UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo del simulador del instrumento Theremin empleando un Leap Motion

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Sancho Martínez, Joan

Tutor: Mollá Vayá, Ramón Pascual

2014-2015

Resumen

El objetivo del proyecto consiste en crear un programa que simula el uso del instrumento musical Theremin y que requiere el uso del dispositivo Leap Motion. La base del programa se ha hecho con ejemplos básicos de otras aplicaciones con librerías de sonido, reconocimiento de gestos y de diseño gráfico. Se ha empleado el lenguaje de programación C++ para desarrollarlo.

Palabras clave: instrumento virtual, theremin, c++, leap motion.

Abstract

The project goal is to create a program that simulate the use of the musical instrument Theremin and requires the use of the Leap Motion device. The basis of the program is made of parts from basic examples of applications with audio, gesture recognition and graphic design libraries. It has used the programing language C++ to develop it.

Keywords : virtual instrument, theremin, c++, leap motion.

Tabla de contenidos

1 Introducción....9
1.1 Motivación....9
1.2 Objetivos....9
1.3 Estructura de la obra....10
2 Estado del arte....12
2.1 El Theremin....12
2.2 Dispositivos de reconocimiento de gestos....13
2.3 Leap Motion....14
3 Análisis del problema....18
3.1 Herramientas básicas....18
3.2 Librerías de sonido....18
3.3 Librerías de diseño gráfico....20
3.4 Herramientas de diseño....22
3.5 Presupuesto....22
3.6 Planificación....26
4 Diseño de la solución....29
4.1 Estructura del programa....29
4.2 Sistema de entrada....30
4.3 Diseño del sistema de audio....31
4.4 Aspecto gráfico....35
5 Implementación....37
5.1 El controlador del Leap Motion....37
5.2 La clase audio....42
5.3 El módulo gráfico....44
5.3.1 Implementación de ventanas....45
5.3.2 Escena gráfica....48
5.3.3 La clase modelo....52
6 Conclusiones....54
6.1 Resultados....54
6.2 Dificultades encontradas....54
6.3 Relación con la carrera....54

6.4 Trabajo futuro.....55
7 Manual de usuario.....57
7.1 Instalación.....57
7.2 Configuración.....57
7.3 Ejecución.....58
8 Bibliografía.....62
9 Glosario.....64

Índice de imágenes

Ilustración 1: Theremin diseñado por la empresa Moog.....12
Ilustración 2: Sistema de coordenadas del Leap Motion.....15
Ilustración 3: Estructura de cómo reconoce el Leap Motion los huesos.....16
Ilustración 4: Diagrama de Gantt de la planificación previa del proyecto.....27
Ilustración 5: Diagrama de Gantt de la planificación real del proyecto.....28
Ilustración 6: Esquema de los módulos del proyecto y sus relaciones.....29
Ilustración 7: Esquema del sistema de ventanas y sus relaciones de acceso. .30
Ilustración 8: Ondas de sonido sinusoidal, cuadrática, triangular y dientes de sierra.....32
Ilustración 9: Pistas de audio idénticas, pero con frecuencias de muestreo distintas.....33
Ilustración 10: Sistema de envoltura ADSR aplicada sobre un sonido.....34
Ilustración 11: Gráfica con las distancias entre la nota A0 y cualquier otra.....40
Ilustración 12: Funcionamiento del método *onFrame*.....41
Ilustración 13: Ciclo de vida del sistema de audio.....44
Ilustración 14: Ciclo de vida de la escena gráfica.....52
Ilustración 15: Ventana de inicio del programa.....58
Ilustración 16: Ventana del simulador del Theremin.....59
Ilustración 17: Ventana de configuración.....60

1 Introducción

1.1 Motivación

Una de las motivaciones por las que me han llevado a elegir este proyecto es la de crear una aplicación que tuviese relación con el mundo de la música. En lo que hace referencia a las aplicaciones que tiene que ver con la música, a parte de los programas de edición y posproducción de audio o los de reconocer canciones a partir de una muestra de audio, se encuentran los que intentan simular, con sonido digital, instrumentos musicales. En esta sección de aplicaciones, se han creado muchos para los instrumentos más comunes de hoy en día, que pueden ser pianos, guitarras, violines o instrumentos de percusión entre otros. En mi caso, en vez de crear otro simulador como los que hay, será crear uno para un instrumento poco común, el Theremin.

Al tener que programar con librerías de sonido, este proyecto me permite realizar un acercamiento a su uso en cuanto al tener que programar un sistema de audio, porque este tipo de librerías no se enseñan en las asignaturas obligatorias de la carrera y pueden servir para diferentes sectores en el mundo de la informática, como en los videojuegos o aplicaciones multimedia.

Por otro lado, el Leap Motion es un dispositivo que ha aparecido hace unos años, con la capacidad de obtener información a partir de la interpretación de gestos, una técnica que últimamente esta siendo explotada por muchas compañías. En este caso, la idea de crear una aplicación que use esta tecnología me puede servir para obtener una idea base de cómo usarlo, para su uso en programas que pueda crear en el futuro que también lo aprovechen.

Por último, el tener que realizar esta aplicación me puede ayudar a comprender mejor como se debe organizar un proyecto de este estilo y también a obtener una formación profesional, teniendo en cuenta que la gente con conocimientos informáticos son cada vez más necesitados en muchos trabajos.

1.2 Objetivos

El objetivo principal del proyecto es desarrollar una aplicación gráfica que simula el uso del Theremin, un instrumento musical que genera sonido partiendo de la distancia que hay entre las manos del usuario y el instrumento.

Otra finalidad de este proyecto es el de usar, como componente indispensable, el Leap Motion para enviar información al programa, mostrando al usuario por una pantalla las

manos que reconozca el dispositivo, informando de como se esta tocando el instrumento.

Entre los objetivos referentes al diseño gráfico de la aplicación, están el de informar al usuario sobre la nota musical que esta tocando en ése instante y mostrar un teclado de piano, que ayude al usuario a relacionar la equivalencia entre tocar un piano y un Theremin.

Resumiendo lo anterior, a partir de éste proyecto se quiere aprender a crear una aplicación musical y saber programarlo, para poder crear un programa al que se le pueda insertar más funcionalidades, haciéndolo más completo y seguir aprendiendo.

Como objetivo paralelo al desarrollo del proyecto está el de incorporar nuevas funciones al UPVGameKernel para mejorar su funcionamiento, partiendo de las que se emplean en la aplicación. El UPVGameKernel es un *kernel* con librerías de programación útiles para el diseño de videojuegos, creado por el departamento DSIC de la facultad de informática de la Universidad Politécnica de Valencia.

1.3 Estructura de la obra

El primer punto, que se encuentra a continuación, consiste en el estado del arte, donde se describe el instrumento que se quiere imitar y cómo está reconocido en la sociedad desde su creación hasta la actualidad. También se habla de los dispositivos de reconocimiento de gestos que se emplean hoy en día, donde se profundiza en la explicación del funcionamiento del Leap Motion.

El siguiente apartado consiste en el análisis del problema, que se trata de analizar los requisitos del programa y las posibles herramientas que se pueden utilizar en el diseño del simulador del Theremin, donde se explica el porqué de la elección de cada una de ellas. También se encuentra un presupuesto de lo que hace falta para el proyecto y una planificación de su duración prevista y real.

Seguido del análisis del problema, se encuentra el diseño de la solución, el apartado que informa de la estructura que formará la aplicación, los tipos de entrada de información que lo compone y la funcionalidad del sistema de audio y gráfico que tendrá.

El cuarto punto se trata de la implementación del programa, que consistirá en una explicación más profunda del funcionamiento del controlador del Leap Motion, el sistema de sonido y la interfaz gráfica de la aplicación.

Después de la implementación, se encuentra el apartado de conclusiones, donde se expone los resultados obtenidos, tanto en el proyecto como en el aprendizaje de nuevas herramientas. También se expone las dificultades encontradas, la relación del proyecto con la carrera y los nuevos objetivos que se pueden emplear en trabajos futuros a partir de los resultados.

En el siguiente punto se encuentra el anexo del manual de usuario, que se trata de una guía por pasos para instalar y ejecutar la aplicación en un ordenador. También muestra la forma de acceder a la configuración de los dispositivos empleados por el programa.

Por último, se hayan los anexos de bibliografía, que informan de dónde se ha obtenido información para hacer el proyecto, y glosario, que explica el significado de algunos términos que se emplean en el campo de la informática y que se muestran a lo largo de este trabajo escrito.

2 Estado del arte

2.1 El Theremin

El Theremin es un instrumento musical inventado por Leon Theremin en 1919 y fue uno de los primeros instrumentos electrónicos, donde lo más curioso es que para usarlo no hay que tener ningún contacto físico con él.

Este instrumento se ejecuta acercando y alejando la mano de cada una de las antenas correspondientes, sin llegar a tocarlas. La antena de la derecha suele ser recta y en posición vertical, y sirve para controlar la frecuencia o tono, de modo que cuanto más cerca esté la mano derecha de la misma, más agudo será el sonido producido. La antena izquierda es horizontal con forma de bucle, y sirve para controlar el volumen del sonido generado, de la forma que cuanto más cerca de la misma esté la mano izquierda, más bajo será el nivel sonoro.

Originalmente, el timbre de lo theremines se parece a algo situado entre un violonchelo y la voz humana. En la actualidad existen modelos con la tecnología *MIDI*, que permite la inclusión de cualquier timbre al instrumento utilizando archivos *VST* o *samples*, aunque los sonidos que puedan producir no sean muy parecidos a los de los instrumentos que los generan en la realidad, al no estar pensado el diseño original en ese sentido.

Ilustración 1: Theremin diseñado por la empresa Moog

La gente lo conoce probablemente por su uso en películas de ciencia ficción y terror de la década de los 40 y 50 para ambientar escenas, donde destacan *Ultimátum a la Tierra* de Robert Wise, *Recuerda* de Allfred Hitchcock y *Días sin huella* de Billy Wilder, aunque también ha llegado a usarse como instrumento solista en un entorno orquestal, como hizo Clara Rockmore utilizándolo en un repertorio romántico clásico. Incluso muchos grupos de música lo han utilizado para obtener sonidos psicodélicos. Entre ellos se encuentran Led Zeppelin, Pink Floyd, Muse, The Mars Volta y The Gathering entre muchos.

Actualmente existen músicos que todavía usan el Theremin y son reconocidos a nivel internacional, donde algunos de ellos son Jean Michael Jarre, Lydia Kavina, Barbara Buchholz y Ernesto Mendoza [1].

Hoy en día, la empresa mejor conocida por la venta de este instrumento musical es Moog[1], reconocida principalmente por la venta de instrumentos digitales como el sintetizador. Otras empresas que hacen la competencia a este tipo de venta son Harrison Instruments, Theremaniacs, PaiA electronics y Jaycar, que también venden en kits de componentes sueltos[2] para que el cliente lo construya por sí mismo.

En el mundo del software se han creado varias aplicaciones que simulan este instrumento[3], aunque todas ellas requieren el uso de una pantalla táctil en el caso de los *smartphones* o la necesidad de usar un teclado e incluso el mismo instrumento con conexión *MIDI* para el caso de los ordenadores.

En mi caso se trata de desarrollar una aplicación como esas utilizando un dispositivo de reconocimiento de gestos, que nos pueda permitir ejecutar el simulador como si se estuviera tocando un Theremin de verdad.

2.2 Dispositivos de reconocimiento de gestos

El reconocimiento de gestos consiste en interpretar gestos humanos a partir de fórmulas matemáticas [2]. Los gestos pueden ser cualquier movimiento corporal, pero normalmente se originan a partir de la cara o las manos. Esto permite a la gente comunicarse con una máquina e interaccionar con naturalidad sin dispositivos mecánicos. Usando este concepto, es posible usar los dedos en un espacio libre para relacionar movimientos del cursor con el movimiento del usuario y podría hacer que los dispositivos convencionales de entrada de hoy en día se hagan redundantes, como el ratón, el teclados o incluso las pantallas táctiles.

La capacidad de realizar un seguimiento de los movimientos de una persona y determinar qué gestos pueden ser, se pueden lograr a través de diversos dispositivos. Entre ellos se mencionan:

Kinect, de Microsoft [3], que utiliza una cámara para obtener información de los gestos que genera cualquier persona delante de ella. En un principio su utilidad sólo estaba disponible en las consolas de la misma compañía, las Xbox 360 y Xbox One, pero más

1 MOOG MUSIC INC.. *Theremins.* <http://www.moogmusic.com/products/Etherwave-Theremins>

2 THE THEREMIN WORLD. *Theremins and Theremins kits. <http://www.thereminworld.com/theremin-store>*

3 THE THEREMIN WORLD. *Theremin Software.* <http://www.thereminworld.com/Software>

tarde Microsoft creo una *SDK* programada en C++ para que se pudiera usar en el sistema operativo Windows 7 de Windows. Para poder generar información de los gestos, el Kinect obtiene información de todo el cuerpo, aprovechando desde los gestos de la cara hasta los movimientos de las piernas.

Cámara Intel RealSense 3D, también denominada como F200 y diseñada por Intel Corporation [4], esta formado por diversos componentes que permiten procesar sonido, escanear caras, generar escenarios de realidad aumentada y funcionar como una cámara web entre otras cosas. Es compatible con los sistemas operativos mas recientes de Windows, se programa con C++, C#, Java o JavaScript y se puede controlar con el motor gráfico Unity.

Leap Motion, creado por Leap Motion Inc. [5], que solo permite el reconocimiento de gestos de las manos mediante el uso de sensores por infrarrojo. El dispositivo en el que está montado tiene unas dimensiones físicas menores comparado con los ya mencionados. Es compatible con los diversos sistemas operativos de Windows, Linux y Apple y tiene distintas *APIs* para poder programar con él en C++, Java, C# y Python entre otros lenguajes de programación. A parte, se puede complementar con dispositivos visuales, permitiendo la creación de aplicaciones de realidad virtual y aumentada. También tiene compatibilidad con diversos motores gráficos como Unity y Unreal Engine.

Ring Zero, de la compañía Logbar [6], que está pensado para enviar información de gestos a un *smartphone* con el sistema operativo Android 4.4 de Android o IOS 7 de Apple. Este dispositivo tiene forma de anillo, compuesto por luces LED y un vibrador, para informar de notificaciones del *smartphone*. Como desventaja, esto solo puede reconocer gestos del dedo de la mano al que esta puesto, aunque permite al usuario diseñar sus propios movimientos personalizados. En lo referente a la programación, utiliza una API con node.js, pero que en el futuro la compañía permitirá a los programadores el uso de una SDK para Android, IOS e incluso la posibilidad de aprovechar el uso de los dispositivos de Oculus para crear aplicaciones de realidad virtual.

2.3 Leap Motion

Como ya hemos mencionado, este componente reconoce los gestos de las manos, contando también los dedos y los objetos situados entre ellos con las mismas proporciones. Tiene la capacidad de hacer los cálculo de reconocimiento con una alta precisión y velocidad, permitiendo obtener muchos estados reconocidos con mucha información detectada en poco tiempo.

También se ha mencionado que para reconocer estos gestos utiliza sensores de infrarrojo, añadiendo que su sensor se emite en dirección vertical al espacio en el que usa, en el caso de que el dispositivo está posicionado correctamente, y se proyecta en el ambiente como si se tratase de un cono del revés.

La detección y el seguimiento funciona mejor si el dispositivo se sitúa en un ambiente iluminado sin recibir la luz directamente y tiene una vista bien contrastada de la silueta del objeto que intenta reconocer. El software del Leap Motion combina sus sensores con información interna del modelo humano de las manos, para ayudar a poner condiciones sobre la forma del objetos que debe reconocer.

Aplica un sistema cartesiano de coordenadas, donde el origen se sitúa en el centro del dispositivo. Las coordenadas X y Z se sitúan en un plano paralelo al de la superficie donde se encuentra el Leap Motion, de modo que el eje X se encuentra en paralelo al borde alargado del dispositivo, con el incremento de su valor hacia la derecha, y el eje Z en perpendicular al de la X, con el incremento a valores positivos en dirección al usuario. El eje de coordenadas Y ya se había mencionado antes hacia donde apunta, siendo perpendicular a los otros dos ejes y sin tener valores negativos.

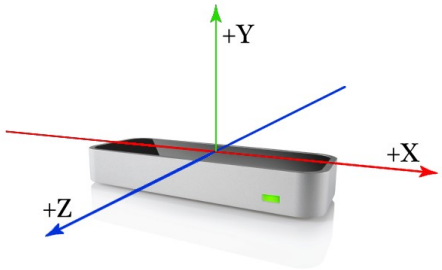

Ilustración 2: Sistema de coordenadas del Leap Motion

Como el controlador del Leap Motion reconoce las manos, dedos y herramientas en su campo de visión, este proporciona al usuario todo el conjunto de información en un objeto *frame*. Cada *frame* representa un estado con su lista de entidades detectadas, así como los gestos reconocidos. Este objeto es la base de donde el usuario obtiene toda la información posible.

Anteriormente se había mencionado que el software del dispositivo contiene un modelo de las manos. Pues este modelo también se utiliza para hacer un seguimiento predictivo de las manos reconocidas incluso cuando parte de una no es visible. El modelo de las manos siempre ofrece la posición de los cinco dedos, aunque el seguimiento es óptimo cuando la silueta de toda una mano es claramente visible. El Leap Motion utiliza las partes visibles de la mano, su modelo interno y las observaciones para calcular la posición más probable de las partes de ella que no son visibles por el controlador.

Aunque no se aprecia en muchas aplicaciones con el Leap Motion, este dispositivo es capaz de reconocer más de dos manos en un mismo *frame*, aunque se recomienda mantener un máximo de dos manos en su campo de visión para poder obtener una calidad óptima en el seguimiento de movimientos.

También se había mencionado que proporciona información de cada dedo de la mano. Si todo o parte del dedo no es visible por el controlador, las características de ese dedo se estiman basándose en observaciones recientes y por el modelo interno de la mano. Cada dedo contiene información sobre su identidad en la mano que se encuentra y de los huesos que lo componen.

Estos huesos que lo forman son el metacarpiano, falange proximal, falange media y falange distal. En el modelo del dedo pulgar se encuentra el defecto de que no se ajusta al sistema de anatomía estándar, porque el verdadero pulgar tiene un hueso de menos comparado con el resto. Sin embargo, para facilitar al programador, al modelo del Leap Motion se le ha añadido un hueso de más en la base del pulgar con longitud cero, haciendo referencia al metacarpiano y substituyendo el nombre del resto de los huesos, haciéndolos equivalentes a los otros dedos [7].

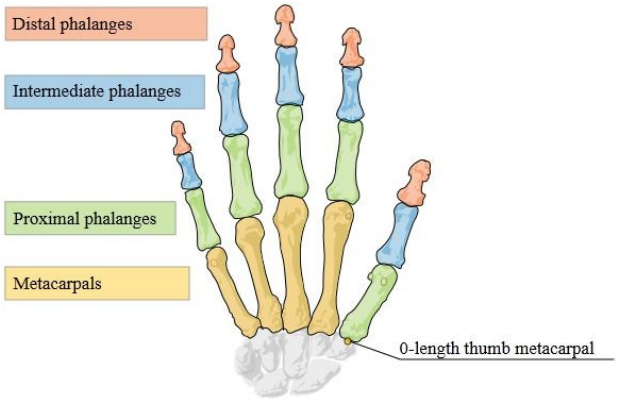

Ilustración 3: Estructura de cómo reconoce el Leap Motion los huesos

El software también puede reconocer ciertos patrones de movimiento como gestos que puedan indicar una intención por parte del usuario. Puede reportar los gestos en un *frame* de la misma manera que informa de los datos referentes al seguimiento de las manos. Los movimientos que reconoce el dispositivo son:

Generación de un circulo hecho por un dedo.

Movimiento linear de la mano y sus dedos.

Movimiento de tocar con un dedo, como si se tratase de tocar la tecla de un teclado de ordenador.

Desplazamiento de un dedo, como si se fuese a tocar la pantalla del ordenador.

3 Análisis del problema

3.1 Herramientas básicas

Para la creación de una aplicación que consiste en simular un instrumento mediante un dispositivo de gestos, se tiene que tener en cuenta que debe contener un sistema de retro-alimentación, que informa al usuario con una salida de video e audio, mientras que la aplicación recibe como información de entrada los gestos y movimientos de manos que vaya generando el Leap Motion. Con esta idea, la aplicación requiere de *plugins* o librerías de programación que puedan generar información sonora y visual.

Como herramientas base para el proyecto, se ha pensado en utilizar como lenguaje de programación el de C++, porque a parte de seguir siendo un estándar para crear muchos programas de distintos tipos, permite la posible inserción de funciones creadas por el UPVGameKernel, diseñado por el DSIC de la facultad de informática, que también esta escrito en ese lenguaje.

El UPVGameKernel se trata de un *kernel* que aprovecha librerías externas para el uso de sonido, inteligencia artificial y de *sockets* para conectarse por Internet entre otros. También contiene funcionalidades para el uso de algunos dispositivos de entrada, donde se encuentran métodos ya implementados para el Leap Motion, aunque en este proyecto no nos interesan por la obsolescencia de sus librerías, que no aprovechan tanto el dispositivo como lo hacen las bibliotecas de hoy en día. Por ese motivo, en vez de utilizar las funciones del Leap Motion ya creadas por este *kernel*, se le añadirán de nuevas, actualizando además sus librerías.

Como es posible que se tenga en cuenta el uso de archivos por parte del UPVGameKernel, el entorno en el que se desarrolla el programa debe de ser el Visual Studio. Por los requisitos mencionados anteriormente y por la posibilidad de acceder a su uso por parte del ordenador que se utiliza para la realización del proyecto, la aplicación se realiza con el sistema operativo Windows 7.

3.2 Librerías de sonido

En la actualidad se puede encontrar una gran variedad de librerías de sonido, donde cada uno de ellos tiene sus características que los hacen más distintos a los otros, sus ventajas e inconvenientes. Muchas de estas bibliotecas también contienen funciones de diseño gráfico, de modo que se podrían catalogar mejor como librerías multimedia. Entre todas ellas se han tenido en cuenta las siguientes:

Allegro, de Allegro developers [8], que es una librería multiplataforma dirigida a la creación de videojuegos con funciones multimedia, que soporta la manipulación de imágenes y de texto a parte de sus controles de audio y de *MIDI*. Su biblioteca esta escrita en C, pero esta diseñado para poder usarlo con C++, Objective-C, Lua y Python entre otros lenguajes de programación. Usa una licencia de código abierto y tiene una comunidad de usuarios que contribuyen en la creación de librerías externas para añadir nuevas funciones.

Open Audio Library, más conocido como OpenAL y desarrollado por Creative Technology [9], es otra librería con una *API* multiplataforma diseñada de forma eficiente para la generación de canales de salida de audio espacializado simulando espacio 3D. Gracias a esta característica, se ha utilizado en diversas plataformas de videojuegos por su forma de dar realismo en el ambiente de cualquier juego.

Open Media Library, también denominada OpenML y diseñado por Khronos Group [10], es más bien un estándar de entorno multiplataforma que permite la captura, transporte, procesamiento, visualización y sincronización de diferentes medios, tanto gráficos 2D o 3D como flujos de audio o video. Soporta flujos asíncronos entre *hardware*, OpenGL para el procesamiento de video acelerado y también tiene control de visualización profesional. OpenML es de código abierto como otras de las librerías creadas por la misma compañía.

Simple and Fast Multimedia Library, conocido por las siglas SFML [11] y creado por SFML-Team, también es una librería multimedia. Su arquitectura contiene varios módulos que permiten dar soporte de bibliotecas gráfica, de audio y el uso de *sockets* para conectarse a la red. Está escrito en C++ y permite el desarrollo de programas en C, Java, Python, Ruby y Rust entre otros lenguajes. Es multiplataforma, compatible con Windows, Linux y OS X, y también se encuentra en el sector del software de código abierto.

Simple DirectMedia Layer, desarrollado por Sam Lantinga [12] y nombrado abreviadamente como SDL, es un conjunto de bibliotecas multimedia implementadas en C que proporcionan funciones básicas para realizar operaciones de dibujo en 2D y gestión de sonido y música. Es compatible con otros lenguajes de programación, como C++, Java, Erlang, Python o Lua entre otros, y proporciona herramientas para el desarrollo de videojuegos. Estas librerías también se distribuyen bajo la licencia de código abierto, cosa que le ha permitido un gran avance en la evolución de su contenido.

FMOD, de Firelight Technologies [13], que es más bien un motor de audio para videojuegos, permite ejecutar y mezclar diversos archivo de audio en distintos formatos. Aun así, contiene una *API* de bajo nivel para programar en C, C++ y C#, y es

compatible en muchas plataformas, tanto en los sistemas operativos de computadora como en las consolas de videojuegos. También puede ser integrado en motores gráficos, como CryEngine, Unity y Unreal, para poder programar aplicaciones tanto para el uso básico como profesional.

DirectSound, creado por Microsoft, es un componente software que aprovecha las librerías de audio de DirectX [14], una biblioteca multimedia. Proporciona muchas capacidades como grabación y mezclado de audio, adición de efectos al sonido, posicionamiento del sonido en un espacio 3D. Su *API* permite el uso de C y C++ para programar con él y solo es compatible con los sistemas operativos de Windows.

Para las necesidades del proyecto, se ha elegido las librerías de sonido FMOD, porque es multiplataforma, con lo que se pueden aprovechar las mismas bases aprendidas en una plataforma para usarlas en otras. También porque su motor se usa en aplicaciones profesionales y para obtener una formación en su uso, aunque a las empresas que lo usan posiblemente prefieran a gente que sepa usar su herramienta ya diseñada que a aquellos que saben emplear sus funciones de bajo nivel. Otro motivo de su elección es la de la posible aportación de funciones ya creadas por la *API* de UPVGameKernel, que entre las bibliotecas en las que se basan su uso está el FMOD, a parte de las utilizadas para OpenAL y SDL.

3.3 Librerías de diseño gráfico

En los objetivos de este proyecto se pide que haya una referencia visual del simulador del instrumentos, mostrando la forma del Theremin y de las manos entre otros. Para ello hace falta el uso de librerías gráficas, donde las *APIs* más utilizadas para su uso son:

Open Graphic Library, o OpenGL, diseñado por Silicon Graphics Inc. y desarrollado por Khronos Group [15], que en realidad es una especificación estándar que define una *API* multilenguaje, donde los fabricantes de hardware crean librerías de funciones que se ajustan a sus requisitos. A partir de este estándar se han formado bibliotecas externas que añaden características no disponibles en el propio OpenGL. Estas librerías pueden ser:

Glu, el cual ofrece funciones de dibujo de alto nivel basadas en primitivas de OpenGL.

Glut, que se trata de una *API* multiplataforma que facilita el manejo de ventanas e interacción por medio de teclado y ratón.

Glui, que consiste en una interfaz basada en Glut, que proporciona elementos de control como botones o cajas de selección.

Glew, el cual es una librería multiplataforma que ayuda en consultar y cargar extensiones de OpenGL.

Direct3D, de la propiedad de Microsoft, que forma parte del conjuntos de bibliotecas multimedia DirectX [14]. Solo está disponible para los sistemas de Windows y para sus consolas Xbox. La *API* que contiene permite su uso en C y C++, y está compuesto por otras dos:

API de modo inmediato, que da soporte a todas las primitivas de procesamiento 3D que permiten las tarjetas gráficas, como las luces, los materiales o el control de profundidad. Además trabaja fundamentalmente con los llamados dispositivos, que son los encargados de realizar la renderización de la escena.

API de modo retenido, construido sobre el anterior y presenta una abstracción de nivel superior ofreciendo funcionalidades pre-construidas de gráficos como jerarquías o animaciones. El modo retenido ofrece muy poca libertad a los desarrolladores, siendo el modo inmediato el que más se usa.

En la elección de la *API* para el uso de las librerías gráficas se ha pensado en utilizar la de OpenGL, por su posibilidad de ser usado en distintos sistemas y porque ya se había obtenido experiencia en su uso mediante la librería externa Glut.

Por otro lado, existen librerías externas de diseño basadas en OpenGL que permiten mejorar la apariencia de una aplicación y que se han recomendado utilizar en el proyecto. Éstas son:

Qt [16], que se trata de un marco de trabajo multiplataforma para crear interfaces gráficas de aplicaciones, además de que contiene soporte en los compiladores que emplea Visual Studio, entorno de programación imprescindible para el proyecto. A parte de permitir hacer interfaces, también da la posibilidad de crear funciones que permiten el acceso a la red de Internet y el uso de dispositivos *Bluetooth.* Sus bibliotecas están escritas en C++ y permite la inserción de otras librerías externas OpenGL para facilitar su uso en la implementación de una escena gráfica. Contiene diferentes tipos de licencia de uso, de modo que permite un uso gratuito, pero con limitaciones que no existen en su licencia de pago.

Open Asset Import Library, también conocida como Assimp [17], que consiste en unas librerías gráficas, que permiten importar en un programa archivos referentes a modelos de diseño creados por aplicaciones de modelado en 3D, como el Blender,

Maya o 3ds Max. Están escritas en C++, aunque también contiene interfaces para el uso de C y permite la unión de complementos externos para programar con otros lenguajes de programación, como C#, Python y BlitzMax.

3.4 Herramientas de diseño

Como se pretende emplear modelos gráficos para mejorar el aspecto de la escena gráfica, es necesario utilizar una aplicación que permita crealos y exportarlos para que se puedan utilizar. Entre las herramientas más famosas están las siguientes:

Blender [18], una herramienta multiplataforma que a aparte de crear diseños, permite la creación de animaciones, edición de audio y video, y desarrollo de videojuegos, gracias a su motor gráfico interno. Contiene una gran variedad de primitivas geométricas, permite el uso de luces y renderizado a tiempo real. Esta aplicación es de licencia de distribución libre y se emplea también para añadir efectos tridimensionales en películas[4].

3ds Max, que es una aplicación de creación de gráficos y animaciones en tres dimensiones creada por la empresa Autodesk [19]. Su arquitectura está hecha por *plugins* y se trata de uno de los programas de animación tridimensional más utilizado para la creación de videojuegos, anuncios y películas. Contiene también una extensa variedad de primitivas de objetos, la utilización de sistemas de partículas, iluminación y creación de interfaces de usuario personalizable. Se permite su uso en varios sistemas operativos, como Microsoft Windows, Mac OS X o Linux, y tiene sólo licencias de pago, a excepción de su versión para estudiantes, que contiene sus limitaciones.

Maya, también desarrollada por Autodesk [20], es un programa informático utilizado para el desarrollo de gráficos en 3D por computadora, efectos especiales y animaciones. Se caracteriza por su potencia de cálculo y las posibilidades de expansión y personalización de su interfaz gráfica y herramientas. Su distribución es parecida a la de 3ds Max, por su compatibilidad con diferentes sistemas operativos y por su licencia de pago, aunque tiene una versión gratuita para su uso no comercial.

Como el uso de éste tipo de herramienta sólo se va a utilizar para crear en un principio un modelo del Theremin en tres dimensiones, se ha pensado en utilizar Blender, por su licencia de uso gratuito, su comunidad de usuarios que resuelven dudas y proporcionan tutoriales para su uso y por sus pequeños conocimientos en su uso por parte de alguna asignatura de la carrera de informática.

3.5 Presupuesto

4 BLENDERNATION. *Blender Used in Previz for Captain America: the Winter Soldier.* <http://www.blendernation.com/2014/05/02/blender-used-in-previz-for-captain-america-the-winter-soldier/>

Para la creación del proyecto, se ha realizado una estimación del presupuesto resultante teniendo en cuenta el coste del personal, de la oficina donde se desarrolla el proyecto, de su coste de limpieza, y de los recursos que se deben utilizar, tanto para la creación del programa como para probarlo en un ordenador distinto.

Sobre el coste del personal se supone que solo hay un programador que se encarga de realizar todo el proyecto, donde su duración estimada es de un total de cuatro meses. Teniendo en cuenta que el sueldo mínimo para un desarrollador es de 1000 euros netos al mes[5] y que no hay días de vacaciones durante el proceso de implementación, se calcula un coste total mínimo de 4000 euros en total.

Referente al gasto en el uso del local de trabajo, se calcula que el coste del alquiler de una oficina cuesta como mínimo 120 euros al mes, suponiendo que se necesite una sala de 20 m², costando seis euros el metro cuadrado[6]. Como se va a necesitar trabajar en una oficina para el proyecto durante el tiempo estimado, se obtiene un gasto de 480 euros en total.

Junto a este coste, también se le añade el precio de contrato de asistentes para la limpieza de la oficina, que sólo se necesitará su servicio una vez a la semana, de modo que se deberá usar esto un mínimo de diecisiete veces. Suponiendo que el tiempo necesario para la limpieza de la sala al día es de 10 euros, se deberá gastar unos 170 euros.

En el caso de los recursos que se usan, por el lado de *software* se han utilizado librerías y aplicaciones gratuitas, gracias a sus licencias de código abierto y de uso no comercial. Estos se han utilizado durante el proceso:

Entorno de programación Visual Studio.

Aplicación y bibliotecas de FMOD Sound System.

Librerías Glut de OpenGL.

Bibliotecas y aplicación de Qt.

Librerías de Assimp.

Aplicación Blender.

5 TECNOEMPLEO. *Encuentra empleo en la mayor bolsa de Trabajo TIC*. <Tecnoempleo.com>

6 IDEALISTA. *Alquiler de oficinas en Torrent*. <http://www.idealista.com/alquiler-oficinas/torrent-valencia/>

Aplicación Gimp.

SDK del Leap Motion.

LibreOffice para generar documentos de texto y diagramas.

El único *software* que se ha tenido que pagar es el de los sistemas operativos utilizados en cada ordenador, que se tratan de Windows 7 para la implementación de la aplicación y Windows 8 para el ordenador de pruebas.

Sobre el *hardware*, el proyecto se ha realizado en un ordenador portátil que se puso en venta hace unos años, con las siguientes características:

Procesador Intel Core i7-2670QM, con una velocidad de procesamiento de 2.2 GHz y capacidad de *overclocking* hasta 3.1 GHz.

Memoria *RAM* de 6GB de capacidad.

Tarjeta gráfica Nvidia GeForce GT 540M de 2GB de capacidad.

El computador en el que se realizan las pruebas de funcionamiento del programa es un ordenador fijo de torre con componentes diseñados hace tiempo, para comprobar la eficiencia del programa respecto a ordenadores con una potencia de cálculo inferior al del que se ha usado para programarlo. Sus componentes importantes son los siguientes:

Procesador Intel Core Duo Quad Q6600, con 2.4 GHz de velocidad de cómputo.

Memoria *RAM* con 4GB de capacidad.

Tarjeta gráfica Nvidia GeForce GTX 550 Ti de 1GB de capacidad.

Sobre el coste de las tecnologías, sólo se tiene en cuenta el precio de ambos ordenadores, que ya venían con el sistema operativo y el conjunto de componentes *hardware* ya incorporado, de modo que el ordenador portátil llega a costar 700 euros y el ordenador de torre unos 600 euros. En total se necesita gastar en materiales 1300 euros.

A todo esto también se debe añadir el coste del dispositivo Leap Motion, que sólo se ha utilizado uno y se ha tenido que tener en cuenta también el precio de envió a parte de su valor en el mercado, que en total llega a valer 100 euros. Añadiéndolo al coste del material empleado, se alcanza un gasto de 1400 euros.

A continuación se muestra el resumen del presupuesto inicial del proyecto, donde se calcula el coste total que se requiere en su desarrollo.

Coste del personal	4000 €
Coste del alquiler de la oficina	480 €
Coste de mantenimiento de la oficina	170 €
Coste del material del proyecto	1400 €
Total	**6050 €**

Tabla 1: Tabla de presupuestos planteada al inicio del proyecto

Al final del desarrollo del proyecto, se ha observado que se ha necesitado más tiempo en su realización, de modo que ésto afecta en el presupuesto total, en concreto al coste del personal, alquiler de la oficina y de su mantenimiento.

Al principio se predecía que la duración del proyecto sería de cuatro meses, mientras que el resultado final ha sido de cinco meses. Esto cambia el coste del personal mínimo a 5000 euros. En cuanto al alquiler de la oficina, su precio sube a un pago mensual de más, con lo que el coste final de alquiler es de 600 euros. Referente al mantenimiento de la oficina, se requiere el servicio de limpieza durante cuatro semanas más, de modo que el coste de mantenimiento sube a 210 euros en total. En lo que hace referencia al material del proyecto, no se ha necesitado nada más de lo previsto, con lo que se mantiene el mismo coste del material del proyecto.

En la tabla siguiente se muestra el resumen del coste final del proyecto, donde se calcula el total y la desviación en comparación al presupuesto inicial.

Coste del personal	5000 €
Coste del alquiler de la oficina	600 €
Coste de mantenimiento de la oficina	210 €
Coste del material del proyecto	1400 €
Total	**7210 €**
Total inicial	6050 €
Diferencia de costes	1160 €
Desviación	**19,17 %**

Tabla 2: Tabla de presupuestos final y desviación con la inicial

3.6 Planificación

En este apartado se presenta el diagrama de Gantt previo que representa la duración aproximada del proyecto en un principio, que requiere un mínimo aproximadamente de cuatro meses de trabajo por parte del desarrollador en el caso de que se dedicase sólo a eso.

Este diagrama se divide en el tiempo necesario para buscar información, implementar el módulo controlador del Leap Motion, crear el sistema de audio, diseñar la interfaz gráfica, generar la memoria del proyecto y corregir los errores que puedan aparecer en la ejecución del programa y que hayan en el contenido de la memoria del proyecto. Se puede ver que hay semanas en las que el desarrollador debe trabajar en distintas secciones del proyecto, por la posible relación que puedan haber entre ellos en el momento de implementar el programa. También se tiene en cuenta que el desarrollador ya tiene suficientes conocimientos en la sección del diseño gráfico, demostrando que se emplearía menos tiempo en ello.

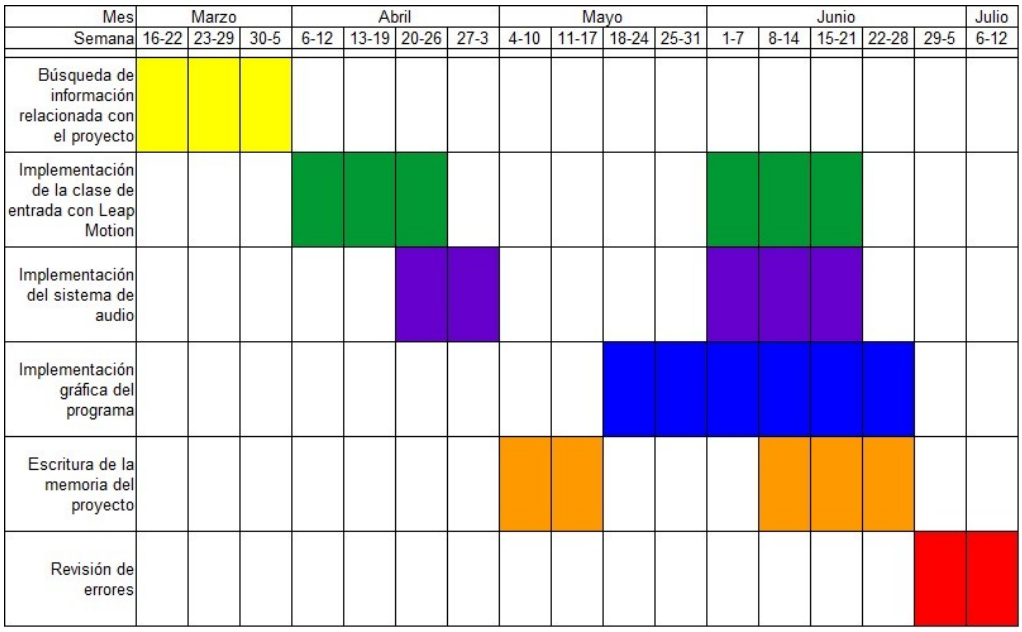

Ilustración 4: Diagrama de Gantt de la planificación previa del proyecto

La planificación final ha sido muy distinta por falta de tiempo y por la inserción de nuevas librerías de diseño gráfico. Los siguientes problemas son los que han afectado en la planificación:

El tener que terminar el trabajo por parte de la carrera de la universidad ha obligado a poder empezar a realizar el proyecto un mes más tarde.

En la búsqueda de las herramientas necesarias para desarrollar el proyecto se ha necesitado más tiempo. Éste problema ha ocurrido sobretodo en el caso de las librerías necesarias para el uso de sonidos, porque se ha intentado buscar a fondo uno que al final no podía cubrir todos los objetivos planteados en el proyecto.

El uso de una estructura inicial mal hecha ha impedido durante un periodo de tiempo el poder avanzar en la implementación, de modo que se ha tenido que adaptar todo el código a una estructura basada en objetos. El problema apareció cuando se iba a unir en la aplicación el sistema de audio, implementado a parte, con el módulo del programa que crea la escena gráfica.

Las recomendaciones por parte del tutor de añadir nuevas funcionalidades proporcionadas por otras herramientas para mejorar la idea principal del proyecto, ha requerido un tiempo extra por la necesidad de informarse y obtener práctica en el uso de dichas herramientas. Esto se ha visto en el uso de las librerías de Qt, que mejoran el aspecto visual de la aplicación mediante interfaces gráficas y se ha necesitado entender cómo se debían emplear en el proyecto. También se ha visto éste problema con las bibliotecas de Assimp, para importar escenas gráficas, que se ha empleado mucho tiempo en la búsqueda de información sobre su uso.

La necesidad de cambiar algunas librerías por otras tras haber implementado con ellas parte del programa ha provocado una ralentización en el progreso del proyecto. En la aplicación ha surgido el problema cuando se pensó en crear la interfaz gráfica a partir de Qt después de haber implementado parte del programa con Glui, que también sirve para añadir interfaces.

Los problemas de incompatibilidad de algunas librerías de programación con el entorno de programación han provocado la necesidad de gastar tiempo en buscar soluciones para resolverlos. Esto ha ocurrido en el caso de utilizar la herramienta Qt para el proyecto, porque en un principio su última versión tenia problemas de compatibilidad con el entorno de Visual Studio.

Como resultado de todos éstos problemas, la planificación final ha sido bastante distinta a la inicial, tanto en el tiempo necesario al que se debía aplicar a cada parte del proyecto como en el tiempo total para hacerlo.

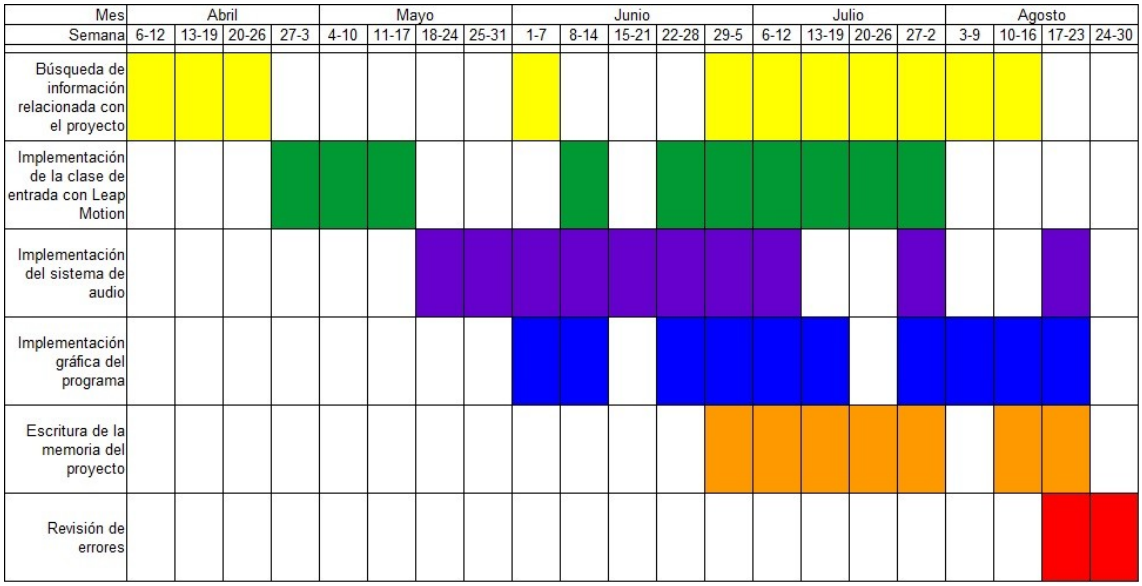

Ilustración 5: Diagrama de Gantt de la planificación real del proyecto

4 Diseño de la solución

4.1 Estructura del programa

El simulador se puede dividir en tres módulos: el sistema de sonido, el renderizado gráfico y el controlador del Leap Motion.

El módulo con el sistema de sonido se usa para generar el sonido del programa, generando un oscilador que cambia su frecuencia de tono según le vayan enviando notificaciones para que la actualice. También se encarga de actualizar la forma de la onda del sonido y su nivel de volumen, dependiendo de como lo quiere el usuario.

El controlador del Leap Motion se encarga de darle funcionalidad al dispositivo para que actúe en relación al programa. Esta funcionalidad consiste en almacenar en cada momento la posición de las manos y el valor de la frecuencia y volumen del sonido que se deberá escuchar en ese instante. Esta información se envía a las otras clases para actualizar el sistema de sonido y el aspecto gráfico de la aplicación.

El tercer módulo es la que genera la interfaz que podrá ver el usuario y que ejecuta el programa. Esta se debe de encargar de dibujar en pantalla la forma del instrumento musical y las manos que el dispositivo de reconocimiento de movimientos capture en cada momento. A parte también muestra en forma de texto la frecuencia del tono y la nota musical del sonido que se está generando en ese momento a partir de la información que le envía el módulo del sistema de audio.

Se supone que cada uno de los objetos que representan cada uno de ellos se ejecutan en bucles paralelos entre ellos, enviando información a las otros en el momento que lo pidan para actualizar parte de sus estados.

La dependencia que hay entre ellos se muestra en la siguiente imagen, donde cada flecha representa en que dirección se trasmite la información de un módulo a otro. También muestra cual de ellos contiene el programa principal de la aplicación y por donde se obtiene la información por parte del usuario, mostrando también los módulos que generan la salida de información en forma de sonido o video.

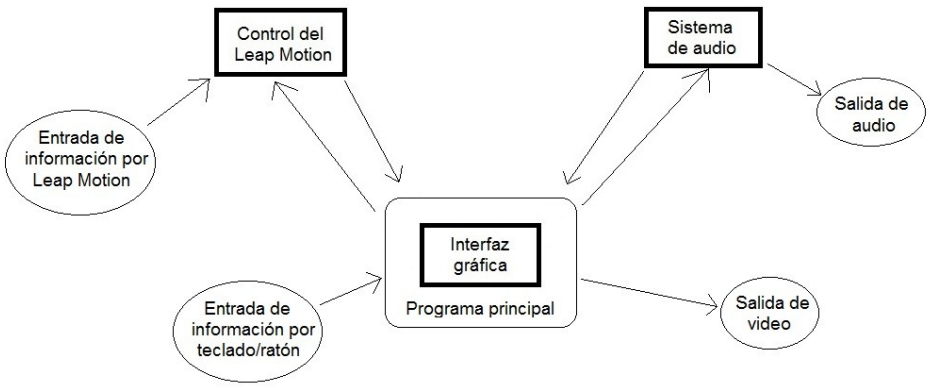

Ilustración 6: Esquema de los módulos del proyecto y sus relaciones

Referente al sistema de ventanas de la aplicación, está compuesto por aquellas que son básicas: la ventana de inicio, la principal del simulador, la de los créditos del programa y la de configuración de la aplicación. La forma de acceder a cada una de estas ventanas por parte del usuario se puede ver en el esquema de abajo, donde también se muestra desde donde empieza el programa y a partir de que ventanas se puede cerrar la aplicación.

Se tiene que tener en cuenta en el esquema que los enlaces de doble sentido permiten acceder a esa ventana para después volver a aquella en la que se encontraba antes el usuario. Como ejemplo esta el caso de acceder a la ventana de créditos desde la ventana del simulador, que una vez llegado a esa esa ventana, sólo se puede volver a la del simulador, sin poder acceder a la ventana de inicio.

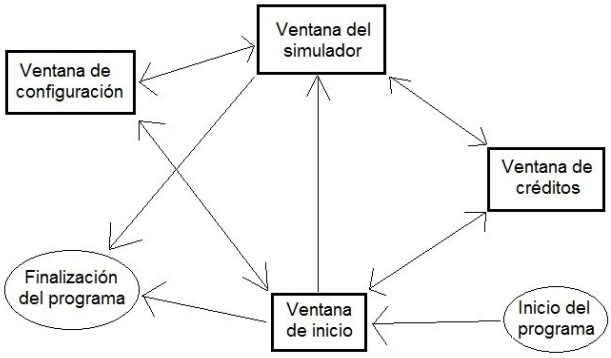

Ilustración 7: Esquema del sistema de ventanas y sus relaciones de acceso

4.2 Sistema de entrada

En el diseño de la entrada de información que el usuario envía al programa se permite el uso de tres dispositivos, implementados en diversas clases de la aplicación. El por qué de esto es por las limitaciones que tienen cada uno de ellos, aunque en un futuro se podrá mejorar la funcionalidad de uno de ellos para que pueda hacer la mayoría de las acciones que se permiten hacer con los otros.

Uno de ellos, y que es el fundamental por lo que pide en los objetivos del proyecto, es el Leap Motion, con el que ya hemos explicado en los apartados anteriores su funcionamiento y como actúa en nuestro programa.

Otro dispositivo es el teclado, donde sus funciones están implementadas en el módulo que se encarga de la parte gráfica de la aplicación. A partir de este dispositivo, el usuario puede activar y desactivar funciones de dentro del sistema de audio y de la interfaz gráfica. Estas funciones son:

Activación del sonido, que partiendo de una selección de tipos de osciladores, avisa al sistema de sonido de cómo debe de ser la onda del sonido que se va a generar. Para poder activar el sonido, primero se debe detener el audio que esté sonando anteriormente.

Detención del sonido, que como su nombre indica, sirve para detener el sonido que esté ejecutando el programa en ese instante.

Activación o apagado de "Modo Afinado", que permite al usuario generar en el sonido sólo las frecuencias de audio que hacen referencia a las notas musicales de la escala estándar. Cuando se mueve la mano sobre el Theremin, el instrumento genera un sonido con un rango de frecuencias que van desde la nota que hace referencia a la posición inicial de la mano hasta otra nota, equivalente a la posición final en la que se encuentra la mano. Con el llamado "Modo Afinado", se quiere que suene solo aquellas frecuencias de sonido que equivalen a notas musicales, sin tener que generar el sonido del resto de frecuencias.

Mostrar o esconder teclado de piano, que muestra en pantalla un teclado de piano. Este teclado pinta de un color distinto aquella tecla musical equivalente a la nota que esté generando el Theremin en ese momento. En el caso de que no sea exacta a la que genera un piano, se escoge aquella tecla donde su frecuencia de sonido se aproxima a la que genera el simulador.

Salir de la aplicación, que se utiliza para desactivar el controlador del Leap Motion, apagar el sistema de sonido y cerrar la ventada del programa.

El tercer dispositivo es el ratón, que se utiliza para interaccionar con todos los botones con los que se compone la aplicación, permitiendo acceder o cerrar ventanas y utilizar las mismas funciones que las del teclado. Por otra parte también permite ajustar la posición de la cámara de la escena gráfica, que se genera en la ventana del simulador. Éste tipo de funcionalidad permite al usuario observar los objetos de la escena en distintos ángulos.

4.3 Diseño del sistema de audio

El sistema de audio del programa se ejecuta mediante la creación de un *buffer* de memoria, donde se le aplica el número de canales de audio que se van a utilizar en la reproducción o el tipo de sonido que se empleará por parte del programa. Este tipo de aplicaciones o modificaciones sobre el sistema, se van ejecutando al mismo tiempo que las funciones empleadas por parte de otros módulos de la aplicación, de modo que estaríamos hablado de un proceso paralelo al resto del programa. Los únicos momentos en los que se deben sincronizar el sistema de sonido con otro modulo son

cuando se debe recibir información para modificar algunos parámetros del sonido, aunque no requiere la programación de un sistema de control de turnos para la modificación de variables globales, por parte de diferentes procesos de otros objetos de la aplicación.

En la generación del sonido, se emplea un sistema de procesamiento de señal digital, también conocido por las siglas DSP, que permite al motor FMOD la implementación de filtros personalizados o la creación de señales de audio complicadas para obtener sonidos de alta calidad.

Referente a su uso en este proyecto, se utilizará para poder generar un tono por defecto y luego se le va cambiando su nivel de volumen y frecuencia de tono. También se añade la posibilidad de cambiar la forma de la onda que utiliza un sonido creado de esta manera, teniendo como onda estándar la sinusoidal, que es la más simple para generar sonidos. A partir de ella se puede cambiar a otros tipos de ondas, que a diferencia de la sinusoidal producen varias formas de distorsión en el sonido resultante. Las que se permiten usar en estas librerías de audio son la onda cuadrática, triangular, y con forma de dientes de sierra.

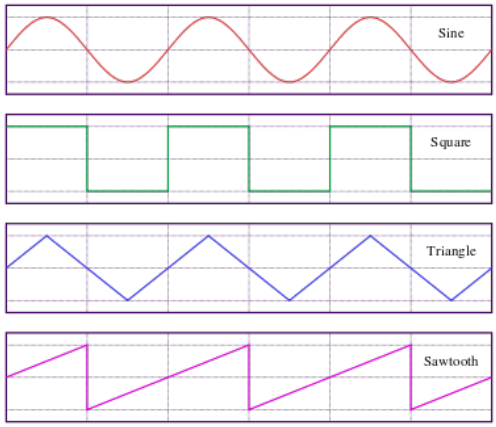

Ilustración 8: Ondas de sonido sinusoidal, cuadrática, triangular y dientes de sierra

Para la modificación de la tonalidad del audio que se genera, FMOD emplea un método de *pitching* a partir de un valor decimal entre 0 y 1, aunque esto no nos permite obtener la frecuencia del tono del sonido resultante, para informar a otros objetos del programa. Aun así, existe otro modo en FMOD para cambiar la tonalidad de un sonido y saber a qué frecuencia hace referencia, y se trata de cambiar solo la frecuencia de muestreo del *buffer* donde se encuentra la pista de audio.

La frecuencia de muestreo es la cantidad de veces que se toma una muestra de audio por unidad de tiempo. Según su valor, el sistema de audio puede reproducir diferentes tamaños de información de una pista de sonido en una unidad de tiempo, de modo que partiendo de una pista de audio, si bajamos el valor de la frecuencia de muestreo,

el sistema reproduce menos información de sonido por unidad de tiempo, alargando la duración de la pista y bajando la tonalidad del sonido. En el caso contrario, el sistema reproduce mucha más información por unidad de tiempo, provocando la reducción de la duración y el aumento del tono de la pista de audio.

En la siguiente ilustración, obtenida partiendo con el programa de posproducción y edición de audio Audacity[7], se puede ver un ejemplo de cómo cambia la duración de un sonido modificando su frecuencia de muestreo. El valor de esa frecuencia en cada una de las pistas de audio se muestran subrayadas en rojo.

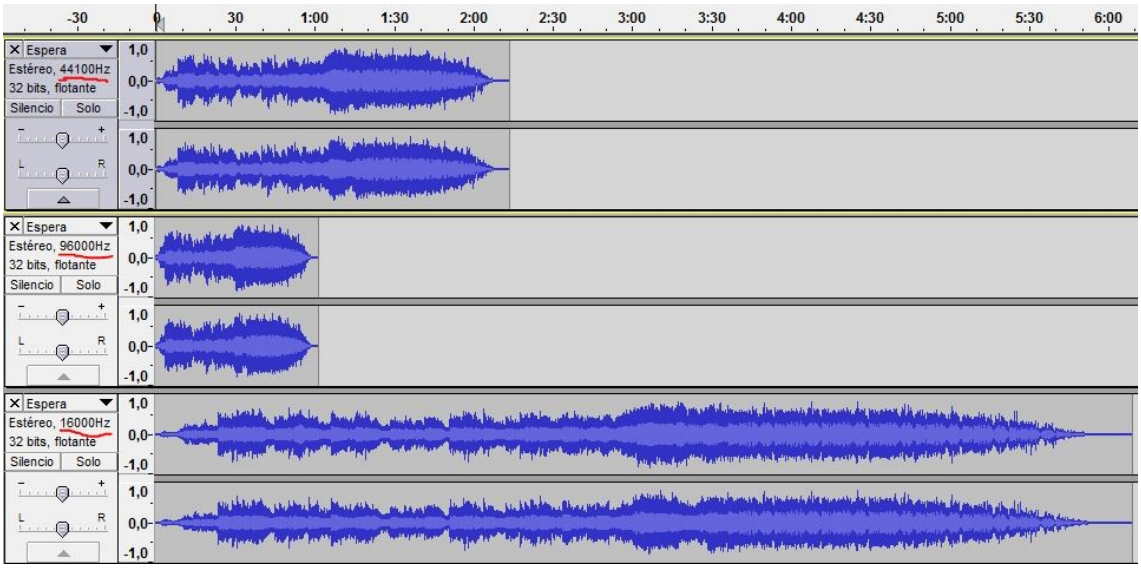

Ilustración 9: Pistas de audio idénticas, pero con frecuencias de muestreo distintas

En nuestro sistema, sabiendo el valor de la frecuencia de muestreo y el tono que tienen al principio de la ejecución, se calcula la relación que hay entre estos dos, de modo que si cambiamos la frecuencia de muestreo del sonido a otro valor, se puede obtener a qué tonalidad se esta generando, partiendo del producto de la nueva frecuencia de muestreo por la relación calculada al principio.

Por otro lado también se ha diseñado dentro del sistema de sonido una función que permite modificar la amplitud de la onda del sonido en distintas formas durante un periodo de tiempo, partiendo del volumen del audio. Esto permite a un sonido DSP el uso de un sistema de envoltura ADSR.

El sistema de ADSR consiste en modular aspectos del sonido, aunque a menudo siempre se modifica su volumen [21]. Normalmente se emplea en sintetizadores para poder imitar a otros instrumentos en su forma de generar un sonido, de modo que partiendo de una onda digital de audio y un buen empleo del sistema de envoltura se puede obtener un timbre parecido al de un instrumento acústico o eléctrico real. Las

7 AUDACITY TEAM. *Audacity.* <http://web.audacityteam.org/>

siglas ADSR consisten en las fases por las que pasa un sonido que es modificado de esta forma, que son:

Attack o ataque, donde se especifica en cuanto tiempo requiere un sonido en alcanzar el máximo nivel de volumen empezando des del silencio. En un instrumento musical, el nivel de volumen máximo depende de la forma con la que se toca el instrumento, alcanzando un nivel de volumen alto si se ejecuta una nota con fuerza, o obteniendo un volumen bajo si se toca con suavidad.

Decay o decaimiento, que consiste en el periodo de tiempo que tarda un sonido en reducir su volumen a un nivel estable sin cambios.

Sustain o sostenimiento, donde se representa el tiempo en el que se mantiene el volumen del sonido obtenido en la fase anterior. En los instrumentos musicales consiste en mantener una nota musical durante un periodo de tiempo, manteniendo un dedo sobre una cuerda en el caso de una guitarra o siguiendo soplando aire en el caso de una flauta. A partir de que empieza a descender el volumen, se pasa a la última fase del sistema.

Release o relajación, que consiste en el tiempo que tarda un sonido en reducir su nivel de volumen des de la fase anterior hasta el silencio del audio. Normalmente, en un instrumento se le aplica esta fase cuando dejamos de mantener una nota musical, de modo que su duración en esta fase suele ser corta a no ser que se produzca mucha resonancia en el sonido por parte de muchos aspectos, como el ambiente en el que suena o los materiales del mismo instrumento musical.

A continuación se muestra un esquema de cómo varia el volumen de la señal de un sonido en cada una de las fases ADSR mencionadas anteriormente.

Ilustración 10: Sistema de envoltura ADSR aplicada sobre un sonido

Su uso en la aplicación no tiene mucha utilidad, porque la señal de audio que genera un Theremin mantiene siempre el mismo nivel de volumen durante las fases de la envoltura ADSR, independientemente de su control de volumen por parte del usuario. Aun así, este sistema se puede emplear para añadir nuevas funcionalidades al *kernel*

de videojuegos creado por la facultad de informática, como objetivo paralelo mencionado al principio de la memoria de trabajo.

4.4 Aspecto gráfico

La aplicación, como ya se ha explicado al principio del apartado de diseño de la solución, consiste en un conjunto de ventanas, donde el usuario accede primero a la ventana de inicio, que contiene una imagen, representando la portada del programa, y un conjunto de botones para acceder al resto de ventanas de la aplicación y cerrar la propia ventana junto a la aplicación.

Entre las otras ventanas del programa se encuentra la de los créditos, que muestra un texto con la información del creador y desarrollador de cada una de las partes que se han creado en la aplicación. También contiene un botón para cerrar la ventana y poder volver a aquella ventana, donde el usuario anteriormente ha pedido mostrar la ventana de créditos.

Entre las ventanas de la aplicación también está la de configuración del programa, que permite al usuario modificar algunas propiedades sobre el funcionamiento del simulador. Estas propiedades son el límite del nivel máximo del volumen al que puede alcanzar el Theremin simulado, y la octava musical máxima a la que puede llegar a alcanzar el instrumento. Esta última propiedad sólo hace referencia al rango de frecuencias de sonido que pueda generar el simulador sin llegar a tener contacto físico con el Theremin, pudiendo generar siempre las notas más graves.

Otra característica de esta ventana es que contiene una pareja de botones, que permiten al usuario cerrar la ventana aplicando los cambios elegidos en el funcionamiento del programa o cerrar la ventana sin modificar ninguna propiedad de la aplicación.

Finalmente queda por mencionar la última y más importante ventana del sistema, la del simulador del Theremin, que contiene casi todo el funcionamiento principal del programa y enlaza con el sistema de entrada por parte del Leap Motion, y el sistema de salida de audio y video.

Esta ventana contiene un acceso a funciones de librerías gráficas, permitiendo mostrar cualquier diseño gráfico que emplea OpenGL. Las funciones gráficas se dividen en tres fases:

Inicialización, que ejecuta todas las funciones y declaraciones necesarias nada más abrir la ventana. Entre ellas está la activación del sistema de audio y del Leap Motion,

la iniciación de algunas variables u objetos, y la activación de algunas funciones gráficas.

Dibujo, que se genera en bucle hasta que se cierra la ventana con funciones gráficas. Esta fase se utiliza para dibujar la escena con sus contenidos gráficos y para ejecutar otras funciones que actualizan propiedades, tanto de la escena que se dibuja en ese instante como de otras partes del programa. En el caso de la aplicación, a parte de dibujar el Theremin, las manos que se reconocen y un teclado de piano, actualiza la frecuencia de sonido y el nivel de volumen que debe generar el sistema de sonido, a parte de que debe informar a ése sistema de qué rango de frecuencias puede reproducir.

Redimensión, que contiene aquellas funciones que se deben emplear en el programa cuando se cambia las dimensiones de la ventana. Normalmente se usa para adaptar las propiedades de la cámara de la escena dibujada para que no se produzcan problemas de visualización de los objetos gráficos.

Uso de ratón, donde se encuentran aquellos métodos que se activan cuando se utiliza el ratón como dispositivo de entrada. En la aplicación se utiliza para cambiar el punto de observación de los objetos dibujados.

Por otra parte, la ventana del simulador contiene una barra de menú, con la capacidad de generar botones que permiten al usuario activar o parar el sonido del programa, acceder a las ventanas de créditos y configuración de la aplicación, cambiar el tipo de onda del sonido y cerrar la aplicación entre muchas otras. Todas estas funcionalidades se pueden acceder mediante el uso del teclado como sistema de entrada, como se había explicado en el apartado de sistema de entrada, permitiendo al programa el uso de comandos rápidos.

También contiene una barra de tareas, donde se encuentran las acciones típicas por parte de este tipo de programas, adornadas cada una con una imagen para ayudar al usuario a comprender la funcionalidad de ésa acción.

5 Implementación

5.1 El controlador del Leap Motion

La clase que permite controlar el funcionamiento del Leap Motion tiene como objetivo obtener información de la posición de las manos del usuario y enviársela a aquel módulo de la aplicación que contiene el uso de librerías gráficas para dibujarlas. Como también se pretende añadir el uso del controlador ya implementado por el UPVGameKernel, se tiene que modificar este controlador para que pueda ejecutar las funciones necesarias que requiere la aplicación, porque el *kernel,* que dispone ya de un módulo para el uso del Leap Motion, no los tiene implementados por defecto.

Aprovechando el controlador que se tiene por defecto del UPVGameKernel, se obtienen las funciones necesarias para crear un objeto referente al Leap Motion, junto al de inicializar y finalizar el uso del dispositivo por parte de cualquier aplicación que lo quiera usar. Para ello, se crea un objeto del tipo *controller*, que contiene toda la información que puede generar el dispositivo. Para poder obtener esa información, se crea también un objeto del tipo *Listener*, que se conecta con el *controller* para poder filtrar toda la información que se genera en la que sea necesaria para el programa.

Partiendo del objeto *Listener*, se crea una clase que hereda sus métodos para poder tratar con esa información, donde se crean las siguientes variables para poder almacenar todo lo referente al posicionamiento trigonométrico y forma de las manos:

UGKhandsGL, que consiste en un *array* de vectores, donde se almacenan las posiciones de los puntos de flexión de los huesos que compone cada mano.

UGKpalma, del tipo *array,* donde se almacena en vectores la posición de la palma de cada una de las manos.

UGKdireccionManos, que se encarga de guardar en una lista de matrices el angulo con el que está girado cada una de las palmas en referencia al dispositivo.

UGKdireccionHuesos, que contiene la misma funcionalidad que la variable anterior, con la diferencia de que se emplea para los huesos de cada mano.

UGKcentroHuesos, que consiste en una lista de vectores, donde se guarda la posición central de cada hueso.

UGKlongitudHuesos, también del tipo *array*, con la funcionalidad de almacenar el valor de la longitud de cada hueso de la mano.

UGKanchuraHuesos, que se utiliza para guardar en una lista de valores reales la anchura de cada hueso.

UGKnumManos, donde se almacena el número de mano que haya reconocido el Leap Motion en un *frame*.

A parte de éstas variables, también se encuentran aquellas que son necesarias que almacenan los datos necesarios para el sistema de sonido. Entre ellas se encuentran las que calculan el valor de la nueva frecuencia de tono y nivel del volumen, que sirven para informar al sistema de audio que debe actualizar las propiedades de su canal de audio. También están las que limitan el nivel máximo de volumen y el número de octavas de notas con las que se pueden reproducir en el sistema de sonido, que son accesibles por parte del usuario mediante una ventana de la aplicación.

Dentro de esta clase, ya creada por parte del UPVGameKernel, se utilizan métodos que se activan partiendo de los estados por los que pasa el controlador. Algunos de ellos son los que se activan cuando el dispositivo se conecta a la aplicación o se desconecta durante su uso, aunque no tienen ninguna utilidad en la implementación de la aplicación.

El método que si que hace falta es la de la función *onFrame*, que se encarga de generar la información necesaria a partir de lo que reconoce el Leap Motion en un *frame* para enviársela al resto de módulos de la aplicación. Para su implementación referente a la obtención de los huesos de las manos, se ha aprovechado código de otro proyecto, que sirve para guiar a principiantes en lo referente a dibujar manos a partir del Leap Motion y que se encuentra en dominio público[8].

La función *onFrame*, empieza calculando cuantas manos se han reconocido y lo guarda en la variable *UGKnumManos*. Para cada una de estas manos hace lo siguiente:

Guarda la posición de la palma de la mano en *UGKpalma* y la dirección a la que apunta la mano en *UGKdireccionManos.*

Comprueba si esa palma se sitúa dentro de la zona referente al de la ejecución del control de volumen del Theremin, que desde un principio se debe saber por parte del desarrollador del programa. En el caso de que esté dentro, se compara la posición Y de la palma con un valor ya guardado anteriormente por parte de otras manos detectadas. Si el que se ha calculado recientemente tiene un valor inferior,

8 GITHUB. *Magnetic Mesh* <https://github.com/leapmotion-examples/MagneticMesh>

se guarda su valor, y en el caso contrario no lo hace, manteniendo siempre el valor más bajo.

Después se analiza los dedos de las manos y para cada uno de esos dedos, los huesos que lo componen.

Para cada hueso, se almacena su posición central en *UGKcentroHuesos* y su punta superior en *UGKhandsGL* de la misma forma que en la posición de la palma. También se almacenan la longitud de cada hueso en *UGKlongitudHuesos*.

En la obtención de información de los huesos, también se comprueba para cada uno de ellos y con la guardada anteriormente por otra mano, la distancia a la que están de la posición donde debe estar la antena del Theremin que genera el tono del sonido, almacenando al final aquella que sea más corta.

Una vez hecho este procedimiento, si no se han ocupado del todo los *arrays* donde se guarda la información obtenida anteriormente, se rellenan con valores que no afecten en la funcionalidad de la aplicación.

Finalmente, partiendo de los valores obtenidos sobre las distancias de las manos a las antenas del Theremin, la función *onFrame* termina obteniendo el valor de la frecuencia y nivel de volumen del sonido que se debe generar en ese momento, partiendo de funciones que relacionan la distancia de las manos con el valor de cada uno de estos parámetros.

Referente a la función de la frecuencia de audio, hay que tener en cuenta que la relación entre la frecuencia tonal de una nota musical y otra no sigue un patrón lineal. Como la diferencia de frecuencia de sonido entre una nota y otra no es constante, se ha usado una función que obtiene la frecuencia de una nota a partir de la frecuencia del tono de la nota más baja que puede percibir el oído humano, la relación entre una nota cualquiera y la siguiente, y la distancia en semitonos entre la nota más baja que se usa como referencia y aquella que se quiere obtener.

Frecuencia de nota X = Frecuencia de nota A0 * [(2^(1/12)) ^ (Distancia en semitonos entre A0 y X)]

En la siguiente imagen se muestra la gráfica que representa la frecuencia de audio a partir de la nota musical que se encuentra a cierta distancia de la nota más grave.

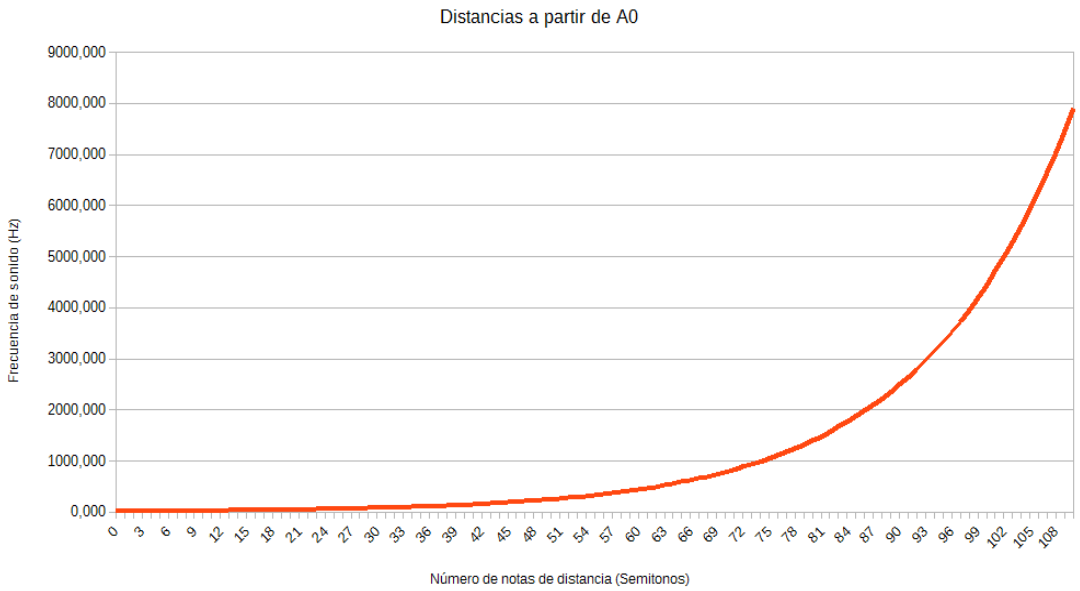

Ilustración 11: Gráfica con las distancias entre la nota A0 y cualquier otra

Como se puede ver, no se trata de una función lineal, pero que se puede aplicar al programa, adaptando el número de notas de distancia con el intervalo de longitud que hay en la escena gráfica entre la posición de la antena del Theremin y el punto más alejado de la antena que permite todavía generar un sonido.

En cuanto a la función que calcula el volumen del sonido, consiste en una función lineal donde el volumen se obtiene calculando la diferencia entre la distancia entre una mano y la antena del volumen dividida por la distancia máxima con la que se permite modificar el valor del volumen.

En el siguiente esquema se puede observar un breve resumen del nuevo funcionamiento que tiene la función *onFrame*:

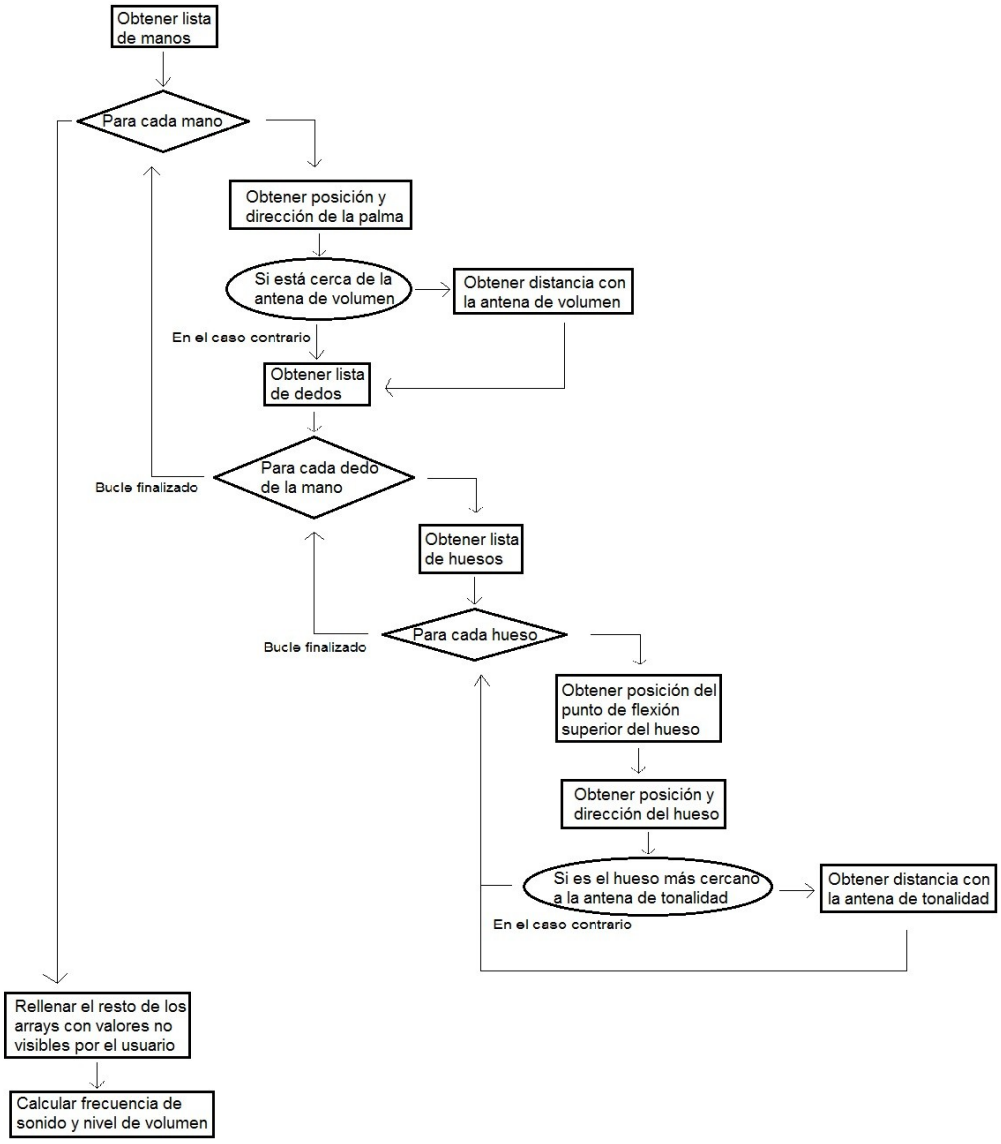

Ilustración 12: Funcionamiento del método *onFrame*

Con los valores obtenidos en la función *onFrame*, esta clase los puede enviar a los módulos que se lo pidan partiendo de los métodos de obtención de información de su constructor. Esos métodos devuelven el valor de cualquier variable ya mencionada anteriormente partiendo de una posición dentro de su *array.* También existen las funciones que modifican el valor de las variable que limitan el volumen y las frecuencias de tono del sonido.

Como en la implementación de la obtención de las partes de las manos se utilizan funciones que no soporta la librería de programación del Leap Motion que contiene el UPVGameKernel, se ha tenido que crear un fichero a parte que controla el tipo de bibliotecas que se deben usar, entre las del *kernel* y las que se utilizan en el programa. Con ello, en el caso de que se usen las librerías de versiones anteriores, se debe

avisar al compilador de que no compile el código referente al que se acaba de mencionar.

5.2 La clase audio

La clase que contiene el funcionamiento del sistema de sonido del programa está integrado en el *kernel* creado por el departamento de informática, para añadir más funcionalidades que podrán ser reutilizadas por otros desarrolladores.

Como en el módulo de sistemas de audio del UPVGameKernel no existe ningún uso de las librerías empleadas en éste proyecto, ha hecho falta crear un nuevo sistema de sonido basado en las librerías de FMOD Studio. Como consecuencia, se ha tenido que aprovechar la clase ya definida por el *kernel* para crear un sistema de sonido, separando las funciones ya creadas de las que se han añadido de nuevas. Esto se debe hacer porque la aplicación sólo contiene métodos de programación relacionados con el uso de sonido DSP, mientras que las funciones que emplea el UPVGameKernel sirven para reproducir y controlar archivos de audio, que no se utilizan en este caso.

Con las nuevas modificaciones en el *kernel*, cuando se vaya a utilizar las librerías de FMOD Studio, se crearán los siguientes tipos de variables globales para que el sistema de audio pueda emplear los métodos necesarios con el objetivo de cubrir los objetivos del proyecto:

System, que consiste en un puntero al objeto que representa al sistema de FMOD. Ésta variable es necesaria para poder crear un sistema de sonido.

Channel, que sirve para crear un número cualquiera de canales de audio, siempre que FMOD Studio lo pueda soportar. También se utiliza para añadirle propiedades referente al sonido que pueda reproducir, como activar o detener la reproducción de un sonido, o controlar el nivel de volumen del canal de audio entre otros.

DSP, que se trata de un puntero al objeto de sonido DSP, y se utiliza para añadir al sistema un sonido de este tipo. Partiendo de esta variable, se le puede modificar la propiedad de tipo de onda que genera, para poder cambiar de onda sin tener que crear distintos sonidos DSP.

FMOD_Result, que se utiliza para referenciar todas las funciones que se van generando durante el uso del sistema de audio, para después devolver un código de respuesta por parte del sistema. Éste código que devuelve sirve para avisar al desarrollador o usuario dónde se ha producido un fallo en el sistema, en el caso de que algo no funciona bien y se bloquea el programa. En el caso contrario devuelve un mensaje informando de que no se ha encontrado ningún problema.

DSPfrequency, que se emplea para modificar la frecuencia de muestreo del sonido DSP que genera un canal de audio del sistema.

DSPvolumen, que sirve para modificar el nivel de volumen del canal de audio del sistema. En la aplicación solo modifica el canal que genera un sonido del tipo DSP.

Para poder inicializar el sistema de sonido del programa, esta clase requiere de un método de iniciación, que crea el objeto referente al sistema y asigna en un principio el número de canales de audio que dispone a la variable del tipo *channel*. Después inicializa el uso de FMOD Studio, asignando el máximo número de canales que puede disponer el sistema y creando un oscilador DSP con una frecuencia de sonido por defecto, referenciado a la variable *DSP*.

Una vez se ha arrancado el sistema de sonido, el programa ya puede pedirle al sistema de audio que reproduzca el sonido DSP, modifique el volumen de salida, cambiar la forma de la onda del oscilador DSP o detener el sonido.

Para que el sistema reproduzca un sonido, se emplea un método que se activa en el caso de que se quiera cambiar el tipo de onda del sonido DSP en ése momento. Para ello, primero detiene el uso del canal de audio, si ya estaba generando un sonido. Luego activa el uso del canal de audio, añadiendo al objeto *channel* las propiedades de la variable *DSP* para que el sistema de audio reproduzca un sonido DSP con un tipo de oscilador de onda, elegida anteriormente por el usuario.

Luego está el método que hace lo contrario al anterior, detener la reproducción del sonido del sistema. Para ello, este método desactiva el uso del canal que esté utilizando el sistema de audio, deteniendo su uso y informando al sistema por medio del objeto apuntado con la variable *channel* de que no tiene ningún canal de audio en uso.

En el caso de cambiar la frecuencia de tono del sonido que se reproduce, existe un método que, partiendo de una nueva frecuencia de muestreo, guarda en una variable la frecuencia del canal que emite el sonido DSP, modifica dicha variable asignándole la nueva frecuencia, y luego vuelve a asignar al objeto *channel*.

Para modificar el volumen de un canal de audio se siguen los mismos pasos que el del método de cambiar la frecuencia del sonido, pero en vez de cambiar la frecuencia se cambia el nivel de volumen.

Como se había mencionado en el apartado de diseño de la solución, también se ha insertado el sistema ADSR dentro de un método, que se activa cuando se empieza a reproducir un sonido. Ésta función, partiendo del volumen inicial, el nivel de volumen

que se quiere al terminar las fases de *attack* y *decay*, y el tiempo de duración de cada una de sus fases, genera una rampa ascendente o decreciente de volumen desde el principio hasta el final de la duración temporal de una fase. En el caso de la fase *sustain* no se genera ninguna rampa de volumen, porque dicha fase consiste en mantener durante un tiempo el volumen correspondiente al obtenido al final de la fase anterior, que es el de *decay.* Como ya se había mencionado antes, el sistema ADSR en el funcionamiento del Theremin no tiene mucho sentido emplearlo, porque el nivel de volumen que genera el instrumento es siempre constante, aunque puede servir para mejorar el UPVGameKernel y ser aprovechado en otras aplicaciones de audio.

Cuando se cierra el sistema, libera de la memoria principal del ordenador el uso del sonido DSP y del sistema de audio, y luego cierra el uso del sistema de audio.

A continuación se muestra un esquema donde se muestra el ciclo de vida del sistema de sonido y cuándo se puede ejecutar cada una de sus funciones según van pidiendo otros objetos del programa.

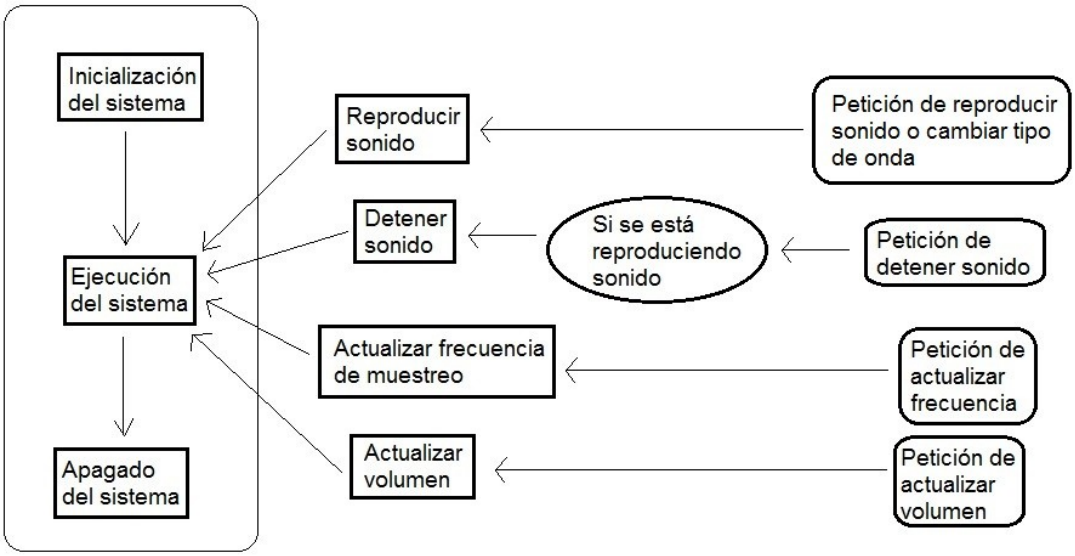

Ilustración 13: Ciclo de vida del sistema de audio

5.3 El módulo gráfico

El módulo gráfico de la aplicación es la que permite visualizar la forma en que se comporta los dos módulos mencionados anteriormente y es donde se concentra casi todo el código implementado del programa. Para la explicación del sistema gráfico, este módulo se puede dividir en tres partes: Las ventanas de la aplicación, la creación de la escena gráfica en una de ellas y la implementación del importador de modelos ya creados.

5.3.1 Implementación de ventanas

La creación de cada una de las ventanas puede resultar muy tedioso en el caso de que se implementasen desde cero, pero con el uso de las librerías de Qt se pueden crear partiendo de una base como interfaz gráfica. Los tipos de ventanas que se utilizan en el programa son los siguientes:

Main Window, que consiste en una ventana que viene con las barras superiores de menú y tareas, y una barra inferior de estado por defecto. Todos éstos contenedores de objetos viene al principio vacíos, para que el desarrollador las pueda personalizar. También contiene las típicas funciones para maximizar o minimizar el tamaño de la ventana.

Dialog, que a diferencia de la otra ventana, viene por defecto sin nada y tampoco tiene funciones rápidas para esconder o reajustar al máximo la ventana. Normalmente se utiliza para mostrar notificaciones o usar como ventana de opciones ajustables.

Además, en cada una de estas ventanas se le pueden añadir objetos utilizables por el usuario. Los que se ven en la interfaz de la aplicación que se desarrolla son:

Push Button, que se trata del típico botón de ventana que activa funciones a partir de la acción de pulsarlo.

Group Box, con la función de contenedor de objetos, para agrupar otros objetos de interfaz gráfica dentro de sus límites.

Slider, que consiste en un botón deslizable sobre una linea vertical u horizontal, calculando un valor numérico partiendo de sus límites delimitados por el desarrollador y su posición.

Widget, también con la función de contenedor, sirve para delimitar el espacio de uso de cualquier ventana. Es indispensable para poder crear una ventana, incluso si está vacía.

Label, que se trata del objeto con el que el desarrollador puede insertar textos dentro de la interfaz gráfica. También se utiliza delimitar el tamaño de las imágenes que se le pueden añadir.

Action, que es un botón al que se le asigna una función, seguido de una imagen o tecla rápida de acceso como complementos

Icon, donde se puede almacenar un grupo de imágenes, utilizables por otros objetos de interfaz gráfica.

Menu, que sirve de contenedor de objetos *Action.*

Menu Bar, que también consiste en un contenedor para situar objetos del tipo *Menu* a lo largo de una barra superior de la ventana.

Status Bar, que tiene la misma función que el Menu Bar, sólo que permite la inserción de botones. Normalmente el desarrollador adorna esos botones con un texto o imagen, para que se pueda entender mejor su función por parte del usuario.

Spacer, que sirve para poner una distancia de separación entre dos objetos de forma vertical u horizontal.

Layout, con la función de ordenar con un patrón los objetos que lo envuelve en una linea horizontal, vertical o en forma de malla.

Para la creación de la cada una de las ventanas gráficas, se utiliza la herramienta Qt Design, que codifica los diseños creados en archivos capaces de compilar en el entorno Visual Studio, de modo que en el proyecto se pueden acceder a su uso empleando clases, donde crean un objeto con las propiedades de la ventana diseñada.

En el apartado del diseño de las solución se había mencionado el contenido de cada una de las ventanas de la aplicación, de modo que en este apartado se va a explicar los métodos que contienen cada una de ellas.

La forma en la que se permite la utilización de funciones de programación en los objetos de la interfaz gráfica consiste en utilizar métodos de programación denominados *slots*. Los *slots* son funciones que se pueden activar partiendo de una reacción del objeto gráfico al que le asigna. Un ejemplo puede ser la reacción de un objecto del tipo *Push Button* al ser clicado con el ratón, de modo que genera una señal al *slot* asignado, ejecutando las funciones de su método.

La ventana de inicio contiene un *slot* en cada uno de sus *Push Button*. La del botón “Salir” ejecuta el método que cierra la ventana, cerrando también el programa.

En el caso del botón “Configuración”, crea y muestra la ventana de configuración del programa, formado por dos *Push Button* y un *Group Box* con *sliders* y *labels,* con la función de almacenar en variables los valores modificados en los *sliders* de la ventana, si el usuario ha pedido cambiar propiedades. Esto es necesario hacerlo en esta ventana porque la ventana de configuración modifica valores utilizados en el controlador de entrada de la ventana del simulador, que todavía no se ha creado, de

modo que no se le puede asignar nuevos cambios y se deben guardar ésos valores cambiados en variables a parte.

El botón “Acerca de” abre la ventada de información del programa, compuesto por *labels* dispersos en un *Group Box* y un *Push Button* para cerrar la ventana.

La funcionalidad del botón “Empezar” crea la ventana del simulador del instrumento, de modo que en el caso de que se hayan hecho cambios en la configuración de la aplicación, se utilizan las variables de almacenamiento, mencionadas anteriormente, para asignar un nuevo valor a las propiedades del controlador del Leap Motion. Después muestra la ventana maximizada del Theremin y cierra la ventana de inicio.

La ventana del simulador consiste en un *Main Window* y tiene la función principal de utilizarse para mostrar la escena gráfica, que se encuentra en una clase del programa a parte. Para que esta ventana pueda mostrar la escena, se debe modificar las propiedades del *widget* que utiliza la ventana por defecto para que pueda referenciar la clase de la escena, de modo que crea y muestra la escena, a parte de que inicializa el uso del sistema de audio y del Leap Motion, en el instante que se crea el *Main Window*.

Una vez referenciado la clase con funciones gráficas y de sonido a la ventana del simulador, se puede crear los *slots* necesarios para acceder a las propiedades del sistema gráfico, sonoro y de entrada. Para ello, solo se necesita una variable global entera, que se encarga de almacenar el último tipo de onda del sonido que ha generado el programa. El por qué de necesitar esta variable es porque el resto de funciones sólo necesitan acceder a parámetros de otras clases sin ninguna condición externa.

Los métodos que modifican dicha variable son las que activan el sonido de la aplicación, llamando al método de reproducir sonido con la propiedad de tipo de onda que se quiera, situado en el objeto que se encarga del audio del programa, creado dentro del objeto que controla la escena gráfica del simulador. Una vez hecha ésa llamada, se almacena en dicha variable un número, que representa el tipo de onda que genera el sistema de sonido.

Partiendo de éstos métodos, se crea el *slot* para activar el audio. Éste método contiene una implementación en *switch* que depende de la variable entera mencionada anteriormente, de modo que si se ha generado anteriormente un sonido, al volver a activar el audio del programa, la aplicación pedirá al sistema de sonido el tipo de onda que se haya reproducido por última vez, llamando al método que activa sonido. En el caso de que se haya activado por primera vez, se informará al sistema de audio de que genere un sonido con la onda del tipo sinusoidal.

El *slot* que permite detener la reproducción del audio consiste en una llamada al método de detener el audio que contiene el sistema de sonido.

A parte de los métodos que se encargan del sonido, se encuentran también en la ventana del simulador aquellas que modifican propiedades de la escena gráfica.

El método que permite mostrar un teclado de piano para guiar al usuario consta de un acceso y modificación de una variable, situada en el objeto que se encarga del renderizado de la escena, para que se pueda dibujar en la ventana. En el caso de que ya se mostraba por pantalla, inhabilita su visualización.

El otro método consiste en activar el “Modo Afinado” de la aplicación. Para ello, se accede también a una variable del objeto que dibuja la escena, para que se puedan aplicar los cambios necesarios tanto en la escena gráfica como en el sistema de audio. Si ya se encontraba activado, pedirá al generador de la escena que vuelva a tipo de afinación inicial.

Por último, en esta ventana se hallan las funciones de acceder a la ventana de configuración del programa, acceder a la de los créditos de la aplicación y de cerrar el programa. Sus implementaciones son idénticas a las del la ventana de inicio, con la excepción de que en el uso de la ventana de configuración no hace falta variables de almacenamiento, porque sus cambios se pueden efectuar directamente al sistema de entrada, que ya está creado. También existen diferencias en la función de cerrar el programa, de modo que a parte de cerrar la ventana, desactiva el uso del Leap Motion y del sistema de sonido, que se habían activado en la inicialización de la ventana del simulador.

Con los métodos *slot* implementados, solo falta referenciarlos como acciones en las barras de menú y tareas. Para ello, se crean tres objetos contenedores del tipo *Menu* de la barra de menú. En cada uno de estos contenedores se añaden botones del tipo *Action*, junto al *slot* que deben activar, su tecla de acceso rápido correspondiente y una imagen del tipo *Icon* que lo haga mas comprensible para el usuario.

Para el caso de la barra de tareas, sólo se añade aquellos botones que necesite acceder el usuario para un uso simple de la aplicación, que son los de activar o detener el sonido y cambiar a uno de los cuatro tipos de onda de sonido.

Las ventanas de configuración y créditos del programa sólo tienen la función de cerrar la ventana mediante el uso de un *Push Button*.

5.3.2 Escena gráfica

La escena donde se muestran los objetos diseñados con funciones OpenGL, se crea a partir de una clase, que al mismo tiempo sirve para crear un controlador del Leap Motion y el sistema de sonido. En esta clase se necesitan las siguientes variables:

CSound, que se trata de un objeto con el sistema de audio del programa. Con él se puede acceder a los métodos de programación implementados para el control del audio.

UseLeap, que consiste en un objeto formado por la clase que se encarga del controlador del Leap Motion. Es necesario para acceder a la información referente a la posición donde se deben crear las manos en la escena.

Modelo, que es un objeto de la clase Modelo, con las funciones necesarias para importar un diseño gráfico a partir de un archivo. Sirve para importar el modelo del Theremin creado con Blender.

EstadoLeap, que se utiliza para almacenar, con un número entero, si el Leap Motion se ha inicializado o cerrado en la aplicación.

MAfinado, que permite al programa ejecutar o no las funciones referentes a los de la utilización del “Modo Afinado”.

MPiano, que se usa con el mismo objetivo que *MAfinado*, con la diferencia de habilitar o deshabilitar el renderizado del teclado de piano.

LastPos, que almacena la última posición del ratón en base a un sistema de coordenadas situado en el plano de la pantalla.

PosCamara, que consiste en un *array* con la posición de la cámara de la escena. Se utiliza para situar la cámara en diferentes sitios en la escena gráfica.

Texto, que se trata de una matriz de letras, donde se almacenan los caracteres y números que se deben mostrar en pantalla. Cada fila de la matriz consiste en una información distinta.

Frecuencias, que contiene una lista de frecuencias de sonido en orden desde la nota musical más grave hasta una nota aguda, situada en la décima octava musical. La lista está hecha en base a la escala musical estándar [22].

Notas, que es una lista de notas musicales en orden desde la más grabe hasta un Do (C en nomenclatura estándar) en la novena octava.

Relación, que consiste en un valor decimal con la relación entre la frecuencia de muestreo del programa y la nota musical con la que se inicializa el sonido DSP del sistema de sonido en hercios.

Nota, que consiste en un puntero a una posición de las listas *Frecuencias* y *Notas*, para obtener su contenido.

El funcionamiento de esta clase se basa en métodos heredados de las librerías de OpenGL, de modo que primero se ejecuta la fase de inicialización seguida de la de dibujo, que se ejecuta en bucle hasta que se cierra el programa. También existen la fase de redimensión de la ventana de la escena gráfica y la del uso del ratón como dispositivo de entrada, que se activan antes de la fase de dibujo en el caso de que se utilice el ratón o se cambia las dimensiones de la ventana.

En la fase de inicialización, se activan el uso del dispositivo Leap Motion, el sistema de audio y el importador de modelos. También se activa el uso de luces en el escenario, permitiendo la aparición de sombras, y la declaración inicial de las variables mencionadas anteriormente.

Durante la fase de dibujo de la escena, se empieza actualizando las propiedades del sistema de audio y algunas variables de esta clase. Para ello, se obtiene la frecuencia de sonido calculada por el objeto *UseLeap* en ese momento. Luego se modifica la frecuencia de sonido que genera el objeto *CSound* dependiendo del valor de *MAfinado*.

Si es negativo, se modifica la frecuencia de sonido de *CSound y* se obtiene la nota musical relativa a la frecuencia generada. Para obtener la nota musical a partir de la frecuencia de tono, se utiliza un método de búsqueda donde partiendo de la variable *Frecuencias*, se obtiene la posición de la frecuencia de sonido más próxima a la que se ha calculado, para almacenarla en la variable *Nota* y utilizarla para encontrar la nota musical más próxima en la variable *Notas.*

En el caso de que *MAfinado* contenga un valor positivo, se calcula y almacena en *Nota* la posición de la frecuencia de sonido más próxima a la que se ha calculado. Luego, mediante el valor guardado en *Nota,* se le asigna al sistema de sonido una frecuencia de la lista *Frecuencias* y se obtiene la nota musical de la lista *Notas.*

A continuación se le asigna a CSound el nivel de volumen calculado por *UseLeap*, seguido de la actualización de la matriz *Texto*, cambiando los valores de la frecuencia, nota musical, volumen y si se encuentra activo o no el “Modo Afinado”

Una vez finalizada la actualización de diversas propiedades del sistema, se posiciona la cámara del escenario, apuntando hacia donde se debe dibujar el Theremin. Luego se llama a un método del objeto *Modelo* para que dibuje el instrumento en la escena, seguido por la creación de las manos reconocidas en ése instante por el controlador del Leap Motion.

Para poder dibujar las manos en el escenario gráfico, primero se debe comprobar si se ha detectado alguna mano en ése momento, comprobando el contenido de *UGKnumManos* del objeto *UseLeap*, de modo que no se hace nada en el caso de que no se haya rastreado nada. Si por el contrario de han detectado manos, se renderiza cada una de las manos detectadas, accediendo a aquellas funciones del objeto *UseLeap* que devuelven la posición de cada parte de la mano, según se van pidiendo. El dibujo de una mano empieza con la creación de la palma, dibujando una esfera plana y un anillo, trasladándolos a su posición y girándola en su dirección respecto al contenido de *UGKpalma* y *UGKdireccionManos*. Después se crean diversas esferas, que representan el punto de unión de los diversos huesos y se posicionan según lo indica la variable *UGKhandsGL*. Finalmente se dibujan los huesos de la mano, creando cilindros planos de la longitud obtenida en *UGKlongitudHuesos* y de un radio equivalente al de *UGKanchuraHuesos*, posicionados y girados en una dirección partiendo de las variables *UGKcentroHuesos* y *UGKdireccionHuesos.*

Después del renderizado de las manos, se crea en la escena el texto de información del programa al lado superior de la ventana, donde se muestra el contenido de la matriz *Texto*.

Finalmente se dibuja en la parte inferior de la pantalla, en el caso de que lo quiera el usuario, un teclado de piano, destacando aquella tecla que representa el sonido generado por el Theremin, utilizando la variable *Nota*.

La fase de redimensión sólo emplea una función que actualiza la perspectiva de la cámara de la escena respecto al ancho y alto de la ventana del simulador.

También existe una función que se activa cuando se usa el ratón. En ésta función se calcula la diferencia entre la posición anterior y actual del ratón, para emplear la diferencia de la posición horizontal en el cálculo de la la rotación de la cámara alrededor del Theremin en el eje X y la diferencia de altura en el cálculo de la rotación sobre el eje Y. Para hacerlo, se transforma las diferencias de posición del ratón en grados y luego se actualiza los valores de la variable *PosCamara*, calculando la nueva posición de la cámara según su ángulo.

A continuación se puede observar un resumen del ciclo de vida de la escena gráfica, representando cada una de sus fases, junto a las funciones que se ejecutan en cada una de ellas.

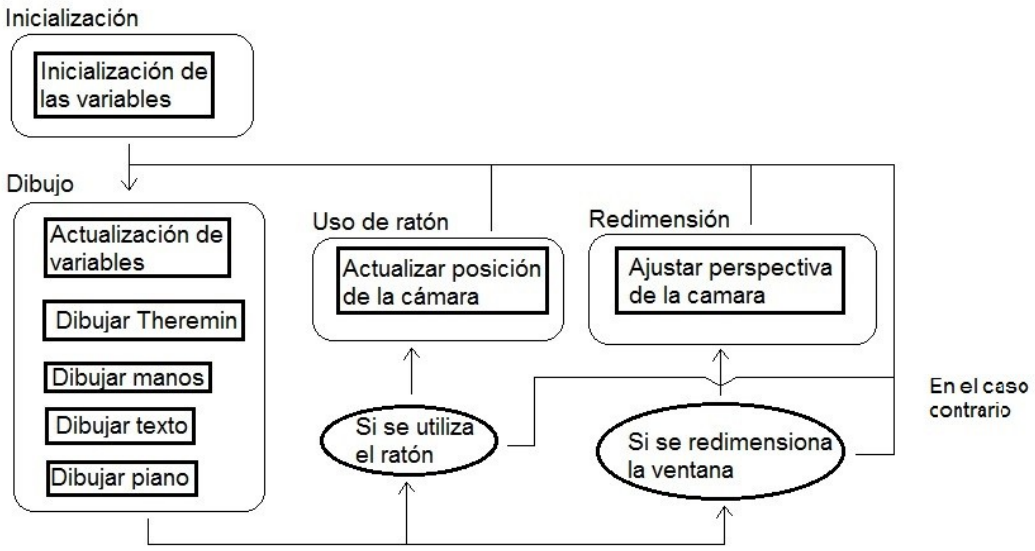

Ilustración 14: Ciclo de vida de la escena gráfica

5.3.3 La clase modelo

En la escena gráfica se utiliza un modelo diseñado anteriormente con Blender, de modo que hace falta un sistema de importación de modelos al programa para que se muestren en pantalla.

El modelo creado consiste en un diseño simple del Theremin con sus dos antenas de control de volumen y tonalidad. Se tiene que tener en cuenta que las antenas deben de estar situadas en las posiciones que se emplean en la clase del controlador del Leap Motion para calcular los valores de volumen y frecuencia, porque el usuario se guía en el uso del instrumento a partir del modelo gráfico. En el caso de que no coincidan estas posiciones, el usuario puede llegar a entender que la aplicación no funciona como debe ser.

Para importar el diseño del instrumento, se crea una clase con acceso a las librerías de Assimp y a las funciones de OpenGL necesarias para posicionar los objetos importados en la escena, ya que la clase que crea la escena ya se encarga de insertar el resto de funcionalidades gráficas.

Esta clase se ha implementado a partir de un ejemplo hecho por los mismos creadores de la librerías de Assimp, que permite mediante funciones de OpenGL cargar y mostrar

modelos, con superficies coloreadas y sin texturas, en una escena por defecto[9], aunque está implementado en el lenguaje de programación C. Tras adaptarla a C++ y cambiar algunos métodos ya definidos, se ha conseguido que sólo importase un diseño gráfico a una escena ya creada.

Para la utilización de esta clase, se requiere el uso de la variable global *scene*, del tipo *aiScene*, que permite almacenar una escena gráfica diseñada a partir de un archivo creado por un diseñador tridimensional de modelos.

Al comienzo de su uso, se requiere ejecutar su método de inicialización, que consiste en acceder al archivo donde se encuentra la escena gráfica que se quiera importar y guardarla en la variable *scene*.

A partir de entonces, se le puede llamar para que añada a una escena lo que hay almacenado en *scene*. Para ello, se renderizan las superficies que se componen cada uno de sus modelos y se le aplica los materiales que contienen, que consiste en colorear cada superficie según la información contenida en cada uno de los modelos.

En el caso de que se cierra el programa, se utiliza un método que borra el contenido guardado en *scene*, porque las librerías de Assimp pueden almacenar temporalmente su contenido y puede provocar fallos en importaciones de escenas futuras.

9 GITHUB. *SimpleOpenGL*. <https://github.com/assimp/assimp/tree/master/samples/SimpleOpenGL>

6 Conclusiones

6.1 Resultados

El resultado obtenido al final de la aplicación ha sido bastante satisfactoria. Se ha podido alcanzar todos los objetivos, con la posibilidad de proponerse otros de nuevos a partir de estos. Se han realizado pruebas en un ordenador a parte y se ha observado que el programa realiza todas sus funciones sin problemas de uso con el procesador y la memoria principal.

Desde mi punto de vista, he aprendido a utilizar una gran variedad de herramientas de programación utilizadas durante la implementación que no se han visto en la carrera, como el FMOD, Leap Motion o Qt, y también a manejar mejor la librería OpenGL a nivel de desarrollo de código. También he aprendido más sobre el uso del lenguaje de programación C++ y el entorno de programación Visual Studio.

Por otro lado me he dado cuenta de que es importante una buena planificación inicial, sobretodo en la estructuración del código, para resolver los problemas de inserción de funcionalidades nuevas creadas por otros programas.

6.2 Dificultades encontradas

La mayoría de dificultades se han encontrado en la búsqueda de las herramientas necesarias para el proyecto y aprender cómo se utilizan, a parte de los problemas que generaban al intentar insertarlas en el entorno de trabajo del proyecto. En el caso de la herramienta Qt, que permite diseñar interfaces gráficas en un programa, daba problemas de compatibilidad con el entorno de programación Visual Studio, y para resolverlo se ha tenido que utilizar una versión anterior a la actual de Qt para que no diera problemas.

Otro problema que ha surgido en el proyecto ha sido la de añadir a la aplicación el sistema de sonido, después de haber implementado parte de la estructura para dibujar la escena del simulador. La solución ha consistido en cambiar la estructura del código por una basada en objetos, donde se representan en cada uno de ellos una parte empleada en el programa. Esto se podía haber evitado si se hubiese planificado des del principio la utilización de este tipo de estructura.

6.3 Relación con la carrera

El haber cursado algunas asignaturas de la carrera ha sido importante para llegar a obtener los resultados mencionados anteriormente, sobretodo aquellas que se centran

en el diseño de la estructura, las interfaces y el uso de OpenGL para la creación de un programa. La asignatura de *Interfaces Persona Computador* ha ayudado a conocer estilos y consejos en el diseño de una interfaz gráfica. Las de *Ingeniería del Software* y *Introducción a la Programación de Videojuegos* me han enseñado a crear una estructura en el programa que permita cambar su funcionalidad y reutilizar código para otros programas que empleen las mismos métodos de programación. En el caso de la creación de un escenario gráfico, la asignatura *Introducción a los Sistemas Gráficos Interactivos* ha resultado útil, porque permite conocer el funcionamiento de las librerías de OpenGL y de cómo añadir propiedades a la escena para hacerla más realista.

A parte de estas asignaturas, también han sido importante aquellas que me han enseñado a programar, tanto en el aprendizaje del lenguaje de programación C++ como en la enseñanza de técnicas óptimas que mejoran la eficiencia de un programa.

6.4 Trabajo futuro

Aunque se hayan podido lograr los objetivos del proyecto, este programa puede llegar a contener muchas más funcionalidades respecto a su sistema de audio y la interfaz gráfica.

En el caso del audio del programa, se encuentra la opción de utilizar *plugins VST*, para reproducir sonidos semejantes a los que producen en la realidad algunos instrumentos musicales. También se podría crear un *ecualizador* de audio con funciones de cambiar el nivel de sonoridad de señales graves y agudas, de modo que requiere la implantación de un controlador del *ecualizador* en la interfaz gráfica de la aplicación.

Sobre el sistema gráfico, se puede mejorar la parte de las sombras de cada objeto para demostrar mejor la existencia de un espacio tridimensional en el simulador del Theremin, de modo que proyecten una sombra sobre la superficie de otro objeto.

Como se encuentra implementado un importador de modelos tridimensionales en el programa, se puede crear un diseño de las manos que aparecen en el simulador, además de que pueda mover las articulaciones dependiendo de los gestos reconocidos con el Leap Motion. También se puede implementar una herramienta que permita al usuario seleccionar un diseño del Theremin y de manos, entre un abanico de modelos ya creados, para que se muestren por la pantalla.

Referente al sistema de entrada, existe la posibilidad de cambiarlo por uno que sólo necesite el uso del reconocedor de gestos de las manos. Para ello, se tendría que cambiar también la interfaz gráfica a una que ejecute funciones partiendo de la

posición de las manos y los gestos que genera, o otra que emplee colisiones entre las manos y otros objetos gráficos para activar funcionalidades del programa.

En cuanto al programa en general, está la opción de hacerlo multiplataforma, para poder ejecutarlo también en sistemas operativos de Apple y Linux.

Si se llega a mejorar el programa, se podría pensar en la opción de ponerlo en el mercado y añadir contenidos de expansión que puedan ser monetizados, como la posibilidad de insertar más modelos del instrumento, diseños de manos o *plugins VST*.

7 Manual de usuario

7.1 Instalación

Para instalar la aplicación existen dos formas:

Descargar el ejecutable junto a las carpetas de “Modelos” y “Resources”, y guardarlo en un lugar accesible del ordenador. En el caso de lo quiera el usuario, crear un enlace directo al ejecutable y situarlo en el escritorio.

Descargar el proyecto completo y abrirlo con el Visual Studio Professional o Ultimate de la versión 11.0.5 o superior. Compilar el proyecto y crear un enlace directo en el escritorio del ordenador, partiendo del ejecutable obtenido en la compilación. En el caso de encontrar problemas de enlaces durante la compilación, se debe modificar las propiedades de las librerías que se utilizan. La lista de librerías que se usan se puede encontrar en el archivo “README” del proyecto.

Tanto los archivos de su ejecución como el proyecto entero se encuentran en un repositorio de GitHub[10].

7.2 Configuración

Los dispositivos necesarios para aprovechar al máximo las funciones del programa son los siguientes:

Leap Motion.

Ratón.

Teclado.

La configuración del Leap Motion se puede modificar accediendo a la barra de tareas del escritorio, pulsando el botón derecho y luego al botón “Configuración”. También se puede acceder a su configuración abriendo la aplicación de la tienda del Leap Motion y seleccionar dentro de la pestaña “Archivo” de la barra de herramientas la opción “Configuración del controlador”. Dentro del panel de control se puede modificar la altura en la que se permite el uso del dispositivo, mejorar el rastreo de manos con poca iluminación o re-calibrar el dispositivo, dependiendo de los gustos del usuario.

En el caso del teclado o el ratón, existe la posibilidad de que sus propiedades se puedan modificar partiendo de una aplicación instalada por parte del dispositivo. En el

10 GITHUB. *Theremin Digital*. <https://github.com/joasanm/Qt-Theremin.git>

caso general, se puede acceder a su configuración abriendo el panel de control y luego seleccionar la opción de “Ratón” o “Teclado” según lo que se quiera configurar.

7.3 Ejecución

Una vez instalado el ejecutable y configurado sus dispositivos, abrir el ejecutable. Lo siguiente que aparecerá será la ventana de inicio de la aplicación. En ella se encuentra un grupo de botones.

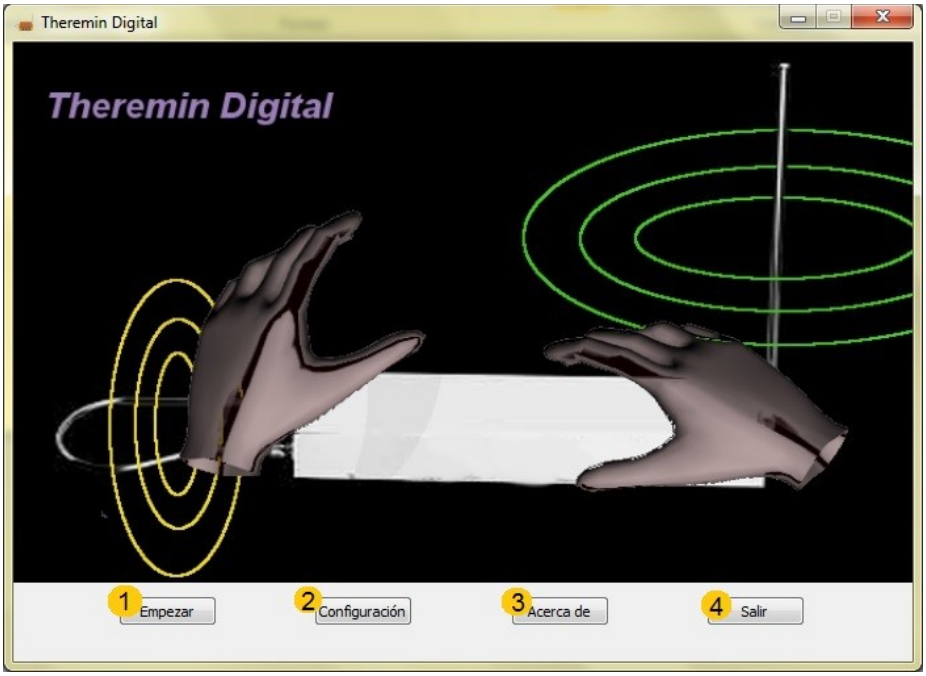

Ilustración 15: Ventana de inicio del programa

(1) El botón “Empezar”, que permite acceder a la ventana principal del programa, donde se ejecuta el simulador digital del Theremin.

(2) El botón “Configuración”, que abre una ventana donde se puede configurar el simulador del Theremin.

(3) El botón “Acerca de”, que muestra al usuario una ventana con información del producto.

(4) El botón “Salir” se utiliza para cerrar la aplicación.

La ventana del simulador se compone de lo siguiente:

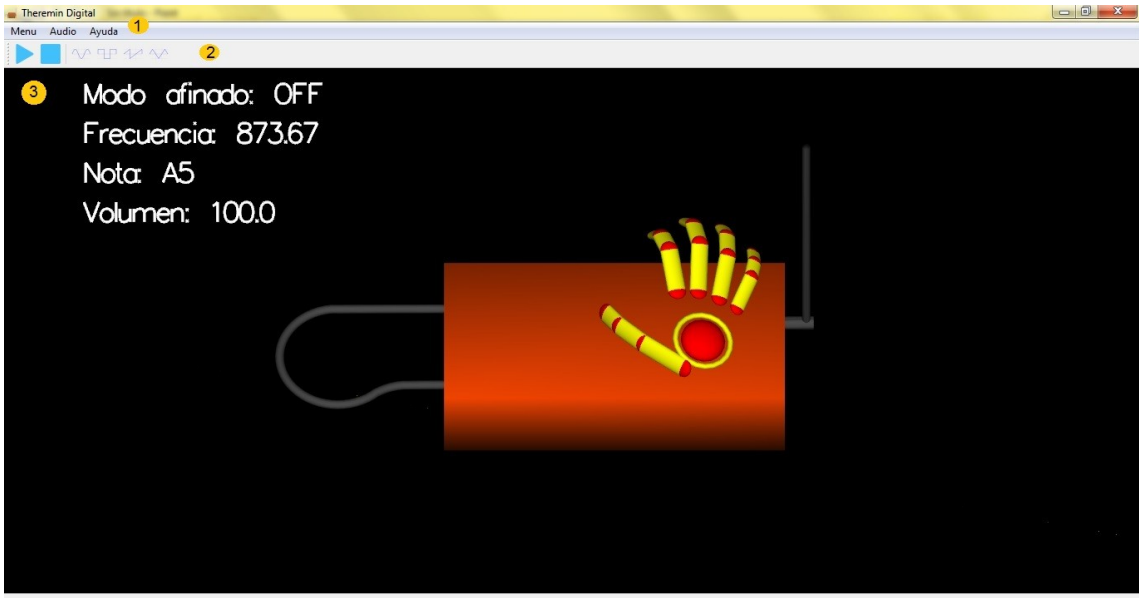

Ilustración 16: Ventana del simulador del Theremin

(1) Barra de menú, compuesta por tres desplegables:

Menú, que contiene las acciones de activar y desactivar el sonido, mostrar o ocultar un teclado de piano de referencia, habilitar o inhabilitar el “Modo Afinado”, acceder a la configuración del simulador y salir del programa.

Audio, con acciones modificadoras de la onda de sonido. La aplicación permite la selección de onda sinusoidal, cuadrada, triangular y dientes de sierra.

Ayuda, que permite acceder a la ventana de créditos de la aplicación.

(2) Barra de herramientas, que se compone de las acciones más típicas del programa. Éstas acciones son activar sonido, apagar sonido y cambiar el tipo de onda del audio que se genera.

(3) Ventana gráfica, donde se muestra el Theremin, las manos que reconoce el Leap Motion y la información del sonido que se está generando. La información del sonido muestra si está o no activado el “Modo Afinado”, la frecuencia de sonido que se está generando en ése momento, la nota musical más próxima a la frecuencia generada y el porcentaje de volumen al que se está emitiendo el sonido.

Con el uso del ratón, se permite cambiar el punto de vista con el que se puede ver el instrumento y las manos, pulsando el botón derecho sobre la ventana y arrastrándolo hacia cualquier dirección.

En el caso del teclado, el programa contiene un conjunto de teclas rápidas con las que se pueden activar las acciones existentes en las barras de menú y tareas. En la siguiente tabla se encuentran las teclas rápidas y sus funciones.

Tecla	Función
Barra espaciadora	Activar sonido
M	Desactivar sonido
A	Habilitar/Deshabilitar Modo afinado
H	Mostrar/Ocultar teclado
Esc	Cerrar aplicación
S	Activar onda sinusoidal
Q	Activar onda cuadrada
D	Activar onda dientes de sierra
T	Activar onda triangular

Tabla 3: Tabla de teclas rápidas

La función de “Modo Afinado” consiste en generar por parte del usuario aquella frecuencias de audio que representan notas musicales exactas.

El teclado que se puede mostrar en la ventana permite al usuario qué nota musical está generando en el caso de que se pudiera aplicar en un piano.

En la ventana de configuración, el programa permite modificar, mediante el uso de *sliders*, el número de octavas que puede llegar a generar el instrumento y el nivel máximo de volumen que se puede alcanzar. Se tiene que tener en cuenta que el Theremin genera un sonido muy agudo cuando hay contacto físico con la antena que controla la frecuencia del sonido, de modo que el simulador genera en ése caso una nota equivalente a un Do (C en nomenclatura estándar) situado una octava musical por encima al número de octavas permitidas por el usuario.

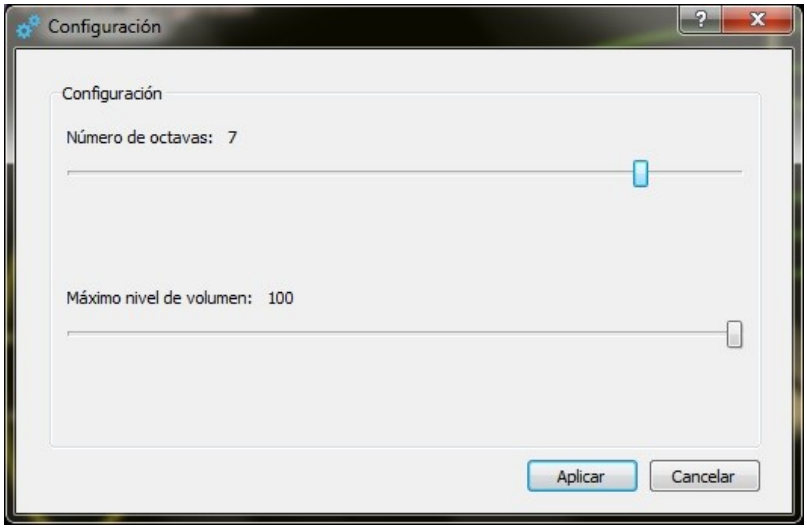

Ilustración 17: Ventana de configuración

Una vez decidida la configuración, se pulsa al botón “Aplicar” para hacer cambios en las propiedades del programa, o el botón “Cancelar” para no aplicar ningún cambio.

8 Bibliografía

[1] WIKIMEDIA FOUNDATION INC.. *Theremin*. <https://en.wikipedia.org/wiki/Theremin> [Consulta:5 de julio de 2015]

[2] WIKIMEDIA FOUNDATION INC.. *Reconocimiento de gestos*. <https://es.wikipedia.org/wiki/Reconocimiento_de_gestos> [Consulta:5 de julio de 2015]

[3] MICROSOFT. *Kinect for Windows*. <http://www.microsoft.com/en-us/kinectforwindows/> [Consulta:5 de julio de 2015]

[4] INTEL CORPORATION. *RealSense Technology*. <http://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html> [Consulta:5 de julio de 2015]

[5] LEAP MOTION INC.. *Leap Motion*. <https://www.leapmotion.com/> [Consulta:14 de febrero de 2015]

[6] LOGBAR INC.. *Ring*. <http://logbar.jp/ring/en> [Consulta:5 de julio de 2015]

[7] LEAP MOTION INC.. *Leap Motion Developer Portal*. <https://developer.leapmotion.com/> [Consulta:1 de marzo de 2015]

[8] ALLEGRO DEVELOPERS. *Allegro*. <http://alleg.sourceforge.net/> [Consulta:8 de julio de 2015]

[9] CREATIVE TECHNOLOGY. *OpenAL Soft*. <http://www.openal-soft.org/> [Consulta:8 de julio de 2015]

[10] KRONOS GROUP. *OpenML*. <https://www.khronos.org/openml/> [Consulta:8 de julio de 2015]

[11] LAURENT GOMILA. *Simple and Fast Multimedia Library*. <http://www.sfml-dev.org/> [Consulta:4 de abril de 2015]

[12] SAM LANTINGA. *Simple DirectMedia Layer Homepage*. <https://www.libsdl.org/> [Consulta:8 de julio de 2015]

[13] FIRELIGHTS TECHNOLOGIES PTY LTD. *Fmod*. <http://www.fmod.org/> [Consulta:1 de mayo de 2015]

[14] MICROSOFT. *DirectX*. <https://www.microsoft.com/windoes/directx> [Consulta:7 de julio de 2015]

[15] KRONOS GROUP. *OpenGL*. <https://www.opengl.org/> [Consulta:7 de julio de 2015]

[16] QT COMPANY. *Qt*. <http://www.qt.io/> [Consulta:31 de julio de 2015]

[17] ASSIMP DEVELOPMENT TEAM. *Assimp*. <http://assimp.sourceforge.net/> [Consulta:6 de julio de 2015]

[18] BLENDER. *Blender*. <https://www.blender.org/features/> [Consulta:5 de agosto de 2015]

[19] AUTODESK. *3Ds Max*. <http://www.autodesk.com/products/3ds-max/overview> [Consulta:5 de agosto de 2015]

[20] AUTODESK. *Maya*. <http://www.autodesk.es/products/maya/overview> [Consulta:5 de agosto de 2015]

[21] WIKIAUDIO. *ADSR envelope*. <http://en.wikiaudio.org/ADSR_envelope> [Consulta:25 de julio de 2015]

[22] MICHIGAN TECHNOLOGICAL UNIVERSITY. *Physics of Music*. <http://www.phy.mtu.edu/~suits/Physicsofmusic.html> [Consulta:7 de julio de 2015]

9 Glosario

ADSR: Término utilizado en el entorno musical para definir la evolución temporal de amplitud de un sonido.

API: *Application Programming Interface*. Conjunto de subrutinas, funciones y procedimientos que ofrece una biblioteca de programación para ser utilizado por otro software como una capa de abstracción.

Array: Tipo de variable de almacenamiento, que permite guardar una serie de elementos de un mismo tipo. Desde un punto de vista lógico, se puede visualizar como un conjunto de elementos ordenados en fila.

Bluetooth: Especificación industrial para Redes Inalámbricas de Área Personal, que posibilita la transmisión de información de cualquier formato mediante un enlace de radiofrecuencias.

Buffer: Espacio de memoria dinámica donde se almacena una pista de audio y se le aplican modificaciones para después reproducirlo a través de un altavoz.

Controller: Clase del Leap Motion definida como la interfaz principal del dispositivo. La información que genera la envía a los objetos *Listeners* que lo componen, ejecutando sus funciones de forma distribuida.

Ecualizador: Dispositivo que modifica el volumen en frecuencias de la señal que genera una pista de audio. Permite variar de forma independiente la intensidad de los tonos básicos.

Frame: Función del Leap Motion que contiene información de todo lo que el dispositivo ha reconocido. Un programa que emplea el Leap Motion genera un *frame* distinto en cada momento.

Frecuencia de muestreo: Número de muestras por unidad de tiempo que se toman de una señal continua para producir una señal discreta, con el objetivo de convertir una señal analógica en digital.

Hardware: Partes físicas de un sistema informático, donde sus componentes son eléctricos, electromecánicos o mecánicos.

Kernel: *Software* que constituye una parte fundamental del sistema operativo. Es el encargada de gestionar recursos a través de servicios de llamada al sistema para dar a los programas un acceso seguro al *hardware* del ordenador.

Listener: Clase del Leap Motion donde se definen las acciones que se deben ejecutar a partir del estado del objeto *Controller*.

MIDI: *Musical Instrument Digital Interface*. Estándar tecnológico que describe un protocolo, interfaz digital y conectores que permiten que varios instrumentos musicales electrónicos, computadoras y otros dispositivos relacionados se conecten y comuniquen entre si.

Overclocking: Término que consiste en alcanzar una mayor velocidad de reloj y mejor rendimiento para un componente electrónico por encima de las especificaciones del fabricante.

Pitching: Función de las librerías de audio de FMOD, que permite cambiar la frecuencia del tono de un sonido dependiendo del grado que se le asigna para aumentar o disminuir dicha frecuencia.

Plugin: Complemento de una aplicación que se relaciona con otra para añadir una nueva función específica al programa.

RAM: *Random Access Memory.* Componente que se utiliza como memoria de trabajo de un computador para el sistema operativo, los programas y la mayor parte del *software.*

Sample: Muestra de sonido grabado en cualquier tipo de soporte para reutilizarlo posteriormente en un instrumento musical. Puede estar transformado mediante efectos sonoros.

Slider: Componente gráfico que consiste en un botón deslizable sobre una barra.

Slot: Método empleado en Qt para crear una comunicación entre un componente gráfico y las funciones de un programa.

Smartphone: Teléfono móvil construido sobre una plataforma informática móvil con una mayor capacidad de almacenamiento de datos, que permite realizar procesos semejantes al de un a microcomputadora comparado con el teléfono convencional.

Socket: Concepto abstracto donde dos programas, situados en computadoras diferentes, pueden intercambiar cualquier flujo de datos de manera fiable y ordenada.

Software: Soporte lógico de un sistema informático que comprende el conjunto de componentes que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos.

Switch: Función de los lenguajes de programación para escoger una opción entre muchas a partir del contenido de una variable o un valor dado.

VST: *Virtual Studio Technology*. Interfaz estándar desarrollada para conectar sintetizadores de audio y *plugins* de efectos a editores de audio y sistemas de grabación. Permite reemplazar el *hardware* tradicional de grabación por un estudio virtual con herramientas *software.*