UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Sistema de realidad aumentada para simular instrumento de percusión

TESIS

Que para obtener el título de

Ingeniero Mecatrónico

P R E S E N T A

Mauricio Morales Rodríguez

DIRECTOR DE TESIS

M.A. Luis Yair Bautista Blanco

Ciudad Universitaria, Cd. Mx., 2017

Mateo 22:36-40

Mateo 5-7

AGRADECIMIENTOS

A Dios por que soy, existo, y vivo. Por hacer de mí lo que el día de hoy soy, lo que puedo hacer, y lo que es mi entorno. Éste trabajo y todo yo es gracias a Él.

A mi amá, Rosa María Alejandrina, por llevarme por el buen camino, siempre estar conmigo y para mí, y porque ha dado y sigue dándolo todo por mí. Mi más grande inspiración. Gracias a ella soy lo que soy hoy. Ejemplo de perseverancia, esfuerzo, trabajo duro, y de humanidad, entre muchas otras cosas.

A Jose, mi segunda madre, por apoyarme siempre y siempre estar para mi amá y para mí.

Éste trabajo es de los 4.

A toda mi familia que me ha apoyado y siempre ha estado ahí en las locuras.

Y a todos los amigos y personas importantes que han estado presentes en mi camino, imposible de mencionar a todos, pero a todos los llevo en mi corazón: Patti, Michelle, 怡璇 , しょうたろう, Ramiro, 재갑, Mimi, Mariana, Danya, Óscar, Amaury, Ramsés, Tahiri, Izaak, Isaac, Bruno, Chris, Dra. Cristina, Moisés, Memo, etc.

INDICE

LISTA DE FIGURAS

LISTA DE TABLAS

INTRODUCCION

La generación de música requiere de contar con los elementos necesarios para poder expresar y materializar de forma sonora lo que se tenga concebido en la cabeza.

En éste trabajo se muestra la generación de un sistema funcional que pretende emular un instrumento de percusión para brindar de una nueva herramienta en la generación de ritmos que, además, solucione problemas presentados en los usuarios de estos instrumentos.

En el Capítulo 1 se narran los aspectos y vivencias que llevaron a generar la idea de éste trabajo.

El Capítulo 2 es una continuación del Capítulo 1 al profundizar en las razones de querer materializar la idea.

En el Capítulo 3 se desglosa la idea para identificar los elementos que la componen y así saber que es necesario en la elaboración del trabajo.

En el Capítulo 4 se muestran los pasos a seguir para la elaboración del trabajo, junto con el objetivo del mismo generado con base en los capítulos anteriores.

El Capítulo 5 es un inicio en la obtención de los elementos necesarios para la elaboración del proyecto, al buscar y mencionar distintas tecnologías que pueden obtener dichos elementos, y conocer productos que busquen satisfacer el mismo objetivo presentado en el capítulo anterior.

En el Capítulo 6 se busca profundizar y ampliar la idea generada al buscar información de usuarios de instrumentos de percusión, y con base en dicha información, definir necesidades, requerimientos, y especificaciones para generar un concepto, y, posteriormente, generar una solución para la elaboración del trabajo, cuyos elementos a usar son definidos en el Capítulo 7, y cuya elaboración se muestra en el Capítulo 8.

En el Capítulo 9 se realizan pruebas al sistema elaborado, y se compara con las especificaciones mencionadas en el Capítulo 6, para identificar que elementos del trabajo deben ser mejorados y con que fin, mencionados en el Capítulo 10.

Finalmente, las conclusiones y comentarios sobre el trabajo son presentados en el Capítulo 11.

1 ANTECEDENTES

La música desde siempre ha sido una de las formas mas bellas del ser humano para expresar sentimientos, dar a conocer ideas o, incluso, poder modificar el comportamiento de una persona. Para el autor de éste trabajo, la música es una de sus mayores pasiones. Ha dedicado gran parte de su vida estudiándola y disfrutándola, de modo que no hay día en el cual no tenga interacción con ella de forma auditiva o práctica, mediante algún instrumento.

Es la pasión por la música lo que da origen, y fundamenta, éste proyecto.

El autor tuvo la oportunidad de realizar un intercambio estudiantil en Taiwán, República de China, por cinco meses. Cinco meses en los cuales hubo un inmenso aprendizaje académico, cultural, espiritual y personal. Un aprendizaje que llevó a entender aspectos y definiciones importantes en la vida de todo ser humano.

Durante el intercambio estudiantil hubo un abandono total de la práctica musical. Se siguió escuchando música, desde melodías estudiadas en su totalidad hasta melodías nunca antes escuchadas o imaginadas propias del lugar, pero no se contó con un instrumento con el cual satisfacer la necesidad de interpretar las melodías escuchadas, o la necesidad de crear nuevas melodías o ritmos.

En la universidad donde se realizó el intercambio estudiantil, Universidad Nacional de Taiwán de Ciencia y Tecnología, existe la posibilidad de tomar prestados diversos instrumentos: piano, guitarras, bajo, etc. Esta posibilidad pudo haber satisfecho las necesidades mencionadas en el párrafo anterior, pero, dado el tiempo de traslado entre el lugar de residencia y la escuela, aproximadamente de treinta minutos en bicicleta o metro, la inspiración, idea musical, o impulso de interpretación, se desvanecía.

Esta situación deja al descubierto la necesidad de poder liberar la adrenalina musical generada, en el lugar de residencia. Para poder liberar ésta adrenalina, y al no contar con un instrumento físico, se recurrió a la práctica de tocar una *batería de aire*.

Tocar una batería de aire es imitar que se toca una batería, sin contar con una físicamente. Ésta práctica no satisface la necesidad musical creativa de generar nuevos ritmos, pero satisface la necesidad de “interpretar” una melodía en el lugar en donde se encuentre sin necesidad de trasladarse o de preparar un instrumento. Es una interacción musical kinestésica.

El segundo gran pilar de este proyecto es el desarrollo tecnológico. Fue, también, durante los cinco meses del intercambio estudiantil que HTC, compañía taiwanesa, introdujo al mercado su visor de realidad virtual VIVE.

Visor que posee la capacidad de conocer la ubicación espacial del usuario a través de los sensores que se encuentran en el visor y controles del mismo. Los controles, además de proporcionar la posición de las manos del usuario, permite conocer la orientación de las mismas. Se estudiaron las capacidades del sistema y se contó con la oportunidad de probarlo en un evento de desarrollo tecnológico en Taiwán, Computex.

La capacidad de detección espacial de VIVE y la necesidad del autor de tener una interacción musical en el lugar en donde se encontrara, manifestada en la práctica de tocar una batería de aire, dio a luz la idea de éste trabajo: Un entorno virtual en el cual un músico pueda satisfacer sus necesidades creativas, sin la necesidad de contar con un instrumento físico; en este caso, un instrumento de percusión.

Un elemento fundamental en la generación de música, y en cualquier aspecto humano, es el poder transmitir y dar a entender una idea. La interacción natural con las personas es algo que la realidad virtual, por su naturaleza, no puede ofrecer.

Con base en el párrafo anterior, se añade a la idea generada el uso de realidad aumentada. Con el uso de ésta tecnología, se puede contar con esa interacción humana, tan necesaria, para los procesos creativos musicales de una forma natural, o mas directa, y, además, ofrece una comunicación, de la misma naturaleza, con los espectadores del proceso.

2 JUSTIFICACION

Satisfacer la necesidad musical de interpretar y componer ritmos en un instrumento de percusión, de una forma sencilla y sin necesidad de preparar o transportar un instrumento, además de poder interactuar con las personas que se encuentren colaborando u observando el proceso creativo, es la primera justificación de éste proyecto.

Para la segunda justificación de este proyecto, hay que regresar, una vez más, a Taiwán. *Administración estratégica de alta tecnología* fue una de las materias que se cursaron en el intercambio estudiantil que, además de generar las bases de ésta segunda justificación, ayudó a definir aspectos profundos en la vida del autor.

En esta materia se estudiaron, de forma general, distintos casos y problemáticas de diversas empresas, así como el comportamiento y proyecciones a futuro de diferentes industrias. Dentro de las industrias estudiadas, se encuentra la realidad aumentada. En específico, se estudió el uso de ésta tecnología en la industria automotriz.

La industria automotriz es quien ha invertido mas tiempo y dinero en aplicaciones usando realidad aumentada. BMW es un gran ejemplo. Ha desarrollado aplicaciones que facilitan la reparación, diseño, venta y promoción de sus productos. Por ejemplo, desde el 2007, cuenta con una aplicación diseñada para facilitar el mantenimiento y reparación de carros. Mediante unos lentes, escanea la parte frontal interna del carro y proyecta un modelo tridimensional de las partes que ahí se encuentran. Dependiendo de la pieza a reparar o cambiar, se dan instrucciones al portador de los lentes para realizar la tarea, junto con proyecciones de donde se encuentran las piezas y la forma de cambiarlas de posición. En la Figura 1 se muestra la aplicación.

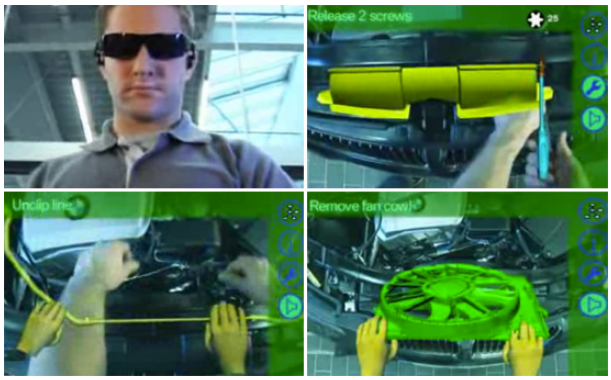

Figura 1 Aplicación de BMW de realidad aumentada para servicio técnico [2]

BMW ha anunciado el lanzamiento de una aplicación para Android, la cual usa una nueva tecnología desarrollada por Google llamada Tango. Con esta aplicación buscan incrementar sus ventas al mostrar dos modelos de sus autos, sin necesidad de contar con ellos físicamente. La aplicación permite una interacción con el modelo virtual, es decir, se puede abrir y cerrar las puertas para ver el interior, entre otras posibilidades. En la figura 2 se muestra la aplicación.

Figura 2 Aplicación de BMW y Google de realidad aumentada para muestra de carros [3]

Realizando un estudio más profundo y enfocado solamente al mercado de la realidad aumentada, se pueden observar movimientos que justifican las proyecciones que se han hecho para la industria en el 2020.

Actualmente existen muchas empresas impulsando la venta de visores de realidad aumentada. Microsoft, Sony, Meta, son ejemplos de este grupo. Además, existen otras empresas que están por entrar al mercado como Apple o Leap Motion. La existencia de una cantidad tan grande de empresas implica una gran competencia. Esta competencia obliga a las empresas, dependiendo de su plan de negocios, a generar mejores sensores, mejores diseños, mejor uso de materiales, mayor velocidad, mejores experiencias, o soluciones más baratas. Cada una de éstas mejoras no solo involucra a la empresa que desean estos cambios, también involucra a otros sectores de donde obtienen las partes de sus productos como electrónica, materiales, manufactura, entre otros. Esto se traduce en una gran actividad económica generada por la venta de hardware.

Al existir un impulso tan grande en la venta de hardware, es natural que haya gran actividad en el desarrollo de software para las distintas plataformas que ofrecen las empresas. El software que se desarrolla se dirige, al igual que el hardware, a distintos sectores como manufactura, entretenimiento, salud, finanzas, mercadotecnia, etc. El desarrollo de software, busca simplificar procesos, mejorar técnicas, ofrecer nuevas herramientas, entre otras cosas, resultando en un deseo de la gente de poder contar con esas herramientas o entretenimiento. Al ser enorme la cantidad de sectores en los que la realidad aumentada tiene aplicación, la actividad económica generada por software, naturalmente, es enorme. Finalmente, al existir nuevas y mejores herramientas generadas por software, se espera que las ventas en hardware incrementen.

Se espera que Asia sea quien lidere el mercado de realidad virtual y aumentada para el 2020. Se han realizado grandes inversiones en China para la manufactura de estos visores; además Corea y Japón son también grandes competidores no sólo en hardware, con Samsung o Sony, también en software. En la Figura 3 se muestran las predicciones regionales del mercado para el 2020. [6]

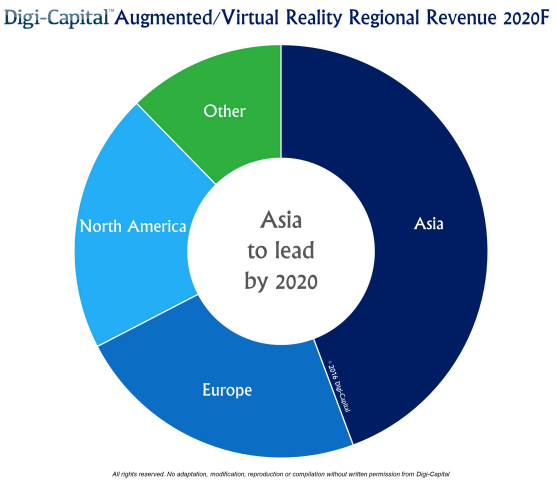

Figura 3 Predicciones regionales para 2020 de distribución del mercado de realidad aumentada [6]

Con base en los párrafos anteriores, se espera que el mercado de realidad aumentada crezca, en promedio, a un 80% de tasa compuesta de crecimiento anual, generando un mercado, para el 2020, de 90 billones de dólares. Además, se espera que sea en Asia y Norteamérica donde se generen mayores ganancias por componente. En la Figura 4 se muestran las predicciones por sector del mercado de realidad aumentada para el 2020. En la Figura 5 se muestran las predicciones de ganancias de 2014 a 2024 por dispositivo en Asia. [6][7]

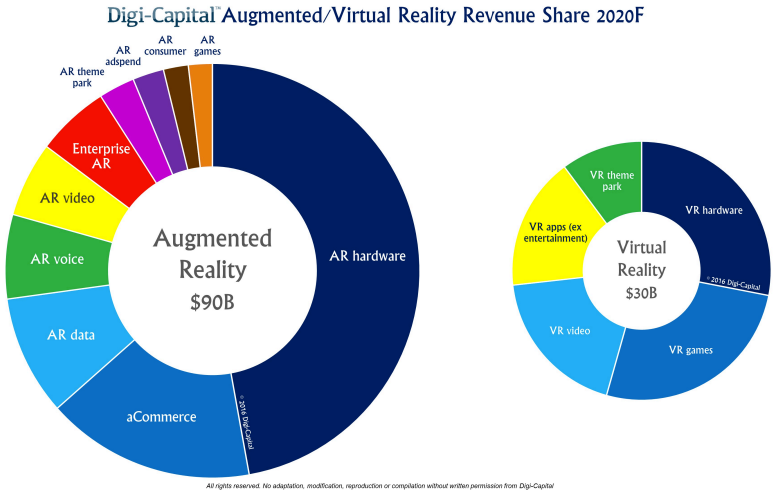

Figura 4 Predicciones por sector del mercado de realidad aumentada para 2020 [6]

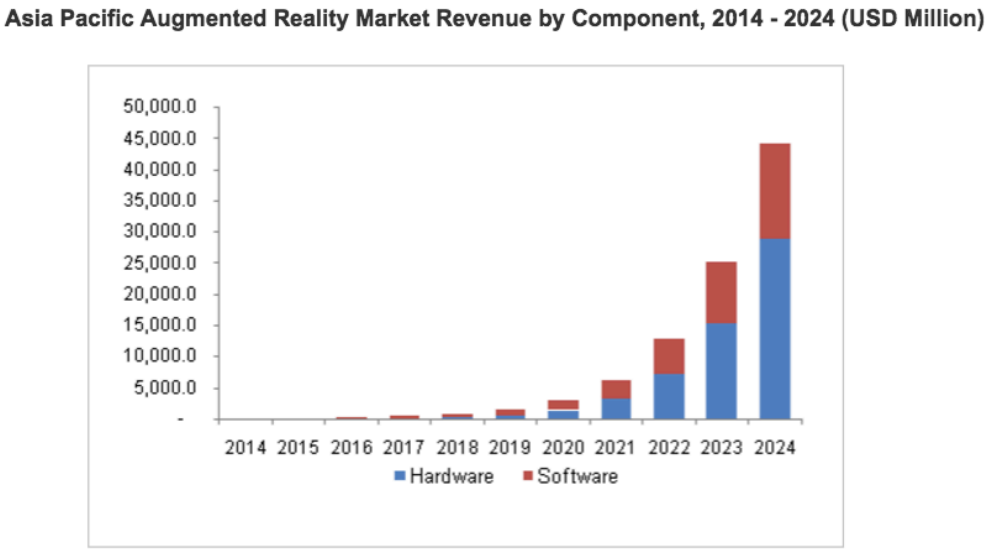

Figura 5 Predicciones de 2014 a 2024 de ganancias por dispositivo en Asia [5]

Aprovechar las oportunidades generadas por esta tecnología, obtener parte de los 90 billones de dólares esperados para el 2020, y aprovechar la escasez de aplicaciones musicales con esta tecnología, conforman la segunda justificación de éste proyecto.

3 DESCRIPCION DE PROBLEMA

Una vez justificada la idea de la cual surge este trabajo, es conveniente separar sus piezas fundamentales para poder simplificar el proceso de materialización del mismo.

Con base en el capítulo 1, se identifican 3 bloques principales que construyen la idea de este proyecto:

1. Obtención de posición y orientación espacial de las manos del usuario para reproducir un sonido en función de sus movimientos.
2. Entorno virtual para proveer al usuario una retroalimentación espacial y visual de sus movimientos, acompañada de los sonidos generados, dentro del entorno, por los mismos.
3. Realidad aumentada para que el usuario pueda tener interacción con el entorno real.

4 OBJETIVOS

Es importante, en cualquier aspecto de la vida, contar con un objetivo. Una luz que guíe el andar en este mundo y algo a lo cual aferrarse en momentos de duda. Descrito el problema de este trabajo, es posible generar un objetivo, de una forma estructurada y sencilla, que guíe su materialización.

OBJETIVO GENERAL

Desarrollo de software que permita la interacción con un instrumento virtual de percusión, con base en la interacción con un instrumento de percusión real, mediante la obtención de posición y orientación espacial de las manos, sin perder la capacidad de interacción con el entorno, con el uso de realidad aumentada.

Para facilitar la realización de este proyecto, se generan objetivos específicos para definir puntos que deben ser realizadas a lo largo de este proceso.

OBJETIVOS ESPECIFICOS

- Obtención de información de usuarios de instrumentos de percusión, para generar una interacción con el instrumento virtual parecida a la interacción con un instrumento real
- Selección de tecnologías que permitan una interacción con el instrumento virtual
- Desarrollo de entorno virtual, y hardware necesario, para emular el instrumento de percusión
- Realización de pruebas en usuarios de instrumentos de percusión

5 ESTADO DEL ARTE

Con el fin de obtener resultados fructíferos de la obtención de información de usuarios de instrumentos de percusión, se deben conocer, de forma general, algunas tecnologías o herramientas que puedan satisfacer el objetivo general de éste proyecto. En específico, los 3 bloques descritos en el capítulo 3.

POSICION Y ORIENTACION EN EL ESPACIO

HTC VIVE

HTC Vive tuvo gran importancia en la gestación de la idea para este trabajo debido a su capacidad de obtener la posición espacial del usuario, y la posición y orientación espacial de sus manos, u otro dispositivo físico en donde se encuentren los controles.

Para obtener dicha información, el sistema cuenta con dos cámaras infrarrojas, las cuales deben ser colocadas en los extremos del área de movimiento, las cuales obtienen la información de los sensores ubicados en el visor y en los controles para, posteriormente, calcular la posición del usuario y sus controles, además de la orientación de los mismos.

Los controles cuentan con 24 sensores cada uno para lograr un seguimiento preciso de los mismos. [8]

En la Figura 6 se muestran los elementos de HTC VIVE. En el centro se encuentra el visor, en la parte superior las cámaras infrarrojas, y en la parte inferior los controles.

Figura 6 HTC VIVE [9]

LEAP MOTION

Generalmente, la forma de interactuar con elementos virtuales es mediante el uso de algún control. VIVE utiliza los controles para poder detectar las manos del usuario u otro objeto al que estén sujetos.

Leap Motion ofrece la capacidad de interactuar con elementos virtuales mediante el movimiento de las manos del usuario. Es decir, se cuenta con un modelo virtual de manos, las cuales se mueven en función del movimiento de las reales.

El dispositivo cuenta con sensores infrarrojos y dos cámaras, los cuales siguen el movimiento de cada uno de los dedos. Al tener la posición de cada uno de los dedos, se puede obtener la posición y orientación espacial de las manos.

En la Figura 7 se muestra el uso de Leap Motion para interactuar con una flor virtual.

Figura 7 Leap Motion [11]

KINECT

Leap Motion proveé la capacidad de interactuar con elementos virtuales mediante movimientos de mano. Kinect permite la interacción con elementos virtuales mediante movimientos del cuerpo.

Kinect es un sensor de Microsoft que fue lanzado para ser usado con la consola XBOX 360, posteriormente se distribuyó software para ser implementado con computadoras.

Microsoft ha desarrollado varias versiones del sensor. Este cambio ha modificado el método en que los sensores realizan su seguimiento, pero, en general, han seguido el mismo principio. El sensor mide el tiempo en el cual una onda de luz viaja desde si mismo a cada uno de los puntos presentes en una imagen obtenida por una cámara. Esto genera un mapa de profundidad y, con el uso de algoritmos, es posible calcular la posición y movimientos del usuario.

En la Figura 8 se muestra la última versión del sensor Kinect.

Figura 8 Kinect [12]

ENTORNO VIRTUAL

Existen una gran cantidad de software dedicado a la elaboración de entornos virtuales. Unity y Unreal son dos de los más conocidos y usados por distintos desarrolladores.

En ambos software es posible desarrollar un entorno virtual. La diferencia entre los mismos, principalmente, son la cantidad y calidad de herramientas brindadas, calidad de gráficos, lenguaje de programación, precios, y tamaño de tienda de elementos.

UNITY

Unity ofrece una versión gratis la cual contiene todas las herramientas básicas para la creación de un entorno virtual. Entre las herramientas, está la posibilidad de usar el entorno creado en plataformas móviles como iOs o Android.

El tamaño de su tienda virtual es superior a la ofrecida por Unreal, dando una mayor libertad creativa en el proceso de elaboración de entornos.

Dentro de Unity, es posible usar dos lenguajes de programación: C# o Javascript, brindando flexibilidad al momento de programar dependiendo de las funciones que se deseen lograr.

En el momento en que se deseé lanzar un proyecto, es necesario pagar por una versión más avanzada del software, la cual permite monetizar, dar seguimiento de los recursos computacionales usados, obtener datos de uso, entre otras cosas. El precio de la versión avanzada varía entre 35 y 125 dólares por mes.

UNREAL

Unreal ofrece todas su herramientas gratis hasta el momento del lanzamiento del proyecto. Ofrece todo lo necesario para crear un entorno virtual, seguir los recursos computacionales usados, compatibilidad con plataformas móviles, entre otras cosas.

La calidad de gráficos es superior a la ofrecida por Unity, ofreciendo una experiencia con mas detalles y calidad.

Dentro de Unreal, el único lenguaje posible para programar es C++. Existe otro método para programar mediante la creación de conexiones entre los elementos usados. Esto ofrece una oportunidad a los desarrolladores que no sepan programar en C++ u otro lenguaje.

A diferencia de Unity, cuando es lanzado un proyecto, no hay que pagar por una versión avanzada del software, pero hay que pagar un 5% de las ganancias generadas con el mismo.

SOFTWARE REALIDAD AUMENTADA

Similar a la elaboración de entornos virtuales, existe una gran cantidad de software para incorporar realidad aumentada. Unity ofrece una opción, no tan elaborada o precisa como otros software, pero permite proyectar elementos virtuales sobre una imagen digital obtenida de la cámara de la computadora o dispositivo móvil.

Vuforia y Kudan son software muy parecido, en cuanto a las herramientas que ofrecen, y de los más populares existentes. Entre sus herramientas están la proyección de elementos virtuales sobre marcadores, proyección sin marcadores en dispositivos que cuenten con sensores para conocer su ubicación, compatibilidad con dispositivos móviles, etc. La principal diferencia entre ambos software es su precio.

VUFORIA

Vuforia ofrece, en su versión gratis, todas las herramientas necesarias para incorporar realidad aumentada al software en desarrollo, limitando la cantidad de marcadores que pueden ser usados, y restringiendo el servicio de nube.
Cuando es lanzado el software, el precio mínimo es de un pago de 500 dólares, con pagos mensuales dependiendo de la capacidad con la que debe contar el software y en cuyo caso hay que contactar a la empresa.

KUDAN

A diferencia de Vuforia, Kudan ofrece todas sus herramientas en su versión gratis. El precio depende del giro del desarrollador del software. Si el desarrollador es una institución educativa, una organización sin fines de lucro, o una MiPyMe y los ingresos son menores a 1.2 millones de dólares, las herramientas son gratis; en caso de ingresos mayores se deben pagar 1252 dólares al año. Para desarrolladores con otro rol, el precio debe ser acordado con la empresa.

HARDWARE REALIDAD AUMENTADA

El uso de realidad aumentada requiere de un dispositivo que proyecte elementos virtuales sobre una imagen digital o sobre lo que está observando el usuario de forma natural.

Para brindar una interacción virtual semejante a la real, es necesario que el dispositivo sea colocado frente a los ojos del usuario; por lo tanto, para este trabajo, se requiere de un visor de realidad aumentada, o de algún accesorio que permita poner otro dispositivo en la misma posición.

HOLOLENS

Hololens es un visor de realidad aumentada desarrollado por Microsoft. Funciona con el sistema operativo Windows 10 y permite la interacción con el entorno virtual mediante gestos realizados por la mano.

El visor cuenta con todo lo necesario para funcionar por sí mismo, es decir, sin usar algún procesador externo. Es inalámbrico y su batería dura aproximadamente 3 horas.

Los elementos virtuales pueden ser colocados sobre los elementos reales, mediante un mapeo del espacio en donde se encuentre el usuario. Dichos elementos permanecen en el área en donde se colocaron, aunque el usuario cambie su orientación o posición. Esta característica se logra con el uso de una IMU colocada dentro del visor. En la Figura 9 se muestra el dispositivo, cuya versión ofrece un campo de proyección de elementos virtuales reducido, aproximadamente de treinta y cinco grados, pero Microsoft no ha dado cifras oficiales. [19]

Figura 9 Microsoft Hololens [21]

Para realizar el mapeo del espacio, Hololens cuenta con 4 cámaras de reconocimiento de entorno y un sensor de profundidad, como Kinect. Además, cuenta con dos cámaras de alta definición para tomar fotos y video y un sensor de luz [20]. En la Figura 10 se muestra el arreglo de cámaras y sensores en el interior de Hololens

El precio de este dispositivo es de 3000 dólares [21].

Figura 10 Arreglo de sensores y cámaras de Hololens [22]

META2

Una de las principales características que brinda Hololens es su capacidad de funcionar sin necesidad de un procesador externo. Meta2 sí necesita de una computadora para poder realizar las proyecciones.

Cuenta con un sistema operativo basado en neurociencia, el cual ofrece una curva de aprendizaje nula. Es decir, idealmente, el usuario no necesita acostumbrarse a la tecnología, debido a la forma de interacción que ofrece el sistema con los elementos virtuales, similar a Leap Motion. Contiene un arreglo de sensores que permite la interacción con las manos y seguimiento espacial.

Los elementos virtuales pueden ser colocados en cualquier lugar en el espacio e interactuar con el medio real y, por medio de una IMU, permanecerán en el mismo espacio aún cuando el usuario gire o se mueva.

A diferencia de Hololens, Meta2 ofrece un campo de proyección de elementos virtuales de 90 grados. El precio de Meta2 es de 950 dólares [23]. En la Figura 11 se muestra el visor Meta 2.

Figura 11 Visor Meta 2 [23]

GOOGLE CARDBOARD

Una alternativa para usar realidad aumentada, sin necesidad de un visor diseñado específicamente para eso, es mediante el uso de un teléfono inteligente o tableta. Google ofrece un accesorio para poder usar un teléfono en la misma posición que un visor.

Cardboard es un diseño libre de visor lanzado en 2014, con el cual se puede insertar un teléfono dentro de la estructura, generalmente hecha en cartón, y poder contar con una experiencia similar a la ofrecida por otros visores. El teléfono realiza todo el procesamiento y, al no contar con los sensores necesarios para seguir las manos del usuario, no permite una interacción con los elementos virtuales. En la figura 12 se muestra el diseño ensamblado con un teléfono en su interior.

Figura 12 Google Cardboard [25]

SOLUCIONES EXISTENTES

Al conocer, de forma general, algunas tecnologías que pueden satisfacer el objetivo general de este trabajo, se debe investigar si existe algún producto que satisfaga, completamente o parcialmente, la necesidad que se presenta en el capítulo 1, y, en caso de existir, si usa las tecnologías antes mencionadas.

AERODRUMS

Aerodrums es un software que emula un instrumento de percusión que, con el uso de visión artificial, detecta la posición espacial de dos objetos esféricos sujetos en los extremos de unas baquetas. Conociendo la posición de las esferas, mediante algoritmos, produce un sonido dependiendo de los movimientos del usuario. Para los pedales, se usa el mismo principio. Se colocan dos paneles del mismo material en los pies del usuario y el software detecta sus movimientos. En la Figura 13 se muestra Aerodrums en uso.

Figura 13 Aerodrums [27]

Cuenta con una interfaz de dos dimensiones en donde se puede observar la posición en donde se encuentran las baquetas y la posición en donde deben de ser agitadas para producir los sonidos deseados. En la Figura 14 se muestra la interfaz de Aerdodrums.

El precio del producto es de 159.82 dólares [26].

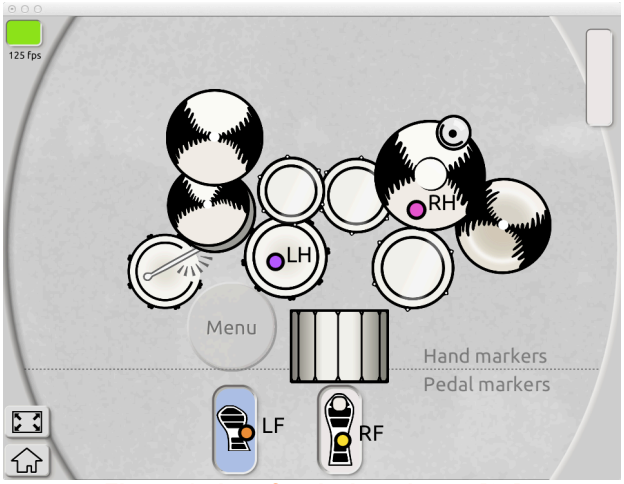

Figura 14 Interfaz de Aerodrums [27]

FREEDRUM

Freedom son sensores que pueden ser colocados sobre las baquetas y pies de los usuarios. Mediante unas IMU en su interior, se calcula la posición espacial de las mismas y con esa información, se produce un sonido en función de los movimientos del usuario. En la Figura 15 se muestran los sensores.

Cuenta con una aplicación con la cual se pueden modificar los sonidos producidos. Actualmente, el producto se encuentra en Kickstarter y para poder recibir dos sensores hay que contribuir con 69 dólares o más [28].

Figura 15 Sensores Freedrum [29]

6 DISENO CONCEPTUAL

Una vez que se conocen distintas tecnologías que pueden satisfacer el objetivo general mencionado en el capítulo 4, se necesita información de usuarios de instrumentos de percusión para, una vez analizada, poder construir una experiencia virtual que satisfaga sus necesidades.

Moisés Cabrera es un baterista profesional que ha tocado en distintas bandas de diversos géneros, ha participado en concursos internacionales y nacionales resultando entre los mejores. Además, junto con su hermano, cuenta con un estudio en donde ayuda a la grabación y producción de los artistas que acuden a él. Se realizó una entrevista con él y fue quien brindó la mayor cantidad de información útil para la elaboración de la experiencia virtual. En la Figura 16 se muestra a Moisés.

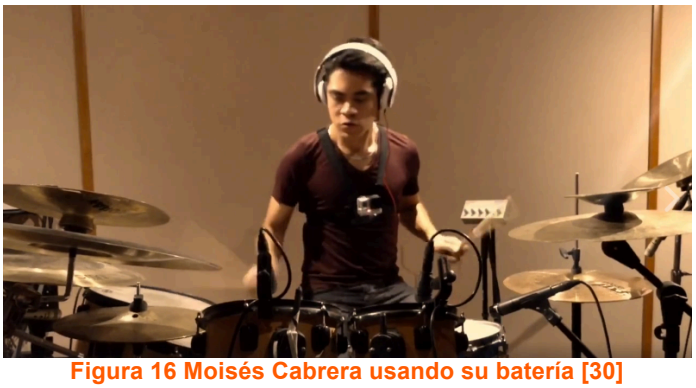

Figura 16 Moisés Cabrera usando su batería [30]

NECESIDADES

Durante la entrevista, se habló sobre lo que necesita un baterista de su instrumento y baquetas, desde un punto de vista práctico, hasta un punto de vista técnico, y cuales son los puntos que podrían resolverse o mejorarse con la elaboración de este proyecto.

De la información obtenida se abstraen las siguientes necesidades:

- Debe ser de fácil transporte
- Debe sonar como una batería real o MIDI
- El sonido de ciertos elementos debe variar dependiendo de la zona en donde se realicen las colisiones con otro elemento virtual
- Debe poder personalizarse
- El hardware, de ser necesario, debe poder acoplarse a las baquetas con las que ya se cuentan
- Debe brindar libertad de movimiento

REQUERIMIENTOS

Conocidas las necesidades a satisfacer, se interpretan con el fin de visualizar la forma en que se abordaran las mismas en la elaboración del sistema. Una vez interpretadas, se obtienen los siguientes requerimientos:

Tabla 1 Necesidades y Requermientos

NECESIDADES	REQUERIMIENTOS
Debe ser de fácil transporte	La cantidad de elementos, su tamaño, y peso debe ser menor a los de una batería real
Debe sonar como una batería real o MIDI	Contar con modelo virtual de batería Contar con modelo virtual de baquetas cuya orientación y posición espacial cambien en función de los movimientos del usuario para realizar las colisiones con la batería virtual Reproducción de sonidos reales o MIDI al existir una colisión entre un elemento de la batería virtual con las baquetas virtuales
El sonido de ciertos elementos debe variar dependiendo de la zona	Definir zonas y asignar un sonido a cada una
Debe poder personalizarse	Posibilidad de cambiar colores, posición y cantidad de los elementos virtuales
El hardware, de ser necesario, debe poder acoplarse a las baquetas con las que ya se cuentan	En caso de necesitar hardware en las baquetas, debe ser de un tamaño lo suficientemente pequeño para poder ser sujetado a las baquetas y su peso no debe sobrepasar el peso de las mismas, además debe ser inalámbrico para no interferir en los movimientos del usuario.

Debe brindar libertad de movimiento	Proyección de batería en frente del usuario El sistema debe cubrir el mismo volumen de interacción que se requiere al tocar con una batería real

ESPECIFICACIONES

Para abordar las necesidades, hay que contar con parámetros, métricas o especificaciones, interpretadas de los requerimientos, que permitan limitar o definir las capacidades de proyecto. De los requerimientos, se interpretan las siguientes especificaciones:

Tabla 2 Requerimientos y especificaciones

REQUERMIENTOS	ESPECIFICACIONES
La cantidad de elementos, su tamaño, y peso debe ser menor a los de una batería real	Cantidad máxima de elementos físicos Tamaño máximo de elementos físicos Peso máximo de elementos físicos
Contar con modelo virtual de batería	Número de elementos virtuales Elección de elementos virtuales Medidas de elementos virtuales
Contar con modelo virtual de baquetas cuya orientación y posición espacial cambien en función de los movimientos del usuario para realizar las colisiones con la batería virtual	Medidas de baquetas
Reproducción de sonidos reales o MIDI al existir una colisión entre un elemento de la batería virtual con las baquetas virtuales	Sonidos reales o MIDI de elementos escogidos

El sonido de ciertos elementos debe variar dependiendo de la zona en donde se realicen las colisiones con otro elemento virtual	Elementos con variación de sonido Zonas de variación o sin sonido
Posibilidad de cambiar colores, posición y cantidad de los elementos virtuales	Capacidad dentro de software de modificar la batería virtual
En caso de necesitar hardware en las baquetas, debe ser de un tamaño y peso pequeños para poder ser sujetado a las baquetas, además debe ser inalámbrico para no interferir en los movimientos del usuario.	Tamaño máximo de hardware Peso máximo de hardware
Proyección de batería en frente del usuario	Uso de visor de realidad aumentada
El sistema debe cubrir el mismo volumen de interacción que se requiere al tocar con una batería real	Volumen de interacción mínimo

A cada especificación se le asigna un valor, o rango de valores, que debe ser cumplido en la elaboración del sistema. Los valores asignados, y la justificación de porque se eligieron esos valores, son los siguientes:

Tabla 3 Especificaciones con valores y justificaciones

ESPECIFICACIONES	VALOR	JUSTIFICACION
Cantidad máxima de elementos físicos	< 11 elementos	Un equipo de batería básico, sin contar los soportes ni piezas pequeñas, tiene, generalmente 11 piezas.
Tamaño máximo de elementos físicos	< 2.3 [m3]	Para facilitar traslado el tamaño debe ser menor

Peso máximo de elementos físicos	< 70 [kg]	Es el peso mínimo de un equipo básico de batería, incluyendo pedales y soportes.
Número de elementos virtuales	> 8 elementos	Son los elementos de un equipo básico de batería que producen sonidos.
Elección de elementos virtuales	1 tarola 2 toms de aire 1 tom de piso 1 bombo 2 platillos 1 hi-hat	Los elementos mostrados son los elementos de un equipo básico de batería que producen sonidos.
Medidas de elementos virtuales	Tarola: 6”x14” Toms de aire: 10”x12”, 9”x10”, 8”x8” Tom de piso: 15”x16”, 14”x14” Bombo: 18”x22” Platillos: 8”, 18”, 20”, 22” Hi-hat: 14”	Son las medidas de los elementos reales
Medidas de baquetas	5A	Es la medida más usada por bateristas
Sonidos reales o MIDI de elementos escogidos	Tarola Toms de aire Tom de piso Bombo Platillos Hi-hat	Los elementos mostrados son los elementos de un equipo básico de batería que producen sonidos.
Elementos con variación de sonido	Platillos	En la zona de la campana tienen un sonido particularmente distinto al resto del platillo

Zonas de variación o sin sonido	Tarola Toms de aire Tom de piso Platillos Hi-hat	La tarola, toma de aire, y tom de piso, no producen sonido si se comisiona con ellos por los extremos o por abajo, o en su defecto, producen un sonido diferente. Platillos varían en la campana.
Capacidad dentro de software de modificar la batería virtual	> 8 elementos Color Posición	Son 8 los elementos de un equipo básico de batería que producen sonidos.
Tamaño máximo de hardware	Largo < 11 [cm] Ancho < 4 [cm] Profundo < 5 [cm]	Con base en baquetas 5A, se considera que dichas medidas no afectarán el uso de las mismas
Peso máximo de hardware	200 [g]	Se limita a este valor para no modificar la sensación del usuario al usar sus baquetas
Uso de visor de realidad aumentada	Visor	La batería debe ser proyectada frente al usuario
Volumen de interacción mínimo	1.344 [m3]	Es el volumen de interacción que, generalmente, es usado en un equipo de batería básico

CONCEPTO

La información obtenida de los usuarios de instrumentos de percusión, lograron brindar una guía de como poder satisfacer sus necesidades, mediante los requerimientos y especificaciones obtenidos. Con dicha guía es posible definir un concepto que describa el funcionamiento del software a elaborar. El concepto definido es el siguiente:

Software que emule un instrumento de percusión, brinde sonidos reales o MIDI, dependiendo de los elementos usados en la batería y la zona de colisión entre los mismos y una baqueta virtual, cuyo hardware, incluido el visor de realidad aumentada donde se utilizará, sea de fácil transporte, por número de elementos, tamaño y peso, y no interfiera con los movimientos del usuario en su volumen de interacción.

PROPUESTAS

El bloque más importante en la elaboración de este trabajo es la obtención de la posición y orientación espacial de las manos o baquetas del usuario, ya que sin esta información, aunque se contara con un entorno virtual que emule una batería, no podría funcionar.

Debido a las diferentes capacidades que ofrecen cada una de las tecnologías para obtener dicha información, la solución de este trabajo se construirá sobre la elección de una de estas, con base en su volumen de interacción y, en caso de contar con hardware para las baquetas o manos, su tamaño y peso. Posteriormente, dependiendo de las capacidades de la tecnología escogida, se darán solución a las demás especificaciones.

Con base en el capítulo 5 se generan las siguientes propuestas:

PROPUESTA 1: HTC VIVE

HTC VIVE brinda una detección de posición y orientación de los controles dentro de una habitación de 4.5 x 4.5 metros, por lo que satisface la especificación de libertad de movimiento en el volumen mínimo de interacción [31].

En la Figura 17 se representa el área dentro de la cual un usuario puede moverse.

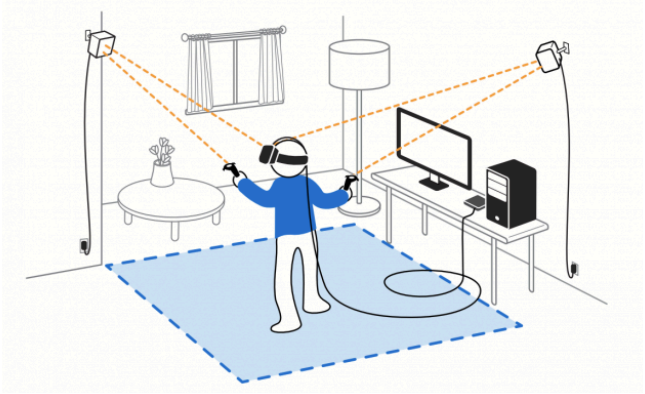

Figura 17 Área de uso de HTC VIVE [31]

El sistema sigue la posición y orientación de los controles, por lo que será necesario desarrollar un hardware o acoplar los controles a las baquetas del usuario.

Los controles, por su diseño, no brindan libertad de movimiento al usuario si se acoplan a las baquetas; es necesario usar otro hardware compatible con el sistema. VIVE cuenta con seguidores, más pequeños que los controles, que pueden ser acoplados con cualquier objeto y obtener su posición y orientación espacial. En la Figura 18 se muestra uno de estos seguidores.

Figura 18 Seguidor de VIVE [8]

Estos seguidores pueden ser fácilmente acoplados a las baquetas o manos del usuario, por lo que las especificaciones de peso máximo de hardware cumplen, sin embargo el tamaño máximo no lo cumple, pero se puede modificar ésta especificación si se coloca en las manos del usuario.

PROPUESTA 2: LEAP MOTION

Como se mencionó en el capítulo 5, Leap Motion ofrece detección de posición y orientación espacial de las manos del usuario, además de poder reconocer gestos y movimientos de los dedos del mismo.

Un beneficio de la plataforma es que permite el seguimiento de objetos con forma semejante a la de un lápiz o bastón, como las baquetas. Lo que descarta la necesidad de usar hardware en las baquetas. En la Figura 19 se muestra un ejemplo de esta característica de Leap motion.

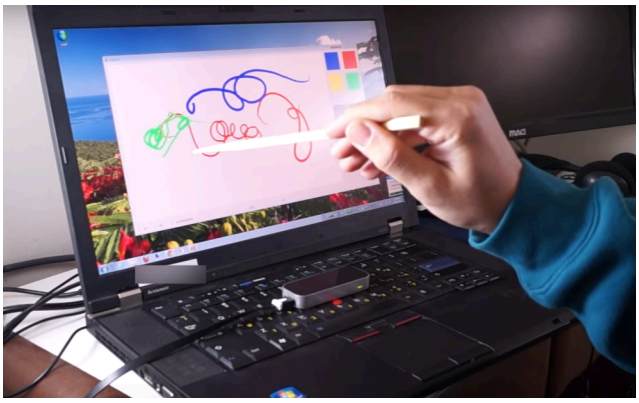

Figura 19 Control de programa con un lápiz mediante el uso de Leap Motion [20]

El volumen de interacción del dispositivo es de 60 centímetros por encima del dispositivo, 60 centímetros de ancho por cada lado del dispositivo con un ángulo de 150, y 60 centímetros de profundidad por cada lado del dispositivo con un ángulo de 120. Éste volumen no cumple con el volumen mínimo de interacción [32]. En la Figura 20 se muestra una representación de dicho volumen.

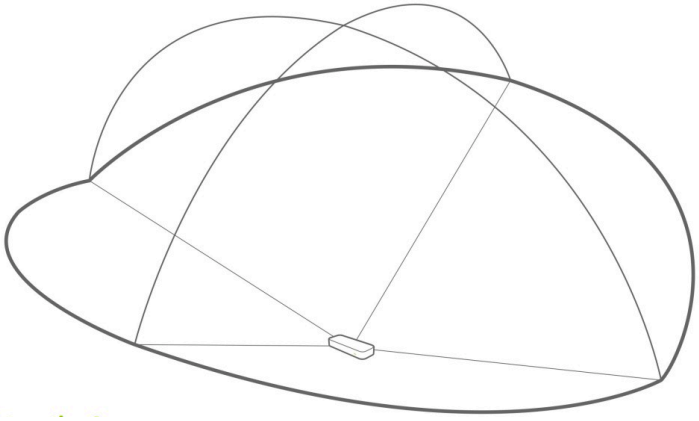

Figura 20 Rango de Leap Motion [33]

PROPUESTA 3: KINECT

Kinect ofrece la detección de distintas partes del cuerpo para poder conocer los movimientos del usuario, pero no permite conocer la orientación de las manos o baquetas del usuario, por lo que es necesario desarrollar hardware para obtener dicha información.

El dispositivo, por la cantidad de software elaborado para su uso, cuenta con la flexibilidad necesaria para usarse con plataformas distintas a Microsoft, facilitando y dando libertad a la elaboración del hardware necesario para las baquetas.

El volumen de interacción es de 1.2 metros frente al dispositivo hasta 3.5 metros, con un ángulo de visión de 43.5 grados, lo que permite una altura de 1.8 metros. Este volumen cumple con el volumen de interacción mínimo [34].

En las Figuras 21 y 22 se representa dicho volumen.

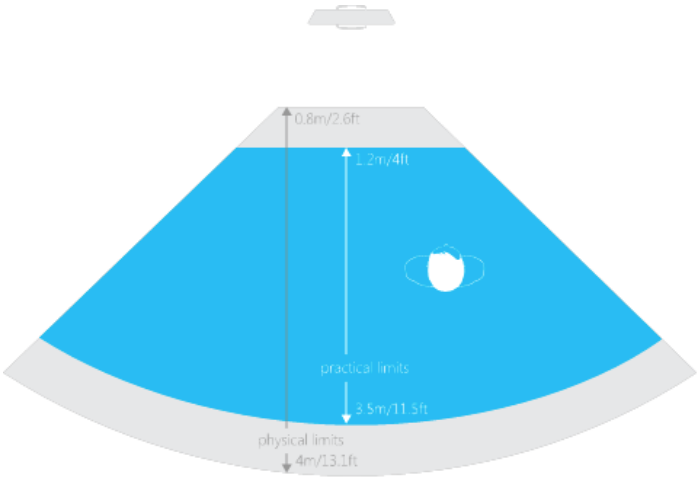

Figura 21 Vista superior de rango de Kinect [34]

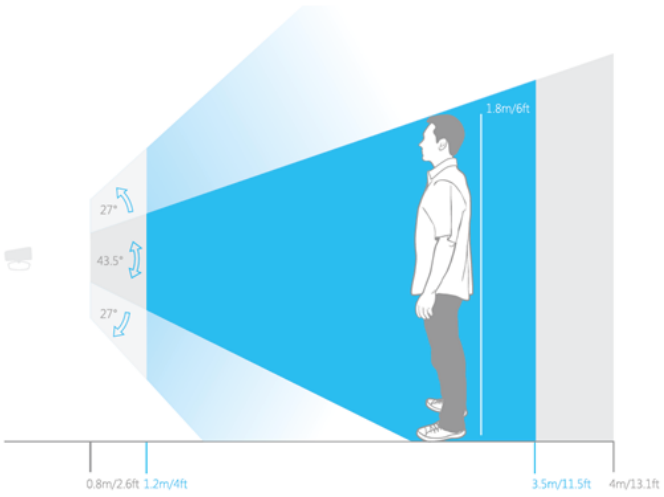

Figura 22 Vista lateral de rango de Kinect [34]

PROPUESTA 4: HOLOLENS / META 2

Ambos visores ofrecen detección de la posición y orientación espacial de las manos de usuario, lo que elimina la necesidad de desarrollar hardware para las baquetas. Sin embargo, al ser dispositivos diseñados para interactuar con los hologramas dentro del rango natural de vista, no cumplen con el volumen de interacción mínimo especificado.

ELECCION

En la Tabla 4 se agrupan las distintas propuestas, las especificaciones que se consideran para realizar la elección de una tecnología, y su relación.

Tabla 4 Relación entre propuestas y especificaciones importantes

ESPECIFICACIONES	PROP 1	PROP 2	PROP 3	PROP 4
Volumen de interacción mínimo	Cumple	No cumple	Cumple	No cumple
Tamaño máximo de hardware	Cumple con modificaciones	_____	Desarrollar	_____
Peso máximo de hardware	Cumple con modificaciones	_____	Desarrollar	_____

Se observa que la propuesta 1 satisface las especificaciones, con ciertas modificaciones, por lo que se considera que es la mejor tecnología para este trabajo. Sin embargo, no se cuenta con una disponibilidad inmediata para proseguir con la elaboración del sistema, por lo que se descarta el uso de esta tecnología para este trabajo.

Las propuestas 2 y 4 son descartadas por no cumplir con el volumen mínimo de interacción.

La propuesta 3 cumple con el volumen de interacción mínimo, además de contar con una gran cantidad de software para su uso en diversas plataformas; no permite la obtención de la orientación de las manos o baquetas del usuario, pero es posible desarrollar hardware, cumpliendo con las especificaciones, que realicen dicha función.

Al analizar las distintas propuestas, se elige Kinect, propuesta 3, como la tecnología sobre la cual se desarrollara la solución.

SOLUCION

Al escoger Kinect para obtener la posición de las manos, es necesario el desarrollo de hardware para obtener la orientación de las baquetas del usuario. Éste hardware debe ser inalámbrico, cumplir con el peso y tamaño máximo especificado, y poder acoplarse a unas baquetas 5A. En la Figura 27 se muestra un bosquejo del hardware.

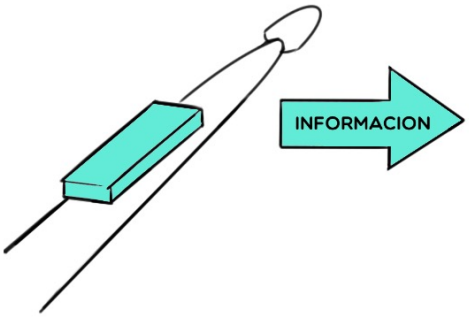

Figura 23 Bosquejo del hardware de baquetas

Es necesario emular los pedales de la batería con otros dispositivos. Estos dispositivos deben ser pequeños y de poco peso para cumplir con las especificaciones de tamaño y peso del sistema completo. Sin embargo, para enviar la información, pueden ser o no inalámbricos. En la Figura 28 se muestra un bosquejo de los pedales.

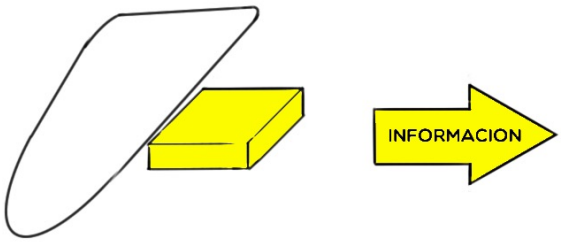

Figura 24 Bosquejo de hardware de pedales

Para obtener la información enviada del hardware de las baquetas y de los pedales, es necesario un receptor que además de recibir la información, pueda enviarla a la computadora y juntarla con la información de Kinect. En la Figura 29 se muestra un diagrama de la transmisión de información y un bosquejo del receptor.

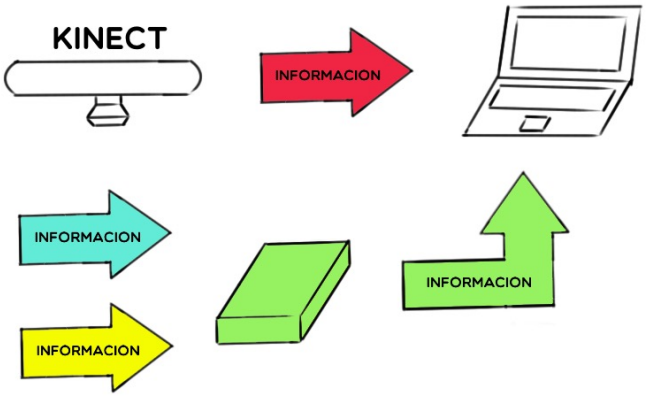

Figura 25 Diagrama de transmisión de información y bosquejo de receptor

El entorno virtual debe contener, por lo menos, 8 elementos que produzcan uno o mas sonidos dependiendo de la zona de colisión entre ellos y los modelos de unas baquetas de tamaño 5A. En la Figura 30 se muestra un bosquejo del entorno.

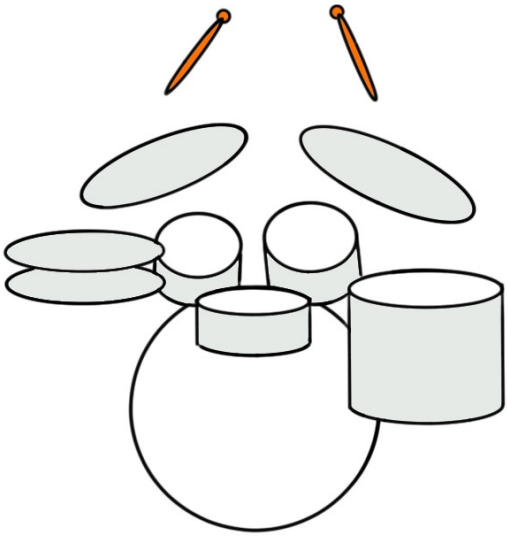

Figura 26 Bosquejo de entorno virtual

La batería debe ser proyectada frente al usuario, de modo que la posición de cada elemento virtual sea parecida a la posición real. Para poder interactuar de manera natural con la batería, las baquetas deben moverse en función de los movimientos del usuario; para esto es necesario enviar al entorno el paquete de información de Kinect, hardware de baquetas, y pedales. En la Figura 31 se muestra un bosquejo del uso del software con visor.

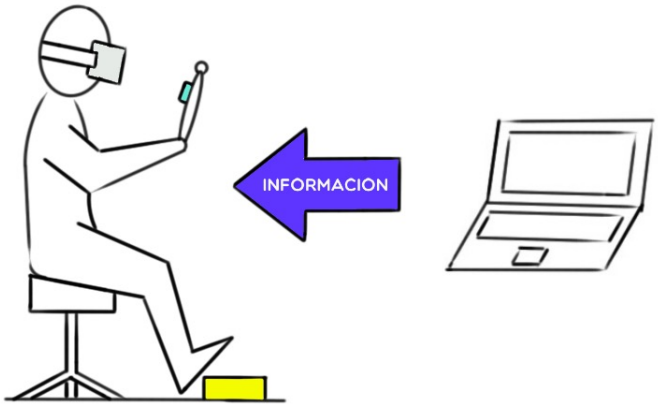

Figura 27 Bosquejo de uso de software

7 DISENO DE CONFIGURACION

En el capítulo 3 se obtuvieron los 3 grandes bloques que constituyen este trabajo:

- Posición y orientación espacial
- Entorno virtual
- Realidad aumentada

Con la solución generada, y con base en las especificaciones, es posible elegir los componentes y software con los que se desarrollará el trabajo.

POSICION Y ORIENTACION ESPACIAL

Se ha escogido Kinect como la base de éste proyecto. Para la obtención de su información se usará Processing, software diseñado para artes visuales, ya que cuenta con una librería que permite dicha obtención sin la necesidad de usar software de Microsoft. Para poder usar dicha librería, es necesario el uso del modelo 1414 de Kinect, debido a que las versiones posteriores sólo pueden ser utilizadas con software de Microsoft. En la Figura 32 se muestra dicho modelo.

Figura 28 Kinect 1414 [36]

Con lo mencionado en el párrafo anterior, se ha cubierto la obtención de la posición espacial de las manos del usuario. Se menciona en la solución que la orientación espacial que se utilizará será la de las baquetas, debido a que el usuario puede mover las baquetas con sus dedos, sin cambiar la orientación de las manos.

Para la obtención de la orientación espacial se usará un MPU6050. Este dispositivo es una unidad de medida inercial, IMU por sus siglas en inglés, que cuenta con un giroscopio de tres ejes, un acelerómetro de tres ejes, y un procesador de movimiento digital, DMP por sus siglas en inglés [37]. Dicho procesador provee de la orientación espacial del dispositivo en ángulos de Euler o cuaterniones, además brinda la información de la velocidad y aceleración angular. En la Figura 33 se muestra el componente.

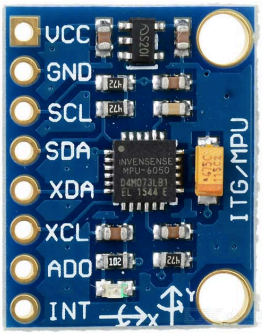

Figura 29 MPU6050 [38]

Para enviar los datos del MPU6050 al receptor de forma inalámbrica, se usarán módulos Bluetooth HC-05. Estos módulos utilizan el protocolo de puerto serial para la transmisión de datos, pueden ser configurados como maestros y esclavos, además de su velocidad, nombre, entre otras cosas, mediante comandos AT. En la Figura 34 se muestra el módulo.

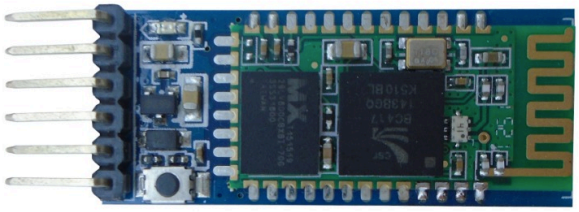

Figura 30 HC-05 [40]

Para la obtención de los datos del MPU6050 y su transmisión con el módulo HC-05, se usará un microcontrolador Arduino Pro Mini, debido a sus medidas y características. Existen dos modelos de este microcontrolador, uno de 3 [V] y otro de 5 [V], debido a la diferencia en voltajes de alimentación. Ambos modelos ofrecen lo mismo, con excepción de la velocidad a la que pueden trabajar, siendo el de 5 [V] más rápido que el de 3 [V]. Para la elaboración del hardware, se utilizará el modelo de 5 [V]. En la Figura 35 se muestra el microcontrolador.

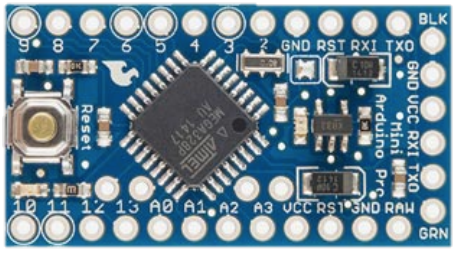

Figura 31 Arduino Pro Mini [41]

Los dispositivos mencionados cumplen con las especificaciones dadas para el desarrollo de hardware de las baquetas.

Para el receptor se usará un Arduino Due, debido a la capacidad de procesamiento y la cantidad de puertos de comunicación serial, para la obtención de datos vía Bluetooth, con los que cuenta. En la Figura 36 se muestra dicho microcontrolador.

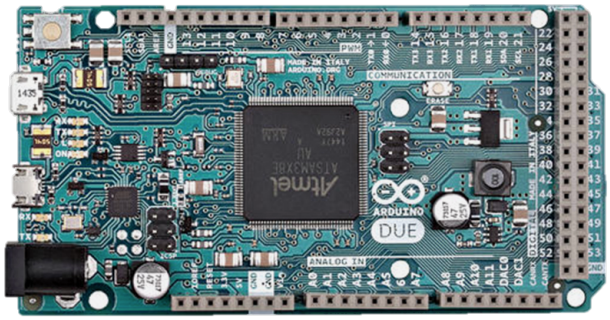

Figura 32 Arduino Due [42]

Finalmente, para los pedales se usaran botones, debido a su tamaño y peso. Dado que no se especifica que el envío de información de los pedales sea o no de forma inalámbrica, se realizará mediante conexiones con el Arduino Due, debido a la facilidad de este método en comparación del uso de módulos Bluetooth. En la Figura 37 se muestra uno de los botones mencionados.

Figura 33 Pushbutton [43]

Al usar la plataforma de Arduino, es posible enviar los datos obtenidos de los pedales y las baquetas a Processing para poder juntarlos con la información obtenida de Kinect.

ENTORNO VIRTUAL

La elaboración del entorno virtual requiere de los modelos virtuales de los elementos de una batería, y darles un comportamiento a cada uno, ya sea que realicen o no una acción.

Para la elaboración de los modelos virtuales se usará Fusion 360, software de modelado en computadora de Autodesk, y para la elaboración del entorno virtual Unity, debido a que ya se ha trabajado con estos software.

REALIDAD AUMENTADA

Se usará Unity para la implementación de realidad aumentada, debido a la facilidad de uso y la estabilidad de proyección sin necesidad de usar marcadores.

Para la proyección del entorno virtual, debido a que no se cuenta con una disponibilidad inmediata de Hololens o Meta 2, se optará por usar un teléfono, para lo cual es necesario el uso de Google Cardboard.

Al usarse un teléfono, no es posible enviar los datos de posición y orientación espacial mediante un cable, por lo que se debe buscar una forma de poder enviar dichos datos de la computadora al teléfono. Esta comunicación se puede realizar con el uso de software orientado a Internet de las cosas, con lo cual se crea un nuevo bloque en la elaboración de este trabajo.

INTERNET DE LAS COSAS

Para que el entorno virtual reciba la información de la posición y orientación espacial de las manos y baquetas del usuario, es necesario el uso de un software que pueda integrarse en Processing y Unity que logre enviar información a través de Internet.

Spacebrew es una plataforma, servicio y herramienta para coreografiar espacios interactivos, permite la conexión entre distintos objetos a través de internet, permite seleccionar el camino por el que se transmitirá información, brinda de una interfaz dinámica donde se puede cambiar el camino de conexión y tener una retroalimentación de la misma [46], entre otras cosas. Este software es compatible con Processing y Unity, por lo que se elegirá para la transmisión de información.

8 ELABORACION

Una vez definidos los componentes y software que se usarán, se procede a juntar cada uno de ellos para, en conjunto, materializar la idea que se planteó en el capítulo 1.

POSICION Y ORIENTACION ESPACIAL

KINECT

El primer componente que se usará será Kinect, debido su gran importancia. Para lograr usar este dispositivo en una computadora Apple, con la que el autor trabaja, es necesario contar con 4 software:

- Xcode: Software para desarrollar aplicaciones para dispositivos de Apple, además de permitir el uso de la terminal de la computadora [48]

- XQuartz: Versión libre del sistema de ventanas X, usado para generación de gráficos e interacción con dispositivos externos [50] [51]
- CMake: Software libre para correr, probar y empaquetar software. Se usa para controlar la compilación, y generar archivos y espacios para usarse en el entorno de compilación deseado [52]
- MacPorts: Software para instalar y actualizar software a través de la línea de comandos [53]

Con los software mencionados instalados, se instalan las librerías necesarias, a través de la terminal, para poder interactuar con Kinect:

- Libtool: Librería genérica para soporte de librerías [55]
- Libusb: Librería que provee de acceso genérico con dispositivos USB [56]
- OpenNI: Librería que permite interactuar con Kinect, hecha por la misma compañía que diseñó el sensor [57]
- SensorKinect: Librería que permite la comunicación entre Kinect y OpenNI [58]
- NiTE: Librería que utiliza la información obtenida de Kinect para poder seguir las manos o cuerpo del usuario, también hecha por la compañía que diseñó el sensor [59]

Con las librerías mencionadas instaladas, es posible correr los ejemplos del procesamiento de NiTE. En la Figura 34 se muestra uno de dichos ejemplos, donde se sigue la posición de la mano derecha del usuario.

Figura 34 Ejemplo NiTE

Ahora que existe una comunicación entre Kinect y la computadora, Processing debe poder contar con dicha información. Para lograr esta obtención de información es necesario instalar la librería de SimpleOpenNI en Processing [62]. En la Figura 35 se muestra el algoritmo para la obtención, y muestra de la información de Kinect en Processing.

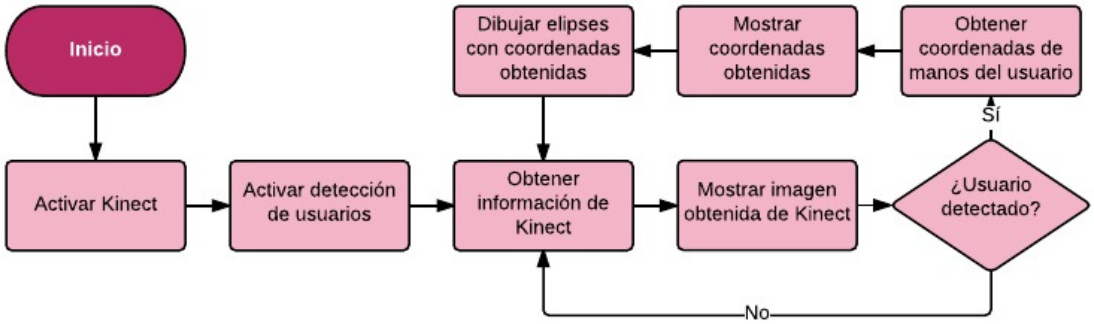

Figura 35 Algoritmo para obtención, muestra, y dibujo de coordenadas de manos de usuario

En la Figura 36 se muestra la implementación del algoritmo en Processing. La mano derecha y sus coordenadas aparecen de color verde, la mano izquierda y sus coordenadas aparecen de color azul.

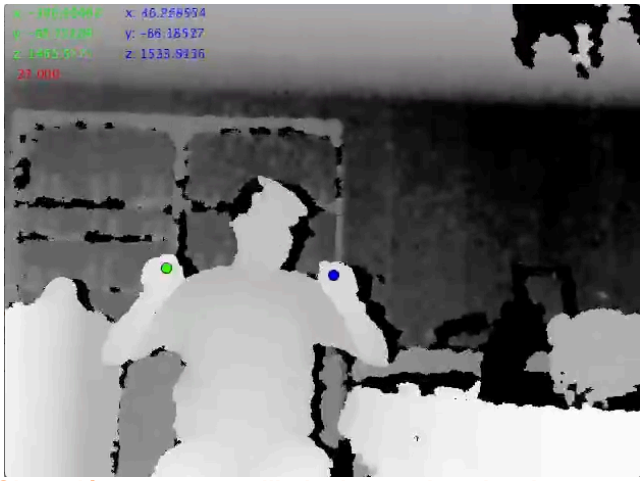

Figura 36 Obtención, muestra, y dibujo de coordenadas de manos de usuario

Para usar los datos no es necesario dibujar esferas en las coordenadas obtenidas de las manos del usuario ni mostrar la imagen obtenida por el sensor. En el la Figura 37 se muestra el algoritmo reducido.

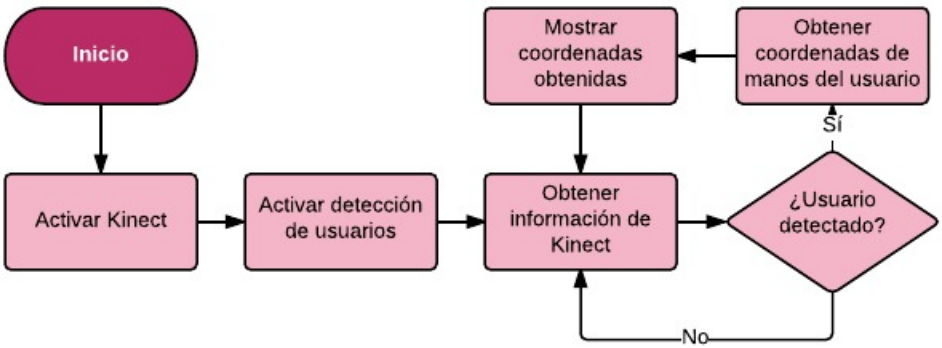

Figura 37 Algoritmo reducido para obtención, muestra, y dibujo de coordenadas de manos del usuario

Para facilitar el posterior envío de información, las coordenadas se agruparan en una cadena de caracteres. Los primeros términos pertenecerán a las coordenadas de la mano derecha, seguidas de las de la mano izquierda. Para separar los datos, entre cada coordenada se pondrá una coma como se muestra a continuación, donde el subíndice *D* se refiere a las coordenadas de la manos derecha y el subíndice *I* a la mano izquierda:

$$X_D, Y_D, Z_D, X_I, Y_I, Z_I$$

En la Figura 38 se muestra la implementación del algoritmo reducido en Processing. En rojo se muestra la información obtenida de Kinect.

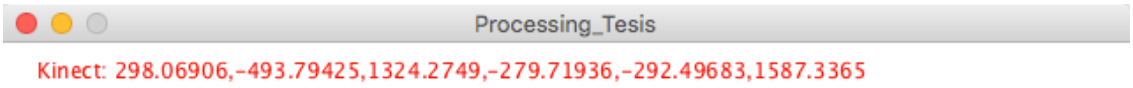

Figura 38 Información obtenida de Kinect en Processing

MPU6050

Con la obtención de datos de Kinect, se prosigue a la obtención de la orientación espacial de las baquetas, mediante MPU6050. Al ser dos baquetas, se usarán dos dispositivos, además de dos módulos HC-05 para la transmisión de información, y dos Arduino Pro Mini para realizar el envío de la información.

Para usar el MPU6050, es necesario usar un Arduino Pro Mini. El circuito que se usará se muestra en la Figura 39.

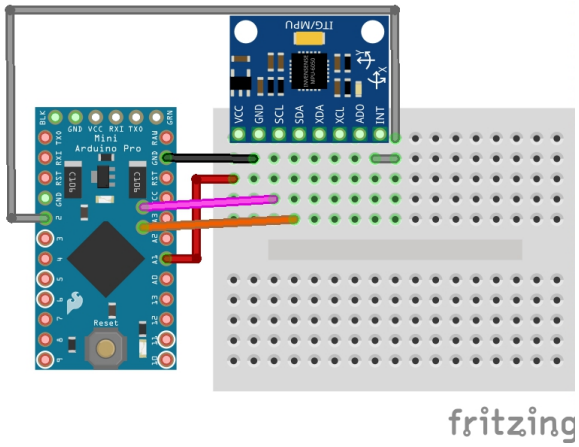

Figura 39 Circuito de baquetas - Arduino y MPU6050

Para obtener datos correctos del MPU6050, es necesario calibrar el dispositivo y obtener los desfases que presente para, posteriormente usarlos. Para realizar dicho proceso, se utilizará un código elaborado por Luis Ródenas [63], basado en un código para la obtención de los datos elaborado por Jeff Rowberg [64].

Para utilizar los códigos, se requiere instalar tres librerías:

- Wire: Librería para comunicación I2C entre Arduino y otro dispositivo.
- I2Cdev: Librería para comunicación I2C entre Arduino y otro dispositivo, basada en Wire [64]
- MPU6050: Librería para comunicación I2C entre Arduino y un MPU6050 [64]

Se requiere de la comunicación I2C debido a que es una de las dos formas en las que un MPU6050 puede comunicarse con otro dispositivo [65].

La comunicación I2C, es un bus bidireccional de información que requiere de dos canales, uno para transmisión de información serial de 8 bits, y otro para una señal de reloj, que marca cuando transmitir información. Su nombre viene de sus siglas en inglés *Inter-Integrated Circuits*, circuitos inter-integrados en español [67].

Para establecer una conexión es necesario contar con la dirección del dispositivo con quien se desea transmitir información. En el caso de un MPU6050, la dirección es 0x68 [66].

El algoritmo para la calibración se muestra en la Figura 40.

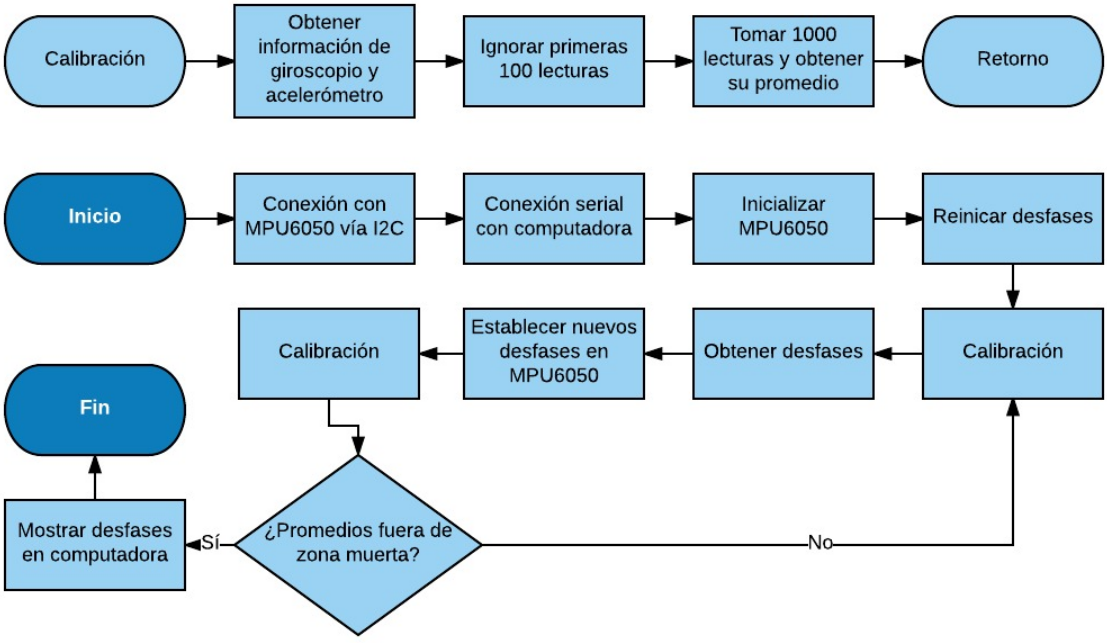

Figura 40 Algoritmo para calibración de MPU6050

Instaladas las librerías, se corrió el programa de calibración y se obtuvieron los desfases mostrados en la Tabla 5 para ambos MPU6050:

Tabla 5 Desfases de MPU6050

	aceIX	aceIY	aceIZ	giroX	giroY	giroZ
Derecha	-2726	-2340	1341	61	-19	-2
Izquierda	413	-1473	1682	89	-69	38

El código para la obtención de datos de un MPU6050 hace uso del FIFO integrado en dicho dispositivo. Un FIFO es un espacio de memoria dedicado a guardar temporalmente información antes de ser usada, llamado buffer, en el cual el primer elemento en entrar, *First In* en inglés, es el primero en salir, *First Out* en inglés.

Con ambos desfases obtenidos, se ingresan en el código para la obtención de datos del MPU6050, cuyo algoritmo se muestra en la Figura 41.

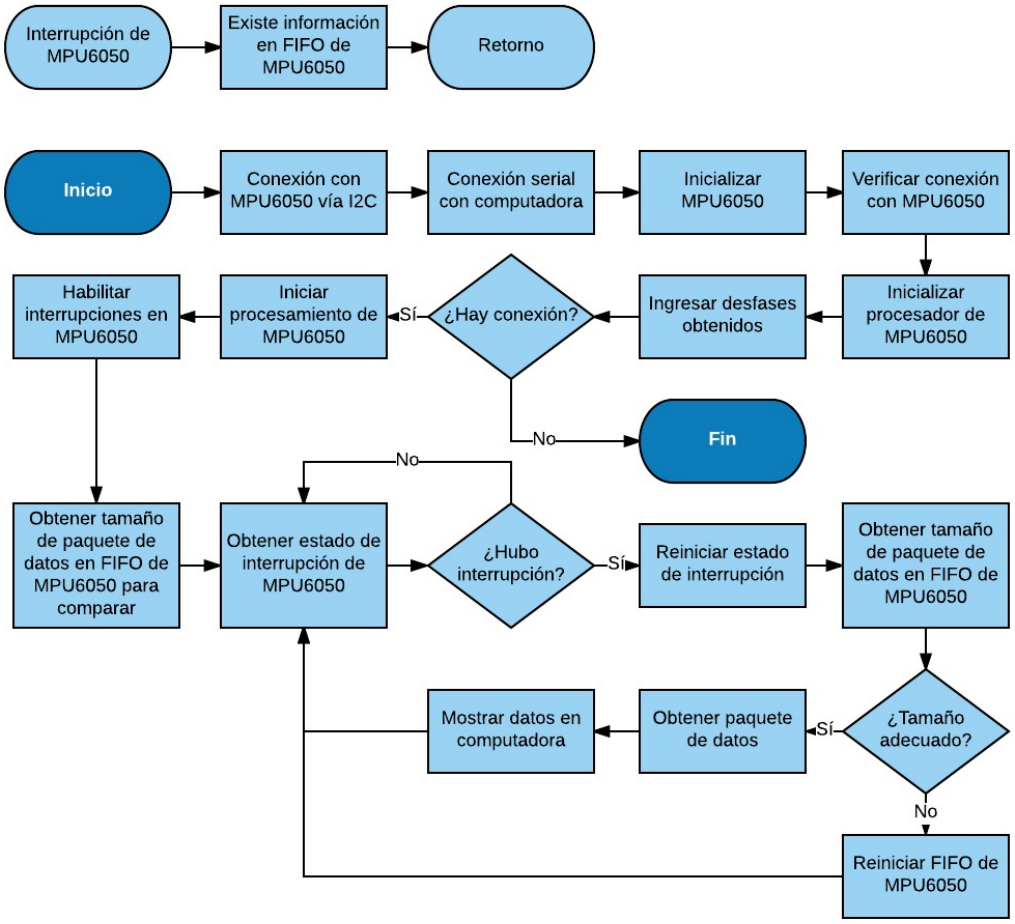

Figura 41 Algoritmo para obtención de datos de MPU6050

Los MPU6050 pueden brindar su orientación espacial en forma de ángulos de Euler o cuaterniones, además pueden brindar su velocidad y aceleración angular.

Los ángulos de Euler son tres ángulos con los cuales se puede describir la orientación espacial de un objeto. Cada ángulo es una rotación, en sentido antihorario, sobre un eje de un sistema de coordenadas tridimensional. *Yaw* es el ángulo sobre el eje z, *pitch* sobre el eje y, y *roll* sobre el eje x.

En la Figura 42 se muestran dichos ángulos sobre sus respectivos ejes.

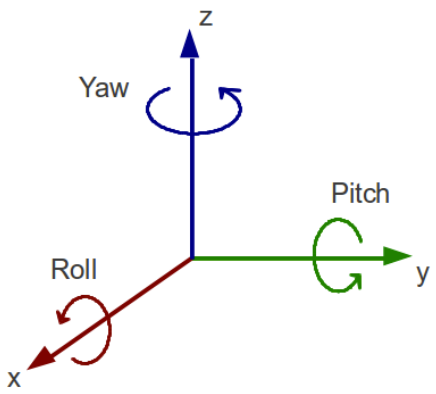

Figura 42 Ángulos de Euler [70]

El uso de estos ángulos tiene varias desventajas, se mencionan algunas:

- El orden en el cual se apliquen las rotaciones sobre cada eje, influye en la orientación final. Es decir, no son conmutativos.
- Si un eje es rotado de forma tal que sea paralelo con otro eje, se cancelará un ángulo. A este efecto se le llama bloqueo Gimbal.
- Rotaciones de una orientación a otra requieren una gran cantidad de procesamiento y no se realizan de una forma suave.

Los cuaterniones son entes matemáticos descubiertos por Hamilton en 1843, en su deseo de representar los números complejos en tres dimensiones. Cuentan con una parte real y tres imaginarias, con sus respectivas definiciones. La representación de los cuaterniones se muestran en la ecuación 1, y en las ecuaciones 2 a 5 se muestran las definiciones de sus partes imaginarias [72].

$$Q = q_w + q_x\hat{\imath} + q_y\hat{\jmath} + q_z\hat{k} \quad (1)$$
$$\hat{\imath}^2 = \hat{\jmath}^2 = \hat{k}^2 = -1 \quad (2)$$
$$\hat{\imath}\hat{\jmath} = -\hat{\jmath}\hat{\imath} = \hat{k} \quad (3)$$
$$\hat{\jmath}\hat{k} = -\hat{k}\hat{\jmath} = \hat{\imath} \quad (4)$$
$$\hat{k}\hat{\imath} = -\hat{\imath}\hat{k} = \hat{\jmath} \quad (5)$$

Con el uso de cuaterniones se pueden realizar rotaciones evitando algunos inconvenientes presentados con el uso de los ángulos de Euler. Por dicha razón, se obtendrá la orientación espacial de los MPU6050 en forma de cuaterniones.

Se obtendrá también la aceleración angular de los dispositivos para una posible necesidad de uso en futuros pasos. La información obtenida se agrupa en una cadena de caracteres, separando cada término por una coma, de la siguiente forma, las aceleraciones sobre cada eje se representan con una *a*:

$$q_w, q_x, q_y, q_z, a_x, a_y, a_z$$

El envío de la información obtenida de los MPU6050 por los Arduino Pro Mini, será enviada mediante módulos HC-05. Para la transmisión de información entre el Arduino y el módulo Bluetooth, se requiere comunicar ambos dispositivos mediante comunicación RS232.

La comunicación RS232, al igual que I2C, es una comunicación serial. No requiere de una señal de reloj, utiliza un solo bus, por el cual se envían bits de información, pero se utilizan dos buses separados, uno para el envío de información, *TX*, y otro para recibir información, *RX*, y, además requiere de una referencia.

Los códigos para calibración y obtención de datos de un MPU6050 utilizan dicha comunicación para poder enviar los datos a la computadora. En este caso, ya no se enviarán los datos a la computadora, se enviarán al pin RX del HC-05. Por lo tanto el algoritmo usado no cambiará.

Antes de enviar información vía Bluetooth, es necesario configurar el módulo. La configuración utiliza comandos AT, que son comandos diseñados para poder configurar dispositivos de transferencia de datos [73].

Para realizar dicha configuración, se usará un código realizado por Hakim Bitar [73], cuyo algoritmo se muestra en la Figura 43.

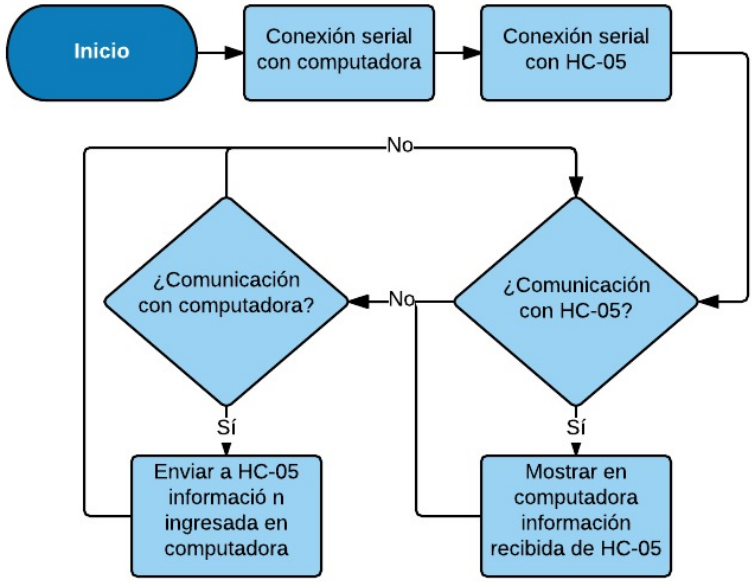

Figura 43 Algoritmo para configuración de HC-05

Se realizan las siguientes acciones con el módulo:

- AT + MODE = 1: Establece el módulo como maestro, debido a que es quien brindará la información al receptor.
- AT + ADDR?: Regresa la dirección única del módulo para que otros dispositivos puedan conectarse con éste.

Las direcciones obtenidas de los módulos son:

- Baqueta derecha: 2016:5:235436
- Baqueta izquierda: 2016:5:238223

Para el envío de la información del MPU6050 mediante HC-05, se utilizará el circuito de la Figura 44, y el algoritmo de la Figura 45.

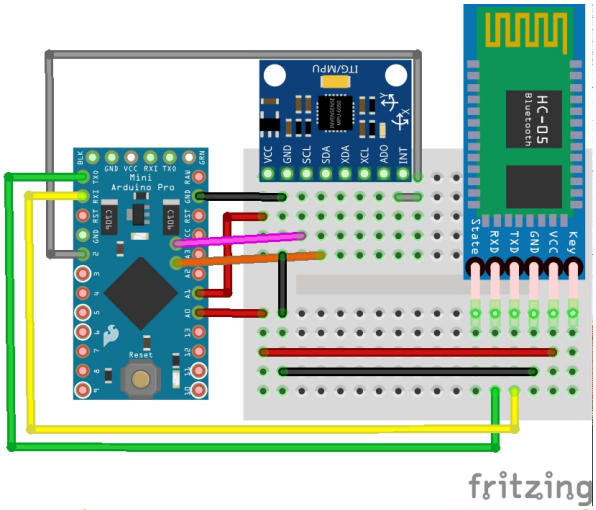

Figura 44 Circuito de baquetas - Arduino, MPU6050, HC-05

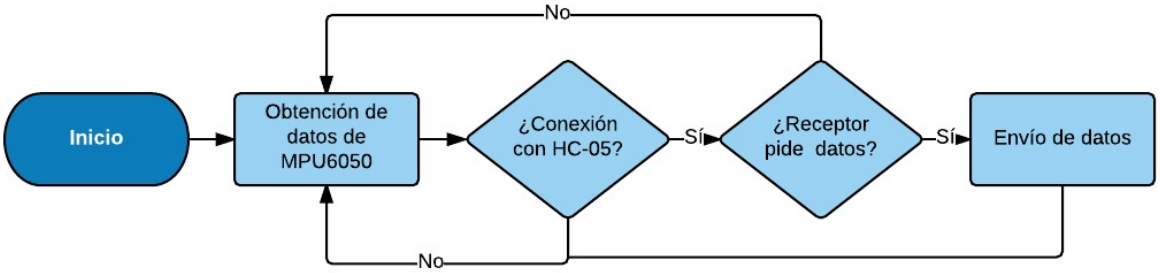

Figura 45 Algoritmo para envío de datos de MPU6050 con HC-05

El dispositivo para las baquetas generado se muestra en la Figura 46. Las dimensiones del dispositivo son 9 [cm] de largo, 3.7 [cm] de ancho, y 2.5 [cm] de profundidad, con lo cual se cumplen las especificaciones generadas para dicho hardware.

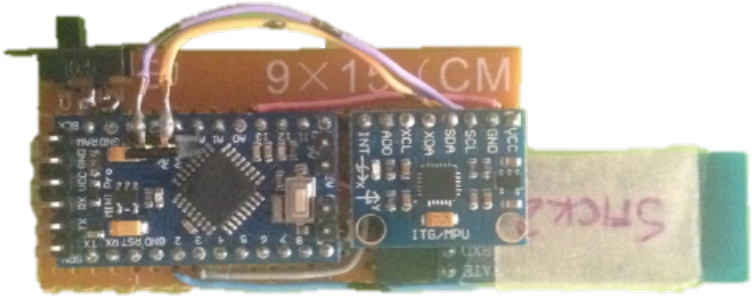

Figura 46 Dispositivo para baquetas

La alimentación del dispositivo, se puede realizar con el uso de tres baterías de botón recargables, cada pila de 3.6 [V], con las que no se violan las especificaciones para hardware de baquetas. En la Figura 47 se muestra una de las baterías mencionadas.

Figura 47 Batería de botón recargable [74]

RECEPTOR Y PEDALES

Para mandar la información de las baquetas a la computadora, es necesario contar con un receptor que realice el envío. Dicho receptor, además, debe de obtener el estado de los emuladores de pedales, si son presionados o no.

Para recibir la información se utilizarán módulos HC-05 y un Arduino Due, como se muestra en la Figura 48.

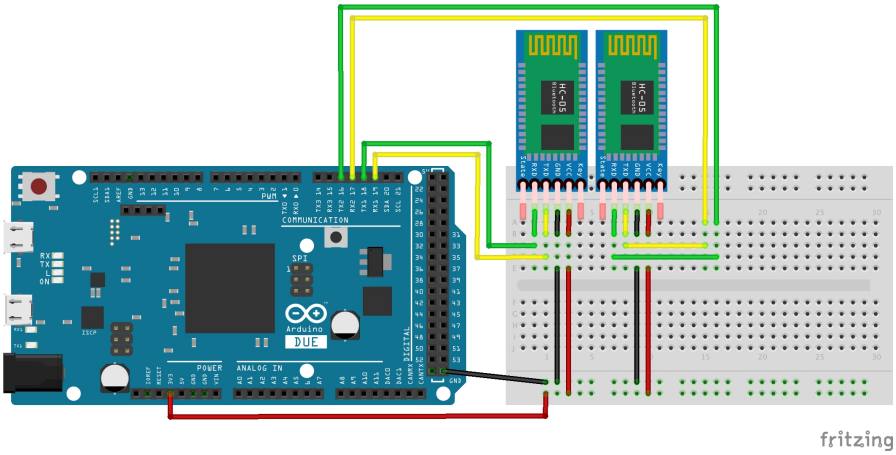

Figura 48 Circuito de receptor – Arduino, HC-05

Se configurarán con el código usado para las baquetas. Se ingresan los siguientes comandos para cada módulo:

- AT + MODE = 0: Establece el módulo como esclavo, debido a que serán quienes reciban la información
- AT + CMODE = 0: Indica que el modulo debe conectarse con otro modulo en específico y con ningún otro.
- AT + BIND = “Address”: Indica la dirección con la cual se debe conectar el modulo. La direcciones se obtuvieron previamente.

Una vez configurados los módulos, al momento de conectarse con las baquetas, solicitarán su información y comenzarán a recibir la cadena de caracteres con la información de su orientación espacial y aceleración angular.

Ambas cadenas de caracteres, una por baqueta, se agruparan en una sola cadena para poder realizar el envío de información a Processing. Los primeros términos pertenecerán a la información de la baqueta derecha, representados por una letra *D*, los términos posteriores pertenecerán a la baqueta izquierda representados por una letra *I*. Ambas cadenas se separan por una coma. La cadena de caracteres quedará de la siguiente manera:

$$q_{wD}, q_{xD}, q_{yD}, q_{zD}, a_{xD}, a_{yD}, a_{zD}, q_{wI}, q_{xI}, q_{yI}, q_{zI}, a_{xI}, a_{yI}, a_{zI}$$

El algoritmo utilizado se muestra en la Figura 49

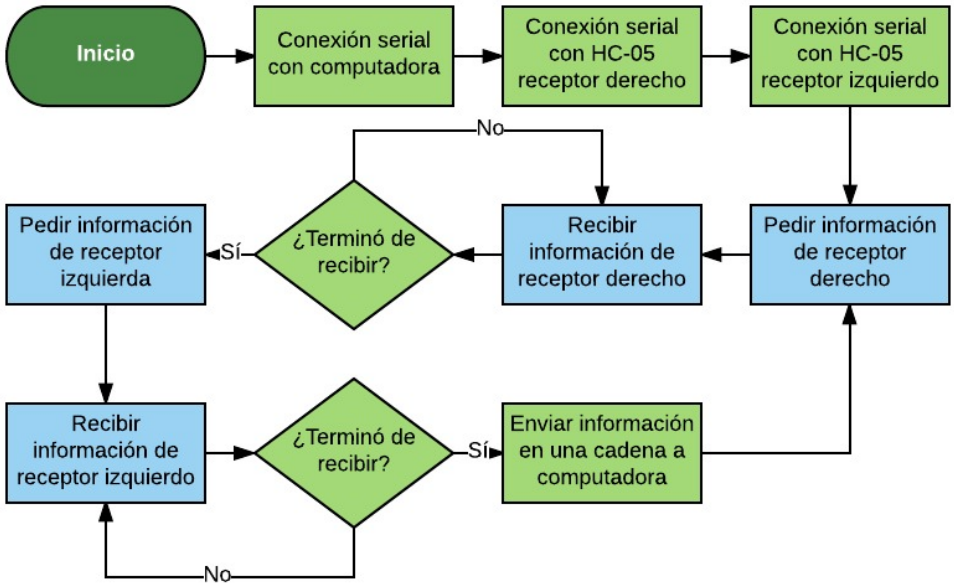

Figura 49 Algoritmo para recepción de datos de baquetas

Para emular los pedales se utilizarán botones, los cuales, al ser presionados, añadirán a la cadena de información un uno, en caso contrario se añadirá un 0. En éste trabajo se usarán 3 pedales, dos para bombo y uno para hi-hat. El orden en el que serán añadidos en la cadena de información se muestra a continuación, con *B* se representan los pedales del bombo, y con *H* representa al pedal de Hi-hat:

$$q_{wD}, q_{xD}, q_{yD}, q_{zD}, a_{xD}, a_{yD}, a_{zD}, q_{wI}, q_{xI}, q_{yI}, q_{zI}, a_{xI}, a_{yI}, a_{zI}, B_1, B_2, H$$

El circuito que se usará para recibir la información de los pedales se muestra en la Figura 50 y el algoritmo a utilizar se muestra en la Figura 51.

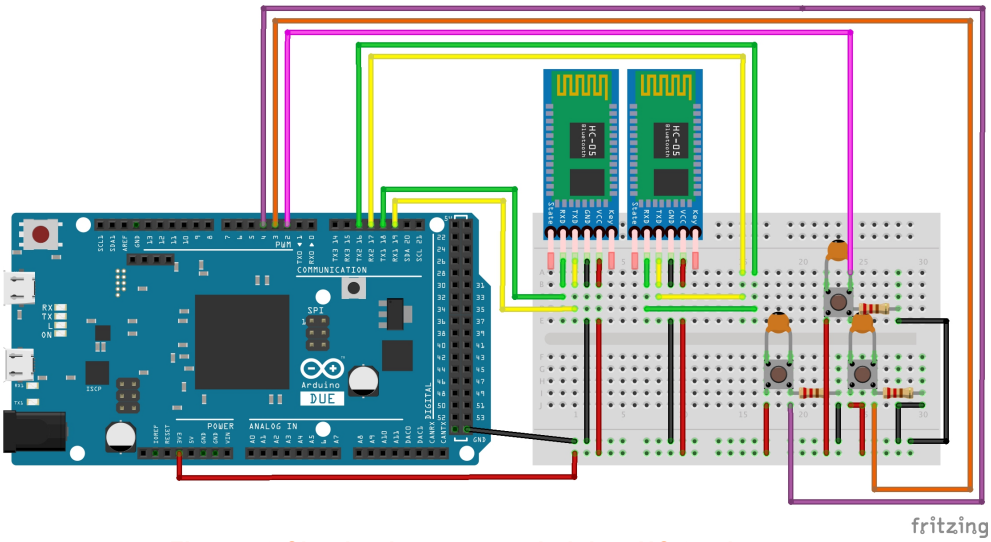

Figura 50 Circuito de receptor - Arduino, HC-05, botones

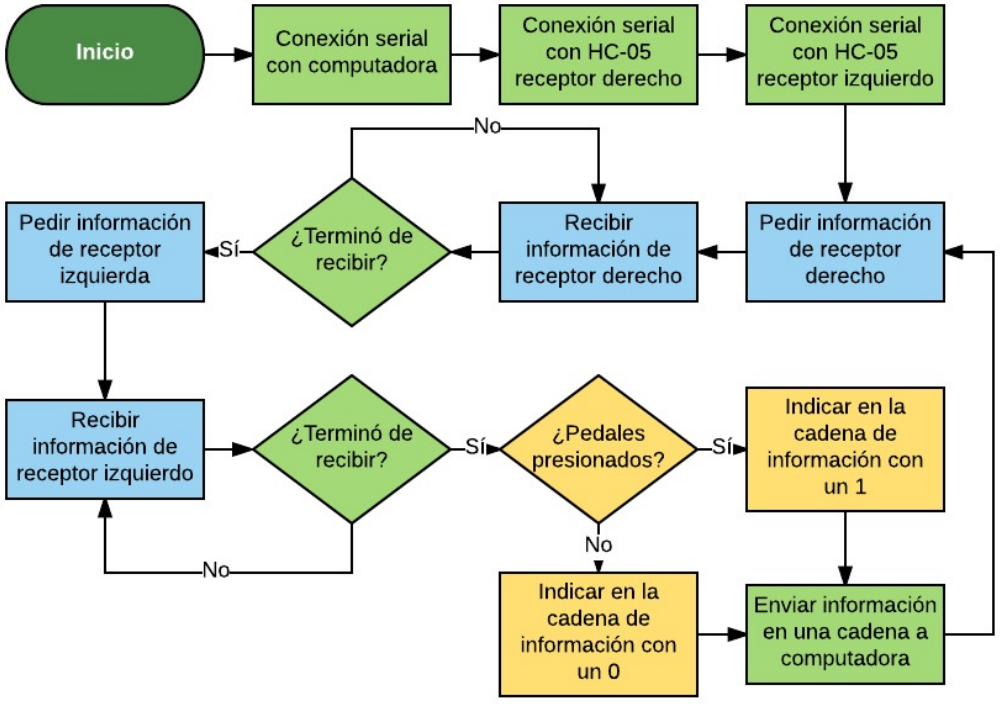

Figura 51 Algoritmo para recepción de datos de baquetas y pedales

Los dispositivos generados para los pedales se muestran en la Figura 52.

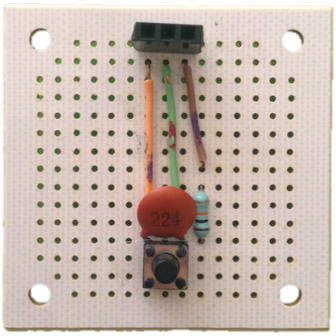

Figura 52 Emulador de pedales

Con la información de las baquetas y los pedales disponible en un puerto de la computadora, debe poder ser usada por Processing para juntarla con la información obtenida de Kinect.

Para acceder a la información del receptor, es necesario instalar la librería Serial, la cual está hecha para poder leer y escribir información con dispositivos externos [75], con el receptor en este caso.

La información obtenida, se agrupará con la información obtenida de Kinect en una sola cadena de datos. Los primeros términos pertenecerán a los obtenidos de Kinect, seguidos de los obtenidos de las baquetas, separados por una coma. La cadena será la siguiente:

$$X_D, Y_D, Z_D, X_I, Y_I, Z_I, q_{wD}, q_{xD}, q_{yD}, q_{zD}, a_{xD}, a_{yD}, a_{zD}, q_{wI}, q_{xI}, q_{yI}, q_{zI}, a_{xI}, a_{yI}, a_{zI}, B_1, B_2, H$$

Una vez instalada la librería, se obtienen los datos y se muestran en la pantalla usando el algoritmo mostrado en la Figura 53, el cual usa el algoritmo utilizado con Kinect mencionada anteriormente.

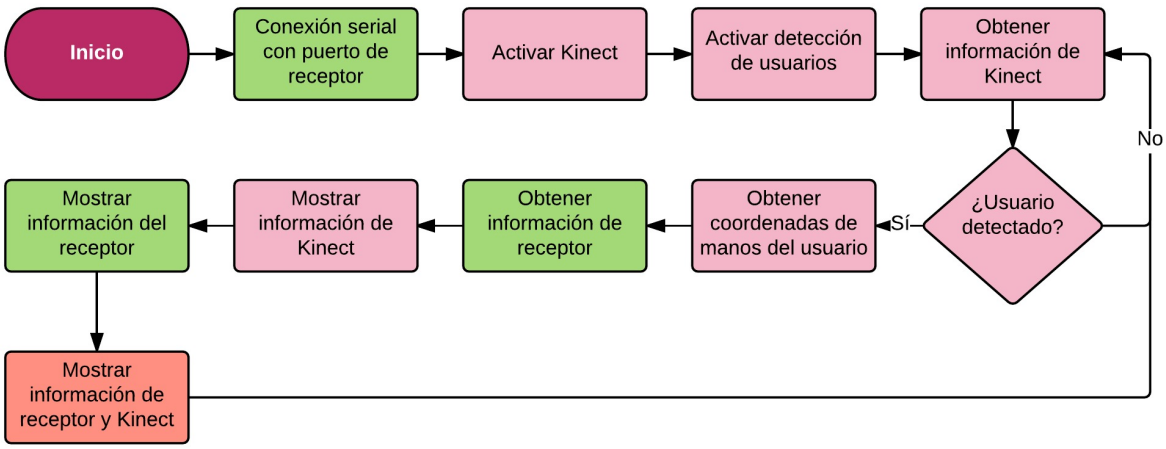

Figura 53 Algoritmo para obtener información de receptor en Processing

En la Figura 54 se muestra la implementación de dicho algoritmo en Processing. En rojo se muestra la información obtenida de Kinect, en verde la información obtenida del receptor, y en naranja se muestra la información concatenada.

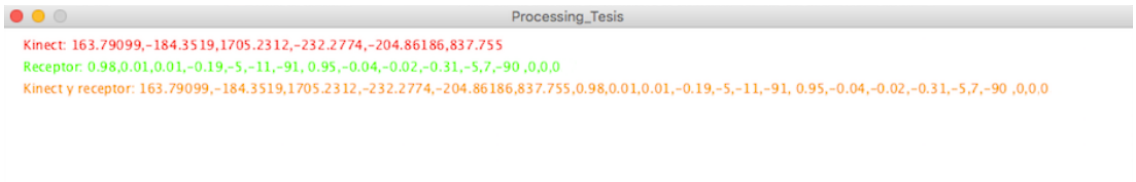

Figura 54 Información de Kinect y receptor en Processing

INTERNET DE LAS COSAS

Una vez que se cuenta con toda la información de la posición y orientación espacial de las baquetas en Processing, debe ser enviada por algún medio a Unity para su uso en el entorno virtual. Para realizar dicho envío se usarán las herramientas de Spacebrew [46].

Spacebrew realiza la transmisión de información, de un punto a otro haciendo uso de algún servidor público o local en el cual se pueda correr su plataforma, mediante conexiones entre los puntos deseados. En el caso de usar un servidor local, la plataforma debe correr desde la terminal de la computadora.

En la Figura 55 se muestra la plataforma de Spacebrew.

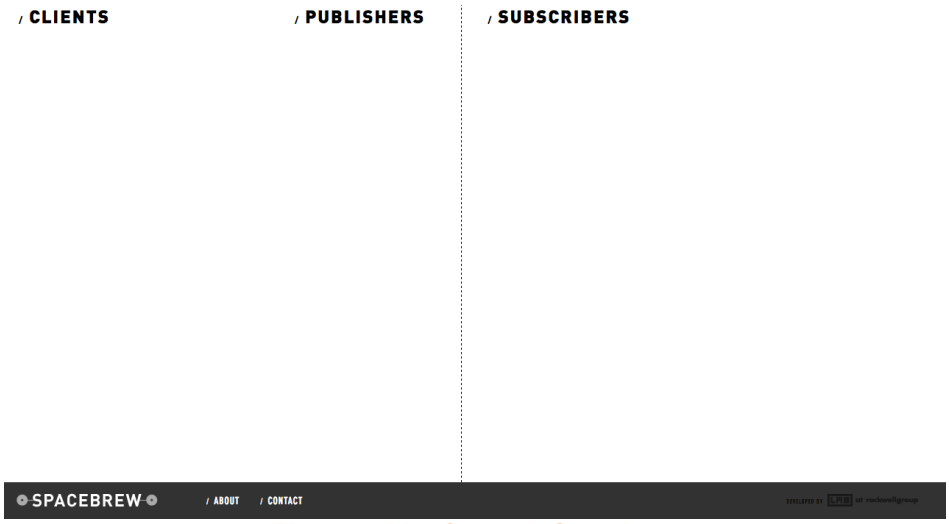

Figura 55 Plataforma de Spacebrew

Spacebrew sólo permite el envío de información de tres tipos:

- Cadena de caracteres
- Booleanos
- Enteros

La plataforma requiere de cuatro elementos para poder realizar la transmisión de información:

- Clientes: Dispositivos y/o software que envían y/o reciben información
- Editores: Objeto del cual es enviada la información del cliente al que pertenece
- Suscriptores: Objeto que recibe la información de algún editor y la proporciona al cliente al que pertenece
- Conexiones: Para poder transmitir información entre un editor y un suscriptor, se deben realizar conexiones manuales entre los editores y suscriptores deseados

Un editor puede enviar información a mas de un suscriptor de diferentes clientes, y un cliente puede tener mas de un editor y suscriptor, dependiendo del tipo de información que se envíe y/o reciba.

En este trabajo habrá dos clientes: Processing y Unity. Ambos tendrán sus respectivos editores y suscriptores necesarios.

Para que Processing logre realizar la conexión con el servidor donde se esté corriendo la plataforma de Spacebrew y crear sus respectivos editores y suscriptores se necesita instalar la librería de Spacebrew [46].

Una vez instalada la librería, se necesita de lo siguiente para que Processing pueda usar la plataforma de Spacebrew:

- Para la conexión como cliente:
    - Dirección de servidor donde se encuentra la plataforma de Spacebrew
    - Nombre de cliente
    - Descripción de cliente
- Para la generación de editores:
    - Nombre de editores
    - Tipo de información a enviar de los editores
    - Información que enviarán los editores
- Para la generación de suscriptores:
    - Nombre de los suscriptores
    - Tipo de información a recibir de los suscriptores

Antes de enviar la información de Processing a Unity, se enviará la información de Processing a Processing para comprobar que la transferencia de información se realiza de forma adecuada.

Processing se conectará al servidor como cliente y se generará un editor para enviar la información de la posición y orientación espacial de las baquetas, y un suscriptor para recibirla y después mostrarla. El algoritmo que se utilizará, utilizando el algoritmo de la Figura 53, se muestra en la Figura 56.

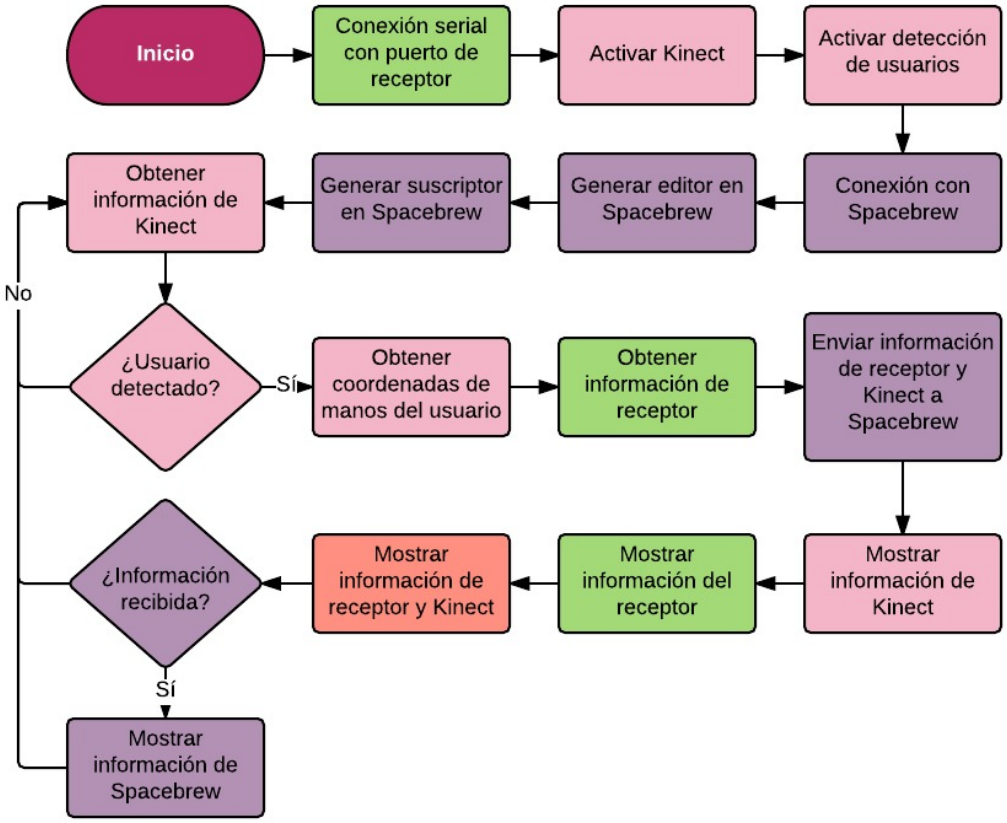

Figura 56 Algoritmo para envío y recepción de información de Spacebrew

Una vez implementado el código en Processing, se generan todo lo necesario para realizar la transferencia de información, pero, para realizarla, se debe realizar la conexión entre editor y suscriptor de forma manual. En la Figura 57 se muestran los elementos generados por Processing en Spacebrew sin conexiones.

Figura 57 Cliente de Processing en Spacebrew sin conexiones

En la Figura 58 se muestran los elementos con la conexión hecha.

Figura 58 Cliente de Processing en Spacebrew con conexiones

Establecida la conexión, Processing recibirá la información que envíe. En la Figura 59 se muestra en rojo la información obtenida de Kinect, en verde la información obtenida del receptor, en naranja la información concatenada, y en morado la información recibida de Spacebrew.

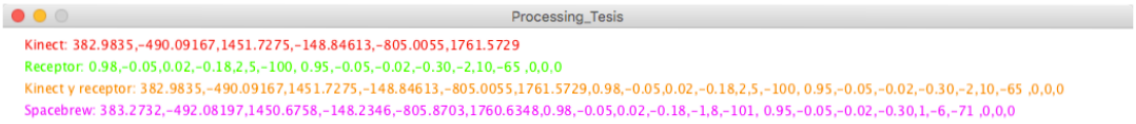

Figura 59 Información de Kinect, receptor, y Spacebrew en Processing

Comprobada la transferencia de información, con el uso de Spacebrew, de Processing a Processing, la información se enviará a Unity que, al igual que Processing, requiere de las herramientas de Spacebrew para poder conectarse al servidor [77].

El paquete de herramientas de Spacebrew en Unity, contiene un objeto que contiene a su vez dos scripts:

- SpacebrewClient: Realiza conexión con servidor, crea editores y suscriptores deseados, y contiene métodos para envío y recepción de información.
- SpacebrewEvent: Con el uso de los métodos para recibir información de SpacebrewClient, maneja la información recibida para su uso en Unity.

Un script es un componente que permite activar o desactivar eventos, modificar propiedades de los objetos a los que pertenecen, obtener información de usuarios, entre otras cosas [78].

SpacebrewClient requiere de los mismos elementos que Processing para realizar la conexión con el servidor donde se corre la plataforma de Spacebrew, generar editores, y generar suscriptores:

- Para la conexión como cliente:
  - Dirección de servidor donde se encuentra la plataforma de Spacebrew
  - Nombre de cliente
  - Descripción de cliente
- Para la generación de editores:
  - Nombre de editores
  - Tipo de información a enviar de los editores
  - Información que enviarán los editores
- Para la generación de suscriptores:
  - Nombre de los suscriptores
  - Tipo de información a recibir de los suscriptores

Con el objeto importado en la escena de Unity se genera todo lo necesario para poder obtener la información de Spacebrew. En la Figura 60 se muestran los elementos generados sin conexiones.

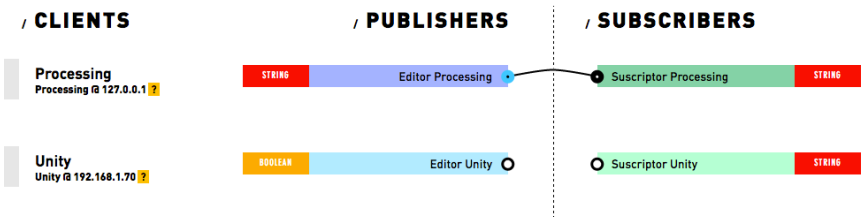

Figura 60 Clientes de Unity y Processing sin conexiones entre ellos

Se realiza la conexión entre el editor de Processing y el suscriptor de Unity para poder recibir la cadena de caracteres con la información de la posición y orientación espacial de las baquetas, e información de los pedales. Dicha conexión se muestra en la Figura 61.

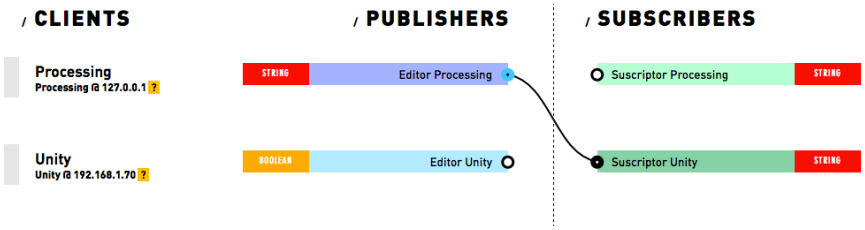

Figura 61 Clientes de Unity y Processing con conexiones entre ellos

La información recibida aún no puede usarse. Se creará un nuevo objeto de nombre *_Data* y se añadirá un nuevo script de nombre *DataManager*, para obtener la información de SpacebrewEvent.

DataManager se encargará de obtener la cadena de caracteres recibida en SpacebrewEvent para poder obtener los distintos elementos que componen la cadena, gracias a la separación por coma que existe entre cada elemento de la cadena, y guardarlos de forma que puedan ser usados en Unity. Los elementos se guardarán en los siguientes arreglos:

- Coordenadas de baqueta derecha
- Coordenadas de baqueta izquierda
- Cuaternion de baqueta derecha
- Aceleración angular de baqueta derecha
- Cuaternion de baqueta izquierda
- Aceleración angular de baqueta izquierda
- Pedales

El algoritmo de DataManager se muestra en la Figura 62.

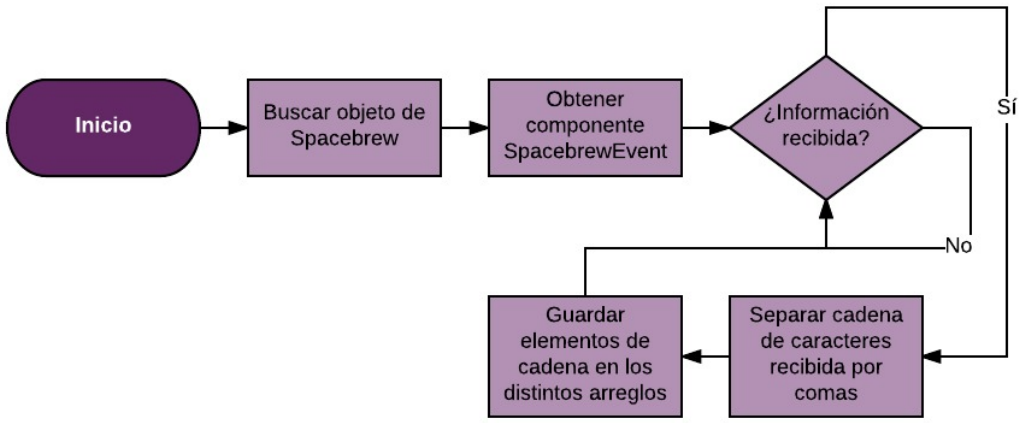

Figura 62 Algoritmo para obtención de información recibida de Spacebrew

En la Figura 63 se muestran algunos elementos obtenidos de Spacebrew por el objeto _Data.

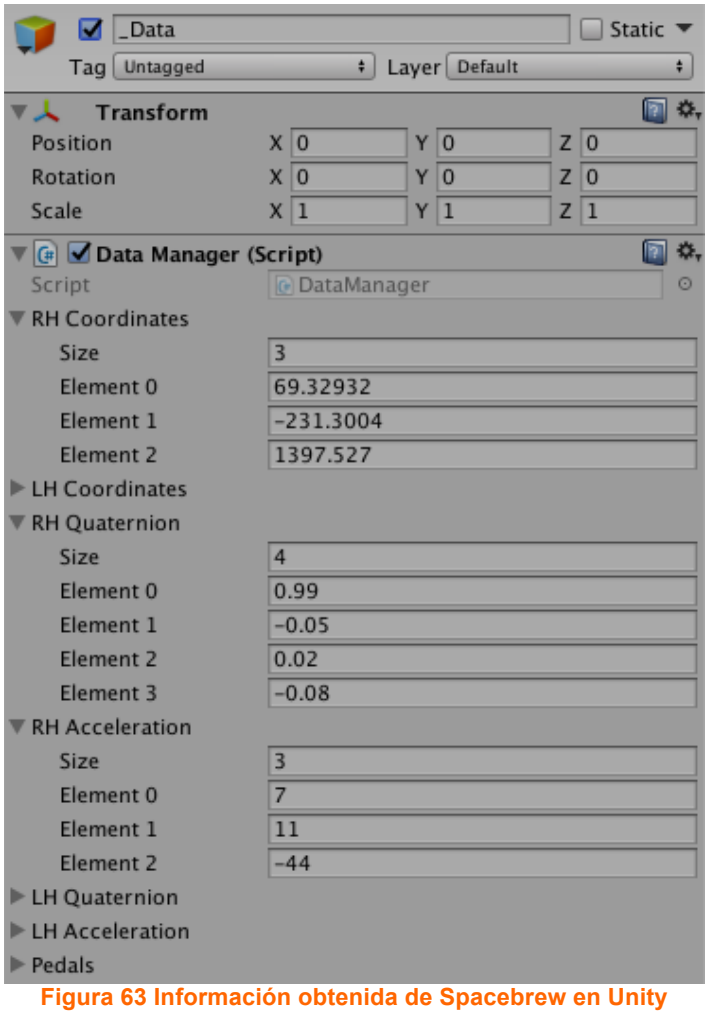

Figura 63 Información obtenida de Spacebrew en Unity

ENTORNO VIRTUAL

MODELOS VIRTUALES

La información del usuario ya se encuentra en Unity lista para usarse. Para la elaboración de los modelos virtuales de los elementos de la batería se usará un tom proporcionado por Moisés Cabrera. Dicho tom, de 15” x 16”, se muestra en la Figura 64.

Figura 64 Tom de piso 15" x 16"

Para los modelos de los platillos se usará el perfil mostrado en la Figura 65.

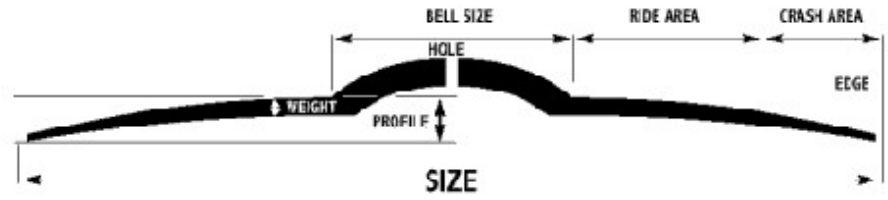

Figura 65 Pefil de platillo [79]

Los elementos de los cuales se realizarán modelos virtuales son:

- Tarola 6” x 14”
- Toms de aire:
  - 8” x 8”
  - 9” x 10”
  - 10” x 12”
- Tom de piso 15” x 16”
- Bombo 18” x 22”
- Platillos:
  - 8”
  - 14”
  - 18”
  - 20”
  - 22”

Para el modelo virtual de las baquetas se usaron baquetas 5A, mostradas en la Figura 66.

Figura 66 Baquetas 5A [82]

En las Figuras 67 a 78 se muestran los elementos generados en Fusion 360. Los colores fueron elegidos al azar para poder identificarlos.

Figura 67 Modelo virtual de baqueta 5A

Figura 68 Modelo virtual de tarola 6" x 14"

Figura 69 Modelo virtual de tom 8" x 8"

Figura 70 Modelo virtual de tom 9" x 10"

Figura 71 Modelo virtual de tom 10" x 12"

Figura 72 Modelo virtual de tom 15" x 16"

Figura 73 Modelo virtual de bombo 18" x 22"

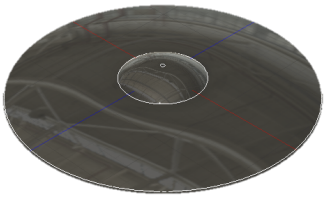

Figura 74 Modelo virtual de platillo de 8"

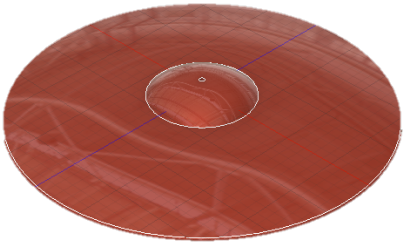

Figura 75 Modelo virtual de platillo de 14"

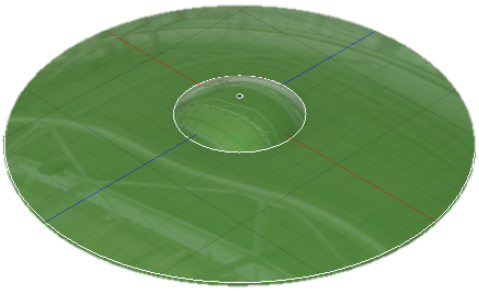

Figura 76 Modelo virtual de platillo de 18"

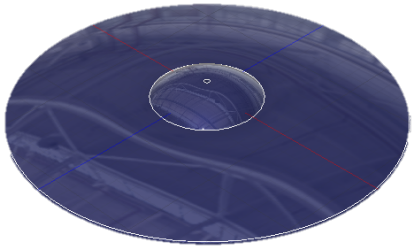

Figura 77 Modelo virtual de platillo de 20"

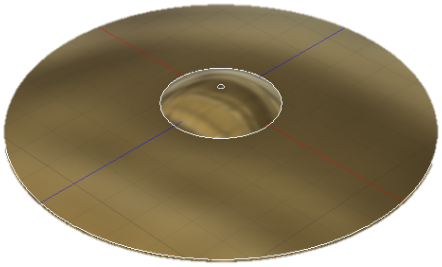

Figura 78 Modelo virtual de platillo de 22"

Para poder utilizar los modelos realizados en Unity, se requiere de archivos obj de los modelos virtuales. Fusion 360 ofrece la obtención de dichos archivos, pero al momento de realizar el trabajo no se contó con dicha función por lo que se usará Blender, software de modelado 3D, para obtener los archivos.

Una vez obtenidos los archivos obj, se importan en la escena de Unity, se ubican y orientan en el espacio, y se les asigna un color como se muestra en la Tabla 6. Las unidades de las magnitudes de posición espacial son unidades de Unity, y las unidades de orientación espacial son grados. El origen de los tambores se encuentra en la parte inferior al cetro, y el de platillos se encuentra al centro del mismo en la parte inferior de la campana.

Tabla 6 Posición y orientación espacial de elementos virtuales en Unity

ELEMENTO	X	Y	Z	ROLL	PITCH	YAW	COLOR
Tarola	0	53	-2	0	0	0	Rojo
Tom 8x8	-14	70	42	-43	0	0	Rosa
Tom 9x10	18	68.2	39	-43	0	12	Amarillo
Tom 10x12	51	68	30	-43	0	20	Azul
Tom 15x16	55	27	-20	0	0	0	Verde
Bombo Der	33	30	24	90	0	-17	Negro
Bombo Izq	-33	30	24	90	0	17	Negro

Platillo 8	0	100	40	0	0	0	Blanco
Platillo 18	80	80	-40	0	0	0	Morado
Platillo 20	55	120	48	-15	0	5	Naranja
Platillo 22	-50	120	55	-15	0	-5	Naranja
Hi Hat Sup	-45	85	5	0	0	0	Gris
Hi Hat Inf	-45	85	-10.5	180	0	0	Gris
Baqueta D	20	0	0	0	0	0	Rosa
Baqueta I	-20	0	0	0	0	0	Verde

Los valores de la Tabla 6 fueron escogidos con base en la posición y orientación espacial de los elementos de un kit de batería real de forma que fuera lo mas semejante posible. En la Figura 79 se muestra dicho instrumento.

Figura 79 Kit de batería

Todos los elementos se agruparon dentro de un objeto, excepto las baquetas que son tratados como objetos individuales, llamado _DrumSet. En la Figura 80 se muestran dichos objetos.

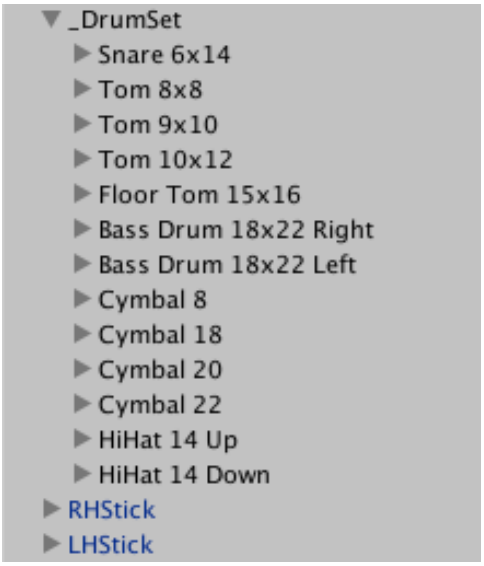

Figura 80 Objetos _DrumSet, y baquetas

En las Figuras 81 a 83 se muestran distintas vistas de los elementos ubicados en Unity.

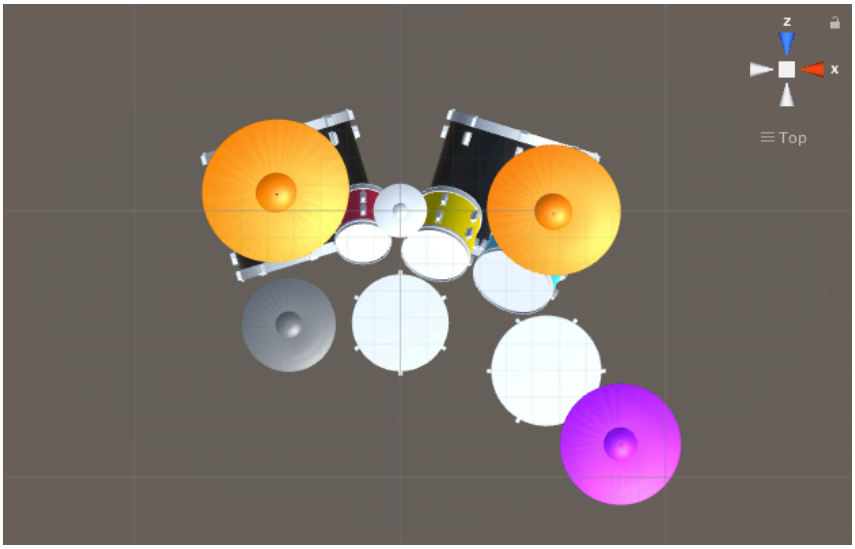

Figura 81 Vista superior de batería virtual

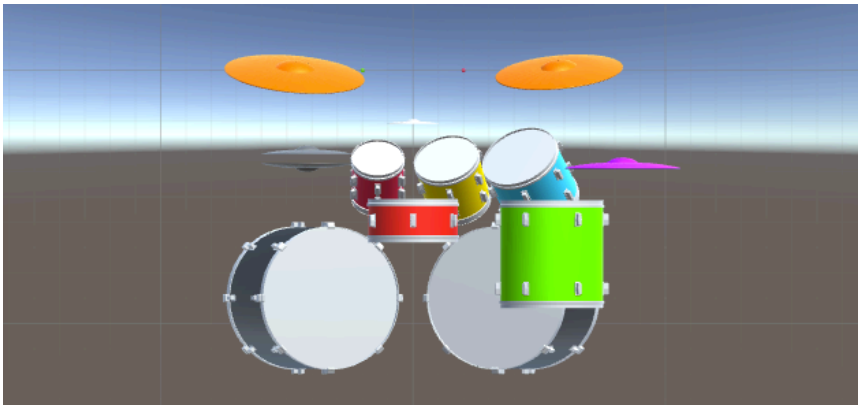

Figura 82 Vista frontal de batería virtual

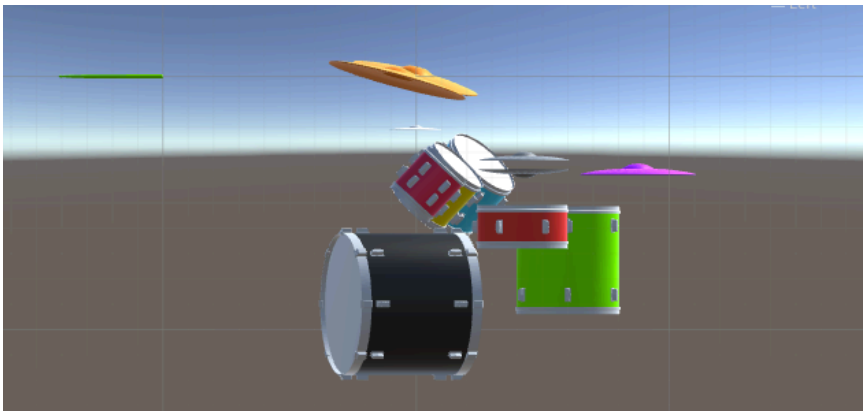

Figura 83 Vista lateral de batería virtual

Todos los elementos de Unity se escalaron tal que cada unidad del entorno de Unity sea equivalente a un centímetro real.

POSICION Y ORIENTACION ESPACIAL DE BAQUETAS VIRTUALES

Las baquetas deben de cambiar de posición y orientación en función de la información de la posición y orientación espacial de las baquetas obtenida de Spacebrew. Para realizar dicho comportamiento, se añade un script a cada modelo de baqueta en el cual obtiene la información del objeto _Data y modifica la posición y orientación del modelo virtual de las baquetas.

Como se mencionó anteriormente, cada unidad de Unity representa un centímetro real. Las coordenadas obtenidas por Kinect se encuentran en milímetros por lo que hay que dividir las coordenadas entre diez para lograr movimientos adecuados dentro del entorno.

Además, Kinect debe de colocarse a aproximadamente a 195 [cm] frontal del usuario y aproximadamente a la altura del pecho, como se muestra en la Figura 84, para que los movimientos en el entorno sean adecuados.

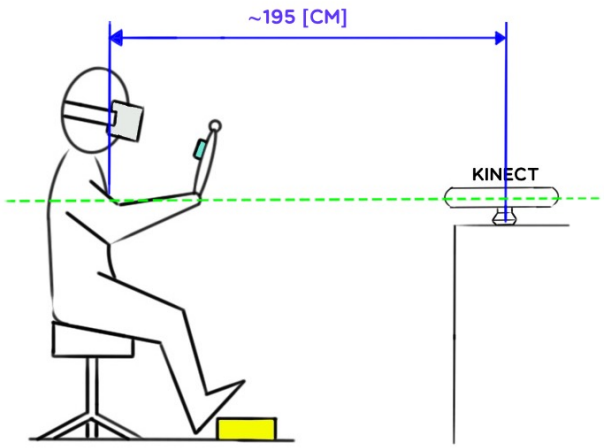

Figura 84 Ubicación de Kinect con respecto al usuario

Una desventaja de usar un MPU6050 sin magnetómetro para la obtención de la orientación espacial de las baquetas es un desfase de ángulo en el eje z. Dicho desfase no permite que los movimientos de la baqueta virtual sean iguales a los de la real. Para eliminar el desfase, se utilizará algebra de cuaterniones.

Un cuaternión puede ser interpretado como una rotación S sobre un vector $\bar{V}$. La ecuación 1 puede ser escrita de las siguientes formas:

$$Q = S + V_x\hat{\imath} + V_y\hat{\jmath} + V_z\hat{k} \qquad (6)$$

$$Q = [S, \bar{V}] \qquad (7)$$

Donde:

$$S = \cos\left(\frac{\theta}{2}\right) \quad (8)$$

$$\bar{V} = \sin\left(\frac{\theta}{2}\right) \quad (9)$$

Cuando se corre el proyecto en Unity, antes de recibir información de Spacebrew, la baqueta se encontrará en una posición inicial P. En el momento que se reciba la información de Spacebrew, y el script de las baquetas obtenga dicha información, la baqueta será rotada, sobre un cuaternión Q_D, a una posición P'. Dicha rotación se representa con la ecuación 10.

$$Q_D * P = P' \quad (10)$$

Para eliminar el desfase es necesario premultiplicar la ecuación 10 por el inverso de Q_D y así lograr obtener la ecuación 11 con la cual se representa la posición real de las baquetas.

$$Q_D^{-1} * Q_D * P = Q_D^{-1} * P'$$

$$I * P = Q_D^{-1} * P$$

$$P = Q_D^{-1} * P' \quad (11)$$

Para obtener el inverso de un cuaternión se utiliza la ecuación 12.

$$Q^{-1} = \frac{Q^*}{|Q|^2} \quad (12)$$

Si el cuaternión es unitario, es decir de magnitud 1, como es el caso de los cuaterniones brindados por el MPU6050, la ecuación 12 se reduce a la ecuación 13.

$$Q^{-1} = Q^* \quad (13)$$

El cuaternión Q_D cambia cada que los MPU6050 son reiniciados, por lo que es necesario obtener un Q_D cada que se corra el proyecto en Unity.

La obtención de Q_D se realizará con el hardware de las baquetas colocados sobre una superficie plana y sin moverse. Al correr el proyecto se ignoraran las primeras catorce lecturas, la lectura quince será el Q_D. Con el cuaternión obtenido se aplicará la ecuación 13 para rotar las baquetas.

El algoritmo de los scripts agregados en las baquetas se muestra en la Figura 85.

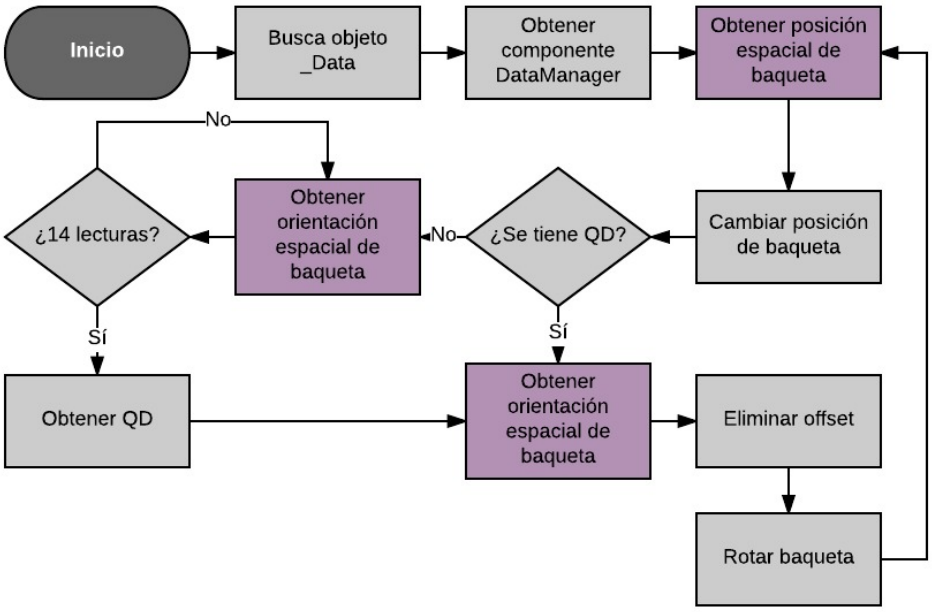

Figura 85 Algortimo de script en baquetas

La implementación del algoritmo se muestra en la Figura 86.

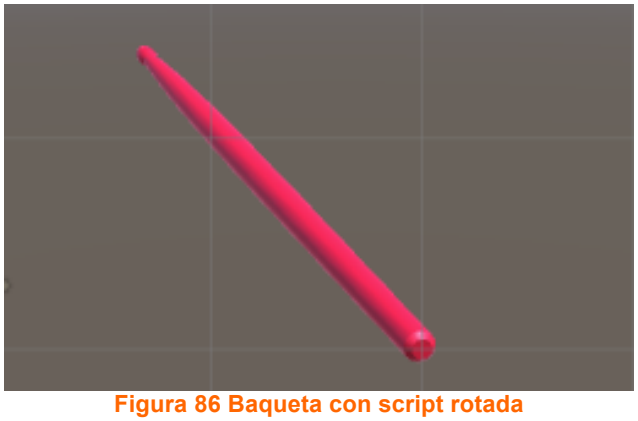

Figura 86 Baqueta con script rotada

REPRODUCCION DE SONIDOS

Ahora que las baquetas se mueven en función de los movimientos del usuario, aunque atraviesen algún elemento de la batería virtual, no producirá ningún sonido. Para esto es necesario importar los sonidos de cada elemento e indicar, por medio de rigidbodies y colliders, que zonas de cada elemento producirán determinado sonido.

Un rigidbody es un componente que permite el comportamiento físico para un objeto [80]. Es decir, el objeto podrá ser sometido a fuerzas, como la gravedad, tener masa, entre otras cosas.

Un collider es un componente invisible que define la forma de un objeto para los propósitos de colisiones físicas [81]. En este trabajo se usará con el fin de saber si otro objeto ha entrado en el collider para activar una acción, en este caso reproducir un sonido. Para poder conocer si un objeto ha entrado en el collider, el objeto, las baquetas, deben contar con un rigidbody.

Los sonidos que se utilizarán serán sonidos MIDI, los cuales se obtendrán con TuxGuitar, software para escritura y reproducción de música [84], para cada elemento virtual de la batería además de un sonido que emule colisión con la parte externa de algún tambor y sonidos para las campanas de los platillos, con excepción del hi-hat.

Los sonidos obtenidos se importan en Unity y se agrupan en un objeto llamado _AudioManager. Dicho objeto se muestra en la Figura 87.

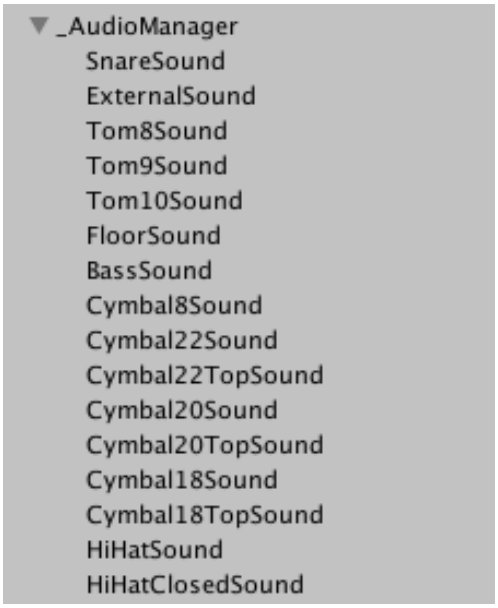

Figura 87 Objeto _AudioManager

Los collider de las baquetas se pondrán manualmente, debido a la sencillez de su función, sólo de atravesar elementos, y geometría alargada. El collider usado es una caja y se muestra en la Figura 88. Además estos son los elementos que tendrán un rigidbody para que los colliders de la batería sepan que algo ha entrado.

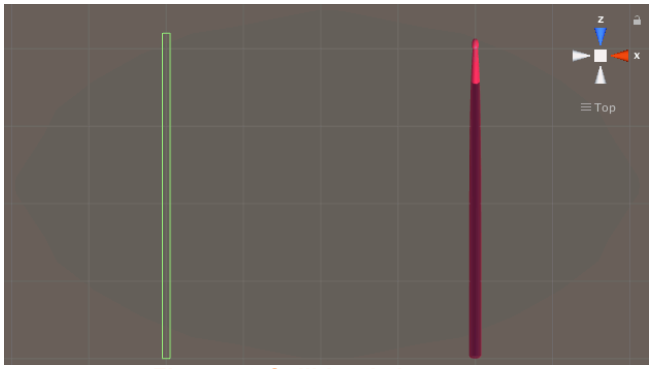

Figura 88 Collider de baquetas

En el caso de los bombos no es necesario generar colliders, ya que la información para reproducir su sonido es recibida de Spacebrew, por lo que basta con obtener la información de DataManager en _Data y reproducir su sonido ubicado en _AudioManager. El algoritmo del script para bombo se muestra en la Figura 89.

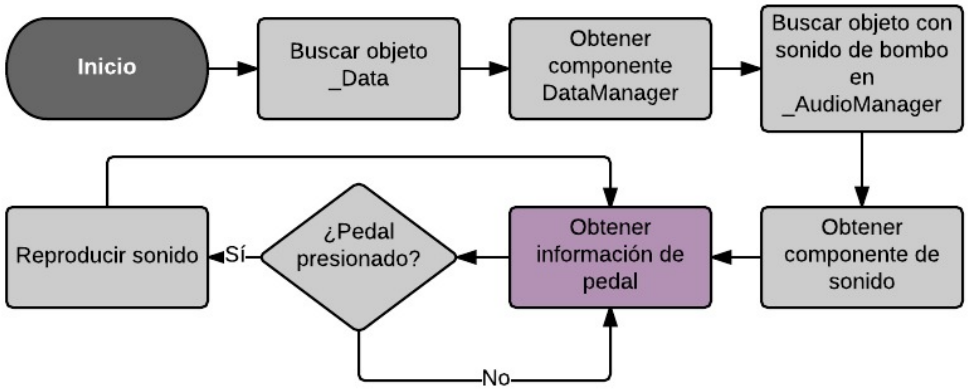

Figura 89 Algoritmo para reproducción de sonidos de bombo

Los platillos y tambores tienen una geometría que no puede ser cubierta directamente con las geometrías proporcionadas por Unity para la generación de colliders. Para solucionar esto, se usarán varios colliders en forma de caja, rotados y con el ancho suficiente para poder cubrir el volumen de los elementos.

No es suficiente contar con un solo arreglo de colliders por elemento, ya que el sonido del elemento no debe ser reproducido si se realiza una colisión por la parte inferior o laterales del elemento o, en el caso de los platillos, si se realiza una colisión en la campana.

Para la generación de los colliders se utilizarán dos scripts diferentes, uno para tambores y otro para platillos, debido a la diferencia de sonidos y geometrías. Para esto el código realizado por Kode 80 para la generación de colliders será de gran ayuda [86].

En el caso de los tambores se pondrá un arreglo de colliders que cubra el volumen del elemento, uno que cubra la parte inferior, y otro que cubra solamente la parte externa del elemento sin cruzar con el arreglo del volumen interno. Cada colisión reproducirá un sonido diferente, del elemento o externo, dependiendo de los arreglos en los cuales exista la colisión, con base en la Tabla 7 en donde un 1 indica que hubo una colisión y un 0 indica lo contrario.

Tabla 7 Sonidos reproducidos por tambores en función de colisiones en sus arreglos de colliders

Arreglo Inferior	Arreglo Externo	Arreglo Interno	Sonido
0	0	0	-
0	0	1	Elemento
0	1	0	Externo
0	1	1	Elemento
1	0	0	-
1	0	1	-
1	1	0	Externo
1	1	1	-

Además, para poder observar que los colliders estén posicionados de forma correcta, debido a que se crearán hasta que se corra el proyecto, se generan mallas con la forma y tamaño de los colliders que se crearán, y que desaparecerán cuando se corra el proyecto. Dicha función se utilizará también en el script de los platillos.

El algoritmo del script para tambores se muestra en las Figuras 90 y 91.

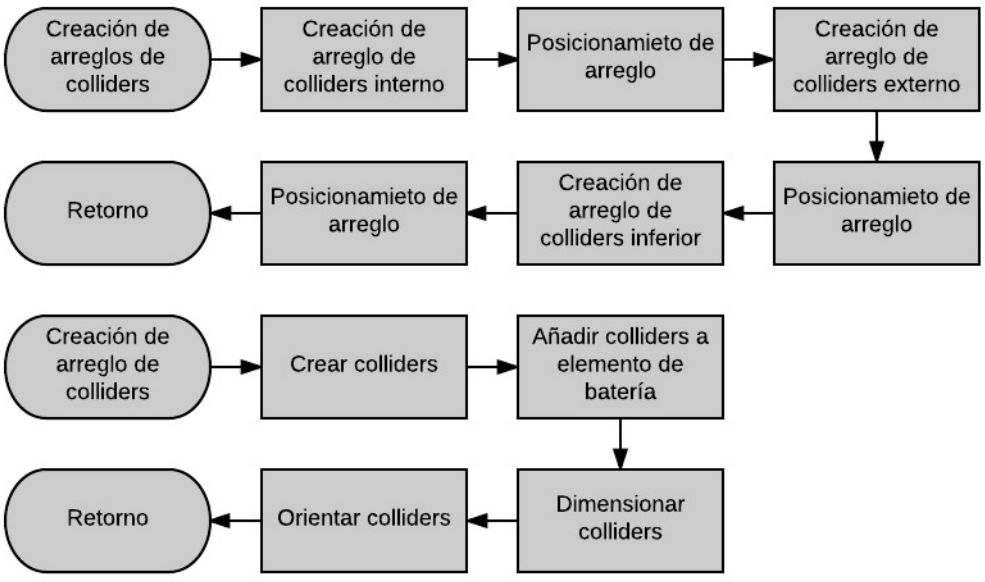

Figura 90 Algoritmo para script de tambores parte 1

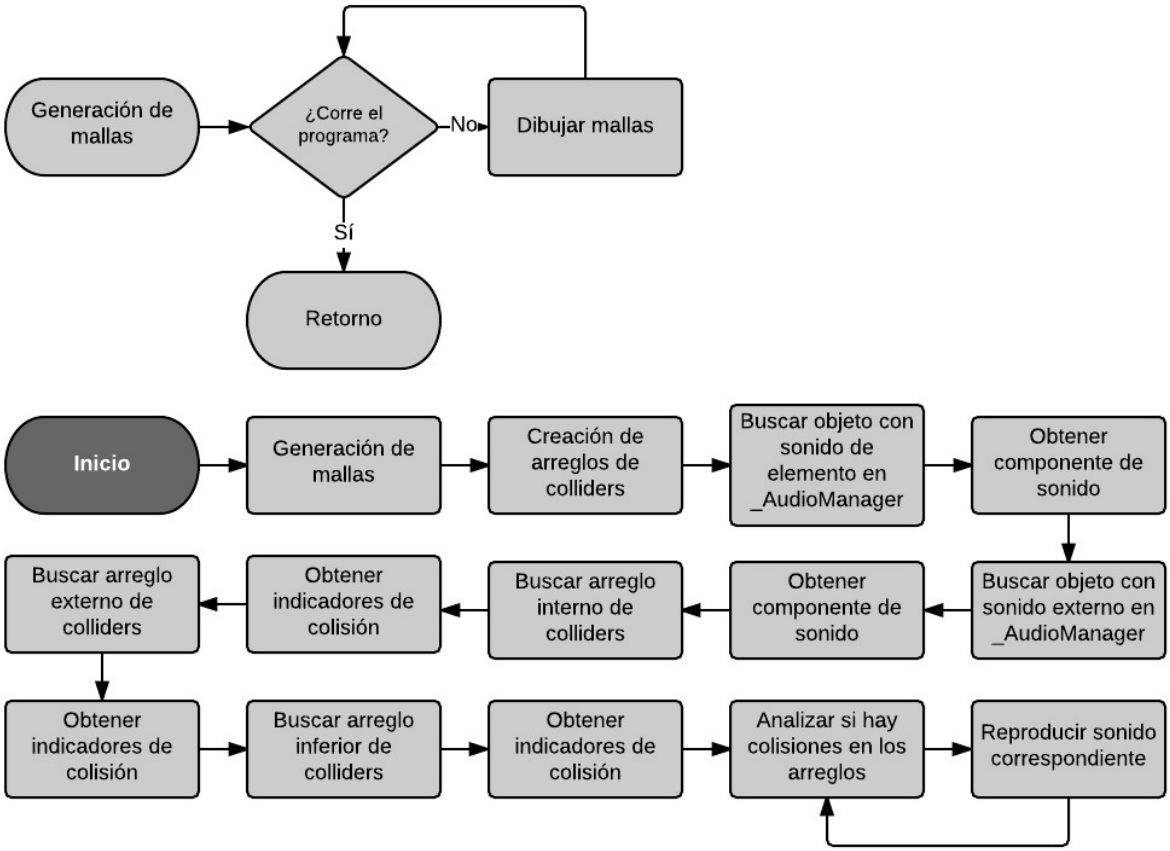

Figura 91 Algoritmo para script de tambores parte 2

Las dimensiones de los colliders, número de colliders, y demás elementos necesarios para su creación se ingresan desde la interfaz de Unity. En la Figura 92 se muestran en la interfaz los elementos del script de la tarola.

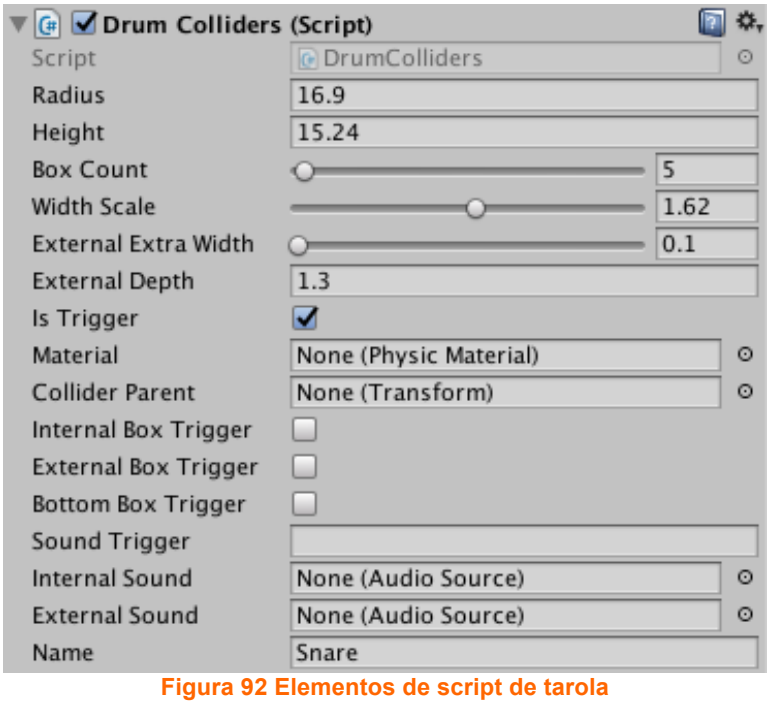

Figura 92 Elementos de script de tarola

En la Figura 93 y 94 se muestra un elemento de batería con sus mallas generadas por el script.

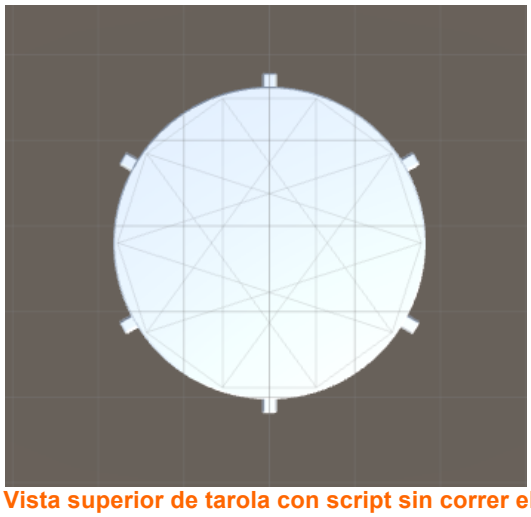

Figura 93 Vista superior de tarola con script sin correr el proyecto

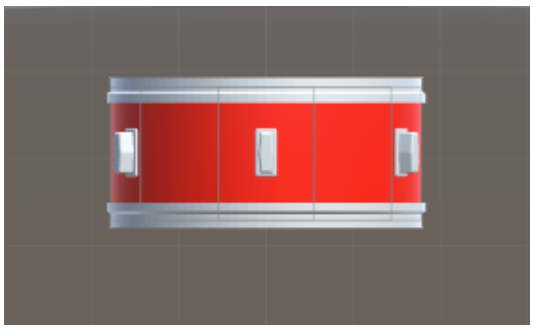

Figura 94 Vista lateral de tarola con script sin correr el proyecto

En las Figuras 95 y 96 se muestra el componente de las figuras anteriores, con el programa corriendo y con los colliders generados.

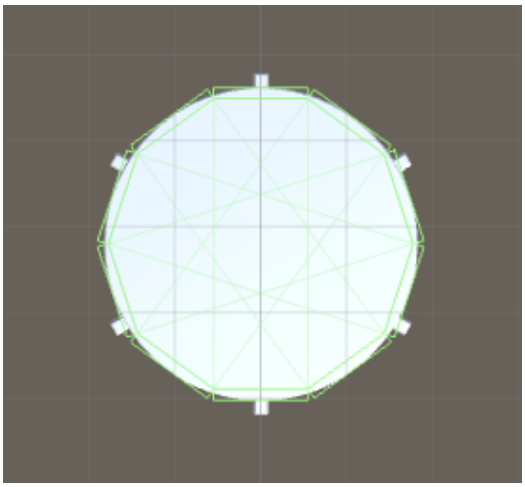

Figura 95 Vista superior de tarola con colliders

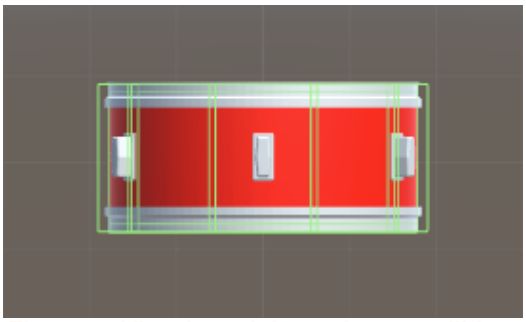

Figura 96 Vista lateral de tarola con colliders

En la Figura 97 se muestran los colliders generados para la tarola dentro de un objeto llamado *SnareTriggerColliders*.

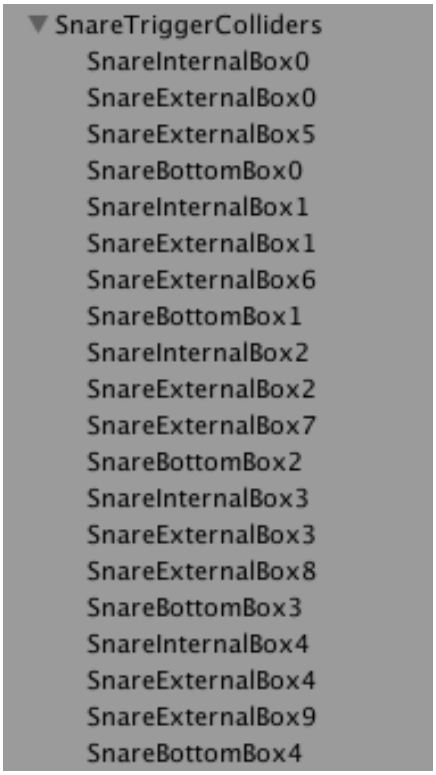

Figura 97 Objeto con colliders generados para la tarola

En el caso de los platillos se pondrá un arreglo de colliders que cubra el volumen del elemento, excepto su campana que tendrá su propio arreglo, y otro que cubra la parte inferior. Cada colisión reproducirá un sonido diferente dependiendo de los arreglos en los cuales exista la colisión, y del elemento ya que ni el platillo de 8", ni el Hi-hat cuentan con sonido de campana, pero, el hi-hat, sí cambia su sonido dependiendo si está, o no, presionado su pedal. La Tabla 8 indica que sonidos generan las colisiones dependiendo si hubo una colisión o no, indicando con un 1 que la hubo y con un 0 que no.

Tabla 8 Sonidos reproducidos por platillos en función de colisiones en sus arreglos de colliders

Arreglo Inferior	Arreglo superior	Arreglo Interno	Sonido
0	0	0	-
0	0	1	Elemento
0	1	0	Campana
0	1	1	Campana
1	0	0	-
1	0	1	-
1	1	0	-
1	1	1	-

El Hi-hat tiene tres casos diferentes de la Tabla 8. Dichos casos y condiciones se muestran en la Tabla 9 donde, en los arreglos, un 1 representa que hubo una colisión y un 0 que no, y en el pedal, un 1 indica que está presionado y un 0 que no lo está.

Tabla 9 Sonidos reproducidos por Hi-hat en función de su pedal y colisiones

Pedal	Arreglo Inf	Arreglo Sup	Arreglo Int	Sonido
0	0	0	1	Hi-hat arriba
1	0	0	1	Hi-hat abajo
0	0	1	0	Hi-hat arriba
1	0	1	0	Hi-hat abajo
0	0	1	1	Hi-hat arriba
1	0	1	1	Hi-hat abajo

El algoritmo del script para platillos es muy similar al de los tambores. En la Figura 98 se muestra la sección del algoritmo modificado.

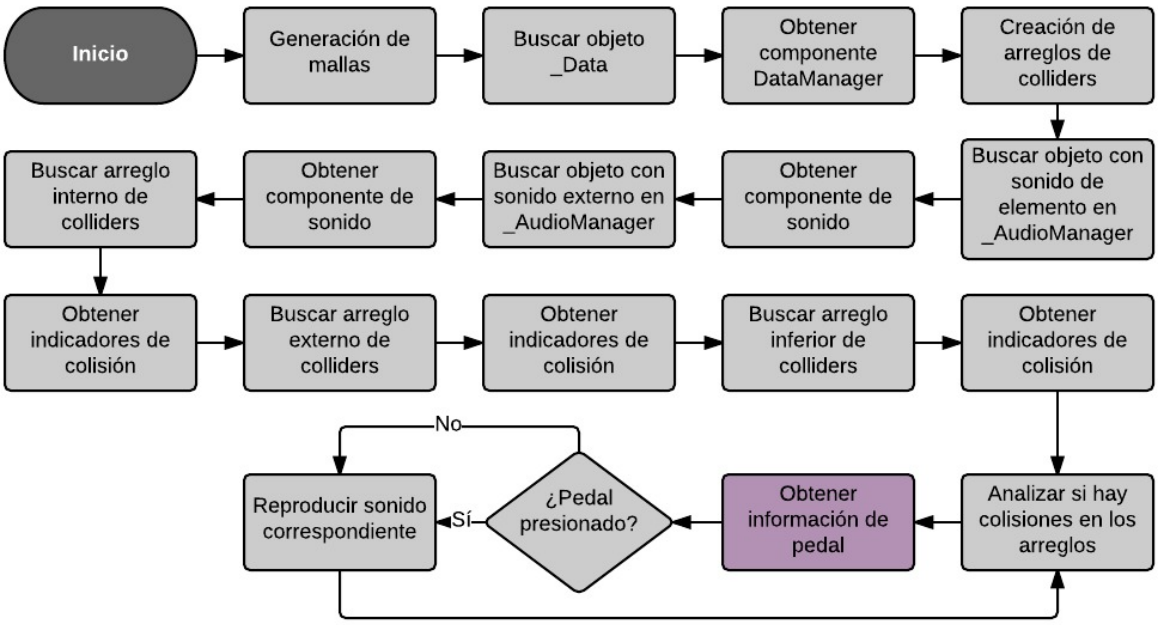

Figura 98 Algoritmo para script de platillos

Las dimensiones de los colliders, número de colliders, y demás elementos necesarios para su creación se ingresan desde la interfaz de Unity. En la Figura 99 se muestran en la interfaz los elementos del script del platillo de 22".

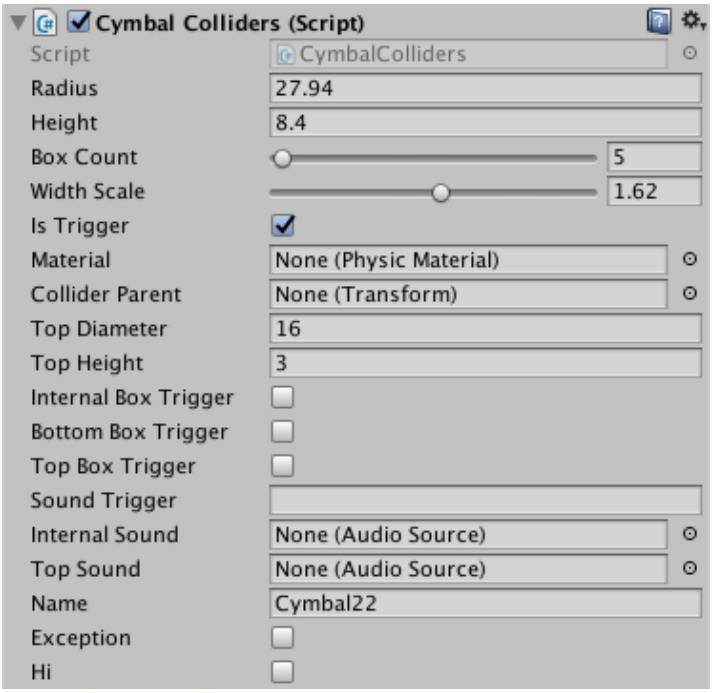

Figura 99 Elementos de script de platillo de 22"

En las Figuras 100 y 101 se muestra un platillo con sus mallas generadas por el script.

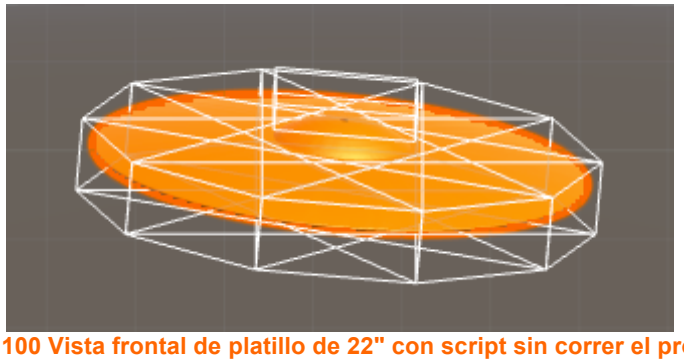

Figura 100 Vista frontal de platillo de 22" con script sin correr el proyecto

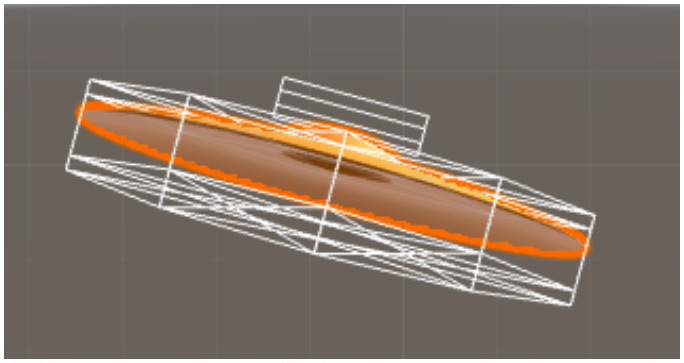

Figura 101 Vista lateral de platillo de 22" con script sin correr el proyecto

En la Figuras 102 y 103 se muestra el componente de las figuras anteriores, con el programa corriendo y con los colliders generados.

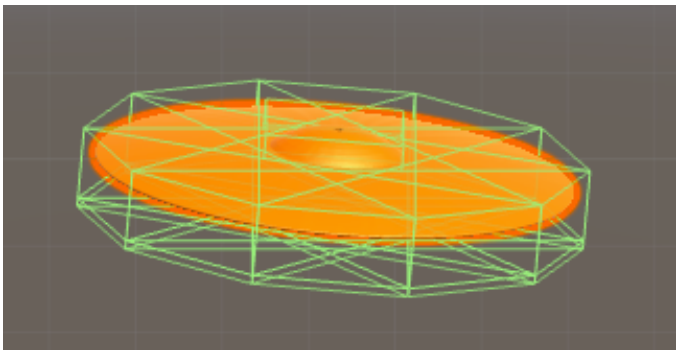

Figura 102 Vista superior de platillo de 22" con colliders

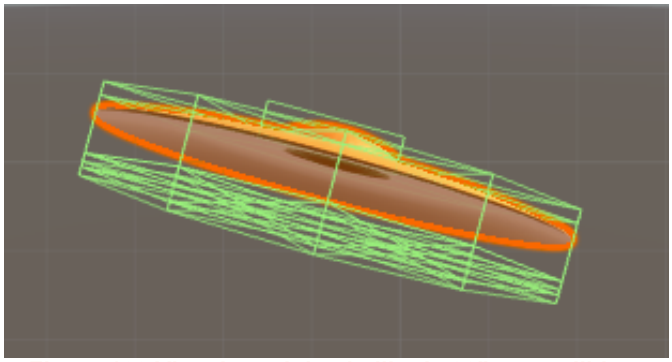

Figura 103 Vista lateral de platillo de 22" con colliders

En la Figura 104 se muestran los colliders generados para elplatillo de 22” dentro de un objeto llamado *Cymbal22TriggerColliders*.

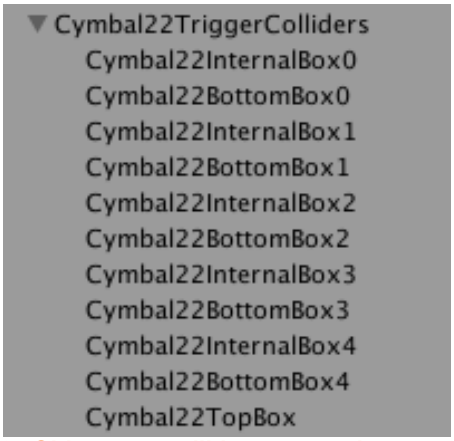

Figura 104 Objeto con colliders generados para la tarola

Además, si el pedal de Hi-hat es presionado, el platillo superior debe cambiar de posición. Para realizar dicha acción, se agrega un script al elemento. El algoritmo del script se muestra en la Figura 105.

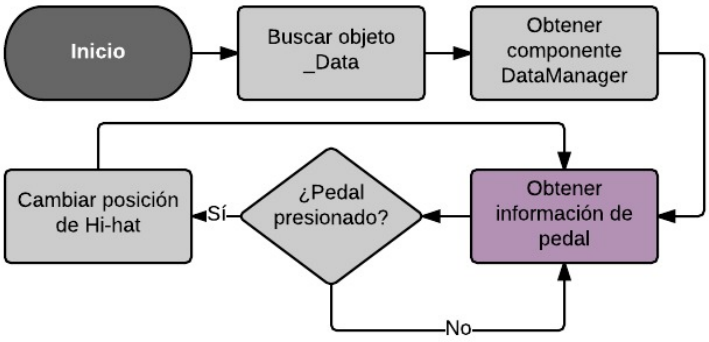

Figura 105 Algoritmo de cambio de posición de Hi-hat

En la Figura 106 se muestra el Hi-hat con el platillo superior en su posición inicial, y en la Figura 107 se muestra el Hi-hat con el platillo superior en una posición diferente.

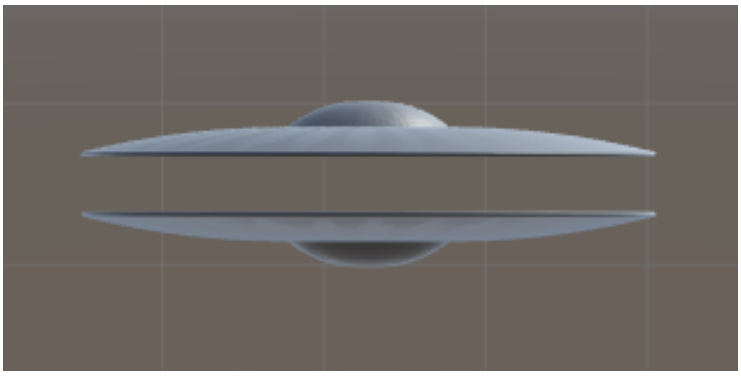

Figura 106 Hi-hat abierto

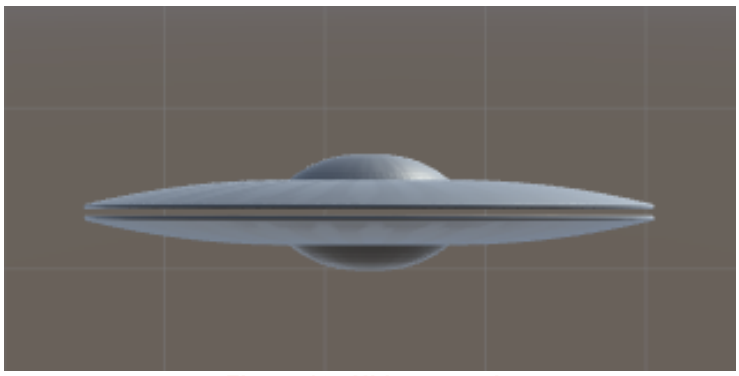

Figura 107 Hi-hat cerrado

En las Figuras 108 y 109 se muestran la batería virtual con los scripts antes de correr el proyecto.

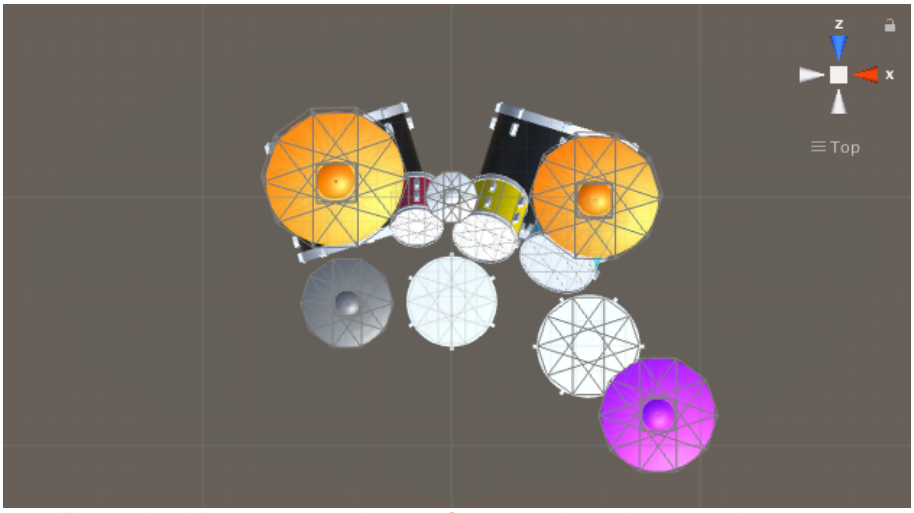

Figura 108 Vista superior de batería con scripts sin correr el proyecto

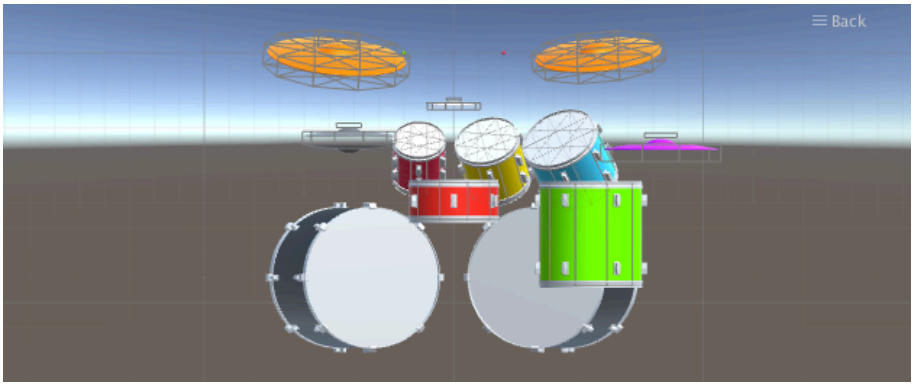

Figura 109 Vista frontal de batería con scripts sin correr el proyecto

En las Figuras 110 y 111 se muestran la batería virtual con los scripts con el proyecto corriendo.

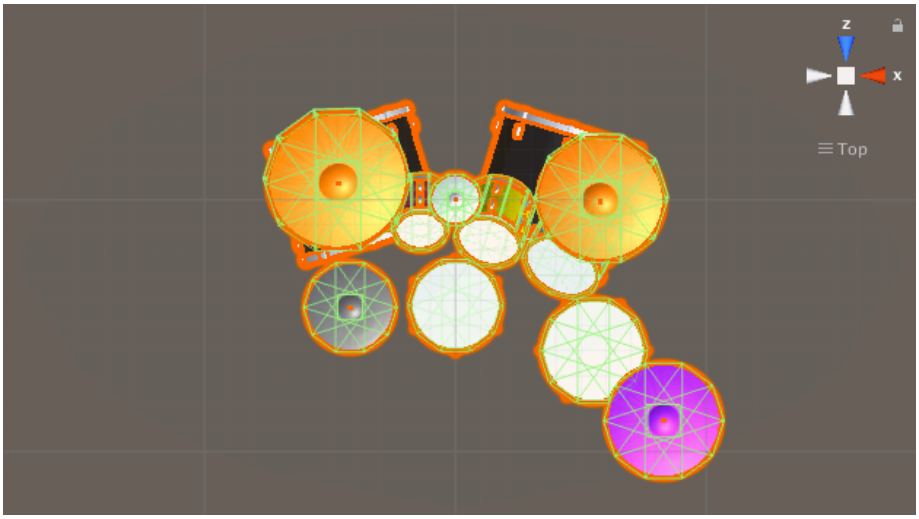

Figura 110 Vista superior de batería con colliders

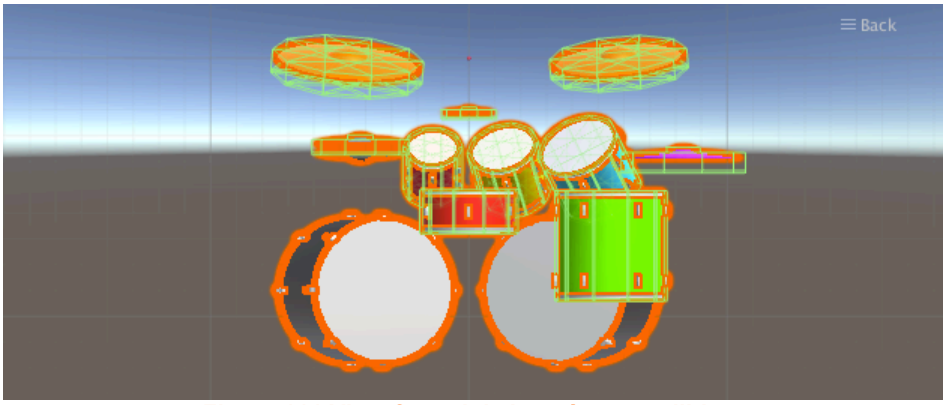

Figura 111 Vista frontal de batería con colliders

Con el proyecto corriendo, dependiendo si las baquetas cruzan un elemento de la batería, se reproducirán el sonido de dicho elemento dependiendo de la zona donde exista la colisión.

REALIDAD AUMENTADA

La realidad aumentada es la tecnología de proyectar una imagen generada por computadora sobre la visión del usuario del mundo real, generando así una visión compuesta [87]

La visión del usuario del mundo real puede ser obtenida a través de la cámara del dispositivo en donde se corra el software de realidad aumentada. Para este proyecto se usará un iPod Touch de 5a generación mostrado en la Figura 112.

Figura 112 Dispositivo donde se correrá el proyecto

En Unity es posible acceder y obtener la información de la cámara del dispositivo en donde se corra un proyecto, ya sea computadora, dispositivo portátil, etc.

Para integrar la información de la cámara del dispositivo dentro del proyecto de Unity se usará un plano. Dicho plano mostrará la información de la cámara y se ubicará frontal a la cámara dentro del proyecto y detrás del modelo virtual de la batería, con dimensiones tales que cubra por completo el rango de visión de la cámara del proyecto.

Al usar un dispositivo cuyo sistema operativo es iOS, el plano debe estar rotado 180 grados para invertir la imagen. Al correr el proyecto en la computadora, la información de la cámara se mostrará invertida, pero cuando se corra en el iPod, la imagen se mostrará como debe.

El plano generado se muestra en la Figura 113.

Figura 113 Batería con plano para realidad aumentada

Para que el plano muestre la información de la cámara se le agregará un script. El algoritmo de dicho script se muestra en la Figura 114.

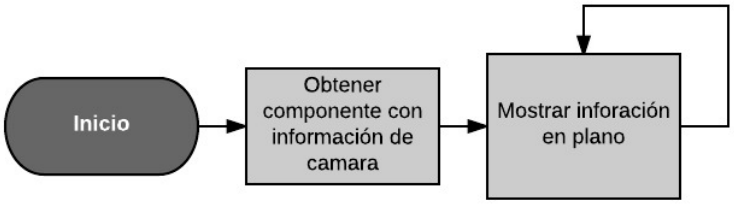

Figura 114 Algoritmo para obtención y muestra de información de cámara de dispositivo

Una vez implementado el script, al correr el proyecto se muestra la información de la cámara de la computadora invertida como se muestra en la Figura 115.

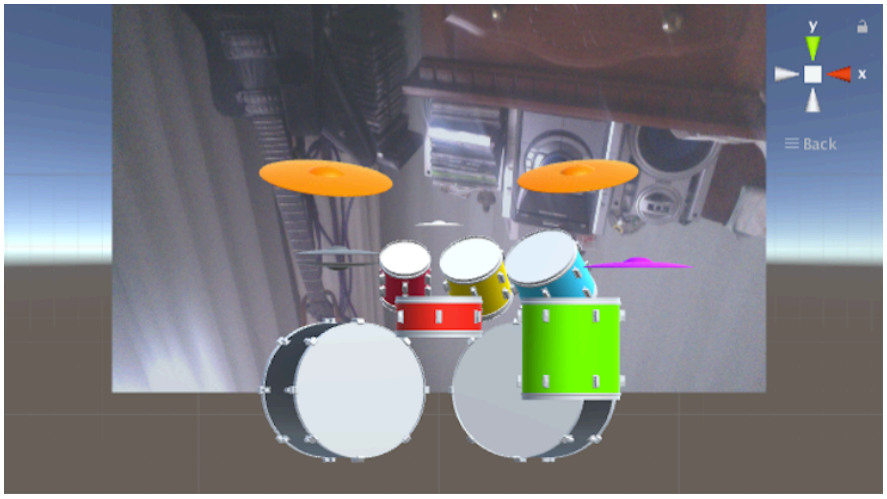

Figura 115 Batería con plano mostrando información de cámara de computadora

Para que el uso del software en el iPod sea lo mas natural posible, no basta con ponerlo frente al usuario con Google Cardboard; la cámara del proyecto debe girar en función de los movimientos del usuario y, además, se debe generar una imagen estereoscópica.

Lo estereoscópico se refiere al proceso en donde dos imágenes de un mismo objeto tomadas desde dos ángulos ligeramente diferentes son vistas al mismo tiempo creando una impresión de profundidad y solidez [89].

Para realizar ambas funciones se usará el paquete de herramientas de Cardboard para Unity [90].

El paquete de herramientas contiene un objeto llamado *GvrMain* el cual reemplaza la cámara del proyecto y brinda de dos cámaras distintas para crear la visión estereoscópica y accede a la información del giroscopio y acelerómetro del dispositivo en donde se corra el proyecto en caso de contar con ellos. En la Figura 116 se muestra el proyecto corriendo con el objeto GvrMain importado.

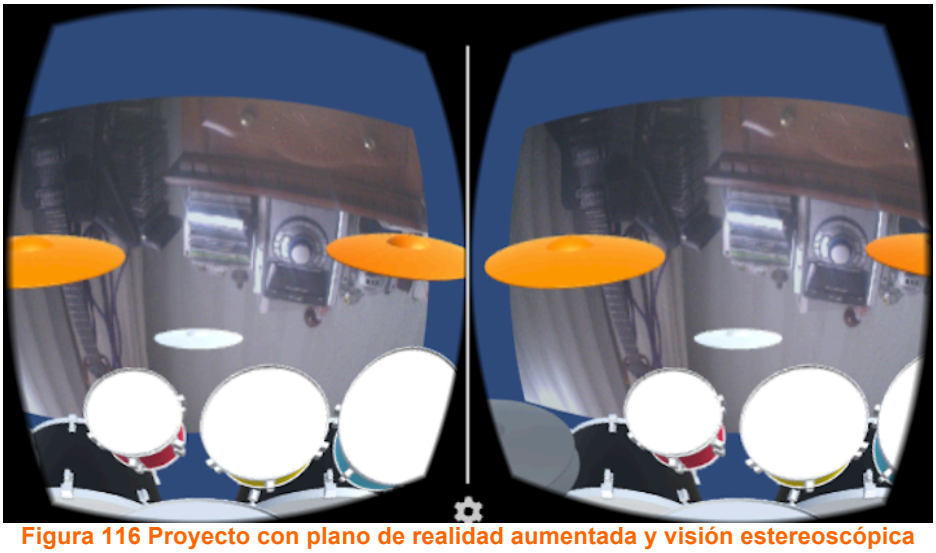

Figura 116 Proyecto con plano de realidad aumentada y visión estereoscópica

Una vez importado el objeto, se crea la visión estereoscopica y las cámaras se mueve con respecto a los movimientos del usuario, pero el plano permanece en la posición en donde se generó. Para solucionar dicho comportamiento, basta con poner el plano dentro del objeto *Head*, encargado de rotar la cámara en función de los movimientos del usuario, para que rote, en su posición local, de la misma forma que las cámaras. En la Figura 117 se muestra el plano dentro de GvrMain.

Figura 117 Objetos de GvrMain

Finalmente, se exportará el proyecto para, mediante Xcode, instalarse en el iPod. En la Figura 118 se muestra el proyecto corriendo en el iPod.

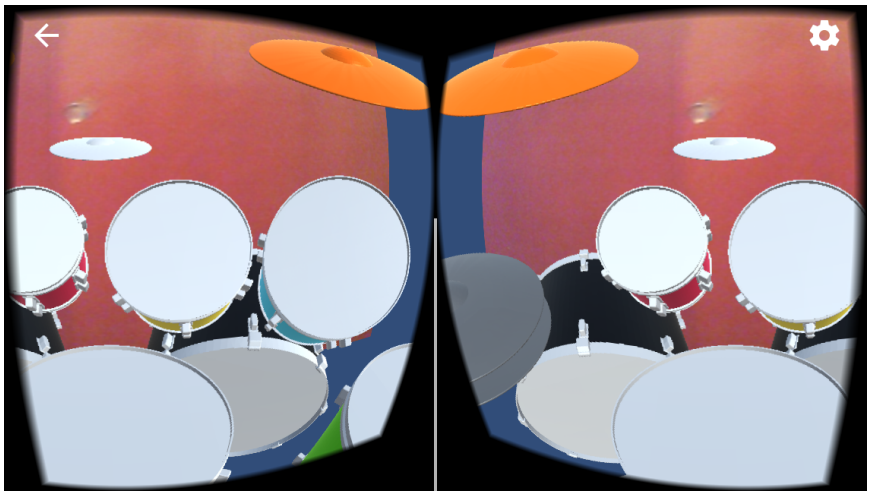

Figura 118 Proyecto corriendo en iPod

9 RESULTADOS

Con el proyecto en el dispositivo móvil, es posible comparar el trabajo completo con los valores de las especificaciones definidos en el Capítulo 6.

Los valores del trabajo y de las especificaciones se muestran en la Tabla 10, en donde se puede observar que todos los valores obtenidos del trabajo satisfacen los valores definidos.

Tabla 10 Comparación de valores definidos para especificaciones con los valores obtenidos

ESPECIFICACIONES	VALOR DEFINIDO	VALOR OBTENIDO
Cantidad máxima de elementos físicos	< 11 elementos	8 elementos
Tamaño máximo de elementos físicos	< 2.3 [m3]	0.00162 [m3]
Peso máximo de elementos físicos	< 70 [kg]	2.5 [kg]
Número de elementos virtuales	> 8 elementos	14 elementos

Elección de elementos virtuales	1 tarola 2 toms de aire 1 tom de piso 1 bombo 2 platillos 1 hi-hat	1 tarola 3 toms de aire 1 tom de piso 2 bombo 3 platillos 1 hi-hat
Medidas de elementos virtuales	Tarola: 6”x14” Toms de aire: 10”x12”, 9”x10”, 8”x8” Tom de piso: 15”x16”, 14”x14” Bombo: 18”x22” Platillos: 8”, 18”, 20”, 22” Hi-hat: 14”	Tarola: 6”x14” Toms de aire: 10”x12”, 9”x10”, 8”x8” Tom de piso: 15”x16”, 14”x14” Bombo: 18”x22” Platillos: 8”, 18”, 20”, 22” Hi-hat: 14”
Medidas de baquetas	5A	5A
Sonidos reales o MIDI de elementos escogidos	Tarola Toms de aire Tom de piso Bombo Platillos Hi-hat	Tarola Toms de aire Tom de piso Bombo Platillos Hi-hat
Elementos con variación de sonido	Platillos	Platillos
Zonas de variación o sin sonido	Tarola Toms de aire Tom de piso Platillos Hi-hat	Tarola Toms de aire Tom de piso Platillos Hi-hat
Capacidad dentro de software de modificar la batería virtual	> 8 elementos Color Posición	14 elementos Diferentes colores Diferentes posiciones
Tamaño máximo de hardware	Largo < 11 [cm] Ancho < 4 [cm] Profundo < 5 [cm]	Largo: 9 [cm] Ancho: 3.7 [cm] Profundo: 2.5 [cm]

Peso máximo de hardware	200 [g]	100 [g]
Uso de visor de realidad aumentada	Visor	Google Cardboard
Volumen de interacción mínimo	1.344 [m3]	~1.8 [m3]

Una vez comprobado que el trabajo cumple con las especificaciones, se pasa a la etapa de pruebas. Para realizar dichas pruebas se les pidió a varias personas probar el trabajo.

Algunas personas son usuarios de instrumentos de percusión, pero no todas. Esto se hace con el fin de poder cubrir un espectro de usuarios más amplio, y obtener la mayor cantidad de retroalimentación posible de distintas fuentes.

Las pruebas se realizaron de la siguiente forma:

- Se dejó que el usuario experimente el entorno libremente. Esto con el fin de que se acostumbre al entorno, compruebe que las baquetas se mueven en función de los movimientos, realicen algunos sonidos, y detecten que pueden ver su entorno con el uso de realidad aumentada.
- Se les pidió a los usuarios que fueran elemento por elemento realizando colisiones en distintas zonas para generar distintos sonidos y el usuario se de cuenta de las opciones sonoras con las que cuenta.
- Finalmente, se les pidió al usuario que intentaran interpretar algún ritmo.

En un principio, los usuarios se sorprendieron al ver el modelo virtual ante ellos. Posteriormente, movieron los brazos para empezar a generar sonidos con el software, pero no todos los movimientos generaban una colisión, esto debido a que, primero, tenían que calibrar sus movimientos con respecto a los del entorno virtual para lograr conocer la ubicación espacial de los elementos.

Una vez que conocen el entorno, empezaron a tocar los diferentes elementos de la batería, descubriendo los distintos sonidos que produce y expresando en su cara diversión, al producir sonidos con sus movimientos. En ésta prueba fue donde los usuarios comenzaron a dejarse llevar e interactuar de una forma más natural con el entorno, tocando más de un elemento a la vez o más rápido.

Cuando los usuarios intentaban generar colisiones más rápido, e interpretar ritmos, ya no se reprodujeron todos los sonidos. Este fenómeno se le atribuye principalmente a la transferencia de información entre la computadora y el dispositivo móvil.

En la Figura 70, se observa que la cadena de información que recibe Processing de Spacebrew es diferente a la que envía, lo que revela un atraso en la recepción de información del entorno. Además, al correr el proyecto en la computadora a la par que en el iPod, los sonidos de reproducían primero en la computadora. En la Figura 119 se muestra el editor de Processing enviando información a dos suscriptores de Unity distintos.

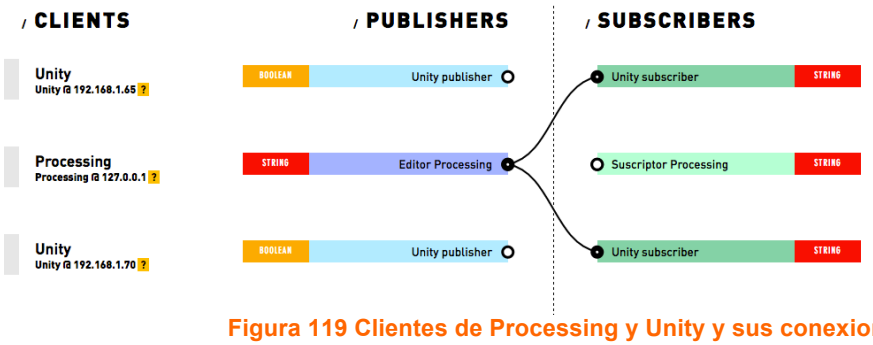

Figura 119 Clientes de Processing y Unity y sus conexiones

En la Figura 120 se muestra a una persona usando el software.

Figura 120 Persona usando visor y hardware de baquetas

Terminadas las pruebas, los usuarios dieron los siguientes comentarios:

- La conexión entre los módulos Bluetooth lleva demasiado tiempo o no se realizó la conexión hasta reiniciar el módulo
- En ocasiones, las baquetas se intercambiaban, es decir, la baqueta derecha se movía con la información de la izquierda y viceversa
- El sonido generado al realizar una colisión entre las baquetas y el exterior de algún tambor es molesto
- El proyecto aún no es amigable al usuario por todos los preparativos que lleva: Kinect, hardware de baquetas, receptor, etc.
- El campo de visión es limitado

- Los movimientos de las baquetas virtuales fueron mas lentos que los reales impidiendo la correcta interpretación de algún ritmo
- No hay alguna retroalimentación física que indique que se ha hecho una colisión
- Los pedales están limitados por las conexiones con el receptor

Con las pruebas hechas por los usuarios se logran satisfacer el objetivo general, y todos los objetivos específicos planteados en el Capítulo 4.

10 TRABAJO A FUTURO

De la retroalimentación obtenida por los usuarios se plantean los siguientes puntos a tratar:

- Cambiar la forma de transmisión de información del hardware de las baquetas y pedales al entorno virtual, de preferencia de una forma directa o sin necesidad de usar Processing para facilitar transmisión de datos y disminuir el tiempo de transmisión
- Mejorar la obtención de posición espacial de las manos del usuario, ya sea con hardware o con algoritmos que utilicen la información disponible, y así lograr que los movimientos dentro del entorno sean lo más parecidos a los reales para lograr interpretar ritmos de forma natural y evitar que las baquetas se intercambien
- Cambiar o eliminar el sonido generado al existir una colisión entre las baquetas y el exterior de un tambor, para evitar que el usuario se distraiga o irrite, con excepción de la tarola, ya que dicho sonido es utilizado
- Facilitar el uso del sistema, disminuyendo y/o mejorando el número de elementos involucrados, y/o proporcionando de una interfaz, para hacer el sistema mas amigable con el usuario
- Utilizar un visor dedicado al uso de realidad aumentada para incrementar el rango de visión y proporcionar de una mejor experiencia al usuario
- Agregar retroalimentación física al existir una colisión entre las baquetas y algún elemento para que, además de tener retroalimentación visual, se cuente con retroalimentación háptica

11 CONCLUSIONES

En éste trabajo se realiza una integración de distintas disciplinas: ciencias de la computación, electrónica, y música. De cierta manera podría parecer que la música es algo completamente distinto a la ingeniería y a la ciencia, pero es justo eso lo que hace relucir la necesidad tan grande de contar con más herramientas que las ofrecidas y adquiridas al estudiar ingeniería porque, desde el punto de vista del autor, la ingeniería es una herramienta para crear nuevas cosas, no un fin, y para crear nuevas cosas se necesita conocer más allá de la ingeniería o lo académico, se necesitan integrar distintas áreas, distintas disciplinas, y eso es algo fundamental para el trabajo.

La integración de distintas áreas contiene un concepto tratado en el capítulo uno: trabajo en equipo. Si bien, en la elaboración de éste trabajo, el autor trabajó sólo, Moisés Cabrera le ayudó a mejorar el concepto que ya se tenía, y, en un futuro, al mejorarlo o adentrarse más en aspectos técnicos, por ejemplo de producción musical, será imposible realizar todos los cambios sólo, por lo que se necesitará de la gente con los conocimientos necesarios para trasladar al trabajo a una mejor posición.

Se usaron tecnologías que llevan varios años disponibles, Kinect por ejemplo, para crear un nuevo concepto. Esto manifiesta la capacidad de las herramientas tecnológicas de seguir generando productos e ideas sin necesidad de ser nuevas, y el fin de generar un bien para las personas, en éste caso personas que gusten de la música.

La transferencia de datos juega un papel crucial en el trabajo al depender de la forma y velocidad de la transmisión para brindar una experiencia real. Es increíble la cantidad de información disponible con respecto a este tema, y la cantidad de herramientas disponibles, con lo cual seguramente se podrá mejorar esta parte del trabajo.

La modularidad fue algo que siempre estuvo en la mente al momento de generar el hardware para las baquetas ya que, en caso de requerir un cambio o, de necesitar algún componente para otra ocasión, se pudiesen retirar fácilmente. Si se retira este concepto del hardware y se define sólo para satisfacer las necesidades de éste proyecto, el hardware generado podrá ser aún más cómodo.

Finalmente, la búsqueda de herramientas y conocimiento es lo que logró realizar este trabajo. Ésta es una habilidad con la cual se debe contar, no sólo para la elaboración de algún proyecto, sino en cualquier aspecto de la vida. Es necesario seguir buscando herramientas para lograr mejorar los distintos bloques que conforman éste trabajo, y por consecuencia generar nuevas soluciones.

12 REFERENCIAS

[1] BMW. BMW Service. Recuperado en marzo de 2016 de: http://www.bmw.com/bs/en/owners/service/augmented_reality_introduction_2.html
[2] BMW Augmented Reality in Practice. Recuperado en marzo de 2016 de: https://realtimecities.wikispaces.com/BMW+Augmented+Reality+in+practice
[3] Sam Shead. BMW hopes Google's augmented reality Tango technology will help it to sell cars. Recuperado en enero de 2017 de: http://www.businessinsider.de/bmw-google-augmented-reality-tango-2017-1
[4] Emma Begley. BMW i pilots augmented reality product visualizer powered by Tango, Google's Smartphone AR technology. Recuperado en enero de 2017 de: https://www.press.bmwgroup.com/global/article/detail/T0266825EN/bmw-i-pilots-augmented-reality-product-visualiser-powered-by-tango-google%E2%80%99s-smartphone-ar-technology?language=en
[5] Grand View Research. Augmented Reality (AR) Market Analysis By Component (Hardware and Software), Display (Head-Mounted Display, Head-Up Display, and Smart Glass), Application (Aerospace & Defense, Medical, Gaming, Industrial, Automotive & E-commerce & Retail) And Segment Forecasts to 2024. Recuperado en agosto de 2016 de: http://www.grandviewresearch.com/industry-analysis/augmented-reality-market
[6] Digi-Capital. Augmented/Virtual Reality revenue forecast revised to hit $120 billion by 2020. Recuperado en agosto de 2016 de: http://www.digi-capital.com/news/2016/01/augmentedvirtual-reality-revenue-forecast-revised-to-hit-120-billion-by-2020/#.WOMH7RLhDab
[7] Nasdaq GlobeNewswire. Augmented Reality Market size to exceed $165bn by 2024: Global Market Insights Inc. Recuperado en enero de 2017 de: https://globenewswire.com/news-release/2017/01/12/905349/0/en/Augmented-Reality-Market-size-to-exceed-165bn-by-2024-Global-Market-Insights-Inc.html
[8] HTC VIVE. VIVE. Recuperado en agosto de 2016 de: https://www.vive.com/
[9] Mark Walton. PSVR vs. HTC Vive vs. Oculus Rift vs. Gear VR: Which VR Headset Should You Buy?. Recuperado en agosto de 2016 de: https://arstechnica.com/gaming/2016/10/best-vr-headset-2016-psvr-rift-vive/
[10] Leap Motion. Leap Motion. Recuperado en agosto de 2016 de: https://www.leapmotion.com/
[11] Leap Motion. Leap Motion Controller for Mac or PC (Retail Packaging and Updated Software). Recuperado en agosto de 2016 de: https://www.amazon.com/Leap-Motion-Controller-Packaging-Software/dp/B00HVYBWQO
[12] Microsoft. Kinect para Xbox One. Recuprado en noviembre de 2016 de: http://www.xbox.com/es-MX/xbox-one/accessories/kinect
[13] Unity. Unity – Game Engine. Recuperado en agosto de 2016 de: https://unity3d.com/es/
[14] Unity (moteur de jeu). Recuperado en agosto de 2016 de: http://www.wikiwand.com/fr/Unity_(moteur_de_jeu)

[15] Unreal Engine. What is Unreal Engine. Recuperado en agosto de 2016 de: https://www.unrealengine.com/what-is-unreal-engine-4
[16] Unreal Engine. Recuperado en agosto de 2016 de: https://es.wikipedia.org/wiki/Unreal_Engine
[17] Vuforia. Vuforia l Augmented Reality. Recuperado en agosto de 2016 de: https://www.vuforia.com/
[18] Kudan. Home l Kudan. Recuperado en agosto de 2016 de: https://www.kudan.eu/
[19] András Velvárt. Hololens vs Meta2. Recuperado en octubre de 2016 de: https://vbandi.net/2016/03/04/hololens-vs-meta-2/
[20] Tom Warren. Microsoft reveals Hololens hardware specs. Recuperado en agosto de 2016 de: http://www.theverge.com/2016/2/29/11132090/microsoft-hololens-hardware-specifications-developer-kit
[21] Microsoft. Microsoft Hololens. Recuperado en agosto de 2016 de: https://www.microsoft.com/microsoft-hololens/en-us
[22] Atharv Dixit. Microsoft Hololens : A New Aeon of Blended Reality. Recuperado en agosto de 2016 de: http://geeksnation.org/microsoft-hololens-new-aeon-blended-reality/
[23] Metavision. Meta 2. Recuperado en agosto de 2016 de: https://www.metavision.com/
[24] Google. Cardboard. Recuperado en agosto de 2016 de: https://vr.google.com/cardboard/
[25] Google. Google Store. Recuperado en agosto de 2016 de: https://store.google.com/product/google_cardboard
[26] Aerodrums. Aerodrums. Recuperado en agosto de 2016 de: http://aerodrums.com/aerodrums-product-page/
[27] Aerodrums. Aerodrums. Recuperado en agosto de 2016 de: https://www.kvraudio.com/product/aerodrums-by-aerodrums
[28] Freedrum. Freedrum: The Drumkit That Fits in Your Pocket. Recuperado en agosto de 2016 de: https://www.kickstarter.com/projects/freedrum/freedrum-the-drumkit-that-fits-in-your-pocket
[29] Freedrum: Ultimate Air Drums. Recuperado en agosto de 2016 de: https://www.youtube.com/watch?v=vV_Qhn1i1Cc
[30] Moisés Cabrera. Oficial Moy Drums. Recuperado en septiembre de 2016 de: https://www.facebook.com/oficialmoisescabrera/videos/1410981478970154/
[31] Chris Stobing. Oculus Rift vs. HTC Vive: Which VR Headset is right for you?. Recuperado en agosto de 2016 de: https://www.howtogeek.com/246333/oculus-rift-vs.-htc-vive-which-vr-headset-is-right-for-you/
[32] LeapPaint – Simple Painting program controlled by Leap Motion. Recuperado en septiembre de 2016 de: https://www.youtube.com/watch?v=xi_9mBDfE9I
[33] Alex Colgan. How Does the Leap Motion Controller Works?. Recuperado en septiembre de 2016 de: http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/
[34] Microsoft. Skeletal Tracking. Recuperado en octubre de 2016 de: https://msdn.microsoft.com/en-us/library/hh973074.aspx#ID4ENB

[35] Processing. Processing. Recuperado en octubre de 2016 de: https://processing.org/
[36] Kinect Sensor with Kinect Adventures! – Xbox 360 Standard Edition. Recuperado en octubre de 2016 de: https://www.amazon.ca/Kinect-Sensor-Adventures-Xbox-Standard/dp/B002BSA298
[37] IvenSense. MPU-6050 Six-Axis (Gyro + Accelerometer) MEMS MotionTrackingTM Devices. Recuperado en octubre de 2016 de: https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/
[38] GY-521 MPU6050 3-Axis Acceleration Gyroscope 6DOF Module – Blue. Recuperado en octubre de 2016 de: http://www.dx.com/p/gy-521-mpu6050-3-axis-acceleration-gyroscope-6dof-module-blue-154602#.WPbWv1M1_aY
[39] Serial Port Bluetooth (Master/Slave) : HC-05. Recuperado en octubre de 2016 de: https://www.itead.cc/wiki/Serial_Port_Bluetooth_Module_(Master/Slave)_:_HC-05
[40] HC-05 Bluetooth serial pass-through module wireless serial communication with button from machine Wireless HC05 for Arduino. Recuperado en octubre de 2016 de: http://pickmygadget.eu/products/hc-05-bluetooth-serial-pass-through-module-wireless-serial-communication-with-button-from-machine-wireless-hc05-for-arduino/
[41] Arduino. Arduino Pro Mini. Recuperado en octubre de 2016 de: https://www.arduino.cc/en/Main/arduinoBoardProMini
[42] Arduino. Arduino Due. Recuperado en octubre de 2016 de: https://www.arduino.cc/en/Main/arduinoBoardDue
[43] Sparkfun. Mini Pushbutton Switch. Recuperado en octubre de 2016 de: https://www.sparkfun.com/products/97
[44] Autodesk. Cloud Powered 3D CAD/CAM Software for Product Design | Fusion 360. Recuperado en octubre de 2016 de: http://www.autodesk.com/products/fusion-360/overview
[45] Autodesk. Fusion 360 – Autodesk CAM Solutions. Recuperado en octubre de 2016 de: http://cam.autodesk.com/fusion360/
[46] Spacebrew. Spacebrew. Recuperado en octubre de 2016 de: http://docs.spacebrew.cc/
[47] Spacebrew. Spacebrew Server Workshop @ ITP. Recuperado en octubre de 2016 de: https://www.slideshare.net/julioterra/spacebrew-server-workshop-itp
[48] Apple. Xcode – Apple Developer. Recuperado en septiembre de 2016 de: https://developer.apple.com/xcode/
[49] Xcode – logo. Recuperado en septiembre de 2016 de: http://logonoid.com/xcode-logo/
[50] XQuartz. XQuartz. Recuperado en septiembre de 2016 de: https://www.xquartz.org/
[51] The Open Group. The X Windows System TM. Recuperado en septiembre de 2016 de: http://www.opengroup.org/desktop/x/
[52] CMake. CMake. Recuperado en septiembre de 2016 de: https://cmake.org/
[53] MacPorts. MacPorts. Recuperado en septiembre de 2016 de: https://www.macports.org/install.php

[54] MacPorts. The MacPorts Wiki. Recuperado en septiembre de 2016 de: https://trac.macports.org/
[55] GNU. GNU Libtool – The GNU Portable Library Tool. Recuperado en septiembre de 2016 de: https://www.gnu.org/software/libtool/
[56] Libusb. Libusb – A cross-platform user library to acces USB devices. Recuperado en septiembre de 2016 de: http://libusb.info/
[57] OpenNI-Bin-Dev-MacOSX-v1.5.7.10.tar.zip. Recuperado en septiembre de 2016 de: https://mega.nz/#!yJwg1DJS!uJiLY4180QGXjKp7sze8S3eDVU71NHiMrXRq0TA7QpU
[58] PrimeSense. Avin2 / SensorKinect. Recuperado en septiembre de 2016 de: https://github.com/avin2/SensorKinect
[59] NITE-Bin-Dev-MacOSx-v1.5.2.21.zip. Recuperado en septiembre de 2016 de: https://mega.nz/#!XNgGSZDb!m091FXc4U6GwjRfpHK-puPvBjkHdWc6KmQH-_RzXfOw
[60] Glen Mcpherson. How to setup Mirosoft Kinect on Mac OS X 10.8 (Mountain Lion). Recuperado en septiembre de 2016 de: http://blog.nelga.com/setup-microsoft-kinect-on-mac-os-x-10-8-mountain-lion/
[61] 3d Sense. Programming for Kinect 3 – A simple Kinect app in Processing. Recuperado en septiembre de 2016 de: http://3dsense.czprogramming/programming-for-kinect-3-a-simple-kinect-app-in-processing/
[62] Joseph Smith. Processing Libraries. Recuperado en septiembre de 2016 de: https://github.com/technicolorenvy/Processing-Libraries
[63] MPU6050 calibration. Recuperado en octubre de 2016 de: https://codebender.cc/sketch:97892#MPU6050%20calibration.ino
[64] Jeff Rowberg. 12cdevlib. Recuperado en octubre de 2016 de: https://github.com/jrowberg/i2cdevlib
[65] InvenSense. MPU-6000 and MPU-6050 Product Specification Revision 3.4. Recuperado en octubre de 2016 de: https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf
[66] InvenSense. MPU-6000 and MPU-6050 Register Map and Descriptions Revision 4.2. Recuperado en octubre de 2016 de: https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf
[67] NPX. UM10204. Recuperado en octubre de 2016 de: http://www.nxp.com/documents/user_manual/UM10204.pdf
[68] Buffer. Recuperado en octubre de 2016 de: https://techterms.com/definition/buffer
[69] Steven M LaValle. Yaw, pitch, and roll rotations. Recuperado en octubre de 2016 de: http://planning.cs.uiuc.edu/node102.html
[70] Aldebaran. Joints. Recuperado en octubre de 2016 de: http://doc.aldebaran.com/2-1/family/robots/joints_robot.html
[71] Ogre. Quaternion and Rotation Primer. Recuperado en octubre de 2016 de: http://www.ogre3d.org/tikiwiki/Quaternion+and+Rotation+Primer
[72] Vivianne A. Mahecha. Aplicación de los números hipercomplejos o cuaterniones en imágenes de color. Recuperado en octubre de 2016 de:

http://rcientificas.uninorte.edu.co/index.php/ingenieria/article/viewArticle/3026/4464#4
[73] Hazim Bitar. Modify The HC-05 Bluetooth Module Defaults Using AT Commands. Recuperado en octubre de 2016 de: http://www.techbitar.com/modify-the-hc-05-bluetooth-module-defaults-using-at-commands.html
[74] Sparkfun. Coin Cell Battery – 24.5 mm (Rechargeable CR2450). Recuperado en octubre de 2016 de: https://www.sparkfun.com/products/10319
[75] Processing. Serial. Recuperado en octubre de 2016 de: https://processing.org:8443/reference/libraries/serial/index.html
[76] Stacey Mulcahy. Internet of things with Unity3D, Arduino and Spacebrew – Part 2. Recuperado en noviembre de 2016 de: http://thebitchwhocodes.com/2014/07/29/internet-of-things-with-unity3d-arduino-and-spacebrew-part-2/
[77] Spacebrew. spacebrewUnity. Recuperado en noviembre de 2016 de: https://github.com/Spacebrew/spacebrewUnity
[78] Unity. Creando y usando scripts. Recuperado en diciembre de 2016 de: https://docs.unity3d.com/es/current/Manual/CreatingAndUsingScripts.html
[79] Sabian. Education – Anatomy of a Cymbal. Recuperado en diciembre de 2016 de: http://www.sabian.com/en/pages/anatomy-of-a-cymbal
[80] Unity. Rigidbodies. Recuperado en diciembre de 2016 de: https://docs.unity3d.com/es/current/Manual/RigidbodiesOverview.html
[81] Unity. Colliders. Recuperado en diciembre de 2016 de: https://docs.unity3d.com/es/current/Manual/CollidersOverview.html
[82] Vater. Vater VH5AW Los Angeles 5A Wood Tip Hickory Drum Sticks, Pair. Recuperdo en diciembre de 2016 de: https://www.amazon.com/Vater-Percussion-Drumsticks-Wood-Tip/dp/B0002CZQJQ
[83] Blender. Blender.org – Home of the Bender project – Free and Open 3D Creation Software. Recuperado en diciembre de 2016 de: https://www.blender.org/
[84] Julian Casadesus. TuxGuitar. Recuperado en diciembre de 2016 de: https://sourceforge.net/projects/tuxguitar/
[85] TrueFire Support Desk. Using Tab: Working with Guitar Pro Tab (GP5 or PTB) files. Recuperado en diciembre de 2016 de: https://truefire.zendesk.com/hc/en-us/articles/200201296-Using-Tab-Working-with-Guitar-Pro-Tab-GP5-or-PTB-files
[86] Kode80. Various editor/GUI tolos for Unity 3D. Recuperado en diciembre de 2016 de: https://github.com/kode80/unitytools/tree/develop
[87] Oxford dictionary. Augmented reality. Recuperado en enero de 2017 de: https://en.oxforddictionaries.com/definition/augmented_reality
[88] Matthew Hallberg. Markerless Augmented Reality Tutorial. Recuperado en enero de 2017 de: http://wirebeings.com/markerless-augmented-reality.html
[89] Ocford dictionary. Stereoscopic. Recuperado en enero de 2017 de: https://en.oxforddictionaries.com/definition/stereoscopic
[90] Google. Google VR SDK for Unity | Google VR | Google Deevelopers. Recuperado en enero de 2017 de: https://developers.google.com/vr/unity/