UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

“AR-Learning: Interactive book based on augmented reality for educational intentions”

Roberto Gallego Delgado

Diciembre, 2010

ÍNDICE

ÍNDICE DE FIGURAS

Resumen:

En los últimos años la tecnología ha avanzado a grandes pasos, introduciéndose en muchos campos de la ciencia e investigación, ejército, educación y en todos los ámbitos de nuestra vida cotidiana. Es difícil a día de hoy no tener un ordenador, una televisión, un mp4 o una videoconsola. Este avance se ha intentado aprovechar en amplios campos, y uno de ellos es en el de la educación. Hace años era impensable tener en un aula un ordenador por cada dos alumnos, o exponer la lección sobre una pantalla de proyección en vez de en una pizarra. Una de las tecnologías que pueden suponer una innovación en las aulas es la realidad aumentada. Esta técnica permite integrar modelos virtuales 3D a la realidad física mediante un dispositivo de adquisición de vídeo y un ordenador. No se iguala a la realidad virtual, pues ésta última sustituye la realidad física por la virtual, mientras que la realidad aumentada integra las dos realidades para crear un ámbito de interacción con el usuario hasta ahora inexistente.

Nuestro proyecto consiste en crear una aplicación de realidad aumentada aplicada a la educación, y más concretamente, a la educación musical. La realidad aumentada ya ha sido aplicada a la educación en el ámbito de la geografía, la física o el arte, mostrando modelos 3D de un volcán en erupción con todas sus partes, las distintas capas de la tierra, o como debían ser las casas y murallas en la época romana. Nuestro objetivo es crear una aplicación en un ámbito poco conocido todavía por la realidad aumentada, y además aprovechar que podemos crear aplicaciones mucho más interactivas que con cualquier otra materia.

Para ello utilizaremos la librería OSGART, cuyas funciones nos permiten crear modelos virtuales 3D, combinando la biblioteca ARToolKit y OpenSceneGraph, y la librería SDL que nos permitirá implementar el sonido de la aplicación. OSGART basa su funcionamiento en mostrar un marcador cuadrado negro con un dibujo en su centro, y cuando nuestro dispositivo de vídeo lo detecte en su totalidad, muestra o pinta un modelo 3D en la pantalla.

Nuestra aplicación tratará de enseñar los principios básicos de la música al alumno mediante la realidad aumentada, visualizando en 3D los instrumentos, escuchando su sonido cuando interactúen con ellos, etc. También se enseñarán las

notas básicas del pentagrama musical y las cualidades del sonido como son altura, timbre o intensidad. Por último mostramos un ejemplo de una pequeña banda musical.

Conseguido este objetivo, se deja abierta la aplicación para una futura utilización y expansión de la misma con el fin de mejorarla y conseguir otras actividades educativas dentro de esta línea.

Introducción: Aumentando la realidad

La temática principal de este proyecto es la realidad aumentada y su aplicación a la enseñanza a través de las nuevas tecnologías.

La realidad aumentada [23] es un concepto que actualmente está apareciendo cada vez más, pero que causa algo de confusión debido a que todavía no está implantada en aplicaciones usuales en la sociedad. Es un concepto que puede parecerse a la realidad virtual, pero éstas son muy diferentes. La realidad aumentada consiste en un conjunto de dispositivos que añaden información virtual a la información física ya existente [13], mientras que la realidad virtual sustituye a la realidad física. Con la realidad aumentada lo que buscamos es una síntesis de los elementos reales y virtuales, una integración de imágenes cuyo objetivo es que la información adicional que nos muestra nos sirva de ayuda y consiga llevar al usuario a otro nivel de interactividad con la computadora sin sentirse ajeno a ella, como en el caso de la realidad virtual [18]. Con la ayuda de la tecnología la información sobre el mundo real alrededor del usuario se convierte en interactiva y digital. La información artificial sobre el medio ambiente y los objetos pueden ser almacenada y recuperada como una capa de información en la parte superior de la visión del mundo real.

En los últimos tiempos está en auge la aplicación de las nuevas tecnologías en cualquier ámbito, como la educación y el ocio. Siguiendo esta línea, la realidad aumentada ha conseguido hacerse hueco en el mundo de la medicina, los videojuegos, la educación e incluso el ejército. Nuestro objetivo es crear una aplicación educativa para la enseñanza musical, un campo que la realidad aumentada aún no ha abarcado y que nos da total libertad de ideas y aplicaciones. Intentaremos enseñar algo tan palpable como puede ser la música de una forma realmente interactiva, donde el alumno pueda contactar directamente con lo que se le enseña.

Para ello, utilizaremos las librerías OSGART[10] y SDL [20], que nos permitirán dibujar figuras virtuales y ponerles sonido respectivamente. Se creará una aplicación donde se pueda interactuar tanto visual como auditivamente, consiguiendo que se relacionen determinados sonidos con determinados instrumentos, que se distingan instrumentos entre sí o que se aprecien las distintas notas musicales.

Objetivos:

Como ya hemos comentado anteriormente, el abanico de aplicaciones donde podemos utilizar la realidad aumentada es muy amplio. Inicialmente no era el objetivo principal del proyecto, pero surgió la idea de aplicar esta tecnología a la música [1] y todo avanzó sin dificultad.

Propusimos crear una aplicación para la enseñanza musical básica donde el niño pudiese interactuar con ella, siendo esta una forma innovadora y amena de transmitir nuestros conocimientos.

Nuestros principales objetivos son:

- Dar una alternativa de la enseñanza musical tradicional, que en edades tempranas peca de demasiada formalidad, lo que resta interés en la mayoría del alumnado.

- Convertir la realidad aumentada en técnica de aprendizaje, para dar otra visión del temario mucho más amena y divertida, y aplicar las nuevas tecnologías a la enseñanza.

- Crear un programa en el cual podamos plantear un pequeño temario de ámbito musical, aplicando las técnicas de realidad aumentada.

- Lograr que el alumno adquiera unos conocimientos relacionados con la música interactuando con ella misma, ya sea visualmente (observando instrumentos o notas) o auditivamente (escuchándolos).

Capítulo 1

1.1. Educación en el siglo XXI: Nuevas tecnologías y aplicación en la enseñanza musical

La introducción generalizada de las Nuevas Tecnologías de la información y la comunicación en todos los ámbitos de nuestras vidas está produciendo un cambio significativo en nuestra manera de trabajar, de relacionarnos y de aprender [17].

Las Nuevas Tecnologías se plantean ahora como un hecho trascendente y apremiante. Esto es debido a que son el resultado de los avances científico-técnicos de nuestra sociedad y por que provocan cambios de todo tipo en las estructuras sociales, económicas, laborales e individuales. Esta situación trae consigo la creación de nuevos entornos de comunicación, tanto humanos como artificiales, no conocidos hasta la actualidad. Se establecen nuevas formas de integración de los usuarios con las máquinas y se modifican los clásicos roles de receptor y transmisor de información.

El rol de las Nuevas Tecnologías de la información en los procesos de cambio social y cultural cobra particular relevancia en el ámbito educativo. Las tecnologías de la información se aplican al campo pedagógico con el objetivo de mejorar los resultados del sistema escolar y asegurar el acceso al mismo de grupos convencionalmente excluidos.

Sin embargo, para que las Nuevas Tecnologías de la información se apliquen como Nuevas Tecnologías de la educación es preciso que se cumplan ciertos requisitos básicos, tales como contar con el adecuado conocimiento en modelos antropológicos, culturales y educativos que favorezcan una intervención didáctica apropiada, además de una adecuada formación de los profesores y otros especialistas de la educación.

1.1.1. Nuevas Tecnologías, nuevos entornos didácticos

Las Nuevas Tecnologías y su incorporación al ámbito educativo generan la creación de nuevos entornos didácticos que afectan de manera directa tanto a los implicados en el proceso de enseñanza-aprendizaje como al escenario donde se lleva a cabo el mismo. Este nuevo entorno, creado a partir de las Nuevas Tecnologías requiere un nuevo tipo de alumno; más preocupado por el proceso que por el resultado, preparado para la toma de decisiones y elección de su ruta de aprendizaje. En definitiva, preparado para el autoaprendizaje, lo cual abre un desafío a nuestro sistema educativo, preocupado por la adquisición y memorización de información y la reproducción de la misma en función de patrones previamente establecidos.

Es por ello que las Nuevas Tecnologías aportan un nuevo reto al sistema educativo que consiste en pasar de un modelo unidireccional de formación, donde por lo general los conocimientos recaen en el profesor o en su sustituto (el libro de texto) a modelos más abiertos y flexibles, donde la información situada en grandes bases de datos, tiende a ser compartida entre diversos alumnos. Frente a los modelos tradicionales de comunicación que se dan en nuestra cultura escolar, algunas de las tecnologías generan una nueva alternativa donde el aula se convierte en un entorno en el que el alumno puede interactuar con otros compañeros y profesores de una manera mucho más amena y productiva.

Figura 1.1: Aula donde se han instalado ordenadores para el uso del alumnado

Para que los medios queden integrados en el trabajo cotidiano de las aulas, se requiere la participación activa de un elemento clave: el *profesional de la educación*.

Es él quien, en cada situación de aprendizaje, con sus decisiones y su actuación, conseguirá que el medio quede integrado. Desde esta perspectiva es evidente que el papel que debe desempeñar el profesor ha de sufrir un cambio profundo con respecto al que ha ejercido de forma tradicional. El profesor pasará de ser el elemento predominante y exclusivo en la transmisión de conocimientos a convertirse en una pieza clave del proceso de aprendizaje pero no única, como elemento generador y organizador de las actividades de aprendizaje.

El profesor constituye una pieza esencial de todo proceso de mejora de la enseñanza, para lo cual su formación inicial en Nuevas Tecnologías resulta fundamental. De ahí que haya que plantearse seriamente el tema de la formación de docentes en el uso de las Nuevas Tecnologías que garanticen la verdadera integración de estas herramientas en la realidad escolar.

1.1.2. Las nuevas tecnologías y la música

Desde hace años, aunque alejado de los ambientes académicos, las tecnologías empezaron a tomar posiciones en los ambientes musicales. El descenso de los precios y la aparición en nuestras casas como algo habitual de herramientas y programas informáticos cada vez más sencillos de usar y de más fácil adquisición, ha permitido el acercamiento entre la expresión artística, la música y las nuevas tecnologías, sobre todo a nivel de interpretación y composición, es decir, fomentando la creatividad.

La expresión musical contribuye notablemente al desarrollo integral de la personalidad del niño. Es por ello, que debe ocupar un lugar de mayor importancia que el que tiene actualmente en el currículo escolar. El mensaje musical puede dar una información extra que enriquezca y profundice el mensaje verbal y visual para comprender una situación, una época o un lugar, pudiendo ayudar a explorarlas. Todo esto dará lugar a experiencias valiosísimas en la vida del niño, sin que por ello se desvirtúen o anulen los objetivos propios de la materia. Enseñar a escuchar es una tarea que sobrepasa la finalidad artística para situarse a nivel muy alto de ampliación y extensión de las relaciones y aprendizaje humanos.

La integración del audio, gráficos y vídeo en un mismo soporte nos permite la creación y utilización de programas que ayudan en el aprendizaje de lo musical,

dándole al alumno y al docente la sensación de convertir el proceso enseñanza-aprendizaje en un juego. En estos tiempos que corren la expresión musical también tiene que aprovecharse de las posibilidades que ofrecen las nuevas tecnologías, no puede quedarse al margen de este desarrollo, por ello debe explotarse en nuestras escuelas las oportunidades didácticas que ofrece esta unión entre música y tecnología.

Naturalmente, la posibilidad de introducir en las aulas de expresión musical modelos y métodos didácticos de trabajos inéditos hasta ahora, supone enormes esfuerzos por parte del docente para adquirir unos conocimientos previos en el manejo y uso de estas tecnologías, siempre con miras de obtener resultados a largo plazo. Esta tarea supone un esfuerzo por parte del docente y no siempre tiene buena aceptación por parte de los profesores, al igual que en las demás áreas del currículo, pero debemos ser conscientes que la expresión musical tiene que aprovechar de manera muy positiva las posibilidades que ofrecen estas nuevas opciones tecnológicas. Estos medios son muy rentables porque les son familiares a los alumnos, ayudan a su creatividad musical y además resultan muy estimulantes, por tanto no puede sino decirse que es una oportunidad extraordinaria y muy valiosa para su incorporación en el ámbito de la enseñanza musical.

La informática, como parte muy importante de la llamada nueva tecnología, es ideal para fomentar uno de los ejes primordiales de la pedagogía musical, como es la creatividad, algo que viene resaltándose a lo largo de todo este apartado. La informática cuando se domina nos permite manejar el sonido y la música como auténticos expertos. Los sonidos, las melodías, canciones, etc., pueden ser creados, repetidos y transformados con instrucciones muy fáciles de ordenador. Nos proporcionan infinidad de posibilidades compositivas con las que eran impensables hace muy pocas fechas atrás. Ejemplos de esto son programas informáticos como Cubase, Reason, Cakewalk Sonar o Guitar pro [24]. Estos programas nos permiten componer, editar, grabar y masterizar nuestras creaciones como si fuésemos expertos técnicos de sonido y con una calidad comparable a la de un estudio musical, si se sabe utilizar las herramientas de dicho programa. Esta experiencia de creación musical, con las posibilidades que la informática ofrece, es muy gratificante tanto para los alumnos como para los docentes, tanto para aquellos que se están iniciando en el lenguaje musical como para aquellos con cierta soltura en el arte de la composición. Por todo, es irrebatible que la informática se convertirá, si no lo es ya,

en un elemento fundamental en el campo educativo de la educación musical, como ocurre en otros países más avanzados que el nuestro en esta materia.

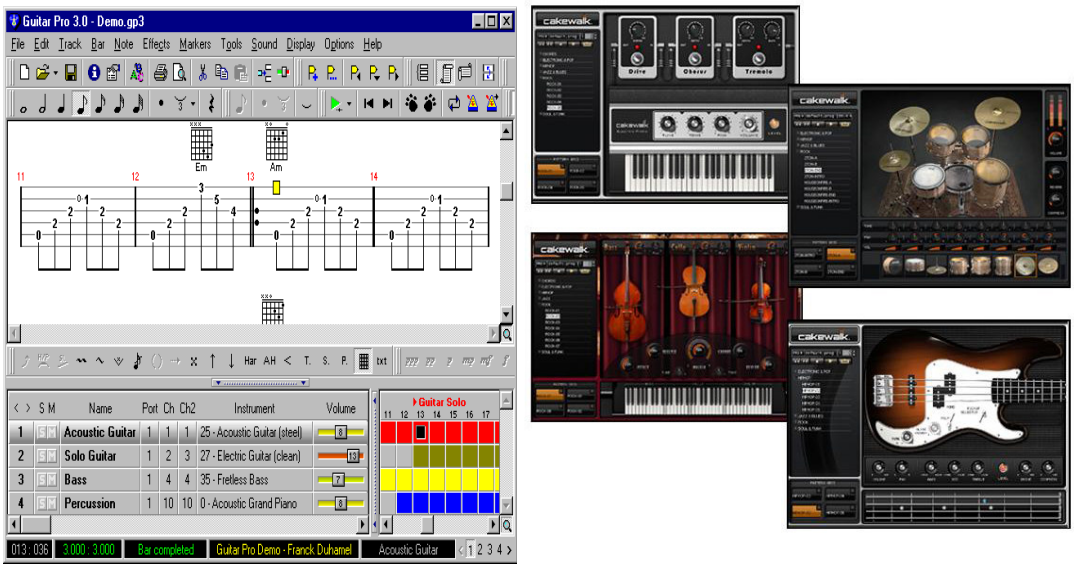


Figura 1.2: Ejemplos de guitar pro (izquierda) y Cakewalk Sonar (derecha)

Se impone por tanto, una apuesta firme y descarada por la introducción de la informática desde los niveles más inferiores de la educación musical, naturalmente bien administrada, junto al resto de materias, proporcionando y potenciando una gran capacidad de creación y expresión en los alumnos y profesores. Hay que dejar claro que será una alternativa que se nos ofrece como consecuencia del avance tecnológico de la sociedad, para permitir al alumno y profesor crear e interpretar música, con la misma facilidad y normalidad con la que se dibuja con pinceles. No debe interpretarse, en ningún caso, como una sustitución de algún método tradicional de didáctica musical, sino como se ha dicho es otra vía para poder desarrollar la creatividad.

Actualmente en el día a día de cualquier escuela, el profesorado se encuentra en una situación complicada, a veces angustiosa, ya que se enfrenta a una situación diferente y cambiante. Aunque está claro que ya no se trata de trasmitir o enseñar conocimientos musicales teniendo, tan solo, en cuenta los métodos musicales creados por pedagogos de música de gran prestigio, sino que es imprescindible actualizarse para entender y aprender las nuevas técnicas donde se analizarán y combinarán los métodos musicales con los avances tecnológicos que se están produciendo en nuestra sociedad, se tendrán en cuenta los diversos estilos musicales que conviven, las diferentes tecnologías, etc.

El desarrollo tecnológico es imparable, nos permite la publicación, almacenamiento y distribución de enormes cantidades de información en todos los ámbitos del saber. Por tanto, es tarea de todos revisar los diferentes campos del conocimiento para hacer una selección, lo más acorde a las exigencias y demandas de la sociedad, de los diferentes contenidos que forman el currículo. Algo está claro, no podemos seguir anclados en los conocimientos que nuestros antepasados nos han dado, la educación debe de dar la posibilidad a todos los individuos de poder crear y desarrollar sus propios aprendizajes, para poder elegir y seleccionar por medio de la imaginación las posibles respuestas que damos a los continuos cambios que se producen. La educación debe proporcionar la formación necesaria para que los alumnos puedan interpretar y reflexionar sobre la información que reciben.

Capítulo 2

2.1. Realidad aumentada: Definición

Como comentamos anteriormente, realidad aumentada es el término que se usa para definir una visión directa o indirecta de un entorno físico del mundo real, cuyos elementos se combinan con elementos virtuales para la creación de una realidad mixta a tiempo real. Consiste en un conjunto de dispositivos que añaden información virtual a la información física ya existente. Con la ayuda de la tecnología (por ejemplo, añadiendo la visión por computador y reconocimiento de objetos) la información sobre el mundo real alrededor del usuario se convierte en interactiva y digital.

La realidad aumentada adquiere presencia en el mundo científico a partir de los años 90 cuando la convergencia de capacidades de procesamiento de imágenes de video en tiempo real, sistemas gráficos por ordenador y nuevas tecnologías de visualización, hizo posible la obtención de una imagen virtual correctamente insertada con la visión del entorno tridimensional que rodea al usuario. En 1992 Tom Caudell crea el termino Realidad Aumentada y meses después Steven Feiner, Blair MacIntyre y Doree Seligmann desarrollan la primera utilización importante de un sistema de Realidad Aumentada en un prototipo, KARMA, presentado en la conferencia de la interfaz gráfica [13].

2.2. Realidad aumentada y realidad virtual

La principal diferencia entre la realidad virtual y la realidad aumentada reside en que mientras la primera crea un mundo virtual independiente de la realidad física, la realidad aumentada incluye los elementos virtuales dentro de dicha realidad física, consiguiendo una interacción directa con el usuario.

En los sistemas de realidad virtual el usuario esta completamente inmerso en el mundo artificial, lo cual le impide interactuar con objetos del mundo real. En contraposición, los sistemas de realidad aumentada no pretender aislar al usuario del mundo real, sino complementar este mediante objetos virtuales e imágenes generadas por ordenador. Los usuarios pueden interactuar con una mezcla de un mundo real y virtual de forma natural.

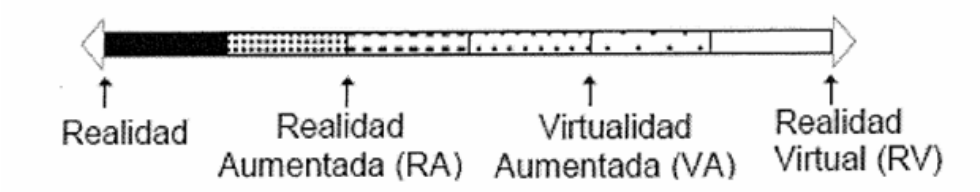


Figura 2.1: Representación simplificada del espacio realidad – virtualidad. Milgram & Kishino, 1994

En definitiva, se podría decir que los sistemas de realidad aumentada trasladan el ordenador al entorno de trabajo real del usuario, mientras que los sistemas de realidad virtual intentan trasladar el mundo al interior del ordenador. O dicho de otro modo, la realidad aumentada lleva la información dentro del mundo real del usuario en vez de llevar al usuario dentro del mundo virtual del ordenador [3].

2.3. Aspectos básicos de la realidad aumentada

Un sistema de realidad aumentada tiene las siguientes características [3]:

- Combinación de elementos virtuales y reales. La información digital se combina con la realidad.
- Procesamiento en tiempo real. Tanto los objetos que deben ser rastreados como la información sobre estos debe proporcionarse a tiempo real.
- Registro 3D. Los objetos reales y virtuales son registrados y alineados geométricamente entre ellos y dentro del espacio para darles coherencia espacial.

Por objeto virtual nos referimos a cualquier imagen que se genere en el dispositivo de visualización y se presente como parte de la realidad aumentada, esto

incluye texto, figuras, imágenes bidimensionales y modelos tridimensionales. Para crear aplicaciones de realidad aumentada necesitamos distinguir que hardware y software necesitamos:

2.3.1. Hardware

Los dispositivos de Realidad aumentada normalmente necesitan un elemento que capture las imágenes, un computador para realizar el procesamiento y un sistema de display para mostrar al usuario la información virtual que se añade a la real.

Los dos principales sistemas de "displays" empleados son la pantalla óptica transparente (*Optical See-through Display*) y la pantalla de mezcla de imágenes (*Video-mixed Display)* [13], que comentaremos más adelante. Tanto uno como el otro usan imágenes virtuales que se muestran al usuario mezclado con la realidad o bien proyectado directamente en la pantalla.

Los Sistemas de realidad aumentada modernos utilizan una o más de las siguientes tecnologías: cámaras digitales, sensores ópticos, acelerómetros, GPS, giroscopios, brújulas de estado sólido, RFID, etc. El Hardware de procesamiento de sonido se incluye en alguno de los sistemas de realidad aumentada. Los Sistemas de cámaras basadas en Realidad Aumentada requieren de una unidad CPU potente y gran cantidad de memoria RAM para procesar imágenes de dichas cámaras.

2.3.2. Software

El software necesario para realizar una aplicación de realidad aumentada es aquel que consiga fusiones coherentes de imágenes del mundo real, obtenidas con cámara, e imágenes virtuales en 3D. Ese mundo real debe ser situado, a partir de imágenes de la cámara, en un sistema de coordenadas. Dicho proceso se denomina registro de imágenes, del cual también hablaremos más adelante. Este proceso usa diferentes métodos de visión por ordenador, en su mayoría relacionados con el seguimiento de vídeo.

Algunos ejemplos de software que podemos utilizar para crear aplicaciones de realidad aumentada pueden ser la librería OSGART aplicable en C++ o el software Multi-plataforma ATOMIC Authoring Tool.

2.4. Funcionamiento

Como comentamos anteriormente, para que esta tecnología funcione necesitamos un dispositivo de adquisición de video, una computadora para procesar los datos adquiridos y visualizar el resultado y unos marcadores que son detectados por la cámara y nos permiten crear uno u otro modelo virtual. En tiempo real, la computadora procesa la perspectiva en la que el sujeto ve las cosas, y calcula e inserta elementos predeterminados en ella, haciéndolos parte de su realidad [6].

Figura 2.2: Representación del funcionamiento básico de la realidad aumentada

Para estudiar dicho funcionamiento, podemos dividirlo en tres partes fundamentales: visualización (salida), ubicación de objetos virtuales en el mundo real (registro) y métodos de interacción (entrada). A continuación se explican cada una de estas partes.

2.4.1. Visualización

La realidad aumentada se consigue con dispositivos de visualización, ya sean la típica pantalla de ordenador o móvil, o dispositivos más avanzados similares a los de la realidad virtual, como cascos o gafas, con pequeñas pantallas de cristal líquido. Estas pantallas deben funcionar cono si fuesen lentes transparentes para que pueda observarse el mundo real y permitir la introducción de objetos virtuales. Existen dos métodos de visualización [3]:

- El método directo o *optical see through display* el cual utiliza un espejo semireflectante para ver el mundo real a través de éste y al mismo tiempo reflejar las imágenes por ordenador.

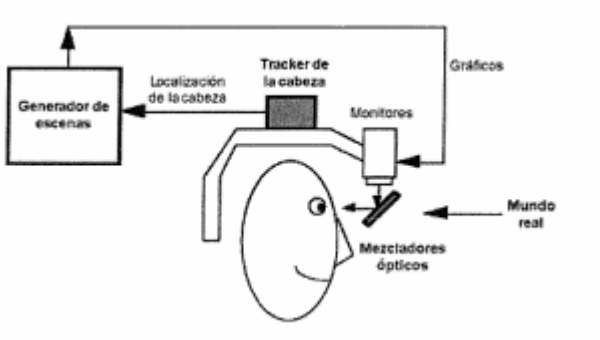


Figura 2.3: Método directo de visualización

- El método indirecto o *video–mixed display* el cual a diferencia del anterior, cubre totalmente la visión de la persona y la reemplaza por imágenes obtenidas por una cámara de vídeo mezcladas con los objetos virtuales.

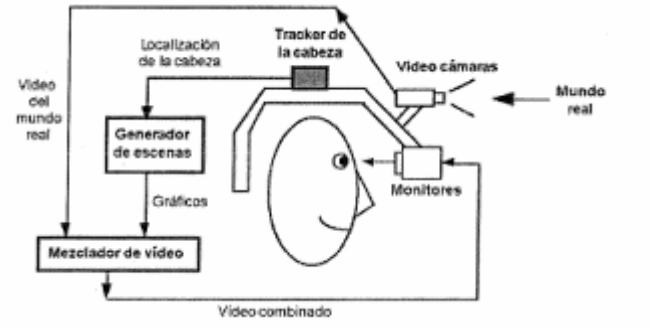


Figura 2.4: Método indirecto de visualización

El método directo se basa en la tecnología óptica, la cual utiliza visores montados en la cabeza del usuario que permiten la visión a través de ellos. Estos visores se basan en mezcladores ópticos que son parcialmente transparentes, por lo que el usuario puede ver a través de ellos, y reflexivos, de tal forma que reflejan la imagen que les llega del monitor.

El problema principal de este sistema es la cantidad de luz que dejamos pasar del exterior. Normalmente los mezcladores ópticos reducen bastante la luz incidente, de tal forma que actúan como gafas de sol cuando no están funcionando. El problema

de dejar pasar mucha luz es que los gráficos generados pueden resultar oscurecidos en un ambiente luminoso. El fenómeno inverso es igualmente problemático. Para solucionar este problema lo que se hace es filtrar la luz en función de su longitud de onda, de tal forma que si utilizamos un monitor monocromo, se deja pasar toda la luz del exterior exceptuando la longitud de onda del color que emite el monitor.

El método indirecto a diferencia del anterior se basa en la tecnología de video. Este sistema utiliza visores totalmente cerrados, por lo que contiene todos los elementos mencionados en el anterior método además de una o dos cámaras que se encargan de capturar la vista del mundo real que recibirá el usuario. De esta forma, el mezclador se encarga de integrar la imagen de video recibida por las cámaras y los modelos virtuales. La forma más fácil de realizar esta integración es mediante el croma key, donde los modelos virtuales se generan con un fondo uniforme de un color conocido y prohibido en los elementos generados, que el mezclador de video sustituye por la señal capturada por las cámaras.

Figura 2.5: Icuiti VR920 video see-through (Izqda) y optical see through binocular r(drcha)

Otra forma de implementar la realidad aumentada es mediante un monitor (que será la técnica que nosotros utilizaremos) donde el resultado se muestra directamente en la pantalla, sin necesidad de elementos extra que ponernos. Una variante es la representación en estéreo, mediante técnicas de combinación de las dos imágenes correspondientes a cada ojo en el monitor, siendo necesario entonces la utilización de gafas estereoscópicas por el usuario.

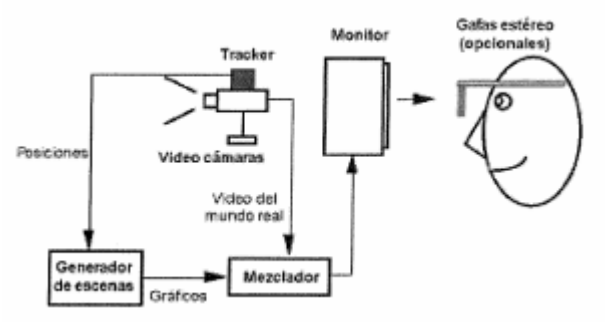


Figura 2.6: Aplicación de realidad aumentada sobre un monitor

En cuanto a ventajas o desventajas de utilizar una tecnología u otra, podemos decir que la tecnología óptica es más sencilla de implementar, ya que solo se tiene que preocupar de un flujo (el de los gráficos generados) mientras que en la tecnología de video existen dos (imagen real capturada y elementos generados) y cada uno de ellos con un retardo. Además, en la tecnología de video hay que tener en cuenta los problemas de resolución. Por otra parte, el sistema basado en la tecnología de video permite una fusión más flexible (por ejemplo se pueden eliminar elementos reales completamente) resultando una visualización mucho mas realista. El sistema de tecnología de video también puede corregir distorsiones generadas por las cámaras por medio de procesamiento automático, mientras que fabricar ópticas libres de distorsión es caro y complejo. Otra de las ventajas de la tecnología de video es que podemos utilizar métodos de registro de 3D utilizando técnicas de visión por computador, y es más sencillo hacer compatibles el enfoque y el contraste entre las imágenes sintéticas y reales.

2.4.2. Registro de objetos virtuales

Definimos un objeto virtual como cualquier imagen generada por computador que se genere en el dispositivo de visualización y se presente como parte de la realidad aumentada, ya sea texto, figuras, imágenes bidimensionales o tridimensionales [3].

Uno de los problemas de la realidad aumentada es “fusionar” el mundo virtual y real, que estos objetos virtuales puedan integrarse en el mundo real de tal forma que cuando el usuario se mueva los objetos parezcan conservar su posición al igual que los objetos reales mantienen su posición cuando el usuario cambia de sitio.

Para ello necesitamos alinear los sistemas de coordenadas de ambos mundos de forma que resulten coherentes.

Figura 2.7: Problema del registro.

Vemos en el apartado a) de la figura 2.7 la mezcla del mundo real (libro) con una gráfica de realidad aumentada (cubo). En el apartado b) de la misma figura el observador se desplaza, sin registro del objeto virtual (el cubo conserva su posición en la pantalla pero cambia su posición en el mundo real), mientras que en el apartado c) vemos el desplazamiento con registro del objeto virtual (el cubo conserva su posición aunque el observador se desplace, tal y como lo haría si fuera real).

Para conseguir que los objetos que los objetos virtuales conserven su posición, el sistema extrae de manera automática o manual información desde las imágenes reales que nos permitan determinar la posición del dispositivo de captura con respecto del mundo real. Este proceso se denomina registro de objetos virtuales. Con este proceso conseguimos como resultado final una matriz de transformación entre ambos sistemas de coordenadas.

Podemos resolver este problema con varias técnicas:

- Método de *tracking*: Con esta técnica asociamos la ubicación del objeto virtual a un sensor de posición. El *tracking* normalmente se realiza por hardware con dispositivos comerciales. Los localizadores (*trackers*) pueden ser ópticos, magnéticos o mecánicos.
- Reconocimiento de patrones: En esta técnica utilizamos técnicas de visión por ordenador para identificar objetos o imágenes reales y dibujar sobre ellos los objetos virtuales. Esta técnica esta asociada con el método indirecto ya que se utiliza con una cámara de video. En nuestro proyecto utilizaremos un

sistema basado en esta técnica llamado OSGART que ya comentaremos mas adelante.

- Si queremos utilizar aplicaciones de realidad aumentada en exteriores utilizaremos el sistema de posicionamiento global (GPS) para determinar la posición de la persona y generar las imágenes virtuales respecto de su posición.

2.4.3. Métodos de interacción

Conseguido ya que los objetos virtuales se presenten en el mundo real, debemos de crear formas de interactuar con ellos modificándolos, incluyendo otros, actualizándolos o cambiando alguna de sus propiedades.

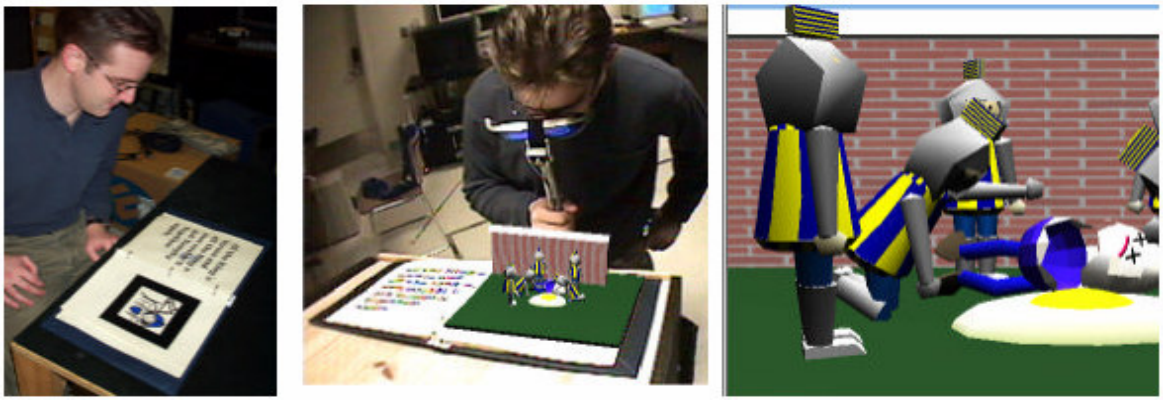
Hay varias formas de interactuar con los objetos virtuales en la realidad aumentada [2]:

- **Interfaces tangibles basadas en el uso de marcadores:**

  Las interfaces tangibles son aquellas interfaces [26] que aumentan el mundo real añadiéndole información digital para su complementación. Los sistemas de marcadores, como OSGART, permiten implementar este tipo de interfaces. En ellas, podemos manipular un elemento real con un marcador colocado y los resultados se reflejan en los movimientos del objeto virtual asociado.

  Ejemplos de este método son el Magic book [27] o el fingARtips [28]. El primero de ellos se trata de un libro con marcadores donde el usuario mediante una pantalla de mano visualiza modelos en 3D de lo que esta leyendo en dicho libro. Cuando el usuario visita las páginas del libro a través de la pantalla de mano se puede ver el contenido virtual superpuesto a las páginas reales. La novedad del libro es que el usuario puede "volar" por la escena e introducirse en ella como un elemento virtual más. Además, cuando varios usuarios miran en la misma página del libro y uno de ellos esta introducido en ella los demás le ven como otro avatar más de la realidad

virtual pudiendo interactuar con el.



a) Realidad física *b) Realidad aumentada* *c) Inmersión en la realidad virtual*

Figura 2.8: Usos del magic book

Otro de los ejemplos es el fingARtips. Esta aplicación tiene como objetivo desarrollar técnicas que permitan a una persona utilizar su mano para interactuar con el contenido virtual en la realidad aumentada.

Se profundiza en la forma en la que interactúan nuestras manos con el contenido virtual y en las técnicas necesarias para facilitar esta interacción. Por ejemplo, la percepción de la profundidad es una de las claves importantes que tenemos que tener en cuenta con el fin de permitir a los usuarios capturar o modificar el contenido virtual fácilmente.

FingARtips investiga técnicas básicas de interacción útiles para muchos tipos diferentes de aplicaciones. Por ejemplo, la aplicación de prueba es un escenario de planificación urbana, donde los usuarios pueden manipular fácilmente los modelos arquitectónicos con su mano y cambiarlos a su antojo.

a) Ejemplo básico *b) Aplicación de prueba*

Figura 2.9: Ejemplos de fingARtips

- **Interacción basada en movimiento corporal:**

Otra forma de interacción en los sistemas de realidad aumentada consiste en la detección y seguimiento del movimiento del cuerpo a algún miembro del mismo. Podemos interactuar con el seguimiento de manos y dedos, de la cabeza, o la orientación de dedos y ojos.

Existen distintas tecnologías para realizar el seguimiento de la posición y orientación de las manos, como *tracking* magnético, *tracking* inercial, sistemas de reconocimiento basados en visión, etc. Podemos poner como ejemplos de sistemas de interacción la implementación de un sistema de gestos realizada en 2003 por Kaiser E. et al. [25] Los gestos se basan en el uso de la mano y el análisis de la posición de la cabeza. Se implementaron cuatro tipos de movimientos: apuntar, empujar, voltear y rotar.

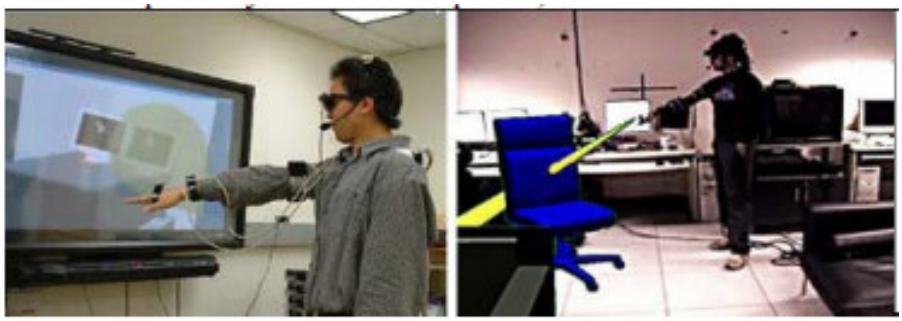


Figura 2.10: Técnica basada en la detección de gestos

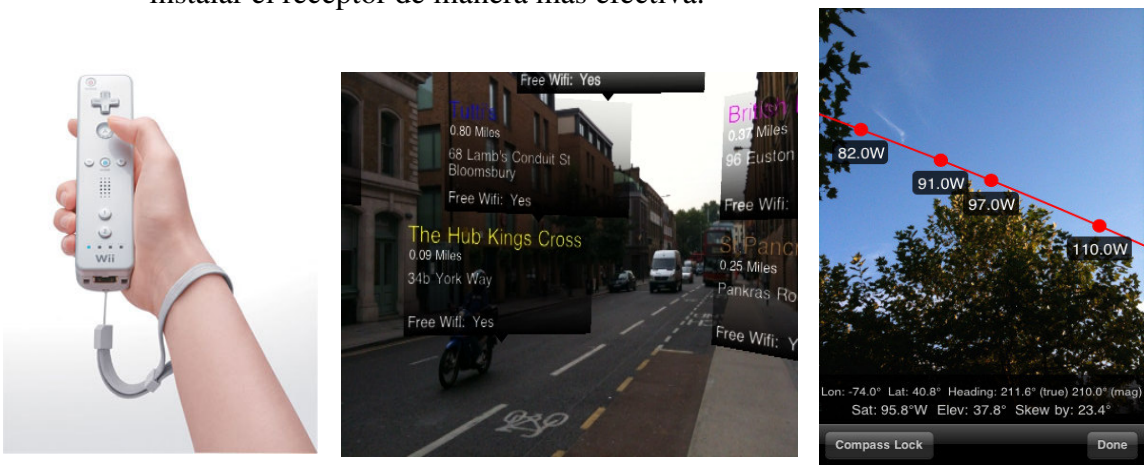
- **Interacción basada en dispositivos de bajo coste:**

Nos referimos a dispositivos de bajo coste cuando queremos nombrar dispositivos fácilmente accesibles en el mercado. Muchas técnicas de realidad aumentada se basan en dichos dispositivos incorporando sistemas de *tracking* y los utilizan como dispositivos de interacción. Ejemplos claros nos da Looser J. et al. en [29] con la videoconsola de Nintendo Wii, que aplica varias técnicas de interacción en realidad aumentada mediante el mando wiimote o el famoso iphone, que ya tiene varias aplicaciones de realidad aumentada cuyas utilidades son diversas: entretenimiento, utilidades geográficas, multimedia, etc.

La aplicación Worksnug [30] es una aplicación respaldada por el

mundo empresarial (por Plantronics), y cuyo objetivo es identificar los puntos Wi-Fi y lugares potenciales de trabajo – desde cafeterías hasta alquiler de oficinas para profesionales- con opiniones de usuarios, que abarcan desde la prestación de energía hasta la atmósfera, los niveles de ruido e incluso la calidad del café. Aunque actualmente sólo está disponible para Londres, están previstas en breve las versiones para San Francisco, Nueva York, Berlín, y Madrid.

Otro ejemplo es la aplicación Dishpointer [30]. Dishpointer comenzó como una hábil integración de Mapas de Google (Google Maps) y se ha convertido en una aplicación para el iPhone, y está también a unos días del lanzamiento para los teléfonos móviles Android. La versión para móviles de Dishpointer está diseñada para que los instaladores de radiodifusión por satélite o teléfonos móviles, apunten sus teléfonos hacia el cielo y reciban una superposición virtual de los satélites más cercanos, de modo que sepan como instalar el receptor de manera más efectiva.



a) Mando Wiimote *b)Aplicación Worksnug* *c)Aplicación Dishpointer*

Figura 2.11: Varias aplicaciones de AR con dispositivos de bajo coste

- **Interacción multimodal:**

Como su nombre indica, las técnicas de interacción multimodal se basan en varios métodos para interactuar con la realidad aumentada. Estos sistemas procesan métodos de entrada combinados (voz, lápiz, táctil, gestos, movimientos corporales, etc.) de forma coordinada con la salida multimedia

del sistema [31].

Un ejemplo de esto es la implementación de un sistema de videoconferencia basado en realidad aumentada con integración de gestos, seguimiento de movimiento de ojos y análisis de la dirección de la cabeza [32]. El usuario es capaz de interactuar con el sistema usando gestos de las manos con marcadores y movimientos de ojos y cabeza.

Figura 2.12: Sistema de videoconferencia basado en realidad aumentada

2.5. Aplicaciones de la realidad aumentada

La realidad aumentada es una tecnología en incesante evolución y cada vez es más conocida y aplicada a los distintos campos y ramas de la ciencia, la tecnología, la educación, el ejército o el entretenimiento. Las utilidades y aplicaciones en este momento son muchísimas, por lo que nos limitaremos a exponer las aplicaciones más importantes en los ámbitos más conocidos:

2.5.1. Aplicaciones militares

Una de las aplicaciones de la realidad aumentada en el ámbito militar es un dispositivo para la cabeza de los pilotos de aviones y helicópteros [33], donde sobre una pieza de vidrio transparente se proyecta toda la información necesaria para el piloto y que se superpone sobre la vista del exterior. Esto nos permite seguir simultáneamente lo que ocurre en el exterior del avión y tener a mano todos los datos del avión sin desviar la mirada. Este dispositivo nos da información básica de

navegación y de vuelo, además de poder rastrear objetos 3D permitiendo seguir la trayectoria de objetos en movimiento a tiempo real. Uno de los primeros aviones en utilizar este dispositivo fue el TSR – 2 (1965)

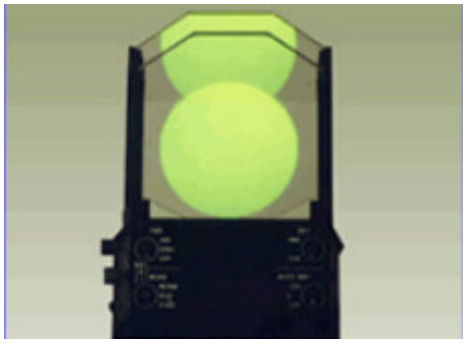


Figura 2.13: Dispositivo de realidad aumentada sobre pantalla de vidrio

2.5.2. Aplicaciones en el campo de la medicina

La aplicación de realidad aumentada en este campo se centra en la cirugía mínimamente invasiva. Es una aplicación que superpone en tiempo real la reconstrucción 3D de las estructuras internas del paciente sobre la imagen de video del mismo [3]. Esto supone una ayuda para el cirujano, puesto que ve en todo momento como es la estructura interna del paciente, pudiendo planificar mejor la operación. Además supone una reducción en los costes de la operación y en el tiempo invertido en ella.

Figura 2.14: Visualización de estructuras 3D en la cabeza del paciente

Esta técnica ya se esta utilizando rutinariamente para 1 ó 2 casos semanales y la metodología es la siguiente:

- Registro de los datos del paciente mediante métodos no invasivos (resonancia, tomografía axial, etc)
- Se realiza el modelo 3D
- Se visualiza en la pantalla de resultados las imágenes reales con las provenientes de la resonancia

2.5.3. Aplicaciones en diseño y producción

Otra de las industrias en aplicar la realidad aumentada en su campo es la automotriz, ayudando a sus diseñadores e ingenieros a visualizar nuevos prototipos, al poder modificar a su antojo el prototipo sin necesidad de las tradicionales estructuras de arcilla u otros materiales [4]. También podemos simular las propiedades y respuestas físicas del auto de forma precisa y rápida. Compañías como Audi, BMV o Nissan han estado utilizando estos sistemas y han llegado a ahorraste en algunos proyectos hasta 10 millones de euros.

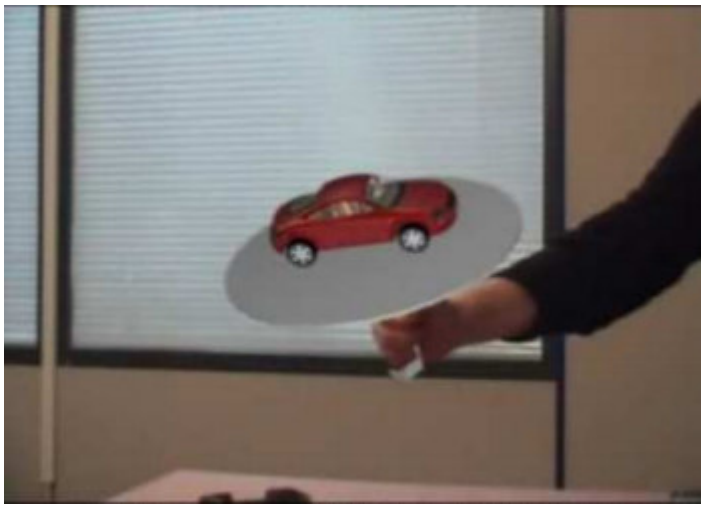


Figura 2.15: Modelo 3D de un diseño de automóvil

Figura 2.16: Comprobación de las características físicas del automóvil mediante RA

En la figura 2.15 vemos un modelo en 3D de un automóvil para el estudio de su diseño y sus características. En la figura 2.16 vemos como se pueden comprobar las características físicas del automóvil mediante nuestro modelo virtual. Un golpe seco en la mesa hace que ésta tiemble, generando efectos físicos realistas en nuestro auto virtual y podemos observar como responde éste al estímulo. También se puede utilizar la realidad aumentada de forma similar a como se utiliza en la cirugía, utilizando ésta para ayudar al mecánico a visualizar correctamente la parte dañada de un automóvil o en tareas de mantenimiento del mismo.

2.5.4. Aplicaciones a la educación

En este campo es donde la realidad aumentada nos interesa y donde se puede conseguir y se han conseguido aplicaciones muy directas para mejorar el estudio del alumno. Los alumnos aprender directamente interactuando con los elementos virtuales. La aplicaciones más comunes de la realidad aumentada en la educación son los libros basados en RA. Estos libros tienen la posibilidad de ver en 3D los elementos sobre los que se está estudiando e interactuar y modificarlos, viendo su evolución, cambiándolos, etc. Y en general aprendiendo de ellos. Varios ejemplos son la visualización de un volcán en erupción, los planetas o los dinosaurios. Las posibilidades en ese sentido son infinitas y nosotros aprovecharemos esto para crear otra aplicación educativa para el aprendizaje musical.

a) Volcán en erupción en RA *b) Dinosaurio en RA*

Figura 2.17: Aplicaciones de la realidad aumentada a la educación

También podemos conseguir que el alumno aprenda las partes del cuerpo sobreponiéndolas en nuestro propio cuerpo para una identificación más rápida y directa, En el mundo educativo van saliendo ya aplicaciones, una de ellas es learnAR [34]. Colocando un marcador delante de la cámara se superpone la animación correspondiente en la pantalla y en el aula un profesor puede emplearlo para realizar una demostración a toda la clase proyectando la imagen en una pizarra digital o puede ser utilizado por los alumnos a través de sus ordenadores ya sea en el aula o en su casa.

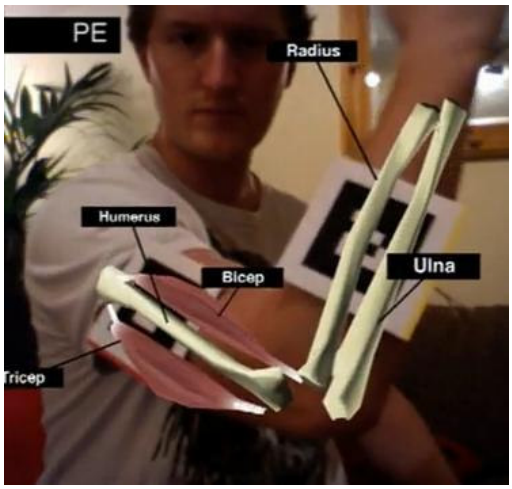


Figura 2.18: Aplicación learnAR mostrando huesos y músculos del brazo

Actualmente la mayoría de proyectos educativos basados en realidad aumentada se usan en museos, exposiciones, parques temáticos, etc. Pero no en las aulas debido al todavía caro coste de dichas aplicaciones.

2.5.5. Aplicaciones de entretenimiento

La realidad aumentada aplicada al mundo del entretenimiento nos lleva

principalmente a un campo de acción: los videojuegos. Gracias a la realidad aumentada el usuario salta la barrera virtual que le separa del videojuego y se sumerge en el mismo, siendo parte directa del desarrollo de su aventura. Ejemplos de esto es la consola wii, como mencionamos anteriormente, que a través del registro del movimiento tridimensional de un control (wiimote) hace que el usuario interactué con los elementos de la pantalla [6]. El jugador no será un "espectador" sentado en el sillón sino que participará en el mundo que se le presenta.

Figura 2.19: Aplicación de wii sports golf

Otro ejemplo de videojuego de realidad aumentada es el Sony Eyetoy. Eyetoy [6] es un conjunto de cámara y software desarrollado por Sony para la Playstation II y que se encuentra en proceso de actualización para la consola Playstation III. El videojuego consta de una cámara que captura la imagen del jugador, la recorta y la pone en la pantalla dentro del juego. Cuando el juego comienza, el software registra los movimientos del usuario y le permite interactuar con los elementos virtuales preprogramados.

a) Imagen del videojuego Eyetoy *b) Cámara Eyetoy*

Figura 2.20: Juego Eyetoy para playstation II

Una aplicación para las videoconsolas portables como la PSP es el juego *Invizimals* [35]. *Invizimal*s triunfó en el pasado E3 2009 y está levantando un gran número de buenas críticas por lo original de su propuesta.

Se trata de un título que aplica la realidad aumentada para plagar las casas de unos pequeños monstruos llamados Invizimals. Gracias a la realidad aumentada, el jugador es capaz de ver los monstruos, aparentemente invisibles, que habitan a su alrededor con la ayuda de la cámara Go!Cam y una trampa que se incluirán con el juego.

Figura 2.21: Muestra del videojuego Invizimals

Plantea un sistema de juego muy sencillo: el usuario debe capturar una serie de curiosas criaturas, 123 para ser precisos, siguiendo las instrucciones de caza de cada uno de los monstruos. Lo novedoso de Invizimals es que dichas criaturas son imperceptibles a simple vista y el jugador sólo puede localizarlas gracias a suGo!Cam para PSP, interactuando directamente con la realidad física y encontrando dichos seres en tu habitación, en tu salón, en el parque…

2.5.6. Otras aplicaciones

La realidad aumentada esta llegando a realizar aplicaciones para diversas utilidades, debido a su auge en los últimos años y la utilidad y reducción de costes y tiempo que ella trae consigo. Además de los ejemplos que se han nombrado en este capítulo, tanto en este apartado como en el anterior, se han realizado aplicaciones por ejemplo para los sistemas de navegación, cálculo de rutas de robots [6], visualización y anotación o incluso en la televisión [4]. En arquitectura se está empezando a

utilizar también debido a la obvia utilidad de poder visualizar y modificar los diseños de edificios y obras sin necesidad de maquetas ni figuras que simulen dicho edificio. En diseño de interiores se aplica de forma parecida pudiendo visualizar el salón de nuestra casa con unos muebles u otros, con distintas cortinas, distinta distribución de los elementos, etc.

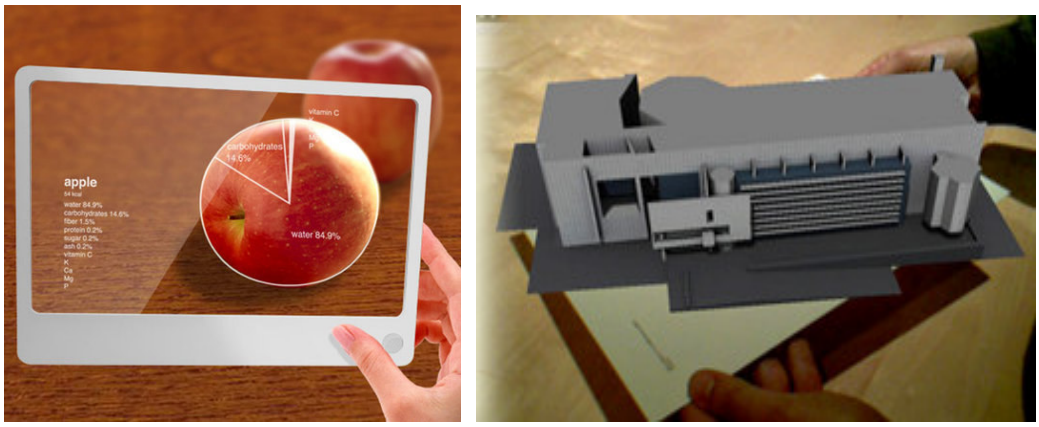


Figura 2.22: Ejemplos de RA para anotaciones extra (izqda) y arquitectura (drcha)

Otro campo que se está aprovechando de esta tecnología es la arqueología [33]. Mediante un software de realidad aumentada conseguimos ver un yacimiento o un edificio antiguo derruido como fue en los orígenes, dándonos una idea mucho más directa de cómo eran esos edificios y como se destruyeron, y estudiar de una forma más profunda como vivían las gentes que lo habitaban. Podíamos seguir nombrando aplicaciones de realidad aumentada, pero en este momento las posibilidades de realizar aplicaciones con esta tecnología son infinitas, por lo que nombramos estos ejemplos para hacer una idea de hasta donde podemos llegar y las posibilidades que ésta nos ofrece.

2.5.7. Aplicaciones futuras

En el futuro podríamos encontrar aplicaciones de este estilo [13]:

- Aplicaciones de multimedia mejoradas, como pseudo pantallas holográficas virtuales o "holodecks" virtuales (que permiten imágenes generadas por ordenador para interactuar con artistas en vivo y la audiencia).
- Conferencias virtuales también en estilo "holodeck".

- Sustitución de pantallas de navegación del coche: inserción de la información directamente en el medio ambiente. Por ejemplo, las líneas de guía directamente en la carretera.
- Con los sistemas de RA se puede entrar en el mercado de masas, viendo los letreros virtualmente, carteles, señales de tráfico, las decoraciones de Navidad, las torres de publicidad y mucho más.
- Cualquier dispositivo físico que actualmente se produce para ayudar en tareas orientadas a datos (como el reloj, la radio, PC, fecha de llegada / salida de un vuelo, una cotización, PDA, carteles informativos / folletos, los sistemas de navegación para automóviles, etc.) podrían ser sustituidos por dispositivos virtuales.

Capítulo 3

3.1. Librería OSGART

OSGART es un conjunto de librerías que simplifica el desarrollo de la realidad aumentada o aplicaciones de realidad mixta, combinando la biblioteca ARToolKit [21] con OpenSceneGraph [9]. OSGART no actúa solo como nodekit, sino que dispone de 3 funciones principales:

- Integración de alto nivel de entrada de vídeo
- Registro espacial (marcadores o seguidores múltiples)
- Registro fotométrico (oclusiones y sombras)

Con OSGART obtenemos todos los beneficios de las características de OpenSceneGraph (procesador de alta calidad, carga de varios tipos de archivo, etc.) directamente en realidad aumentada. Al igual que con ARTooKit, podemos desarrollar así aplicaciones y prototipos interactivos para interacciones con el usuario.

El éxito de ARToolKit ha sido en gran parte por su simplicidad, disponibilidad, y homogeneidad, y no por sus posibilidades de programación de alto nivel. Por lo tanto, queremos reproducir esta idea en otra generación de caja de herramientas como es OSGART.

En primer lugar cambiar GLUT por OpenSceneGraph nos da un nivel más alto de calidad gráfica, llegando a la calidad gráfica que podemos ver en los juegos estándar. En segundo lugar, la versión básica del juego de herramientas también es gratuita y se puede utilizar fácilmente como lo pueden hacer con ARToolKit. En tercer lugar, el kit de herramientas se mantiene en la misma idea simple de "todo-en-uno" con principios simples para desarrollar una aplicación (creación de un vídeo, inicializar perseguidor, etc.)Los usuarios de forma natural pueden desarrollar la aplicación en C + +, pero también tienen acceso a Python, Lua, o Ruby (mediante el osgBindings o módulo osgIntrospection). Y, por último, nuestra caja de herramientas ha sido diseñada para ir más allá de la posibilidad de un kit de

herramientas de AR por proporcionar servicios de alto nivel de vídeo, modular registro espacial dinámico y registros visuales de alta calidad.

Características:

- Diseño Orientado a Objetos.
- Soporte de múltiples entradas de vídeo.
- Integración de objeto de vídeo de alto nivel (fondo de vídeo, planos, vallas, etc.)
- Video concepto de sombreado (GLSL)
- Soporte de múltiples funciones basadas en marcadores naturales o tecnologías de seguimiento (ARToolKit, ARToolKit + +, arttag, bazar, etc)
- Interfaz de programación de aplicaciones en C + +, pero también en Python, Lua, Ruby, C # y Java (ya sea a través osgBindings o osgIntrospection)
- Fácil desarrollo de aplicaciones interactivas.
- Utilización de todas las posibilidades de OpenSceneGraph (3D de alta calidad de representación, las estadísticas, la exportación de 3DS Max / MAYA, Soporte de Nodekits (OpenAL, ReplicantBody), de múltiples plataformas, etc)

3.2. Funcionamiento

3.2.1.ARToolKit

Las aplicaciones realizadas con la librería OSGART se basan en el funcionamiento de la librería ARToolKit, cuyo patrón de funcionamiento es la video-detección de marcadores y la posterior creación de modelos virtuales 3D superponiéndolos en la imagen real que tenemos, visualizando la realidad física y al objeto virtual. Los marcadores utilizados por la librería se componen de un cuadrado negro con un cuadrado blanco en su centro donde se dibuja un elemento sencillo que reconoce la aplicación y donde aparece el objeto virtual. El software diferenciará cada marca por el dibujo pintado en su interior y mostrará el modelo virtual 3D que haya asociado a cada una de ellas.

Figura 3.1: Marcador de ejemplo de la librería ARtoolkit

Cuando la aplicación detecta el marcador en la señal de video, estudia la posición, orientación y tamaño del marcador visual y calcula la posición y orientación relativa de la cámara respecto al marcador. Después de esto, utiliza esa información para dibujar el objeto 3D sobre el marcador mediante OpenSceneGraph. Estas librerías lo que hacen es decirle a ARToolKit como mostrar un modelo tridimensional y el objeto aparece sobre el marcador en la posición, orientación y tamaño correspondiente al punto de vista de la cámara, siendo totalmente coherente con el entorno real que le rodea.

Figura 3.2: Proceso de trabajo de una aplicación ARToolKit

En la figura 3.11 podemos observar un diagrama donde se explica el funcionamiento de ARtoolkit. Podemos dividirlo en los siguientes pasos [6]:

1. Obtenemos la señal de video en vivo del mundo real mediante nuestra cámara.

2. Establecemos un valor de umbral, de tal forma que los píxeles de la señal capturada que lo superen se transformen en negro y los que no en blanco.
3. Se buscan y localizan todos los marcos negros de marcadores de la imagen (En verdad cuando umbralizamos la imagen el marco aparece blanco y el cuadrado blanco aparece negro).
4. Se compara el interior del marco con los dibujos de marcadores guardados en la memoria.
5. Si coincide con algún marcador guardado, se utiliza la información de tamaño y orientación asociado al patrón reconocido para compararlo con el que se ha detectado, y así se calcula la posición y orientación relativas de la cámara en relación al marcador y se guarda la información.
6. Con dicha información se establece la posición y orientación de la cámara virtual en el espacio, lo que permite al software determinar la perspectiva correcta para el elemento que se agregará.

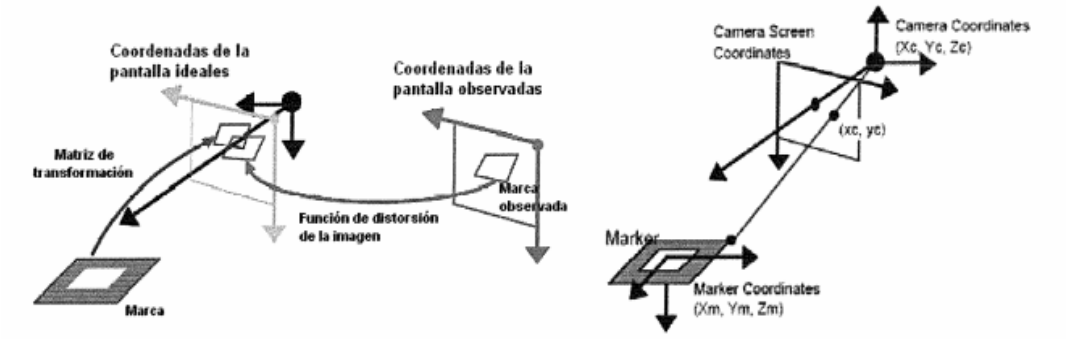


Figura 3.3: Sistemas de coordenadas de ARtoolkit y cálculo de las coordenadas de la cámara con respecto al marcador

7. Se dibuja el modelo virtual sobre el patrón visual, se renderiza y se muestra la imagen resultante. Ésta contiene la imagen del mundo real y del objeto virtual superpuesto, alineado en perfecta posición con el patrón.
8. Ininterrumpidamente se repite el proceso para cada fotograma hasta que termine la aplicación.

Todo este proceso se puede llevar a cabo utilizando los ejemplos incluidos en ARtoolkit.

Existen algunas limitaciones en los sistemas de realidad aumentada basados en visión por computador como ARtoolkit. Una de ellas es que el objeto virtual solo

aparece cuando la marca se visualiza completamente en la pantalla, de lo contrario no aparece nada, limitando el tamaño y los movimientos que realizamos con la marca, además de no poder tapar con el dedo ni con ningún otro elemento la marca puesto que el objeto virtual desaparecería.

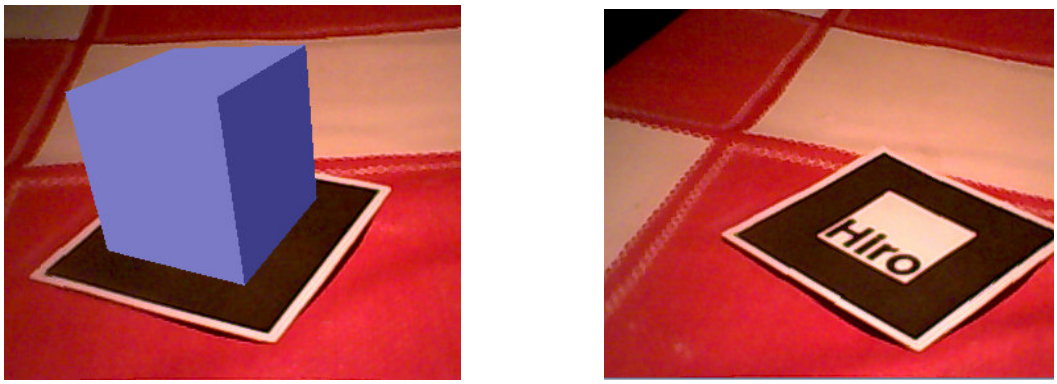


Figura 3.4: Ejemplo de aplicación del ejemplo SimpleTest para comprobar la visualización del marcador

Como vemos en la figura 3.13, en la imagen de la izquierda se ve totalmente el marcador, por lo que el objeto virtual (cubo) aparece en la imagen, En cambio, en la imagen de la derecha, la esquina superior derecha del marcador no se ve, por lo tanto la aplicación no detecta la marca y el objeto virtual no aparece.

El tamaño además nos dice que mientras más grande sea el marcador, detectaremos la marca a mayor distancia. En la figura 3.14 vemos un gráfico mostrándonos el rango efectivo de detección según el tamaño del patrón [21]. El dibujo realizado en el cuadro blanco también nos limita la distancia de detección, siendo más fácil de detectar cuanto más simple sea el marcador. La desventaja que tiene esto es que marcadores simples pero parecidos pueden confundirse dentro de la aplicación y mostrar objetos virtuales que en verdad no corresponden a esa marca.

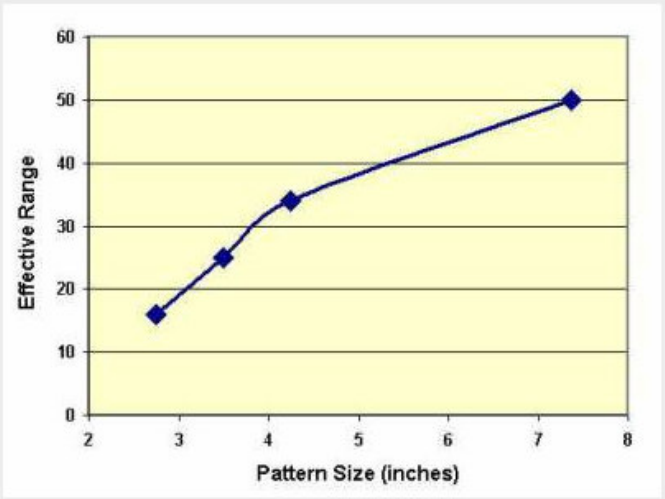


Figura 3.5: Rango efectivo de detección según el tamaño del patrón

Una última limitación que en nuestro caso nos ha dado más de un quebradero de cabeza pero al final resultaba ser algo muy simple, es la de la cantidad de luz que necesita detectar la cámara para poder reconocer el patrón. Como describimos anteriormente, se establece un umbral y se transforman los píxeles a negro o blanco según superen dicho umbral o no. El problema surge cuando la falta de luz hace que todos los píxeles se transformen a blanco, por lo que la aplicación no detecta el marcador, y no podemos hacer que aparezca el modelo virtual.

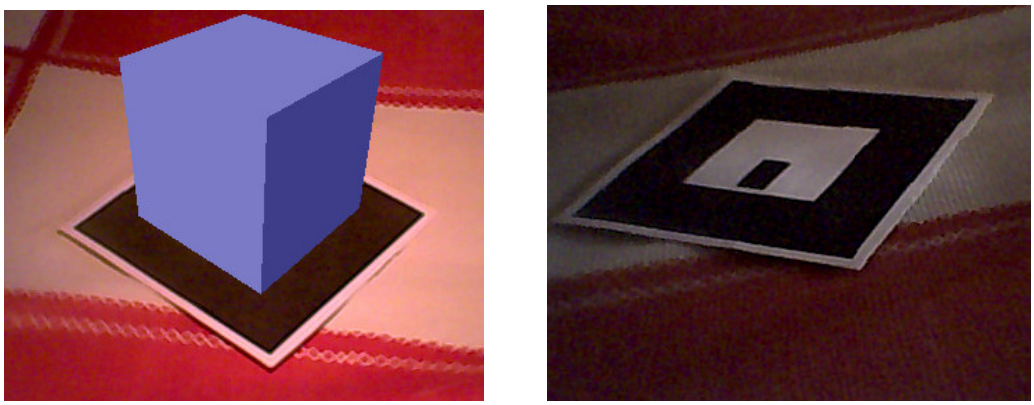


Figura 3.6: Ejemplo de la aplicación simpleTest para comprobar el efecto de la luz incidente en la cámara

En la figura 3.15 podemos observar como en la imagen de la izquierda hay luz suficiente para que la aplicación reconozca el marcador, mientras que en la imagen de la derecha aunque se visualice la totalidad de la marca, la luz incidente en la cámara no es suficiente para que los píxeles claros superen el umbral y se conviertan a negro.

3.2.2. OpenSceneGraph

El OpenSceneGraph es una herramienta gráfica multiplataforma para la elaboración de gráficos de alto rendimiento de aplicaciones tales como simuladores de vuelo, juegos, realidad virtual y visualización científica. Se basa en el concepto de un Scenegraph, proporcionando un marco orientado a objetos en la parte superior de OpenGL. Esto libera al desarrollador de la aplicación y optimización de gráficos de bajo nivel de las llamadas y proporciona muchas utilidades adicionales para el desarrollo rápido de aplicaciones gráficas.

Con OpenSceneGraph nuestro objetivo es hacer que los beneficios de la tecnología de la escena gráfica estén al alcance de todos los usuarios, tanto comerciales como no comerciales. Los puntos fuertes de OpenSceneGraph son su rendimiento, escalabilidad, portabilidad y las ganancias de productividad asociadas con el uso de un escenario gráfico con todas las funciones.

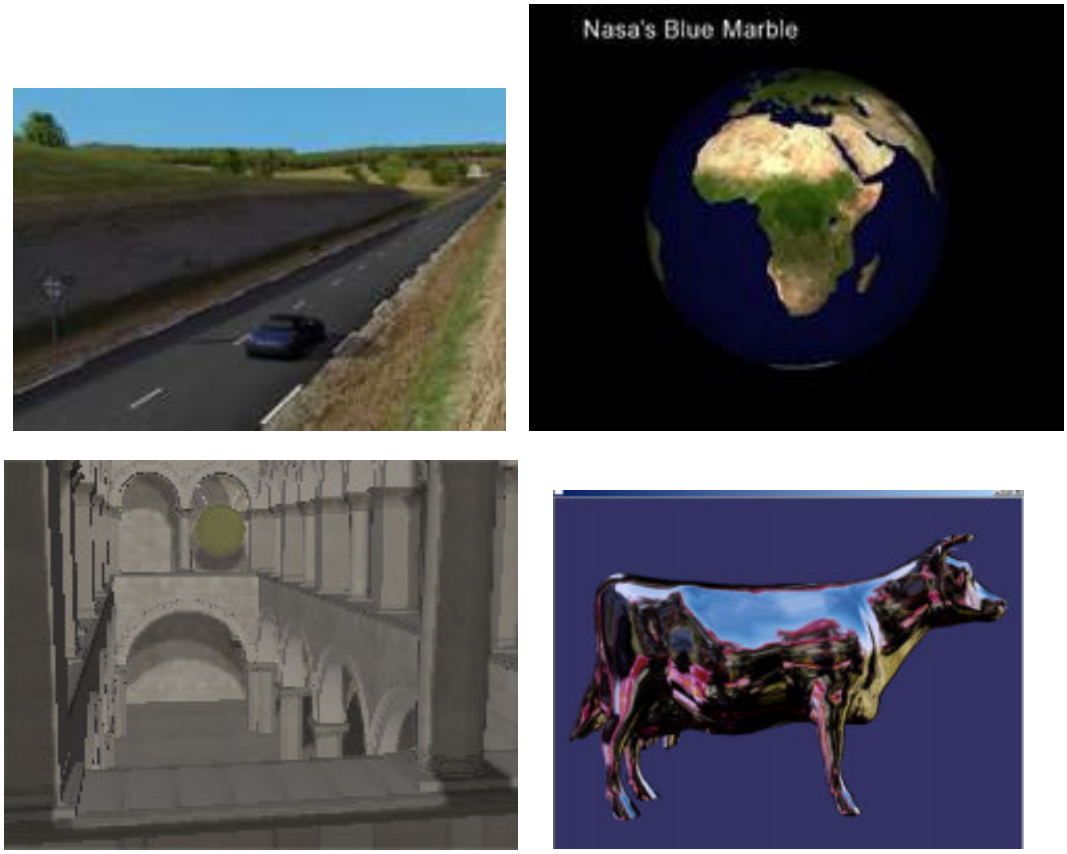


Figura 3.7. Algunas imágenes creadas por OpenSceneGraph

El objetivo de OSGART es integrar las imágenes creadas en OpenSceneGraph en la aplicación de realidad aumentada que realiza ARToolkit, para conseguir ver los modelos 3D creados, suponiendo una herramienta más potente que únicamente ARToolKit por las ventajas que hemos nombrado anteriormente.

A continuación vemos un esquema básico del funcionamiento de OpenSceneGraph, en la creación de un modelo 3D simple:

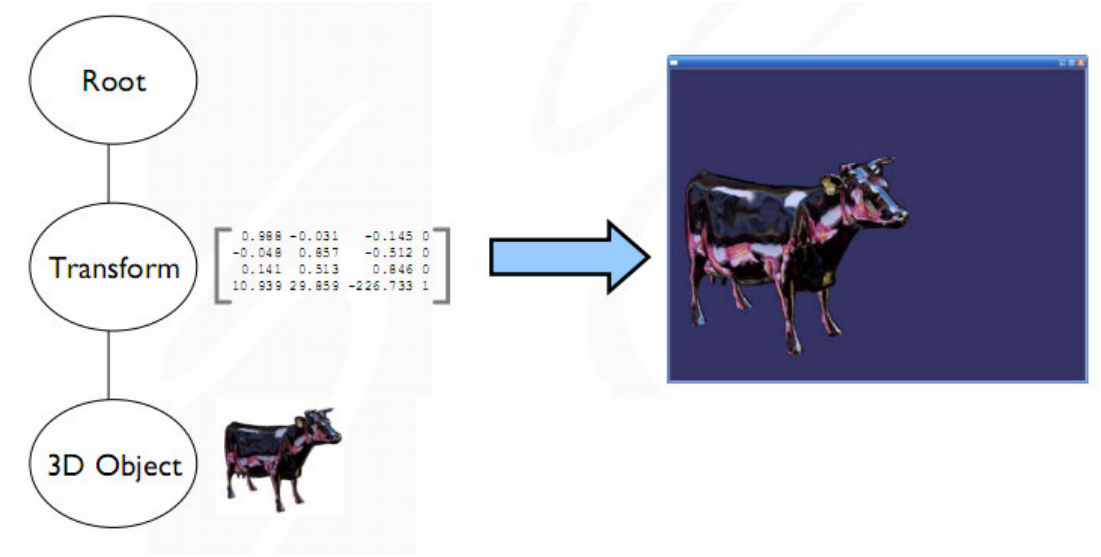


Figura 3.8: Funcionamiento básico OSG

3.3.3. OSGART

Por último vamos a ver como OSGART fusiona las dos librerías anteriormente mencionadas para crear la aplicación de RA.

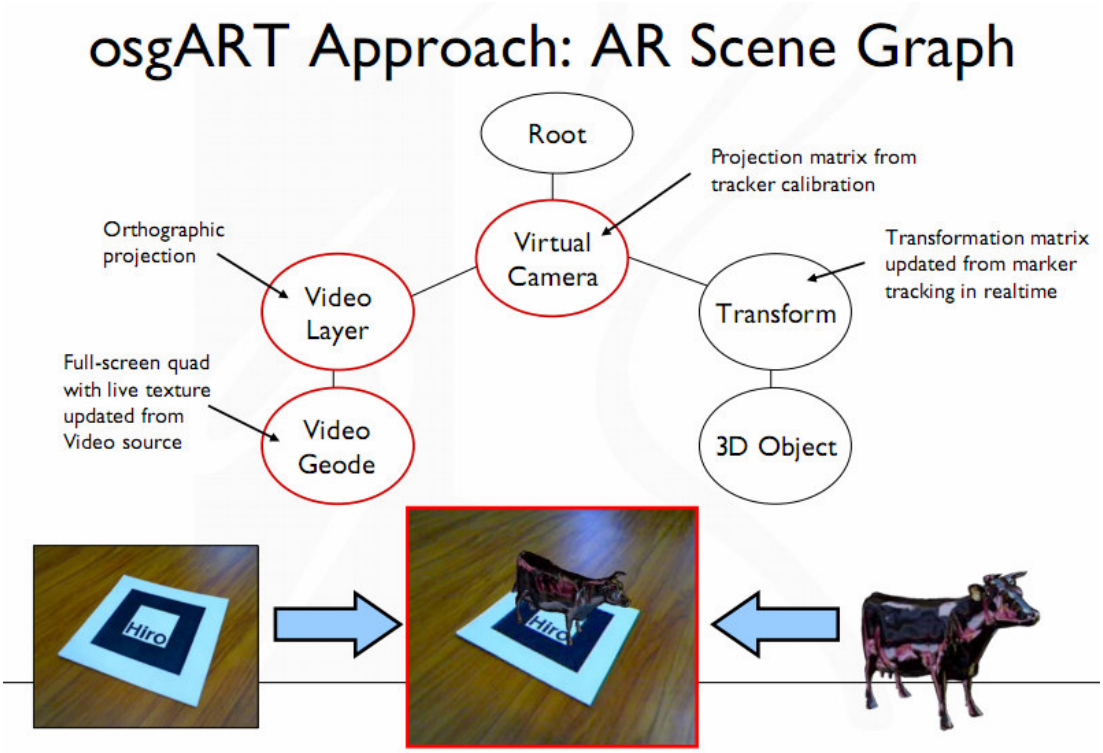


Figura 3.9: Funcionamiento de OSGART

Vemos en la figura como desde la raíz sale ahora la cámara, incluyendo la capa de vídeo y la matriz de transformación. La capa de vídeo crea la geoda y la matriz de

transformación crea el objeto 3D, consiguiendo así la aplicación de RA.

3.4. Archivos necesarios

Ya hemos visto el funcionamiento y los requisitos de una aplicación de ARtoolkit y las limitaciones que puede tener, y también hemos visto la herramienta OpenSceneGraph con la que diseñamos el modelo 3D. Ahora vamos a explicar cuales son los archivos que cargar para mostrar los modelos 3D, en los cuales se basa nuestra aplicación, y como cambiar dichos modelos por otros que seleccionemos.

En primer lugar necesitamos un archivo que nos reconozca en cada momento el patrón que estamos enfocando y los distinga de los anteriores. Este archivo es el archivo patt. El otro archivo que necesitamos es el que nos da todos los datos referentes al modelo 3D. Este archivo puede tener varias extensiones, (todas las que reconoce OpenSceneGraph). En nuestros archivos son archivos. 3ds. Cuando instalamos OSGART, dentro de la carpeta osgart_2.0_RC3, nos encontramos con la carpeta bin. En dicha carpeta están los ejecutables o los bin de las aplicaciones que desarrollamos. En dicha carpeta tenemos que tener otras dos: data y media. En data introducimos los archivos patt (son los archivos referentes a los marcadores) que posteriormente mencionaremos como se crean, el archivo camera_para.dat, que especifica los parámetros de la cámara, y en la carpeta media debemos tener los archivos .3ds. Si como es nuestro caso, introducimos sonido a la aplicación, debemos crear una tercera carpeta denominada Sonido, donde guardamos todos los archivos .wav de los sonidos que vamos a utilizar.

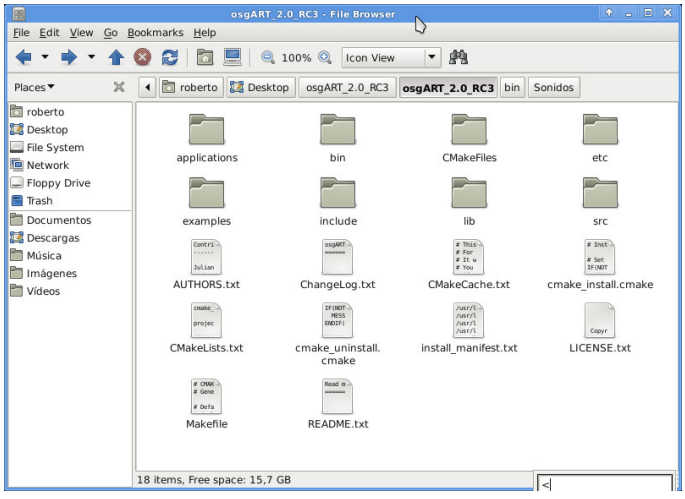


Figura 3.10: Carpeta osgART_2.0_RC3

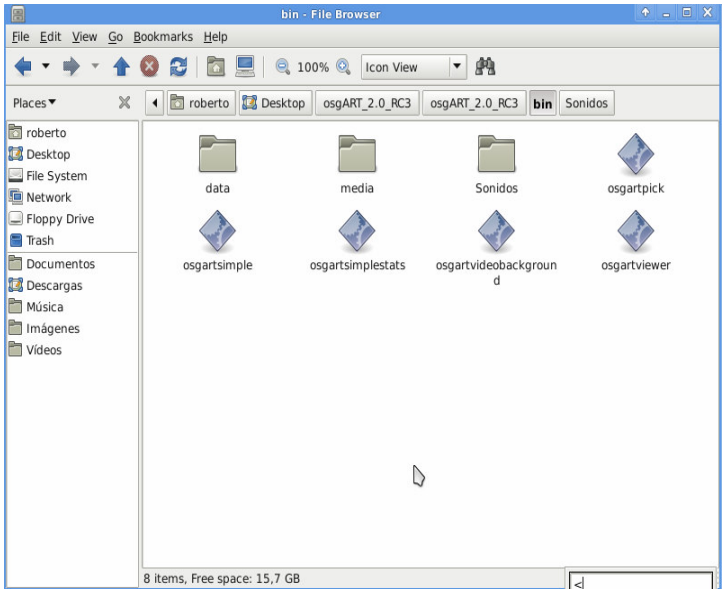


Figura 3.11: Carpeta bin

Situando estos archivos en los directorios adecuados no debe haber ningún problema a la hora de ejecutar la aplicación.

Capítulo 4

4.1. SDL

libSDL es el acrónimo de library Simple Directmedia Layer. SDL [20] es una librería multimedia multiplataforma. Está diseñada para proporcionar acceso de bajo nivel al audio, teclado, ratón, joystick y a los dispositivos de video 2D. Es una librería muy utilizada en diferentes tipos de aplicaciones entre ellas se encuentra software para reproducir video MPG, emuladores y un gran número juegos y adaptaciones de juegos entre plataformas.

SDL soporta Linux, Windows, Windows CE, BeOS, MacOS, MacOS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX y QNX. El código de la librería nos permite trabajar en aplicaciones para AmigaOS, Dreamcast, Atari, AIX, OSF/Tru64, RISC OS, SymbianOS y OS/2 pero estos sistemas no son sosportados oficialmente.

SDL está escrita en C pero trabaja nativamente con C++. Se puede utilizar en otros muchos lenguajes ya que existen adaptaciones que permiten hacerlo. Alguno de los lenguajes en los que se puede utilizar son Ada, C#, Eiffel, ERlang, Euphoria, Guile, Haskell, Java, Lisp, Lua, ML, Objetive C, Pascal, Perl, PHP, Pije, Pliant, Python, Ruby y Smalltalk.

SDL nos proporciona una capa de abstracción con todo lo necesario para crear aplicaciones multimedia y videojuegos. Está compuesta por subsistemas, y estos subsistemas son sostenidos por diferentes APIs que nos proporcionan acceso a las diferentes partes hardware.

En la siguiente figura podemos ver los distintos subsistemas de los que se compone:

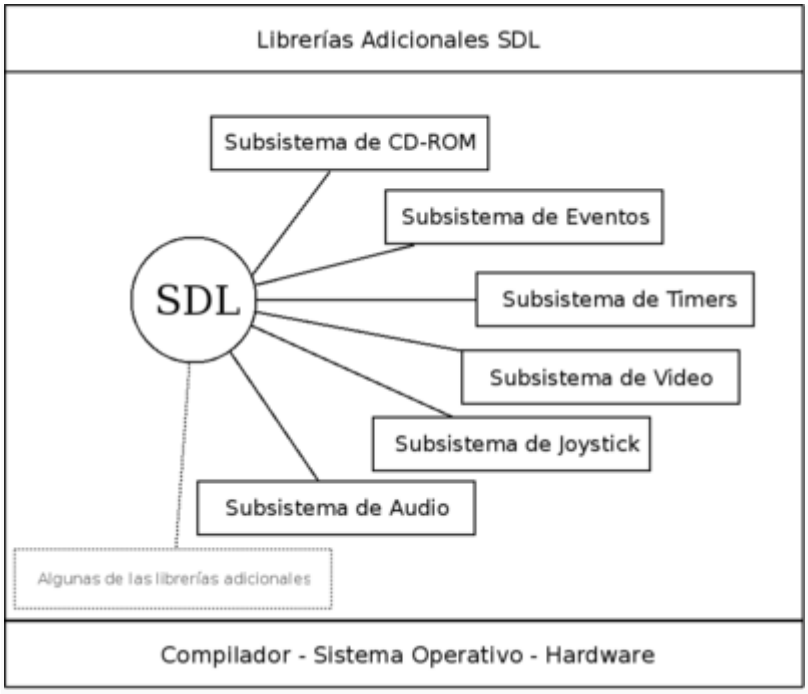


Figura 4.1. SDL y sus subsistemas

4.2. Composición de SDL

La librería SDL se compone de varios subsistemas, suficientes para la creación de videojuegos y aplicaciones multimedia. Estos subsistemas son:

- Vídeos y gráficos
- Eventos de entrada
- Sonido
- Manejo del CD ROM
- Timers
- Gestión de dispositivos
- Red

En nuestro caso solo queremos el sonido, por lo cual nos centraremos en el a lo largo de esta explicación.

4.2.1. Subsistema de sonido

Este es el aspecto más débil de SDL. La librería proporciona un API simple para averiguar las capacidades que nos proporciona la tarjeta de sonido y lo que se necesita saber para reproducir un sonido. Podemos iniciar el subsistema de sonido a 8 o 16 bits, mono o estéreo. El soporte para reproducir un sonido real es ínfimo, y para la tarea de reproducir sonidos deberemos de utilizar librerías auxiliares que nos proporcionen una manera más cómoda de trabajar.

Por las razones expuestas anteriormente nos vamos a apoyar en la librería auxiliar SDL_mixer, que nos proporciona un soporte y unas funciones para reproducir cualquier sonido de forma efectiva y realizar aplicaciones con él.

4.3. SDL_mixer

Como hemos mencionado el subsistema de audio es uno de los mas tediosos de utilizar de SDL. En nuestra ayuda acude la librería SDL_mixer para facilitarnos el manejo del sistema de sonido.

SDL_mixer es un complemento que mejora el subsistema de audio de SDL. Está preparada para manejar múltiples sonidos al mismo tiempo además de la música. Es más, si te sientes capaz, puede especificar la manera de mezclar la música y aplicar varios efectos (como el fade-out) en tu aplicación manejando punteros a funciones que realicen esta tarea.

SDL_mixer se encarga de realizar el mix o mezcla de canales de audio por nosotros de forma automática lo que nos ahorra el desarrollo de un sistema de mezclado. Los formatos con los que trabaja esta librería son *wav*, *mp3*, *midi*, *Ogg Vorbis*, *MOD*, *IT*, *S3M* y *VOC*. Como podrás observar es compatible con muchos de los formatos deseables.

Esta librería se basa en *chunks*. Un chunk no es más que un sonido producido por algún efecto concreto o un sonido en especial, como puede ser el golpeo de una

pelota en un juego de tenis o el choque de dos espadas en un juego de acción. Nosotros en nuestra aplicación lo que haremos básicamente será cargar los sonidos como chunks, y asignarlos a diferentes canales dentro de cada función correspondiente, sin mucha complicación por que no es lo que buscamos.

Capítulo 5

5.1. Implementación

En este capítulo nos centraremos en explicar el funcionamiento del código de nuestra aplicación y las modificaciones que realizamos para conseguir el resultado que nosotros queríamos, así como el entorno de desarrollo que elegimos y la tecnología utilizada.

5.2. Entorno software

El entorno elegido para desarrollar nuestra aplicación ha sido un entorno bajo el sistema operativo *Linux*. Se escogió este sistema operativo debido a que es el utilizado en los centros de educación, y para ampliar horizontes y expandir las aplicaciones de realidad aumentada a otro entorno diferente a *Windows*.

Todas las librerías mencionadas anteriormente serán utilizadas y son de libre distribución. Para la creación de los modelos 3D utilizamos el programa Blender, programa utilizado para el modelado, animación y creación de gráficos y modelos 3D, ayudados de [7] y [8].

De manera ordenada, el entorno elegido y las librerías y programas utilizados para el desarrollo de la aplicación han sido los siguientes:

- Sistema operativo Linux
- Lenguaje de programación C++
- Librería OpenSceneGraph
- Librería OSGART
- Librería SDL
- Librería ARtoolkit
- Programa Blender

5.3. Desarrollo de la aplicación

Nosotros hemos basado nuestra aplicación en el ejemplo *osgartsimple* pero toda aplicación básica de OSGART (incluido este ejemplo) tiene esta estructura básica:

5.3.1. Implementación básica

Para empezar implementamos la visión estándar de OSG, es decir, la ventana donde se modelan los objetos 3D:

```
#include <osgViewer/Viewer>
#include <osgViewer/ViewerEventHandlers>
int main(int argc, char* argv[])  {
// Create a viewer
osgViewer::Viewer viewer;
// Create a root node
osg::ref_ptr<osg::Group> root = new osg::Group;
// Attach root node to the viewer
viewer.setSceneData(root.get());
// Add relevant event handlers to the viewer
viewer.addEventHandler(new osgViewer::StatsHandler);
viewer.addEventHandler(new osgViewer::WindowSizeHandler);
viewer.addEventHandler(new osgViewer::ThreadingHandler);
viewer.addEventHandler(new osgViewer::HelpHandler);
// Run the viewer and exit the program when the viewer is closed
return viewer.run();
}
```

Con este código creamos la ventana simple que obtenemos con OSG. A continuación lo que vamos a hacer es cargar y configurar el plugin de video, crear una captura de video y enlazarla con scene-graph para poder realizar la aplicación de realidad mixta.

Para ello incluimos este código con el que añadimos el plugin de video:

```
 // Preload the video and tracker
int _video_id = osgART::PluginManager::getInstance()-
>load("osgart_video_artoolkit2");
// Load a video plugin.
osg::ref_ptr<osgART::Video> video =
dynamic_cast<osgART::Video*>(osgART::PluginManager::getInstance()-
>get(_video_id));
// Check if an instance of the video stream could be created
if (!video.valid()) {
// Without video an AR application can not work. Quit if none found.
osg::notify(osg::FATAL) << "Could not initialize video plugin!" <<
std::endl;
exit(-1);
}
// Open the video. This will not yet start the video stream but will
// get information about the format of the video which is essential
// for the connected tracker.
video->open();
```

Con este código añadimos y configuramos el plugin de video. Ahora vamos a añadir el video a tiempo real:

```
osg::Group* createImageBackground(osg::Image* video) {
osgART::VideoLayer* _layer = new osgART::VideoLayer();
_layer->setSize(*video);
osgART::VideoGeode* _geode = new
osgART::VideoGeode(osgART::VideoGeode::USE_TEXTURE_2D, video);
addTexturedQuad(*_geode, video->s(), video->t());
_layer->addChild(_geode);
return _layer;
}
```

Este código se escribe antes de la función principal. Ahora dentro de la función principal escribimos lo siguiente:

```
osg::ref_ptr<osg::Group> videoBackground =
createImageBackground(video.get());
videoBackground->getOrCreateStateSet()->setRenderBinDetails(0, "RenderBin");
root->addChild(videoBackground.get());
video->start();
```

Con esto ya tenemos creada la captura de video a tiempo real enlazada a scene-graph.

En este momento nuestro programa solo captura imágenes, pero no detecta nada. Ahora implementaremos el tracking de vídeo para que detecte los marcadores. En primer lugar cargamos el plugin de tracking o seguimiento y lo asociamos al plugin de video:

```
int _tracker_id = osgART::PluginManager::getInstance()-
>load("osgart_tracker_artoolkit2");
osg::ref_ptr<osgART::Tracker> tracker =
dynamic_cast<osgART::Tracker*>(osgART::PluginManager::getInstance()-
>get(_tracker_id));
if (!tracker.valid()) {
// Without tracker an AR application can not work. Quit if none found.
osg::notify(osg::FATAL) << "Could not initialize tracker plugin!" <<
std::endl;
exit(-1);
}
// get the tracker calibration object
osg::ref_ptr<osgART::Calibration> calibration = tracker-
>getOrCreateCalibration();
// load a calibration file
if (!calibration->load(std::string("data/camera_para.dat")))
{
// the calibration file was non-existing or couldnt be loaded
osg::notify(osg::FATAL) << "Non existing or incompatible calibration file"
<< std::endl;
exit(-1);
}
// set the image source for the tracker
tracker->setImage(video.get());
osgART::TrackerCallback::addOrSet(root.get(), tracker.get());
// create the virtual camera and add it to the scene
osg::ref_ptr<osg::Camera> cam = calibration->createCamera();
root->addChild(cam.get());
```

Nos fijamos que camera_para.dat es el archivo que anteriormente mencionamos y que nos da los parámetros de la cámara. El siguiente paso es cargar la marca y activarla, asociarla con un nodo de transformación y añadir el nodo de transformación al nodo de la cámara. La marca que cargaremos será la marca hiro y se especificará el patrón a cargar como patt.hiro, que debera encontrarse en la carpeta media como especificamos en el capítulo 3:

```
osg::ref_ptr<osgART::Marker> marker = tracker-
>addMarker("single;data/patt.hiro;80;0;0");
if (!marker.valid())
{
// Without marker an AR application can not work. Quit if none found.
osg::notify(osg::FATAL) << "Could not add marker!" << std::endl;
exit(-1);
}
marker->setActive(true);
osg::ref_ptr<osg::MatrixTransform> arTransform = new osg::MatrixTransform();
osgART::attachDefaultEventCallbacks(arTransform.get(), marker.get());
```

Con esto nuestro programa además de capturar el video a tiempo real ya realiza un tracking de nuestra marca. Si queremos que se imprima en la consola en todo momento la posición de la marca añadimos este código:

```
osgART::addEventCallback(arTransform.get(), new
osgART::MarkerDebugCallback(marker.get()));
```

Ahora vamos a añadir el objeto 3D al patrón asociado para completar la aplicación de realidad aumentada.

```
std::string filename = "media/guitarra.3ds";
//Load a content
      arTransform->addChild(osgDB::readNodeFile(filename));
      arTransform->getOrCreateStateSet()->setRenderBinDetails(100,
"RenderBin");

cam->addChild(arTransform.get());
```

Con esto ya tenemos cargado el modelo virtual de una guitarra. Para cargar múltiples modelos lo único que debemos hacer es poner distintos nombres a las variables:

```
osg::ref_ptr<osgART::Marker> markerA = tracker-
>addMarker("single;data/patt.guitarra;80;0;0");
osg::ref_ptr<osgART::Marker> markerB = tracker-
>addMarker("single;data/patt.piano;80;0;0");

if (!markerA.valid())
      {
      // Without marker an AR application can not work. Quit if none found.
            osg::notify(osg::FATAL) << "Could not add marker!" <<
std::endl;
            exit(-1);
      }
```

```
if (!markerB.valid())
	{
	// Without marker an AR application can not work. Quit if none found.
		osg::notify(osg::FATAL) << "Could not add marker!" <<
std::endl;
		exit(-1);
	}

	markerA->setActive(true);

	markerB->setActive(true);

	osg::ref_ptr<osg::MatrixTransform> arTransformA = new
osg::MatrixTransform();

osgART::attachDefaultEventCallbacks(arTransformA.get(),markerA.get());


	osg::ref_ptr<osg::MatrixTransform> arTransformB = new
osg::MatrixTransform();

osgART::attachDefaultEventCallbacks(arTransformB.get(),markerB.get());



std::string filenameA = "media/guitarra.3ds";

	arTransformA->addChild(osgDB::readNodeFile(filenameA));

	arTransformA->getOrCreateStateSet()->setRenderBinDetails(100,
"RenderBin");


	std::string filenameB = "media/piano.3ds";

	arTransformB->addChild(osgDB::readNodeFile(filenameB));

	arTransformB->getOrCreateStateSet()->setRenderBinDetails(100,
"RenderBin");


cam->addChild(arTransformA.get());

cam->addChild(arTransformB.get());
```

Ahora hemos cargado dos modelos distintos (guitarra y piano) con dos marcadores distintos(patt.guitarra y patt.piano), de la misma manera que antes pero especificando con A o B si se refiere a uno u a otro. En nuestra aplicación creamos de esta forma 16 marcas distintas, repitiendo este mismo proceso.

Con este código ya conseguimos crear una aplicación de realidad aumentada que reconozca distintos patrones y muestre distintos modelos al mismo tiempo. La figura 5.1 la implementación que hemos llevado a cabo hasta ahora:

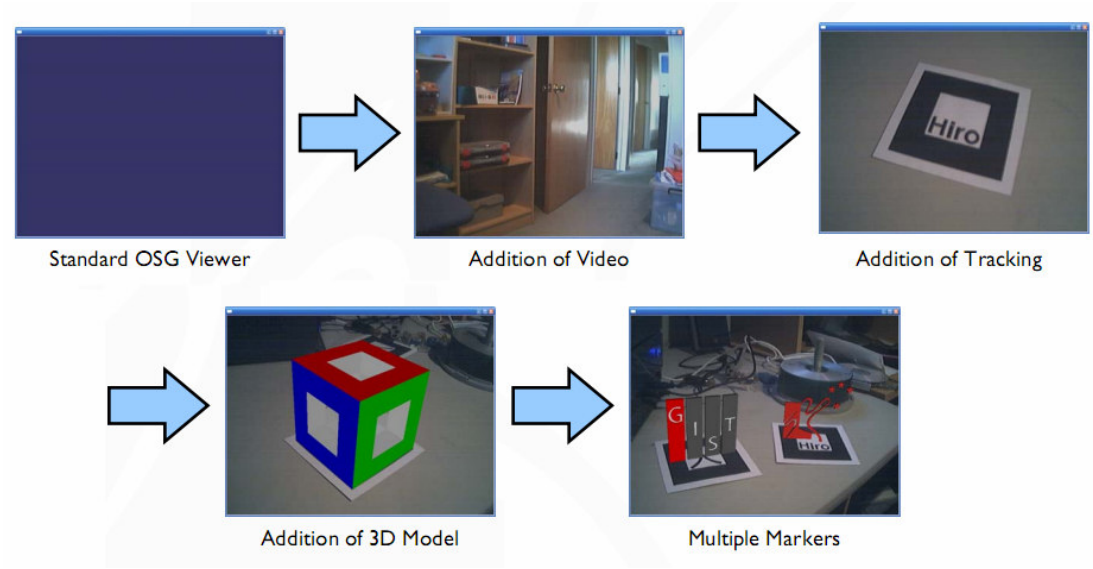


Figura 5.1. Esquema gráfico de la implementación

5.3.2. Adición de sonido

Ahora debemos utilizar la librería SDL y SDL_mixer para añadir el sonido a la aplicación. No buscamos en esta aplicación implementar una función de audio demasiado complicada, simplemente buscamos que se pueda reproducir el sonido, por lo que no profundizaremos mucho en esta librería.

El código que utilizamos para cargar los sonidos es el siguiente:

```
int Sonara()
{
    // Iniciamos el subsistema de video

    if(SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO) < 0) {

//      cerr << "No se pudo iniciar SDL: " << SDL_GetError() << endl;
        exit(1);
    }

    atexit(SDL_Quit);


    // Inicializamos la librerÃa SDL_Mixer

    if(Mix_OpenAudio(MIX_DEFAULT_FREQUENCY, MIX_DEFAULT_FORMAT,\
                  MIX_DEFAULT_CHANNELS, 4096) < 0) {

//      cerr << "Subsistema de Audio no disponible" << endl;
        exit(1);
    }

    // Cargamos un sonido
```

```
    Mix_Chunk *sonido1;

    sonido1 = Mix_LoadWAV("./Sonidos/guitarra.wav");

    if(sonido1 == NULL) {

//     cerr << "No se puede cargar el sonido" << endl;
       exit(1);

    }

    // Establecemos el volumen para el sonido

    int volumen1 = 100;

    Mix_VolumeChunk(sonido1, volumen1);


    // Creamos dos canales

    Mix_AllocateChannels(2);

    // Introducimos el sonido en uno de los canales
    // En el canal 1 con reproducciÃ³n infinita (-1)

    Mix_PlayChannel(1, sonido1, 0);

    // Variables auxiliares

   SDL_Event evento;
    SDL_EnableKeyRepeat(SDL_DEFAULT_REPEAT_DELAY,\
                    SDL_DEFAULT_REPEAT_INTERVAL);


    // Bucle finito

int i;
i==0;

    for( i==0;i++;i<1) {
                if(evento.key.keysym.sym == SDLK_ESCAPE) {
                 return 0;
             }
       }
}
```

Como mencionamos en el capítulo anterior, nos vamos a basar en la carga de chunks, siendo cada chunk cargado el sonido de cada instrumento. En el código se puede ver como inicializamos el sistema de video y la librería SDL_Mixer, creamos el chunk asignándole un sonido, le asignamos un volumen y un canal, y le reproducimos. La función es simple, y nosotros creamos catorce funciones en total, similares a la anterior (las siguientes se llamaban Sonarb, Sonarc, Sonard…) para cada sonido, y una función final que a continuación exponemos con la aplicación de banda musical, donde en vez de uno se cargan 6 instrumentos a la vez:

```
int Sonartodo()
{

   // Iniciamos el subsistema de video

    if(SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO) < 0) {

//     cerr << "No se pudo iniciar SDL: " << SDL_GetError() << endl;
       exit(1);
    }

    atexit(SDL_Quit);


    // Inicializamos la librerÃa SDL_Mixer

    if(Mix_OpenAudio(MIX_DEFAULT_FREQUENCY, MIX_DEFAULT_FORMAT,\
                  MIX_DEFAULT_CHANNELS, 4096) < 0) {

//     cerr << "Subsistema de Audio no disponible" << endl;
       exit(1);
    }

    // Cargamos un sonido

    Mix_Chunk *sonido15;
    Mix_Chunk *sonido16;
    Mix_Chunk *sonido17;
    Mix_Chunk *sonido18;
    Mix_Chunk *sonido19;
    Mix_Chunk *sonido20;

    sonido15 = Mix_LoadWAV("./Sonidos/bajo1.wav");
    sonido16 = Mix_LoadWAV("./Sonidos/piano1.wav");
    sonido17 = Mix_LoadWAV("./Sonidos/acustica1.wav");
    sonido18 = Mix_LoadWAV("./Sonidos/electrica1.wav");
    sonido19 = Mix_LoadWAV("./Sonidos/tambor1.wav");
    sonido20 = Mix_LoadWAV("./Sonidos/violin1.wav");


    if(sonido15 == NULL) {

//     cerr << "No se puede cargar el sonido" << endl;
       exit(1);

    }
    if(sonido16 == NULL) {

//     cerr << "No se puede cargar el sonido" << endl;
       exit(1);

    }
    if(sonido17 == NULL) {

//     cerr << "No se puede cargar el sonido" << endl;
       exit(1);

    }
    if(sonido18 == NULL) {

//     cerr << "No se puede cargar el sonido" << endl;
       exit(1);

    }
    if(sonido19 == NULL) {

//     cerr << "No se puede cargar el sonido" << endl;
       exit(1);
```

```
    }
    if(sonido20 == NULL) {

//     cerr << "No se puede cargar el sonido" << endl;
       exit(1);

    }


    // Establecemos el volumen para el sonido

    int volumen15 = 100;
    int volumen16 = 100;
    int volumen17 = 100;
    int volumen18 = 100;
    int volumen19 = 100;
    int volumen20 = 100;

    Mix_VolumeChunk(sonido15, volumen15);
    Mix_VolumeChunk(sonido16, volumen16);
    Mix_VolumeChunk(sonido17, volumen17);
    Mix_VolumeChunk(sonido18, volumen18);
    Mix_VolumeChunk(sonido19, volumen19);
    Mix_VolumeChunk(sonido20, volumen20);



    // Creamos canales

    Mix_AllocateChannels(6);

    // Introducimos el sonido en uno de los canales
    // En el canal 1 con reproducciÃ³n infinita (-1)


    Mix_PlayChannel(-1, sonido15, 0);
    Mix_PlayChannel(-1, sonido16, 0);
    Mix_PlayChannel(-1, sonido17, 0);
    Mix_PlayChannel(-1, sonido18, 0);
    Mix_PlayChannel(-1, sonido19, 0);
    Mix_PlayChannel(-1, sonido20, 0);



    // Variables auxiliares

   SDL_Event evento;
    SDL_EnableKeyRepeat(SDL_DEFAULT_REPEAT_DELAY,\
                    SDL_DEFAULT_REPEAT_INTERVAL);


    // Bucle finito

int i;
i==0;

    for( i==0;i++;i<1) {
                if(evento.key.keysym.sym == SDLK_ESCAPE) {
                 return 0;
            }
    }
}
```

Con estas funciones ya teníamos introducido el sonido en nuestra aplicación. El problema se daba cuando poníamos una condición para que sonase el sonido

cuando apareciese una marca, y al sonar el sonido cumpliendo dicha condición, la captura a tiempo real se detenía hasta que terminase el sonido, debido a que eran dos procesos distintos. Para solucionar este problema utilizamos hilos o hebras.

5.3.3. Hilos y condiciones de funcionamiento

Para poder integrar los hilos en nuestro código tuvimos que desarrollar una clase en otro fichero .h aparte, y en este fichero incluir toda la implementación referida a la carga de sonidos y las condiciones que se tienen que dar para que suene cada uno. En primer lugar en el .h definimos todas las funciones que generar los sonidos (las vistas anteriormente), posteriormente creamos los punteros de las funciones que vamos a utilizar:

```
void *func1 (void *);
void *func2 (void *);
void *func3 (void *);
void *func4 (void *);
void *func5 (void *);
void *func6 (void *);
void *func7 (void *);
void *func8 (void *);
void *func9 (void *);
void *func10 (void *);
void *func11 (void *);
void *func12 (void *);
void *func13 (void *);
void *func14 (void *);
void *func15 (void *);
```

A continuación declaramos los hilos y sus atributos:

```
pthread_t thread1, thread2, thread3, thread4, thread5, thread6, thread7,
thread8, thread9, thread10, thread11, thread12, thread13, thread14,thread15,
thmain;
pthread_attr_t attr;

int pthread_create (pthread_t *tid, const pthread_attr_t
*attr, void *(*start_routine)(void*), void *arg);
```

Ahora creamos las funciones, ligándolas a los hilos y adjudicándole a cada una de ellas una de las funciones de reproducción de sonido:

```
void *func1 (void *arg)
{
pthread_t tid = pthread_self(); /* identificador de thread*/
Sonara();
pthread_exit(NULL); /* Provoca la terminaciÃ³n del thread*/
}

void *func2 (void *arg)
{
pthread_t tid = pthread_self(); /* identificador de thread*/
Sonarb();
pthread_exit(NULL); /* Provoca la terminaciÃ³n del thread*/
}
void *func3 (void *arg)
{
pthread_t tid = pthread_self(); /* identificador de thread*/
Sonarc();
pthread_exit(NULL); /* Provoca la terminaciÃ³n del thread*/
}
void *func4 (void *arg)
{
pthread_t tid = pthread_self(); /* identificador de thread*/
Sonard();
pthread_exit(NULL); /* Provoca la terminaciÃ³n del thread*/
}
…
```

Creadas ya las funciones de reproducción de sonido y las funciones con los hilos, creamos ahora la clase:

```
class MarkerProximityUpdateCallback : public osg::NodeCallback {

      private:
            osg::MatrixTransform* mtA;
            osg::MatrixTransform* mtB;
            osgART::Marker *a;
            osgART::Marker *b;
            osgART::Marker *c;
            osgART::Marker *d;
            osgART::Marker *e;
            osgART::Marker *f;
            osgART::Marker *g;
            osgART::Marker *h;
            osgART::Marker *i;
            osgART::Marker *j;
            osgART::Marker *k;
            osgART::Marker *l;
            osgART::Marker *m;
            osgART::Marker *n;
            osgART::Marker *p;
            osgART::Marker *o;

            bool veoa;
            bool veob;
            bool veoc;
            bool veod;
            bool veoe;
            bool veof;
            bool veog;
            bool veoh;
            bool veoi;
            bool veoj;
            bool veok;
            bool veol;
            bool veom;
            bool veon;

            float mThreshold;
```

Esta es una clase creada para calcular la proximidad de las marcas a la que nosotros añadimos nuestra información. Definimos las marcas y posteriormente creamos unas variables que luego nos servirán para decir cuando debe sonar el sonido y cuando no.

```
void te_veoa(){
        veoa=true;
        veob=true;
        veoc=true;
        veod=true;
        veoe=true;
        veof=true;
        veog=true;
        veoh=true;
        veoi=true;
        veoj=true;
        veok=true;
        veol=true;
        veom=true;
        veon=true;

       std::cout<<"sonido"<<std::endl;

}

void te_veob(){
        veoa=true;
        veob=true;
        veoc=true;
        veod=true;
        veoe=true;
        veof=true;
        veog=true;
        veoh=true;
        veoi=true;
        veoj=true;
        veok=true;
        veol=true;
        veom=true;
        veon=true;


       std::cout<<"sonido2"<<std::endl;

}
…
```

Estas funciones usan las variables definidas anteriormente, y lo que haremos será que cuando estas variables estén en *false*, el sonido pueda sonar, e inmediatamente después se ejecutan estas funciones para que las variables se pongan *true*, para que no se repita el sonido hasta que otro factor vuelva cambiar las variables a *false*.

Definimos ahora la parte pública de la clase:

```
public:


MarkerProximityUpdateCallback(
            osg::MatrixTransform* mA, osg::MatrixTransform* mB,
osgART::Marker* _a,
osgART::Marker*_b,osgART::Marker*_c,osgART::Marker*_d,osgART::Marker*_e,osgA
RT::Marker*_f,osgART::Marker*_g,osgART::Marker*_h,osgART::Marker*_i,

      osgART::Marker*_j,osgART::Marker*_k,osgART::Marker*_l,osgART::Marker*
_m,osgART::Marker*_n,osgART::Marker*_p,osgART::Marker*_o,
                  float threshold) :
osg::NodeCallback(),
                                    mtA(mA),
mtB(mB),a(_a),b(_b),c(_c),d(_d),e(_e),f(_f),g(_g),h(_h),i(_i),j(_j),k(_k),l(
_l),m(_m),n(_n),p(_p),o(_o),
                                    mThreshold(threshold) {

                  thmain = pthread_self();
                  pthread_attr_init (&attr);
            }

            virtual void operator()(osg::Node* node, osg::NodeVisitor* nv)
{

                  /** CALCULATE INTER-MARKER PROXIMITY:
                        Here we obtain the current position of each
marker, and the
                        distance between them by examining
                        the translation components of their parent
transformation
                        matrices **/
                  osg::Vec3 posA = mtA->getMatrix().getTrans();
                  osg::Vec3 posB = mtB->getMatrix().getTrans();
                  osg::Vec3 offset = posA - posB;
                  float distance = offset.length();
```

El código que está dentro del constructor hace que la propia función main sea un hilo y así poder combinar la captura de vídeo a tiempo real con la reproducción del sonido. A partir de aquí definimos las condiciones para la reproducción del sonido:

```
                  if (a->valid() && p->valid() && !veoa){
                      te_veoa();
                  pthread_create (&thread1, &attr, func1, NULL);

                  }
                  if (b->valid() && p->valid() && !veob){
                      te_veob();
                  pthread_create (&thread2, &attr, func2, NULL);

                  }
                  if (c->valid() && p->valid() && !veoc){
                      te_veoc();
                  pthread_create (&thread3, &attr, func3, NULL);

                  }

                  if (d->valid() && p->valid() && !veod){
                      te_veod();
                  pthread_create (&thread4, &attr, func4, NULL);
```

```
}

if (e->valid() && p->valid() && !veoe){
    te_veoe();
pthread_create (&thread5, &attr, func5, NULL);

}

if (f->valid() && p->valid() && !veof){
    te_veof();
pthread_create (&thread6, &attr, func6, NULL);

}

if (g->valid() && p->valid() && !veog){
    te_veog();
pthread_create (&thread7, &attr, func7, NULL);

}

if (h->valid() && p->valid() && !veoh){
    te_veoh();
pthread_create (&thread8, &attr, func8, NULL);

}

if (i->valid() && p->valid() && !veoi){
    te_veoi();
pthread_create (&thread9, &attr, func9, NULL);

}

if (j->valid() && p->valid() && !veoj){
    te_veoj();
pthread_create (&thread10, &attr, func10, NULL);

}

if (k->valid() && p->valid() && !veok){
    te_veok();
pthread_create (&thread11, &attr, func11, NULL);

}

if (l->valid() && p->valid() && !veol){
    te_veol();
pthread_create (&thread12, &attr, func12, NULL);

}

if (m->valid() && p->valid() && !veom){
    te_veom();
pthread_create (&thread13, &attr, func13, NULL);

}

if (n->valid() && p->valid() && !veon){
    te_veon();
pthread_create (&thread14, &attr, func14, NULL);

}

if (a->valid() && b->valid() && c->valid() && d->valid()
&& e->valid()  && g->valid() && !veon){
    te_veon();
pthread_create (&thread15, &attr, func15, NULL);

}
```

A cada marca le hemos asociado una letra en orden alfabético, y lo que hacemos con estos if´s es decirle al programa que si la marca es detectada, y a su vez se detecta la marca p (que corresponde al modelo 3D de un Play) y el parámetro "veox" asociado a esa marca esta en *false*, suene el sonido. Si nos fijamos el último de los if´s nos dice que ha menos que detecte 6 marcas, no sonará. Este último representa la función banda musical, de la que más adelante hablaremos.

Por último necesitamos algo que nos devuelva el valor *false* a las variables "veox" para poder escuchar el mismo sonido u otro distinto. Para ello:

```
if (o->valid()){
                      veoa=false;
                      veob=false;
                      veoc=false;
                      veod=false;
                      veoe=false;
                      veof=false;
                      veog=false;
                      veoh=false;
                      veoi=false;
                      veoj=false;
                      veok=false;
                      veol=false;
                      veom=false;
                      veon=false;
                     }
```

Con este if especificamos que si detectamos la marca "rewind" todas estas variables vuelven a estar en false y podemos empezar otra vez la actividad, con el mismo instrumento o sonido, o distinto.

Por último, en el fichero .cpp en la parte final escribimos este código:

```
root->setUpdateCallback(new
MarkerProximityUpdateCallback(arTransformA.get(), arTransformB.get(),
markerA, markerB,
       markerC, markerD, markerE, markerF, markerG, markerH, markerI,
markerJ, markerK, markerL, markerM, markerN, markerP, markerO, 200.0f));
       video->start();
```

Y también incluimos el fichero .h:

```
#include "m.h"
```

Capítulo 6

6.1. Resultados

Ya hemos explicado el código que hemos implementado para realizar la aplicación. Ahora nos queda explicar como es realmente nuestro programa y sus aplicaciones. En este proyecto educativo hemos intentado crear un único ejecutable que desarrolle cuatro actividades educativas distintas.
Explicamos cada una de ellas y mostramos los resultados:

6.1.1. Instrumentos

Esta primera actividad consiste en mostrar distintos instrumentos musicales, para que el alumno vea de forma virtual dichos instrumentos y después interactué con ellos mostrando una segunda marca con el modelo virtual de "Play". Cuando se muestre el marcador de "Play", se reproduce el sonido del instrumento objetivo, de forma que el alumno aprende a distinguir los distintos tipos de instrumentos y el sonido de cada uno de ellos.
La actividad consta de los siguientes instrumentos:

1. Instrumentos de cuerda:

   Se tratan de instrumentos que se basan en producir sonido mediante las vibraciones de una o más cuerdas, cuya vibración resuena en la caja de resonancia que tienen. Estas cuerdas están tensadas entre dos puntos del instrumento y se hacen sonar raspando, percutiendo o frotando la cuerda. Dentro de este grupo de instrumentos podemos distinguir los instrumentos de cuerda frotada, los de cuerda pulsada y los de cuerda percutida. En nuestra aplicación tenemos uno o más ejemplos de cada tipo de instrumento de cuerda, para hacer una clasificación completa de los mismos y así enseñar al alumno todo el abanico básico de tipos de instrumentos. De cuerda frotada tenemos el ejemplo del violín, instrumento básico en cualquier orquesta y muy común dentro de la música:

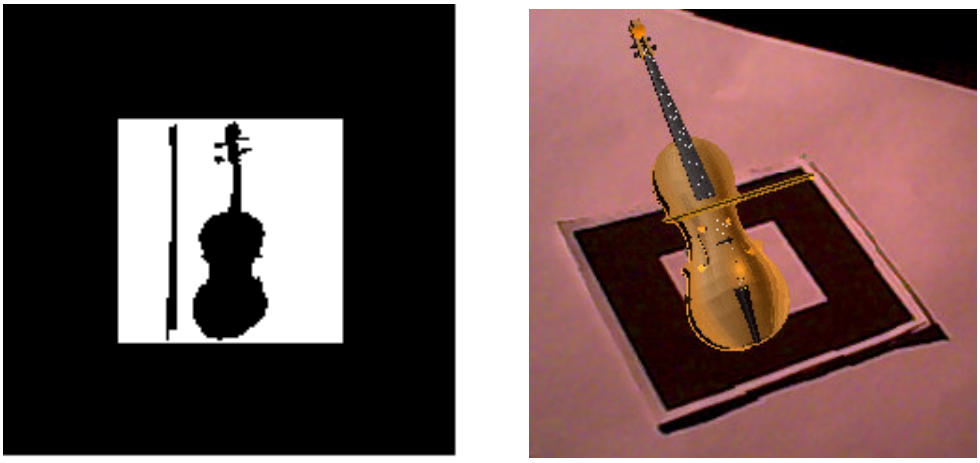


Figura 6.1: Marcador y modelo virtual del instrumento violín

De cuerda pulsada tenemos tres ejemplos básicos de la música de hoy y muy presentes en la música escuchada día a día como son la guitarra eléctrica, la guitarra acústica o española y el bajo:

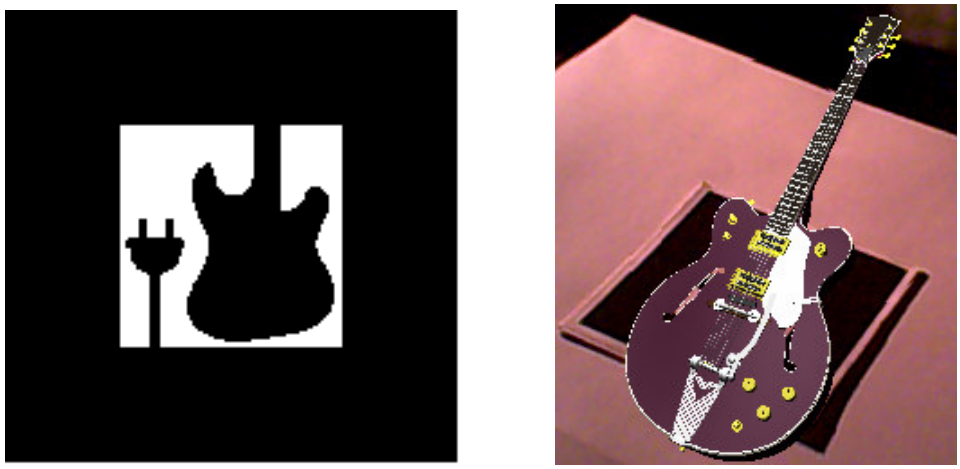


Figura 6.2: Marcador y modelo virtual del instrumento guitarra eléctrica

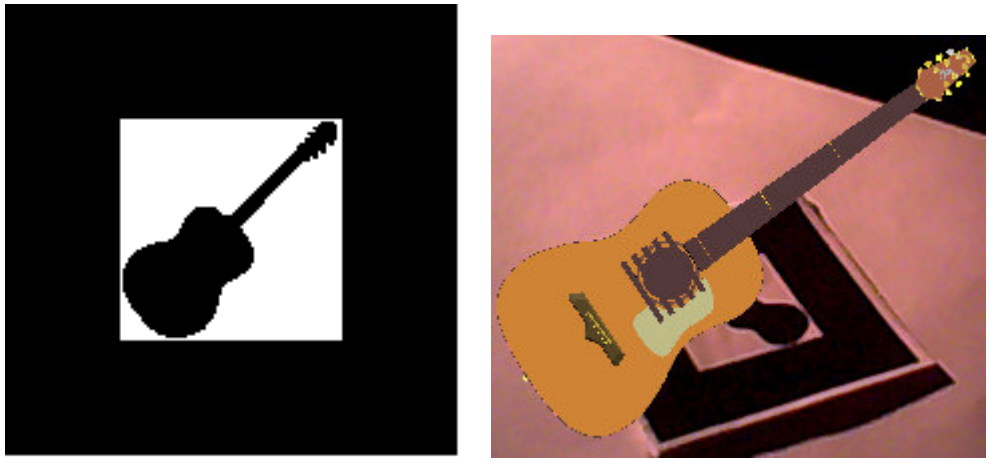


Figura 6.3: Marcador y modelo virtual del instrumento guitarra acústica

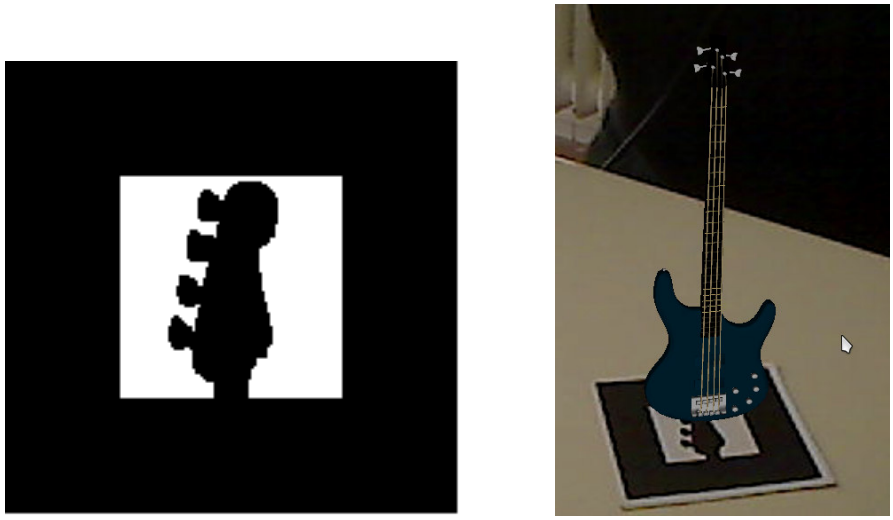


Figura 6.4: Marcador y modelo virtual del instrumento bajo

Por último como ejemplo de cuerda percutida hemos optado por unos de los instrumentos más conocidos de la historia de la música como es el piano:

Figura 6.5: Marcador y modelo virtual del instrumento piano

2. Instrumentos de viento:

   Estos instrumentos basan su funcionamiento en la vibración de una columna de aire que pasa por su interior para producir el sonido, sin necesidad de membranas vibratorias o que el instrumento vibre por si mismo. Para este tipo de instrumentos disponemos de dos ejemplos sencillos que nos permiten mostrar al alumno la forma y el sonido característico de los instrumentos de viento. Estos ejemplos son la flauta travesera y la trompeta. Ambos son instrumentos comunes en las bandas de desfiles y son de los más utilizados junto con el saxofón o el trombón.

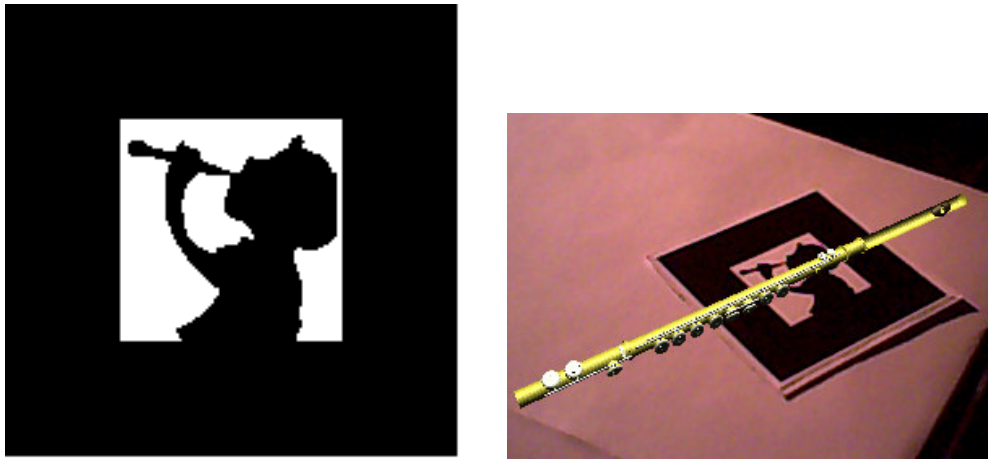


Figura 6.6: Marcador y modelo virtual del instrumento flauta

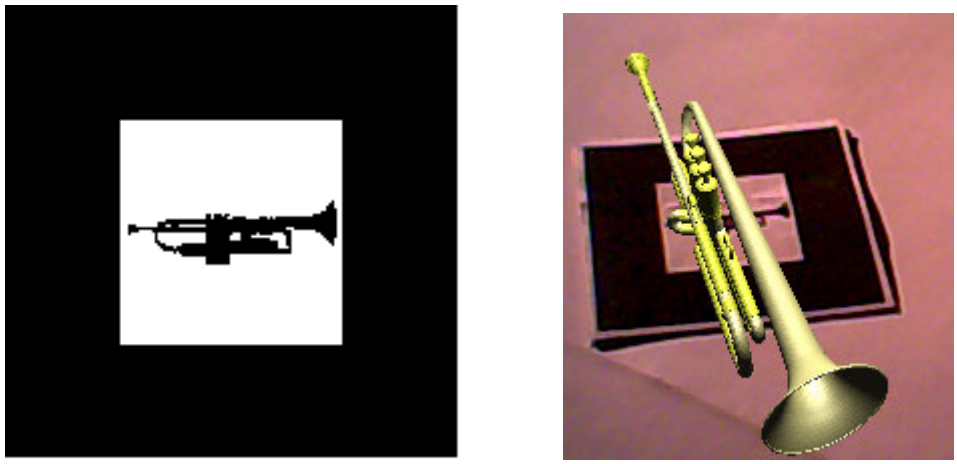


Figura 6.7: Marcador y modelo virtual del instrumento trompeta

3. Instrumentos de percusión:

   Los instrumentos de percusión son aquellos que generan el sonido al ser golpeados o agitados. Para los instrumentos de percusión solo escogimos un ejemplo debido a que estos instrumentos son más complicados de crear como modelo virtual, además de que aún habiendo una gran variedad de instrumentos de este tipo, son todos de funcionamiento y estética similar. Elegimos un tambor simple para la fácil comprensión del niño, aprovechando también la gran diferencia de sonido de estos instrumentos con los otros mencionados anteriormente:

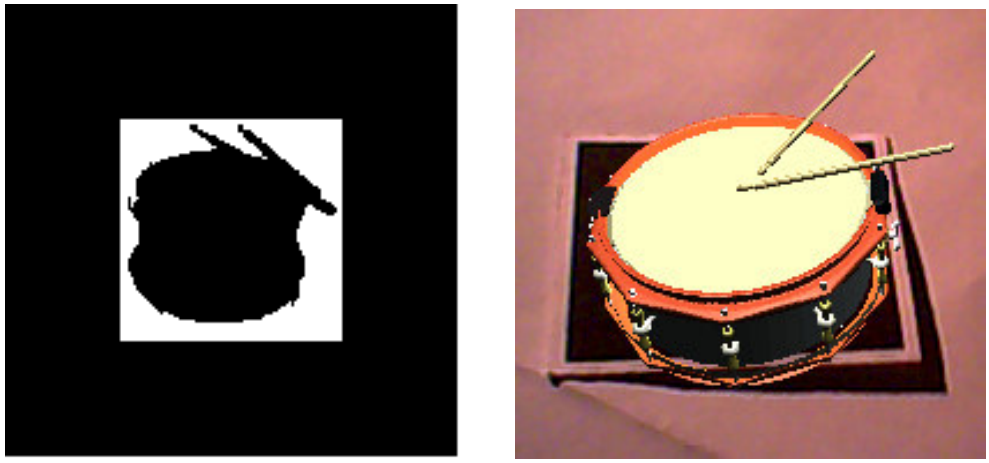


Figura 6.8: Marcador y modelo virtual del instrumento tambor

La marca que hace que cuando se detecte suene el instrumento que hayamos seleccionado es la marca “Play” como hemos mencionado anteriormente y se muestra en pantalla de esta forma:

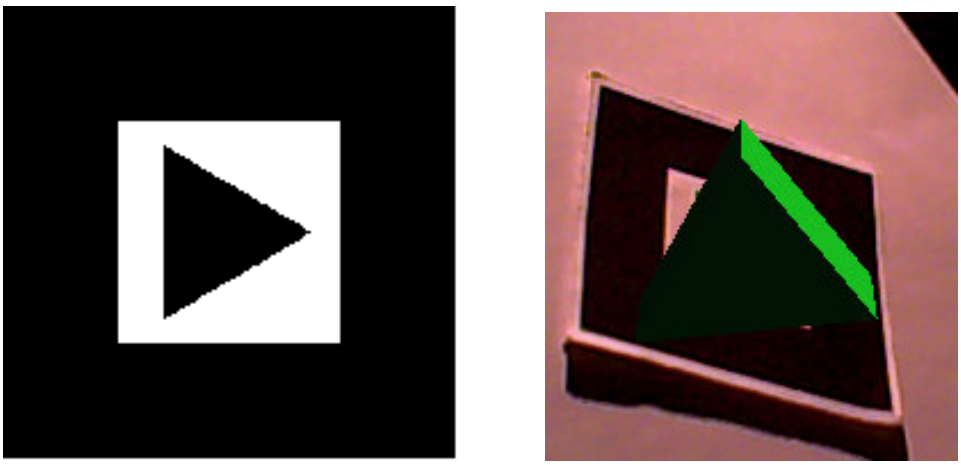


Figura 6.9: Marcador y modelo virtual de “Play”

Sin la aparición de esta plantilla no es posible la reproducción del sonido, así garantizamos la interacción con el usuario. La otra marca que debemos comentar es la marca rebobinar o “Rewind” que nos permite que el mismo instrumento suene una y otra vez siempre y cuando mostremos esta marca antes del instrumento. La marca tiene esta plantilla y se muestra de la siguiente forma:

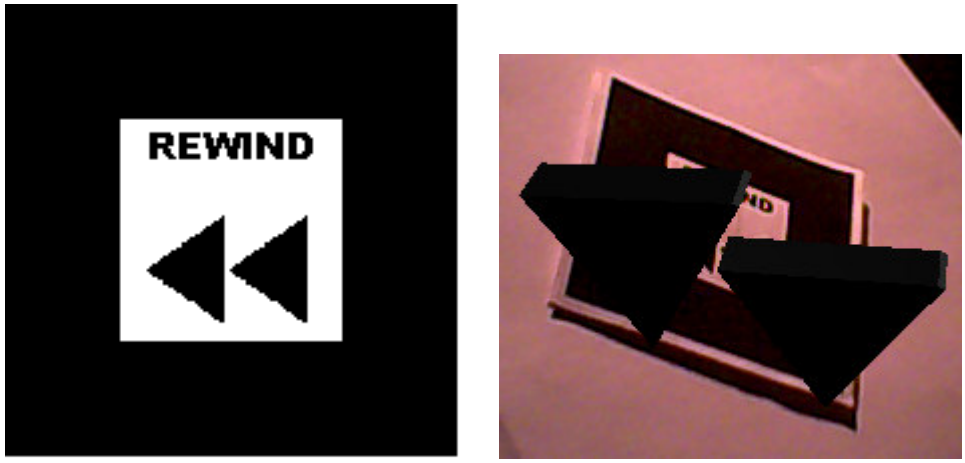


Figura 6.10: Marcador y modelo virtual de "Rewind"

Aquí acaba la primera actividad que hemos implementado con el objetivo de que el alumno aprenda a distinguir instrumentos básicos de la música, su tipo y su sonido, interactuando con ellos de una forma diferente y divertida.

6.1.2. Notas musicales

La segunda actividad propuesta en esta aplicación es algo parecida a la primera y se basa en el aprendizaje de las notas musicales. Ahora se trata de mostrar al alumno las distintas notas musicales que podemos poner en un pentagrama y su valor con respecto a su duración en un compás. Para ello hemos implementado las cuatro notas principales de un pentagrama (corchea, negra, blanca y redonda) y hemos puesto en su marcador su valor con respecto a un compás 4/4. Elegimos un compás 4/4 debido a que es el compás básico que se enseña cuando el alumno se inicia en el aprendizaje de la música. En este caso también esperamos a mostrar la marca "Play" para que se reproduzca el sonido y la marca "Rewind" para volver a escucharlo. Los sonidos serán reproducciones donde se escuchará un diapasón marcando el compás 4/4 y en cada caso un piano tocará notas a ritmo de corchea, negra, redonda o blanca.

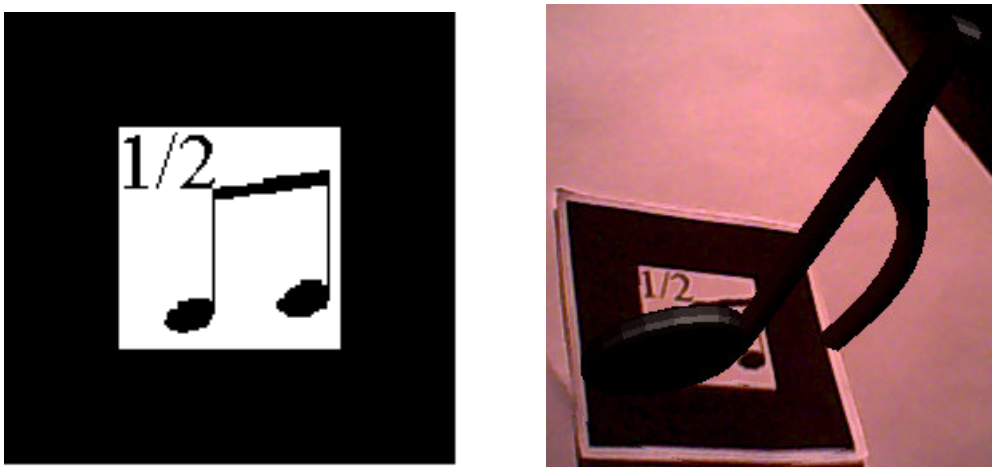


Figura 6.11: Marcador y modelo virtual de la nota corchea

La corchea dura medio tiempo en un compás 4/4.

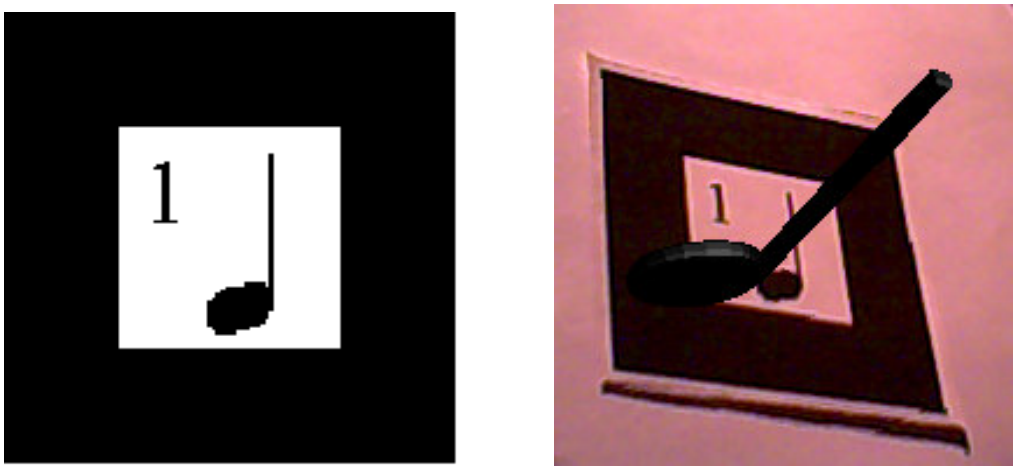


Figura 6.12: Marcador y modelo virtual de la nota negra

La negra dura un tiempo completo en un compás 4/4. Es la base del compás 4/4 debido a que ella es la unidad con la que contamos, es decir, en un compás 4/4 caben cuatro negras.

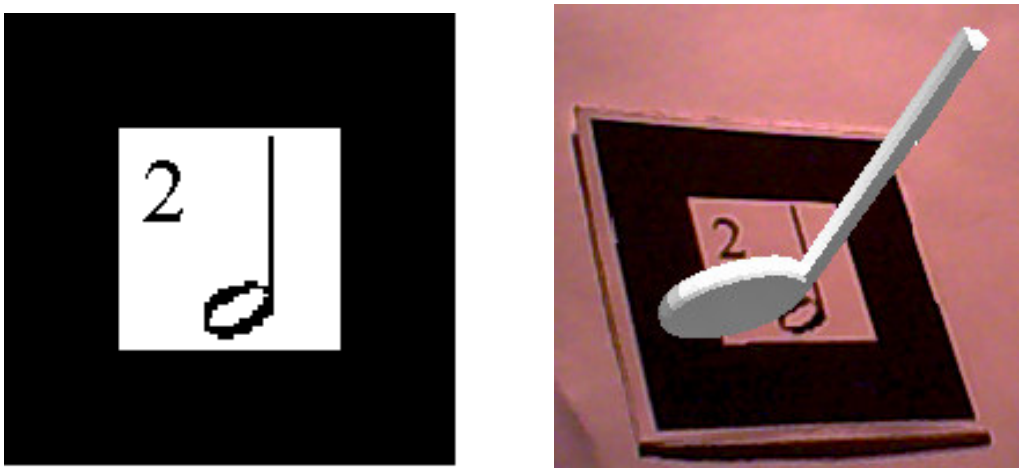


Figura 6.13: Marcador y modelo virtual de la nota blanca

La blanca dura dos tiempos en un compás 4/4.

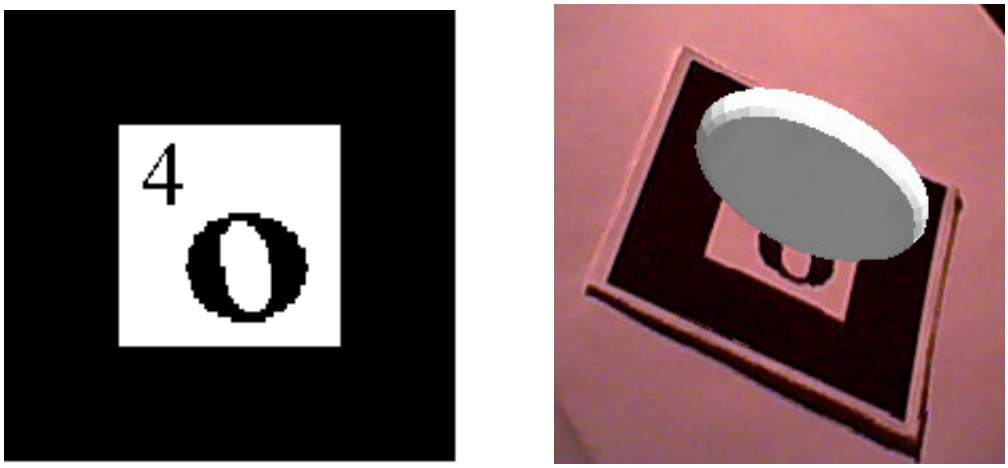


Figura 6.14: Marcador y modelo virtual de la nota redonda

La redonda dura los cuatro tiempos de los que se compone el compás 4/4.

De esta forma conseguimos que el alumno comprenda la duración de las distintas notas, y entienda su posición en un compás.

6.1.3. Altura y timbre

En esta actividad intentamos enseñar dos propiedades del sonido como son la altura y el timbre. La primera de ellas se trata del atributo del sonido que nos distingue un sonido agudo y otro grave, y nos dice que dos sonidos son distintos aún teniendo la misma intensidad y procediendo de la misma fuente, si su frecuencia es distinta. Nos da la definición de octava, refiriéndose a ella como a la distancia o región en frecuencia de dos sonidos con la misma intensidad y misma fuente, pero separados en frecuencia con una proporción de dos a uno (el doble), siendo la misma nota musical pero uno más agudo y otro más grave.

Para enseñar esta propiedad, volvemos a utilizar el modelo virtual del piano (debido a que el sonido que se escucha es reproducido por un piano) y lo que se escucha es la nota “do” tocada tres veces, cada vez más aguda, permitiendo a nuestros oídos percibir como es el mismo sonido, pero con distinta frecuencia (más agudo, como acabamos de puntualizar).

Figura 6.15: Marcador y modelo virtual de la cualidad "altura"

La segunda actividad nos permite enseñar la propiedad o cualidad del sonido llamada timbre. Esta propiedad nos permite distinguir dos sonidos de misma intensidad y frecuencia, pero de distintas fuentes. Se puede decir que el timbre de un sonido es su *matiz* o identidad según sea su fuente. Se trata de que el alumno pueda distinguir un "do" tocado por una trompeta de el mismo "do" tocado por un piano.

Con este objetivo en mente, creamos un sonido donde se reproducen unos segundos de canción, tocados primero por una guitarra acústica, luego por un piano y luego por un bajo. En los tres casos son exactamente las mismas notas, pero se escuchan distintas por que los instrumentos que las interpretan no son los mismos. Para esta actividad el modelo virtual creado se compone de los tres instrumentos mencionados, la guitarra acústica, el piano y el bajo:

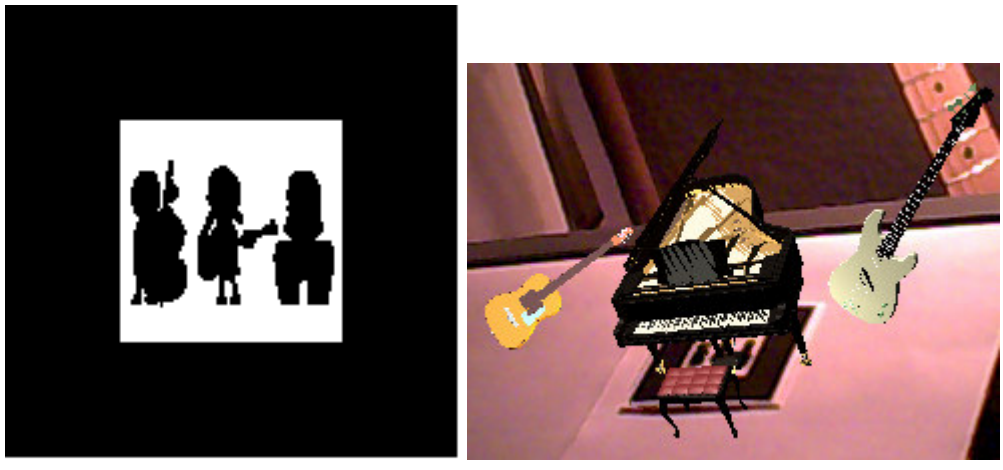
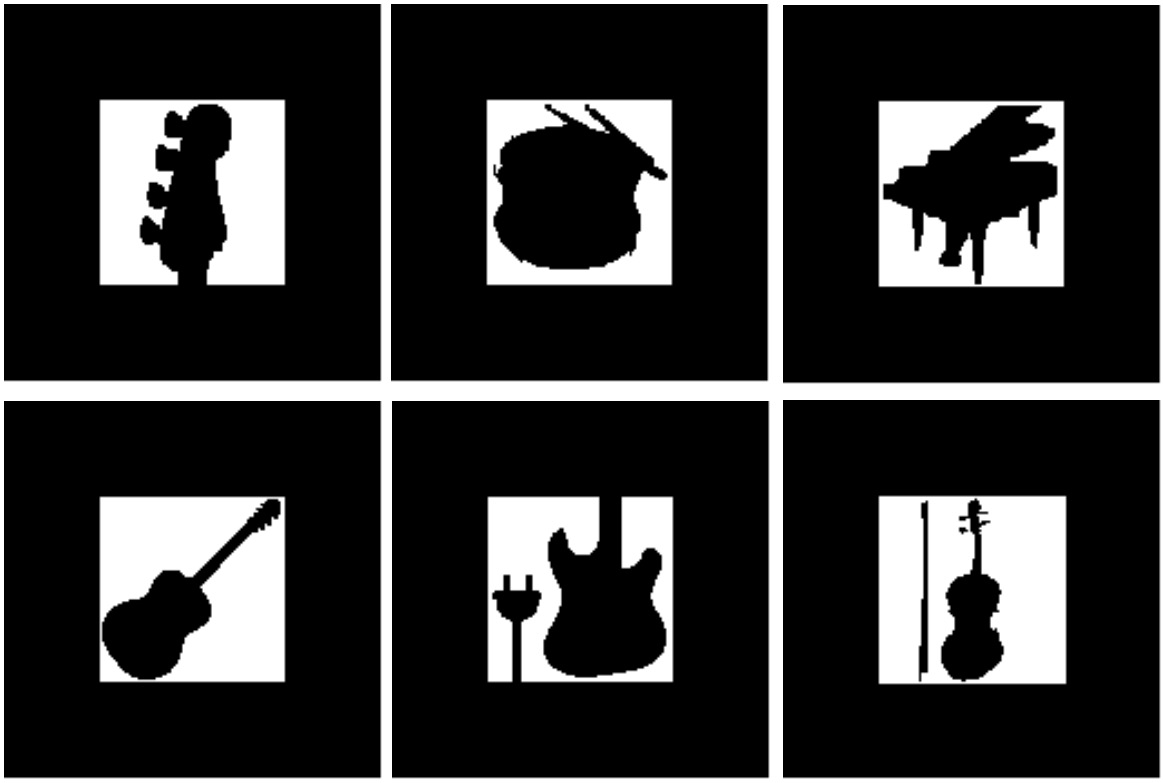


Figura 6.16: Marcador y modelo virtual de la cualidad "timbre"

6.1.4. Banda musical

Con esta última actividad lo que pretendemos mostrar es como los instrumentos que anteriormente hemos escuchado solos se complementan para formar una canción. Se mostrarán seis instrumentos y en el momento que aparezcan todos sonarán a la vez creando una melodía, y el alumno podrá escuchar y distinguir cada instrumento y comprender la función de cada uno dentro de una banda musical:



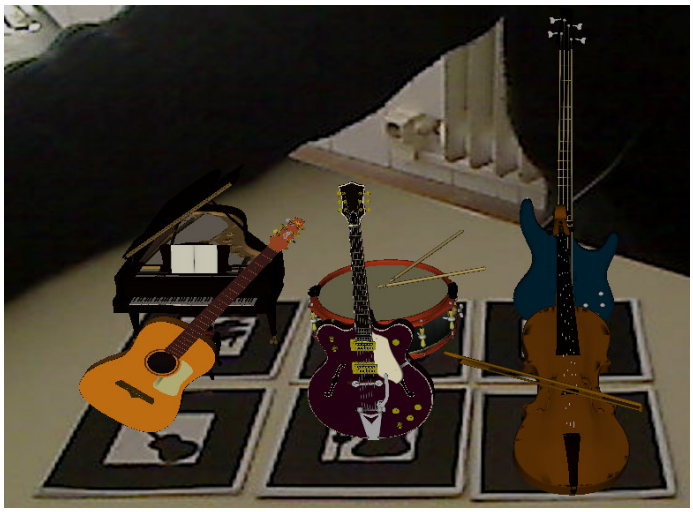


Figura 6.17: Marcador y modelo virtual de la banda musical

Con esta actividad concluimos nuestra aplicación, habiendo cumplido todos los objetivos propuestos al principio de de la misma, y con la satisfacción de ver como el trabajo realizado puede tener una aplicación real en la enseñanza de alumnado de poca edad.

Capítulo 7

7.1. Conclusiones

El objetivo principal de nuestro proyecto era crear una aplicación de realidad aumentada con fines educativos, que consiguiese enseñar de una forma más amena los contenidos al alumno. En general se ha cumplido este objetivo, creando una aplicación donde el alumno interactuar con los instrumentos y aprende a la vez los principios básicos de la música. Se ha conseguido una aplicación sencilla y fácil de utilizar y que supone el punto de partida para aplicaciones posteriores más complejas y completas.

Otro objetivo conseguido es la fácil manipulación de muestra aplicación para crear mas instrumentos, incluir mas sonidos, etc. Simplemente implementando mas veces lo descrito anteriormente podemos incluir mas sonidos y mas instrumentos, sin ninguna dificultad.

7.2. Trabajos futuros

Respecto a los trabajos futuros, se pueden crear varias aplicaciones a partir de la realizada en este proyecto. Un de ellas sería mejorar la actividad “banda musical”. La aplicación consistiría en que cuando mostremos la marca “Play” para cualquier instrumento empiece a sonar una canción, pero solo la parte de ese instrumento (por ejemplo, si mostramos la batería, solo sonaría la batería de la canción) para luego ir añadiendo instrumentos en la pantalla y entonces sonarían, pero justo por donde va la canción, es decir, sonarían todos en su compás correcto. La idea es que el alumno ponga y quite instrumentos dentro del transcurso de la canción para que viese más claramente cual es la función de cada uno dentro de la canción, que distinga cuales son los instrumentos que realizan la parte rítmica (guitarras acústicas, batería o bajo) y los instrumentos que realizan la parte “creativa” que pueden ser las guitarras

eléctricas, trompetas, etc. como se pretende ahora, pero de una manera más clara.

Otra de las ideas para seguir aplicando la música a la realidad aumentada es crear marcas de las distintas notas musicales (do, re, mi, fa, sol, la, si) y cada vez que este un instrumento mostrado en pantalla y mostremos una de las notas, que ese instrumento toque dicha nota. Esta aplicación requeriría un gran conjunto de sonidos (7 por cada instrumento) y nos serviría tanto para volver a recordar la cualidad de timbre (un sonido es distinto tocado por distintos instrumentos aún siendo la misma nota) como para conseguir adaptar y enseñar al oído al timbre de cada nota, para en un futuro distinguir sin problemas un "re" de un "do", un "sol" de un "si", etc.

7.3. Objetivos personales alcanzados

Este proyecto no solo ha supuesto para mí la creación de una aplicación de realidad aumentada en C++ y su posterior escritura, sino que ha supuesto el crear un proyecto yo solo desde su principio hasta su fin, realizando todas las tareas necesarias para el desarrollo del mismo y con mi única responsabilidad. Este proyecto no solo consta de lectura y modificación de un código, sino que abarca mucho más. Ha supuesto el aprender por mi mismo unas técnicas de programación para mí desconocidas hasta entonces, ha supuesto el diseño de más de 25 marcas para su prueba en dicho proyecto, he aprendido a manejar Blender para poder crear los modelos virtuales que ahora vemos en este proyecto. En definitiva, ha supuesto el ponerme en frente de un proyecto que va más allá de simplemente estudiar un código y modificarlo.

Los estudiantes no nos ponemos al frente de nada parecido en ninguna asignatura de la carrera, y sabemos que cuando entremos en el mundo laboral no solo tendremos que saber lo estudiado, sino que debemos saber desenvolvernos con cualquier cometido que nos asignen y tener una responsabilidad en la creación y ejecución de los proyectos futuros.

Bibliografía

[1] J.M.Peula, J.A.Zumaquero, C.Urdiales, A.M.Barbancho, F.Sandoval. *"Realidad Aumentada aplicada a herramientas didácticas musicales"*. Ed. Universidad de Málaga.

[2] Roberto Garrido (1) y Alex García Alonso (2). *"Técnicas de interacción para sistemas de realidad aumentada"*. (1) Unidad de Construcción y Desarrollo del Territorio, LABEIN – Tecnalia, Parque Tecnológico de Bizkaia. (2) UPV – EHU, Facultad de informática. Donosita – San Sebastián – Guipúzcoa.

[3] Tapia López, Joan. *"Juego de Realidad Aumentada de tanques"*. 2008

[4] Juan de Urraza. "*La Realidad Aumentada"*. Ed. Universidad católica "Nuestra Señora de la Asunción".

[5] Herrero Ibáñez, Manuel. *"Realidad Aumentada: ARToolKit para animación de personajes"*. Ed. Universidad Politécnica de Valencia.

[6] Negrete Montero, Cristóbal Roberto. *"Investigación Aplicada de las técnicas de Augmented Reality para la Presentación y Simulación en Tiempo Real de Proyectos de Diseño"*. Ed. Universidad Católica de Chile. 2006

[7] Duke Atkinson, Manny Collins, Keith Simmons, Casey Lee. *"Introduction to ARToolKit & Blender as seen at Makerfaire RI 2009"*.

[8] César Córcoles. *"Introducción al Blender"*.
http://mosaic.uoc.edu/2009/11/20/manual-de-introduccion-a-blender/

[9] http://www.openscenegraph.org/projects/osg

[10] http://www.osgart.org/wiki/Main_Page

[11] http://pfc-albertocorralesgarcia.blogspot.com/2008_02_01_archive.html

[12] Descarga de modelos virtuales: http://www.turbosquid.com

[13] http://es.wikipedia.org

[14] http://foro.aumentality.com/

[15] Descarga de modelos virtuales: http://www.bibliocad.com/

[16] http://www.artoolworks.com

[17] Juan Bautista Romero Carmona. *"Lenguajes educativos: Nuevas tecnologías y música"*. III Congreso Online. 2006.
http://www.cibersociedad.net/congres2006/gts/comunicacio.php?&id=413

[18] Betina Lippenholtz. *"La realidad aumentada. Educación e inmersión. Una buena dupla para reflexionar sobre las posibilidades de las nuevas tecnologías"*. http://portal.educ.ar/debates/educacionytic/inclusion-digital/la-realidad-aumentada-educacio.php. 2008

[19] Descargas de modelos virtuales: http://www-roc.inria.fr/gamma/download/download.php

[20] http://softwarelibre.uca.es/wikijuegos/Temario/SDL_mixer

[21] Página principal de ARtoolkit: http://www.hitl.washington.edu/artoolkit/

[22] http://www.artoolworks.com/products/stand-alone/osgart/

[23] http://www.realidadaumentada.info/

[24] http://www.guitar-pro.com

[25] Ed Kaiser, Alex Olwal, David McGee, Hrvoje Benko, Andrea Corradini, Xiaoguang Li, Phil Cohen, and Steven Feiner. Mutual disambiguation of 3d multimodal interaction in augmented and virtual reality. In *ICMI '03: Proceedings of the 5th international conference on Multimodal interfaces*, pages 12–19, New York, NY, USA, 2003. ACM.

[26] Hiroshi Ishii and Brygg Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems,* pages 234–241, New York, NY, USA, 1997. ACM.

[27] Mark Billinghurst, Hirkazu Kato, and Ivan Poupyrev. The magicbook: Moving seamlessly between reality and virtuality. *IEEE Comput. Graph. Appl.,* 21(3):6–8, 2001.

[28] Volkert Buchmann, Stephen Violich, Mark Billinghurst, and Andy Cockburn. Fingartips: gesture based direct manipulation in augmented reality. In *GRAPHITE '04: Proceedings of the 2nd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 212–221, New York, NY, USA, 2004. ACM.

[29] Julian Looser, Mark Billinghurst, Raphael Grasset, and Andy Cockburn. An evaluation of virtual lenses for object selection in augmented reality. In *GRAPHITE '07: Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, pages 203–210, New York, NY, USA, 2007. ACM.

[30] http://www.redessociales10.com/10-sorprendentes-aplicaciones-de-realidad-aumentada-para-iphone

[31] Sharon Oviatt. Ten myths of multimodal interaction. *Commun. ACM*, 42(11):74–81, 1999.

[32] I. Barakonyi, H. Prendinger, D. Schmalstieg, and M. Ishizuka. Cascading hand and eye movement for augmented reality videoconferencing. *3dui*, 0, 2007.

[33] Sergi Jordà. *"Realidad Aumentada: Taller de Sistemes Interactius 2003 – 2004"*. Ed. Universidat Pompeu Fabra. 2006

[34] http://www.learnar.org/

[35] http://www.invizimals.com/

[36] http://www.artoolworks.com/community/osgart/

Anexo I

Instalación de OSGART y SDL:

Para instalar OSGART como comentamos anteriormente, necesitamos tanto ARToolKit como OSG. Veamos como instalamos cada uno de ellos:

1º ARToolKit:

Para instalar ARToolKit tenemos que instalar previamente estas librerías:

freeglut3-dev
libgstreamer0.10-dev
libgstreamer-plugins-base0.10-dev (quizás este no es necesario)
libxi-dev
libxmu-headers
libxmu-dev
libjpeg62-dev
libglib2.0-dev
libgtk2.0-dev

Nos descargamos de la página official de ARToolKit la última versión del mismo y descomprimimos en el escritorio:

- *tar zxvf ARToolKit-2.72.1.tgz*

Seguidamente realizamos estas acciones:

En 32 bits: (no probado):

- *cd {ARToolKit}*

./Configure
Make

En 64 bits, tenemos que hacer make con la opcion -fPIC, eligiendo VideoGStreamer. Explicamos esto por pasos:

- cd {ARToolKit}
- nano ó gedit Configure
Buscamos la opción 5, driver VideoGStreamer
Buscar la variable CFLAG, linea 111, y añadimos -fPIC.
Esta es la línea original:
CFLAG="-O $GST_INCLUDE -I/usr/X11R6/include"
La sustituimos por esta:
CFLAG="-O $GST_INCLUDE -fPIC -I/usr/X11R6/include"
Guardamos los cambios.
- ./Configure
- make
Ahora vamos al directorio del bash (normalmente /home/usuario/) y lo abrimos con gedit o nano (>>>gedit .bashrc).y copiamos lo siguiente al final del mismo:

-export ARTOOLKIT_CONFIG="v4l2src device=/dev/video0 use-fixed-fps=false ! ffmpegcolorspace ! capsfilter caps=video/x-raw-rgb,bpp=24 ! identity name=artoolkit ! fakesink"

Es necesario para el correcto funcionamiento de ARToolKit. A continuación recargamos el bash:

>>>source .bashrc

siempre en la dirección donde está el bashrc (en este caso /home/usuario/).

Probamos ARToolKit, ejecutando en la carpeta cd {ARToolKit/bin}
./simpleTest

En este ejemplo que probamos debe capturar el vídeo de la cámara y si enseñamos el patrón “hiro.patt”, mostrará un cubo en 3D.

2º OSG:

Instalamos cmake si todavía no lo tenemos y copiamos el script osg2.8.3.py en /home/usuario/. Seguidamente en la consola tecleamos:

>>>python osg2.8.3.py

y se instalará. (necesitamos conexión a internet)

3º OSGART:

Descargamos OSGART de la web http://www.osgart.org.

- *http://osgart.org/files/osgART_2.0_RC3.zip*

Descomprimimos osgART_2.0_RC3.zip

- *cd osgART_2.0_RC3*
- *cd osgART_2.0_RC3* (Hay 2 carpetas seguidas)

En este directorio, vamos a realizar el cmake, pero antes copiamos en el bash lo siguiente:

- *export CFLAGS="$CFLAGS -fPIC -fpermissive"*
- *export CXXFLAGS="$CXXFLAGS -fPIC -fpermissive"*
- *export ARTOOLKIT_2_ROOT= “/home/..../ARToolKit”*

(En nuestro caso:*export ARTOOLKIT_2_ROOT="/home/roberto/Escritorio/ARToolKit")*

Justo debajo de lo copiado anteriormente.

Seguidamente tecleamos:

- *cmake . && make*

Y si todo va bien.

- *sudo make install*

Cuando ejecutemos el ejemplo osgartsimple en la carpeta bin, nos dirá que no encuentra la carpeta data para cargar los parámetros de la cámara. Tenemos que irnos a la carpeta etc/data. Ahí encontramos una carpeta que pone artoolkit2.copiamos todo lo que hay en su interior y nos vamos a la carpeta bin. Allí creamos una nueva carpeta llamada data y copiamos en su interior el contenido de artoolki2. Deberíamos

tener camera_par.dat y 4 ficheros patt. Si es así ahora si nos pillará los parámetros de la cámara y los patrones.

Aún así nos dará un error y saldrá la pantalla en negro. Para solucionarlo buscamos el archivo *ARToolKitVideo.cpp(osgART_2.0_RC3/osgART_2.0_RC3/src/osgART/Video/ARToolKit)* y cambiamos la condición del if. (simplemente modificar el if quitandole el 0==).
volvemos a compilar e instalar las librerías.

```
--- ARToolKitVideo.cpp.orig   2010-02-09 13:15:27.000000000 +0100
+++ ARToolKitVideo.cpp   2010-02-09 15:32:48.000000000 +0100
@@ -307,7 +307,7 @@
    osg::Timer t;
  if (0 == ar2VideoCapNext(video)) //sustituimos esto
  if (ar2VideoCapNext(video))//por esto
    {
      if (newImage = (unsigned char*)ar2VideoGetImage(video))
      {
```

Volvemos a hacer *cmake . && make* y *sudo make install* y ya debe funcionar. Ahora mismo esta completamente instalado OSGART y podemos trabajar con él y realizar nuestras propias aplicaciones.

Para conseguir un tracker fino, que no tiemble tanto, recurrimos a un filtro temporal:
Nos vamos a este fichero:
- /home/... RutaCarpetaOsgART .. /src/osgART/Tracker/ARToolKit/SingleMarker.cpp
Y editamos el fichero cambiando esta línea:
76 arGetTransMat(markerInfo, patt_center, patt_width, patt_trans);
por esta otra:
76 arGetTransMatCont(markerInfo, patt_trans, patt_center, patt_width, patt_trans);
recompilamos y volvemos a instalar OSGART..

Vamos a instalar ahora la librería SDL:

4º SDL:

Descargamos la versión SDL-1.2.14
Copiamos la carpeta descomprimida de SDL-1.2.14 en el directorio deseado (en nuestro caso el escritorio)
Nos vamos dentro de la carpeta SDL-1.2.14
- *cd SDL-1.2.14*
- *make*
- *sudo make install*

Y las librerias adicionales las instalamos directamente desde la consola, tanto las lib... como las lib...-dev:

libsdl-image1.2
libsdl-image1.2-dev
libsdl-mixer1.2
libsdl-mixer1.2-dev
libsdl-net1.2
libsdl-net1.2-dev
libsdl-ttf2.0-0
libsdl-ttf2.0-dev

Instalados todos los componentes y librerías, ahora solo nos queda sustituir ciertas carpetas para tener nuestra aplicación instalada en nuestro PC.

En osgART_2.0_RC3/osgART_2.0_RC3/bin, metemos las carpetas data (lleva los archivos patt y el *camera_par.pat*), media (lleva los modelos 3D) sonidos. En la carpeta osgART_2.0_RC3/osgART_2.0_RC3/examples/osgartsimple, sustituimos la carpeta osgartsimple por la que proporcionamos.
En usr/local/lib compiamos los archivos libosg.so, libosgDB.so, libosgGA.so y libosgViewer.so y creamos una carpeta en usr/local/ llamada lib64 y copiamos estos archivos dentro de esta carpeta. Si no ha habido ningún problema, nos vamos al directorio osgART_2.0_RC3/osgART_2.0_RC3/examples/osgartsimple/ y tecleamos:
-./osgartsimple
Y se ejecuta nuestra aplicación.

Anexo II

Calibración de la cámara

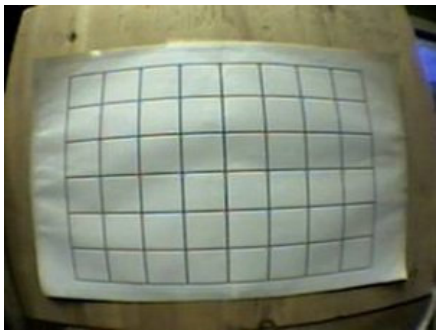
En ARtoolkit, el fichero *camera_para.dat* contiene las propiedades de la cámara. Estos parámetros son suficientes para una amplia gama de cámaras, pero con una sencilla técnica de calibración podemos generar un fichero con los parámetros específicos de configuración de la cámara que utilicemos. Mientras mejor sea la calibración de la cámara menor distorsión tendrá la imagen.
Vamos a explicar en este anexo como calibrar la cámara de video:

Podemos realizar la calibración de dos formas:

- Calibración en dos pasos: Es un método complicado de utilizar pero ofrece muy buena precisión.
- Calibración en un solo paso: Se trata de un método más fácil de usar y que nos da una precisión bastante buena pero no tanta como el método anterior.

Calibración de dos pasos

Para utilizar este método imprimimos los archivos *calib_cpara.pdf* y *calib_dist.pdf*, situados en el directorio ARtoolkit\patterns. El primero se trata de una cuadrícula de líneas y debe escalarse hasta conseguir una separación entre líneas de 40 milímetros exactamente. El segundo fichero contiene una matriz de 6 x 4 puntos y también debe escalarse hasta conseguir una separación entre líneas de 40 milímetros exactamente.



Figuras A y B: Patrones calib_dist y calib_cparam

Las propiedades de la cámara que se van a medir son: el punto central de la cámara, la distorsión de la lente y la longitud focal de la cámara. El programa *calib_dist* se utiliza para medir el punto central de la imagen y la distorsión de la lente, mientras que el programa *calib_cparam* mide las demás propiedades. Los programa se encuentran en ARtoolkit\bin.

Ejecutamos primero el programa *calib_dist* para medir el espacio entre los puntos y utiliza esa información para calcular la distorsión de la lente. Colocamos la cámara de forma que veamos todo el patrón. Ahora hacemos clic con el botón izquierdo del ratón para congelar la imagen, y posteriormente clic en el botón izquierdo en cada punto de patrón, arrastrando sin soltar hasta que aparezca un rectángulo negro encima del punto. Hacemos esto con todos los puntos, de izquierda a derecha y de arriba abajo, es decir, empezamos por la esquina superior izquierda y vamos a la izquierda, y debemos terminar en la esquina inferior derecha.

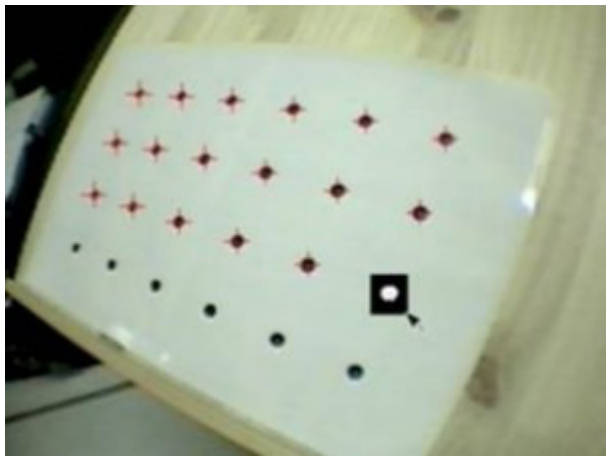


Figura C: Patrón calib_dist mostrando uno de los rectángulos que debemos realizar

Cuando el software del programa detecta el rectángulo, dibuja una cruz en cada punto. Debe de hacerse el rectángulo hasta que el programa lo reconozca y pinte la cruz. Cuando hemos terminado con los 24 puntos damos clic en el botón izquierdo

del ratón, se descongela la imagen y el programa guarda la posición de los puntos.

Ahora debemos repetir este proceso de 5 a 10 veces, con el patrón enfocado desde distintas posiciones. Mientras más veces lo hagamos, más precisa será la calibración.

Una vez terminado este proceso, damos clic al botón derecho del ratón y el programa comenzará a realizar los cálculos. Tras unos segundos el programa nos da los valores de centro X Y y el factor de distorsión de a cámara. Estos datos son utilizados posteriormente en *calib_cparam*. Para comprobar si estos parámetros son correctos volvemos a hacer clic en el botón izquierdo del ratón, y el programa nos muestra la primera imagen tomada con líneas cruzadas entre todos los puntos. Si estas líneas pasan por el centro de los puntos, la calibración es correcta.

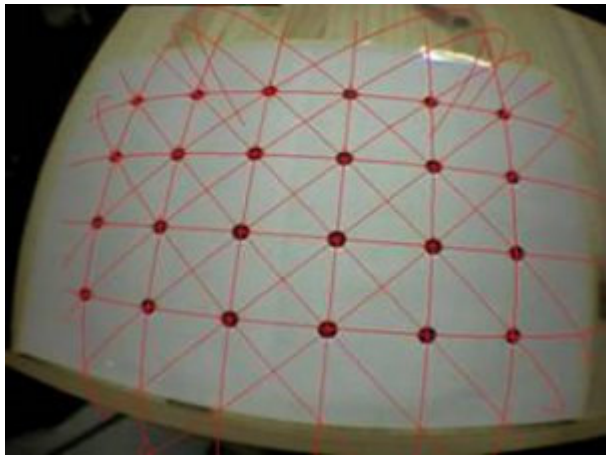


Figura D: Patrón calib_dist con las líneas rojas cruzándose correctamente

Haciendo clic en el botón izquierdo visualizamos la siguiente imagen tomada y haciendo clic en el botón derecho salimos de la aplicación.

Pasamos ahora a ejecutar el archivo *calib_cparam*, utilizado para encontrar la distancia focal de la cámara y otros parámetros. Cuando inicializamos esta aplicación debemos introducir los parámetros obtenidos en *calib_dist*.

Colocamos el patrón justo enfrente de la cámara perpendicular a ella y hacemos clic en el botón izquierdo del ratón para grabar un fotograma. Una línea horizontal blanca aparece superpuesta en la imagen. Movemos esta línea hasta situarla encima de la primera línea horizontal superior del patrón. La línea se mueve con las teclas arriba y abajo. Cuando la línea este situada presionamos “Enter” y se convertirá en azul, y volverá a salir otra línea blanca, que situamos en la siguiente línea horizontal del patrón y así sucesivamente hasta completar todas las líneas horizontales.

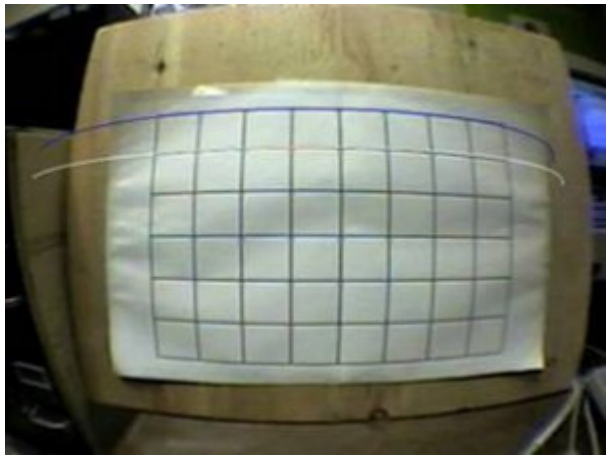


Figura E: Colocación de líneas horizontales

Cuando acabemos con esto, la línea nos aparecerá en vertical. Hacemos el mismo proceso pero esta vez de izquierda a derecha.

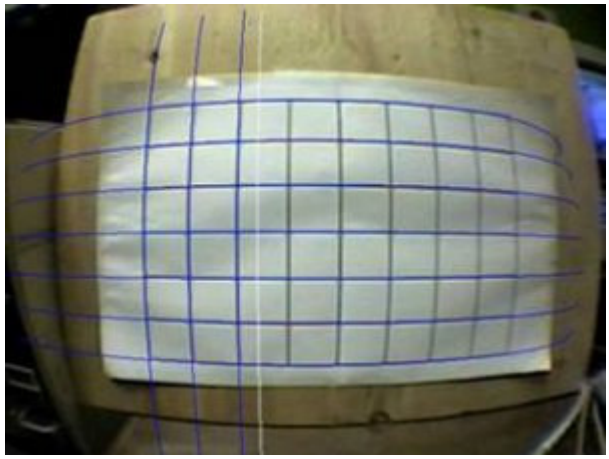


Figura F: Colocación de líneas verticales

Terminado esto, alejamos el patrón de la cámara unos 10 cm. y repetimos el mismo proceso. Repetimos el proceso un total de 5 veces, alejándolo cada vez 10 cm., y después de la quinta vez el programa automáticamente calculará los parámetros de la cámara. El programa pedirá un nombre para el fichero. Escribimos el nombre y hacemos clic en el botón derecho del ratón para salir de la aplicación.

Calibración en un paso

En este caso se realizar prácticamente los mismos pasos que en el caso anterior. Imprimimos el archivo *calib_dist.pdf* y ejecutamos el programa *calib_camera2*. Los pasos a seguir son los mismos que con el programa *calib_dist*.

Anexo III

Creación de nuevos marcadores

ARtoolkit nos proporciona varios patrones o marcadores ya incluidos dentro de ARtoolkit, pero puede ocurrir como es en nuestro caso que no sean suficientes para los objetivos que nosotros tenemos. Si queremos crear nuestros propios marcadores, lo primero que tenemos que hacer es imprimir el archivo "blankPatt" contenido en la carpeta ARToolKit\patterns, y pintar en su interior la figura que queremos para nuestro marcador. También podemos hacer esto desde el propio ordenador con el programa paint o con el programa Adobe Photoshop como en nuestro caso.

Una vez creado nuestro propio marcador, nos dirigimos al directorio ARToolKit\bin y ejecutamos "mk_patt". El programa se abrirá y nos preguntará por los parámetros de la cámara.

Presionando la tecla "Enter" nos saldrá los parámetros de nuestra cámara. Damos a "Aceptar" y se nos abre la ventana con la imagen capturada por la cámara a tiempo real. Ahora debemos poner nuestro marcador delante de la cámara, y cuando ésta lo detecte dibujará líneas rojas en la esquina superior izquierda del marcador y líneas verdes en la esquina inferior derecha. Así nos indica que está detectando correctamente el marcador. Si los colores se invierten está mal detectado, y hay que tener mucho cuidado con esto.

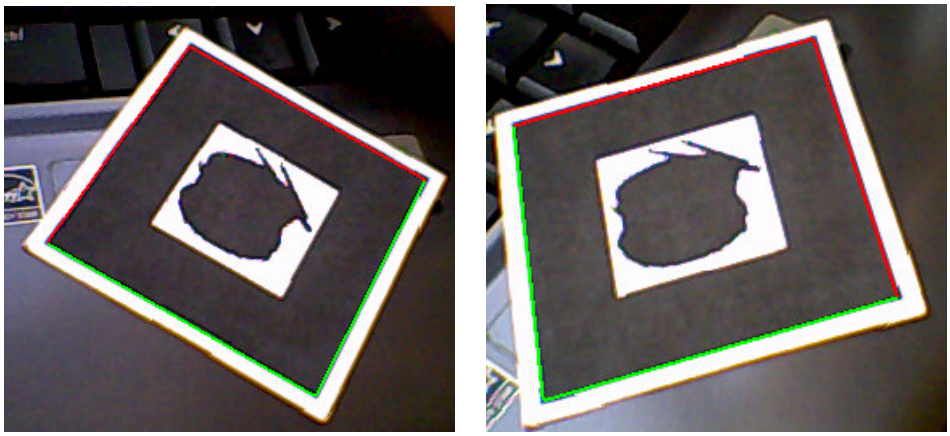


Figura G: Marcador detectado correctamente a la izquierda y marcador mal detectado a la derecha

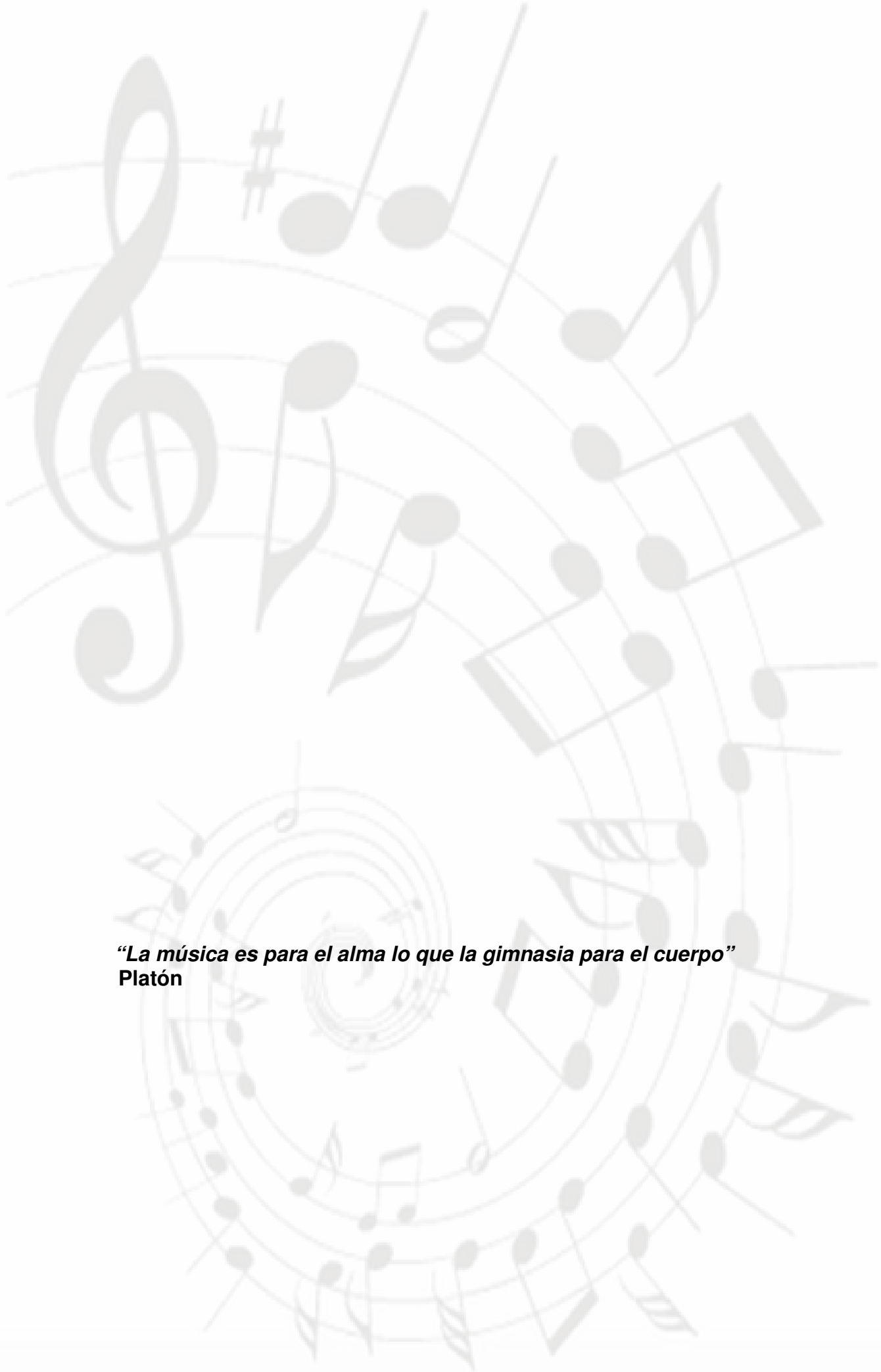
Una vez detectado el marcador, pulsamos con el botón izquierdo del ratón la pantalla de la imagen de la cámara y en la consola de comandos nos preguntará el nombre del marcador. Escribimos el nombre que queramos siempre y cuando antes pongamos patt., debido a que si no es así no se guardará como un archivo de marcador. Pulsamos “Enter” y el marcador se nos guardará en ARToolKit\bin. Únicamente nos queda copiarlo y pegarlo en la carpeta Data de nuestra aplicación.

UNIVERSIDAD DE EXTREMADURA

“AR-Learning: Interactive book based on augmented reality for educational intentions”

Roberto Gallego Delgado

ÍNDICE



"La música es para el alma lo que la gimnasia para el cuerpo"
Platón

INTRODUCCIÓN

¡¡Bienvenido a ARLearning!! ¡A partir de ahora te sumergirás en el fabuloso mundo de la Realidad Aumentada y podrás aprender de una forma mucho más divertida y amena todo lo relacionado con el maravilloso mundo de la música!

En primer lugar conoceremos los principales tipos de instrumentos musicales y su sonido. Seguidamente aprenderemos la escala musical, y la duración de las notas que ponemos en ella, y por último, aprenderemos algunas de las divertidas cualidades del sonido y cómo es un grupo musical, mediante actividades que te transportarán dentro del mundo virtual. ¡Será fantástico!

INSTRUMENTOS MUSICALES

Los instrumentos musicales son objetos creados por el hombre para crear sonidos y música. Hay muchos tipos diferentes de instrumentos y en cada tipo hay muchísimos instrumentos distintos, pero vamos a intentar enseñar los principales.

Hay tres grupos principales de instrumentos:

- Cuerda
- Viento
- Percusión

¿Sabrías decir cuantos instrumentos musicales conoces y de que tipo son?

INSTRUMENTOS DE CUERDA

Este primer tipo de instrumentos se caracterizan en que la forma en la que consiguen crear un sonido es mediante la vibración de cuerdas. Esto es muy sencillo:
La cuerda está sujeta en dos puntos y cuando nosotros la pulsamos o frotamos, la cuerda “vibra” y esa vibración genera un sonido.

Dentro de estos instrumentos vamos a estudiar:

-Guitarra española
-Guitarra eléctrica
-Bajo eléctrico
-Violín
-Piano

¿Reconoces alguno de ellos?

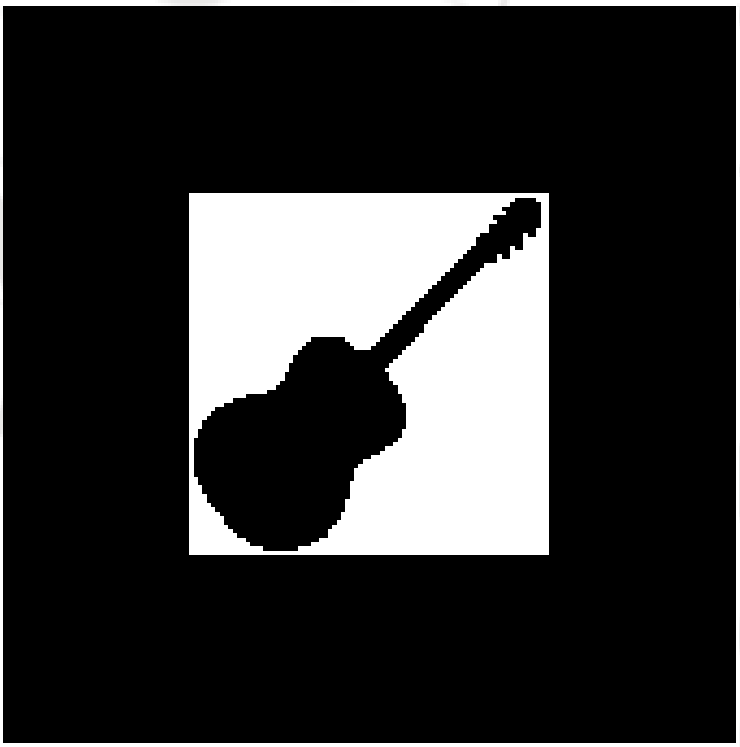
GUITARRA ESPAÑOLA

La guitarra española es un instrumento de cuerda formado por una caja de madera, un mástil y seis cuerdas.

En la caja de madera tiene un agujero (caja de resonancia) donde la vibración de la cuerda entra y produce el sonido. Según donde pongamos nuestro dedo a lo largo del mástil el sonido será diferente.

Se utiliza mucho para tocar flamenco, sevillanas, folclore... y también es muy utilizada por los cantautores.

En la siguiente página tenemos una marca con un dibujo de guitarra en el centro. Si lo enfocas con la cámara del ordenador ¿Qué sucede?



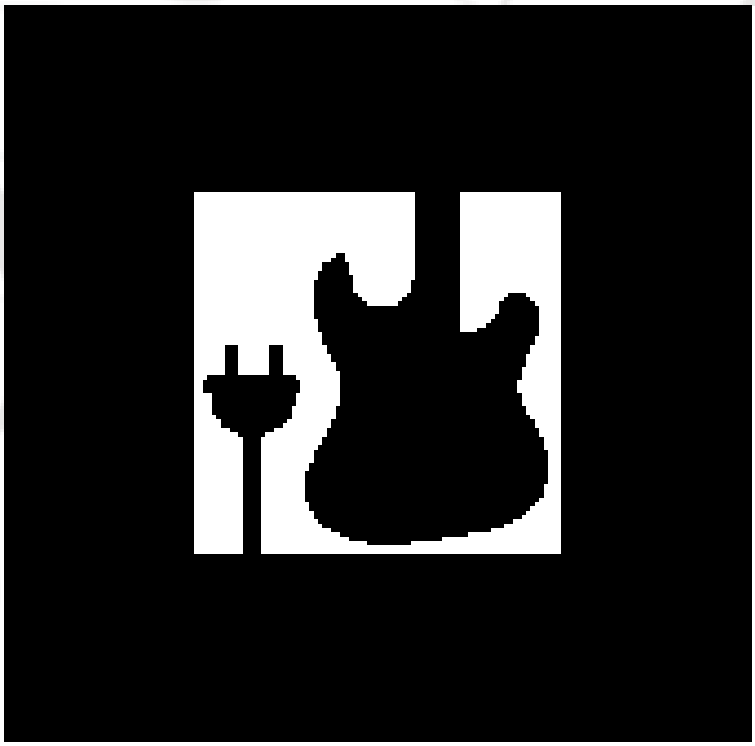
Ahora pon la marca de PLAY donde te indicamos abajo. ¿Qué ha sucedido ahora?

GUITARRA ELÉCTRICA

La guitarra eléctrica es un instrumento de cuerdas de metal formado también por un cuerpo, un mástil y 6 cuerdas (normalmente). Tiene unas pastillas que convierten las vibraciones de las cuerdas en señales eléctricas.

Normalmente estas guitarras no tienen caja de resonancia y necesitan de una especie de altavoz llamado amplificador para conseguir crear algún sonido, puesto que no tiene el “agujero” donde el sonido se crea en la guitarra española. Es muy utilizada en todos los estilos de música.

¿Reconoces las diferencias?



Ahora vuelve a poner la marca de PLAY donde te indicamos abajo.

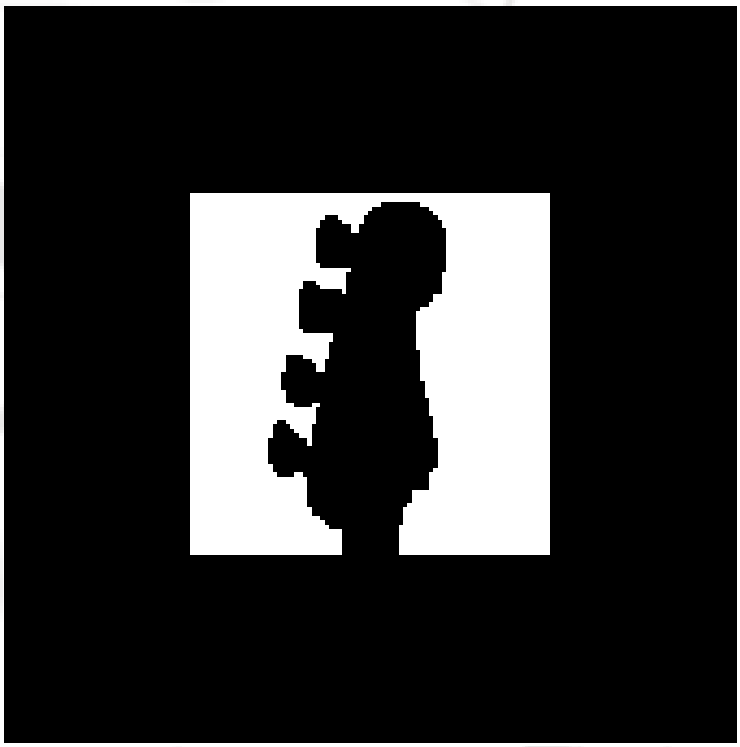
¿Notas la diferencia en el sonido?

BAJO ELÉCTRICO

El bajo eléctrico es un instrumento similar a la guitarra eléctrica, de cuerdas de metal formado por un cuerpo, un mástil y en este caso 4 cuerdas (normalmente). También tiene unas pastillas que convierten las vibraciones de las cuerdas en señales eléctricas, y por ello también necesita un amplificador para conseguir crear los sonidos.

El bajo se distingue de la guitarra principalmente porque sus cuerdas son mucho más gruesas, por lo que creamos sonidos mucho más graves. El bajo eléctrico se utiliza prácticamente en casi todos los estilos de música, puesto que lleva el "ritmo" de la canción.

¿Conocías este instrumento?



Escucha el sonido ahora. ¿Escuchas como el sonido es mucho más grave que en las guitarras?

VIOLÍN

El violín es un instrumento de cuerda, pero en este caso, la cuerda no se “toca” con los dedos o con una púa sino se frota con un arco. Es más pequeño que las guitarras aunque tiene una forma parecida, con un cuerpo y un mástil, y 4 cuerdas.

El arco con el que se toca este instrumento es una vara de madera a la que sujetamos en los extremos una tira de crines de caballo, para conseguir crear un sonido suave cuando frotamos la cuerda.

¿Habías visto este instrumento alguna vez?

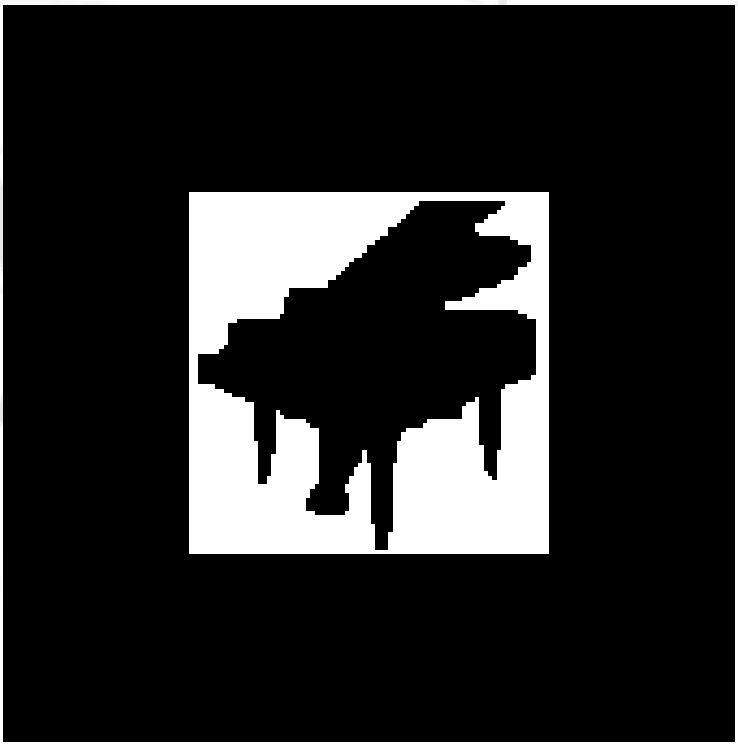
Escuchando el sonido ¿Recuerdas haber escuchado este instrumento antes?

PIANO

El piano es un instrumento de cuerda, pero es un instrumento “extraño”. A la vista de todos, el piano tiene un teclado, que es donde nosotros pulsamos para producir sonido, pero en realidad al pulsar la tecla del piano lo que hacemos es golpear a una cuerda que esta en su interior, creando así el sonido.

El piano suele utilizarse para la música clásica, o estilos como jazz o blues, y es considerado como uno de los instrumentos más elegantes y que produce uno de los sonidos más bellos.

¿Recuerdas este instrumento en alguna película o serie de TV?



¿Qué sonido de los instrumentos de cuerda que hemos visto te gusta más?

INSTRUMENTOS DE VIENTO

Los instrumentos de viento se caracterizan porque la forma en la que producen sonido es mediante la “vibración” del aire dentro del instrumento, sin cuerdas ni otros elementos.

Los instrumentos de viento se forman mediante uno o varios tubos y el músico sopla por uno de los extremos del instrumento y ese aire vibra dentro del mismo creando el sonido.

En este apartado estudiaremos:

-Flauta travesera
-Trompeta

¿Conocías este tipo de instrumentos?

FLAUTA TRAVESERA

La flauta travesera es un instrumento de viento, y se compone de un tubo con un número determinado de agujeros en un lateral, y una boquilla para soplar. Según qué agujeros tapemos, el sonido creado será uno u otro.

La flauta travesera se utiliza sobre todo en música clásica o de cámara, pero en los últimos tiempos se ha empezado a utilizar en distintos tipos de música, como en la música folk, el jazz o el rock.

¿Te recuerda a otro instrumento parecido esta flauta?

¿Notas la diferencia entre este sonido y los vistos anteriormente?

TROMPETA

La trompeta es un instrumento de viento incluido en los instrumentos de viento metal, debido a que esta hecha de metal. Esta fabricada de un tubo de casi 2 metros de largo doblado sobre si mismo formando diferentes “huecos” donde el aire vibra para así crear el sonido. Este tubo acaba en una campana por donde sale el sonido.

La trompeta, como la flauta travesera, se suele utilizar en la música clásica o en bandas municipales, pero algunos estilos de música latina o el jazz también la han incluido en sus composiciones.

¿Has visto este instrumento en la banda de tu ciudad?

¿Te resulta diferente este sonido al de la flauta travesera?

INSTRUMENTOS DE PERCUSIÓN

Los instrumentos de percusión son un tipo de instrumento musical que crea sonido mediante el golpeo del mismo, a través de una membrana que vibra cuando es golpeada. Es la forma de crear sonidos más antigua.

Estos instrumentos tienen la facilidad de adaptarse perfectamente a los demás instrumentos musicales, y forman la parte rítmica, siendo una parte fundamental en la creación de la canción.

Este tipo de instrumentos pueden producir tanto rítmos básicos (tambor, batería, tam-tam...) como notas musicales (xilófono). En esta unidad veremos uno de los instrumentos básicos de percusión, como es el tambor.

¿Has llegado a tocar alguna vez un instrumento de percusión?

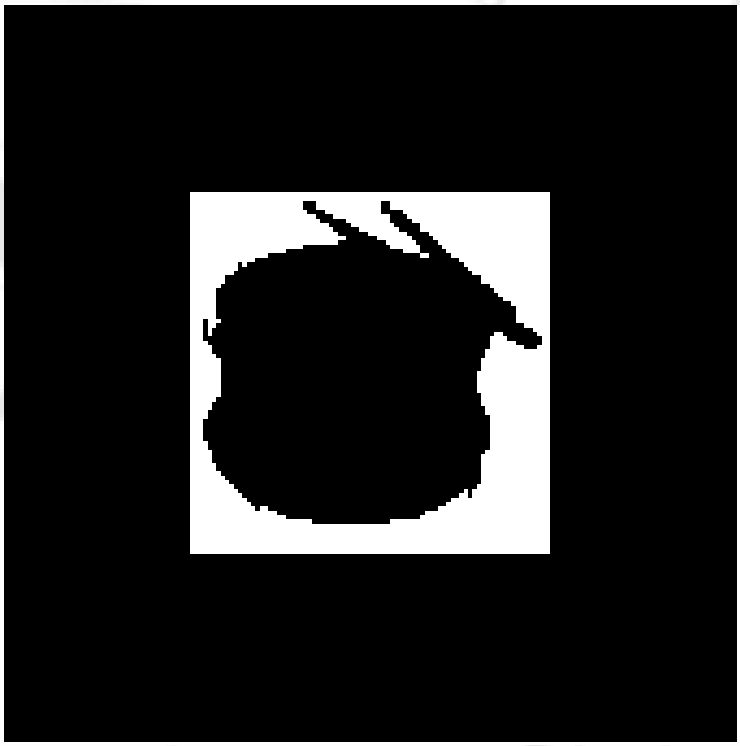
TAMBOR

El tambor es un instrumento de percusión basado en el golpeo de una membrana que vibra situada en una caja de resonancia redonda que hace que el sonido generado se amplifique.

El tambor se puede golpear con la mano o con unos instrumentos de madera alargados llamados baquetas.

Es un instrumento muy utilizado para llevar el ritmo básico en bandas, comparsas y todo tipo de grupos musicales de celebraciones populares.

¿Has visto este instrumento en las celebraciones de tu ciudad?

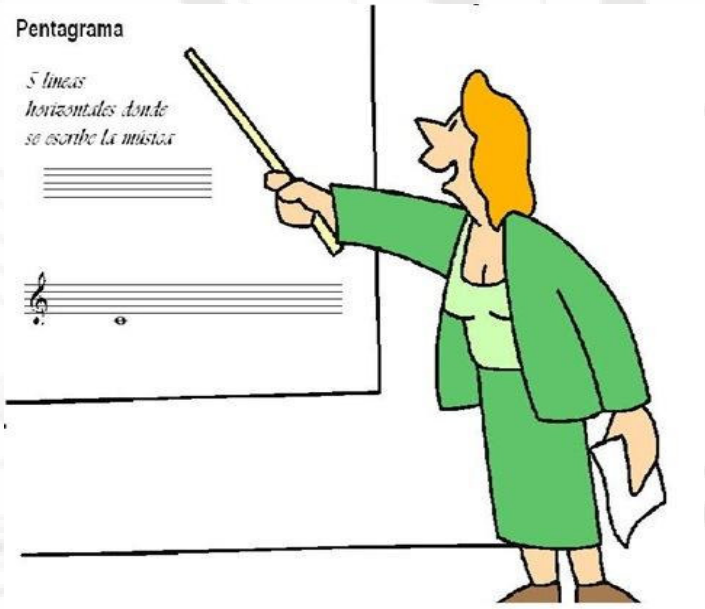


¿Reconoces el sonido de este instrumento?

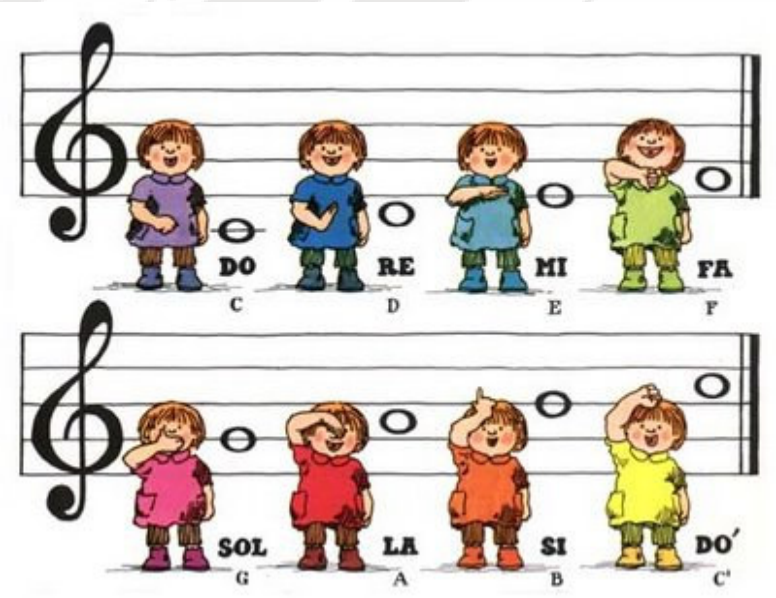
ESCALA MUSICAL

La escala musical esta formada por 7 notas de diferentes sonidos, y son las que utilizamos para crear todo tipo de música. Están ordenadas dentro de un pentagrama, que es el lugar donde escribimos las notas y donde las diferenciamos unas de otras. El pentagrama se compone de 5 líneas horizontales donde escribimos las notas, tanto encima de las líneas como en sus espacios.

Las diferentes notas que existen son: DO, RE MI, FA, SOL, LA, SI, y se representan en el pentagrama con símbolos que nos dicen la duración de las notas, para poder crear una canción.

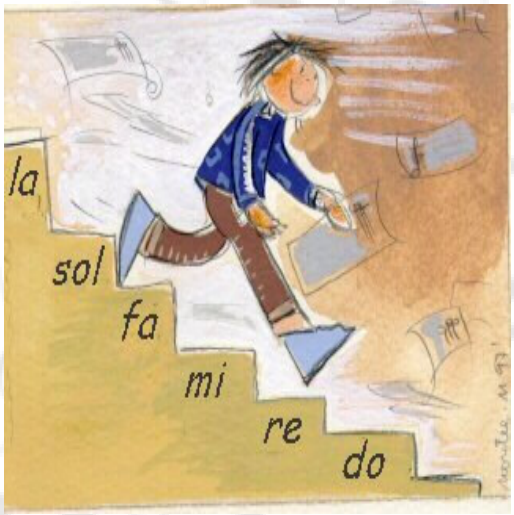


La escala musical podemos tratarla como una escalera, donde abajo están las notas graves y mientras vamos subiendo las notas son más agudas.



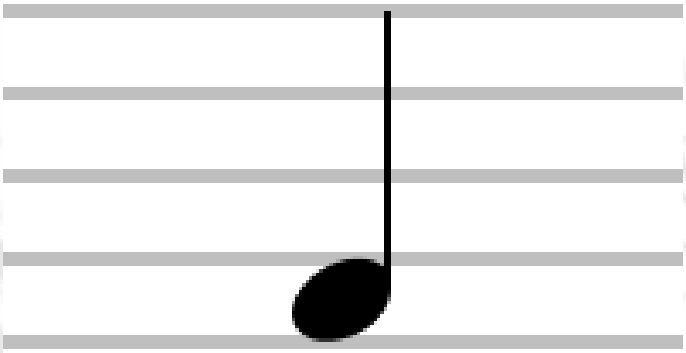
Podemos subir esta escalera o bajarla, o podemos coger distintas notas en distintos escalones y formar una canción.
El problema que tenemos ahora es cómo decir cuanto dura cada nota. Para eso hemos creado distintas figuras que nos dicen cuanto dura cada una. Estas son las principales:

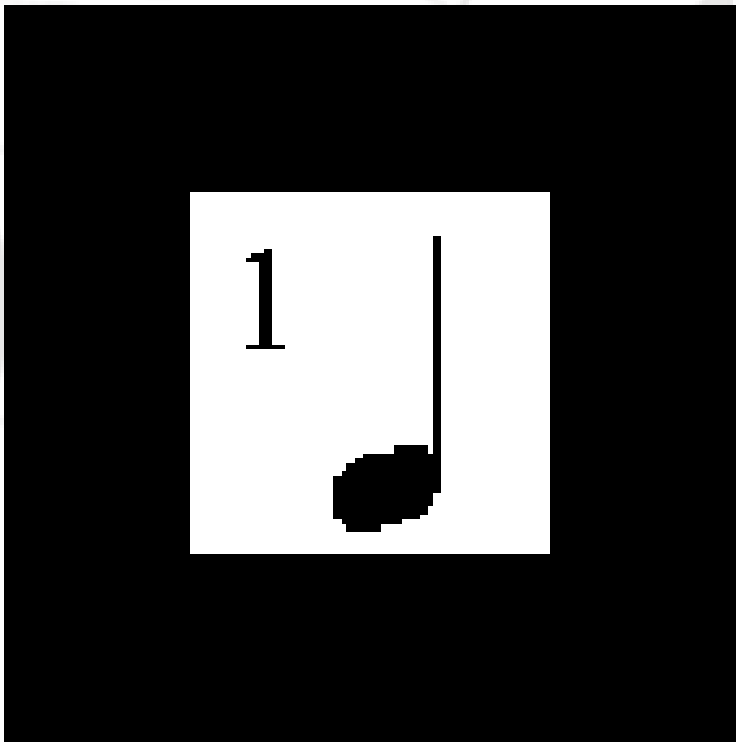
-Corchea
-Negra
-Blanca
-Redonda



NEGRA

La negra es la figura de referencia, por tanto empezamos primero por ella. Todas las demás notas se basan en ella y duran más tiempo que ella, o menos. Nos indica una duración de sonido en el tiempo. Se representa así:



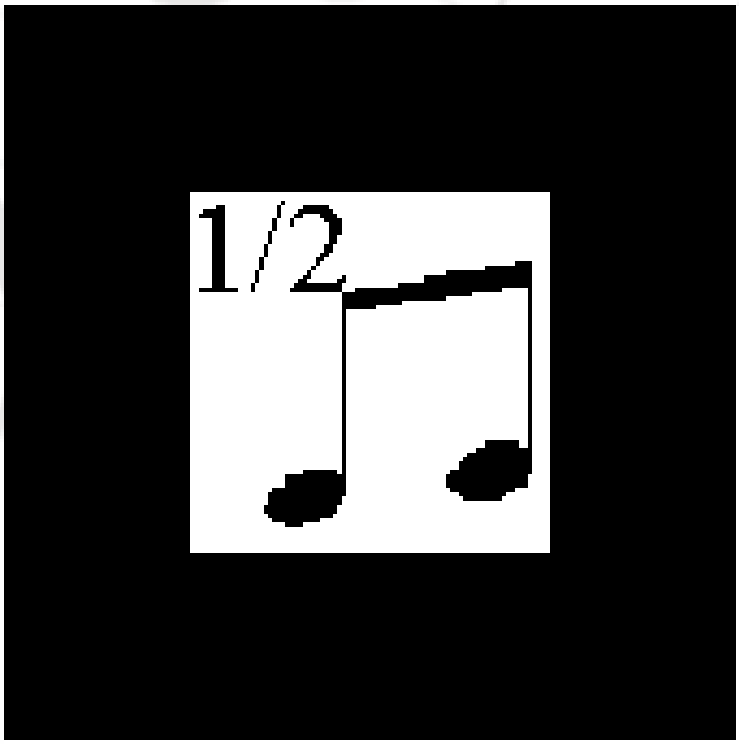


Fíjate que el sonido del piano coincide con el sonido del metrónomo.

CORCHEA

La corchea es una figura cuya duración es la mitad de una negra. Esto quiere decir que en el tiempo que dura una negra podemos poner dos corcheas. Su forma es la siguiente:

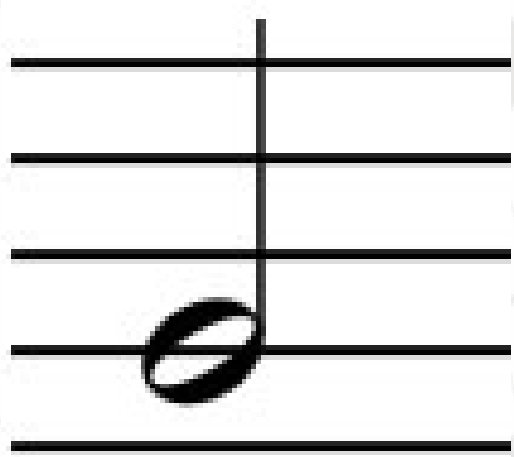
Si se juntan dos corcheas pueden adoptar esta forma:

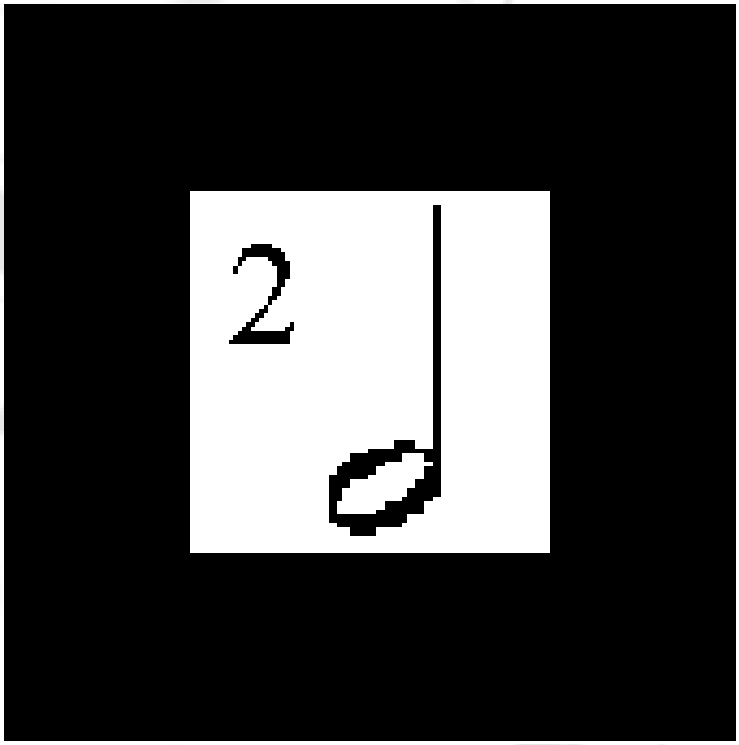


¿Eres capaz de escuchar cómo cada golpe de metrónomo suenan dos corcheas de piano?

BLANCA

La blanca es una figura que dura dos veces más que la negra, es decir, el doble. Esto quiere decir que en la duración de una blanca caben dos negras, o cuatro corcheas. Su figura es ésta:

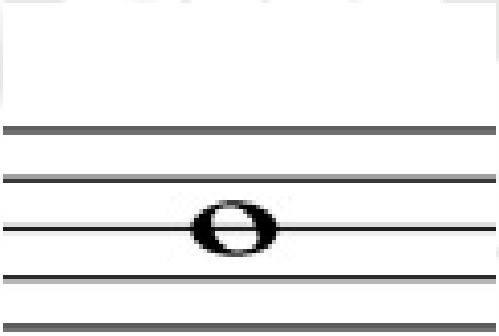


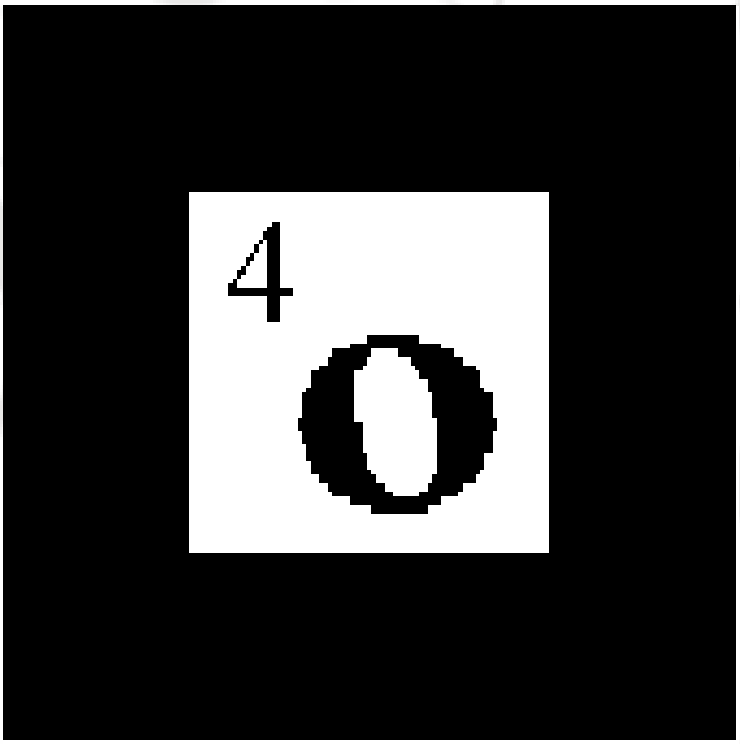


¿Escuchas cómo cada dos golpes de metrónomo suena uno de piano?

REDONDA

La redonda es una figura que dura cuatro veces más que la negra. Esto quiere decir que en la duración de una redonda caben dos blancas, cuatro negras u ocho corcheas. Su figura es:





¿Escuchas cómo cada cuatro golpes de metrónomo suena uno de piano?

CUALIDADES DEL SONIDO

Ahora vamos a ver distintas cualidades que tiene el sonido y que escuchándolas seremos capaces de distinguirlas fácilmente. Estas cualidades son:

-Altura
-Timbre
-Intensidad

Con estas cualidades podremos distinguir sonidos iguales que suenan en distintos instrumentos y aprender el concepto de sonido grave y sonido agudo.

ALTURA

La altura es la cualidad del sonido que nos da el “tono” del mismo, es decir, cómo escuchamos nosotros ese sonido, si lo escuchamos grave o lo escuchamos agudo. Veamos el ejemplo para aclararnos:

Al escuchar el piano, podemos escuchar cómo es la misma nota (es DO) pero cada vez se escucha más grave. Comprobamos entonces como una misma nota puede ser más grave o más aguda. ¿Lo escuchas?

TIMBRE

El timbre es la cualidad que nos permite distinguir sonidos creados por diferentes instrumentos, siendo incluso la misma nota musical. Debido a esta misma cualidad es posible reconocer a una persona por su voz, puesto que cada persona tiene un “timbre” de voz distinto.

Escuchamos como la canción es siempre igual, pero distinguimos perfectamente cuando la toca la guitarra, el piano o el bajo. ¿Oyes la diferencia?

INTENSIDAD

La intensidad del sonido es la cualidad que nos dice si un sonido es fuerte o débil. Esta cualidad es muy fácil de entender:

Nosotros podemos susurrar una frase, decirla en voz alta, o leerla gritando. La altura y el timbre de estos sonidos serían los mismos (somos siempre la misma persona y leemos el texto siempre de la misma forma, ni más agudo ni más grave) pero escuchamos la frase más alto o más bajo.

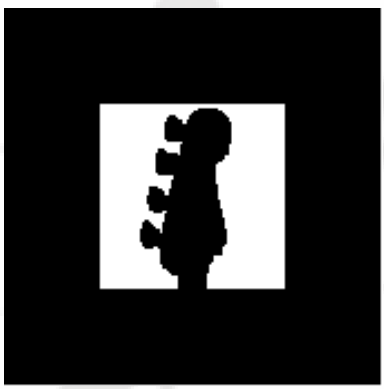
A esto se le define como intensidad del sonido.

¿Es fácil, no?

BANDA MUSICAL

Por último vamos a ver que instrumentos pueden formar normalmente una banda musical. Los instrumentos básicos son los que nos dan el ritmo de la canción, y estos son el bajo y la percusión (en nuestro caso un tambor). Después añadimos instrumentos que pueden ser utilizados tanto para acompañar rítmicamente como para realizar melodías. Estos son la guitarra acústica. Finalmente vemos los instrumentos que normalmente crean los solos y las melodías principales de la canción. Estos son la guitarra eléctrica y en menor medida el violín y el piano.

Esto es totalmente a gusto del grupo y puedes crear melodías con instrumentos rítmicos y crear ritmos con cualquier instrumento.¡¡ La música esta a disposición de nuestra imaginación!!






Al escuchar la banda nos damos cuenta de los instrumentos que realizan un ritmo constante y los que van creando una melodía. También podemos escuchar la diferencia de sonidos según los instrumentos y cómo trabajan conjuntamente para crear la canción. ¡¡¡Intenta escuchar distintas canciones y ver qué instrumentos suenan y qué ritmo de la canción realizan!!!

CONCLUSIONES

Esperamos que este divertido libro haya servido para aprender los conceptos básicos de la música, como qué tipos de instrumentos existen, cómo suenan, los principios del lenguaje musical, cómo son el pentagrama y las distintas notas, y las diferentes cualidades del sonido.

Con esto intentamos que os divirtáis aprendiendo estos conceptos y lo veáis como algo divertido y ameno, y que disfrutéis con la música.

Esperamos que lo hayáis pasado muy bien.

¡¡Hasta la próxima!!