en los sonidos ó la polaridad de una de las señales está invertida, la amplitud es substractiva. En refuerzo sonoro este concepto es crucial en lo que se refiere a gabinetes, arreglos o sistemas multiamplificados donde debe asegurarse que la relación de tiempo entre frecuencias ó vías sea lo más coincidente posible para reproducir la respuesta en frecuencia sin alteración de fase.

- **Hi End**

  El rango agudo-alto del espectro de frecuencias audible.

- **Hi Pass Filter**

  Filtro pasa altos.

- **Inverse Square Law**

  Ley del cuadrado Inverso, Situación acústica en la cual, el nivel de sonido sufre un cambio inversamente proporcional al cuadrado de la distancia donde surge la fuente de sonora.

- **LFO**

  Un oscilador de baja frecuencia que se utiliza para modificar la frecuencia o la amplitud de un sonido.

- **Limitador**

  Proceso dinámico que limita el nivel máximo de una señal.

- **Loop**

  Fragmento musical que, fijando un punto de inicio y otro de finalización, se repite sin pausa el tiempo que determinemos. Puedes encontrar infinidad de loops en prácticamente cualquier manifestación musical bailable.

- **MIDI**

  Musical Instruments Digital Interface, Protocolo de información desarrollado a principios de los ochentas del siglo pasado por sintetizadores, y fabricantes de instrumentos electrónicos para permitir a dispositivos la comunicación de datos de ejecución musical entre uno y otro.

- **MP3**

  Formato de compresión de audio que permite reducir la capacidad en megabytes de un fichero de sonido sin que se pierda calidad.

- **MTC**

  Midi Time code, Código de tiempo digitalizado dentro del formato MIDI expresado también en Horas, minutos, segundos y frames como el VITC y el LTC.

- **Multipistas**

  En la tecnología de grabación tradicional, la habilidad de superponer múltiples señales de audio diferentes a la vez. En los programas MIDI, la habilidad de superponer múltiples flujos de información MIDI.

- **NC**

  Serie de curvas utilizadas para tazar el ruido de fondo de espacios en términos de percepción humana. Las curvas NC permiten mediciones más altas de presión sonora en ruido de baja frecuencia que en ruido de alta frecuencia ya que el oido humano es menos sensible al sonido grave comparado con la sensibilidad al agudo, algunas clasificaciones son como siguen: Estudios de Grabación-NC 10, Cabinas de doblaje-NC 20, Teatros de alta calidad-NC 25, Máximo permitido en teatros-NC 30.

- **Octava**

  Una relación de frecuencia de 2:1. Una distancia (intervalo) musical de 12 semitonos.

- **Off-Axis**

  Ángulo fuera de eje. Punto donde el nivel de una bocina cae 6dB comparado con el nivel de salida que cubre On-Axis (En eje). En microfonía se aplica un concepto igual donde Off-Axis describe el punto donde alguna fuente (voz o instrumento) se ubica fuera del patrón polar o patrón de captación (Pick-Up Pattern) del transductor.

- **Oscilador**

  Un aparato electrónico capaz de generar una forma de onda recurrente, o un proceso digital utilizado por un sintetizador para generar lo mismo.

- **PAN**

  Control que sirve para desplazar señales de audio a un punto en la imágen estéreo o envolvente de un sistema de sonido, creando el efecto de prescedencia con sonidos proviniendo desde un origen específico (Izquierdo, Derecho, Central, Surround Izq., Surround Der. LFE, etc).

- **Patches**

  Conocidos también como programas, timbres o voces. El nombre utilizado para los sonidos que puede generar un aparato MIDI.

- **Pitch Bend**

  Un controlador MIDI que permite variar la afinación de un sonido.

- **Rango Dinámico**

  Rango de intensidad en la señal respecto al ruido de fondo de un dispositivo medido en decibeles. Dividido en el mundo análogo en: relación señal a ruido, nivel nominal o programa, y headroom o tolerancia que comienza a partir de 0 VU o señal nominal, y donde el límite se establece como nivel pico (clipping, peaking). En el dominio digital este concepto se conoce como Cuantización, Signal to noise-Error (Relación señal a error) y representa el grado de resolución con el que se codificó una señal, medida igualmente en decibeles de escala completa dBFS donde cero representa el nivel máximo o pico. La calidad de resolución está comprendida en bits, por ejemplo la precisión del CD es de 16 bits y en formatos profesionales se pueden tener hasta 24 bits que equivalen a 145.8 db (donde cada bit representa 6 db multiplicado por el número de bits con la suma de 1.8 al resultado final)

- **RCA**

  Compañía que desarrolló el conector Phone Jack o Pin Jack de formato desbalanceado, es utilizado para audio en nivel consumidor, también utilizado para transmitir señal digital S/PDIF y video

- **RF**

  Radio Frecuencia, Frecuencia o tipo de oscilación referida a circuitos eléctricos y radiación electromagnética, con un rango de 3 Hz a 300 GHz que corresponde también a las señales eléctricas de corriente alterna para detectar y producir ondas de radio.

- **Ruido Blanco**

  Ruido que contiene todas las frecuencias o densidad de poder espectral plana, con misma energía por cada incremento de frecuencia, su sonido es brillante dado que las frecuencias agudas contienen mucha energía, ej. ruido del televisor cuando no está sintonizando.

- **Ruido Rosa**

  Ruido que contiene todas las frecuencias o densidad de poder espectral con energía inversamente proporcional a cada incremento de frecuencia, dado que contiene la misma proporción de energía por octava de frecuencia éste ruido es muy utilizado como fuente de referencia para pruebas de sistemas de sonido con analizadores de tiempo real. (Suena como cascada o catarata).

- **Sample**

  Extracto, muestra sonora (puede ser desde un solo sonido hasta un fragmento de una canción) que se utiliza posteriormente interpolándola en un ritmo determinado.

- **Sampler**

  Un aparato electrónico que puede grabar, modificar y reproducir información de audio digital bajo el control de un flujo de datos MIDI.

- **Secuenciador**

  Programa MIDI o, raramente, un aparato de hardware que puede grabar, editar y reproducir una secuencia de datos MIDI.

- **Timbre**

  La propiedad de un sonido que lo distingue de todos los demás. Color del sonido.

- **Tremolo**

  Una alternancia rápida de dos tonos. Normalmente separados por un intervalo de tercera. En un sintetizador, este efecto puede controlarse normalmente mediante la rueda de modulación o la cantidad de modulación.

- **Velocity**

  Medición de la fuerza con que se pulsa la tecla de un controlador. Se utiliza para determinar las características de volumen de una nota.

- **Waveform**

  Variación en amplitud de una cantidad contra tiempo, tal como la presión sonora contra tiempo o voltaje contra tiempo, etc.

- **XLR**

  Conector de audio para líneas balanceadas (formato profesional), utilizado para micrófoneo, dispositivos de audio y conexión digital AES/EBU, fue desarrollado por la compañía Cannon, las siglas provienen de la serie o modelo llamada X, la letra L, viene de latch, seguro o pestillo agregado y finalmente la R, que proviene de resilient rubber compound una mejora en el material del conector basado en caucho resistente entre los pines de contacto.

Apéndice G

Fiduciales

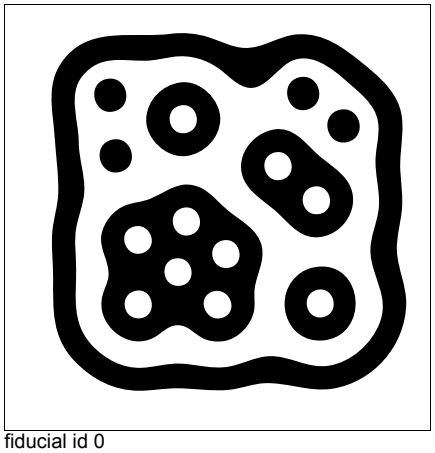
A continuación se detalla en qué consisten los fiduciales de Reactivision lo que permitirá comprender mejor las limitaciones que estos conllevan.

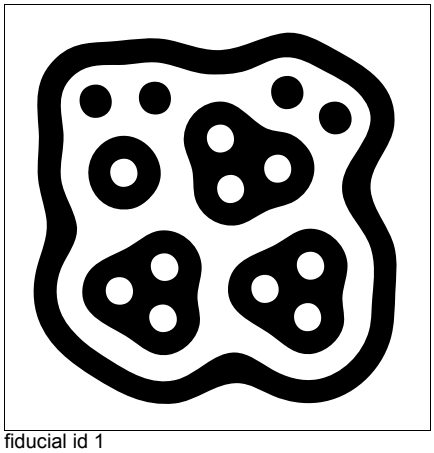
En primer lugar se estudió en detalle los fiduciales estándar de Reactivision para poder entender las limitaciones que estos suponen. Los fiduciales son marcadores impresos que se colocan en la base de las piezas de metacrilato para que puedan ser reconocidos. El diseño de los fiduciales suele ser sencillo debido a la necesidad de que un ordenador pueda detectarlos de manera rápida. Por otra parte esta sencillez permite que pequeñas variaciones en el diseño puedan generar un gran número de fiduciales.

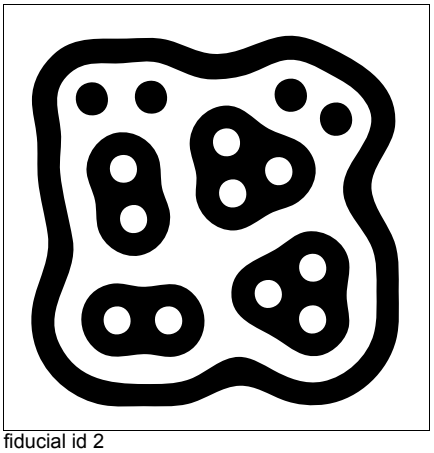
Los fiduciales utilizados son siempre en blanco y negro, ya que como se ha indicado se utiliza luz infrarroja para iluminar el prototipo y si los fiduciales fueran de colores, no podrían ser reconocidos.

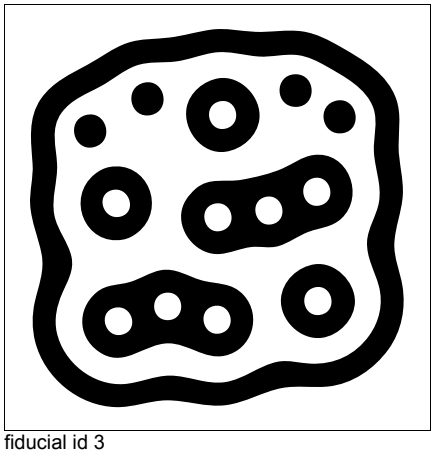
Los fiduciales que Reactivision interpreta tienen forma de ameba, ya que han sido diseñados para optimizar la distancia entre los contornos negros y blancos para que la cámara los detecte bien, usando círculos, que pueden ser concéntricos. Estos círculos proporcionan una orientación que puede ser usada en el procesado de la información. Este diseño en ameba obliga a que los fiduciales cuanto más pequeños sean tengan que tener una forma más o menos cuadrada, lo que limita el tipo de objeto a utilizar o condiciona la estética del este. A su vez para el correcto reconocimiento de los fiduciales, Reactivision obliga a que estos tengan unos tamaños mínimos. Si este tamaño es menor se producen errores y conflictos en la identificación de fiduciales.

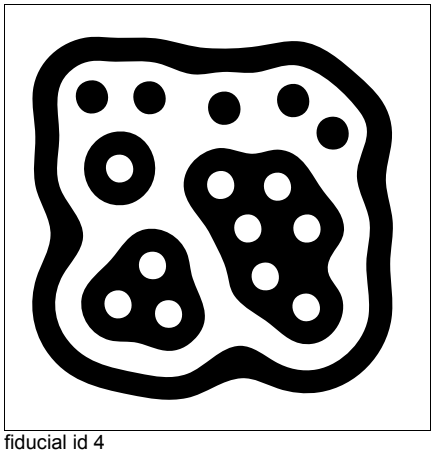
A continuación se incluyen los 179 fiduciales que trae la librería reacTIVision, para poder manipularlos a través de una cámara.

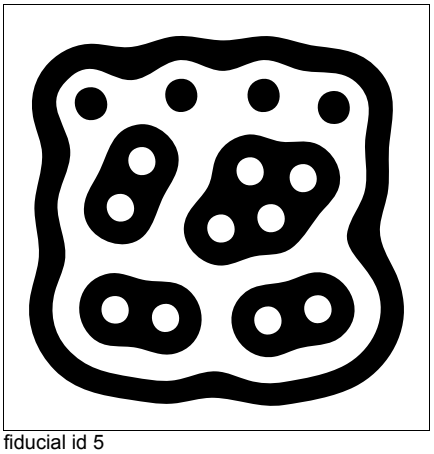

fiducial id 0

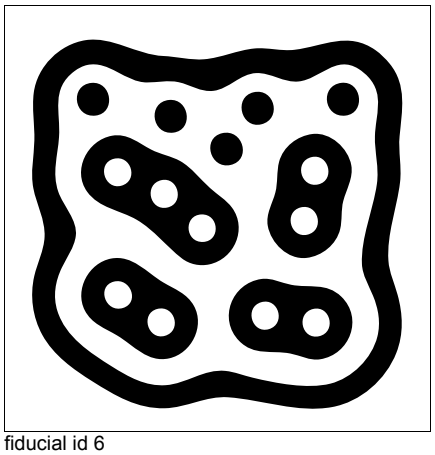

fiducial id 1

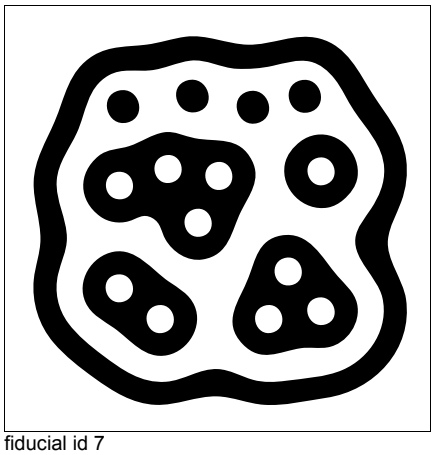

fiducial id 2

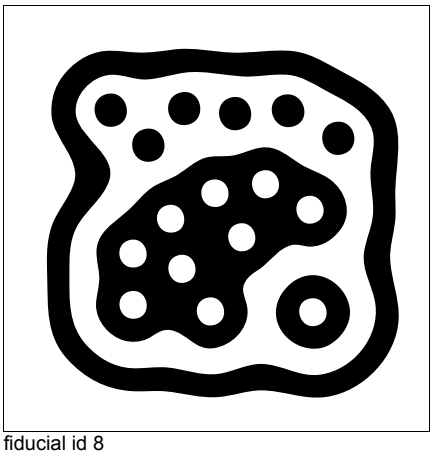

fiducial id 3

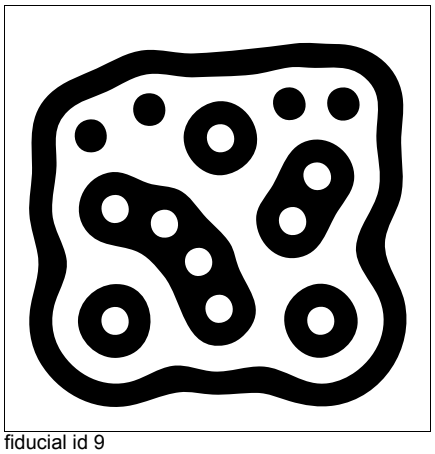

fiducial id 4

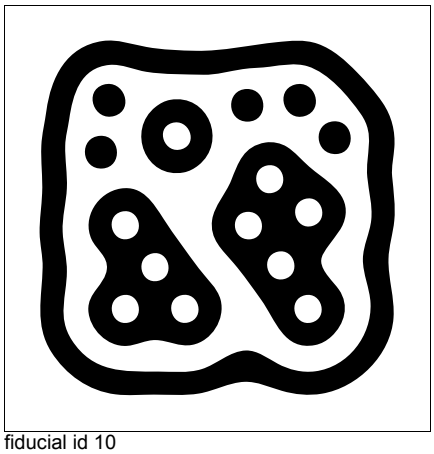

fiducial id 5

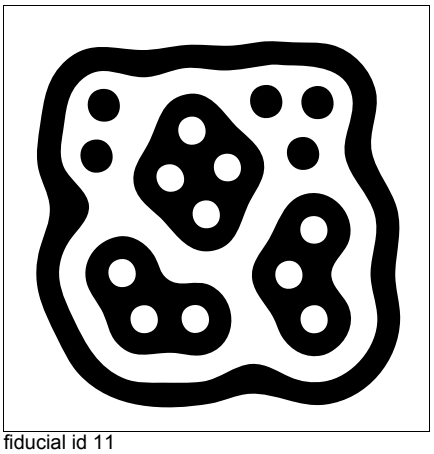

fiducial id 6

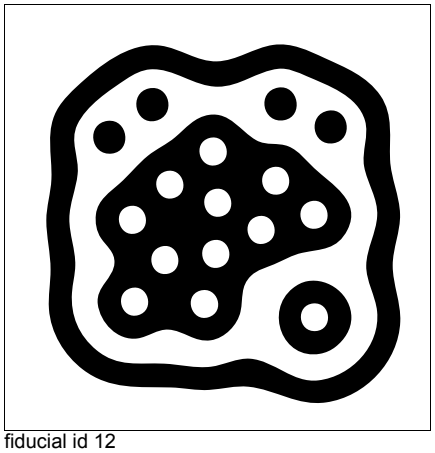

fiducial id 7

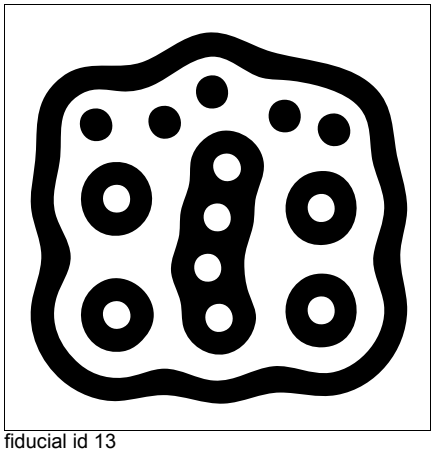

fiducial id 8

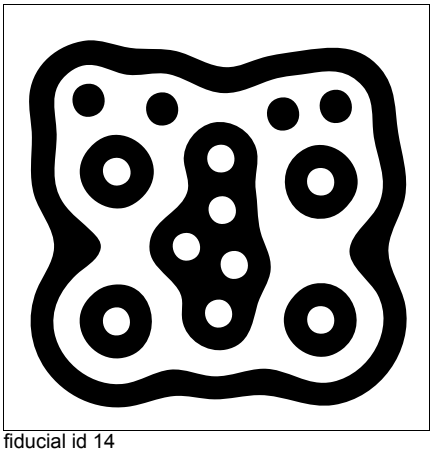

fiducial id 9

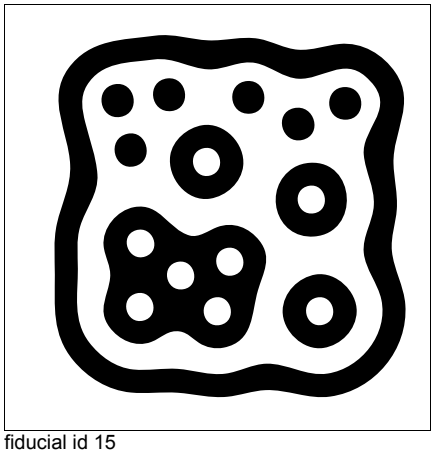

fiducial id 10

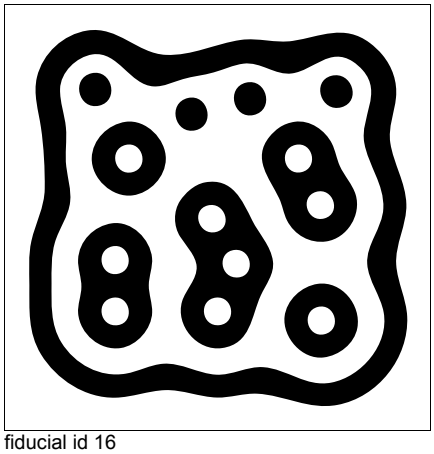

fiducial id 11

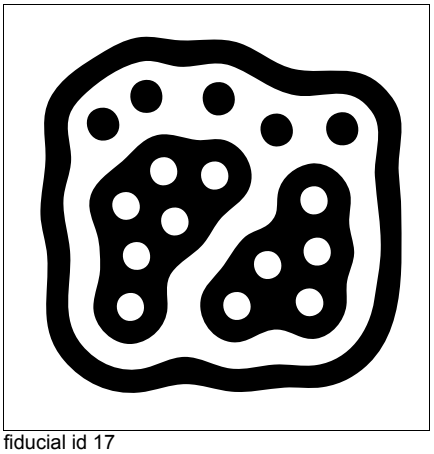

fiducial id 12

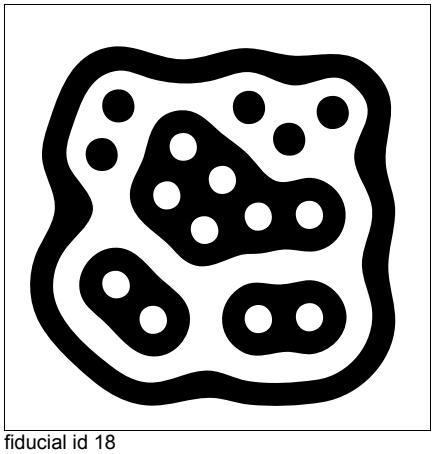

fiducial id 13

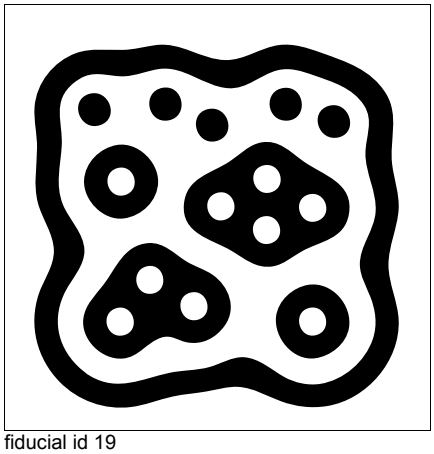

fiducial id 14

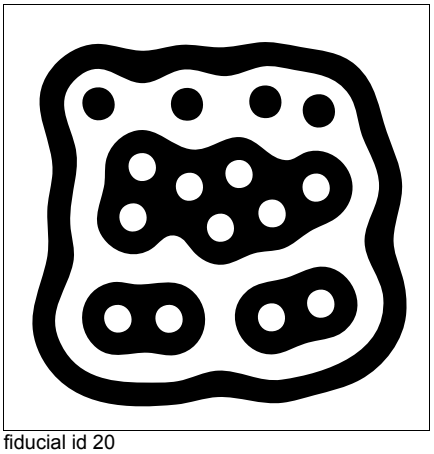

fiducial id 15

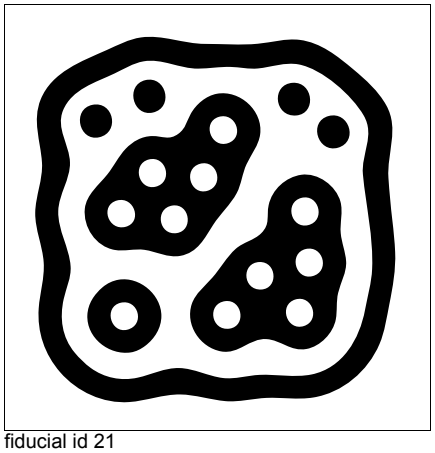

fiducial id 16

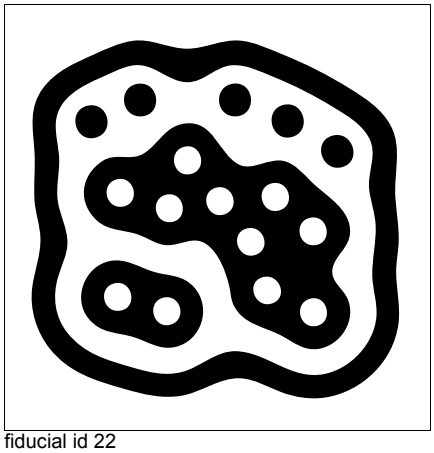

fiducial id 17

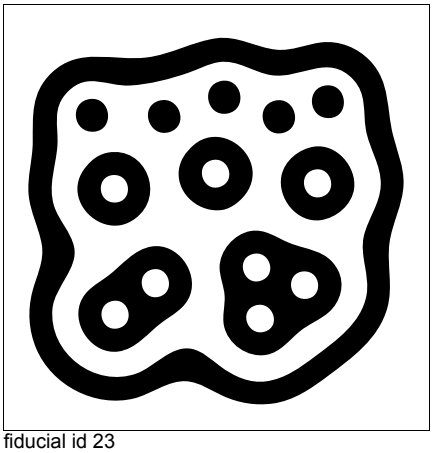

fiducial id 18

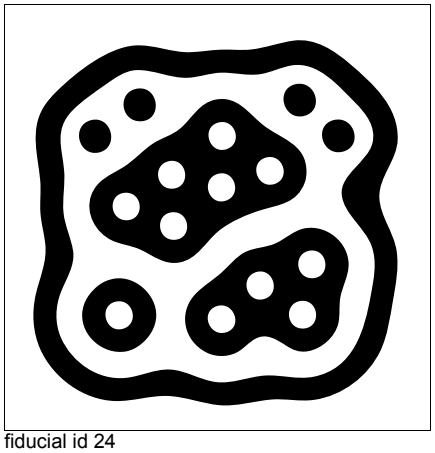

fiducial id 19

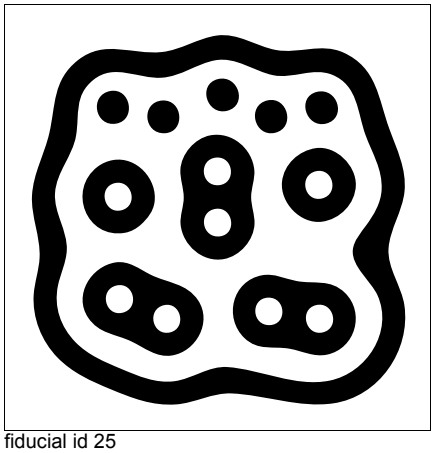

fiducial id 20

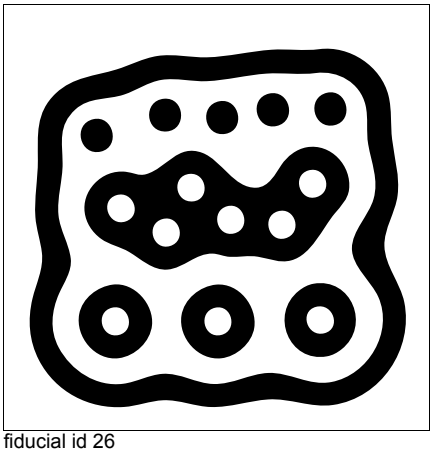

fiducial id 21

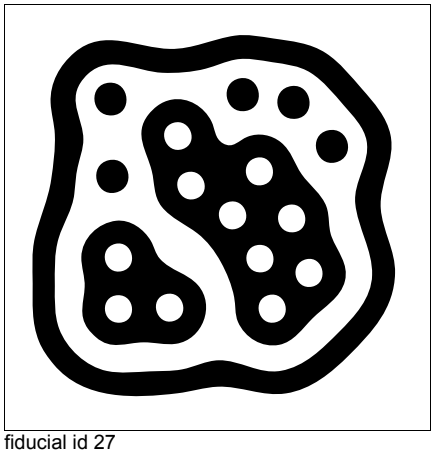

fiducial id 22

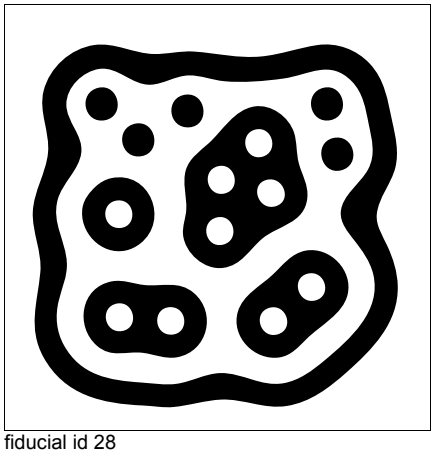

fiducial id 23

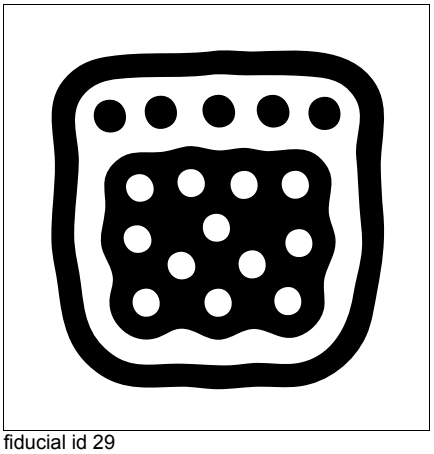

fiducial id 24

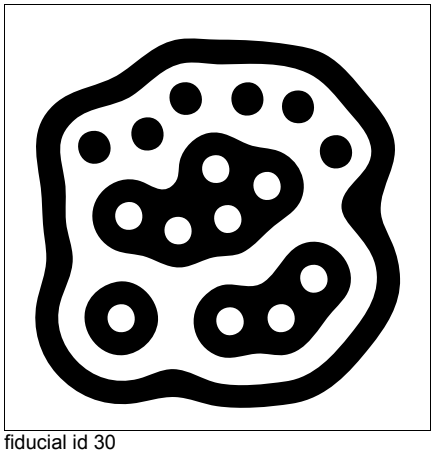

fiducial id 25

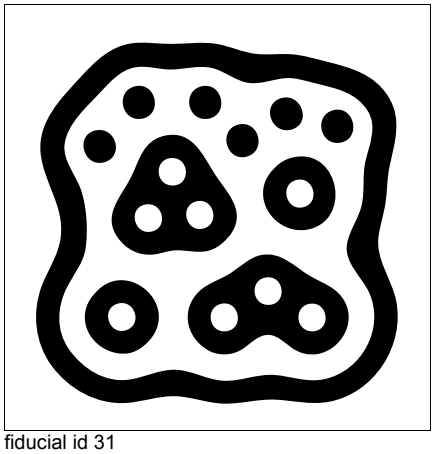

fiducial id 26

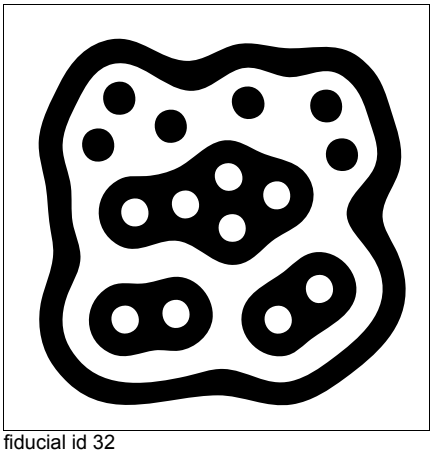

fiducial id 27

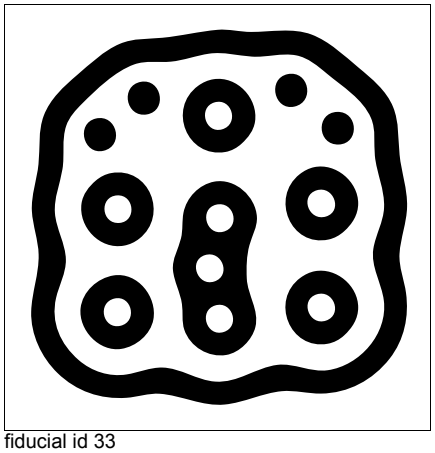

fiducial id 28

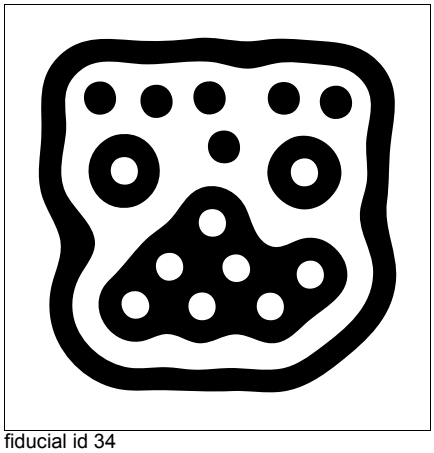

fiducial id 29

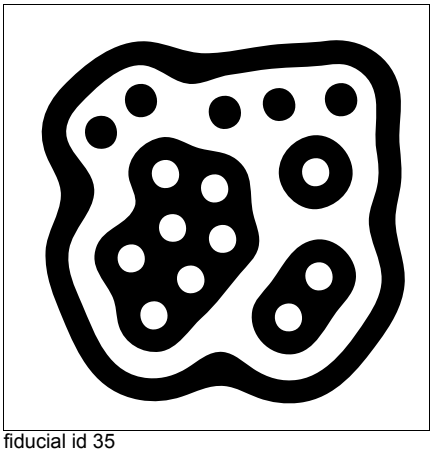

fiducial id 30

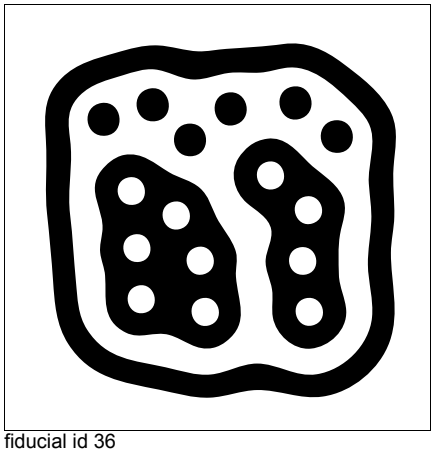

fiducial id 31

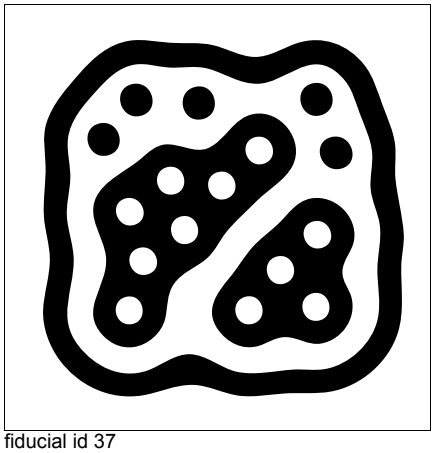

fiducial id 32

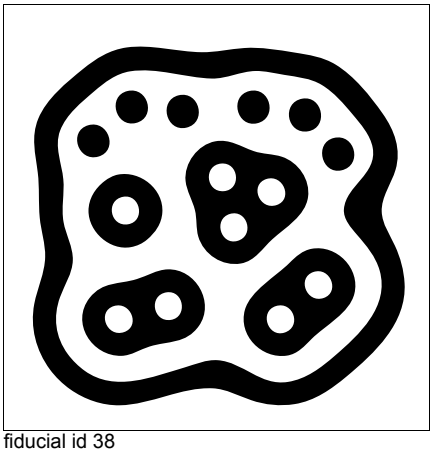

fiducial id 33

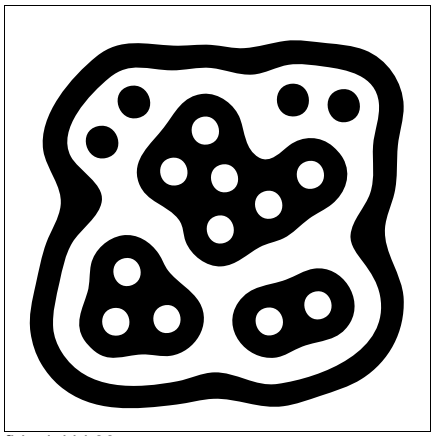

fiducial id 34

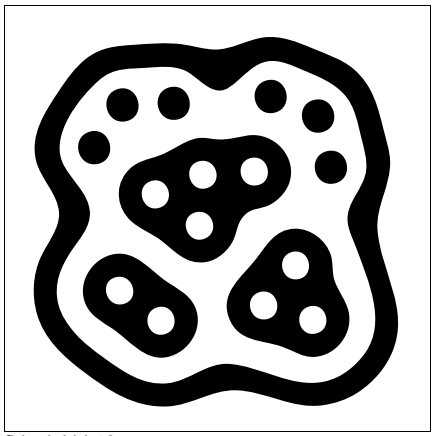

fiducial id 35

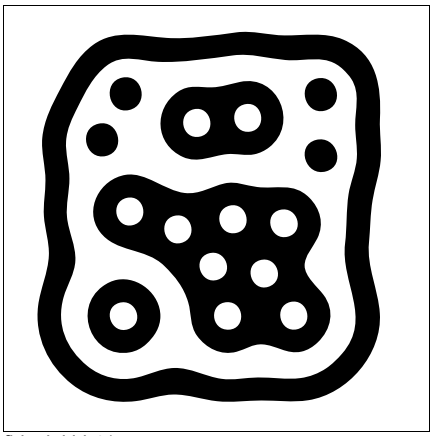

fiducial id 36

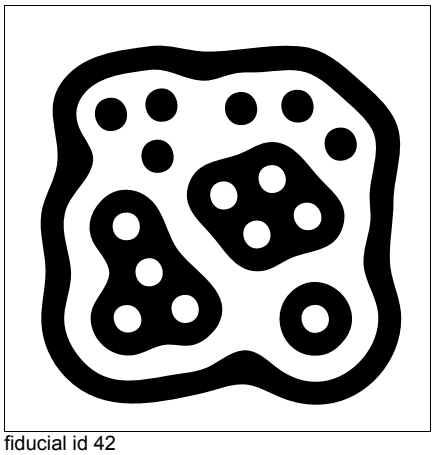

fiducial id 37

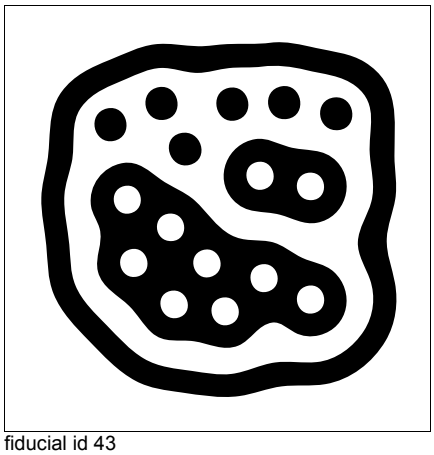

fiducial id 38

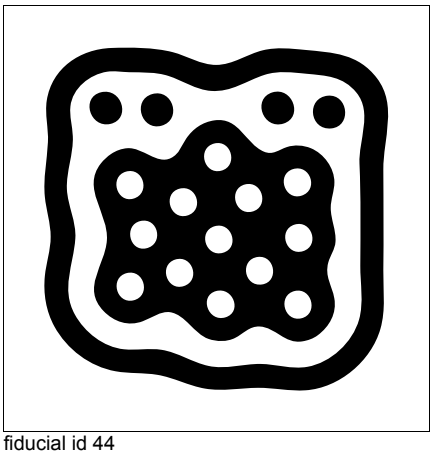

fiducial id 39

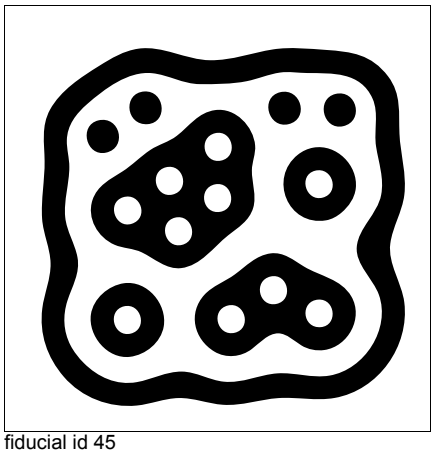

fiducial id 40

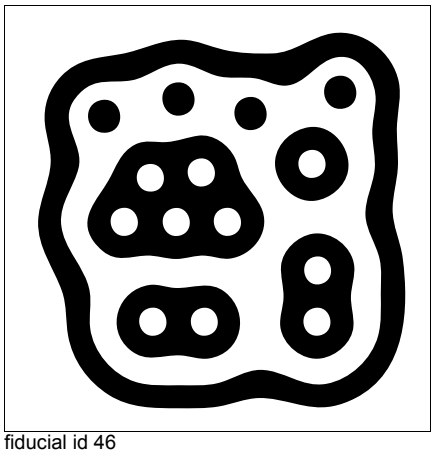

fiducial id 41

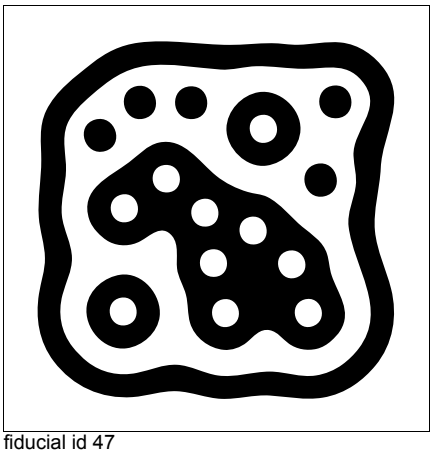

fiducial id 42

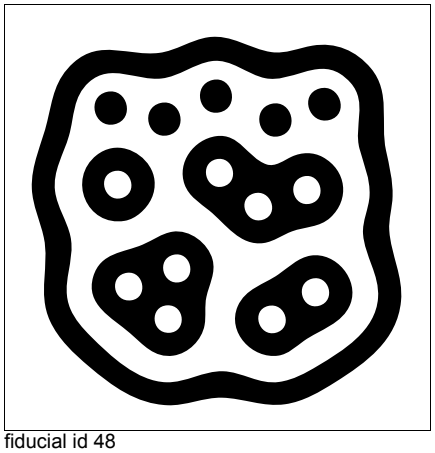

fiducial id 43

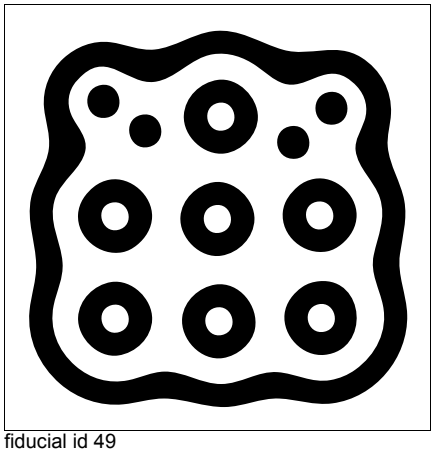

fiducial id 44

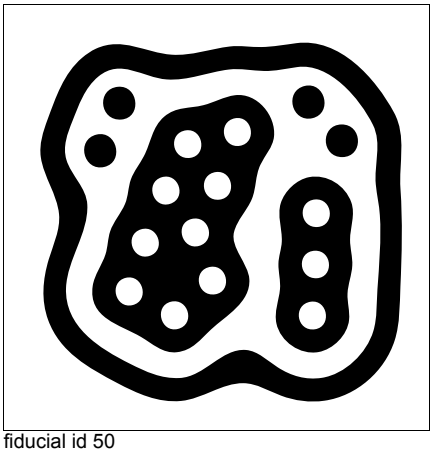

fiducial id 45

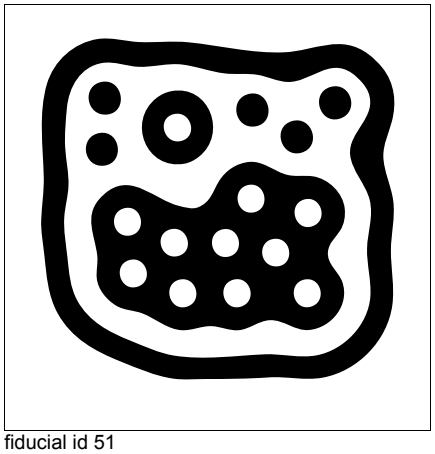

fiducial id 46

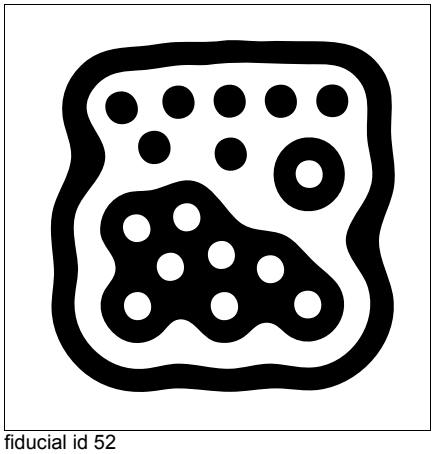

fiducial id 47

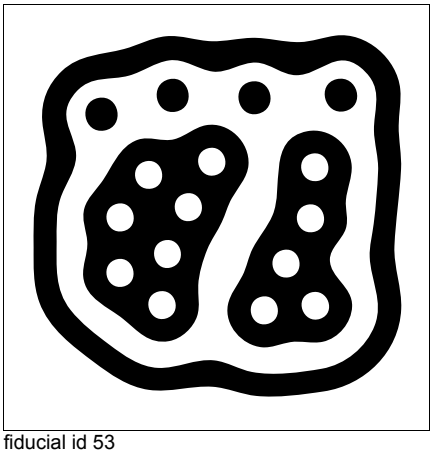

fiducial id 48

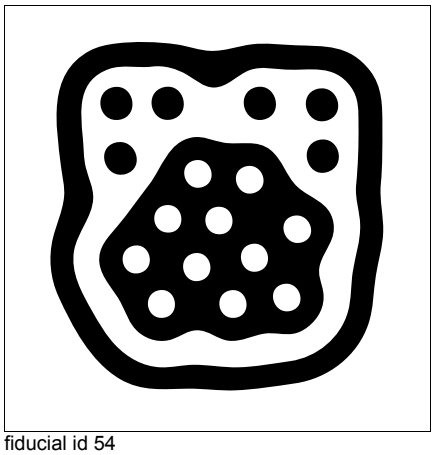

fiducial id 49

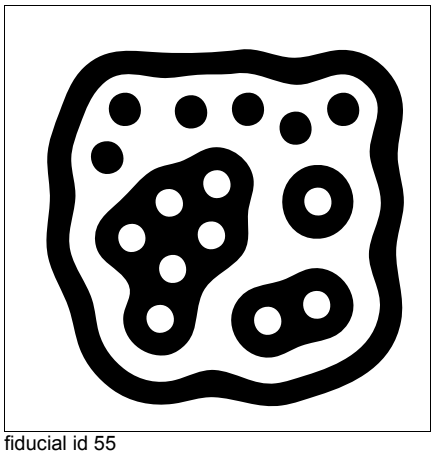

fiducial id 50

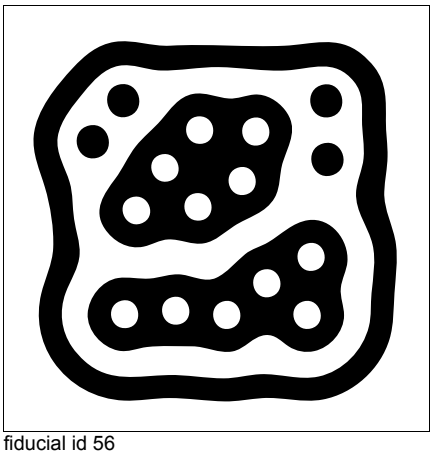

fiducial id 51

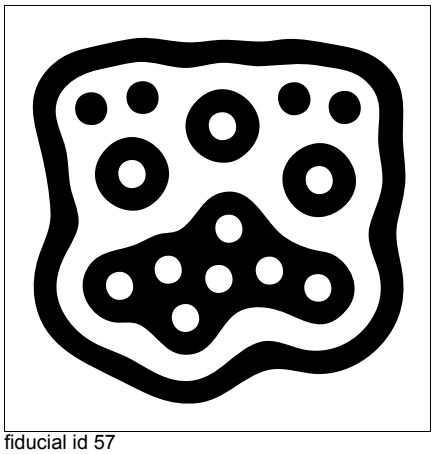

fiducial id 52

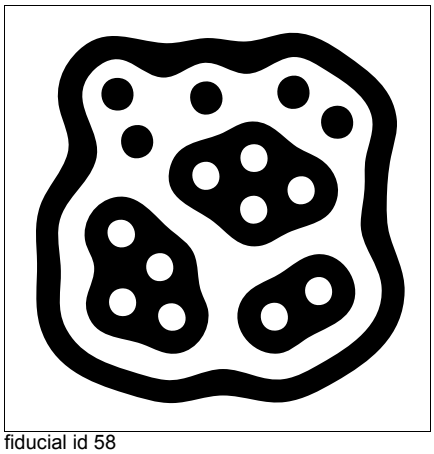

fiducial id 53

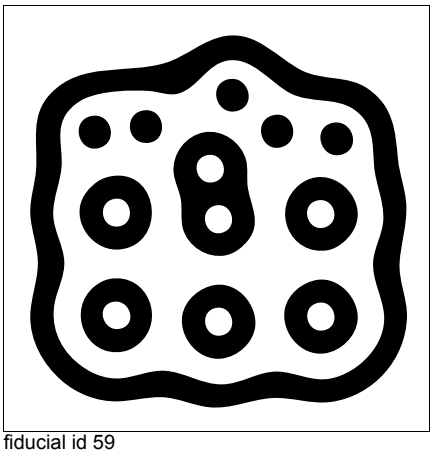

fiducial id 54

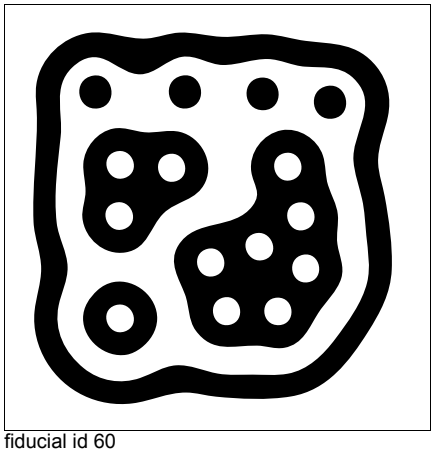

fiducial id 55

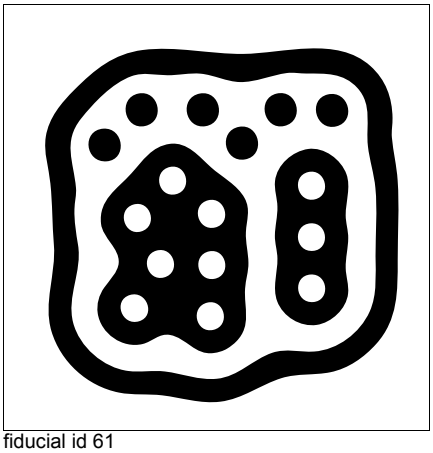

fiducial id 56

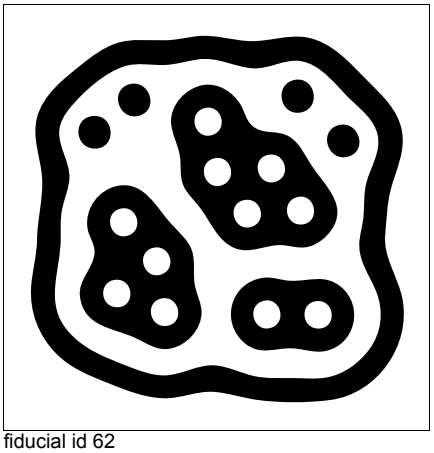

fiducial id 57

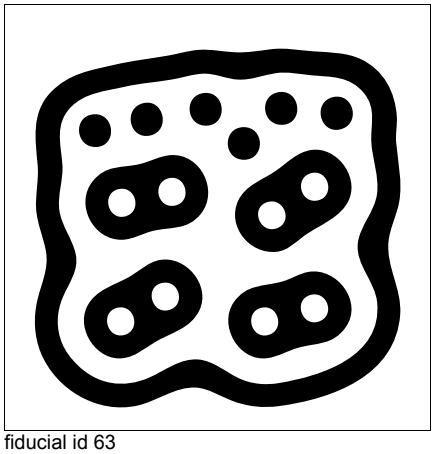

fiducial id 58

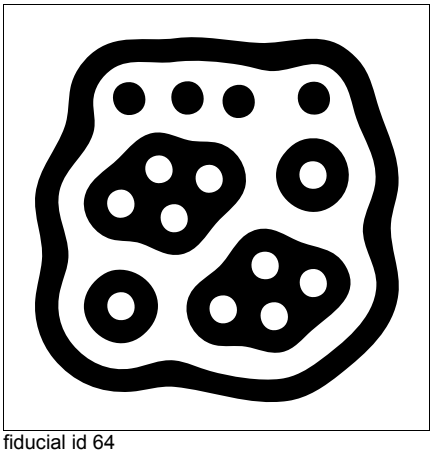

fiducial id 59

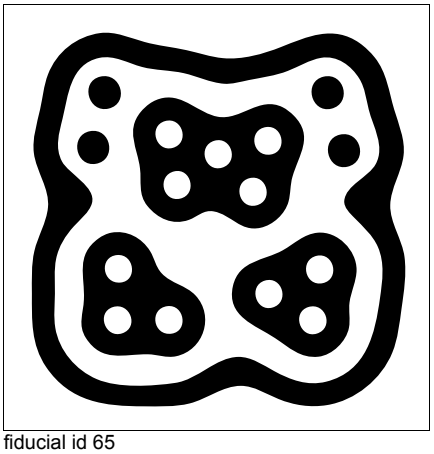

fiducial id 60

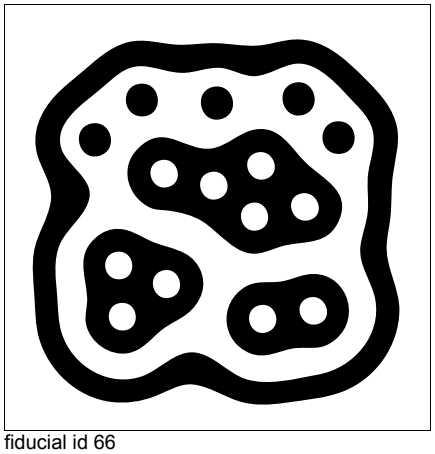

fiducial id 61

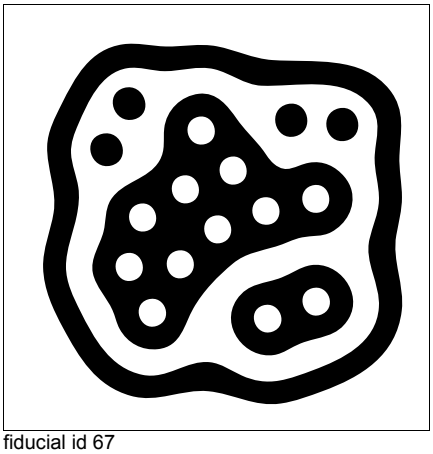

fiducial id 62

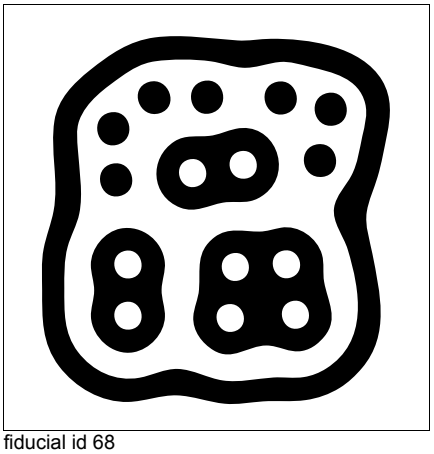

fiducial id 63

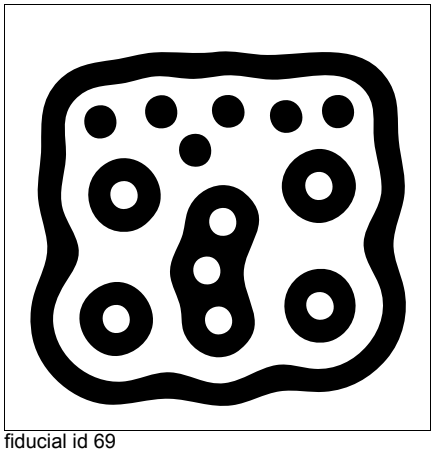

fiducial id 64

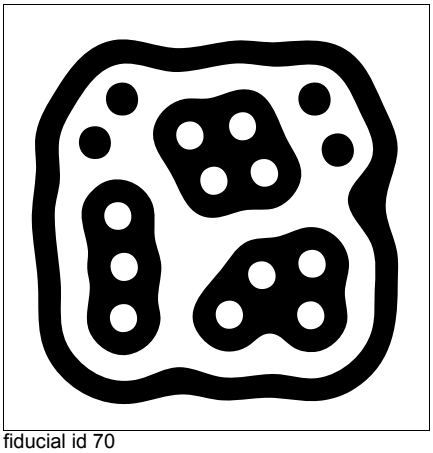

fiducial id 65

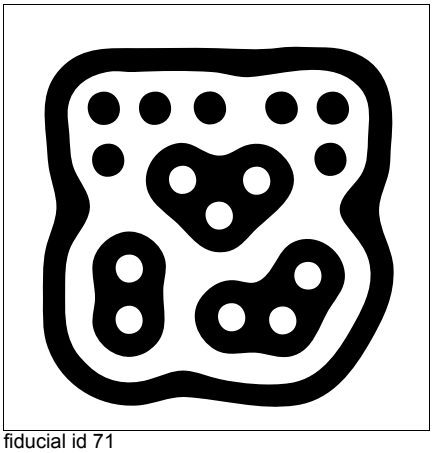

fiducial id 66

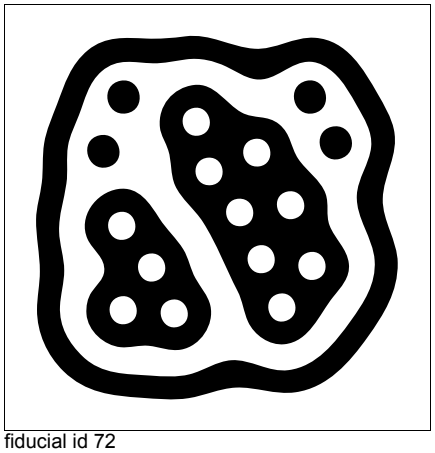

fiducial id 67

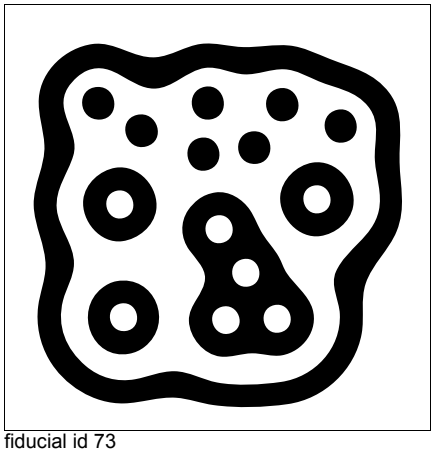

fiducial id 68

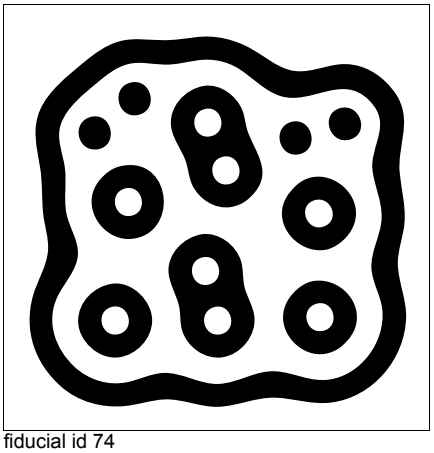

fiducial id 69

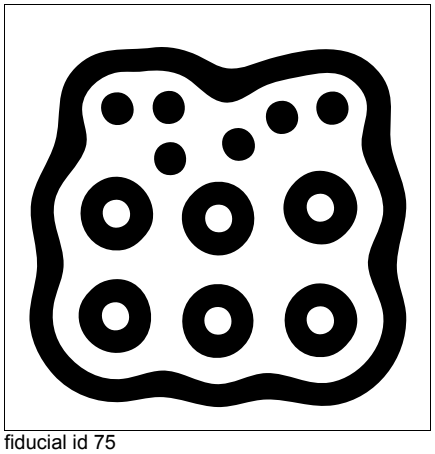

fiducial id 70

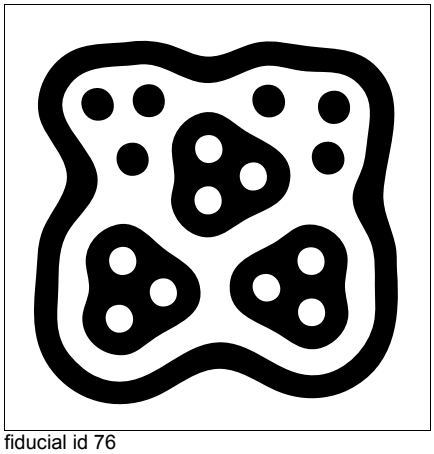

fiducial id 71

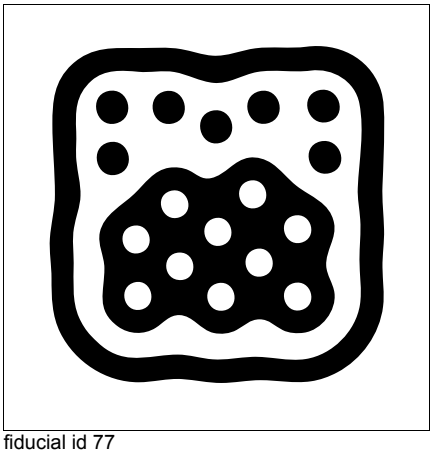

fiducial id 72

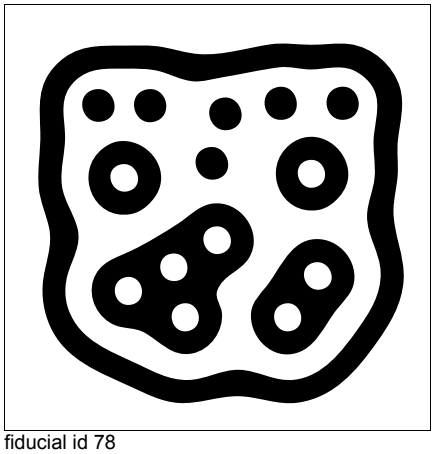

fiducial id 73

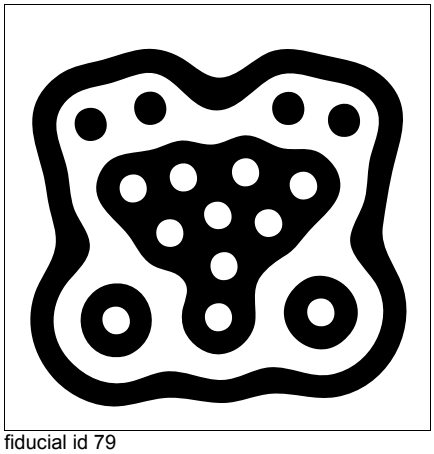

fiducial id 74

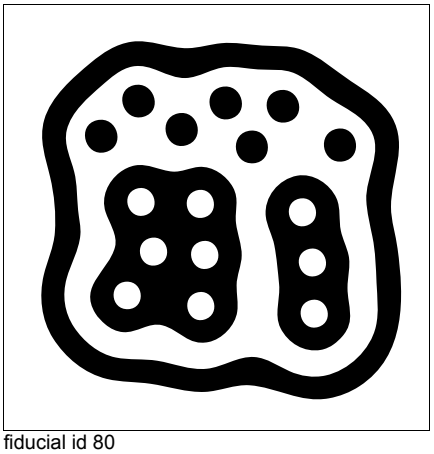

fiducial id 75

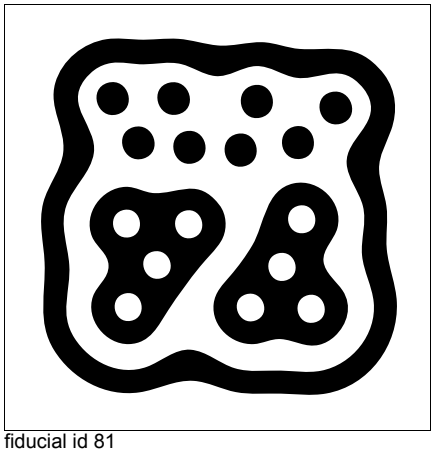

fiducial id 76

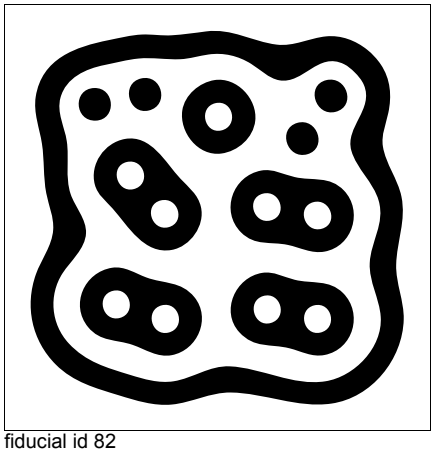

fiducial id 77

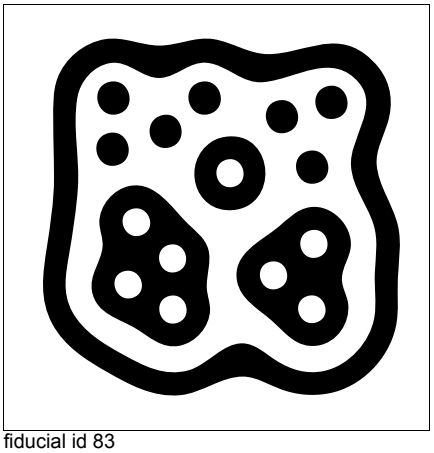

fiducial id 78

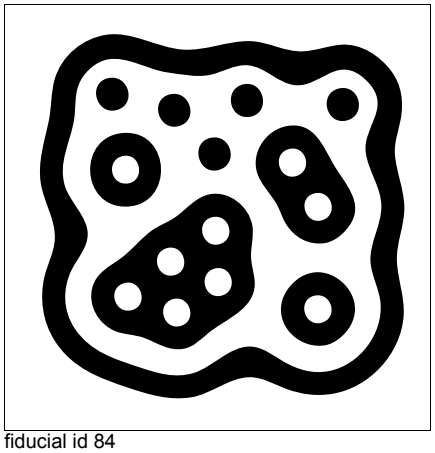

fiducial id 79

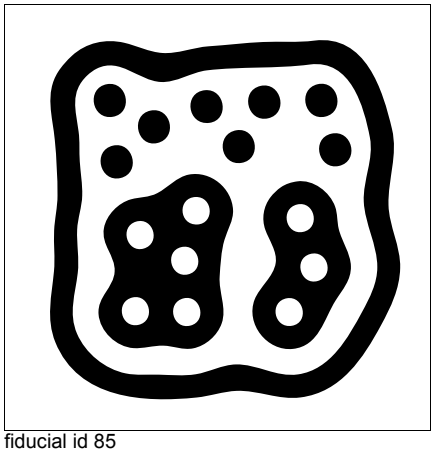

fiducial id 80

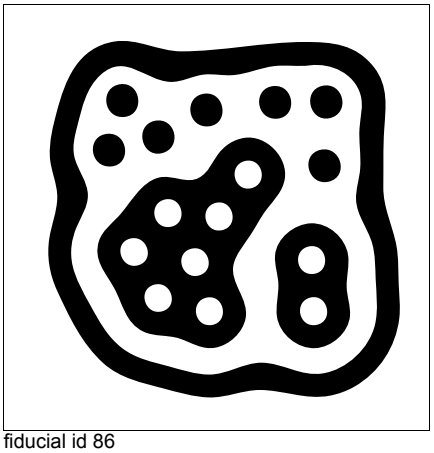

fiducial id 81

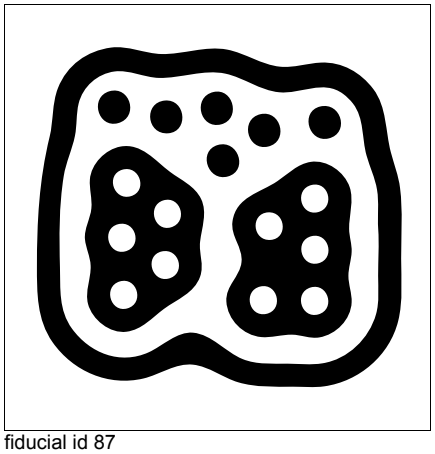

fiducial id 82

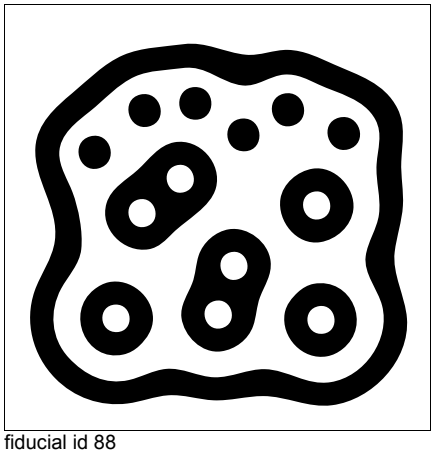

fiducial id 83

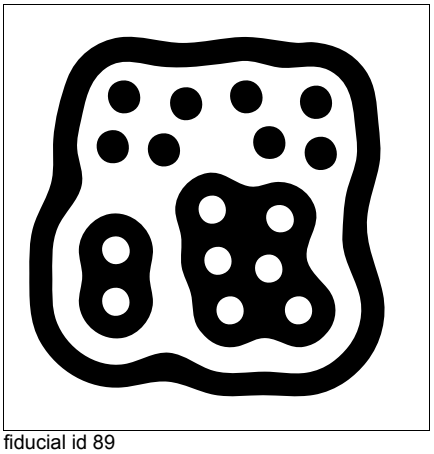

fiducial id 84

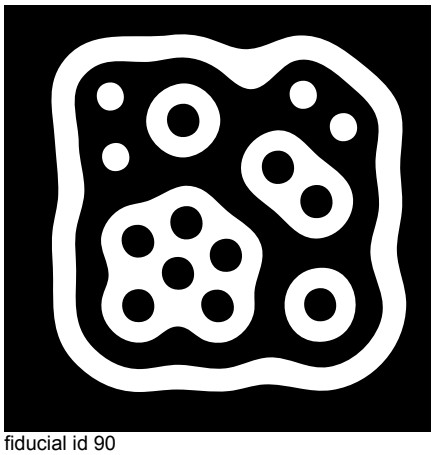

fiducial id 85

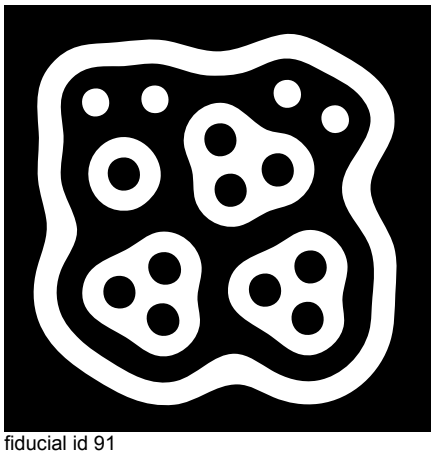

fiducial id 86

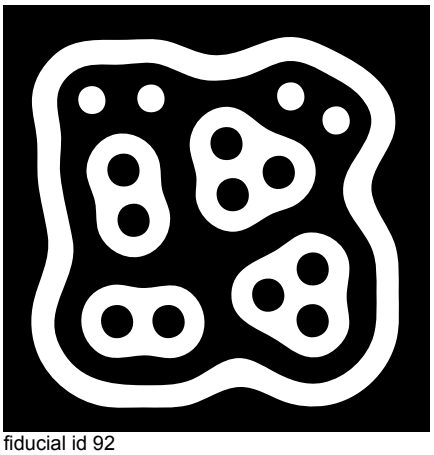

fiducial id 87

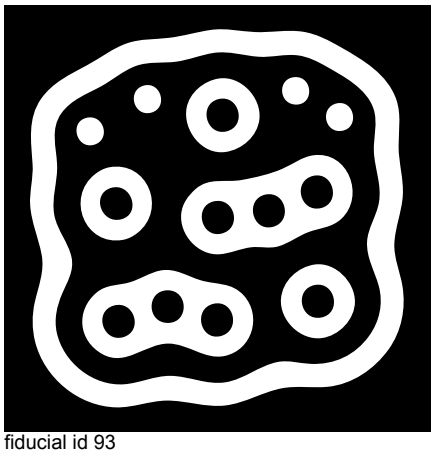

fiducial id 88

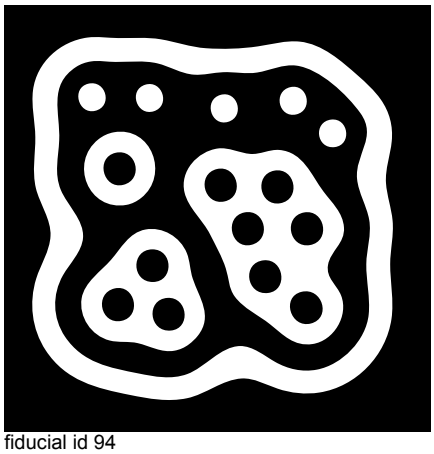

fiducial id 89

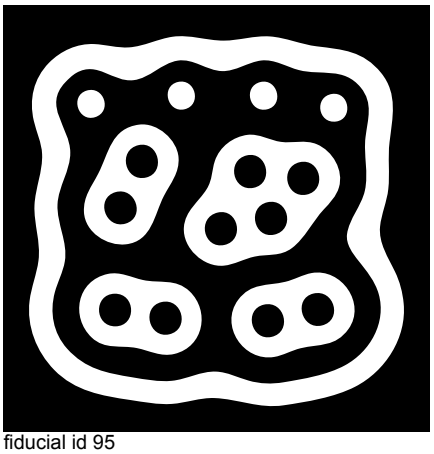

fiducial id 90

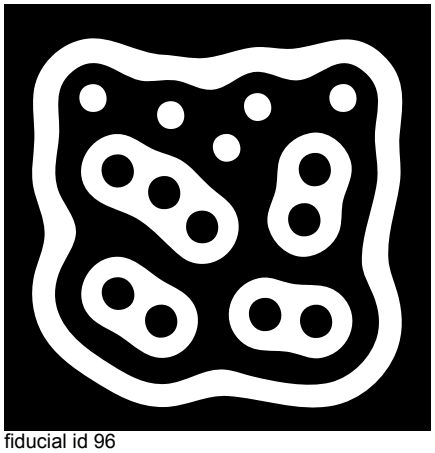

fiducial id 91


fiducial id 92

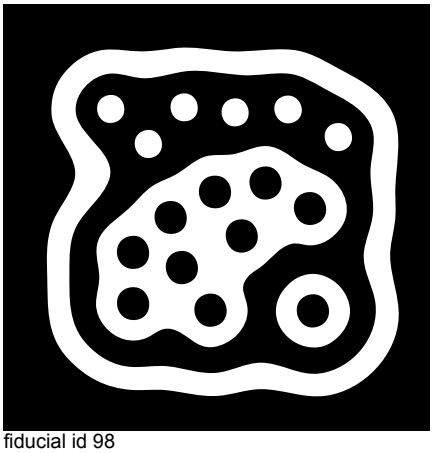

fiducial id 93

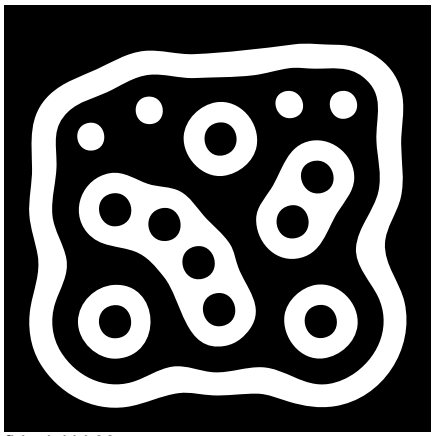

fiducial id 94

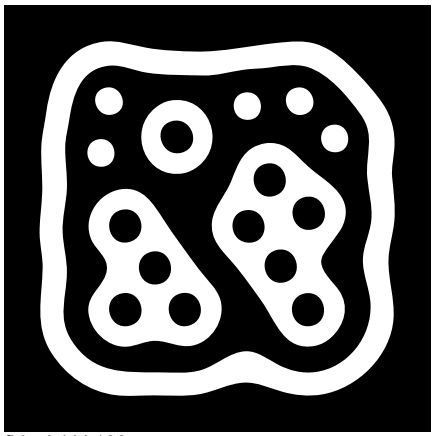

fiducial id 95

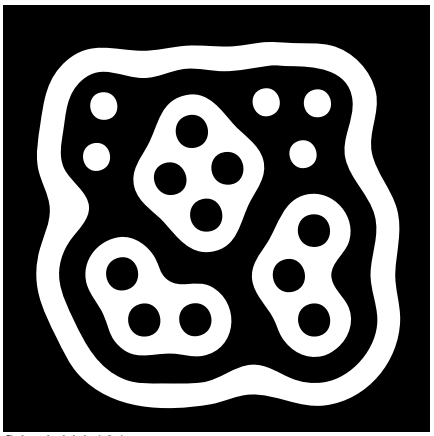

fiducial id 96

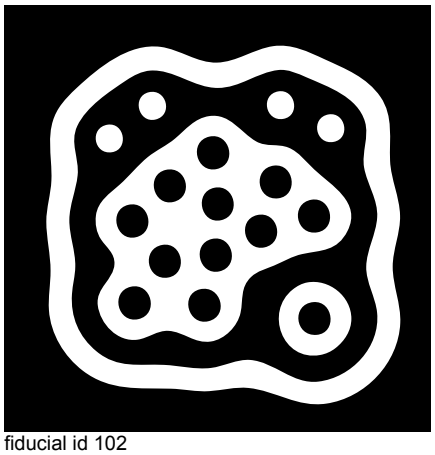

fiducial id 97

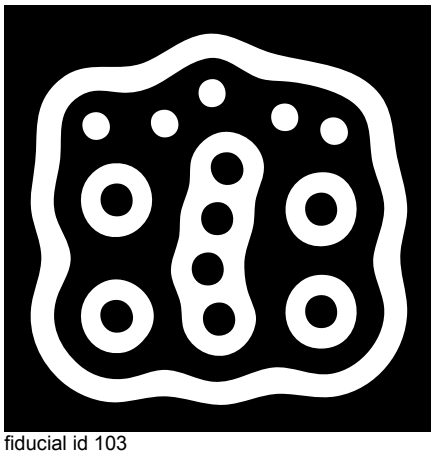

fiducial id 98

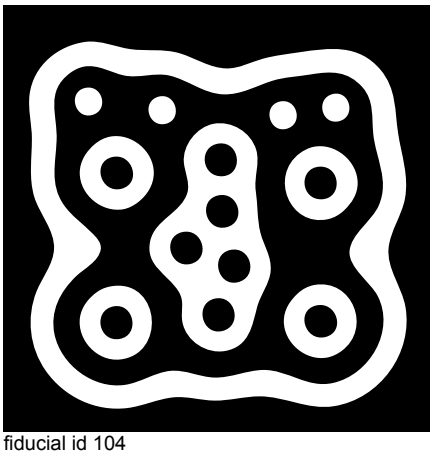

fiducial id 99

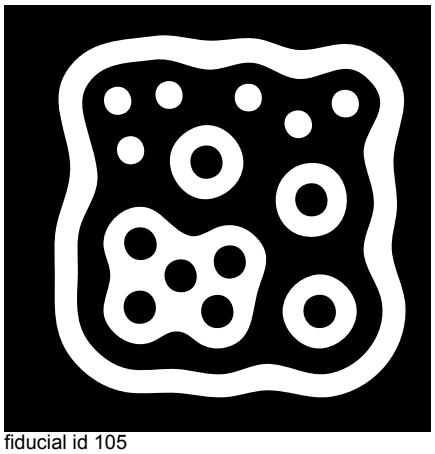

fiducial id 100

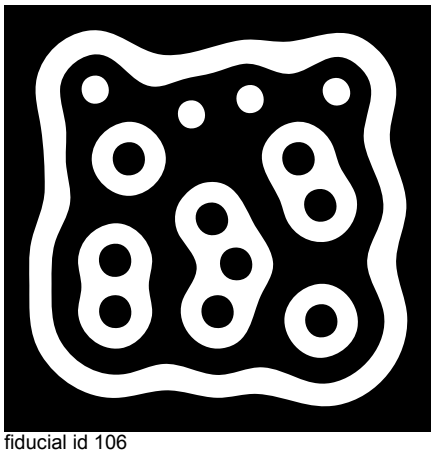

fiducial id 101

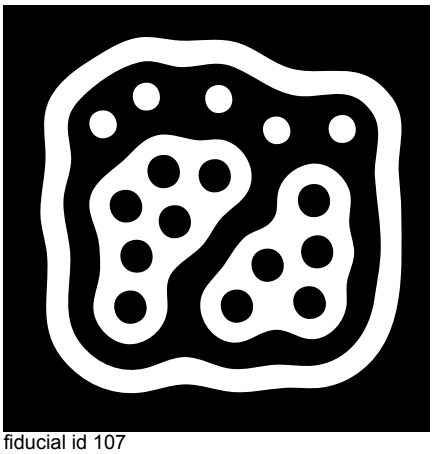

fiducial id 102

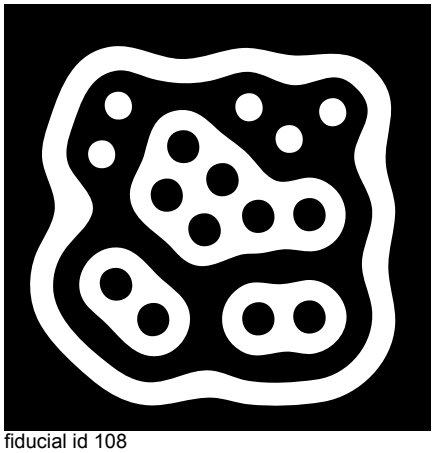

fiducial id 103

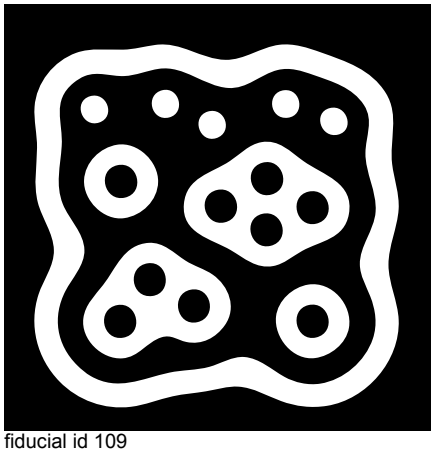

fiducial id 104

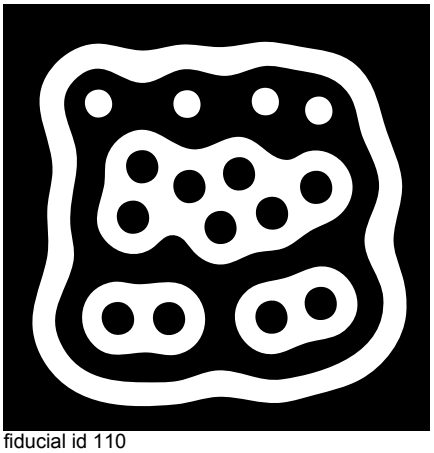

fiducial id 105

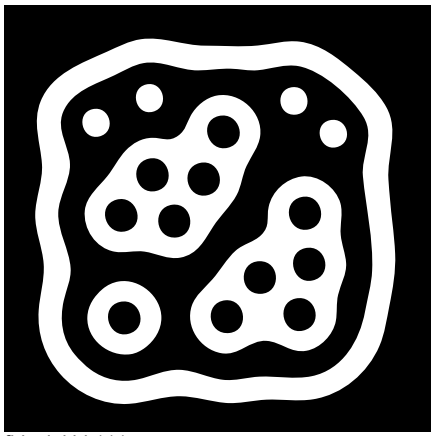

fiducial id 106

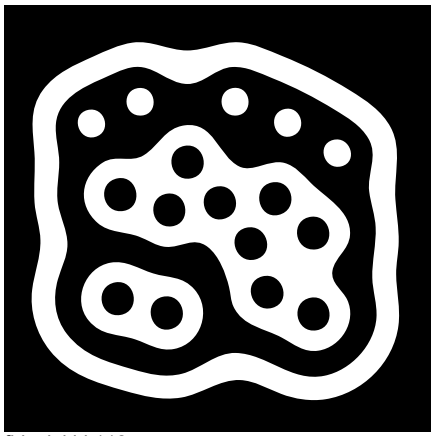

fiducial id 107

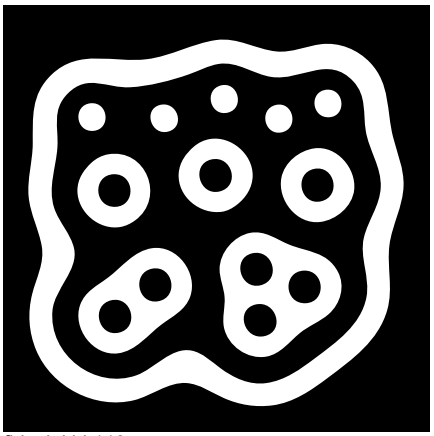

fiducial id 108

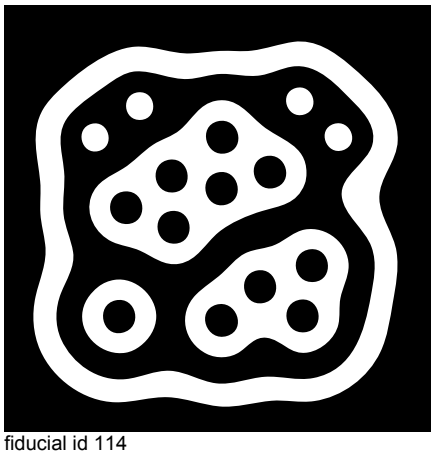

fiducial id 109

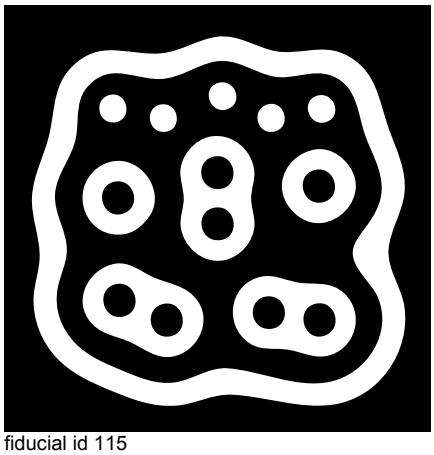

fiducial id 110

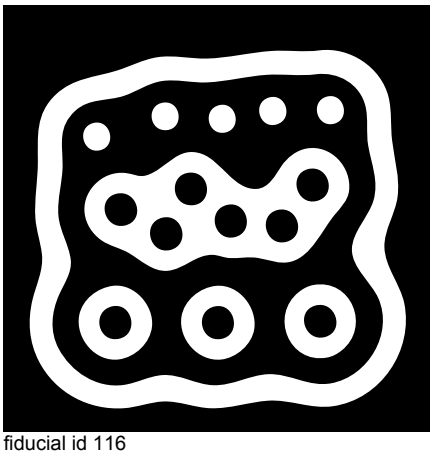

fiducial id 111

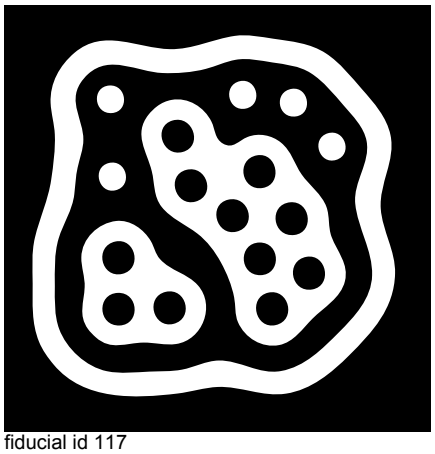

fiducial id 112


fiducial id 113

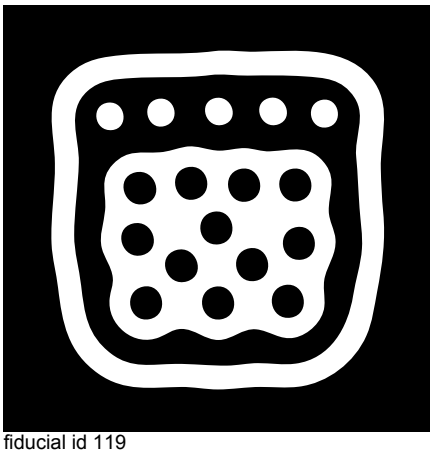

fiducial id 114

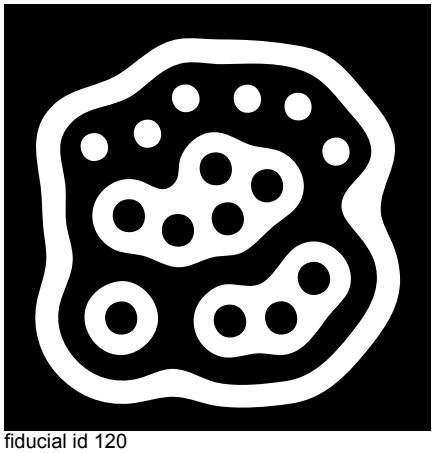

fiducial id 115

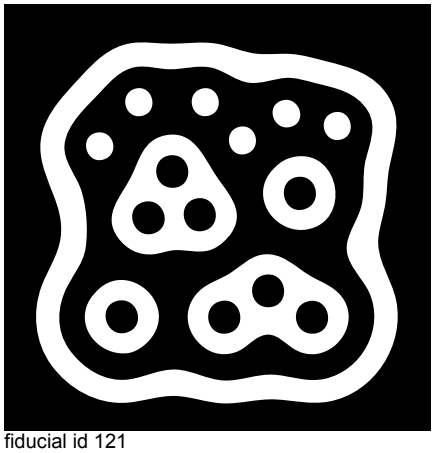

fiducial id 116

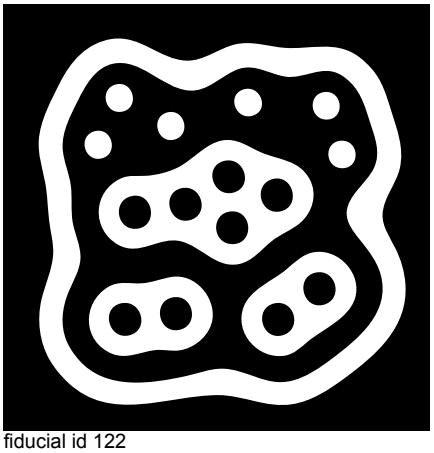

fiducial id 117

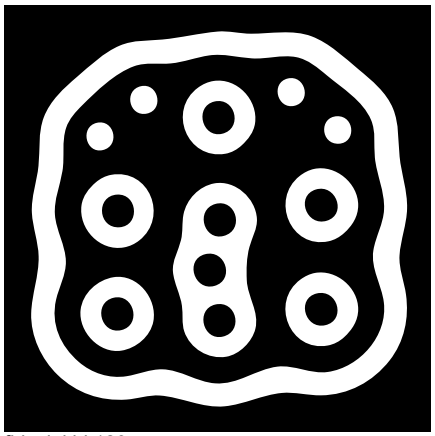

fiducial id 118

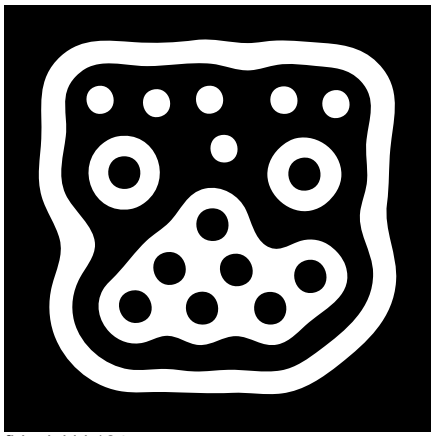

fiducial id 119

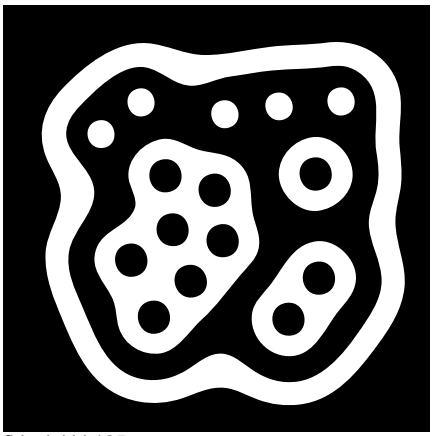

fiducial id 120

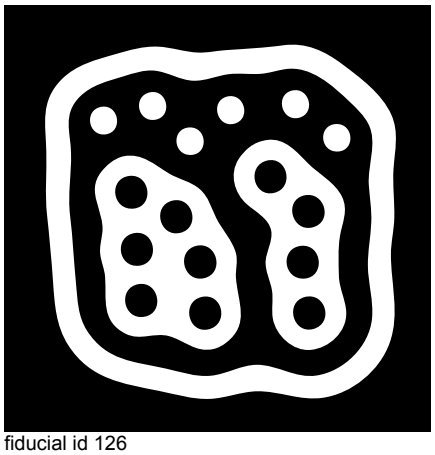

fiducial id 121

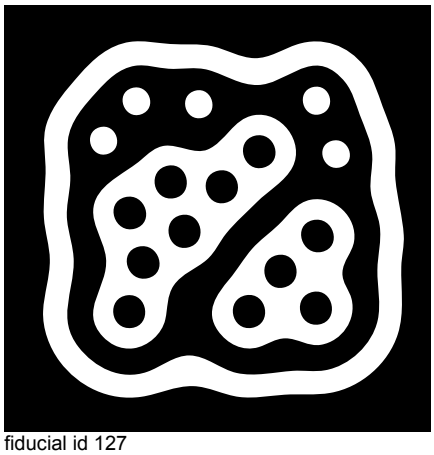

fiducial id 122

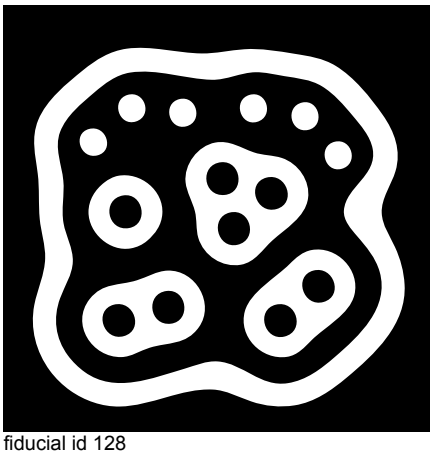

fiducial id 123

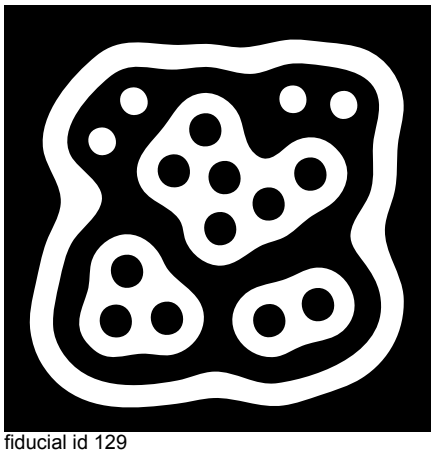

fiducial id 124

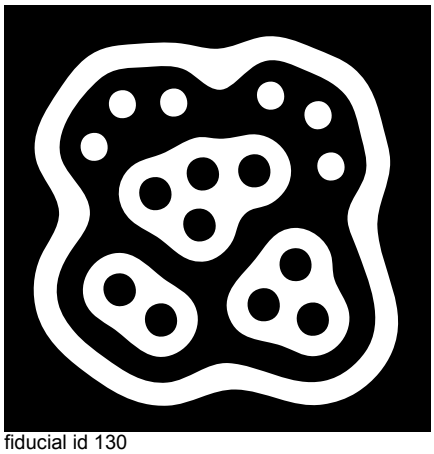

fiducial id 125

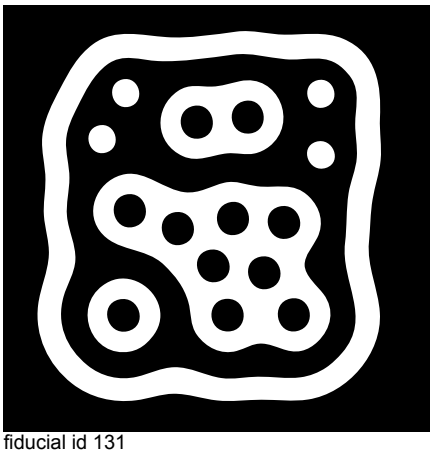

fiducial id 126

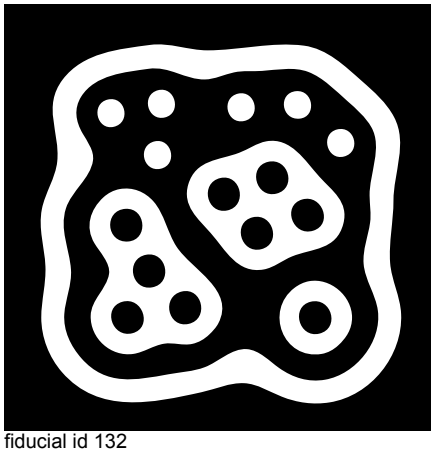

fiducial id 127

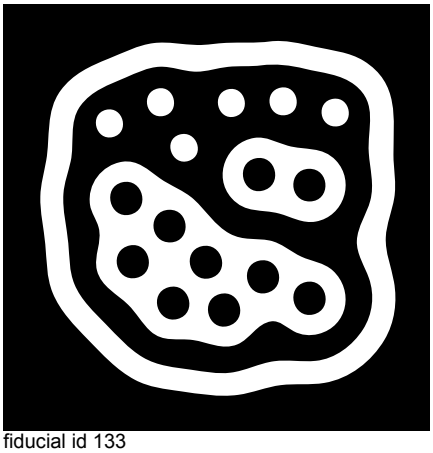

fiducial id 128

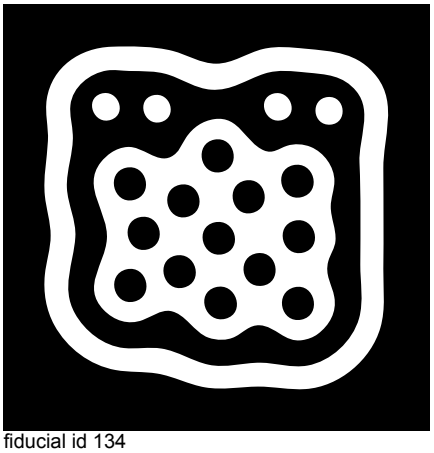

fiducial id 129

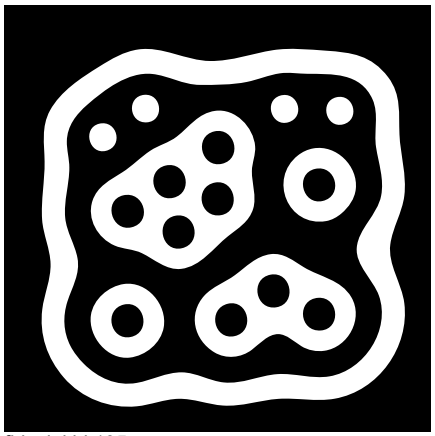

fiducial id 130

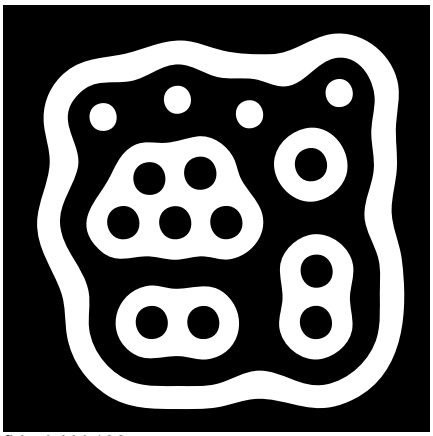

fiducial id 131

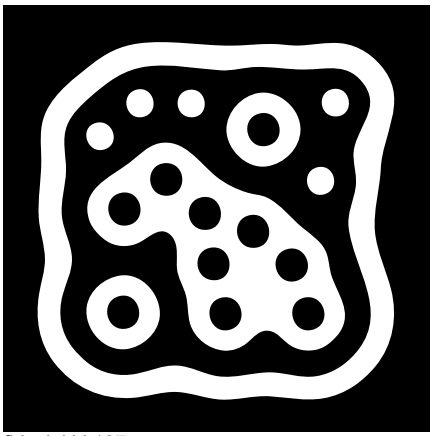

fiducial id 132

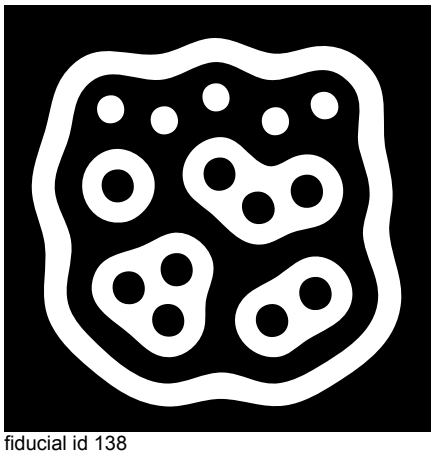

fiducial id 133

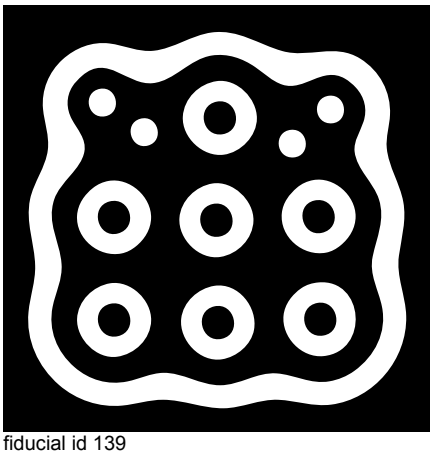

fiducial id 134

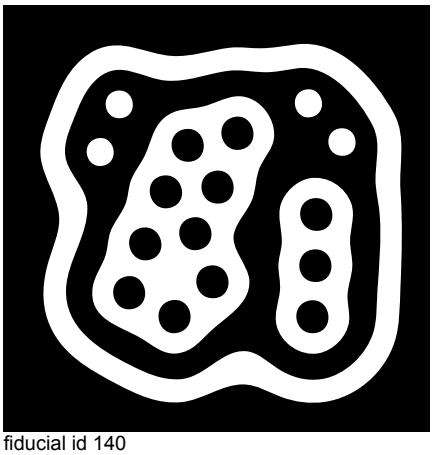

fiducial id 135

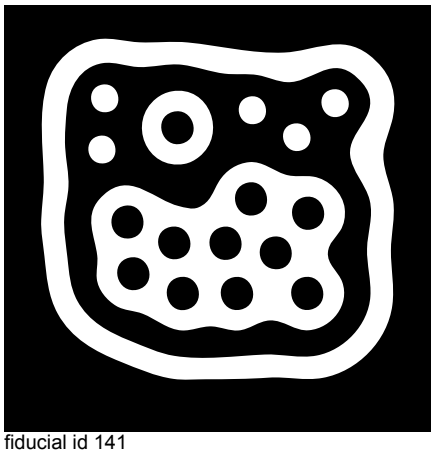

fiducial id 136

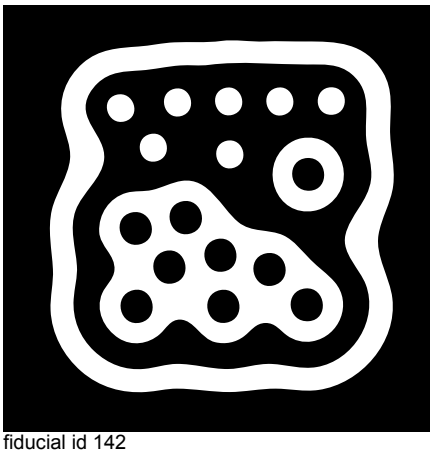

fiducial id 137

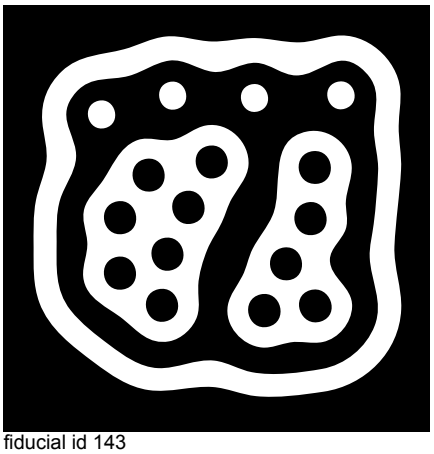

fiducial id 138

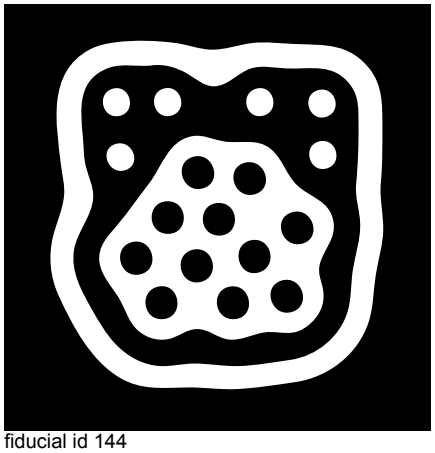

fiducial id 139

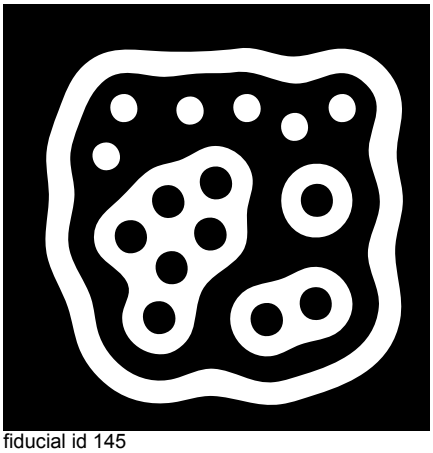

fiducial id 140

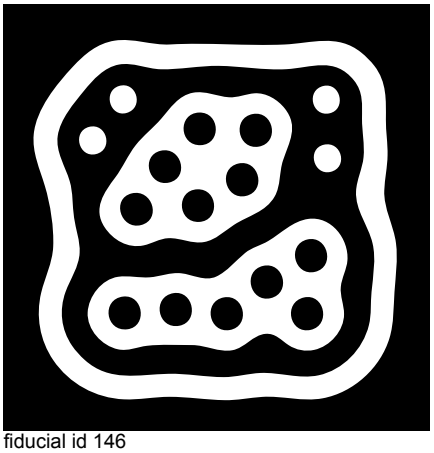

fiducial id 141

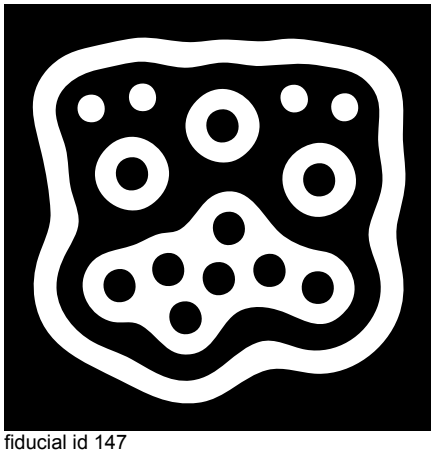

fiducial id 142

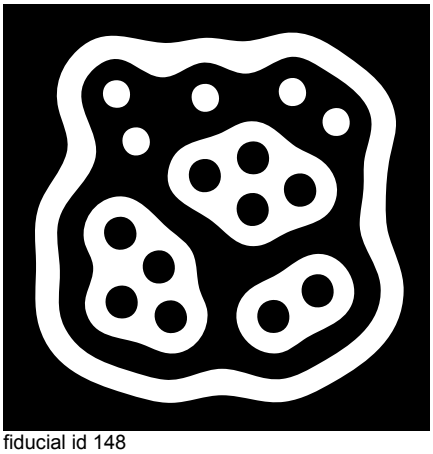

fiducial id 143

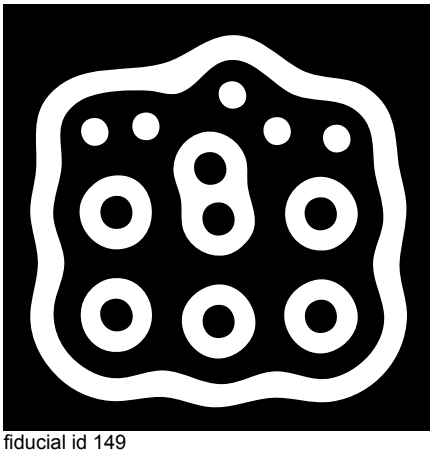

fiducial id 144

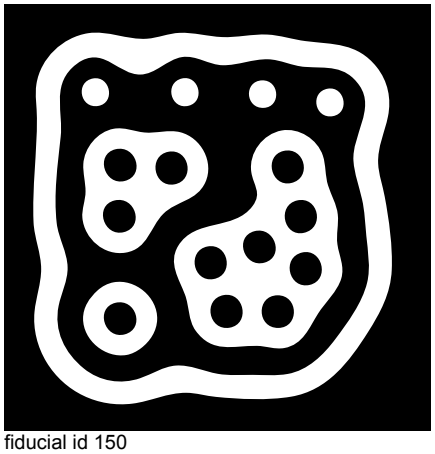

fiducial id 145

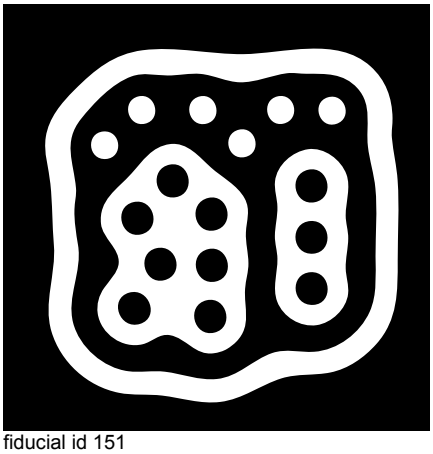

fiducial id 146

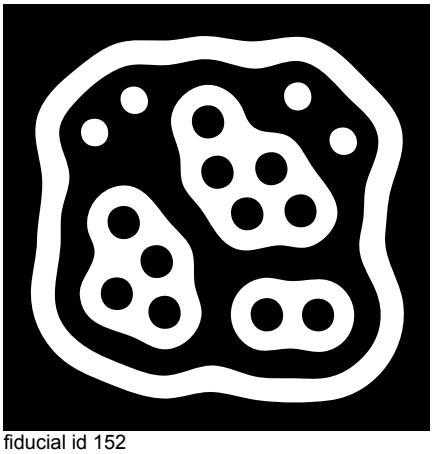

fiducial id 147

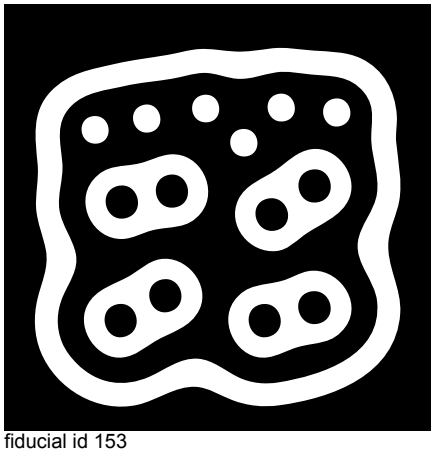

fiducial id 148

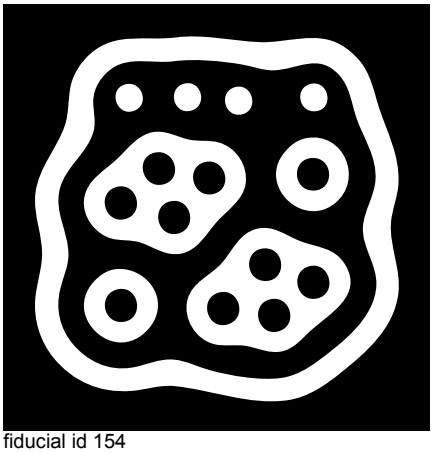

fiducial id 149

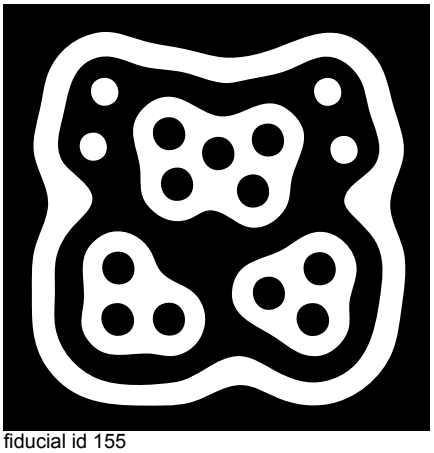

fiducial id 150

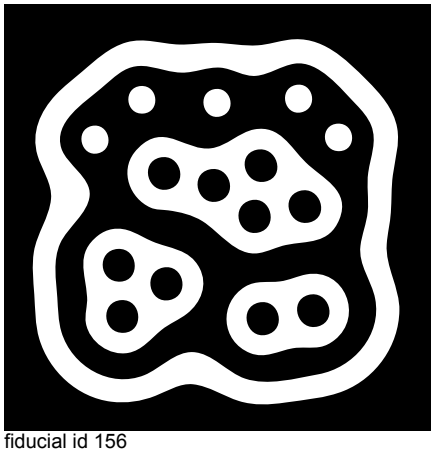

fiducial id 151

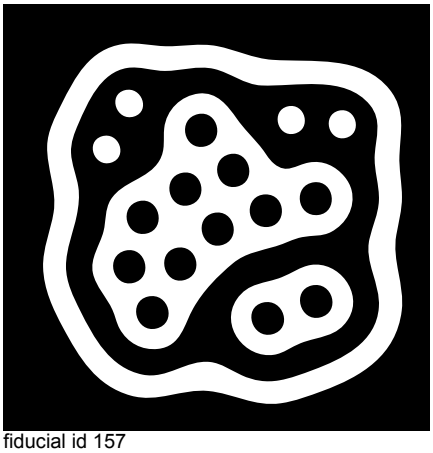

fiducial id 152

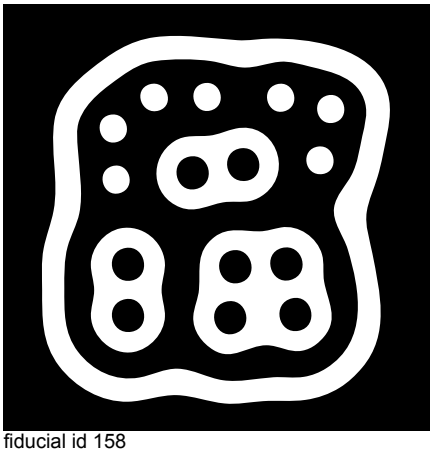

fiducial id 153

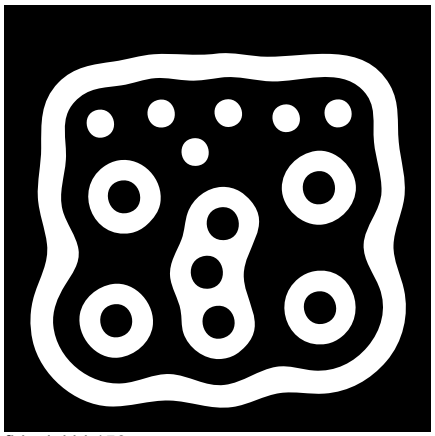

fiducial id 154


fiducial id 155

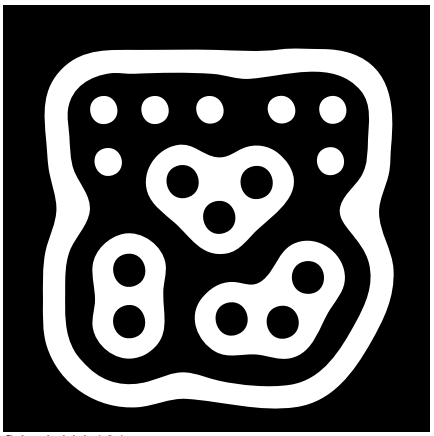

fiducial id 156

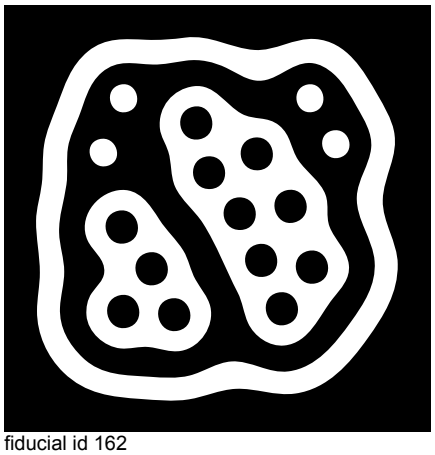

fiducial id 157

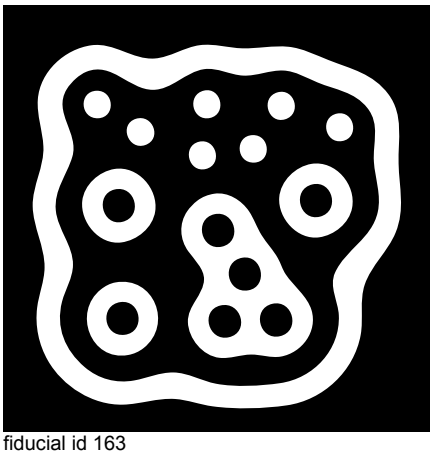

fiducial id 158

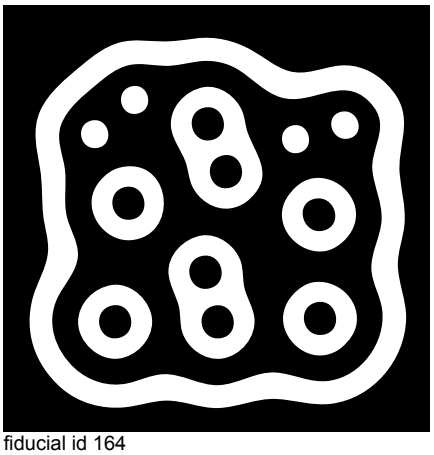

fiducial id 159

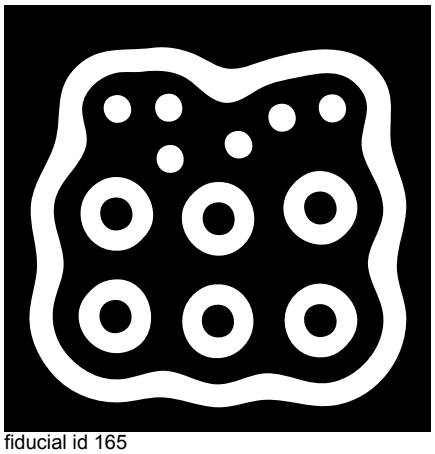

fiducial id 160

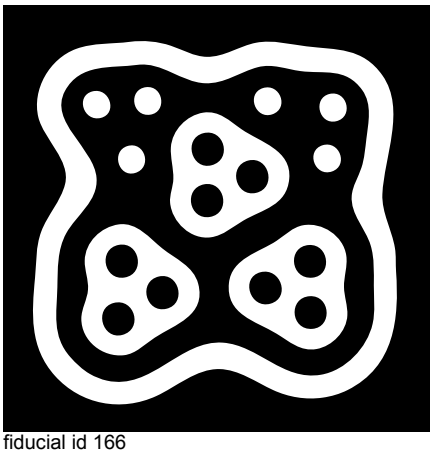

fiducial id 161

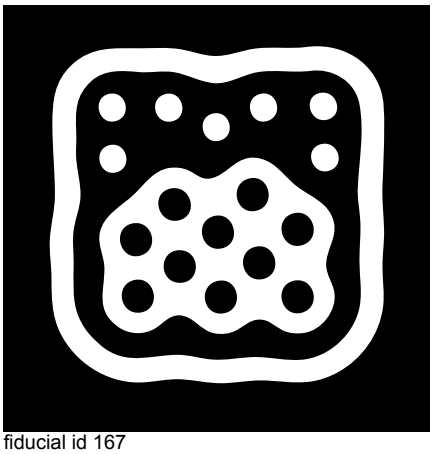

fiducial id 162

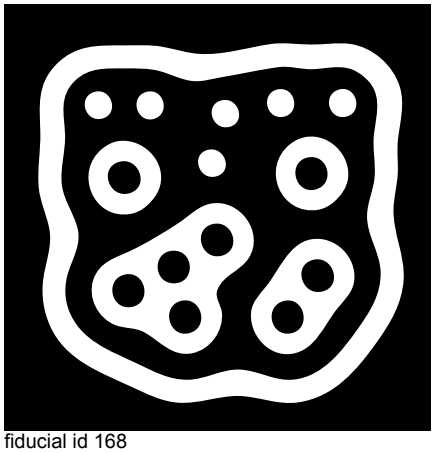

fiducial id 163

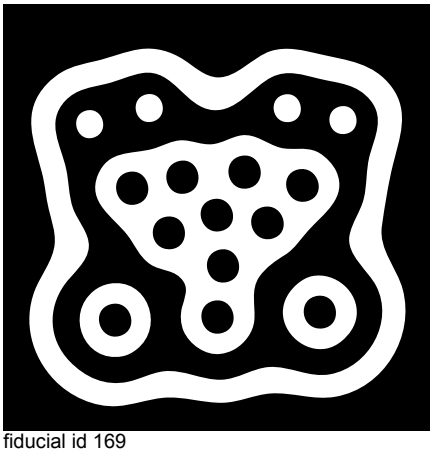

fiducial id 164

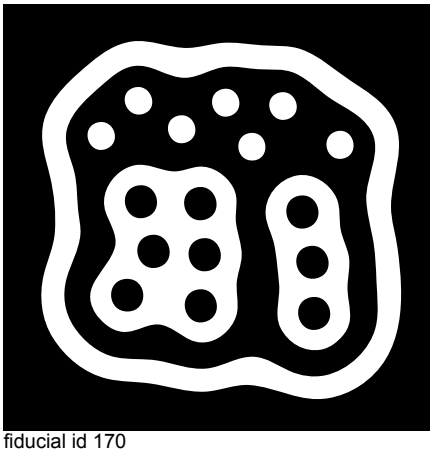

fiducial id 165

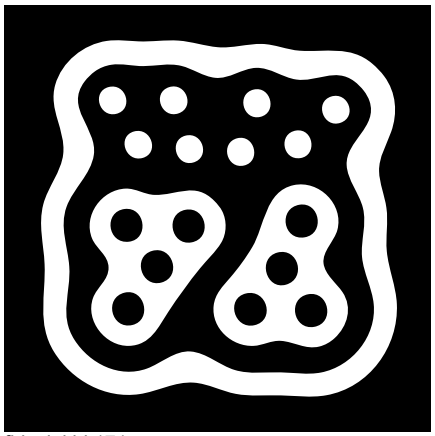

fiducial id 166

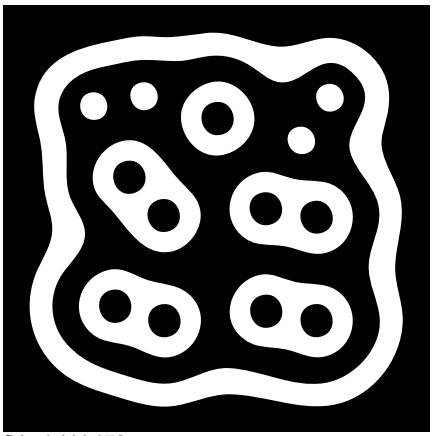

fiducial id 167

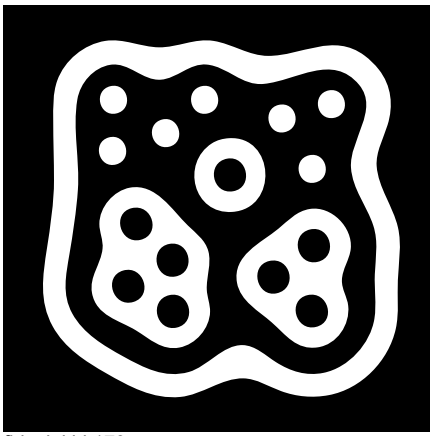

fiducial id 168

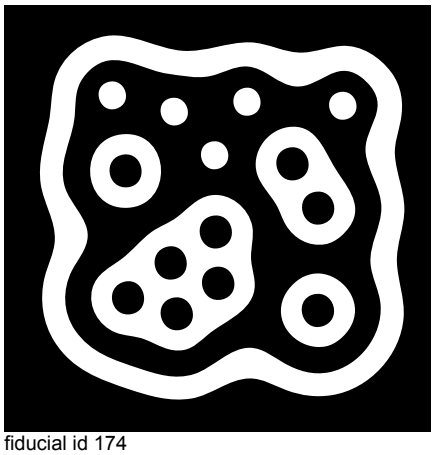

fiducial id 169

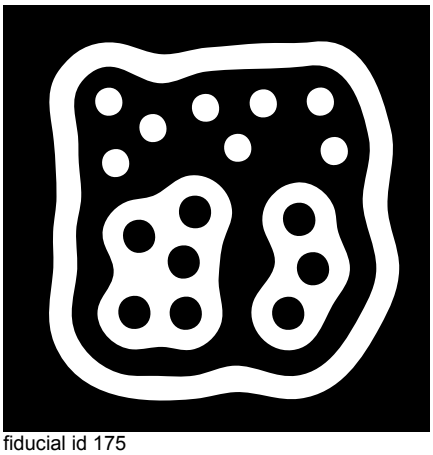

fiducial id 170

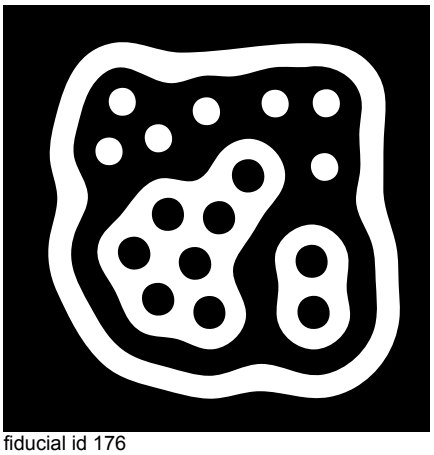

fiducial id 171


fiducial id 172

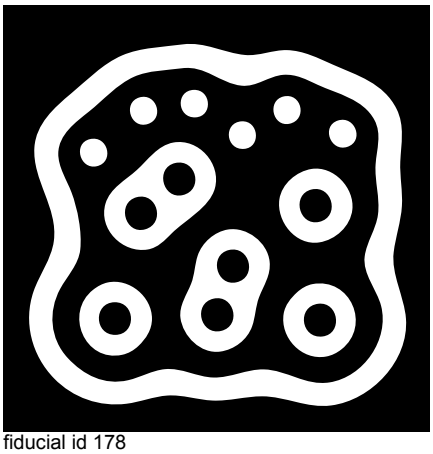

fiducial id 173


fiducial id 174


fiducial id 175


fiducial id 176


fiducial id 177


fiducial id 178


fiducial id 179

Bibliografía

[1] Leslie Lamport *LaTex : A document Preparation System*. Addison-Wesley, 1986.

[2] Christian Rolland *LaTex guide pratique*. Addison-Wesley, 1993.

[3] reacTIVision [Online]. Available: http://reactivision.sourceforge.net/

[4] ReacTable [Online]. Available: http://www.reactable.com/

[5] TUIO [Online]. Available: http://www.tuio.org/

[6] Processing [Online]. Available: http://processing.org/

[7] Silent Drum Project [Online]. Available http://www.jaimeoliver.pe/instrumentos/silent-drum

[8] Ableton Live! [Online]. Available: http://www.ableton.com/

[9] Pure Data [Online]. Available: http://puredata.info/

[10] ProMIDI [Online]. Available: http://creativecomputing.cc/p5libs/promidi/

[11] Jack OS X [Online]. Available http://www.jackosx.com/

[12] Community Core Vision [Online]. Available http://ccv.nuigroup.com/

[13] MAX/MSP [Online]. Available: http://cycling74.com/products/maxmspjitter/

[14] M. Kaltenbrunner, T. Bovermann, R. Bencina, and E. Costanza *TUIO: A protocol for table-top tangible user interfaces*. 2005

[15] M. Kaltenbrunner, R.Bencina and Sergi Jordá *Improved Topological Fiducial Tracking in the reacTIVision System*. 2005

[16] Texas Instrument [Online]. Available: http://www.ti.com/

[17] Inotouch [Online]. Available: http://www.inotouch.co.kr

[18] How TouchScreen Works? [Online]. Available: http://shivamgautam.blogspot.com.es/2011/04/how-touchscreen-works.html

[19] Multitouch Technologies [Online]. Available: http://nuicode.com/projects/wiki-book/files

[20] Norts Labs [Online]. Available http://labs.nortd.com/

[21] Stamtun [Online]. Available: http://www.stantum.com/

[22] Ideum [Online]. Available http://ideum.com/

[23] Sonia Pitaru [Online]. Available http://sonia.pitaru.com/

[24] LoopBe1 [Online]. Available http://nerds.de/en/loopbe1.html

[25] D-touch [Online]. Availabre http://www.d-touch.org/

[26] LiveOSC [Online]. Availabre http://liine.net/livecontrol/

[27] OSCp5 [Online]. Availabre http://www.sojamo.de/libraries/oscP5/

[28] OPen Sound Control [Online]. Availabre http://opensoundcontrol.org/

[29] Soundflower [Online]. Available http://cycling74.com/products/soundflower/