UNIVERSIDAD NACIONAL DE QUILMES
Departamento de Ciencias Sociales
Licenciatura en Música y Tecnología

DISEÑO Y DESARROLLO DE UN SISTEMA PARA LA INTERACCIÓN EN INSTALACIONES MEDIANTE DISPOSITIVOS MÓVILES

ALUMNO:

VALENTIN HOLGADO
DNI: 35619793
LEGAJO: 21342
CONTACTO: val.holgado@gmail.com

DIRECTOR:

ESTEBAN CALCAGNO

FECHA DE ENTREGA:

20 DE NOVIEMBRE DE 2014

Resumen

Esta tesis se centra en el diseño y desarrollo de un sistema para el control de interacción, en tiempo real, de obras musicales e instalaciones sonoras interactivas, las cuales poseen una dificultad de desarrollo implícita debido a que no existe una herramienta puntualmente diseñada para su creación y montaje.

Este sistema consta de dos aplicaciones informáticas que funcionan como prueba de concepto y que facilitan la producción de obras que requieran algún tipo de interacción remota por parte del público. Estas aplicaciones trabajan conjuntamente utilizando un modelo cliente-servidor a través de una red de área local inalámbrica (WLAN), en la que el servidor organiza y administra la información recibida de los clientes y a su vez funciona como nexo entre los clientes y otras aplicaciones para la generación y procesamiento de audio, como Pure Data.

Ambas aplicaciones están programadas en C++ utilizando openFrameworks, un conjunto de aplicaciones informáticas diseñadas para artistas y que presenta una amplia variedad de librerías.

En cuanto a la aplicación cliente, esta corre en el sistema operativo Android, lo cual permite que los espectadores de la obra puedan conectarse a la misma mediante diferentes dispositivos móviles. El uso de dispositivos móviles, y más particularmente de teléfonos inteligentes como plataforma para este proyecto permite aprovechar la creciente disponibilidad y variedad de éstos para ser adaptados e incluidos a la obra sin costo agregado. El uso de estas tecnologías permite, además, disponer los límites de interacción por parte del creador de la obra, pudiendo dirigirse a los espectadores con el fin de asignarles un rol en la obra, indicarles que es lo que deberían estar haciendo, en qué momentos pueden interactuar y de que manera.

La aplicación servidor, desarrollada actualmente para el sistema operativo Windows 7 (a futuro se planea que sea multiplataforma), cuenta con una línea de tiempo similar a la que habitualmente se encuentra en programas de edición de audio y video. En ella se pueden crear secciones, asignar duraciones y manejar distintos eventos, denominados bloques de interacción, que cumplen la función de solicitarle al cliente que, mediante los sensores de su

dispositivo movil, envíen datos a la obra. Los datos son obtenidos por el servidor en forma de símbolos numéricos y cadenas de texto y luego re-enviados a aplicaciones compatibles con el protocolo de comunicación OSC. Fue de particular ayuda para este proyecto que Pure Data, uno de los programas que se utilizó para probar la comunicación OSC, pudiera recibir correctamente toda la información enviada por los espectadores, para así aprovechar las capacidades que posee en cuanto al manejo de datos y audio.

Es importante aclarar que para llevar a cabo esta Tesis fue necesario aprender los lenguajes de programación C++ y Java, además del uso de la interfaz nativa de Java (JNI) y la estructura de aplicaciones Android, para poder desarrollar el sistema, así como software complementario propio del desarrollo sobre estas plataformas, Code::Blocks y Eclipse.

Al tratarse de un sistema inalámbrico que funciona sobre redes de área local, fue necesario también estudiar los protocolos de comunicación disponibles y sus características.

Antecedentes y estado de situación

En 1957 Max Matthews fue responsable de los primeros sonidos generados por computadora[1] utilizando el primer lenguaje de programación orientado al audio denominado MUSIC. Dada su poca velocidad de procesamiento las primeras computadoras eran utilizadas para controlar equipos de generación de sonido, tales como sintetizadores analógicos, pero no para producir sonido directamente en tiempo real, ejemplo de esto es el sintetizador GROOVE diseñado por Matthews y F.R. Moore[2], que utiliza una computadora conectada a un sintetizador analógico controlado por tensión.

Recién a fines de los años '70, con la aparición de las primeras computadoras de menor tamaño y costo, y mayor capacidad de procesamiento, se empezaron a utilizar más frecuentemente en vivo y para generación en tiempo real de sonido[3]. Las novedades más importantes que aporta la computación al arte sonoro son la programación, el control y manejo preciso de datos, y nuevas técnicas de generación y procesamiento de sonido. Por esta época comienzan a aparecer las primeras aplicaciones interactivas orientadas a la música y el sonido, y queda sentado que la programación y el control de datos hacen posible la creación de obras e instalaciones interactivas de manera más fácil y accesible que la electrónica analógica.[4]

En las décadas de 1980 y 1990 se realizaron avances importantes en las nuevas tecnologías musicales, los sistemas de computación continuaron abaratando su costo y el audio digital toma protagonismo en los estudios y en los escenarios. Aparece la aplicación Music Mouse software de acompañamiento musical, simple y accesible que puede ser utilizado por cualquier usuario sin conocimientos previos de programación[5]. En 1983 se crea el estándar MIDI[6]

---

[1] Holmes, Thom, and Terence M. Pender. *Electronic and Experimental Music: Technology, Music, and Culture*. New York: Routledge, 2012. Impreso.

[2] Mathews, M. V., and F. R. Moore. "GROOVE - A Program to Compose, Store, and Edit Functions of Time." *Communications of the ACM* 13.12 (1970): 715-721. Impreso.

[3] Holmes, Thom, and Terence M. Pender. *Op. Cit*.

[4] Jordá, Sergi. "Interactivity and Live Computer Music." *The Cambridge Companion to Electronic Music*. Ed. Nick Collins and Rincón Julio D' Escrivan. Cambridge: Cambridge UP, 2007. Impreso.

[5] Jordà, Sergi. *Op. Cit*.

[6] Huber, David Miles. *The MIDI Manual: A Practical Guide to MIDI in the Project Studio*. Boston: Focal, 2007. Impreso.

para la interconexión de generadores de sonido (ej. sintetizadores), interfaces musicales y computadores, que simplificó y unificó el conexionado entre estos equipos. En los años '80 Miller Puckette comienza con el desarrollo de lo que sería Max/MSP[7], y años más tarde Pure Data[8], ambos entornos gráficos de programación son ampliamente utilizados en la actualidad en instalaciones y performance en vivo. Otros lenguajes de programación destacados son SuperCollider[9], lanzado en 1996, y CSound[10], descendiente directo de MUSIC. Comienza hacia finales de los '90 el desarrollo de OSC (Open Sound Control), protocolo de objetivo similar a MIDI pero compatible con redes ethernet y que permite mensajes de mayor complejidad.

Por esta época comienzan también las primeras incursiones en el área de HCI, Human-Computer Interaction, que hasta este momento se trataba de un área explorada sólo de forma empírica y muy poco desarrollada. Ésta nueva área toma elementos de diversas disciplinas como la informática, la psicología, ciencias de la cognición, ergonomía, diseño, ingeniería, antropología, entre otras, para estudiar el vínculo que se produce entre un humano y una máquina buscando mejorar los medios de comunicación entre ambos para que sea más accesible. El campo de investigación principal de este área es la interacción entre un usuario y un sistema. Esta interacción consta de dos partes: control y realimentación. El usuario controla el sistema y este devuelve información que puede servir al usuario para tomar nuevas decisiones, este ciclo se repite en forma de bucle: a cada entrada de control le corresponde una devolución o feedback. Cuando la cognición de cualquiera de las dos partes (humano o sistema) se deja de lado, es decir, cuando una de las dos partes no evalúa lo recibido para devolver información a la otra parte se rompe este lazo interactivo, y se dice que el sistema es reactivo y no interactivo, ya que las acciones y respuestas de uno no están modificando las del otro[11].

Bert Bongers distingue tres formas de interacción cuando se involucran a sistemas electrónicos: interacción performer-sistema, es el caso más habitual y, por ejemplo, puede tratarse de un músico tocando un instrumento; interacción público-sistema, los casos más habituales incluyen los programas de computadora, consolas, e instalaciones interactivas;

---

[7] "Cycling 74." *Cycling 74*. Web. 09 Junio 2013. <http://cycling74.com/>.

[8] "PD Community Site." *Pure Data*. Web. 09 Junio 2013. <http://puredata.info/>.

[9] "SuperCollider" *SuperCollider*. Web. 29 Junio 2013. <http://supercollider.sourceforge.net/>.

[10] "Csound: C-Based Audio Programming Language" *CSound*. Web. 18 Octubre 2013. http://www.csounds.com/

[11] Noble, Joshua J. *Programming Interactivity: A Designer's Guide to Processing, Arduino, and OpenFrameworks*. Sebastopol, CA: O'Reilly, 2012. Impreso.

interacción perfomer-sistema-público, el público deja su rol pasivo e interactúa con el performer a través de algún sistema electrónico, además de no descartar la interacción directa entre performer y audiencia.[12]

Es necesaria una interfaz que funcione como nexo entre el sistema y el usuario para que la interacción pueda ser posible, está interfaz transforma acciones físicas en señales eléctricas y viceversa, es decir que recibe y envía información al sistema. Dentro de la interfaz, los sensores son los encargados de convertir en impulsos eléctricos los estímulos físicos producidos por el usuario, y los actuadores funcionan a la inversa, devolviendo información al mundo físico a partir de señales eléctricas que generan una representación que pueda percibir el espectador, es decir devuelven el feedback[13] [14].

En la primer década del nuevo milenio aparecieron en el mercado nuevos dispositivos que hicieron aún más accesible la producción de instalaciones interactivas, el sensado de señales externas y la fabricación de equipos autónomos. Son ejemplo de esto las placas open hardware BeagleBoard[15] (2008), Raspberry Pi[16] (2012) y Arduino[17] (2005), esta última diseñada teniendo en cuentas las necesidades de los artistas multimediales. Estas placas acercan la tecnología de microcontroladores (disponibles desde 1971 en sus primeras apariciones) a usuarios sin conocimientos previos de programación y electrónica, Ganando popularidad gracias a la amplia cantidad de libros, tutoriales, vídeos y artículos que se pueden encontrar en Internet de forma libre y gratuita.

Desde el año 2007 están disponibles en el mercado dispositivos móviles, tales como tablets y teléfonos, con sistemas operativos abiertos a la programación de terceros siendo los más utilizados Android[18], iOS[19] y Windows Phone[20]. En sus respectivos centros de

---

[12] Bongers, Bert. *Physical Interaction in the Electronic Arts*. France: IRCAM, 2000. CD-ROM.

[13] Noble, Joshua J. *Op. Cit.*

[14] Bongers, Bert. *Op. Cit.*

[15] "BeagleBoard.org - community supported open hardware computers for making" *BeagleBoard*. Web. 18 Octubre 2013. <http://beagleboard.org/>

[16] "Raspberry Pi." *Raspberry Pi*. Web. 09 Junio 2013. <http://www.raspberrypi.org/>.

[17] "Arduino." *Arduino*. Web. 09 Junio 2013. <http://www.arduino.cc/>.

[18] "Android." *Android*. Web. 18 Octubre 2013. <http://www.android.com/>.

[19] "iOS Dev Center - Apple Developer." *iOS Dev Center*. Web. 18 Octubre 2013. <https://developer.apple.com/devcenter/ios/index.action>

[20] "Windows Phone Dev Center." *Windows Phone*. Web. 18 Octubre 2013. <http://developer.windowsphone.com/>.

aplicaciones se encuentra una gran oferta de software relacionado a la producción musical: sintetizadores, samplers, controladores, etc. Existen también plataformas de programación como Processing[21], basado en Java, y openFrameworks[22], basado en librerías escritas en C++, destinadas a la programación de gráficos, visuales, videos, etc. que permiten integrar elementos del audio digital y de las plataformas de desarrollo físicas anteriormente mencionadas para la generación de programas e instalaciones audiovisuales interactivas.

[21] "Processing.org." *Processing.org*. Web. 18 Octubre 2013. <http://processing.org/>.

[22] "openFrameworks." *openFrameworks*. Web. 18 Octubre 2013. <http://www.openframeworks.cc/>.

Introducción y justificación del tema elegido

Como conclusión del estudio sobre instancias previas de interacción por parte del público mediante medios electrónicos se puede determinar que si bien existen usos de tecnologías como microcontroladores, software, celulares, tablets para obras sonoras, visuales o medios mixtos, no existe una herramienta universal de fácil acceso que facilite el diseño y uso de estas tecnologías para los artistas multimedia interesados[23] [24] [25]. La base técnica que hace posible la interacción en estas obras se diseña, construye y programa únicamente para una obra, instalación o performance en particular. Porciones de código pueden ser reutilizadas, pero es necesario generar nuevas aplicaciones y hardware por cada obra. El nuevo código o diseño generado, generalmente no es liberado al público haciendo necesario para los nuevos artistas tener que aprender programación, electrónica y otras diversas disciplinas para poder llevar a cabo este tipo de obras.

Dada esta situación surge la necesidad de comenzar a pensar un sistema universal, de fácil uso, accesible y ampliable para aquellos artistas que quieren diseñar obras interactivas, o agregar interactividad a obras ya existentes. Este sistema debería poder utilizarse en diversos ámbitos, ser portátil y de bajo costo.

La interfaz[26] es aquella parte elemental de la interacción por medios electrónicos que puede abarcar cualquiera de los puntos anteriormente mencionados, o todos ellos a la vez. Se trata de aquel objeto con el que los participantes van a interactuar, obteniendo de ella lo que necesiten para comprender de que trata la obra, qué es lo que deberían estar haciendo convirtiendose en un medio para hacer posible su participación. Se vuelve esencial que este sistema universal se construya en torno a una interfaz que pueda cumplir con estos requisitos.

---

[23] Collins, Nick, Margaret Schedel, and Scott Wilson. *Electronic Music (Cambridge Introductions to Music)*. N.p.: Cambridge UP, 2013. Impreso.

[24] Bongers, Bert. *Physical Interaction in the Electronic Arts*. France: IRCAM, 2000. CD-ROM.

[25] Sommerer, Christa, Lakhmi C. Jain, and Laurent Mignonneau. *The Art and Science of Interface and Interaction Design*. Vol. 1. N.p.: Springer, 2010. Impreso.

[26] Noble, Joshua. *Programming Interactivity, Second Edition*. Sebastopol: O'Reilly, 2012. Impreso.

Las interfaces físicas, constituidas principalmente por elementos tangibles, presentan características que pueden resultar ventajosas para el diseño de obras como lo son la libertad de elección de sensores, poder trabajar la plasticidad del objeto mismo que va a ser utilizado como interfaz o alguno de sus sensores, y poder recibir feedback físico[27]. Este tipo de interfaces, por otro lado, resultan costosas, pocos flexibles para adaptar a otros usos, pueden ocupar mucho espacio físico y pueden limitar severamente la cantidad de personas que participan y la manera en la que participan. La interfaz de control Reactable[28], y otras similares basadas en protocolo TUIO[29], son ejemplos conocidos de este tipo de interfaz, pero que poseen algunas de las limitaciones antes mencionadas, como la restricción del número de participantes y la necesidad de transportar un sistema que además es complejo de montar. Por estas razones este tipo de interfaz queda descartada para el sistema que se plantea en esta tesis.

El uso de cámaras como interfaz, desde webcams hasta cámaras de profundidad como la Asus Xtion[30] o Microsoft Kinect[31], es posible y, si bien presentan una alternativa económica y portátil para la creación de obras interactivas, la limitación que poseen en cuanto a la acción de movimiento, cantidad de personas que puedan estar simultáneamente en el campo visible y espacio necesario para poder usarlo hacen de este tipo de sensores algo poco práctico y poco flexible como para ser considerados para un sistema universal.

Las interfaces virtuales o interfaces gráficas de usuario[32], aquellas que se construyen puramente sobre software y cuya representación gráfica se realiza mediante una pantalla o proyección y algún método de entrada táctil, presentan varias ventajas con respecto a las anteriormente mencionadas. En primera instancia al tratarse de un sistema virtual es posible de forma rápida, y de fácil implementación, modificar la representación gráfica de la información para adaptarse a distintos contextos y para que el usuario pueda comprender más velozmente su uso. Existen diversos tipos de sensores[33] que se pueden interconectar para aumentar las posibilidades en cuanto a las formas de entrada de datos que el usuario tiene a

---

27 Bongers, Bert. *Physical Interaction in the Electronic Arts*. France: IRCAM, 2000. CD-ROM.

28 "Reactable - the Electronic Music Instrument, the Music App and DJ Tool." *Reactable - the Electronic Music Instrument, the Music App and DJ Tool*. N.p., n.d. Web. 17 Nov. 2014. <http://www.reactable.com/>.

29 "TUIO.org." *TUIO*. N.p., n.d. Web. 17 Nov. 2014. <http://www.tuio.org/>.

30 "Asus XTION PRO" *Asus*. N.p., n.d. Web. 17 Nov. 2014. <http://www.asus.com/Multimedia/Xtion_PRO/>.

31 "Microsoft Kinect 2 *Microsoft*. N.p., n.d. Web. 17 Nov. 2014. <http://www.microsoft.com/en-us/kinectforwindows/>.

32 "GUI Definition." *GUI Definition*. N.p., n.d. Web. 17 Nov. 2014. <http://www.linfo.org/gui.html>.

33 links a sensores

su disposición y, mediante las cuales va a participar en obras interactivas. Si bien este tipo de interfaces presentan una gran oportunidad para un sistema universal, su costo hizo difícil el uso generalizado por muchos años.

Los avances tecnológicos de los últimos diez años permitieron que las interfaces táctiles (específicamente pantallas táctiles) fueran más accesibles al público. Actualmente éstas forman parte de un serie de aparatos electrónicos de consumo masivo: computadoras portátiles, GPS, teléfonos celulares y tablets. Estas dos últimas son de particular importancia para este proyecto y se presentan como plataforma ideal para la construcción del sistema propuesto como tesis, ya que son portátiles y de uso cotidiano: es altamente probable que varios espectadores lleven consigo alguno de estos dispositivos. El asentamiento y estandarización de una arquitectura para dispositivos móvil asegura que todos estos equipos compartan varias características, por más que las empresas fabricantes sean distintas. Esto asegura homogeneidad en los diversos dispositivos, haciendo viable su uso como base fundacional para la construcción de este sistema. Las características que comparten todos ellos son:

- Están basados en sistemas operativos estables y bien documentados.[34]
- Permiten la creación de nuevas aplicaciones y ofrecen herramientas y soporte para realizar esta tarea.[35]
- Poseen una serie de sensores integrados[36]:
    - Acelerómetro: Sensa la aceleración en los tres ejes (x, y, z).
    - Pantallas táctiles: Proveen imagen y además permiten utilizarlas como método de entrada de datos. Se pueden obtener coordenadas de la posición de los dedos sobre la pantalla. Existen pantallas que sólo admiten la detección de un dedo, y pantallas *multitouch* que pueden detectar con precisión la presencia de más de un dedo y las coordenadas de cada uno de ellos. Cabe destacar que una pantalla táctil puede ser utilizadas de muchas maneras ya que se pueden dibujar en pantalla información, imagenes, etc.
    - Micrófonos: Permiten la captura de sonido, y por ende su posterior procesamiento por parte de una aplicación. También poseen entrada de micrófono para el uso de micrófonos externos.

---

[34] Siendo Google Android, Apple iOS y Windows Phone los más populares.

[35] Todos ellos poseen Software Development Kit para desarrollar nuevas aplicaciones, según la empresa los costos de este SDK o de la distribución de las aplicaciones puede variar.

[36] "Sensors." *Android Developers*. N.p., n.d. Web. 17 Nov. 2014. <http://source.android.com/devices/sensors/index.html>.

- Cámaras: Permiten la captura de imagenes. Algunos dispositivos poseen dos de ellas, una trasera, para fotos, y una frontal, que ofrece imágenes de menor resolución y está diseñada principalmente para videollamadas.
- GPS (Sistema de posicionamiento global): Mediante el uso de satélites y antenas de telecomunicación permiten obtener con cierta precisión (errores cercanos al metro) las coordenadas en las que se encuentra posicionado el aparato.
- Magnetómetro: Mide el campo geomagnético ambiente en los tres ejes (x, y, z). Su uso más inmediato es como brújula. No está disponible en todos los dispositivos móviles.
- Giroscopio: Mide el ángulo de inclinación en los tres ejes (x, y ,z). No está disponible en todos los dispositivos móviles. Generalmente se usa el acelerómetro para aplicaciones similares.
- Iluminación: Se usan generalmente para sensar la cercanía de una pantalla a la cara de una persona, o para regular automáticamente el brillo de la pantalla.

- Poseen sistemas conexión a redes inalámbricas (WLAN, Wi-Fi), Bluetooth, internet móvil (GPRS, EDGE, 3G, HDSPA, 4G, LTE).
- Poseen elementos para devolver información al usuario: pantalla, parlantes, sistemas de vibración.

Como en toda plataforma informática es necesario tanto el desarrollo y diseño de hardware como de software. En la actualidad los dispositivos móviles tienen tres plataformas base o sistemas operativos como soporte para el software, ellas son: Apple iOS, Windows Phone y Google Android. Este último está basado en GNU/Linux es de código libre y por está razón es muy utilizado por diversos fabricantes de tecnología móvil siendo el sistema operativo móvil más utilizado a nivel mundial. Todos ellos proveen soporte para los sensores anteriormente mencionados ya que forman parte de su arquitectura. Además todos ellos poseen un sistema de distribución de aplicaciones propietario desde el cuál se pueden descargar aplicaciones desarrolladas por terceros, o subir aplicaciones para que otros usuarios las descarguen, cada empresa maneja su distribución de paquetes de manera distinta, y es importante tener en cuenta que si bien se pueden descargar aplicaciones gratuitas, para poder subir y compartir aplicaciones propias en estos sistemas es necesario pagar algún tipo de registro cuyo precio varía según la empresa. Todas estas plataformas deben ser tenidas en cuenta para un sistema que pretende ser versátil y al que puedan acceder la mayor cantidad de personas posibles. Ya que todas tienen el potencial para proporcionar la misma experiencia a los usuarios, y las diferencias que pueden presentarse entre plataformas no afectarían a este sistema en

particular es decir que no existe motivo para dejar fuera a alguna de ellas.

Un sistema para realizar obras sonoras interactivas implica un elemento de comunicación o relación entre los participantes y el resultado sonoro. Habiendo sido determinado el uso de dispositivos móviles como interfaz para los usuarios, y teniendo en cuenta de que se planea usar software de terceros para el trabajo de generación y procesamiento de audio resulta necesario un nexo que vincule a estos dos. Este nexo además de vincular deberá funcionar como un sistema de control y distribución de datos. Teniendo en cuenta que el objetivo de este sistema es la interacción en obras sonoras, y el sonido transcurre en el tiempo, es de gran importancia que este control y distribución de información se realice en una línea de tiempo con contador de tiempo (Fig. 1), similar a la que se encuentra en software de edición de audio.

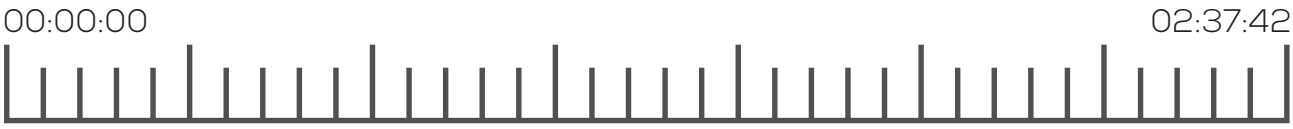


Figura 1. Línea de tiempo con divisiones equidistantes

Ésta constará de una o más pistas y cada usuario estará vinculado directamente a una de estas pistas (Fig. 2).

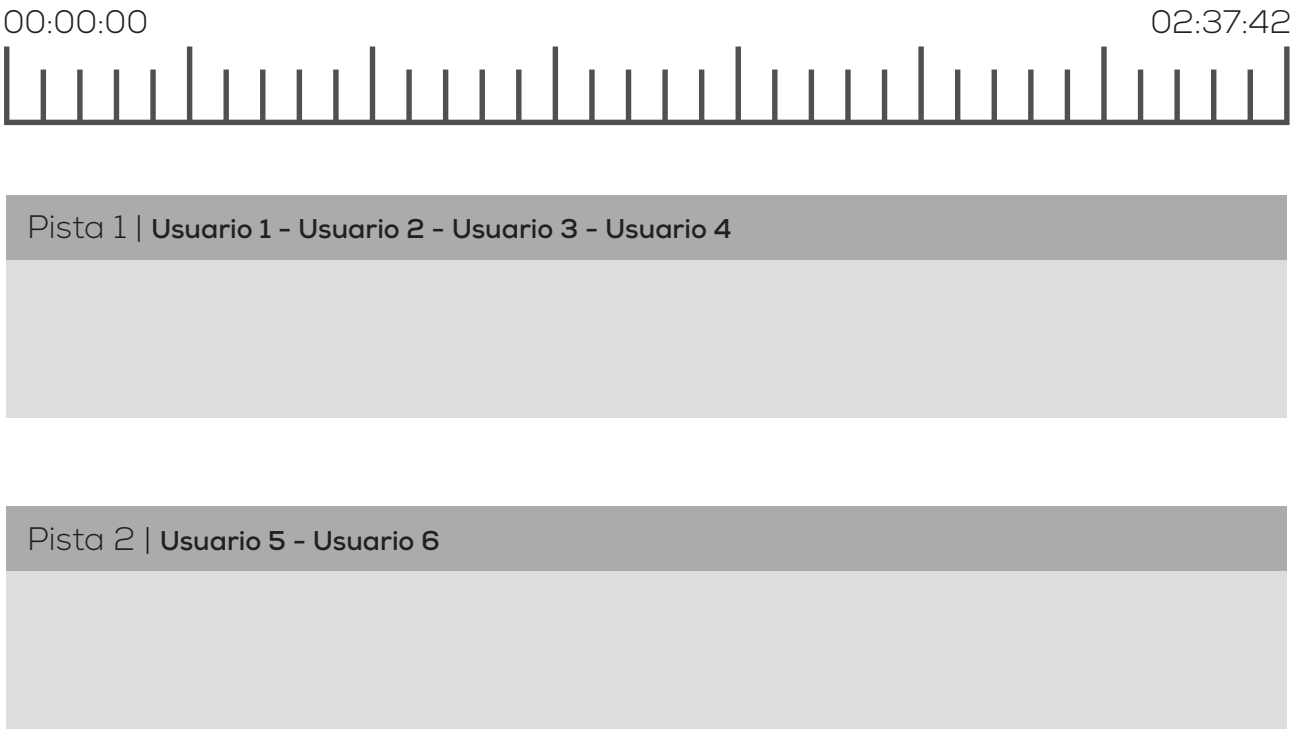


Figura 2. Línea de tiempo y pistas con usuarios asignados a cada una de ellas.

Sobre estas pistas se dispondrán bloques de interacción. Estos bloques tendrán varias funciones simultáneamente: por un lado funcionarán como interruptores o compuertas, es decir que cuando el contador de tiempo coincida con la posición sobre la línea de tiempo de alguno de estos bloques, estos activarán remotamente el envío de datos de alguno de los sensores de los clientes conectados a la pista que contienen a estos bloques (Fig. 3).

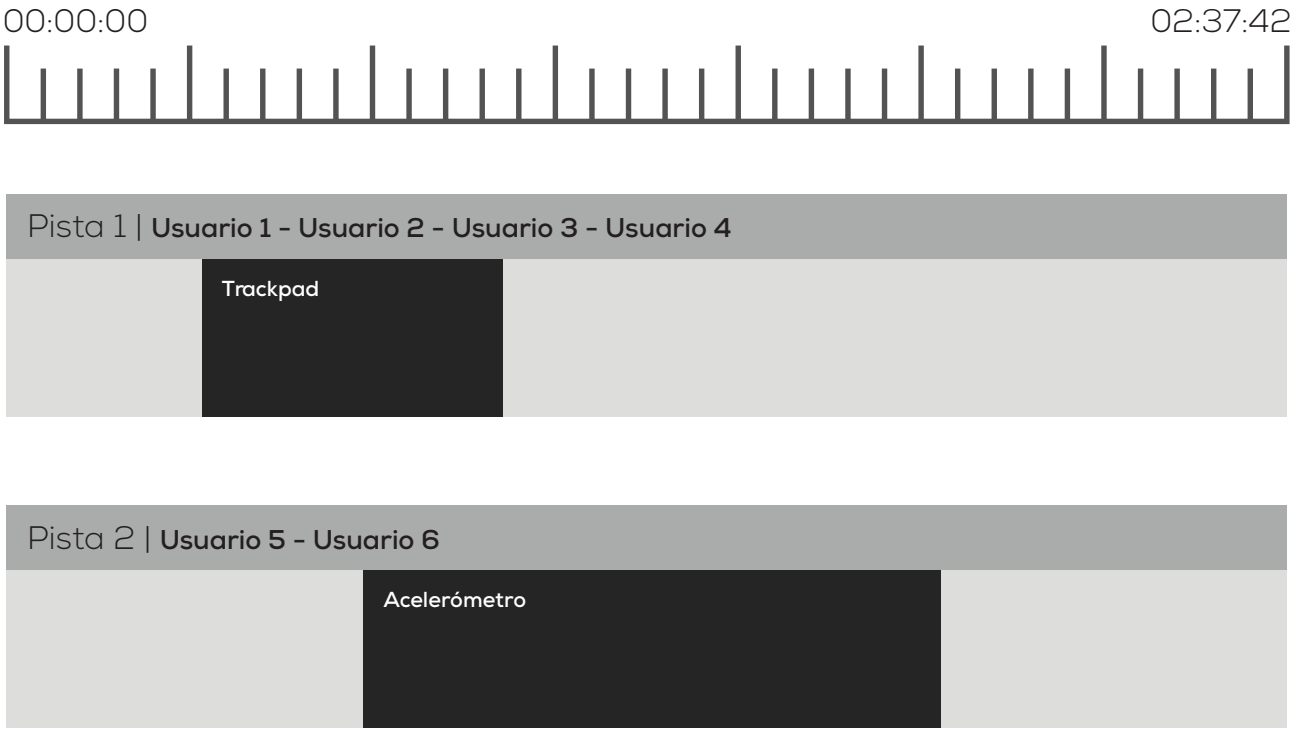


Figura 3. Línea de tiempo y pistas con usuarios asignados a cada una de ellas.

Qué sensor debería enviar datos también quedará determinado por el bloque. Finalmente, estos bloques de interacción enviarán información relevante sobre qué deberían estar haciendo, de qué se trata la obra, o que están controlando en ese momento. Esta información será ingresada por el artista y quedará a su criterio cómo utilizarla. El conjunto de pistas y bloques conformarán una sección. Estas secciones podrán tener distintas duraciones de tiempo, y pueden usarse de distintas formas. El uso más evidente resulta puramente organizativo, trabajar secciones musicales resulta práctico e intuitivo, otros usos posibles pueden ser la posibilidad de cambiar de sección cuando se haya llegado a algún objetivo sonoro, de esta manera este nexo podrá ser controlado externamente por otras aplicaciones para señalar en qué momento es necesario cambiar de sección.

Como se mencionó anteriormente se tomó la decisión de que este sistema pueda

comunicarse con software de terceros para aprovechar la existencia de programas dedicados al audio asegurando un set de herramientas estables, probados, conocidos y familiares para el artista. Esto implica que instancias previas de música generadas con estas herramientas pueden reutilizarse o adaptarse a obras interactivas. También implica que el artista puede diseñar sus propias herramientas para el procesamiento de audio digital si así lo quisiera y conectarlo a este sistema. Mediante mensajes OSC estos programas podrán controlar algunas funciones del software de control, tales como cambiar de sección o detener por completo la obra. Por ejemplo, aprovechando las capacidades que posee Pure Data para el análisis de audio, y el control de datos, éste podría enviar un mensaje al software de control para que cambie de sección cuando se haya cumplido algún objetivo sonoro o musical.

Protocolos para la intercomunicación de aplicaciones

Para que la comunicación entre este sistema de control desarrollado y aplicaciones ya existentes sea viable es necesario conocer que métodos de intercomunicación pueden disponerse. MIDI[37] (Musical Instrument Digital Interface) y OSC (acrónimo de Open Sound Control)[38] resultan los protocolos más utilizados para el envío y recepción de datos. El primero, que fue estandarizado en 1983, presenta algunas dificultades para aplicarse a este proyecto. Los tipos de mensajes que pueden enviarse están predeterminados y el protocolo es poco flexible, sólo permite enviar datos numéricos, quedando fuera la posibilidad de enviar cadenas de texto necesarias para enviar y recibir información de y hacia los espectadores. OSC se desarrolla con la intención de generar un protocolo para usos similares a MIDI, pero mejorando la organización de los mensajes, ampliando las posibilidades de los datos que se pueden enviar y los medios por el cual se pueden transmitir. Es posible enviar y recibir mensajes OSC a través de redes locales de cualquier tipo, a través de internet, o por puertos serie utilizando SLIP (Serial Line Internet Protocol)[39] para el encapsulado de los mensajes.

---

[37] Huber, David Miles. *The MIDI Manual: A Practical Guide to MIDI in the Project Studio*. Boston: Focal, 2007. Impreso.

[38] "Open Sound Control." *Opensoundcontrol.org*. N.p., n.d. Web. 17 Nov. 2014. <http://opensoundcontrol.org/>.

[39] "A NONSTANDARD FOR TRANSMISSION OF IP DATAGRAMS OVER SERIAL LINES: SLIP." N.p., n.d. Web. 17 Nov. 2014. <http://www.rfc-editor.org/rfc/rfc1055.txt>.

Estructura de un mensaje OSC

Los mensajes OSC están compuesto por una address (dirección) que comienzan con una barra diagonal (/) seguido de texto. Se pueden sumar barras diagonales para generar sub direcciones como si se tratara de una carpeta en un sistema operativo.

Por ejemplo:

/touchscreen/x
/touchscreen/y

Luego de la address le siguen los argumentos separados por un espacio. Un mensaje OSC puede tener más de un argumento, y estos argumentos pueden ser de diversos tipos: cadenas de texto, números enteros, números con coma flotante.

Ejemplo de mensaje OSC con tres argumentos:

/touchscreen 125.20 56 usuario1

La especificación OSC[40] establece que los mensajes sean transmitidos a través del protocolo UDP (User Datagram Protocol)[41]. Una actualización a esta especificación (OSC 1.1), la última versión hasta el momento, agrega TCP[42] (Transmission Control Protocol) para el envío de mensajes[43]. Estos deben estar encapsulados utilizando otro protocolo denominado SLIP (Serial In-Line Protocol). Esta especificación de OSC todavía no está en uso, y no existe librería que contemple el envío de mensajes OSC por TCP, las implementaciones que están en uso en algunos programas son adaptaciones hechas por los propios desarrolladores.

---

[40] "Opensoundcontrol.org." *The Open Sound Control 1.0 Specification*. N.p., n.d. Web. 17 Nov. 2014. <http://opensoundcontrol.org/spec-1_0>.

[41] "UDP (User Datagram Protocol)." *Tech-FAQ*. N.p., n.d. Web. 18 Nov. 2014. <http://www.tech-faq.com/udp.html>.

[42] "What Is TCP/IP?" *Microsoft Tech Net*. N.p., n.d. Web. 18 Nov. 2014. <http://technet.microsoft.com/en-us/library/cc775418%28v=WS.10%29.aspx>.

[43] "Features and Future of Open Sound Control Version 1.1 for NIME | CNMAT." *Features and Future of Open Sound Control Version 1.1 for NIME | CNMAT*. N.p., n.d. Web. 17 Nov. 2014. <http://cnmat.berkeley.edu/publication/features_and_future_open_sound_control_version_1_1_nime>

En cuanto a la presentación más práctica del sistema de control desarrollado, sería ideal contar con paquetes ejecutables que no requieran de otros programas o dependencias, es decir, que para aquellos interesados en utilizarlo baste con descargar la aplicación y ejecutarla. Esto es posible si se utiliza un framework contenido multiplataforma para el desarrollo íntegro de la aplicación, porque aseguraría que todos las librerías necesarias ya están incluidas, probadas y funcionando correctamente. Es importante que el sistema de distribución de estas aplicaciones sea de fácil acceso y uso, y que todo el sistema requiera la menor configuración posible para facilitarle tanto a los diseñadores de obras, como a los espectadores, el poder usarlas con éxito.

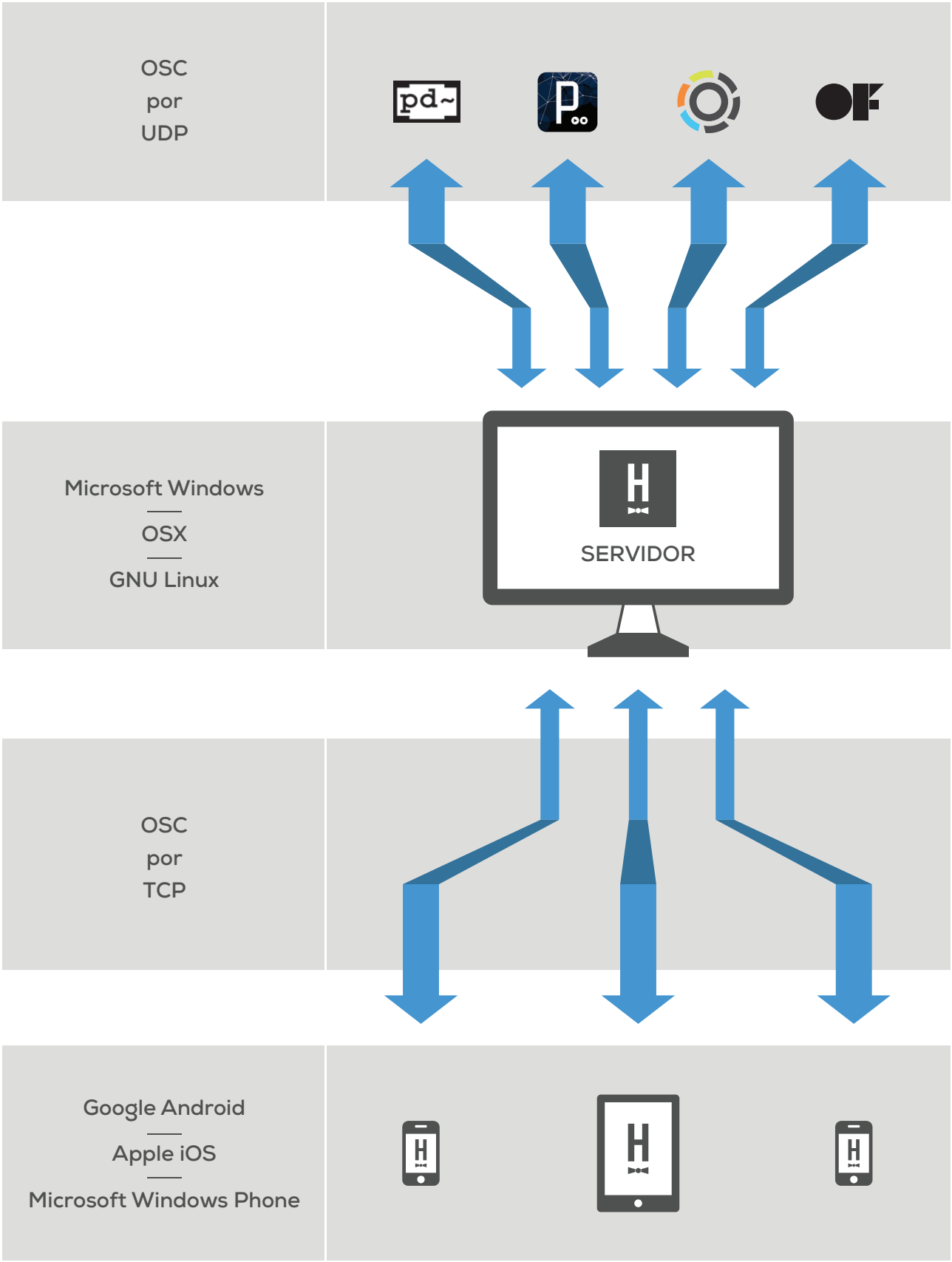


Figura 4. Gráfico esquemático del sistema propuesto.

Consideraciones para la elección de un framework

¿Qué es un framework?

Un framework consta de una serie de librerías que ayudan a simplificar la programación. Facilitan el trabajo con redes, el manejo de archivos, generación y transformación de imagen y sonido, soporte multiplataforma entre otras tareas y requisitos frecuentes relacionadas con la multimedia y la computación en general.

Frameworks existentes

Si bien existen varios frameworks y librerías de este tipo, se contemplaron tres en particular por estar relacionadas con la producción de arte multimedia y porque proveen varias herramientas útiles para este proyecto. Estas son Processing[44] , Cinder[45] y openFrameworks[46] , todas ellas gratuitas y de código abierto. La primera se trata de un paquete que incluye su propio IDE (Integrated Development Environment)[47] y la programación se realiza en Java, provee una amplia variedad de librerías y la documentación es clara y extensa. Cinder y openFrameworks están basados en librerías escritas en C++ y como otras basadas en este lenguaje de programación no tienen IDE propio, utilizan entornos existentes siendo los recomendados por estas plataformas y más populares:

- Microsoft Visual Studio[48] que es propietario y pago, tiene una versión gratuita denominada Express[49], posee un excelente *debugger*[50] ayudando a localizar errores con

[44] "Processing.org." *Processing.org*. Web. 18 Octubre 2013. <http://processing.org/>.

[45] "Cinder." *Cinder*. N.p., n.d. Web. 17 Nov. 2014. <http://libcinder.org/>.

[46] "openFrameworks." *openFrameworks*. Web. 18 Octubre 2013. <http://www.openframeworks.cc/>.

[47] Entorno de desarrollo integrado por su sigla en inglés (IDE, Integrated Development Environment) es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios.

[48] "Visual Studio | MSDN." *Visual Studio | MSDN*. N.p., n.d. Web. 17 Nov. 2014. <http://msdn.microsoft.com/en-us/vstudio/aa718325.aspx>.

[49] "Visual Studio Express." *Visual Studio Express*. N.p., n.d. Web. 17 Nov. 2014. <http://www.visualstudio.com/en-us/products/visual-studio-express-vs.aspx>.

[50] Aplicación o herramienta que permite la ejecución controlada de un programa o un código, para seguir cada instrucción ejecutada y localizar así bugs o errores.

mayor velocidad reduciendo los tiempos de desarrollo. Solamente puede utilizarse en distribuciones de Windows.

- Code::Blocks[51] es un IDE gratuito y libre, multiplataforma, puede instalarse en Microsoft Windows, Apple Mac OS, y diversas distribuciones de Linux.
- Apple XCode[52] es propietario y gratuito. Solamente puede utilizarse en plataformas Mac, siendo el IDE más popular entre los usuarios de esta plataforma.
- Eclipse / ADT[53] [54] es gratuito y es la entorno de desarrollo más utilizado para la programación en Java. Es necesario para compilar aplicaciones para la plataforma Android.

¿Qué necesita este proyecto?

Es importante para este proyecto que el resultado (el sistema de control y administración) sea sencillo de usar, y fácil de aprender, con la mínima configuración necesaria posible. Para ello lo ideal sería una aplicación ejecutable, que no dependa de la instalación de otro software. Un programa de estas características aseguraría que el artista no debería buscar e instalar software adicional, resolver dependencias, actualizar y mantener un paquete complejo de programas. En este aspecto lo que se busca es producir una aplicación de código nativo, cuyo resultado sea un ejecutable o binario. Es necesario que este software pueda correr en varias plataformas para asegurar de que ningún artista o espectador quede sin poder participar o crear obras por no poseer el sistema operativo adecuado, o al menos poder alcanzar al mayor público posible. Los sistemas operativos para computadora más utilizados son Microsoft Windows, GNU/Linux (tiene varias distribuciones siendo Ubuntu una de las más reconocidas), y Apple Mac OS; y en el caso de sistemas operativos para móviles se encuentran Google Android, Apple iOS y Windows Phone a la cabecera. Idealmente este proyecto debería ser compatible con todas estas plataformas, o al menos, que sea posible su expansión a futuro. Existen otros sistemas operativos tanto para computadora y móviles como Solaris[55], BSD[56],

---

51 "Code::Blocks." *Code::Blocks*. N.p., n.d. Web. 17 Nov. 2014. <http://www.codeblocks.org/>.

52 "Apple Developer." *Xcode*. N.p., n.d. Web. 17 Nov. 2014. <https://developer.apple.com/xcode/>.

53 "ADT Plugin." *Android Developers*. N.p., n.d. Web. 17 Nov. 2014. <https://developer.android.com/tools/sdk/eclipse-adt.html>.

54 "Are You Ready for Java™ 8?" *Eclipse Luna*. N.p., n.d. Web. 17 Nov. 2014. <http://eclipse.org/>.

55 "Oracle Solaris 11." *Solaris - Engineered for Cloud*. N.p., n.d. Web. 17 Nov. 2014. <http://www.oracle.com/us/products/servers-storage/solaris/solaris11/overview/index.html>.

56 "The FreeBSD Project." *The FreeBSD Project*. N.p., n.d. Web. 18 Nov. 2014. <http://www.freebsd.org/>.

Ubuntu OS[57], Firefox OS[58] pero son pocos los usuarios que utilizan éstos y pocas las plataformas de desarrollo que los soportan, por estos motivos quedan descartados. Otra necesidad de este proyecto es la comunicación inalámbrica, y por ende es necesario soporte para protocolos de comunicación por red como TCP, y UDP, así como también es necesario soporte para OSC, protocolo elegido para el encapsulado de mensajes.

Para el desarrollo de este proyecto, sin embargo, no se cuentan con todos estos dispositivos, plataformas y sistemas operativos, así que se centrará en los sistemas operativos Microsoft Windows y Google Android.

Comparando los frameworks propuestos

Para tomar la decisión sobre que framework utilizar se analizaron las prestaciones que ofrecen los paquetes propuestos para contrastarlas con los requisitos que tiene este proyecto.

Processing

Se trata de una combinación de lenguaje de programación e IDE para el desarrollo de aplicaciones o applets orientadas a la multimedia. El lenguaje de programación está basado en Java y comparten gran parte de la sintaxis. Al tratarse de un paquete orientado a la multimedia contiene varias librerías para la generación y procesamiento de imagen. Se pueden agregar librerías creadas por terceros para ampliar sus capacidades, entre ellas se encuentran herramientas para el trabajo en redes, tanto por TCP como UDP, manejo de paquetes OSC, generación y procesamiento de audio. Posee soporte para iOS y Android desde la versión 2.3.4. Al tratarse de código interpretado (por su relación con Java) las aplicaciones generadas no son nativas, y necesitan de dependencias. Tiene soporte para Windows, Mac OS X, y GNU/Linux.

---

[57] "Ubuntu Now on Google Cloud Platform." *The Leading OS for PC, Tablet, Phone and Cloud*. N.p., n.d. Web. 18 Nov. 2014. <http://www.ubuntu.com/>.

[58] "Firefox OS - El Teléfono Adaptativo - Grandes Características, Aplicaciones Y Mucho Más Para Teléfonos Inteligentes." *Mozilla*. N.p., n.d. Web. 18 Nov. 2014. <https://www.mozilla.org/es-AR/firefox/os/>.

Cinder

Como Processing, Cinder también está orientado a contenidos multimedia, y tiene varias librerías pero su implementación es distinta. Consta de una serie de librerías de C++ que permiten a los desarrolladores realizar diversas tareas de forma sencilla. No posee IDE propio y está pensado que el programador utilice Microsoft Visual C++ para Windows o Apple XCode para Mac OS X. GNU/Linux no está soportado aún. Siendo un proyecto reciente todavía no tiene soporte para todas los sistemas operativos móviles pudiendo compilarse solo en iOS. Al programar en C++ se puede producir código nativo, aplicaciones binarias, es decir programas ejecutables de alto rendimiento.

openFrameworks

Comparte muchos aspectos de Cinder (C++, capacidad de generar aplicaciones nativas, está compuesto de varias librerías) pero posee soporte para una mayor cantidad de sistemas operativos, además de Windows y Mac OS X, se puede programar y compilar para las plataformas GNU/Linux, Google Android, Apple iOS y Windows Phone. Soporta una mayor cantidad de IDEs: Code::Blocks, XCode, Visual Studio, Eclipse (ADT - Android Developer Tools) para la programación de aplicaciones Android. Existe una librería para el trabajo con líneas de tiempo y pistas única para este framework.[59]

[59] "YCAMInterlab/ofxTimeline." *GitHub*. N.p., n.d. Web. 16 Nov. 2014. <https://github.com/YCAMInterlab/ofxTimeline>.

Framework	Processing	Cinder	openFrameworks
Lenguaje de programación	Processing (basado en Java)	C++	C++; Java (para aplicaciones Android)
¿Puede producir aplicaciones ejecutables?	No	Si	Si
IDE's soportados	Processing	Microsoft Visual Studio, Apple XCode	Microsoft Visual Studio, Apple XCode, Code::Blocks, Eclipse(para Android)
Sistemas operativos soportados	Mac OS X, Windows, GNU/Linux, Android	Mac OSX, Windows	Mac OSX, Windows, GNU/Linux, Android, iOS, Windows Phone
Mínima versión de Android soportada	2.3.3	-	2.2
¿Compatible con trabajo en redes?	Si	Si	Si
¿Compatible con OSC?	Si	Si	Si
¿Tiene librería para el trabajo con líneas de tiempo?	No	No	Si

Figura 5. Cuadro comparativo de frameworks

Dadas estas características (ver Figura 5), openFrameworks resulta el framework más adecuado para el desarrollo de este sistema. Cumple con el requisito de soporte de las plataformas Windows y Android como base para construir las dos aplicaciones que conformarán el sistema, y además provee expansión a todos los sistemas operativos a los que se pretende alcanzar en expansiones futuras. Tiene una gran cantidad de librerías, algunas de ellas resultan muy útiles para este proyecto en particular, como es el caso del trabajo con líneas de tiempo, de uso muy específico y por ende poco habitual.

openFrameworks es un framework, API (Application Programming Interface) o conjunto de librerías y herramientas informáticas comunes para el desarrollo de aplicaciones, programas, instalaciones y performance multimedia, de código abierto y gratuito. El lenguaje de programación principal que se utiliza para el desarrollo es C++, lenguaje orientado a objetos estandarizado en 1989. Otro lenguaje que se utiliza en conjunto con C++ es Java, también orientado a objetos, necesario para la programación de aplicaciones Android dado que es el lenguaje principal utilizado por esta plataforma. Esta mezcla de lenguajes es posible gracias a JNI (Java Native Interface) que permite al código Java que está corriendo en un máquina virtual de Java (JVM, Java Virtual Machine) llamar y ser llamado por código, librerías o aplicaciones nativas (aquellas específicas de una plataforma)[60], en este caso las librerías en C++ de openFrameworks. Aunque esté apuntado a la creación de obras artísticas que requieran de un soporte informático, estas herramientas pueden utilizarse en cualquiera aplicación que requiera acceso al hardware de un sistema informático, simplificando su uso además de proveer funciones para tareas comunes, repetitivas y complejas que se realizan durante el proceso de desarrollo.

openFrameworks Consta con librerías para el manejo de gráficos, textos, 3D, audio, archivos, etc. Alguna de ellas son:

- OpenGL[61], GLEW[62], GLUT[63], libtess2[64] y cairo[65] para trabajo con gráficos

---

[60] "Role of the JNI". *The Java Native Interface Programmer's Guide and Specification*. Retrieved 2008-02-27.

[61] "The Industry's Foundation for High Performance Graphics." *OpenGL News*. N.p., n.d. Web. 18 Nov. 2014. <https://www.opengl.org/>.

[62] "The OpenGL Extension Wrangler Library." *GLEW:*. N.p., n.d. Web. 18 Nov. 2014. <http://glew.sourceforge.net/>.

[63] "GLUT - The OpenGL Utility Toolkit." *GLUT - The OpenGL Utility Toolkit*. N.p., n.d. Web. 18 Nov. 2014. <https://www.opengl.org/resources/libraries/glut/>.

[64] "Libtess2 - Game and Tools Oriented Refactored Version of GLU Tesselator. - Google Project Hosting." *Libtess2 - Game and Tools Oriented Refactored Version of GLU Tesselator. - Google Project Hosting*. N.p., n.d. Web. 18 Nov. 2014. <https://code.google.com/p/libtess2/>.

[65] "Cairographics.org." *Cairographics.org*. N.p., n.d. Web. 18 Nov. 2014. <http://cairographics.org/>.

- rtAudio[66], PortAudio[67], OpenAL[68] Kiss FFT[69] y FMOD[70] para entrada de audio, análisis y salida.
- FreeType[71] para cargar fuentes tipográficas.
- FreeImage[72] para el guardado y cargado de imágenes.
- Quicktime[73], GStreamer[74] y videoInput[75] para la captura y reproducción de video
- Poco[76] es un paquete tiene diversas utilidades: trabajo en redes, manejo de XML, threading, entre otras.
- OpenCV[77] para trabajar con computer vision[78].
- Assimp[79] para cargar modelos 3D.

Las librerías están implementadas de tal manera que el código escrito puede compilarse para diversas plataformas. Oficialmente openFrameworks provee soporte para las plataformas Windows, OSX, Linux, iOS y Android; Windows Phone está soportada por terceros al momento de escribir este proyecto[80], pero está planeada su inclusión en el paquete oficial. Soporta también cuatro IDEs (Apple XCode, Code::Blocks, Microsoft Visual Studio y Eclipse).

---

66 "Latest Updates (Version 4.1.1)." *The RtAudio Home Page*. N.p., n.d. Web. 18 Nov. 2014. <http://www.music.mcgill.ca/~gary/rtaudio/>.

67 "PortAudio - an Open-Source Cross-Platform Audio API." *PortAudio - an Open-Source Cross-Platform Audio API*. N.p., n.d. Web. 18 Nov. 2014. <http://www.portaudio.com/>.

68 "OpenAL." *OpenAL*. N.p., n.d. Web. 18 Nov. 2014. <http://www.openal.org/>.

69 "Keep It Simple, Stupid!" *Keep It Simple, Stupid!* N.p., n.d. Web. 18 Nov. 2014. <http://kissfft.sourceforge.net/>.

70 "FMOD." *FMOD*. N.p., n.d. Web. 18 Nov. 2014. <http://www.fmod.org/>.

71 "FreeType." *The Project*. N.p., n.d. Web. 18 Nov. 2014. <http://freetype.org/>.

72 "What Is FreeImage?" *The FreeImage Project*. N.p., n.d. Web. 18 Nov. 2014. <http://freeimage.sourceforge.net/>.

73 "QuickTime. The Best Place to Play." *Apple*. N.p., n.d. Web. 18 Nov. 2014. <http://www.apple.com/quicktime/>.

74 "News - GStreamer Core, Plugins and RTSP Server 1.4.4 Stable Release." *GStreamer: Open Source Multimedia Framework*. N.p., n.d. Web. 18 Nov. 2014. <http://gstreamer.freedesktop.org/>.

75 "VideoInput Library." *VideoInput Library*. N.p., n.d. Web. 18 Nov. 2014. <http://www.muonics.net/school/spring05/videoInput/>.

76 "Stable Release 1.4.7 and Development Release 1.5.4 Available." *Poco Project*. N.p., n.d. Web. 18 Nov. 2014. <http://pocoproject.org/>.

77 "OpenCV | OpenCV." *OpenCV | OpenCV*. N.p., n.d. Web. 18 Nov. 2014. <http://opencv.org/>.

78 Métodos para adquirir, procesar, analizar y comprender imágenes del mundo físico para producir información numérica o simbólica.

79 "Open Asset Import Library." *Open Asset Import Library*. N.p., n.d. Web. 18 Nov. 2014. <http://assimp.sourceforge.net/>.

80 "OpenFrameworks Developers Can Now Create Universal Apps for Windows - MS Open Tech." *MS Open Tech*. N.p., n.d. Web. 17 Nov. 2014. <http://msopentech.com/blog/2014/07/03/openframeworks-developers-can-now-create-universal-apps-for-windows/>.

Se distribuye bajo la licencia MIT[81] que permite a los desarrolladores utilizar el software generado con estas herramientas en cualquier contexto, sea comercial o gratuito, público o privado, de código abierto o cerrado, por ende no requiere que el código sea liberado.

La documentación de este framework es extensa, pero no tan exhaustiva como lo es la de Processing, depende fuertemente de las comunidades dedicadas tanto a oF como a la programación en general[82] [83]. Fue necesario entonces aprender C++ y Java y cómo comunicar estos lenguajes a través de la interfaz nativa de Java además de cómo utilizar este framework en contexto. Sumado a esto fue necesario aprender también cómo es la estructura de una aplicación Android, como se generan menúes, como se puede acceder a los sensores y otras partes del hardware que provee esta plataforma.

Alternativas para la comunicación inalámbrica

Para que la comunicación entre los espectadores, es decir sus dispositivos, y el nexo, es decir, el servidor, sea posible es necesario utilizar un protocolo para la comunicación de datos en redes. Este protocolo debe ser compatible con alguno de los medios de transmisión de datos inalámbricos que poseen los dispositivos (Bluetooth, WiFi/WLAN, GPRS/EDGE/3G/4G)[84]. WiFi resulta la opción que más adecuada para este proyecto, funciona a nivel local, en espacios abiertos y cerrados, es estable, no depende de la intensidad de señal a una antena de telecomunicaciones. El uso de IP's y máscaras permite darle una identidad a cada elemento conectado a la red, y permite acceder a dos protocolos de comunicación de datos muy útiles: TCP y UDP.

---

81 "The MIT License (MIT)." *The Open Source Initiative*. N.p., n.d. Web. 18 Nov. 2014.

82 "OpenFrameworks - Forums." *OpenFrameworks*. N.p., n.d. Web. 17 Nov. 2014. <http://forum.openframeworks.cc/>.

83 "Stack Overflow." *Stack Overflow*. N.p., n.d. Web. 17 Nov. 2014. <http://stackoverflow.com/>.

84 "Connectivity." *Android Developers*. N.p., n.d. Web. 17 Nov. 2014. <https://developer.android.com/guide/topics/connectivity/index.html>.

TCP vs UDP

Un protocolo de red es un conjunto de reglas que las comunicaciones de datos, a través de una red, siguen para completar una transacción, es decir, para entregar o recibir datos. TCP y UDP[85] son los dos protocolos que se encargan de administrar la conexión entre dos computadoras, o sistemas informáticos, más utilizados, documentados e implementados[86]. TCP es considerado un protocolo confiable, ya que comprueba constantemente que los paquetes se envíen y se reciban de forma correcta, asegurando su transmisión y está orientado a la conexión, es decir que un cliente debe estar conectado a un servidor antes de poder enviar o recibir paquetes de datos. UDP por otro lado no requiere de este tipo de conexión, y por ende no comprueba que los datos enviados se hayan recibido correctamente, o que los paquetes recibidos no estén corruptos. Esta característica, a pesar de transformarlo en un protocolo no confiable, hace que la velocidad de transmisión sea muy alta, ideal para trabajar en aplicaciones de tiempo real. Frecuentemente es utilizado para la transmisión de audio y/o video en tiempo real (Streaming). OSC usa este protocolo para el envío de sus mensajes.

TCP resulta ideal para utilizar como protocolo de comunicación entre los espectadores y el nexo (servidor) tanto porque utiliza una arquitectura servidor-cliente, que es exactamente lo que necesita este proyecto, como por la seguridad que provee al comprobar la integridad de los paquetes. Ya que la base de una obra sonora interactiva es el input de los usuarios que interactúen, es elemental que ese input llegue a destino correctamente. OSC está diseñado para trabajar con UDP como protocolo, la especificación v1.1 de OSC establece que puede utilizarse TCP si los mensajes se encapsulan en SLIP, pero como se comentó anteriormente esta especificación todavía no es de uso general, y resulta necesario adaptar la implementación de OSC que se use para lograr enviar datos a través de este protocolo[87].

Como formato para los mensajes de todo el sistema se eligió OSC que es un protocolo para la comunicación entre distintos tipos de dispositivos como computadoras, tablets, microcontroladores, sintetizadores, entre otros. La especificación más utilizada de OSC es la

---

85 "TCP/IP Protocol Fundamentals Explained with a Diagram." *The Geek Stuff RSS*. N.p., n.d. Web. 17 Nov. 2014. <http://www.thegeekstuff.com/2011/11/tcp-ip-fundamentals/>.

86 Hallberg, Bruce A. *Networking: A Beginner's Guide*. 6th ed. N.p.: McGraw-Hill Osborne Media, 2013. Impreso.

87 "Features and Future of Open Sound Control Version 1.1 for NIME | CNMAT." *Features and Future of Open Sound Control Version 1.1 for NIME | CNMAT*. N.p., n.d. Web. 17 Nov. 2014. <http://cnmat.berkeley.edu/publication/features_and_future_open_sound_control_version_1_1_nime>.

1.0, que es la que acompaña a openFrameworks en la extensión ofxOsc, esta especificación está diseñada para transmitir paquetes a través del protocolo UDP por lo que no se utilizó la extensión exactamente como viene si no que se utilizaron porciones del código y se adaptó para formatear y enviar paquetes a través de TCP, esta adaptación es experimental y de uso exclusivo para esta prueba de concepto, idealmente debería adecuarse a la especificación 1.1 de OSC, que es la última actualización disponible del protocolo. Ésta especificación indica que los paquetes OSC a transmitirse por TCP (o cualquier otro stream-oriented protocol) deben utilizar SLIP (RFC1055)[88] como método de enmarcamiento para delimitar los mensajes. Proyectos como netPD[89] se adecúan a esta especificación, pero todavía no está establecida como la norma y openFrameworks no la soporta nativamente: al momento de escribir este proyecto no tiene extensión alguna que lo haga.

Para la comunicación interna entre servidor y aplicaciones multimedia como Pure Data, MAX/MSP, Processing, etc. se mantiene la implementación de OSC que utiliza UDP ya que existe poco o ningún riesgo de que los paquetes se pierdan en una conexión de estas características.

Desarrollo

Se tomó la decisión de programar la aplicación en C++ por ser un lenguaje de programación capaz de producir código nativo, es decir, código que una vez compilado para una plataforma es capaz de correr nativamente sin necesidad de un intérprete a diferencia de Java, Python o C#. Esto implica mejoras en el rendimiento general del software, pero dificulta la compatibilidad entre distintas plataformas operativas como Windows, Linux y Mac. Para superar este problema se eligió openFrameworks que consta de una serie de librerías para C++ que proveen herramientas para simplificar la programación y asegurarse de que el software pueda correr en más de una plataforma, además al tratarse de una librería de código abierto openFrameworks cuenta con varios colaboradores que proveen extensiones útiles para el objeto de este trabajo. Para el desarrollo de estas aplicaciones se utilizaron los addons

[88] "A NONSTANDARD FOR TRANSMISSION OF IP DATAGRAMS OVER SERIAL LINES: SLIP." N.p., n.d. Web. 17 Nov. 2014. <http://www.rfc-editor.org/rfc/rfc1055.txt>.

[89] "NetPD : Protocol." N.p., n.d. Web. 17 Nov. 2014. <http://www.netpd.org/Protocol>.

ofxNetwork, ofxPd[90], ofxOsc, ofxTimeline[91] (y dependencias en forma de otras extensiones de openFrameworks) y las extensiones propias para Android que provee openFrameworks: ofxAndroid y ofxAccelerometer.

La programación se realizó íntegramente en Windows 7. Para trabajar en esta plataforma y poder producir aplicaciones ejecutables, como requiere la aplicación servidor, openFrameworks tiene como IDE (Integrated Development Environment - Entorno de desarrollo integrado) [92]recomendado y soportado a Code::Blocks (se uso la versión 12.11) que se trata de software gratuito de código abierto. La programación para la aplicación cliente requiere de otro IDE, al tratarse de una aplicación para Android es necesario utilizar el Android SDK (Software Development Kit)[93] distribuido por Google que incluye una distribución de Eclipse (Entorno gratuito ampliamente utilizado en desarrollo de aplicaciones Java), es necesario también el Android NDK (Native Development Kit) implementación de JNI (Java Native Interface) que permite integrar código nativo (por ej. C, C++) con código Java.

Al comenzar con el proyecto la última versión de openFrameworks disponible era la versión 0.8.0 que presentaba problemas de compatibilidad para compilar aplicaciones de Android en Windows, meses después esto fue corregido en el sitio oficial de OF y luego oficialmente soportado en la versión 0.8.1, pero antes de estas correcciones el desarrollo experimental se hizo en Xubuntu (Distribución de Ubuntu Linux con escritorio Xfce) y una vez resuelta la instalación del software necesario y sus dependencias en este sistema operativo se escribió una guía para repetir los pasos necesarios para realizar esta instalación y poder compilar una aplicación de Android en Linux con openFrameworks (Ver Anexo "**Instalación de openFrameworks en Linux**").

Una vez que estos errores fueron resueltos en el código de openFrameworks el trabajo en la aplicación Android fue trasladado a Windows 7 y se escribió otra guía para ordenar las extensiones que son necesarias para el proyecto y sus dependencias, así como instrucciones

---

[90] "Danomatika/ofxPd." *GitHub*. N.p., n.d. Web. 17 Nov. 2014. <https://github.com/danomatika/ofxPd>.

[91] "YCAMInterlab/ofxTimeline." *GitHub*. N.p., n.d. Web. 16 Nov. 2014. <https://github.com/YCAMInterlab/ofxTimeline>.

[92] Se trata de un programa diseñado específicamente para programar, constan principalmente de un editor de texto con funciones extendidas, herramientas para compilar, control de errores de sintaxis, ayudas visuales para la programación entre otras herramientas útiles para el desarrollo de aplicaciones.

[93] "Get the Android SDK." *Android SDK*. N.p., n.d. Web. 17 Nov. 2014. <https://developer.android.com/sdk/index.html?hl=i>.

para la búsqueda e instalación de dos DLL necesarios por las extensiones (openAL y libflac[94]) y una librería necesaria para compilar el código que no viene incluida en Code::Blocks (pthreads). (Ver Anexo "**Instalación de librerías necesarias para este proyecto y solución a problemas encontrados**").

El dispositivo móvil en el que se realizó la mayor parte de la producción fue un teléfono celular genérico (modelo v1277) con un procesador MTK6577 dual-core, 512 MB de RAM, pantalla de 4,3" y Android ICS 4.0.4 como sistema operativo.

Una vez resueltos los problemas de instalación y dependencias la programación se inició haciendo pruebas con las diferentes extensiones y con openFrameworks mismo para aprender su funcionamiento general y, paralelamente se recurrió a bibliografía y videos sobre programación en C++.

Extensiones y librerías relevantes

A continuación una breve reseña de las principales librerías y extensiones que se utilizaron en este proyecto:

ofxNetwork: Forma parte de la distribución oficial de openFrameworks y provee tres clases para trabajos relacionados con redes, estas son ofxTCPClient, ofxTCPServer y ofxUDPManager. Las primeras dos clases forman la base de comunicación en las aplicaciones desarrolladas para este trabajo y fueron elegidas porque se ajustan al modelo cliente-servidor buscado y determinan el tipo de paquete con el que se van a comunicar las aplicaciones: TCP. Al tratarse de una clase pensada para funcionar como servidor ofxTCPServer provee métodos para el monitoreo y control de clientes lo que permite darle una identidad a cada cliente y dirigir paquetes TCP a cada uno de ellos individualmente, grupalmente, o simplemente a todos los clientes conectados. Esto otorga gran flexibilidad y control a la hora de diseñar la obra ya que se pueden evitar problemas o efectos no deseados comunes cuando se trabaja con un número de usuarios estimado pero no predeterminado.

---

[94] "FLAC - Free Lossless Audio Codec." *FLAC - Free Lossless Audio Codec*. N.p., n.d. Web. 18 Nov. 2014. <https://xiph.org/flac/>.

ofxPd[95]: Está basado en libPd[96] y permite integrar un patch de Pd con la aplicación que se está programando, se puede enviar y recibir audio, mensajes y eventos MIDI desde el código en C++. La instancia de Pure Data que se ejecuta mantiene las mismas características que la versión standalone, puede procesar audio, realizar cálculos, etc. En la interfaz de audio a utilizar, la frecuencia de muestreo, profundidad en bits y tamaño de buffer son configurables como en su versión original. La programación del patch se realiza externamente mediante el editor de Pure Data y la aplicación servidor se encarga de abrir e integrarlo al resto del código. De esta manera se pueden modificar los patches de Pd sin necesidad de compilar nuevamente el código en C++, además de facilitar la adaptación de patches anteriores que significa un ahorro de tiempo al poder reutilizar código ya existente y probado. En primeras versiones de la aplicación se utilizó esta extensión, pero finalmente quedó descartada y se decidió utilizar mensajes OSC para la comunicación entre aplicaciones ya que permite mayor flexibilidad para el envío de mensajes a otras aplicaciones que trabajan con OSC. Por otro lado también resulta de utilidad poder ver el patch de Pure Data en funcionamiento, tanto en etapas de desarrollo como durante un evento ya que permiten solucionar problemas, acomodar, programar y experimentar en tiempo real con los objetos que conforman el núcleo sonoro de una obra.

ofxOsc: Forma parte de la distribución oficial de openFrameworks y permite el procesamiento de paquetes y mensajes para que cumplan con la norma OSC, y el envío de estos paquetes a través de UDP que es parte también del protocolo. Dada esta última condición este addon no pudo utilizarse, pero se adaptó parte del código, aquella relacionada con el formateo de paquetes, para poder enviar paquetes Osc a través de TCP.

ofxTimeline[97]: Se trata de una extensión elaborada para implementar una línea de tiempo similar a la que se encuentran en los editores de audio y video, con pistas, código de tiempo, funciones de transporte, opciones para guardar y cargar datos a través de ficheros XML. Incluye herramientas para crear una o más líneas de tiempo con distintos tipos de pistas (switches, flags, video, audio, LFO) e items. Se adaptaron las pistas de tipo switch para ser utilizadas en la aplicación servidor, estas indican a los clientes cuando pueden comenzar a enviar datos de alguno de sus sensores.

---

95 "Danomatika/ofxPd." *GitHub*. N.p., n.d. Web. 17 Nov. 2014. <https://github.com/danomatika/ofxPd>.

96 "Libpd." *Libpd RSS*. N.p., n.d. Web. 19 Nov. 2014. <http://libpd.cc/>.

97 "YCAMInterlab/ofxTimeline." *GitHub*. N.p., n.d. Web. 16 Nov. 2014. <https://github.com/YCAMInterlab/ofxTimeline>.

ofxAndroid[98]: Es necesario para la programación de aplicaciones para Android, viene incluido con la distribución de OF destinada a Android. Requiere del Android SDK[99] (que utiliza como IDE una versión modificada de Eclipse), del NDK[100] (Native Development Kit - Kit de desarrollo nativo) que es una implementación de la JNI[101] [102](Java Native Interface - Interfaz nativa de Java) necesaria para compilar código nativo (como C, C++) compatible con Java (así lo requiere la plataforma Android) y permite hacer llamadas JNI que dan la posibilidad de hacer llamadas desde Java a código en C++ y viceversa. Java es utilizado para la programación de la interfaz de la aplicación incluyendo el menú de opciones y menús contextuales. Algunas funciones de Android son llamadas directamente desde el código nativo mediante métodos como ofxAndroidToast() que dibuja en pantalla un toast (mensajes popup incluidos en Android de duración predeterminada que sirven para informar al usuario de eventos) y así generar un feedback con el usuario.

ofxAccelerometer: Se utiliza en combinación con ofxAndroid y es un add-on exclusivo para dispositivos móviles que tengan acelerómetro, la gran mayoría posee ya que forma parte de los requerimientos de Android. Permite acceder a los datos que proporciona este sensor: la rotación del dispositivo en ejes X, Y, Z (esta última se obtiene teniendo en cuenta los datos de X e Y).

ofxMultitouch: Si la pantalla del dispositivo es multitouch este addon puede acceder a las coordenadas en pantalla de cada uno de los dedos. Útil para implementar gestos. Para este proyecto no fue implementado ya que no todos los dispositivos cuentan con soporte multitouch, y en algunos casos el tamaño de la pantalla resulta poco práctico para el uso de más de un dedo.

---

98 "Android ADT." *OpenFrameworks*. N.p., n.d. Web. 17 Nov. 2014. <http://openframeworks.cc/setup/android-eclipse>.

99 "Get the Android SDK." *Android SDK*. N.p., n.d. Web. 17 Nov. 2014. <https://developer.android.com/sdk/index.html>.

100 "Android NDK." *Android Developers*. N.p., n.d. Web. 17 Nov. 2014. <https://developer.android.com/tools/sdk/ndk/index.html>.

101 Gordon, Rob. *Essential JNI: Java Native Interface*. Upper Saddle River, NJ: Prentice Hall, 1998. Impreso.

102 "Java™ Native Interface." *Java SE 7 Java Native Interface-related APIs and Developer Guides*. N.p., n.d. Web. 17 Nov. 2014. <https://docs.oracle.com/javase/7/docs/technotes/guides/jni/index.html>.

Interfaz Gráfica de Usuario (GUI – Graphic User Interface) de la aplicación servidor

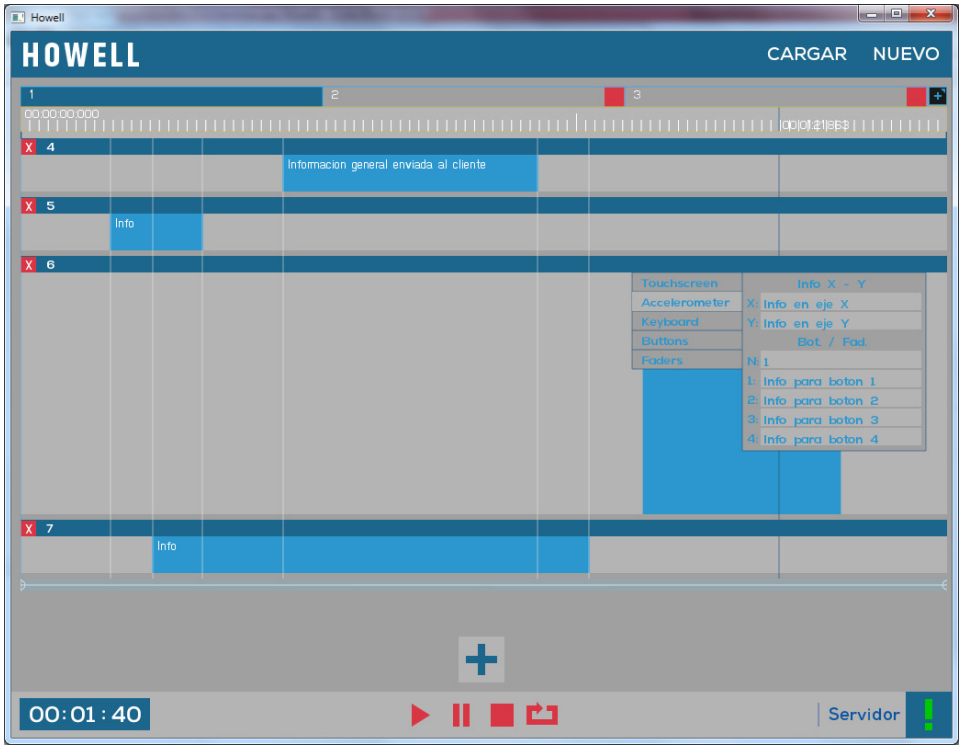


Figura 6. Interfaz gráfica de usuario.

Se tomaron como referentes a distintos programas para edición de video y audio no-lineales (tales como Nuendo[103], Reaper[104], Audition[105], Premiere[106]) por poseer algunos elementos en común: línea de tiempo multipista con ítems, opciones para guardar y cargar las sesiones de trabajo y poder recuperar el trabajo en el futuro, algunos de ellos están diseñados para facilitar la edición y producción en vivo y en tiempo real de audio y video como son los casos de Ableton Live[107] y Resolume Avenue[108]. Para esta aplicación prototipo se buscó una

---

103 "Steinberg Nuendo - Creativity First." *Http://www.steinberg.net/*. N.p., n.d. Web. 17 Nov. 2014. <http://www.steinberg.net/en/products/nuendo_range.html>.

104 "REAPER | Audio Production Without Limits." *REAPER | Audio Production Without Limits*. N.p., n.d. Web. 17 Nov. 2014. <http://www.reaper.fm/>.

105 "Adobe Audition Creative Cloud." *Adobe Creative Cloud*. N.p., n.d. Web. 17 Nov. 2014. <https://creative.adobe.com/products/audition>.

106 "Adobe Premiere Pro CC." *Video Editing Software*. N.p., n.d. Web. 17 Nov. 2014. <https://www.adobe.com/products/premiere.html>.

107 "What Is Live?" *Learn More about Ableton's Music Making Software*. N.p., n.d. Web. 17 Nov. 2014. <https://www.ableton.com/en/live/>.

108 "Resolume Avenue VJ Software." *Resolume VJ Software & Media Server*. N.p., n.d. Web. 17 Nov. 2014. <http://resolume.com/>.

interfaz de usuario sencilla, minimalista, con la menor cantidad de objetos visuales posibles para no perturbar la visión, evitar distracciones durante un evento en vivo y simplificar la programación. La aplicación está pensada para diseñadores que ya conocen el funcionamiento de un editor multipista y para ellos la curva de aprendizaje va a ser menor, pero también es fácilmente accesible por un usuario que no esté acostumbrado a trabajar en este tipo de programas.

La interfaz (Fig. 6) consta de una barra de herramientas que permite el guardado y la recuperación de proyectos, una barra de transporte (con botones que permiten detener la obra y volver al comienzo, pausarla y reanudar el curso de la obra, activar la opción de modo bucle para que el total de la obra comience nuevamente desde el principio una vez finalizada). La línea de tiempo está provista por ofxTimeline que contiene un indicador de la duración total de la obra y distintas subdivisiones temporales, una serie de pistas cuyo número varía según lo que necesite el diseñador, en las pistas se colocan ítems que funcionan como interruptores de duración determinada para el inicio y detención de envíos de datos desde los dispositivos al servidor (Bloques de interacción). Estas permiten indicar la cantidad de usuarios que actúan sobre una pista, si es más de uno los valores son promediados antes de ser enviados al patch de Pure Data.

Funcionamiento y explicación técnica

La comunicación de todo el sistema se da mediante paquetes TCP (Transmission Control Protocol) para asegurar la correcta entrega de paquetes, estabilidad y por funcionar con el paradigma cliente-servidor que necesita este proyecto. TCP a diferencia de UDP (User Datagram Protocol) tiende a agregar una mínima latencia en el caso de estar enviando o recibiendo una gran cantidad de paquetes, en la práctica esta latencia no es perceptible, y las ventajas de TCP resultan muy convenientes.

Como se mencionó anteriormente la comunicación es a través de paquetes TCP enviados a través del método sendRawBytes() que proveen las clases ofxTCPServer y ofxTCPClient, como su nombre lo indica este método envía bytes crudos a diferencia de otros métodos, como

send(string) que formatea y envía paquetes a partir de un tipo de dato string de C++. Antes de enviar a través de este método es necesario darle un formato para que pueda ser decodificado en el otro extremo.

El otro método consiste en el uso de la línea de tiempo, de las pistas y de los ítems, utilizando este método el envío de mensajes a los dispositivos queda automatizado y perfectamente sincronizado, cada pista envía un mensaje a un determinado número de clientes para que envíen datos de uno de sus sensores, están basadas en el tipo de pistas switches que vienen incluidas en el addon ofxTimeline, estos switches o interruptores se activan cuando el indicador de tiempo coincide con la entrada o salida de los ítems, a la entrada del ítem se envía un mensaje para que comience en el envío de datos del dispositivo, y a la salida para que cese ese envío.

Así la estructura de los patches de Pure Data queda conformada del siguiente modo: existe un patch obligatorio de nombre main.pd que es el que va a ser editado directamente por la aplicación servidor para agregar automáticamente los objetos necesarios, el usuario debe programar su patch como un subpatch dentro de este principal para evitar problemas con la edición automática del archivo.

HOWELL

Nombre

El nombre elegido para la prueba de concepto proviene de un cuento breve escrito por Julio Cortázar de título “Instrucciones para John Howell” perteneciente al libro “Todos los fuegos el fuego”.

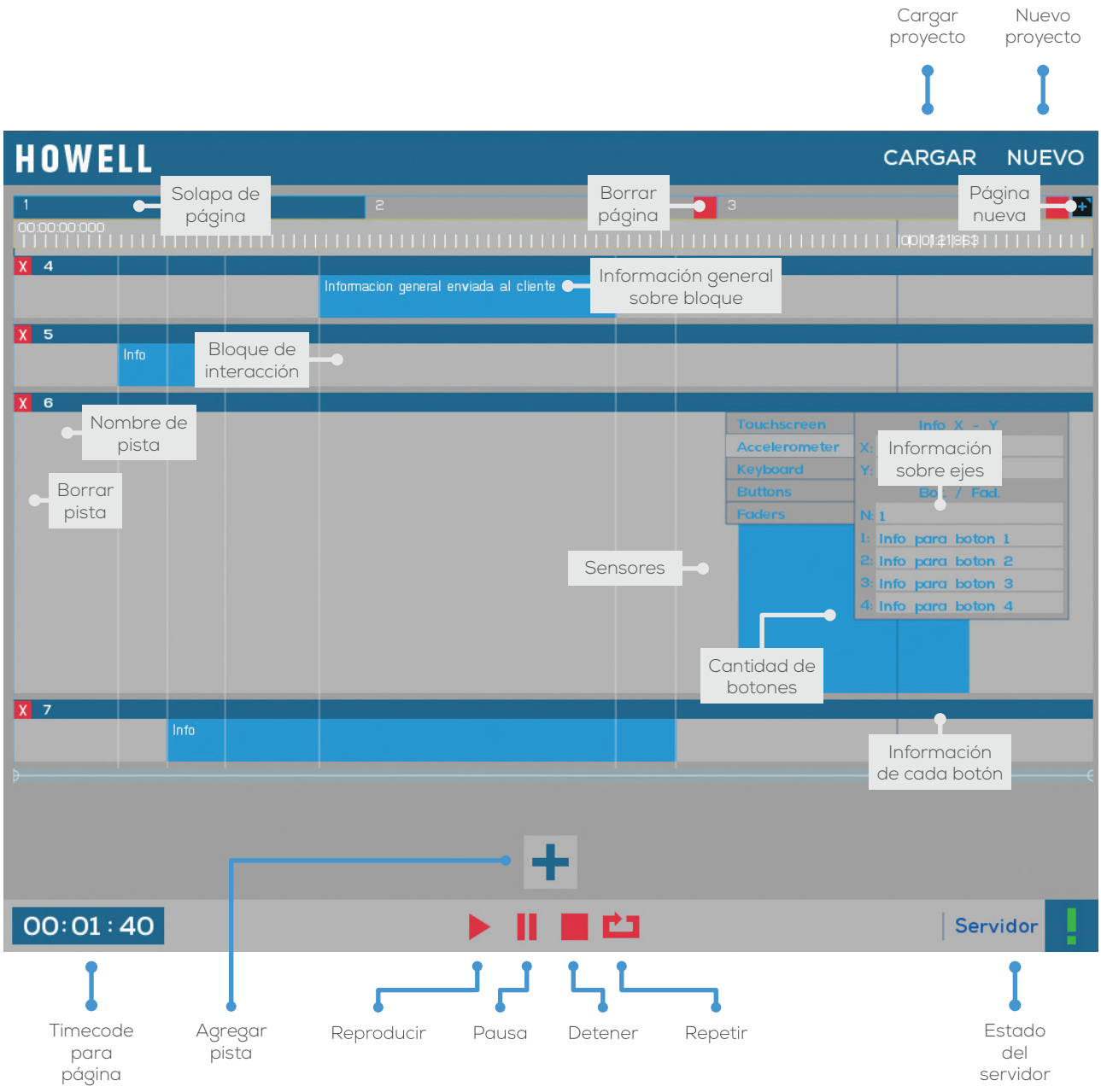
Dice Julio Cortázar en El sentimiento de lo Fantástico (Conferencia en la U.C.A.B)[109]:

“Escribí un cuento fantástico que se llama “Instrucciones para John Howell”, no les voy a contar el cuento; la situación central es la de un hombre que va al teatro y asiste al primer

[109] “Sobre “Instrucciones Para John Howell”” *La Máquina Del Tiempo*. N.p., n.d. Web. 18 Nov. 2014. <http://www.lamaquinadeltiempo.com/cortazar/howell1.htm>.

acto de una comedia, más o menos banal, que no le interesa demasiado; en el intervalo entre el primero y el segundo acto dos personas lo invitan a seguirlos y lo llevan a los camerinos, y antes de que él pueda darse cuenta de lo que está sucediendo, le ponen una peluca, le ponen unos anteojos y le dicen que en el segundo acto él va a representar el papel del actor que había visto antes y que se llama John Howell en la pieza. “Usted será John Howell”. “

John Howell pueden ser todos los espectadores que quieran participar, y por eso la aplicación que llevarían en sus teléfonos y tablets lleva este nombre.



Para la interfaz de usuario se pensó en un diseño sencillo y práctico, con la menor cantidad de elementos en pantalla posible. La pantalla que se presenta ni bien se ejecuta la aplicación cuenta con los siguientes ítems:

- Barra de proyecto: en ella se encuentran dos botones Cargar y Nuevo. Haciendo click sobre el primero se abre una ventana de selección de carpeta en la que debe ubicarse la carpeta del proyecto que se desee cargar (teniendo en cuenta que un proyecto consta de varios archivos individuales y que estos deben estar en la misma carpeta), una vez seleccionada se acepta y el proyecto ahora está cargado en la aplicación listo para ser utilizado. Haciendo click sobre el botón Nuevo se despliega la misma ventana de selección de carpeta que en el caso de Cargar, pero en esta selección se debe tener en cuenta que la carpeta debe estar vacía porque en ella se generará un proyecto, y se borrará cualquier proyecto anterior que hubiera.

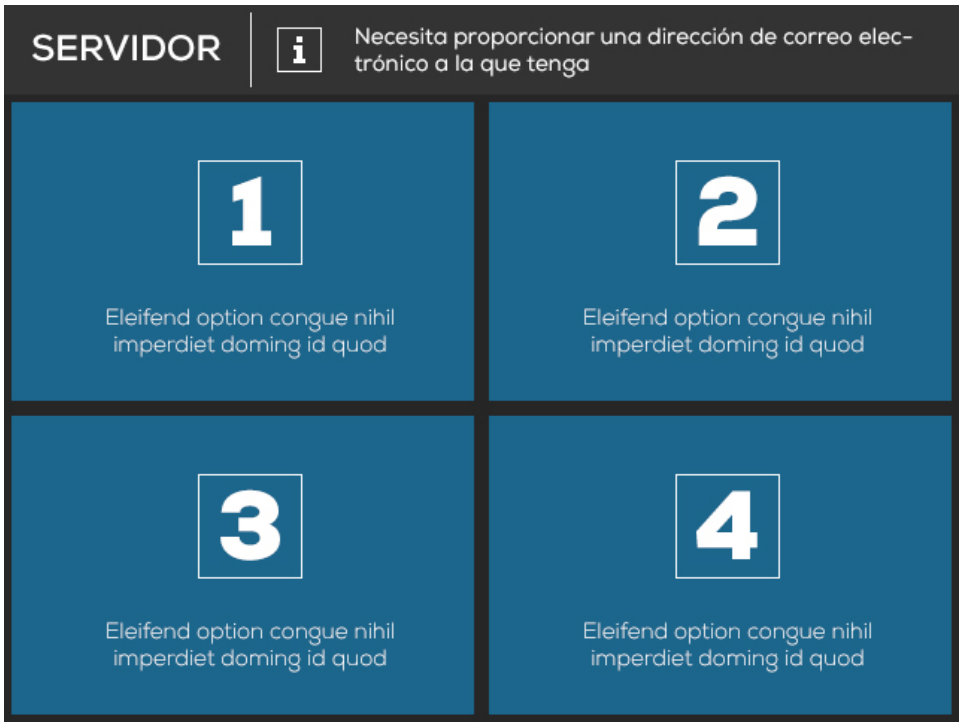
- Línea de tiempo, pistas y bloques de interacción: En el centro de la ventana, una vez cargado un proyecto anterior o creado un proyecto nuevo, puede verse la implementación de ofxTimeline que se utilizó para este proyecto. En la parte superior de esta implementación pueden verse las solapas numeradas de páginas o secciones musicales, a la misma altura se encuentra un botón con un símbolo de más (+) que agrega nuevas solapas de sección al proyecto si el servidor se encuentra offline. Bajo esta solapa se encuentra el ticker con líneas de división de tiempo, en esta barra se muestra la posición actual del contador de tiempo representado en pantalla como una línea que atraviesa todas las pistas. Bajo el ticker comienzan las pistas, cuya cantidad depende del proyecto. Cada pista en su parte superior izquierda cuenta con un botón rojizo con una “X” que funciona como botón para borrar esa pista si el servidor está offline. A la misma altura se encuentra el identificador de pista mostrado por un valor numérico. Seguido de este identificador, si hay clientes conectados al servidor y a esa pista en particular, se muestra/n el/los nombre/s de cliente/s conectado/s. Bajo este header o cabecera de pista, se encuentra el espacio en el que se colocan los bloques de interacción.

- Bloques de interacción: Sobre las pistas se pueden colocar bloques de interacción que tienen la función de indicarle a los clientes conectados cuándo deben participar,

con qué sensor lo van a hacer, y si es necesario, se puede enviar información sobre qué deben hacer. Sobre estos bloques puede escribirse texto que luego aparecerá en las pantallas de todos los clientes asociados a modo de información general. Estos bloques pueden estirarse, acortarse, agregarse, borrarse y moverse según sea necesario. Si se hace click derecho sobre uno de estos bloques se despliega un menú contextual correspondiente al bloque seleccionado.

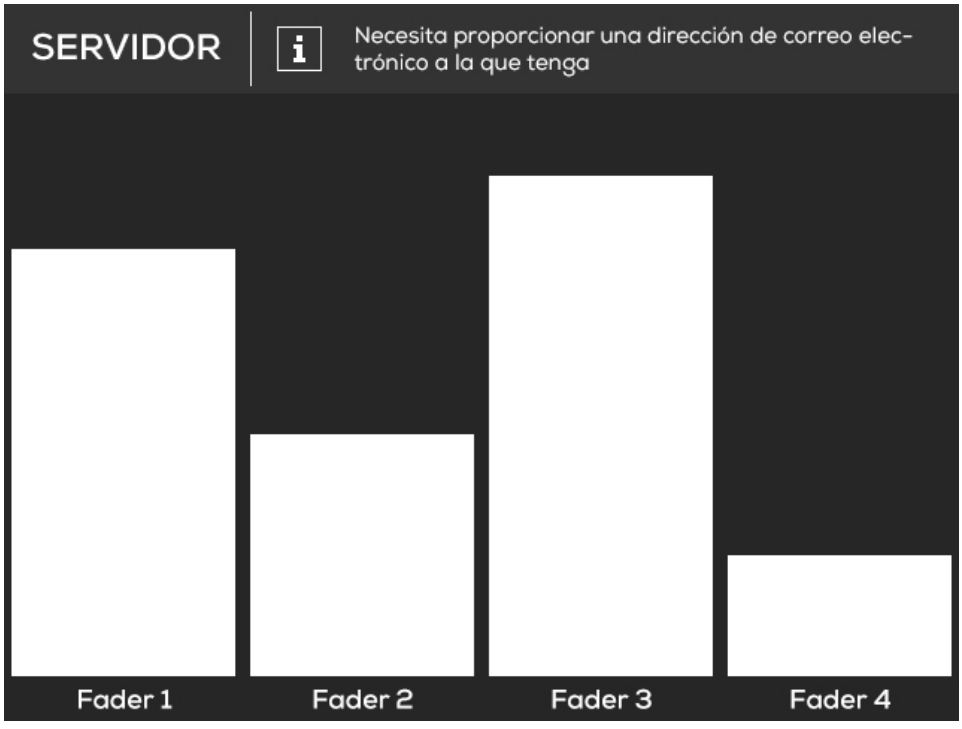
- Menú contextual de los bloques de interacción: Este menú se muestra en pantalla al hacer click derecho sobre alguno de los bloques de interacción. En el menú se encuentran las opciones para elegir el sensor del que se quieren recibir datos (Pantalla táctil, acelerómetro, botones, etc). También se encuentra una sección en la que se puede ingresar información que luego será enviada a cada dispositivo para que los espectadores sepan que parámetros están controlando en ese momento.

- Barra de transporte: En la parte inferior de la ventana se encuentra la barra de transporte con las opciones típicas: botones para reproducir, pausar, detener y volver al inicio o entrar en el modo bucle. A la izquierda de esta barra se encuentra el timecode de la sección actual, es decir, la duración de la sección. Este timecode es modificable para permitir que cada sección pueda tener distintas duraciones. Sobre la barra de transporte con un signo de más "+" se ubica el botón de agregar pistas, cuyo funcionamiento se habilita cuando el servidor está offline. En la parte inferior derecha de la ventana se encuentra el estado del servidor (online / offline) que a su vez funciona como interruptor para cambiar este estado.



Captura de pantalla

Captura de pantalla



Captura de pantalla

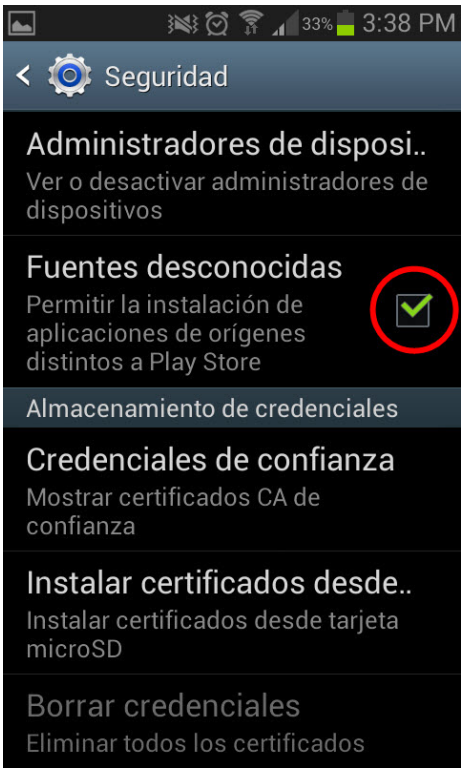
La aplicación para Android: el cliente

En la parte superior de la aplicación cliente puede verse una barra horizontal con que contiene la siguiente información: nombre del servidor o de la obra, que aún no está

implementado pero que permitiría mostrar un nombre personalizado para la obra en curso. Separado por una línea divisoria se encuentra la información que se recibe de los bloques de interacción, cuya función es indicar al espectador de que se trata la sección o que debería estar haciendo.

Bajo esta barra horizontal se encuentra un área cuyo contenido gráfico varía según el sensor que esté activado. Si se encuentran activados los sensores de pantalla o acelerómetros se muestra la información que recibe del bloque de interacción para los ejes X e Y. En el caso de que estén activadas las opciones de faders o botones se dibujan en pantalla la cantidad de botones o faders que indique el bloque de interacción sumado a la información que se recibe para cada uno de estos botones o faders.

Setup e instalación



El setup consta de una computadora corriendo la aplicación servidor, que fue desarrollada y probada en Windows 7 pero puede ser compilada para Linux y Mac OS aunque su correcto funcionamiento no está asegurado. Es necesario un router inalámbrico para proveer una red de área local inalámbrica (WLAN) a la que puedan acceder los dispositivos móviles, el router debe estar configurado para asignarle la IP 192.168.0.10 a la computadora servidor, el cliente está configurado para conectarse automáticamente a un servidor presente en esta IP (también tiene la opción de ingresar manualmente la IP, pero para simplificarle el uso al público se decidió utilizar una IP fija), la conexión entre computadora y router puede ser cableada o inalámbrica, esto no presenta diferencias. La computadora servidor se conecta directamente al sistema de sonido a través de su interfaz de audio (on-board o externa).

La aplicación cliente se instala como cualquier aplicación externa para Android, está

contenida en un paquete .apk (APK - Android Package)[110], idealmente debería poder descargarse de la Play Store (Tienda virtual de Google donde se nuclean una gran cantidad de aplicaciones, tanto gratuitas como pagas) pero se postergó por el costo de U$S25 para registrarse como desarrollador de Android. Funciona en versiones de Android 2.2 (Froyo) en

[110] Son los paquetes estándar de Android que contienen a la aplicación a instalar.

adelante, el soporte de versiones más nuevas depende principalmente de las actualizaciones de openFrameworks al respecto. Al no estar en tienda Play Store es necesario activar la opción para instalar aplicaciones de orígenes desconocidos.

Una vez ejecutada la aplicación servidor el artista ya puede comenzar a diagramar la interacción de su obra, para ello es necesario en primera instancia asegurar que el módulo servidor esté en estado offline, para asegurarse de que ningún cliente pueda conectarse durante este proceso y de qué una vez que el servidor esté online y los clientes comiencen a conectarse puedan distribuirse adecuadamente las pistas existentes. Este estado queda indicado por el ícono en la parte inferior derecha de la ventana, si está de color rojizo el servidor está offline y si está de color verde el servidor está online. Para cambiar el estado basta con hacer click sobre este ícono.

Con el servidor offline, el diseñador puede agregar o quitar secciones o páginas según lo requiera su obra y asignar la duración de cada una de estas secciones usando el timecode dispuesto en la parte inferior izquierda de la ventana. Una vez que se tienen la cantidad de páginas que se necesiten se procede a crear pistas para que los espectadores puedan conectarse mediante el botón que está inmediatamente encima de la barra de transporte.

Teniendo las páginas y pistas en la línea de tiempo se puede proceder a agregar bloques de interacción sobre las pistas. Para ello es necesario hacer click sobre algún espacio vacío dentro de una pista para comenzar a crear un bloque, mover el mouse para indicar la duración, y hacer click nuevamente para finalizar la creación. Estos bloques pueden acomodarse, moviéndose, alargando o acortando su duración.

Si hace click sobre alguno de estos bloques con el botón secundario del mouse (click derecho) se muestra sobre la línea de tiempo un menú contextual en el que se puede ingresar el sensor que se requiere para ese momento específico de la obra. En este menú puede ingresarse también la información que quiere enviarse a los espectadores para guiarlos en la obra.

Se tiene ahora sobre la línea de tiempo una serie de secciones o páginas, pistas y bloques que en conjunto conforman el nexo interactivo de una obra. Para comenzar a utilizar este nexo basta con cambiar el estado del servidor a online, esperar a qué se conecten los clientes, y apretar el botón de play, o la barra espaciadora que cumple la misma función.

El contador de tiempo empezará a correr, y cuando se pose sobre alguno de los bloques le enviará un mensaje OSC a todos los clientes conectados a la pista que contiene el bloque con la información ingresada en el menú contextual. El servidor comenzará a recibir entonces datos de todos los dispositivos conectados y los reenviará por mensajes OSC a través de UDP a la IP 127.0.0.1, puerto 12345. De esta manera, cualquier aplicación que se encuentre en la misma computadora, escuchando los mensajes que se envían a este puerto podrá recibir los mensajes OSC y decodificarlos. En este proyecto se utiliza Pure Data para poder generar audio interactivo.

Cliente

Una vez instalada en el dispositivo móvil y ejecutada, la aplicación busca servidores activos en la IP predeterminada 192.168.0.10 cada cierto intervalo de tiempo para no saturar de paquetes la conexión, y evitar cuelgues en la aplicación. Una vez que la aplicación logra conectarse a un servidor éste le asigna una identificación numérica dada por ofxTCPServer (clase de la extensión ofxNetwork) y en pantalla se muestra un *text field*[111] para ingresar texto. Antes de poder participar en la obra debe ingresar un nombre de usuario en este campo para facilitar la identificación futura del espectador. Una vez ingresado el nombre de usuario el cliente ya se encuentra conectado al servidor quedando a la espera de recibir mensajes que activen algunos de sus sensores o instrucciones sobre su rol en la obra.

El usuario cliente recibe feedback y mensajes de parte del servidor mediante los *toast*[112] de Android, que son mensajes de corta duración que aparecen como pop-ups sobre la aplicación y son propios del sistema operativo, y mediante cadenas de texto dibujadas en pantalla.

---

111 "Text Fields." *Android Developers.* N.p., n.d. Web. 18 Nov. 2014. <http://developer.android.com/guide/topics/ui/controls/text.html>.

112 "Toasts." *Android Developers.* N.p., n.d. Web. 18 Nov. 2014. <http://developer.android.com/guide/topics/ui/notifiers/toasts.html>.

ofxAndroid provee el método ofxAndroidToast(string) para generar toasts de forma sencilla directamente desde C++ sin necesidad de hacer llamadas JNI a código Java manualmente, teniendo en cuenta que este tipo de funciones, propias del sistema operativo Android, se llaman desde el código Java. Por otro lado, cabe mencionar que las llamadas JNI desde código Java a funciones en C++ son más directas y sencillas de implementar.

En cualquiera de los dos casos los mensajes OSC enviados son los mismos, y los valores están normalizados entre 0 y 1 para facilitar el envío a PureData: /touchscreen 1 para que el dispositivo comience a enviar datos del touchscreen, mediante este mensaje lo que se recibe son mensajes de una sola posición del touch, el multitouch no envía datos en este modo. / touchscreen 0 indica al dispositivo que deje de enviar datos de este sensor. /keyboard 1 activa el teclado en pantalla de Android y envía mensajes OSC al servidor con strings como argumento que son redirigidos a Pure Data como símbolos. /accelerometer 1 activa el acelerómetro en el cliente y envía al menos dos valores: x e y según el grado de inclinación del dispositivo, también puede enviar un tercer parámetro: z, que se obtiene a partir de los dos anteriores.

El cliente devuelve los datos a través de mensajes OSC con la misma dirección que recibe los switches. /accelerometer x(float) y(float) devuelve los valores del acelerómetro en floats normalizados. /keyboard string devuelve en formato de string el texto que haya ingresado y enviado el usuario, la aplicación servidor luego reenvía este string a la instancia de Pure Data como symbol. /touchscreen x(float) y(float) envía al servidor las coordenadas de un sólo dedo al tocar la pantalla táctil del dispositivo.

Una vez que el cliente se haya conectado e ingresado su nombre de usuario este último se envía en un mensaje OSC con el siguiente formato /name NombreDeUsuario. El servidor recibe este mensaje y agrega este nombre de usuario como string a un struct de nombre client que reúne variables para identificar a un cliente, además del nombre de usuario se almacena su dirección IP (permite prohibir el futuro acceso al servidor si el administrador lo cree necesario), su clientID, como se mencionó anteriormente es un valor numérico entero

dado por ofxTCPServer para identificar a los clientes por sí solo no sirve para identificar a un usuario en particular porque este número va rotando según las conexiones y desconexiones de los clientes. Este struct client está dentro de un contenedor vector de C++ (superficialmente se lo puede ver como un array dinámico) ya que la cantidad de clientes no es fija el uso de un contenedor dinámico es conveniente.

openFrameworks es una plataforma que provee muchas soluciones a desarrolladores produciendo obras multimedia, y ofxTimeline resultó ser una extensión de esta plataforma muy útil para este proyecto. Una expansión posible consistiría en mantener el uso de estas extensiones cuyo funcionamiento está probado y que se actualizan constantemente, pero mover al menos parte de la interfaz y el manejo de archivos a un gestor de GUI que utilice los recursos propios de cada sistema operativo como QT, o wxWidgets, para no saturar los recursos gráficos de un sistema.

Control de usuarios y límite por pista

Una lista de los clientes conectados que incluya el nombre de usuario con su dirección IP, su número de cliente (es otorgado por una de las clases de la extensión ofxNetwork) y botones para bannear (borrar un usuario del servidor y por ende terminar su participación en la obra inmediatamente) es importante para mantener cierto orden (si se lo necesitara) durante una obra. En el caso de que un espectador haya decidido dejar de participar, o por motivos técnicos no pueda seguir haciéndolo se puede echar del servidor para dejar lugar a nuevos espectadores interesados en participar.

En el caso de que haya pistas en que se requiera que haya un único cliente conectado, o en el caso de querer dar cierta prioridad a algunas pistas por su función en la obra resultaría

conveniente poder limitar la cantidad de usuarios que puedan conectarse a cada pista.

Acceso a más funciones

Los dispositivos móviles cuentan con más sensores de los que fueron implementados en Howell, entre ellos cabe destacar el micrófono como medio de input sonoro, las cámaras que poseen pueden ser útiles en obras multimedia, y el GPS para obras en las que la posición física de los espectadores sea una necesidad.

Tanto para el caso del micrófono como para el caso de las cámaras sería necesario implementar en Howell un sistema de audio y video inalámbrico, tipo de implementación que ya existe en otras aplicaciones por lo que es una opción viable, sería necesario también poder reenviar este audio a otras aplicaciones como Pure Data para su posterior análisis y procesamiento, o realizar internamente este análisis mediante la librería kissfft que viene con el paquete de openFrameworks u ofxPd para integrar un patch de Pure Data al sistema.

El GPS puede ser utilizado para espacialización sonora basada en ubicación y es particularmente eficiente en exteriores.

Lista de servidores u obras en curso dentro de una misma red

En este momento la aplicación cliente busca conectarse a una IP fija por ser la implementación más fácil para facilitarle al usuario el uso del sistema, también puede ingresarse manualmente una IP, pero fue útil para la etapa de desarrollo y no resulta conveniente utilizar este método en una aplicación que se pretende simple. Una vez iniciada la aplicación un menú desplegable con todos los servidores u obras en curso que se encuentren en ese momento funcionando resultaría lo más práctico, bastaría con seleccionar la obra, ingresar su nombre

de usuario y esperar a que le toque su turno para interactuar.

Distribución por Play Store

En este momento la única forma de instalar la aplicación de Android es descargándola e instalándola con un gestor de paquetes, siendo la mejor forma posible la distribución a través de la tienda virtual oficial de Android, Play Store, en la que se nuclean un gran número de aplicaciones tanto gratuitas como pagas. No fue posible utilizar este método de distribución porque cuestiones económicas.

OSC sobre TCP

La implementación de OSC por TCP que se utilizó en este proyecto no cumple con la especificación 1.1 de OSC para el envío de mensajes por este protocolo, para ello habría que encapsular los mensajes con SLIP antes de enviarse por TCP, como lo hace netPD[113].

Howell como control

ofxTimeline provee otros tipos de pistas, para este proyecto se utilizó una adaptación de un tipo de pista denominado Switches, pero incluye también pistas como LFO (Low Frequency Oscillator), control de video, pistas de audio (en las que se pueden cargar archivos de audio y reproducirlos como si fuera un DAW) y pistas para dibujar envolventes que podrían ser enviadas por OSC a cualquier aplicación externa. Duration[114], un software basado en ofxTimeline, provee estas opciones y podría ser utilizado a futuro.

---

113 "Protocol." *Netpd*. N.p., n.d. Web. 16 Nov. 2014. <http://www.netpd.org/Protocol>.

114 "Duration Timeline." *Duration Timeline*. N.p., n.d. Web. 17 Nov. 2014. <http://www.duration.cc/>.

Pure Data integrado

Gracias a la librería libPD adaptada como addon de openFrameworks con el nombre ofxPd es posible compilar aplicaciones con Pure Data integrado, es decir que no es necesario ejecutar una instancia nueva del programa si no que ya se encontraría corriendo como un proceso background del sistema operativo. Esto evitaría el uso de mensajes OSC ya que se pueden usar los sends y receives propios de Pd. Una posible implementación de este addon sería que al crear un proyecto nuevo se genere un archivo de nombre fijo, por ejemplo main.pd. Al crear pistas la aplicación servidor agrega receives en el patch de Pure Data main.pd. Para facilitar la tarea de programación del patch, estos receives ya tienen el nombre correcto de la pista por lo que funcionarían sin problemas. Esto se lograría editando el archivo main.pd como si tratara de un archivo de texto (.txt) y agregando el código correspondiente para crear un objeto receive: #X obj x y r nombredepista, los valores x e y indican las coordenadas de la ubicación del objeto dentro del canvas de Pure Data, luego continúa el nombre del objeto, en este caso r de receive, y el nombre de la pista que le asigna Howell.

Control manual

La aplicación servidor podría tener dos formas de indicar al cliente que comience a enviar datos de sus sensores que no sean mutuamente excluyentes. Una de ellos consiste en botones ubicados en la lista de clientes que permiten seleccionar manualmente el sensor y cliente del que se quieren recibir mensajes, al hacer click en algunos de estos botones de acceso directo se envía un mensaje OSC distinto dependiendo del sensor seleccionado, los mensajes que recibe el servidor con los datos generados por el dispositivo son enviados como sends a Pure Data y se relacionan con el clientID que le fue otorgado.

Conclusiones

Como cierre a esta investigación y al desarrollo de software que acompañó a ésta se puede concluir que un sistema universal para el control de interacción en performances e instalaciones sonoras es posible y viable. Estéticamente este sistema abre las puertas a nuevos métodos de composición interactivos tanto de obras de carácter temporal lineal como no-lineal.

Aquellos artistas que desean integrar interacción personal o colectiva en obras sonoras pueden hacerlo sin necesidad de aprender programación o dedicarse a resolver problemas técnicos ajenos a una producción artística: pueden concentrarse en la composición musical, en el diseño sonoro y en el vínculo interactivo que se genera entre los espectadores y el sonido.

Si bien el software que se presenta adjunto a este proyecto todavía no aprovecha todas las posibilidades que estas tecnologías permiten es lo suficientemente versátil como para poder ser utilizado en diversos contextos y de diversas maneras, y alcanza para comprender el potencial que una herramienta de estas características puede tener.

La aplicación compilada para esta tesis representa un comienzo, una prueba de concepto para iniciar el desarrollo de una aplicación con mayor cantidad de funciones, mayor alcance y estabilidad. La programación para dispositivos móviles basados en Android presenta sus dificultades en un comienzo sobre todo cuando se combina código nativo (C++) y Java, algunas de ellas fueron descritas en esta presentación junto con el método más efectivo que se encontró para solucionarlas, la gran cantidad de información disponible en bibliografía e internet hace accesible estas tecnologías a los artistas multimediales que pretendan utilizarlas sin intermediarios.

Para mayor facilidad en la conexión a un servidor y en la instalación de una obra debería implementarse un sistema de auto-discover para que la aplicación clientes pueda encontrar el servidor automáticamente sin tener que conectarse a una IP fija, y en el caso de que haya más de un servidor por estar utilizándose el sistema en más de una obra dentro de un espacio físico sería conveniente tener una lista de servidores u obras a la que el usuario pueda conectarse

sin tener que ingresar manualmente la IP del servidor, de esta manera conectándose a una red móvil el usuario puede tener una lista inteligible de las obras a las que puede conectarse y participar en un espacio físico delimitado.

La aplicación requerida por los espectadores no es de fácil acceso actualmente, para poder utilizarla se debe descargar e instalar manualmente, esto puede resultar complicado para el usuario ya que debe navegar por la web, descargarlo, habilitar en el sistema operativo la instalación de paquetes de terceros (pasos que varían según la versión de Android) y luego instalarlo utilizando un gestor de paquetes siendo que Android (como la mayoría de sistemas operativos para dispositivos móviles) tiene su centro de aplicaciones llamado Play Store dónde se nuclean la mayor cantidad de aplicaciones.

openFrameworks, sus extensiones y la plataforma Android proveen varias herramientas que no fueron implementadas pero que resultarían convenientes tener presentes tales como diferentes tipos de pista, lista de usuarios conectados y control manual.

Como última reflexión, cabe destacar los conocimientos incorporados por parte del tesista en cuanto al desarrollo de aplicaciones informáticas relacionadas al arte, un campo aún con muchos interrogantes, pero con posibilidades ilimitadas gracias a la información disponible de forma libre y gratuita y los movimientos cada vez más grandes de hardware y software libre.

Sin libertad no hay educación.

Bibliografía y webs relevantes

Perevalov, Denis, and Sodazot. *Mastering OpenFrameworks Creative Coding Demystified: A Practical Guide to Creating Audiovisual Interactive Projects with Low-level Data Processing Using OpenFrameworks*. Birmingham: Packt Publ., 2013. Impreso.

Cancellaro, Joseph. *Exploring Sound Design for Interactive Media*. Clifton Park, NY: Delmar Learning, 2006. Impreso.

Collins, Nick, and Rincón Julio D' Escrivan. *The Cambridge Companion to Electronic Music*. Cambridge: Cambridge UP, 2007. Impreso.

Cook, Perry R. *Real Sound Synthesis for Interactive Applications*. Natick, MA: K Peters, 2002. Impreso.

Holmes, Thom. *Electronic and Experimental Music: Technology, Music, and Culture*. New York: Routledge, 2008. Impreso.

Huber, David Miles. *The MIDI Manual: A Practical Guide to MIDI in the Project Studio*. Boston: Focal, 2007. Impreso.

Kreidler, Johannes. *Loadbang. Programming Electronic Music in Pd. 1ª ed*. Hofheim: Wolke Verlag, 2009. eBook.

Manning, Peter. *Electronic and Computer Music*. Oxford: Oxford UP, 2004. Impreso.

Miranda, Eduardo Reck, and Marcelo M. Wanderley. *New Digital Musical Instruments: Control and Interaction beyond the Keyboard*. Middleton, WI: A-R Editions, 2006. Impreso.

Joshua, Noble. *Programming Interactivity A Designer's Guide to Processing, Arduino, and openFrameworks*. Sebastopol, California: O'Reilly Media, 2009. Impreso.

Puckette, Miller. *Pure data*. Floss Manuals, 2012. eBook.

Sommerer, Christa, Lakhmi Jain, et al. *The Art and Science of Interface and Interaction Design*. Berlin Heidelberg: Springer-Verlag, 2008. Impreso.

Hallberg, Bruce A. *Networking: A Beginner's Guide*. N.p.: n.p., n.d. Impreso.

Stroustrup, Bjarne. *The C++ Programming Language, Fourth Edition*. Boston: Addison-Wesley, 2013. Impreso.

Friesen, Jeff. *Learn Java for Android Development*. Berkley: APress, 2014. Impreso.

Phillips, Bill, and Brian Hardy. *Android Programming: The Big Nerd Ranch Guide*. Atlanta, GA: Big Nerd Ranch, 2013. Impreso.

Artículos

Bongers, Bert. *Physical Interaction in the Electronic Arts*. France: IRCAM, 2000. CD-ROM.

Chabot, Xavier. *"To Listen and to See: Making and Using Electronic Instruments."* Leonardo Music Journal 3 (1993): 11-16. Impreso.

Gluck, Robert J. *"Live Electronic Music Performance: Innovations and Opportunities."* 2007. TS. University at Albany, Haifa. Web.

Páginas Web:

"An Introduction to Sensors and Transducers." *An Introduction to Sensors and Transducers. N.p., n.d. Web. 09 Junio 2013.* <http://www.mfg.mtu.edu/cyberman/machtool/machtool/sensors/intro.html>.

"Android Developers." *Android Developers*. N.p., n.d. Web. 16 Nov. 2014. <http://developer.android.com/>.

"Android SDK." *Android SDK*. N.p., n.d. Web. 09 Junio 2013. <http://developer.android.com/sdk/>.

"Apple Developer." *IOS Dev Center*. N.p., n.d. Web. 09 Junio 2013. <https://developer.apple.com/devcenter/ios/index.action>.

"Cycling 74." *Cycling 74*. N.p., n.d. Web. 09 Junio 2013. <http://cycling74.com/>.

"OpenFrameworks." *OpenFrameworks*. N.p., n.d. Web. 09 Junio 2013. <http://www.openframeworks.cc/>.

"PD Community Site." *Pure Data*. N.p., n.d. Web. 09 Junio 2013. <http://puredata.info/>..

"How to Program in C++." *How to Program in C++*. N.p., n.d. Web. 16 Nov. 2014. <http://cs.fit.edu/~mmahoney/cse2050/how2cpp.html>.

"Protocol." *Netpd*. N.p., n.d. Web. 16 Nov. 2014. <http://www.netpd.org/Protocol>.

"Stack Overflow." *Stack Overflow*. N.p., n.d. Web. 16 Nov. 2014. <http://stackoverflow.com/>.

"CPlusPlus - Reference." - *C++*. N.p., n.d. Web. 14 Nov. 2014. <http://www.cplusplus.com/reference/>.

"C Programming and C++ Programming." *C Programming.com*. N.p., n.d. Web. 16 Nov. 2014. <http://www.cprogramming.com/>.

"Develop for Windows. It's Time." *Windows Dev Center*. N.p., n.d. Web. 16 Nov. 2014. <https://dev.windows.com/>.

"OpenFrameworks Developers Can Now Create Universal Apps for Windows - MS Open Tech." *MS Open Tech*. N.p., n.d. Web. 16 Nov. 2014. <http://msopentech.com/blog/2014/07/03/openframeworks-developers-can-now-create-universal-apps-for-windows/>.

“YCAMInterlab/ofxTimeline.” *GitHub*. N.p., n.d. Web. 16 Nov. 2014. <https://github.com/YCAMInterlab/ofxTimeline>.

Estas instrucciones siguen los pasos de la guía oficial con algunos pasos extras que resultaron necesarios para que los programas compilen correctamente. Esta guía fue probada en Xubuntu 12.04, en Septiembre del año 2013. La guía oficial puede encontrarse en :

http://www.openframeworks.cc/setup/android-eclipse/

1) Primero es necesario instalar Eclipse y JDK:

Para instalar el JDK desde el repositorio:

sudo apt-get install openjdk-6-jdk

Eclipse también está en los repositorios, la versión es algo vieja, pero funciona:

sudo apt-get install eclipse

También es necesario instalar el addon CDT para trabajar con C y C++

sudo apt-get install eclipse-cdt

2) Descargar el **SDK** y **NDK** de Android:

El **SDK** se consigue desde la página oficial :

http://developer.android.com/sdk/index.html

Es necesaria la versión **r8d** del **NDK** para que los programas compilen correctamente, se consigue desde los siguientes links según el sistema operativo:

- OS X: http://dl.google.com/android/ndk/android-ndk-r8d-darwin-x86.tar.bz2
- Linux: http://dl.google.com/android/ndk/android-ndk-r8d-linux-x86.tar.bz2
- Windows: http://dl.google.com/android/ndk/android-ndk-r8d-windows.zip

Descargar y descomprimir ambos paquetes, más adelante se le indican a los programas la ubicación de los mismos.

Entrar en android-ndk-r8d/toolchains/arm-linux-androideabi-4.7/prebuilt/ y renombrar la carpeta linux-x86 (la única carpeta que hay) a linux-i686 . Los makefiles están configurados para buscar esta carpeta y puede dar errores más adelante.

3) Descargar el paquete de **openFrameworks** para Android:

Se consigue desde la página de openFrameworks.cc, al igual que el SDK y el NDK es necesario descargarlo y descomprimirlo:

http://openframeworks.cc/download

4) Instalar **Ant**:

Se consigue desde los repositorios:

sudo apt-get install ant-1.8

5) Indicarle a la librería de **oF** las ubicaciones del **SDK**, **NDK** y **Ant**:

Dentro de la carpeta

openFrameworks/libs/openFrameworksCompiled/project/android/

Existe un archivo de nombre paths.make.default . Hay que renombrarlo a paths.make,

y abrirlo con un editor de texto, y ahi colocar la ubicación correspondiente de cada paquete descomprimido, excepto de **Ant** ya que si se instaló mediante los repositorios la ubicación que figura por defecto es la correcta. Utilizar barras no-invertidas ("/") para la ubicación.

6) Iniciar **Eclipse** e instalar **ADT**

Abrir **Eclipse** y cuando pida un directorio para el espacio de trabajo (workspace) elegir la ubicación de los ejemplos de oF para Android:

openFrameworks/examples/android/

El **ADT** se instala entrando a Help > Install New Software... . Hacer click en 'Add...' y poner la siguiente información:

Name: Android SDK

Location: https://dl-ssl.google.com/android/eclipse/

Hacer click en 'OK' . Cuando aparezcan los paquetes para instalar elegir en "Developer Tools" los paquetes "Android Development Tools" y "Android DDMS", y en la sección "NDK Plugins" elegir "Android Native Development Tools". Hacer click en "Next", aceptar la licencia para instalar los paquetes cuando lo pida y después aceptar reiniciar Eclipse para completar la instalación.

7) Comprobación y configuración de **Eclipse**

Asegurarse que el plugin CDT, y el JDK estén instalados correctamente entrando a Window > Preferences. Si se encuentra una solapa de Java el JDK está instalado, y si aparece una solapa para C/C++, el plugin CDT también fue correctamente instalado.

Dentro de Window > Preferences > Java > Compiler seleccionar **1.6** en 'Compiler compliance level' y hacer click en 'Apply'.

Dentro Window > Preferences > Android en SDK Location indicar la ubicación del SDK

descomprimido anteriormente.

Dependiendo de la versión de Eclipse puede ser necesario indicar la ubicación del NDK, para Eclipse 3.7 esto no es necesario. Dentro Window > Preferences > Android > NDK indicar la ubicación del NDK descomprimido anteriormente.

Dentro Window > Preferences > C/C++ > Code Analysis deseleccionar la casilla de 'Syntax and Semantic Errors' y hacer click en 'Apply'. Esto evita errores cuando intentemos escribir nuestro propio código.

Entrar en Window > Open Perspective > Other... y seleccionar C/C++ .

Entrar en Window > Customize Perspective... y seleccionar, dentro de la solapa Tool Bar Visibility, Android SDK and AVD Manager, quizás sea necesario desplegar las opciones haciendo click en la flecha de la izquierda, y seleccionando los paquetes Android SDK Manager y Android Virtual Device Manager. De esta manera podemos ver los botones para acceder al SDK.

Abrir el Android SDK Manager utilizando los botones generados en el paso anterior e instalar el SDK Platform para Android 4.2 (API 17). Los makefiles están configurados para trabajar con esta versión.

8) Importar **openFrameworks** a **Eclipse**

Antes de importar los proyectos asegurarse de que Project > Build Automatically esté desactivado.

Ir a File > Import y seleccionar General > Existing projects in the workspace...

Luego importar las siguientes carpetas en este orden asegurándose que la opción "Copy projects into workspace" esté destildada.

- openFrameworks/libs
- openFrameworks/libs/openFrameworks
- openFrameworks/addons/ofxAndroid/ofAndroidLib

- openFrameworks/examples/android

De esta última pueden importarse las carpetas de los ejemplos que queramos, no es necesario importar todos a la vez.

9) Compilar **openFrameworks**

En el "Project Explorer" seleccionar el proyecto openFrameworks que importamos anteriormente. Haciendo click en la flecha del icono de martillo (Build) seleccionar Android como objetivo, y luego hacer click en este icono. Esto va a compilar la librería y generar los archivos necesarios para compilar nuestros proyectos.

Si la compilación termina correctamente dentro openFrameworks > Archives deberían aparecer varios archivos con extensión .a , y dentro del proyecto ofAndroidLib > bin > res debería aparecer un archivo de nombre ofandroidlib.jar . Si este último no se genera ir a Window > Open Perspective > Other > Java . Hacer un Project > Clean para limpiar los proyectos y hacer Project > Build All. Esto compila todos los proyectos abiertos inclusive los ejemplos si es que importamos alguno, haciendo click derecho sobre cualquier proyecto podemos seleccionar 'Close Project' para asegurarnos de que estos no compilen y ahorrar tiempo.

10) Compilar proyectos

Seleccionar el proyecto que queremos compilar, y en Build (icono de martillo) seleccionar Release y compilamos. Si se realizaron todos los pasos anteriores el programa debería compilar correctamente.

11) Correr proyectos en dispositivos

Para correr los proyectos en dispositivos Android primero es necesario configurar el dispositivo y habilitar USB Debugging. Entrar a Settings > Applications > Development > USB Debug (En Ice Cream Sandwich, esto es Settings > Developer options > USB Debugging). El dispositivo tiene que estar desconectado de la computadora mientras se hace esto.

Abrir una terminal de Linux y navegar hasta :

carpeta-de-Android-sdk/platform-tools

y hacer:

sudo ./adb kill-server

sudo ./adb start-server

De esta manera evitamos problemas con dispositivos Android no reconocidos.

Lo último que queda por hacer es seleccionar el proyecto que queremos ejecutar y hacer Run > Run As > Android Application. Seleccionamos nuestro dispositivo y debería instalarse la aplicación y correr automáticamente.

Instalación de librerías necesarias para este proyecto y solución a problemas encontrados

Las dependencias necesarias para compilar el proyecto son:

ofxPd:
git clone https://github.com/danomatika/ofxPd.git

ofxTimeline
git clone https://github.com/YCAMInterlab/ofxTimeline.git

ofxTimecode:
git clone https://github.com/YCAMInterlab/ofxTimecode.git

ofxTween:
git clone https://github.com/obviousjim/ofxTween.git

ofxMSATimer:
git clone https://github.com/obviousjim/ofxMSATimer.git

ofxTextInputField:
git clone https://github.com/Flightphase/ofxTextInputField.git

ofxRange:
git clone https://github.com/Flightphase/ofxRange.git

Estas deben descargarse y descomprimirse en la carpeta addons de openFrameworks.

Instalar OpenAL:

Bajar OpenAL Soft y descomprimir. En la carpeta Win 32 renombrar el archivo soft_oal.dll a OpenAL32.dll, copiar este archivo a C:\Windows\System32\

Bajar libFlac-8.dll y copiar en C:\Windows\System32\

Windows CodeBlocks 12.11 pthread faltante

(Traducción de guía encontrada en el repositorio de ofxKinect[115])

En el caso de que un proyecto no se pueda compilar debido a que Code::Blocks no encuentra la librería "pthread" es necesario agregar ésta manualmente. Para ello hay que agregar la librería pthreads-win32 que se puede descargar de la siguiente dirección: http://www.sourceware.org/pthreads-win32/index.html .
Una vez descargado hay que renombrar el archivo lib Pre-built.2 Pre-built.2\lib\x86\libpthreadGC2.a a libpthread y copiarlo en la carpeta MinGW de tu instalación de Code Blocks, por ejemplo: C:\Archivos de Programa (x86)\CodeBlocks\MinGW\lib .
Si no existen es necesario copiar los archivos de header (.h) que se encuentran en Pre-build.2\include en la carpeta de instalación de MinGW dentro de la carpeta de Code::Blocks, por ejemplo C:\Archivos de Programa (x86)\CodeBlocks\MinGW\include .
Icon.RC

El ejemplo que contiene icon.rc tiene que modificarse para codeblocks, y agregar barras diagonales para evitar errores.

MAIN_ICON ICON "..\\..\\..\\libs\\openFrameworksCompiled\\project\\win_cb\\icon.ico"

[115] "OfTheo/ofxKinect." *GitHub*. N.p., n.d. Web. 19 Nov. 2014. <https://github.com/ofTheo/ofxKinect>.

Indice