Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de una aplicación para la Web utilizando el Web Audio API

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Víctor Hernández Medrano

Tutor: Manuel Agustí Melchor

Curso 2018/19

Resumen

Este proyecto de fin de grado se centra en el estudio de técnicas de procesado y síntesis de sonido bajo el control de un navegador web, desarrollando una aplicación para la Web basada en la utilización de una interfaz de programación de aplicaciones implementada en el lenguaje de programación JavaScript y especializada para el control del audio. Con el objetivo principal de entender el correcto funcionamiento de la herramienta y ponerla en práctica en una aplicación web real.

La solución ha sido programar pequeños ejemplos que ayudan a comprender la mecánica de programación y además sirven de base para el desarrollo de la aplicación final.

Después de todo el proceso se ha conseguido elaborar una aplicación web cimentada en el manejo del sonido gracias a los recursos aprendidos y dotándole de interactividad con el usuario.

Palabras clave: aplicación web, sonido, JavaScript, interfaz de programación de aplicaciones, navegador web.

Resum

Aquest projecte de fi de grau se centra en l'estudi de tècniques de processament i síntesi de so sota el control d'un navegador web, desenvolupant una aplicació per a la Web basada en la utilització d'una interfície de programació d'aplicacions implementada en el llenguatge de programació Javascript i especialitzada per al control de l'àudio. Amb l'objectiu principal d'entendre el correcte funcionament de l'eina i posar-la en pràctica en una aplicació web real.

La solució ha sigut programar xicotets exemples que ajuden a comprendre la mecànica de programació i a més serveixen de base per al desenvolupament de l'aplicació final.

Després de tot el procés s'ha aconseguit elaborar una aplicació web cimentada en el maneig del so gràcies als recursos apresos i dotant-li d'interactivitat amb l'usuari.

Paraules clau: aplicació web, so, Javascript, interfície de programació d'aplicacions, navegador web.

Abstract

This final project focuses on the study of sound processing and synthesis techniques under the control of a web browser, developing a Web application based on the use of an application programming interface implemented in the programming language JavaScript and specialized for audio control. With the main objective of understanding the correct functioning of the tool and putting it into practice in a real web application.

The solution has been to program small examples that help to understand the mechanics of programming and also serve as the basis for the development of the final application.

After the whole process, a web application based on sound management has been developed thanks to the resources learned and providing interactivity with the user.

Keywords: web application, sound, JavaScript, application programming interface, web browser.

Glosario

• **Análisis FFT:** análisis frecuencial de una señal basado en una transformada rápida de Fourier.

• **API:** conjunto de protocolos y herramientas usado por desarrolladores.

• **Buffer de sonido:** espacio en memoria utilizado para reunir los bytes que representan los datos de un pequeño sonido.

• **Contexto de audio:** es la instancia donde se crean los nodos de audio y se aplican las modificaciones del sonido.

• **DOM:** Modelo de Objetos del Documento, proporciona una representación estructural de un documento HTML o XML, permitiendo la modificación de su contenido o su presentación visual.

• **Efecto Doppler:** fenómeno por el cual la frecuencia de las ondas del sonido percibida por el oyente varía cuando la fuente se desplaza respecto al oyente.

• **Evento:** acción que realiza el usuario y es captada por la página web para dotar de interacción la visualización.

• **Formato estéreo:** sonido estereofónico, está definido por dos canales de sonido para crear una experiencia más natural al escucharlo.

• **Formato mono:** sonido monoaural, sólo está definido por un canal de sonido.

• **Fuentes:** emisores de sonido, encargados de posicionar el sonido, aplicarle efectos y reproducirlo.

• **Grafo de encaminamiento:** representa las etapas de procesado intermedio a través de los nodos desde la fuente hasta el oyente.

• **Nodos:** puntos de procesado de la señal que confluyen en el mismo destino.

• **Oscilador:** generador de una señal periódica que se utiliza para sintetizar sonidos básicos.

• **Oyente:** representa la posición del usuario en la escena, por lo que hay que calcular la confluencia de todas las fuentes de sonido para esas coordenadas.

• **W3C:** consorcio internacional encargado de generar ayudas y estándares que permiten el desarrollo de la *World Wide Web* de forma prolongada.

• **WebKit:** plataforma para aplicaciones web que funciona como base de múltiples navegadores, como por ejemplo Safari.

Tabla de contenidos

1. Introducción

La evolución en las tecnologías web ha propiciado el desarrollo de aplicaciones que permiten la interacción del usuario. Con la llegada de HTML5, ahora también es posible la comunicación mediante la reproducción de audio y vídeo, la generación de imágenes basadas en mapas de bits, las animaciones realizadas con CSS3 y los gráficos tridimensionales; lo que abre el abanico al desarrollo de aplicaciones complejas como los videojuegos, directamente en la web. No solo esto, sino que también es posible recoger imagen y sonido, en directo, mediante la cámara y el micrófono, que estaba restringido a aplicaciones locales a un computador. Esto facilita el camino a otro nivel de interacción con el usuario, ya que permite el acceso al mundo real (fuera del computador) desde una página web, sin necesidad de complementos de terceros y facilita la portabilidad de este tipo de aplicaciones a la gran variedad de plataformas disponibles para dar soporte a las páginas y aplicaciones web.

Como remarcan algunos artículos[1] de opinión, el mercado de aplicaciones que hacen uso del audio o de la voz para interpelar al usuario están en pleno auge. Esto requiere tanto de capacidades de generación y captura de sonido, como de síntesis y reconocimiento de voz. En el presente trabajo nos vamos a centrar en el uso del sonido y la propuesta del W3C del estándar *Web Audio API*.

El uso de aplicaciones basadas en APIs propietarias y aplicaciones nativas a las plataformas puede trasladarse ahora a las aplicaciones web, gracias al desarrollo de APIs estándares dentro del paraguas de HTML5. Este trabajo está orientado al diseño y desarrollo de una aplicación web que haga uso de técnicas de captura, procesado y síntesis de audio para interactuar con el usuario. Esto hace necesaria una primera etapa de acercamiento a las APIs disponibles y el desarrollo de pequeños ejemplos que muestren si existen diferencias entre las API existentes y las diferentes plataformas de navegadores web. Las capacidades, así validadas, se utilizarán para diseñar y desplegar varios ejemplos de interacción mediante el audio en una aplicación web.

1.1 Objetivos

Los objetivos de este trabajo son elaborar un diseño y desarrollo de varias aplicaciones web que hagan uso de técnicas de procesado y síntesis de sonido para interactuar con el usuario.

Para ello, en primer lugar, será necesario estudiar las opciones disponibles y ver cómo se utiliza el *Web Audio API* con pequeños ejemplos.

Después se podrá entrar a plantear una aplicación que haga uso de la síntesis de sonidos básicos para realizar una versión del clásico juego del Simón.

En tercer lugar, para ofrecer un ejemplo complejo de aplicación basada en el uso del audio, abordaremos la realización de un instrumento musical sencillo para generar sonido mediante muestras (esto es, un banco de sonidos reales) y con una interfaz

[1] https://lapastillaroja.net/2019/03/el-boom-de-los-asistentes-de-voz-y-sus-implicaciones/

compleja que permita la edición de la partitura, la sonorización de esta en sincronía con la actuación sobre el instrumento y la gestión de ficheros que representen la composición que el usuario esté llevando a cabo. Esto supone un ejemplo de aplicación educativa que puede ser utilizada en el contexto de las enseñanzas de primaria donde se emplea la flauta dulce para introducir a los alumnos en el mundo de la música.

1.2 Estructura

Para abordar los objetivos marcados se desarrollan una serie de etapas que componen la estructura del trabajo.

El capítulo 1 contiene la introducción que estamos leyendo y que es una presentación de intenciones.

El capítulo 2, llamado estado del arte, recoge planteamientos similares a nuestro problema y solución, además de herramientas hardware y software que pueden trabajar de manera parecida en torno al tratamiento de sonido.

El capítulo 3, análisis del problema, muestra el problema concreto que se aborda en el trabajo, los requerimientos y sus especificaciones.

El capítulo 4, denominado introducción al *Web Audio API*, nos acerca a la implementación del *Web Audio API* a través de pequeños ejemplos desarrollados para demostrar su funcionamiento.

En el capítulo 5, titulado diseño y desarrollo de la solución propuesta, se exponen los detalles propios de la implementación de las diferentes versiones y aplicaciones planteadas, encontrando en el anexo el código completo de todas las versiones.

El capítulo 6 es asignado al despliegue de las aplicaciones elaboradas en el capítulo anterior en un alojamiento web.

El capítulo 7, destinado a las pruebas, exhibe los resultados del funcionamiento del *Web Audio API* en diferentes navegadores web.

Finalmente, las conclusiones y trabajos futuros resumen los resultados alcanzados y propondrán algunas líneas de expansión sobre los desarrollos realizados.

2. Estado del arte

La primera forma de reproducir sonidos en la web fue a través de la etiqueta HTML <bgsound>, que permitía a los creadores de contenido web reproducir automáticamente música de fondo cuando una persona entraba a sus páginas. Esta función solo estaba disponible en el navegador Internet Explorer, y nunca fue estandarizada ni utilizada por otros navegadores. El ya descontinuado navegador Netscape, de la compañía *Netscape Communications*, implementó una característica similar con la etiqueta <embed>, proporcionando una funcionalidad básicamente equivalente [1].

Flash fue la primera forma de reproducción de audio entre navegadores en la Web, pero tenía el inconveniente significativo de requerir un complemento para ejecutarse. Más recientemente, los proveedores de navegadores se han unido al elemento <audio> de HTML5, que proporciona soporte nativo para la reproducción de audio en todos los navegadores modernos [1].

Aunque el audio en la Web ya no requiere de ningún complemento, la etiqueta <audio> tiene limitaciones significativas para implementar juegos sofisticados y aplicaciones interactivas, como veremos más adelante en el análisis del problema.

En aplicaciones de escritorio, FMOD [7] es un motor propietario de efectos de sonido y una herramienta de creación para videojuegos y aplicaciones desarrolladas por *Firelight Technologies*. Su plataforma FMOD Studio proporciona todas las herramientas necesarias para diseñar, construir y optimizar el audio adaptativo. Además, ayuda a organizar y administrar proyectos con ajustes preestablecidos compartidos y una jerarquía de búsqueda y etiquetado. Nos permite crear sonido y música en evolución dinámica, ajustar el volumen, tono, aleatorización, enrutamiento y efectos DSP, es decir, efectos sobre los procesos de la señal sonora, véase Imagen 1.

Podemos utilizar instantáneas de mezclador para crear paisajes sonoros que reaccionen naturalmente a un juego o realizar una audición rápida de los cambios directamente en el editor, conectarnos a un juego en ejecución, escuchar los cambios en tiempo real en el dispositivo, entre otras muchas funciones.

Imagen 1: Hoja de trabajo de FMOD Studio [7].

FMOD Studio puede usarse [7] con Unity y Unreal Engine 4 o integrarse con un motor de juego personalizado utilizando las versiones en C++ o C#. El enfoque basado en datos de FMOD Studio significa que los comportamientos de audio a través de su interfaz visual, sin programar explícitamente, son accesibles y editables para los diseñadores de sonido. Las herramientas de diseño de sonido reducen la necesidad de asistencia del programador. Los formatos de compresión específicos de la plataforma y las herramientas de creación de perfiles integradas facilitan la obtención de un gran rendimiento.

Otra herramienta de tratamiento del audio es la API de audio desarrollada por *Loki Software*[2] llamada OpenAL [6] (*Open Audio Library*), una interfaz de programación de aplicaciones de sonido multiplataforma y, desde hace un par de años, ya libre de las ataduras que *Creative Labs* imponía mientras mantuvo el sitio web para el renderizado competente de sonido multicanal tridimensional. Esta herramienta ha sido creada para su uso en videojuegos, puede utilizar audio con diferentes frecuencias de muestreo, en formato mono o etéreo y con una profundidad de 8 o 16 bits. El motor de interpretación tiene la función de realizar el manejo del sonido como la atenuación, efecto *Doppler*, etc.

[2] https://www.igdb.com/companies/loki-software

Otra plataforma relacionada con el tratamiento del audio es *Pro Tools*[3], un equipo de producción musical o DAW (*Digital Audio Workstation*), estudio de edición, mezcla y grabación multipista de audio y MIDI (*Musical Instrument Digital Inteface*), que anexa hardware y software. Es usado en todo el mundo para trabajos profesionales [8].

Para terminar, *Cubase*, desarrollado por la compañía *Steinberg*, es uno de los paquetes de software de producción musical más potentes del mundo, véase Imagen 2. Entre sus herramientas podemos encontrar: alineamiento de audio para sincronizar líneas de voz apiladas, posibilidad de componer con acordes de forma creativa, un editor de percusión para crear *beats* y *grooves*, sintetizadores analógicos clásicos, entre otras muchas funciones [9].

Imagen 2: Herramienta *Retrologue 2* de *Cubase* [9].

El 14 de noviembre de 2018, *Steinberg* presentó las versiones actualizadas *Pro*, *Artist* y *Elements* de *Cubase* 10, donde encontramos instrumentos virtuales, efectos y miles de sonidos preparados para cualquier tipo de compositor, tanto profesional como para una persona que está empezando a producir [9].

[3] https://www.avid.com/es/pro-tools

Como apunte final de este capítulo, las aplicaciones de escritorio complejas no son portables entre plataformas. En este trabajo analizaremos nuestras opciones para desarrollar una aplicación web basada en el uso del sonido y con unas características de interactividad alta mediante la propuesta del W3C acerca del *Web Audio API*.

3. Análisis del problema

El desarrollo de aplicaciones web no es exclusivo para teléfonos móviles o tabletas, al igual que las páginas web no son creadas para su visualización sólo en ordenadores personales, esto es una ventaja respecto a las aplicaciones de escritorio que normalmente sólo están disponibles para computadores con un hardware superior al de un dispositivo móvil. Sin embargo, una aplicación web es capaz de soportar una carga de trabajo suficiente como para satisfacer algunas de las competencias que anteriormente comentábamos en el capítulo 2, estado del arte, como podemos ver en la Imagen 3 donde se nos permite aplicar a una grabación efectos de distorsión, retraso, trémolo, cambio de tono, entre otros.

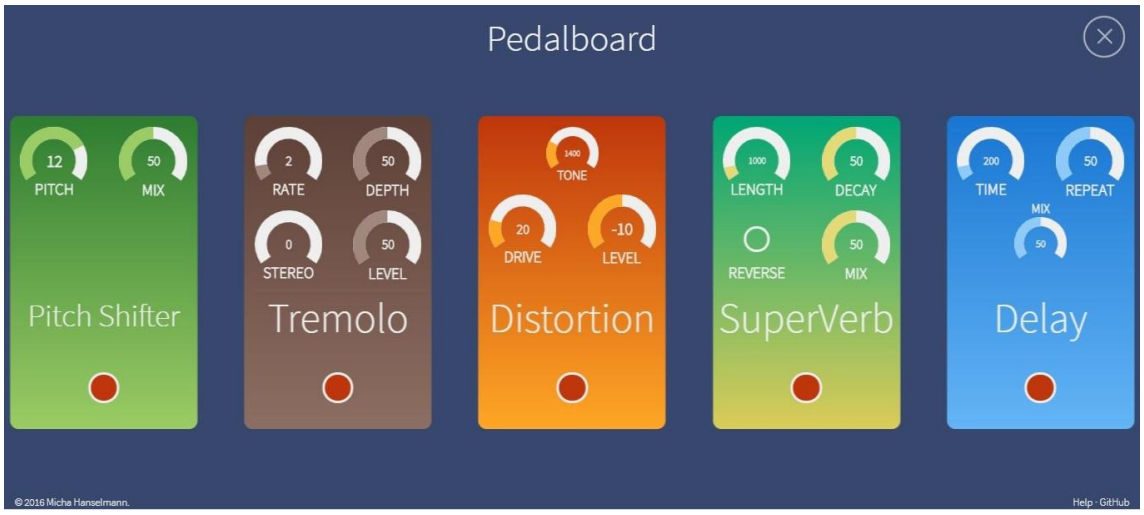


Imagen 3: Aplicación web *Pedalboard* creada por *Micha Hanselmann*. [4]

Otra aplicación web que cumple con el tratamiento del sonido de forma *online* es la desarrollada por *Busy Beats* [5] ,véase Imagen 4, que posibilita la creación de diferentes fuentes de sonido y su representación en el espacio.

[4] https://deermichel.me/pedalboard/#ZGlzdG9ydGlvbiwxNDAwLDIwLC0xMCwx
[5] https://busybeats.co.uk/

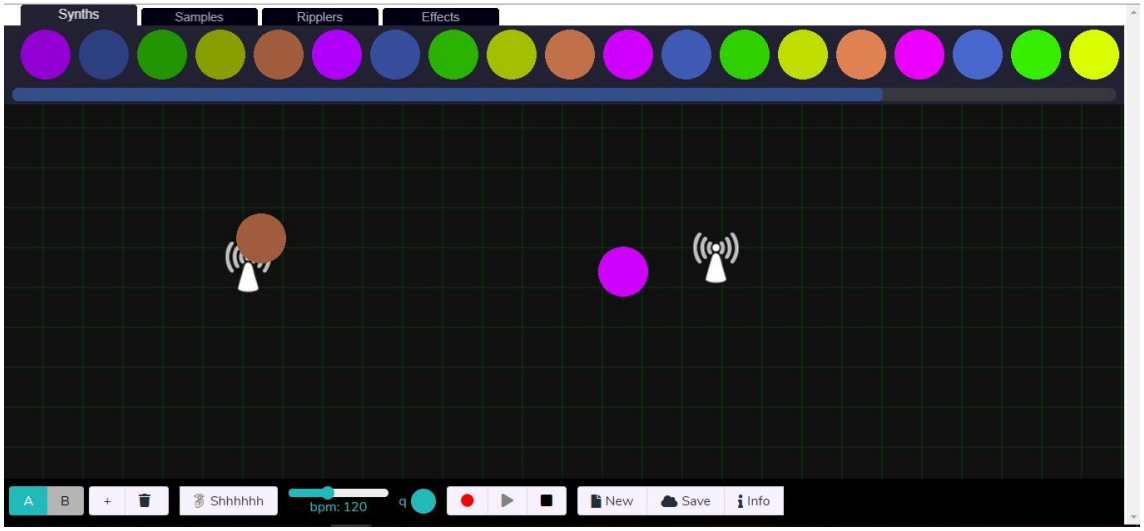


Imagen 4: Aplicación web *Busy Beats*.

Para construir una aplicación web es necesario que pueda ser ejecutada en cualquier tipo de navegador y el elemento HTML <audio> posee una serie de limitaciones[6] que debilitan el hecho de poder trabajar con sonido dentro de una aplicación web, tales como: no hay un control del tiempo demasiado preciso, el límite de reproducción de varios sonidos a la vez es muy bajo, no se puede almacenar previamente un sonido de forma segura, no tiene capacidad de aplicar efectos en tiempo real, no existe forma de analizar el sonido..., entre otras.

Por estas limitaciones, ha habido varios intentos de crear una potente API de audio en la Web para intentar solucionarlas. Un ejemplo notable es la API de datos de audio que fue diseñada y prototipada por *Mozilla Firefox*. El enfoque de *Mozilla* comenzó con un elemento <audio> y amplió su API de JavaScript con características adicionales. Esta API tiene un grafo de encaminamiento del audio limitado, y no se ha adoptado más allá de su primera implementación. Actualmente está en desuso en *Firefox* a favor del *Web Audio API*.

El *Web Audio API* equipa un sistema fuerte y versátil[7] para manejar audio en las plataformas web, habilitando a los programadores entre otras cosas, elegir diferentes fuentes de sonido, añadir efectos al audio, generar visualizaciones de sonidos o emplear efectos espaciales [3].

El primer borrador del *Web Audio API* apareció en W3C [2] durante 2011 aunque el mayor crecimiento se le puede deber a Google Chrome, ya que gracias al interés de Google por su navegador web, los navegadores comenzaron a convertirse en la parte más importante de un ordenador [10].

A diferencia de la API de datos de audio, el *Web Audio API* es un modelo completamente nuevo, completamente separado de la etiqueta <audio>, aunque hay

[6] https://bocoup.com/blog/fieldrunners-playing-to-the-strengths-of-html5-audio-and-web-audio
[7] https://medium.com/samsung-internet-dev/web-audio-en-distintas-plataformas-1be66cbe4fd7

puntos de integración con otras APIs Web. Es una API de JavaScript de alto nivel para procesar y sintetizar sonido en aplicaciones web, pero puede ser difícil de usar para algunos propósitos ya que todavía está en desarrollo.

El objetivo de esta API es incluir capacidades que se encuentran en los motores de juegos modernos y algunas de las tareas de mezcla, procesamiento y filtrado que se encuentran en las aplicaciones modernas de producción de audio de escritorio.

El resultado es una API versátil que se puede usar en una variedad de tareas relacionadas con el sonido, desde juegos hasta aplicaciones interactivas y visualizaciones de síntesis de sonidos muy avanzadas.

El *Web Audio API* implica controlar operaciones de audio dentro de un contexto de audio, y ha sido diseñado para aceptar un grafo de encaminamiento modular, es decir, las operaciones de audio son llevadas a cabo en nodos, que están enlazados juntos para formar un grafo de encaminamiento de audio. Gracias a esta estructura modular se garantiza adaptabilidad para producir controles de audio complejos con efectos activos [21].

4. Introducción al *Web Audio API*

A lo largo de este capítulo, iremos viendo diferentes ejemplos desarrollados para facilitar la explicación del funcionamiento del *Web Audio API* [2], y gracias a estos ejemplos acabaremos con el desarrollo de 2 aplicaciones.

Todas las operaciones de sonido en el *Web Audio API* se manejan dentro de un contexto de audio. Por lo tanto, lo primero que debemos hacer es crear este contexto.

Cabe destacar que, el navegador Safari requiere un prefijo de webKit para ser compatible con AudioContext, por lo que debe usar esa línea en lugar del new AudioContext();

Por lo tanto, para llevar a cabo la puesta en marcha necesitaremos un documento HTML, donde nos centraremos en la ejecución de un *script* en el lenguaje de programación JavaScript que será el encargado del funcionamiento de toda la aplicación.

En este sencillo ejemplo podemos observar la reproducción de una frecuencia gracias a un oscilador y a un analizador de la señal creada por el oscilador, entraremos en detalle de sus funcionamientos un poco más adelante. Insertamos en el documento HTML la etiqueta <script> </script> que debe envolver todo nuestro código JavaScript. Esta etiqueta tenemos la opción de colocarla entre las etiquetas <head> si queremos que el *script* cargue antes que el contenido HTML o entre las etiquetas <body> si queremos que el *script* cargue después que el contenido HTML, ayudando a mejorar la velocidad de carga de la web.

Declaramos al principio del *script* el objeto misNodos donde se almacenarán todos los nodos que crearemos y sus valores.

```
var misNodos = {};
```

Ahora sólo queda crear el contexto, el oscilador y el analizador. Conectamos el oscilador con el analizador y mandamos a reproducir con el método *start()*.

```
//Creamos el Contexto de Audio
const AudioContext = window.AudioContext || window.webkitAudioContext
|| window.mozAudioContext;
const context = new AudioContext();
//Creamos el Oscilador
misNodos.osc = context.createOscillator();
misNodos.osc.frequency.value = 261.63;//En Hz
misNodos.analizador = context.createAnalyser();
misNodos.analizador.fftSize = 1024;
//Conectamos los nodos
misNodos.osc.connect(misNodos.analizador);
misNodos.analizador.connect(context.destination);
//Reproducir
misNodos.osc.start(0);
```

Dentro del contexto de audio, necesitamos crear las fuentes, ya sean por medio de muestras de audio como un archivo de sonido, a través del oscilador o por una transmisión de audio. Pueden existir varias fuentes dentro del mismo contexto de audio.

Una vez creadas las fuentes, encadenamos los nodos de audio que creamos oportunos para realizar nuestra operación, formando un grafo de encaminamiento de audio hasta llegar al destino.

Existen diferentes tipos de nodos de audio en el funcionamiento del *Web Audio API* con diferentes funcionalidades, aunque los únicos que son imprescindibles para el correcto funcionamiento son el nodo fuente y el nodo destino, los agrupamos en estos 4 tipos de nodos como se muestra en la Figura 1.

- Nodos fuente: *buffers*, osciladores, entrada de micrófono...

- Nodos de modificación: balance (*panners*), filtros, compresores...

- Nodos de análisis: en el dominio temporal o frecuencial...

- Nodos de destino: *buffers*, salidas de audio como altavoces...

Figura 1: Grafo de encaminamiento de un contexto de audio.

Los sonidos en una escena 3D son relativos al oyente. Debido a que sólo existe un oyente, sus características son definidas por el contexto de audio y no por un nodo particular. El objeto que representa al oyente es accesible a través de la propiedad *listener* del contexto [18].

Por la brevedad de la exposición nos centraremos en los aspectos más próximos a nuestros objetivos, aunque cabe mencionar la posibilidad de creación de varias fuentes, manejo del volumen, *pitch*, micrófono, *streaming*, FFT, efecto *Doppler* y la posibilidad de utilizar *OfflineAudioContext* si sólo queremos procesar datos de audio sin reproducirlos, entre otras funciones del *Web Audio API*.

4.1 Oscilador

Un oscilador es una forma de onda que se repite y tiene una frecuencia y una amplitud máxima. Una de las características más importantes del oscilador, aparte de su frecuencia y amplitud, es la forma de su forma de onda. Las cuatro formas de onda de oscilador más utilizadas son: seno, triángulo, cuadrado y diente de sierra, aunque también es posible crear formas de onda personalizadas.

Diferentes formas son adecuadas para diferentes técnicas de síntesis y producen diferentes sonidos.

Para representar la forma de onda que se repite utilizaremos la propiedad *type* de nuestro oscilador previamente creado.

```
misNodos.osc.type = 'sine'|'square'|'triangle'|'sawtooth';
```

Para crear la forma de onda personalizada utilizamos el método *setPeriodicWave()* que establecerá el tipo automáticamente en personalizado.

4.2 Tiempo

Una de las cosas más importantes para el desarrollo de una aplicación que utiliza el sonido es administrar el tiempo. Para la precisión necesaria aquí, usar el reloj de JavaScript no es la mejor idea, porque no es lo suficientemente preciso. Sin embargo, el *Web Audio API* viene con la propiedad *currentTime*, que es una marca de tiempo de doble hardware que se puede usar para programar la reproducción de audio. Se inicia cuando se declara el contexto de audio.

Hay dos métodos para trabajar el tiempo, con la propiedad *setValueAtTime* y con la propiedad *exponentialRampToValueAtTime*.

La propiedad *setValueAtTime(value, startTime)* programa el cambio del valor en el momento preciso. Por ejemplo, este es el código si queremos cambiar el valor de frecuencia a 261.6 del oscilador dentro de un segundo.

```
misNodos.osc.frequency.setValueAtTime(261.6, context.currentTime + 1);
```

La propiedad *exponentialRampToValueAtTime(value, endTime)* programa un cambio gradual del valor. Este código reducirá exponencialmente el volumen del oscilador en un segundo, lo cual es una buena manera de detener el sonido suavemente.

```
misNodos.osc.exponentialRampToValueAtTime(0.001, context.currentTime + 1);
```

No podemos usar 0(cero) como *value* porque el valor debe ser positivo, por lo que usamos un valor muy pequeño.

Como apunte final, una vez que se detiene un oscilador, no se puede volver a iniciarlo. Esta característica optimiza el rendimiento. Lo que podemos hacer es declarar una función que será la responsable de crear nodos de oscilador, reproducirlos y detenerlos.

4.3 Buffer

El buffer de audio puede contener uno o más canales y entre sus atributos encontramos la frecuencia de muestreo, longitud de la señal, duración de la señal en segundos y el número de canales del buffer.

Es útil para reproducir fragmentos de audio cortos en un entorno que requiera flexibilidad.

Entre sus métodos observamos el método *start*, utilizado para programar la reproducción del sonido y el método *stop* que detendrá la reproducción.

4.4 Cargar muestras

Para cargar sonidos de corta y media longitud se hace uso de una petición vía http al servidor para buscar los archivos de sonido, como pueden ser del tipo MP3, AAC, WAV, OGG, entre otros.

Otra opción es exponer el audio en el DOM usando la etiqueta <audio>

```
<audio src="pista.mp3" type="audio/mpeg" ></audio>
```

Obtener el elemento de audio y pasarlo al contexto de audio.

```
const audioElement = document.querySelector('audio');
const track = audioCtx.createMediaElementSource(audioElement);
```

En HTML5, los 3 formatos de audio soportados son MP3,WAV y OGG, pero no todos son compatibles con todos los navegadores, véase Tabla 1. Siendo solo el formato MP3 el soportado por todos los navegadores principales.

	Edge	Chrome	Firefox	Safari	Opera
MP3	SI	SI	SI	SI	SI
WAV	NO	SI	SI	SI	SI
OGG	NO	SI	SI	NO	SI

Tabla 1: Formatos de audio soportados por diferentes navegadores web.

4.5 Nodo Analizador

Aunque hay muchos tipos de nodos, como anteriormente hemos descrito, cabe destacar el nodo analizador ya que proporciona en tiempo real datos de la fuente de audio. Este nodo no modifica el sonido por lo que puede estar conectado en cualquier lugar del grafo de encaminamiento del audio y es una buena opción para poder visualizar el sonido y comprender lo que vamos a escuchar.

Los resultados están basados en un análisis FFT de un tamaño de buffer determinado.

Para comprobar su funcionamiento, definimos una serie de elementos <button> para dotar de funcionalidad al oscilador con propiedades como: reproducir, detener, cambiar la frecuencia y cambiar la forma de la onda, como hemos visto en el apartado 4.1 de este capítulo. Finalmente, creamos un elemento <canvas> como se muestra en la Imagen 5.

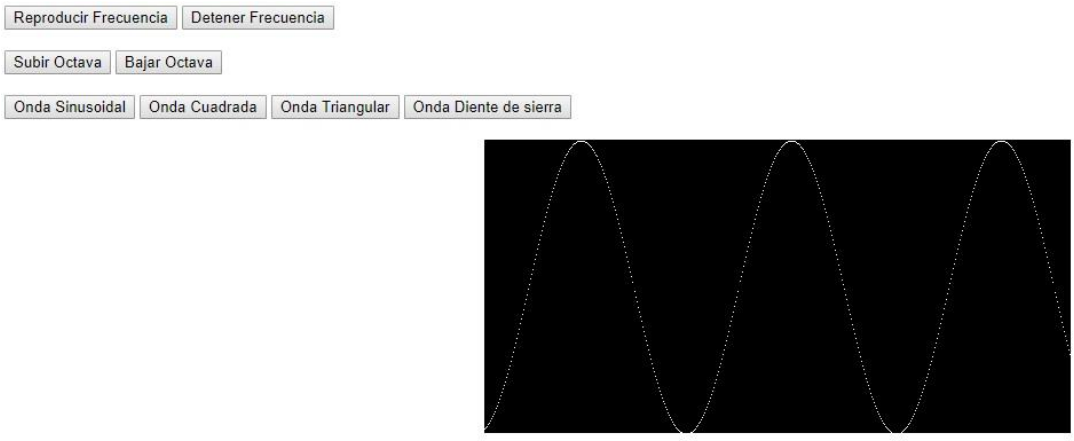


Imagen 5: Aplicación web de un oscilador con nodo analizador.

Para conseguir este aspecto, definimos en el documento HTML los diferentes botones y el canvas.

```
<DOCTYPE html>
<head>
<meta charset="utf-8">
<meta http-equiv="x-ua-compatible" content="ie=edge">
<title>Primeros ejemplos - Oscilador</title>
<meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<div>
<button id='btnReproducir' onclick="reproducirFrecuencia()">Reproducir
Frecuencia</button>
<button id='btnDetener' onclick="detenerFrecuencia()">Detener
Frecuencia</button>
<br>
<br>
<button id='btnSubirOctava' onclick="subirOctava()">Subir
Octava</button>
<button id='btnBajarOctava' onclick="bajarOctava()">Bajar
Octava</button>
<br>
<br>
<button id='sino' onclick="sine()">Onda Sinusoidal</button>
<button id='square' onclick="square()">Onda Cuadrada</button>
<button id='triangle' onclick="triangle()">Onda Triangular</button>
<button id='sawtooth' onclick="sawtooth()">Onda Diente de
sierra</button>
<br>
<br>
<canvas id='canvas' width="512" height="256" ></canvas>
</div>
```

Definimos la función JavaScript visualizar() para conseguir la representación de la forma de la onda.

```
function visualizar() {
//Ajustes del canvas
    var canvas = document.querySelector('#canvas');
    var ctd = canvas.getContext('2d');
    canvas.width = 512;
    canvas.height = 256;
    var muestras = new
Uint8Array(misNodos.analizador.frequencyBinCount);
    misNodos.analizador.getByteTimeDomainData(muestras);
//Dibujamos la forma de la onda
    for (var i = 0; i < muestras.length; i++) {
        var valor = muestras[i];
        var porcentaje = valor / 256;
        var alto = canvas.height * porcentaje;
        var offset = canvas.height - alto;
        var trazo = canvas.width / muestras.length;
        ctd.fillStyle = "#fff";
        ctd.font = "14px Helvetica";
        ctd.fillRect(i * trazo, offset, 1, 1);
        ctd.moveTo(0, 128);
        ctd.lineTo(ancho, 128);
        ctd.stroke();
    }
    requestAnimationFrame(visualizar.bind(this));
};
```

Para dotar de funcionalidad a los botones previamente definidos en HTML utilizamos las siguientes funciones.

```
//Si multiplicamos por 2 la frecuencia subimos 1 octava y si dividimos
por 2 bajamos 1 octava.
function subirOctava(){
  //subimos una octava
  misNodos.osc.frequency.value = misNodos.osc.frequency.value*2;
}

function bajarOctava(){
  //Bajamos una octava
  misNodos.osc.frequency.value = misNodos.osc.frequency.value/2;
}

function detenerFrecuencia(){
  // Detiene la reproduccion de la frecuencia
                misNodos.osc.stop(0);
                audioPlaying = false;
}

//Por defecto: oscilador reproduce onda sinusoidal.
function triangle(){
     //Onda triangular
     misNodos.osc.type = 'triangle';
}
```

```
function square(){
      //Onda cuadrada
      misNodos.osc.type = 'square';
}

function sine(){
      //Onda sinusoidal
      misNodos.osc.type = 'sine';
}

function sawtooth(){
      //Onda Diente de sierra
      misNodos.osc.type = 'sawtooth';
}
```

4.6 Balance de canales

Para el desarrollo de juegos la percepción espacial del sonido es un elemento muy importante. Gracias al nodo *AudioPannerNode* podemos trabajar la espacialidad del sonido parametrizando la posición de la fuente y conseguir un ambiente sonoro en 2 dimensiones o incluso en 3 dimensiones.

Necesitaremos elegir entre balances equipotenciales o algoritmos complejos de alta calidad, obtener la posición y orientación de las fuentes y el oyente y un vector de velocidad de la fuente que controlará tanto la dirección como la velocidad en un entorno 3D y determinará la cantidad de efecto *Doppler* a aplicar.

Cada contexto de audio expone un oyente que representa la orientación y posición del objeto que está escuchando el audio, puede ser cualquier objeto que tenga sentido para el espectador desde esa perspectiva.

El oyente tiene una función *setPosition()* que sitúa al oyente en algún lugar dentro del espacio 3D, por lo tanto, acepta tres coordenadas (x,y,z) y una función *setOrientation()* que rota al oyente y que acepta dos vectores unitarios, uno para la rotación del oyente y el otro vector para la dirección hacia arriba del oyente [19].

4.7 Filtros

Para crear un filtro utilizamos la interfaz *BiquadFilterNode*, que contiene 8 tipos de filtros de audio, activos y pasivos, que podemos utilizar para construir ecualizadores gráficos y otros procesadores sobre de señal de audio.

Como cualquier otro nodo, tenemos que añadirlo a nuestro grafo de encaminamiento y conectarlo adecuadamente con el resto de los nodos. Por ejemplo, para aplicar un filtro de paso bajo, necesitamos indicarlo en su propiedad *type*.

```
misNodos.filtro = context.createBiquadFilter();
misNodos.filtro.type = "lowpass";
misNodos.filtro.frequency.value = 1000;
```

5. Diseño y desarrollo de la solución propuesta

En esta parte del proyecto se mostrarán las etapas llevadas a cabo para conseguir las aplicaciones finales del juego Simon y de la aplicación de la flauta dulce, mostraremos las partes más importantes del código fuente para el desarrollo de las diferentes versiones, encontrando en el Anexo el código completo de todas las versiones y también disponibles en Internet accediendo a sus enlaces correspondientes que iremos indicando en cada caso.

5.1 Juego Simon

Para el desarrollo de esta aplicación web, diseñamos dos etapas claramente diferenciadas, por un lado, el diseño de la interfaz, y por el otro, el desarrollo de la funcionalidad y su implementación. La fusión de las dos etapas dará como resultado la aplicación final.

5.1.1 Diseño de la interfaz

Necesitamos emular el famoso juego Simon de la compañía *Milton Bradley* [8], comúnmente conocida como *MB*, como mostramos en la Imagen 6. Este juego exitoso de la década de los 80 y creado por *Ralph Baer* y *Howard J. Morrison* tiene originalmente cuatro cuadrantes de colores diferentes, formando una forma de disco. El juego irá iluminando de forma aleatoria los diferentes colores y al mismo tiempo que ilumina un color emite un sonido particular. El usuario debe ir seleccionando en el orden correcto la secuencia generada, apoyándose de su memoria sonora y visual.

[8] https://products.hasbro.com/es-es

Imagen 6: Juego Simon de la compañía *MB*.

Más de dos décadas después la compañía ha sacado diferentes modelos mejorando la electrónica del dispositivo, añadiendo más efectos sonoros y visuales [20].

Para conseguir esta interfaz, nos hemos basado en la versión creada por Valentina Morato[9], añadiendo tres interfaces diferentes acorde al nivel de dificultad elegido por el usuario, con 4 colores en el modo más fácil, 6 colores para el modo de dificultad medio y 8 colores para el modo difícil, véase Imagen 7.

Imagen 7: Página principal de selección de la dificultad.

La página principal de la aplicación está compuesta por 3 columnas gracias a la librería de *open source Bootstrap*[10] *3*, añadiendo su hoja de estilos CSS a nuestro directorio.

[9] https://github.com/vmoratog/simon-dice
[10] https://getbootstrap.com/docs/3.3/

Para el diseño de los 3 modos de dificultad, simplemente insertamos los contenedores que hagan falta para representar a los cuadrantes dividiendo la propiedad CSS de la anchura (*width*) por el número de cuadrantes que queramos.

Para demostrar la iluminación de los cuadrantes, cambiamos el color de fondo por un tono más claro de ese color, simulando el efecto visual de iluminación como se puede apreciar en la Figura 2.

Figura 2: Cambio de color de un cuadrante a iluminado.

Separamos la interfaz en dos contenedores , el primero muestra el nivel de dificultad elegido y la puntuación que has de obtener para terminar el modo, además de tu puntuación y tu máxima puntuación. El segundo contenedor muestra el disco, para conseguir esta apariencia utilizamos la propiedad CSS *border-radius*: 50%; este contenedor con forma circular está compuesto por pequeños contenedores, uno para cada color, véase Imagen 8.

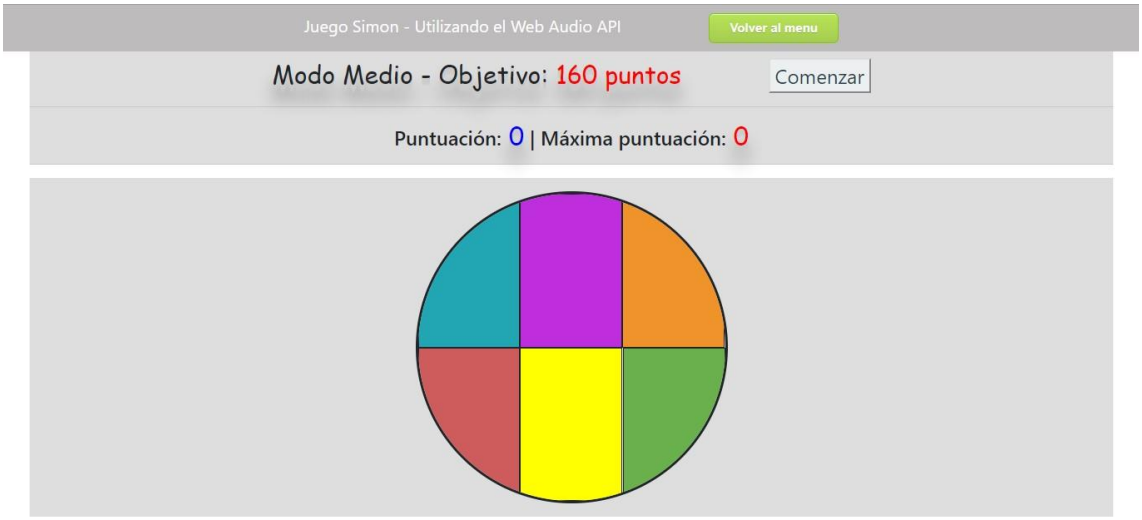


Imagen 8: Página de modo de dificultad medio.

Siendo los otros dos modos de dificultad: fácil y difícil, con una apariencia análoga al ejemplo que acabamos de ver.

5.1.2 Desarrollo e implementación

En este apartado analizaremos el algoritmo utilizado para el correcto funcionamiento del juego. Para servir de referencia utilizaremos como ejemplo el modo de dificultad fácil.

Para el sonido utilizaremos el oscilador como hemos visto en el capítulo 4, cada color representa una frecuencia distinta en el oscilador, veáse Tabla 2.

Color	Frecuencia en el oscilador
Celeste	261.63 Hz
Violeta	329.63 Hz
Naranja	400 Hz
Verde	493.88 Hz

Tabla 2: Correspondencia entre un color y su frecuencia.

Lo primero que colocamos en nuestro *script* Javascript es la declaración de las variables y constantes que utilizaremos en los métodos y funciones.

```
var misNodos = {};
var ULTIMO_NIVEL = 10;
var puntuacionmax = 0;
var cont = document.getElementsByClassName('espaciojuego')[0];

const celeste = document.getElementById('celeste')
const violeta = document.getElementById('violeta')
const naranja = document.getElementById('naranja')
const verde = document.getElementById('verde')
```

Una función importante es la encarga de generar la secuencia aleatoria que devolverá un número del 0 a 3 que corresponde a cada color.

```
generarSecuencia() {
this.secuencia = new Array(ULTIMO_NIVEL).fill(0).map(n =>
Math.floor(Math.random() * 4))
}
```

Con este número ya podemos identificarlo a su color y llamar a la función iluminarsecuencia() cuya labor será la de iluminarlo y poner en marcha el oscilador durante el periodo de tiempo de 0,4 segundos.

```
iluminarSecuencia() {
  for(let i=0; i<this.nivel;i++) {
        let color = this.transformarNumAColor(this.secuencia[i])
        setTimeout(() =>
        this.iluminarColor(color), 1000 *i)
  }
}
iluminarColor(color) {
        cont.style.pointerEvents = "none";
        this.colores[color].classList.add('light')
        const AudioContext = window.AudioContext ||
window.webkitAudioContext || window.mozAudioContext;
        const context = new AudioContext();
        misNodos.osc = context.createOscillator();
        switch (color) {
        case 'celeste':
        misNodos.osc.frequency.value = 261.63;
        misNodos.osc.connect(context.destination);
        misNodos.osc.start(0);
        break;
        case 'violeta':
        misNodos.osc.frequency.value = 329.63;
        misNodos.osc.connect(context.destination);
        misNodos.osc.start(0);
        break;
        case 'naranja':
        misNodos.osc.frequency.value = 400;
        misNodos.osc.connect(context.destination);
        misNodos.osc.start(0);
        break;
        case 'verde':
        misNodos.osc.frequency.value = 493.88;
        misNodos.osc.connect(context.destination);
        misNodos.osc.start(0);
        break;
        }
        setTimeout(() => this.apagarColor(color), 400)
}
```

Podemos observar en la Figura 3 la estructura de directorios de esta aplicación.

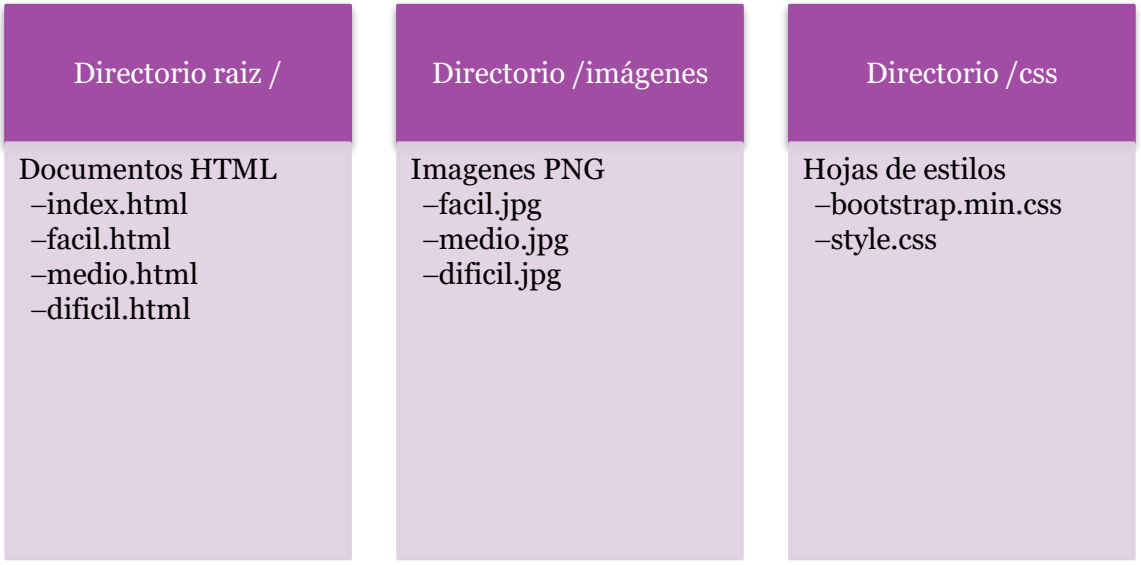


Figura 3: Estructura de directorios del juego Simon.

La dirección URL para acceder a esta aplicación es: http://rcvillena.es/simon/

5.2 Flauta dulce

Para el desarrollo de esta aplicación hemos desarrollado 3 etapas, empezando por una primera etapa muy básica y objeto de ésta hemos obtenido la segunda etapa.

Para terminar, y como resultado de la fusión con las anteriores, una tercera versión más interactiva y emocionante donde el usuario puede participar en una experiencia con el sonido.

5.2.1 Primera etapa

La primera versión de la aplicación se compone de una lista con los diferentes enlaces a los archivos de MP3 correspondientes a las notas naturales DO,RE,MI,FA,SOL,LA,SI y las notas sostenidas DO,RE,MI componiendo las 10 notas principales de la flauta dulce.

Y un lienzo canvas con la imagen pentagrama.png cargada, como observamos en la Imagen 9.

Al hacer clic en cualquier enlace de la lista, se reproduce el archivo correspondiente a su nombre en la lista de sonidos.

Imagen 9: Página de la primera etapa de la aplicación flauta dulce.

Cada enlace de la lista se identifica con la clase = "*nota*" para que cuando el evento *listener* sobre la lista registre un clic, llame a la función *sonarNota*() para reproducirlo.

```
var notas = document.getElementsByClassName('nota');
for (var i = 0, tam = notas.length; i < tam; i++) {
  notas[i].addEventListener('click', function(e) {
    sonarNota(this.href);
    e.preventDefault();
  });
```

La dirección URL para acceder a esta aplicación es: http://rcvillena.es/flauta/primer-desarrollo/

5.2.2 Segunda etapa

En esta segunda versión mantenemos la lista con las diferentes notas y el elemento canvas con la imagen del pentagrama. Como novedad de la etapa anterior, dibujamos mediante círculos en el canvas la posición de la nota que ha sido pulsada y además añadimos un segundo canvas con la imagen flauta.png donde también añadimos círculos para tapar los agujeros correspondientes a la nota pulsada, véase Imagen 10.

Con esto conseguimos el aprendizaje de la colocación de los dedos en el instrumento para sus notas principales.

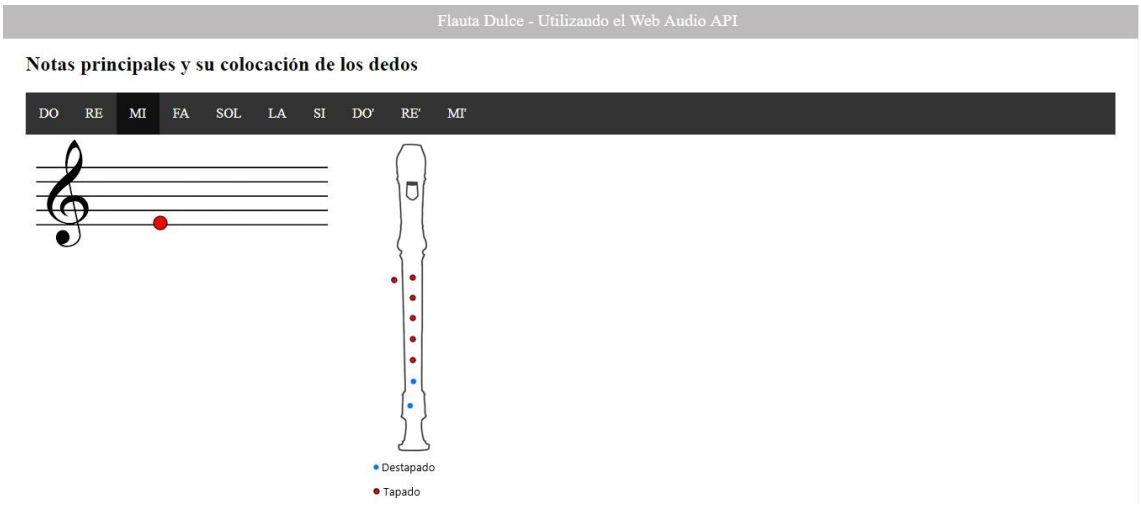


Imagen 10: Página de la segunda etapa de la aplicación flauta dulce.

Para implementar esta funcionalidad hemos creado una función por cada nota que se encarga de pintar los círculos en el canvas. Lo primero que debemos hacer es declarar el elemento canvas y cargar las 2 imágenes pentagrama.png y flauta.png.

```
var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");

var img = new Image();
var img2 = new Image();
img.src = "imagenes/pentagrama.png";
img2.src = "imagenes/flauta.png";

img.onload = function(){
        ctx.drawImage(img, 0, 0);
}
img2.onload = function(){
        ctx.drawImage(img2, 400, 0);
}
```

Ya tenemos el lienzo canvas preparado para colocar los círculos donde le correspondan. A modo de ejemplo, mostramos la función para pintar la nota Re sostenido.

```
function pintarReS(){
        ctx.clearRect(0, 0, canvas.width, canvas.height);
        ctx.drawImage(img, 0, 0);
        ctx.drawImage(img2, 400, 0);
        ctx.beginPath();
        ctx.fillStyle = "red";
        ctx.strokeStyle="black";
        ctx.arc(280, 56, 8, 0, 2 * Math.PI, false);
        ctx.fill();
        ctx.stroke();
        ctx.closePath();
        //FLAUTA-SEGUNDO
        ctx.beginPath();
        ctx.fillStyle = "red";
        ctx.strokeStyle="black";
        ctx.arc(460, 194, 3, 0, 2 * Math.PI, false);
        ctx.fill();
        ctx.stroke();
        ctx.closePath();
}
```

La dirección URL para acceder a esta aplicación es: http://rcvillena.es/flauta/segundo-desarrollo/

5.2.3 Tercera etapa

Tercera versión del desarrollo, más interactiva y dinámica donde gana un mayor peso la apariencia con un mayor uso de la hoja de estilos CSS style.css, dotándole de la importancia que tiene un buen aspecto para una aplicación para la Web.

Se compone de 2 modos diferenciados, editor y reproductor.

El modo editor se basa en un canvas con la imagen pentagrama-grande.png que obtiene las coordenadas x e y al hacer clic dentro de él y dibuja la nota en su ámbito correspondiente, véase Imagen 11.

Imagen 11: Página final en el modo editor de la aplicación flauta dulce.

Debajo del pentagrama compuesto tenemos 5 opciones para interactuar con el usuario: enviar el pentagrama al modo reproductor que veremos a continuación, descargar la composición en fichero del tipo XML, descargar la composición en fichero del tipo PNG, deshacer la última nota colocada y limpiar el pentagrama entero.

La opción de descargar la composición en un fichero de formato XML genera un objeto Blob con los datos de la melodía. Un objeto Blob representa un objeto de tipo fichero de datos planos inmutables [22]. Añadiremos en un array las notas seleccionadas con la notación adecuada para formar un fichero XML como encontramos en la Figura 4. De forma parecía es la funcionalidad de la opción descargar la composición en fichero de tipo PNG que convierte el canvas en un objeto Blob de forma directa con el método *toBlob()*.

```
<?xml version="1.0" encoding="UTF-8" ?>
<notas>
<nota nombre = "re">re</nota>
<nota nombre = "fa">fa</nota>
<nota nombre = "la">la</nota>
<nota nombre = "la">la</nota>
</notas>
```

Figura 4: Ejemplo de composición creada en el editor en formato XML.

El modo reproductor es capaz de reproducir una melodía desde un archivo XML generado por el editor o reproducir una melodía enviada directamente desde el editor.

Cabe destacar el código en lenguaje PHP necesario para la funcionalidad de cargar la composición desde el fichero XML, el cual se envía mediante una llamada Ajax de tipo POST.

```
<?php
$fichero = $_FILES['archivo']['tmp_name'];
$xml = simplexml_load_file($fichero);
if (!$xml) {
        exit;
}
$fichero = "";
foreach ($xml->nota as $nota){
printf("%s,",
        $nota['nombre']);
        }
?>
```

Como opciones disponibles para el usuario nos encontramos: reproducir, pausar o detener el sonido. Además, podemos elegir el tempo musical de reproducción entre 60,90 y 120 BPM (*Beats Per Minute*), véase Imagen 12.

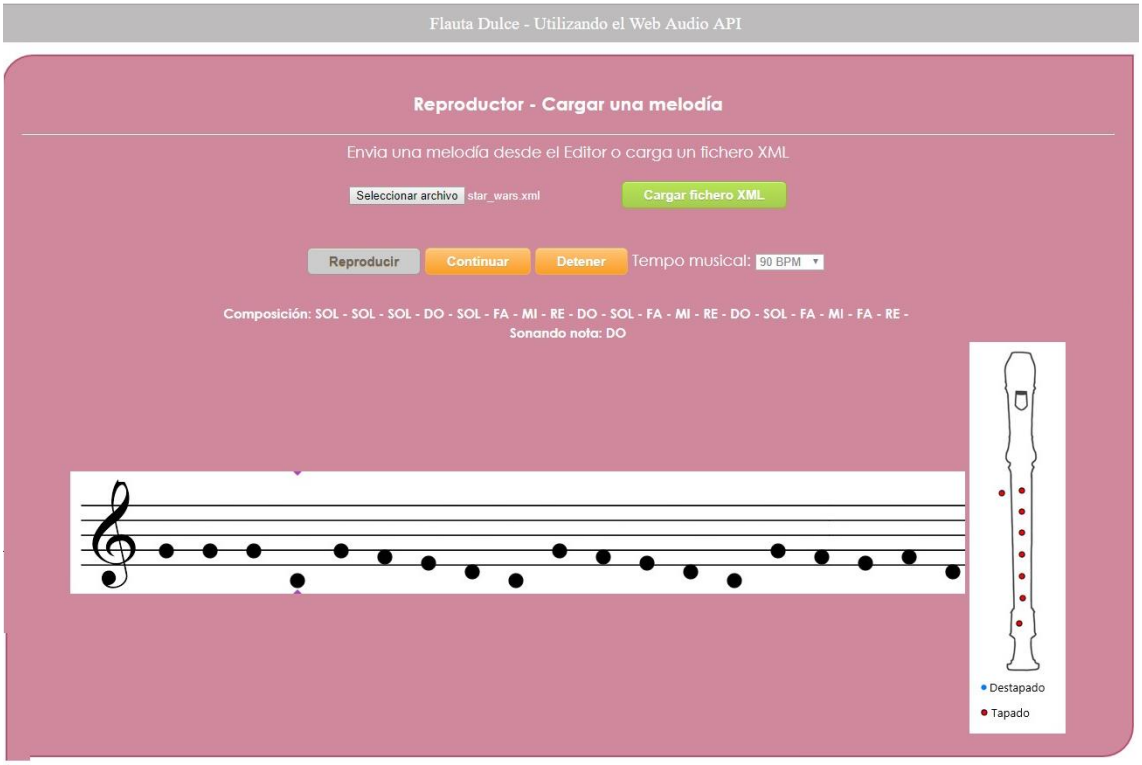


Imagen 12: Página final en el modo reproductor de la aplicación flauta dulce.

La dirección URL para acceder a esta aplicación final es: http://rcvillena.es/flauta/

Podemos observar en la Figura 5 la estructura de directorios de esta aplicación.

Directorio /sonidos	Directorio /imágenes	Directorio /js
Archivos MP3 – do.mp3 – re.mp3 –mi.mp3 –fa.mp3 – sol.mp3 –la.mp3 –si.mp3 –do'.mp3 –re'.mp3 –mi'.mp3	Imagenes PNG – pentagrama.png – flauta.png – pentagrama-grande.png	Archivos JavaScript – sonido.js – reproducir.js – editar.js
Directorio /css	**Archivo PHP**	**Archivo HTML**
Hoja de estilos – style.css	– cargarfichero.php	– index.html

Figura 5: Estructura de directorios de la aplicación flauta dulce.

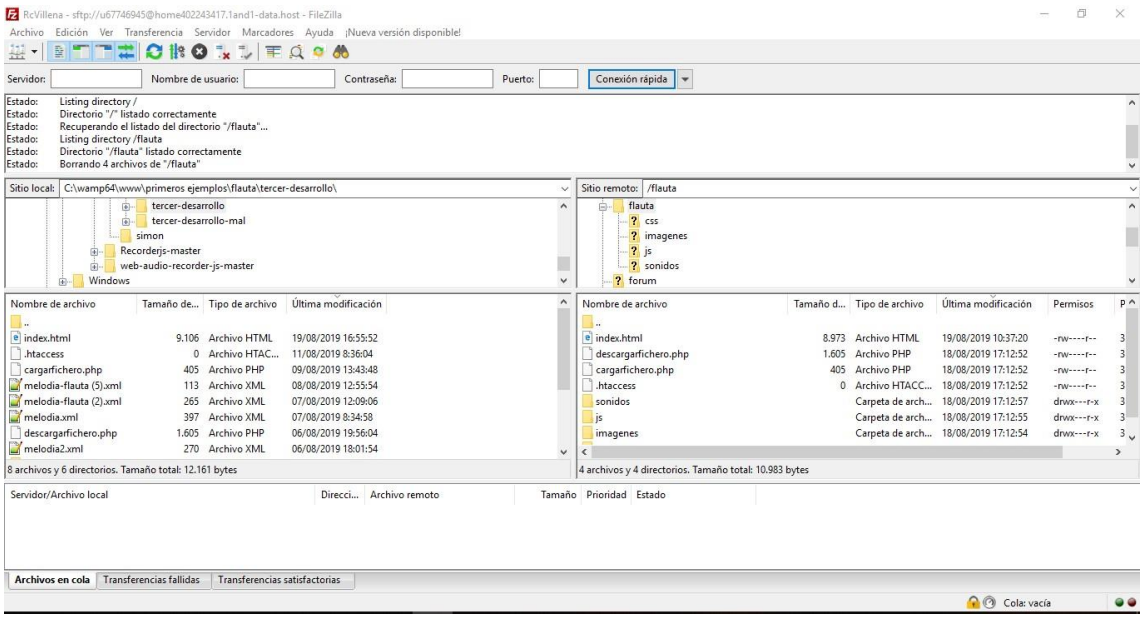
Lo recomendable para el directorio /sonidos donde se encuentran las grabaciones de las diferentes notas de la flauta dulce es contratar las muestras en un banco de sonidos, donde nos permiten obtener una calidad mayor en cualquier proyecto audiovisual que se realice. Debido a la dificultad de obtener unas muestras de este instrumento y debido a la primera fase de desarrollo que se encuentra nuestro proyecto, hemos decidido grabar personalmente los sonidos, obteniendo una posible calidad menor que si lo hubiéramos realizado en un estudio de grabación.

6. Despliegue

Para comprobar el funcionamiento de las aplicaciones necesitamos una pila de software que conste de un servidor web *Apache* con soporte para la programación PHP. En nuestro caso hemos utilizado la distribución para *Microsoft Windows* de *WampServer* 3.1.9 creada por *Romain Bourdon*[11] y que además consta de *OpenSLL* para soporte SSL, base de datos *MySQL*, *MariaDB*, entre otras funciones.

Para la programación de las aplicaciones se ha utilizado el editor de textos y código fuente libre *Notepad++* desarrollado por *Don Ho.*

Para el despliegue en una plataforma web real que facilite el acceso y las pruebas de los desarrollos en diferentes dispositivos, hemos contratado un *hosting* con la empresa *1&1 IONOS,* una de las empresas más importantes en el mundo de los almacenamientos, posicionamientos y dominios web, que nos garantizaba las mismas competencias que nuestro software local. A través de SFTP, gracias al programa *FileZilla* hemos subido los ficheros anteriormente enumerados de nuestras aplicaciones en el punto 5 de esta memoria.



Para su comprobación sólo es necesario acceder desde un navegador web a la dirección que nos habilita la empresa de hosting.

[11] http://www.wampserver.com/en/

6.1 Pruebas de funcionamiento

En este apartado vamos a mostrar el soporte de los diferentes navegadores con el *Web Audio API*.

Para navegadores de PC en Windows

Navegador	**Compatibilidad** con *Web Audio API*	
Internet Explorer	✗	Versión 11 17 de octubre de 2013
Edge	✓	Versión 18 13 de noviembre de 2018
Chrome	✓	Versión 76 30 de julio de 2019
Firefox	✓	Versión 68 9 de julio de 2019
Safari	✓	Versión 12.1 25 de marzo de 2019

Para navegadores de dispositivos móviles o tabletas

Navegador	Compatibilidad con *Web Audio API*	
Safari iOS	✓	Versión 12.3 25 de marzo de 2019
Opera Mini	✗	Ninguna versión 16 de marzo de 2015
Android Browser	✓	Versión 67 7 de mayo de 2017
Blackberry Browser	✗	Versión 10 30 de junio de 2013
Chrome for Android	✓	Versión 75 19 de junio de 2019
Samsung Internet	✓	Versión 9.2 2 de abril de 2019

7. Conclusiones

Los objetivos planteados en el punto 1.1. de la Introducción, han sido cumplidos satisfactoriamente. En el primer objetivo se ha investigado el funcionamiento de la API quedándonos con algunos de los aspectos más relevantes e interesantes para un trabajo de aproximación e introducción en el audio de una plataforma web, completándose de esa forma sin ningún tipo de contratiempo. En el segundo objetivo se ha desarrollado 2 aplicaciones diferentes utilizando técnicas distintas para ampliar el rango de métodos y funciones que es capaz de soportar la plataforma. Para estos desarrollos se han aplicado conocimientos de lenguaje de marcas HTML para el desarrollo de las webs, lenguaje PHP para la lectura de un fichero XML donde se componían las melodías de la flauta dulce, conocimientos en hojas de estilos CSS, y sobre todo, un fuerte trabajo en el lenguaje de programación JavaScript, donde la API de Web Audio trabaja.

Por lo tanto, han servido de ayuda los conocimientos aprendidos en el grado como las asignaturas de Desarrollo Web (3 curso), para la elaboración del documento HTML, o las asignaturas de Tecnología de sistemas de información en la red (3 curso) e Integración de aplicaciones (4 curso) para el desarrollo de la programación en JavaScript y XML. Sin embargo, estos conocimientos adquiridos en las asignaturas se han quedado muy cortos para llegar hasta el nivel que ha requerido el trabajo, teniendo que ampliar fuertemente el conocimiento acerca del funcionamiento de un lenguaje de programación asíncrono y muy difícil como es JavaScript, además del trabajo con la etiqueta <canvas> de HTML para la visualización de los pentagramas que apenas se comenta en la asignatura de Desarrollo Web.

Cabe mencionar, que ninguna asignatura del grado trabaja con el sonido, ni en una plataforma web ni una aplicación de escritorio, requiriendo de esta forma conocimiento de ondas, frecuencias, buffers..., ya que trabajar con un fichero de sonido en una plataforma en la red requiere de más problemas que un fichero de texto, obligando a la sincronización con el tiempo real para sus métodos de gestión, como reproducción o pausa del sonido.

La elaboración de este trabajo ha sido de gran ayuda para ampliar y afianzar mis conocimientos aprendidos en la carrera, pero sin lugar a duda, para aprender a encontrar solución a los problemas que nos podemos encontrar cuando nos enfundamos en un proyecto mayor.

Además, ha despertado mi interés en el mundo del audio, que es muy importante para el desarrollo de cualquier aplicación web o de escritorio y que a veces puede pasar desapercibido para la gran mayoría de usuarios.

7.1 Trabajos futuros

Las ideas que se van a exponer a continuación pueden formar parte de unas posibles ampliaciones a las dos aplicaciones web desarrolladas, pero por su dificultad y por elevar demasiado la complejidad para un trabajo de fin de grado, han quedado descartadas para su implementación en estos nuestros desarrollos.

Para la aplicación web sobre el juego Simon, una posible ampliación sería la implantación de un modo multijugador que con n jugadores sea capaz de generar n secuencias alternas, una secuencia distinta para cada jugador. Estamos hablando de un *script* JavaScript en la línea del desarrollado en el resto de la aplicación que genere para cada jugador una variable independiente con su secuencia y su puntuación correspondiente.

Para la aplicación web sobre la flauta dulce, como futuras ampliaciones cabe señalar la cuestión de terminar la implementación para que el diseño sea "*responsive*", es decir, adaptativo a cualquier tamaño de pantalla. El principal problema para conseguir esto ha sido la dificultad del elemento HTML canvas para adaptarse a las pantallas ya que tenemos desarrolladas funciones JavaScript que detectan la posición en las coordenadas x e y de la pulsación del ratón para la colocación de las notas en el pentagrama. Una pantalla con dimensiones diferentes cambia las coordenadas dificultado el funcionamiento de nuestra función. Otro trabajo futuro es implementar la posibilidad de guardar la reproducción en un fichero de formato MP3 o WAV, añadir notas blancas, corcheas y silencios con una noción de compás en el modo editor y crear más líneas de pentagrama para secuencias más largas.

8. Referencias

[1] SMUS, BORIS (2013). *Web Audio API*. United States of America: Editorial O'Reilly. ISBN: 978-1-449-33268-6.

[2] W3C Candidate Recommendation. *Web Audio API* (18 de septiembre de 2018). [En línea] [fecha de consulta: 2 de abril de 2019]. Disponible en: https://www.w3.org/TR/webaudio/

[3] MDN web docs. *Web Audio API*. [En línea] [fecha de consulta: 10 de marzo de 2019]. Disponible en: https://developer.mozilla.org/es/docs/Web_Audio_API

[4] Estegrafico. *El audio en la web*. [En línea] [fecha de consulta: 11 de abril de 2019]. Disponible en: http://www.estegrafico.com/web-audio-api/

[5] MDN web docs. *Using the Web Audio API*. [En línea] [fecha de consulta: 10 de marzo de 2019]. Disponible en: https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API/Using_Web_Audio_API

[6] OpenAL. *Cross Platform 3D Audio*. [En línea] [fecha de consulta: 20 de marzo de 2019]. Disponible en: https://es.wikipedia.org/wiki/OpenAL

[7] FMOD. [En línea] [fecha de consulta: 20 de marzo de 2019]. Disponible en: https://www.fmod.com/studio

[8] ProTools. [En línea] [fecha de consulta: 21 de marzo de 2019]. Disponible en: https://es.wikipedia.org/wiki/Pro_Tools

[9] Cubase. [En línea] [fecha de consulta: 21 de marzo de 2019]. Disponible en: https://new.steinberg.net/es/cubase/

[10] Toptal. [En línea] [fecha de consulta: 11 de abril de 2019]. Disponible en: https://www.toptal.com/web/web-audio-api-tutorial

[11] Can I Use. [En línea] [fecha de consulta: 23 de agosto de 2019]. Disponible en: https://caniuse.com/#search=web%20audio%20api

[12] Doboism. *Additional browser compatibility tests for specific features*. [En línea] [fecha de consulta: 23 de agosto de 2019]. Disponible en: http://www.doboism.com/projects/webaudio-compatibility

[13] WebPlatform Docs. *Web Audio API*. [En línea] [fecha de consulta: 23 de agosto de 2019]. Disponible en: https://webplatform.github.io/docs/apis/webaudio/

[14] Microsoft. *IE/Edge Platform Status*. [En línea] [fecha de consulta: 23 de agosto de 2019]. Disponible en: https://developer.microsoft.com/en-us/microsoft-edge/platform/status/webaudioapi/

[15] Chrome Platform Status. *Unprefixed Web Audio API*. [En línea] [fecha de consulta: 23 de agosto de 2019]. Disponible en: https://www.chromestatus.com/feature/6261718720184320

[16] Mozilla. *Firefox Platform Status*. [En línea] [fecha de consulta: 23 de agosto de 2019]. Disponible en: https://webkit.org/status/#feature-web-audio

[17] WebKit. *WebKit Platform Status*. [En línea] [fecha de consulta: 23 de agosto de 2019]. Disponible en: https://webkit.org/status/#feature-web-audio

[18] Guachat, J.D (2018). *HTML5 para mentes maestras*. Capítulo 25. Toronto: Editorial Mink Books. ISBN: 978-1545169490.

[19] Tutsplus. *Web Audio and 3D Soundscapes: Introduction*. [En línea] [fecha de consulta: 24 de agosto de 2019]. Disponible en: https://webdesign.tutsplus.com/tutorials/web-audio-and-3d-soundscapes-introduction--cms-22650

[20] Wikipedia. *Simon (juego)*. [En línea] [fecha de consulta: 24 de agosto de 2019]. Disponible en: https://es.wikipedia.org/wiki/Simon_(juego)

[21] MDN web docs. *Conceptos y uso de audio web*. [En línea] [fecha de consulta: 10 de marzo de 2019]. Disponible en: https://developer.mozilla.org/es/docs/Web_Audio_API#Conceptos_y_uso_de_audio_Web

[22] MDN web docs. *Blob*. [En línea] [fecha de consulta: 24 de agosto de 2019]. Disponible en: https://developer.mozilla.org/es/docs/Web/API/Blob

9. Anexos

Juego Simon – Modo difícil – **dificil.html**

```
<!DOCTYPE html>
<html>
<head>
          <meta charset="utf-8">
          <meta http-equiv="x-ua-compatible" content="ie=edge">
          <title>Juego - Simon | Modo dificil</title>
          <meta name="description" content="A way to make sure files have loaded
before playing them">
          <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">
          <link rel="stylesheet" type="text/css" href="css/style.css">
          <script
src="https://cdnjs.cloudflare.com/ajax/libs/sweetalert/2.1.2/sweetalert.min.js"><
/script>
          <script src="https://code.jquery.com/jquery-1.10.2.js"></script>
          <link href="css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
<div id="header">Juego Simon -  Utilizando el Web Audio API
<button onclick="location.href='index.html'" class="myButtonverde volver">Volver
al menu</button>
</div>
<div class="informacion">

<h5 class="parpadeo" >Modo Dificil - Objetivo: </h5><h5 class="parpadeo blanco">
210 puntos</h5>
<button id="btnEmpezar" class="btn-start volver"
onclick="empezarJuego()">Comenzar</button>

<hr>

<h5>Puntuación:<label id ="puntuacion" class="parpadeo azul"></label> | Máxima
puntuación:<label id ="puntuacionmaxima" class="parpadeo rojo2"></label></h5>



<hr>
</div>
<div id="espaciojuego" class="espaciojuego">
<div class="gameboard">
          <div id="celeste" class="colordificil celeste left" data-
color="celeste"></div>
          <div id="violeta" class="colordificil violeta left" data-
color="violeta"></div>
          <div id="naranja" class="colordificil naranja left" data-
color="naranja"></div>
          <div id="gris" class="colordificil gris left" data-color="gris"></div>
          <div id="verde" class="colordificil verde right" data-
color="verde"></div>
          <div id="amarillo" class="colordificil amarillo center" data-
```

```
color="amarillo"></div>
          <div id="rojo" class="colordificil rojo left" data-color="rojo"></div>
          <div id="hueso" class="colordificil hueso left" data-
color="hueso"></div>
</div>
</div>
<script>
          var misNodos = {};
          const celeste = document.getElementById('celeste')
          const violeta = document.getElementById('violeta')
          const naranja = document.getElementById('naranja')
          const verde = document.getElementById('verde')
          const btnEmpezar = document.getElementById('btnEmpezar')
          const ULTIMO_NIVEL = 20
          var cont = document.getElementsByClassName('espaciojuego')[0];
          var puntuacionmax = 0;
          document.getElementById('puntuacion').innerHTML = ' 0';
          document.getElementById('puntuacionmaxima').innerHTML = ' 0';

          class Juego {
          constructor() {
            this.siguienteNivel = this.siguienteNivel.bind(this)
            this.elegirColor = this.elegirColor.bind(this)
            this.inicializar = this.inicializar.bind(this)

            this.inicializar()
            this.generarSecuencia()
            setTimeout(this.siguienteNivel(), 500)
          }

          inicializar() {
            this.toggleBtnEmpezar()
            this.nivel = 1
            this.colores = {
                    celeste,
                    violeta,
                    naranja,
                    verde,
                    amarillo,
                    rojo,
                    gris,
                    hueso
            }
          }

          toggleBtnEmpezar() {
            if(btnEmpezar.classList.contains('hide')) {
                    btnEmpezar.classList.remove('hide')
            } else {
                    btnEmpezar.classList.add('hide')
            }
          }

          generarSecuencia() {
            this.secuencia = new Array(ULTIMO_NIVEL).fill(0).map(n =>
Math.floor(Math.random() * 8))
```

```
107             }
108
109             siguienteNivel() {
110               this.subnivel = 0
111               this.iluminarSecuencia()
112               this.agregarEventoClick()
113             }
114
115             transformarNumAColor(numero) {
116               switch(numero) {
117                       case 0:
118                         return 'celeste'
119                       case 1:
120                         return 'violeta'
121                       case 2:
122                         return 'naranja'
123                       case 3:
124                         return 'verde'
125                       case 4:
126                         return 'amarillo'
127                       case 5:
128                         return 'rojo'
129                       case 6:
130                         return 'gris'
131                       case 7:
132                         return 'hueso'
133               }
134             }
135
136             transformarColorANum(color) {
137               switch(color) {
138                       case 'celeste':
139                         return 0
140                       case 'violeta':
141                         return 1
142                       case 'naranja':
143                         return 2
144                       case 'verde':
145                         return 3
146                       case 'amarillo':
147                         return 4
148                       case 'rojo':
149                         return 5
150                       case 'gris':
151                         return 6
152                       case 'hueso':
153                         return 7
154               }
155             }
156
157             iluminarSecuencia() {
158                       cont.style.pointerEvents = "none";
159               for(let i=0; i<this.nivel;i++) {
160                       let color = this.transformarNumAColor(this.secuencia[i])
161                       setTimeout(() =>
162                       this.iluminarColor(color), 500 *i)
```


```
163             }
164
165           }
166
167           iluminarColor(color) {
168
169   //NO PERMITIR HACER CLICK
170                     cont.style.pointerEvents = "none";
171                     this.colores[color].classList.add('light')
172                     const AudioContext = window.AudioContext ||
173   window.webkitAudioContext || window.mozAudioContext;
174                     const context = new AudioContext();
175                     misNodos.osc = context.createOscillator();
176                     switch (color) {
177                     case 'celeste':
178                     misNodos.osc.frequency.value = 261.63;//Valor inicial de
179   261,63 Hz (nota Do)
180                     misNodos.osc.connect(context.destination);
181                     misNodos.osc.start(0);
182                     break;
183                     case 'violeta':
184                     misNodos.osc.frequency.value = 293.66;//Valor inicial de
185   261,63 Hz (nota Do)
186                     misNodos.osc.connect(context.destination);
187                     misNodos.osc.start(0);
188                     break;
189                     case 'naranja':
190                     misNodos.osc.frequency.value = 329.63;//Valor inicial de
191   261,63 Hz (nota Do)
192                     misNodos.osc.connect(context.destination);
193                     misNodos.osc.start(0);
194                     break;
195                     case 'verde':
196                     misNodos.osc.frequency.value = 349.23;//Valor inicial de
197   261,63 Hz (nota Do)
198                     misNodos.osc.connect(context.destination);
199                     misNodos.osc.start(0);
200                     break;
201                     case 'amarillo':
202                     misNodos.osc.frequency.value = 392.00;//Valor inicial de
203   261,63 Hz (nota Do)
204                     misNodos.osc.connect(context.destination);
205                     misNodos.osc.start(0);
206                     break;
207                     case 'rojo':
208                     misNodos.osc.frequency.value = 440.00;//Valor inicial de
209   261,63 Hz (nota Do)
210                     misNodos.osc.connect(context.destination);
211                     misNodos.osc.start(0);
212                     break;
213                     case 'gris':
214                     misNodos.osc.frequency.value = 493.00;//Valor inicial de
215   261,63 Hz (nota Do)
216                     misNodos.osc.connect(context.destination);
217                     misNodos.osc.start(0);
218                     break;
```

```
                    case 'hueso':
                    misNodos.osc.frequency.value = 550.00;//Valor inicial de
261,63 Hz (nota Do)
                    misNodos.osc.connect(context.destination);
                    misNodos.osc.start(0);
                    break;
                    }
                    setTimeout(() => this.apagarColor(color), 200)
          }

          apagarColor(color) {
            this.colores[color].classList.remove('light')
            misNodos.osc.stop(0);
                    console.log(this.nivel);
          setTimeout(() =>  cont.style.pointerEvents = "auto", this.nivel*20)


          }

          agregarEventoClick() {
            this.colores.celeste.addEventListener('click', this.elegirColor)
            this.colores.verde.addEventListener('click', this.elegirColor)
            this.colores.violeta.addEventListener('click', this.elegirColor)
            this.colores.naranja.addEventListener('click', this.elegirColor)
            this.colores.amarillo.addEventListener('click', this.elegirColor)
            this.colores.rojo.addEventListener('click', this.elegirColor)
            this.colores.gris.addEventListener('click', this.elegirColor)
            this.colores.hueso.addEventListener('click', this.elegirColor)

          }

          eliminarEventosClick() {
            this.colores.celeste.removeEventListener('click', this.elegirColor)
            this.colores.verde.removeEventListener('click', this.elegirColor)
            this.colores.violeta.removeEventListener('click', this.elegirColor)
            this.colores.naranja.removeEventListener('click', this.elegirColor)
            this.colores.amarillo.removeEventListener('click', this.elegirColor)
            this.colores.rojo.removeEventListener('click', this.elegirColor)
            this.colores.gris.removeEventListener('click', this.elegirColor)
            this.colores.hueso.removeEventListener('click', this.elegirColor)

          }

          elegirColor(ev) {
           const nombreColor = ev.target.dataset.color
           const numeroColor = this.transformarColorANum(nombreColor)
           this.iluminarColor(nombreColor)
           if(numeroColor === this.secuencia[this.subnivel]) {
             this.subnivel++
             if(this.subnivel === this.nivel) {
                     this.nivel++
                     this.eliminarEventosClick()
                     if(puntuacionmax < this.nivel*10){
                     puntuacionmax = this.nivel*10;
                     }
                              document.getElementById('puntuacion').innerHTML = '
```

```
'+this.nivel*10;
                                document.getElementById('puntuacionmaxima').innerHTML
= ' '+ puntuacionmax;


                        if(this.nivel === (ULTIMO_NIVEL+1)) {
                                misNodos.osc.stop(0);
                                document.getElementById('puntuacion').innerHTML = '
';
                                this.ganoElJuego()

                        } else {
                          setTimeout(this.siguienteNivel,1000)
                        }
                }
              } else {
                        misNodos.osc.stop(0);
                        document.getElementById('puntuacion').innerHTML = ' ';
               this.perdioElJuego()
              }
            }

            ganoElJuego() {
              swal('','Felicidades! ¡Has ganado el juego!', 'success')
              .then( this.inicializar)
            }

            perdioElJuego() {
            document.getElementById('puntuacion').innerHTML = ' 0';
              swal('','¡Has perdido! Vuelve a intentarlo', 'error')
              .then(() => {
                        misNodos.osc.stop(0)

                        this.eliminarEventosClick()
                        this.inicializar()
              })
            }
            }

            function empezarJuego() {
            var juego = new Juego()
            }
</script>
</body>
</html>
```

Flauta dulce – Desarrollo final – **index.html**

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Aplicacion - Flauta - Tercer desarrollo</title>
<link href="css/style.css" rel="stylesheet" type="text/css" media="all" />
<script src="https://code.jquery.com/jquery-1.10.2.js"></script>
<script src="js/sonido.js"></script>
<script src="js/reproducir.js"></script>
<script src="js/editar.js"></script>
<script src="js/WavAudioEncoder.js"></script>
</head>


<body>
<div id="page">
          <div id="header">Flauta Dulce -  Utilizando el Web Audio API</div>
    <div id="content">
                    <ul hidden>
                      <li><a id="do" href="sonidos/do.mp3" class="track">DO</a></li>
                      <li><a id="re" href="sonidos/re.mp3" class="track">RE</a></li>
                      <li><a id="mi" href="sonidos/mi.mp3" class="track">MI</a></li>
                      <li><a id="fa" href="sonidos/fa.mp3" class="track">FA</a></li>
                      <li><a id="sol" href="sonidos/sol.mp3"
class="track">SOL</a></li>
                      <li><a id="la" href="sonidos/la.mp3" class="track">LA</a></li>
                      <li><a id="si" href="sonidos/si.mp3" class="track">SI</a></li>
                      <li><a id="do'" href="sonidos/do'.mp3"
class="track">DO'</a></li>
                      <li><a id="re'" href="sonidos/re'.mp3"
class="track">RE'</a></li>
                      <li><a id="mi'" href="sonidos/mi'.mp3"
class="track">MI'</a></li>
                    </ul><br><br>


                    <div class="caja-azul">
                              <h3>Editor - Componer una melod&iacutea </h3><hr/>
                              Haz click en el pentagrama para colocar una nota
<br><hr/>
                              <h5>Última nota colocada: <label id
="ultimanota"></label><h5>

                              <div id="container" style="position:relative;">
                              <canvas id="canvas-componer" ></canvas>
                              <div id="output" style="position:absolute;
padding:5px;"></div>
                              </div>

                              <h5>Composición: <label id ="labelmelodia"></label><h5>
                              <button type="button" class="myButton"
id="enviarreproductor" onclick="enviarreproductor()">Enviar al reproductor</button>
                              <button type="button" class="myButton"
```

```
onclick="descargar()">Descargar en fichero XML</button>
                            <button type="button" class="myButton"
id="jpeg">Descargar en fichero PNG</button>
                            <button type="button" class="myButtonn"
onclick="deshacer()">Deshacer</button>
                            <button type="button" class="myButtonn"
onclick="borrar()">Limpiar pentagrama</button>
                    </div>

                    <div class ="caja-roja" id ="reproductor">
                            <h3>Reproductor - Cargar una melod&iacutea </h3><hr/>
                            Envia una melod&iacutea desde el Editor o carga un
fichero XML<br><br>
                            <input type="file" id ="files" accept=".xml"
name="fichero" required>
                            <button type="button" id='envio'
class='myButtonverde'>Cargar fichero XML</button>
                            <div id="progress_bar"><div
class="percent">0%</div></div>

                            <button type="button" class="myButton"
id="btnreproducir" onclick="reproducirMelodia()">Reproducir</button>
                            <input type="button" class="myButton" id="btnpausar"
value="Pausar" onclick="pausar()">
                            <button type="button" class="myButton" id="btndetener"
onclick="detener()">Detener</button>
                            Tempo musical:
 <select id="tempo">
  <option value="" selected disabled>- BPM</option>
  <option value="sesenta">60 BPM</option>
  <option value="noventa">90 BPM</option>
  <option value="cientoveinte">120 BPM</option>
</select>
<br>
<br>
                            <!-- <button type="button" class="myButton"
onclick="guardar()">Guardar en fichero MP3</button><hr/><br><br> -->
                            <label id="sonandomelodia" class="h55"></label><br>
                            <label id="sonandonota" class="h55"></label><br>
                            <canvas id='canvas-reproducir' width="1024" height="300"
></canvas>
                            <canvas id='canvas' width="110" height="448" ></canvas>
                    </div>

          </div>
</div>

<script type="text/javascript">
console.clear();
var buffers = {};
var bufferLen = 4096;
var numChannels = 2;
var mimeType = 'audio/wav';
var context = new(window.AudioContext || window.webkitAudioContext)();
var canvas = document.getElementById("canvas");
var canvas2 = document.getElementById("canvas-componer");
```

```
112 var canvas3 = document.getElementById("canvas-reproducir");
113 var ctx = canvas.getContext("2d");
114 var ctx2 = canvas2.getContext("2d");
115 var ctx3 = canvas3.getContext("2d");
116 var img2 = new Image();
117 var img3 = new Image();
118 var espacio = 60;
119 var espacio2 = 60;
120 var arrayDeNotas = [];
121 var arraycomponer = [];
122 var arraycomponer2 = [];
123 var active_source = null;
124 var tracks = document.getElementsByClassName('track');
125 var pointSize = 8;
126
127 img2.src = "imagenes/flauta.png";
128 img3.src = "imagenes/pentagrama-grande.png";
129
130
131
132 var jpeg = document.getElementById("jpeg");
133 var link = document.createElement('a');
134
135
136 jpeg.addEventListener("click",function(){
137
138  canvas2.toBlob(function(blob){
139     link.href = URL.createObjectURL(blob);
140
141           descargarArchivo(blob, 'composicion.png');
142
143   },'image/png');
144
145
146           function descargarArchivo(contenidoEnBlob, nombreArchivo) {
147              var reader = new FileReader();
148                    reader.onload = function (event) {
149                              var save = document.createElement('a');
150                              save.href = event.target.result;
151                              save.target = '_blank';
152                              save.download = nombreArchivo || 'archivo.dat';
153                              var clicEvent = new MouseEvent('click', {
154                                        'view': window,
155                                                  'bubbles': true,
156                                                  'cancelable': true
157                              });
158                              save.dispatchEvent(clicEvent);
159                              (window.URL ||
160 window.webkitURL).revokeObjectURL(save.href);
161                    };
162                    reader.readAsDataURL(contenidoEnBlob);
163           }
164
165 },false);
166
167 if (ctx2) {
```

```
168
169  var output = document.getElementById("output");
170
171                       canvas2.addEventListener("mousemove", function(evt) {
172                         var mousePos = oMousePos(canvas2, evt);
173                         marcarCoords(output, mousePos.x, mousePos.y)
174                       }, false);
175
176                       canvas2.addEventListener("mouseout", function(evt) {
177                         limpiarCoords(output);
178                       }, false);
179
180
181
182          function marcarCoords(output, x, y) {
183                    if (y > 40 && y < 135 && x > 100 && x < 1000) {
184                              if( y > 40 && y < 50) {//MI'
185                                        output.innerHTML = ("Nota: " + "MI'");
186                              }
187                              if( y > 50 && y < 60) {//RE'
188                                        output.innerHTML = ("Nota: " + "RE'");
189                              }
190                              if( y > 60 && y < 68) {//DO'
191                                        output.innerHTML = ("Nota: " + "DO'");
192                              }
193                              if( y > 68 && y < 78) {//SI
194                                        output.innerHTML = ("Nota: " + "SI");
195                              }
196                              if( y > 78 && y < 86) {//LA
197                                        output.innerHTML = ("Nota: " + "LA");
198                              }
199                              if( y > 86 && y < 95) {//SOL
200                                        output.innerHTML = ("Nota: " + "SOL");
201                              }
202                              if( y > 95 && y < 101) {//FA
203                                        output.innerHTML = ("Nota: " + "FA");
204                              }
205                              if( y > 101 && y < 111) {//MI
206                                        output.innerHTML = ("Nota: " + "MI");
207                              }
208                              if( y > 111 && y < 121) {//RE
209                                        output.innerHTML = ("Nota: " + "RE");
210                              }
211                              if( y > 121 && y < 135) {//DO
212                                        output.innerHTML = ("Nota: " + "DO");
213                              }
214                              output.style.top = (y + 10) + "px";
215                              output.style.left = (x + 10) + "px";
216                              output.style.color = "black";
217                              output.style.backgroundColor = "#FFF";
218                              output.style.border = "1px solid #d9d9d9"
219
220                     }
221                     else {
222                              output.innerHTML = ("Nota: " + "-");
223                              output.style.top = (y + 10) + "px";
```

```
                                output.style.left = (x + 10) + "px";
                                output.style.color = "black";
                                output.style.backgroundColor = "#FFF";
                                output.style.border = "1px solid #d9d9d9"
                        }
                }

                        function limpiarCoords(output) {
                          output.innerHTML = "";
                          output.style.top = 0 + "px";
                          output.style.left = 0 + "px";
                          output.style.backgroundColor = "transparent"
                          output.style.border = "none";
                          canvas.style.cursor = "default";
                        }

                        function oMousePos(canvas2, evt) {
                          var ClientRect = canvas2.getBoundingClientRect();
                          return { //objeto
                            x: Math.round(evt.clientX - ClientRect.left),
                            y: Math.round(evt.clientY - ClientRect.top)
                          }
                          }
}


var tempoS = 0;
var select = document.getElementById('tempo');
select.addEventListener('change',
  function(){
    var tempoSelect = this.options[select.selectedIndex];

          if (tempoSelect.value == "sesenta") {
                  tempoS = 1000;
          }
          if (tempoSelect.value == "noventa") {
                  tempoS = 666,67;
          }
          if (tempoSelect.value == "cientoveinte") {
                  tempoS = 500;
          }

  });


img2.onload = function(){

          ctx.drawImage(img2, 0, 0);

          ctx3.drawImage(img3, 0, 0);
}

img3.onload = function(){
canvas2.width = img3.naturalWidth;
canvas2.height = img3.naturalHeight;
```

```
280           ctx2.drawImage(img3, 0, 0);
281 }
282 $("#enviarreproductor").click(function() {
283           $('html, body').animate({
284           scrollTop: $("#reproductor").offset().top
285           }, 500);
286 });
287
288 $("#canvas-componer").click(function(e){
289      getPosition(e);
290 });
291
292 $("#canvas-componer").on('mouseenter', function() {
293         canvas2.style.cursor = 'pointer';
294       });
295
296 $('#envio').on('click', function (e){
297           borrarreproductor();
298           e.preventDefault(); // Evitamos que salte el enlace.
299           var paqueteDeDatos = new FormData();
300           paqueteDeDatos.append('archivo', $('#files')[0].files[0]);
301           if($('#files')[0].files[0] == null) {
302                     alert("Tienes que seleccionar un fichero XML");
303                     return false;
304           }
305           else {
306                     $.ajax({
307                               data: paqueteDeDatos,
308                               url:   'cargarfichero.php', //archivo que recibe la
309 peticion
310                               cache: false,
311                               contentType: false,
312                               processData: false,
313                               type:  'post', //método de envio
314                               success:  function (response) { //una vez que el archivo
315 recibe el request lo procesa y lo devuelve
316                                         var x = 2;
317                                         cargarpentagrama(response,x);
318
319           document.getElementById('sonandomelodia').innerHTML = 'Composición: '+
320 arraycomponer2;
321
322           document.getElementById('sonandonota').innerHTML = ' ';
323                                         detenido = false;
324                                                   console.log("Deteniendo...");
325                                                   ctx3.lineWidth = 0;
326                                                   ctx3.strokeStyle = "white";
327                                                   ctx3.fillStyle = "white";
328                                                   ctx3.beginPath();
329                                                   ctx3.fillRect(espaciovertical-5, 0,
330 12, 12);
331                                                   ctx3.closePath();
332                                                   ctx3.fill();
333
334
335                                                   ctx3.strokeStyle = "white";
```

```
                                                ctx3.fillStyle = "white";
                                                ctx3.beginPath();
                                                ctx3.fillRect(espaciovertical-5, 135,
12, 5);
                                                ctx3.closePath();
                                                ctx3.fill();

          document.getElementById('sonandonota').innerHTML = ' ';

          document.getElementById("btnpausar").value = "Pausar";

                                                ctx.clearRect(0, 0, canvas.width,
canvas.height);
                                                ctx.drawImage(img2, 0, 0);
                                                clearInterval(i);
                                                espaciovertical = 60;


          document.getElementById("btnreproducir").disabled=false;

          document.getElementById("btnpausar").disabled=true;

          document.getElementById("btndetener").disabled=true;

          document.getElementById("tempo").disabled=false;
                              }//cierro success
        });
          }
});

var reader;
  var progress = document.querySelector('.percent');

  function abortRead() {
    reader.abort();
  }

  function errorHandler(evt) {
    switch(evt.target.error.code) {
      case evt.target.error.NOT_FOUND_ERR:
        alert('File Not Found!');
        break;
      case evt.target.error.NOT_READABLE_ERR:
        alert('File is not readable');
        break;
      case evt.target.error.ABORT_ERR:
        break; // noop
      default:
        alert('An error occurred reading this file.');
    };
  }

  function updateProgress(evt) {
    // evt is an ProgressEvent.
    if (evt.lengthComputable) {
      var percentLoaded = Math.round((evt.loaded / evt.total) * 100);
```

```
392       // Increase the progress bar length.
393       if (percentLoaded < 100) {
394         progress.style.width = percentLoaded + '%';
395         progress.textContent = percentLoaded + '%';
396       }
397     }
398   }
399
400   function handleFileSelect(evt) {
401     // Reset progress indicator on new file selection.
402     progress.style.width = '0%';
403     progress.textContent = '0%';
404
405     reader = new FileReader();
406     reader.onerror = errorHandler;
407     reader.onprogress = updateProgress;
408     reader.onabort = function(e) {
409       alert('File read cancelled');
410     };
411     reader.onloadstart = function(e) {
412       document.getElementById('progress_bar').className = 'loading';
413     };
414     reader.onload = function(e) {
415       // Ensure that the progress bar displays 100% at the end.
416       progress.style.width = '100%';
417       progress.textContent = '100%';
418       setTimeout("document.getElementById('progress_bar').className='';", 2000);
419     }
420
421     // Read in the image file as a binary string.
422     reader.readAsBinaryString(evt.target.files[0]);
423   }
424
425   document.getElementById('files').addEventListener('change', handleFileSelect,
426 false);
427
428 //SONIDO
429 for (var i = 0, len = tracks.length; i < len; i++) {
430   tracks[i].addEventListener('click', function(e) {
431     playTrack(this.href);
432     e.preventDefault();
433   });
434   getBuffer(tracks[i].href);
435 }
436
437           document.getElementById("btnpausar").disabled=true;
438           document.getElementById("btndetener").disabled=true;
439
440 </script>
441 </body>
442 </html>
```

Flauta dulce – Script Javascript sonido – **sonido.js**

```
function stopActiveSource() {
  if (active_source) {
    active_source.onended = null; // manual stop, no event
    active_source.stop(0);

  }
}


function playTrack(url) {
  // get fom our dictionnary
  var buffer = buffers[url];
  // stop the active one if any
  stopActiveSource();
  // create a new BufferSource
  var source = context.createBufferSource();
  // it is now the active one
  active_source = source;



  source.onended = function() {
    active_source = null;
  };

  source.buffer = buffer;
  source.connect(context.destination);
  source.start(0);
  console.log(buffers);
}

function getBuffer(url) {
  var request = new XMLHttpRequest();
  request.open('GET', url, true);
  request.responseType = 'arraybuffer';
  request.onload = function(evt) {
    context.decodeAudioData(request.response, store);
  };
  request.send();
  function store(buffer) {
    buffers[url] = buffer;
  }
}
```

Flauta dulce – Script JavaScript reproductor – **reproducir.js**

```
function borrarreproductor(){
          var ctx = document.getElementById("canvas-reproducir").getContext("2d");
          ctx.clearRect(0, 0, canvas.width, canvas.height);
          ctx.drawImage(img3, 0, 0);
          document.getElementById('sonandomelodia').innerHTML = '';
          document.getElementById('sonandonota').innerHTML = '';
          arraycomponer2 = "";
          espacio = 60;
}

function cargarpentagrama(response,x) {
          ctx3.clearRect(0, 0, canvas.width, canvas.height);
          ctx3.drawImage(img3, 0, 0);
          espacio2 = 60;
          if( x == 1) {
                    arrayDeNotas = response.split(" - ");
                    }
          if( x == 2) {
                    arrayDeNotas = response.split(",");
                    }
          for (var i=0; i < arrayDeNotas.length-1; i++) {
                    var nota = arrayDeNotas[i];
                    switch (nota) {
                    case 'do':
                              espacio2 += 50;
                              ctx3.beginPath();
                              ctx3.fillStyle = "black";
                              ctx3.strokeStyle="black";
                              ctx3.arc(espacio2, 125, 8, 0, 2 * Math.PI, false);
                              ctx3.fill();
                              ctx3.stroke();
                              ctx3.closePath();
                              arraycomponer2 += "DO - ";
                              break;
                    case 're':
                              espacio2 += 50;
                              ctx3.beginPath();
                              ctx3.fillStyle = "black";
                              ctx3.strokeStyle="black";
                              ctx3.arc(espacio2, 115, 8, 0, 2 * Math.PI, false);
                              ctx3.fill();
                              ctx3.stroke();
                              ctx3.closePath();
                              arraycomponer2 += "RE - ";
                              break;
                    case 'mi':
                              espacio2+= 50;
                              ctx3.beginPath();
                              ctx3.fillStyle = "black";
                              ctx3.strokeStyle="black";
                              ctx3.arc(espacio2, 105, 8, 0, 2 * Math.PI, false);
                              ctx3.fill();
                              ctx3.stroke();
                              ctx3.closePath();
                              arraycomponer2 += "MI - ";
```

```
                    break;
                case 'fa':
                    espacio2+= 50;
                    ctx3.beginPath();
                    ctx3.fillStyle = "black";
                    ctx3.strokeStyle="black";
                    ctx3.arc(espacio2, 98, 8, 0, 2 * Math.PI, false);
                    ctx3.fill();
                    ctx3.stroke();
                    ctx3.closePath();
                    arraycomponer2 += "FA - ";
                    break;
                case 'sol':
                    espacio2+= 50;
                    ctx3.beginPath();
                    ctx3.fillStyle = "black";
                    ctx3.strokeStyle="black";
                    ctx3.arc(espacio2, 91, 8, 0, 2 * Math.PI, false);
                    ctx3.fill();
                    ctx3.stroke();
                    ctx3.closePath();
                    arraycomponer2 += "SOL - ";
                    break;
                case 'la':
                    espacio2+= 50;
                    ctx3.beginPath();
                    ctx3.fillStyle = "black";
                    ctx3.strokeStyle="black";
                    ctx3.arc(espacio2, 82, 8, 0, 2 * Math.PI, false);
                    ctx3.fill();
                    ctx3.stroke();
                    ctx3.closePath();
                    arraycomponer2 += "LA - ";
                    break;
                case 'si':
                    espacio2+= 50;
                    ctx3.beginPath();
                    ctx3.fillStyle = "black";
                    ctx3.strokeStyle="black";
                    ctx3.arc(espacio2, 74, 8, 0, 2 * Math.PI, false);
                    ctx3.fill();
                    ctx3.stroke();
                    ctx3.closePath();
                    arraycomponer2 += "SI - ";
                    break;
                case "do'":
                    espacio2+= 50;
                    ctx3.beginPath();
                    ctx3.fillStyle = "black";
                    ctx3.strokeStyle="black";
                    ctx3.arc(espacio2, 65, 8, 0, 2 * Math.PI, false);
                    ctx3.fill();
                    ctx3.stroke();
                    ctx3.closePath();
                    arraycomponer2 += "DO' - ";
                    break;
```

```
112                     case "re'":
113                             espacio2+= 50;
114                             ctx3.beginPath();
115                             ctx3.fillStyle = "black";
116                             ctx3.strokeStyle="black";
117                             ctx3.arc(espacio2, 56, 8, 0, 2 * Math.PI, false);
118                             ctx3.fill();
119                             ctx3.stroke();
120                             ctx3.closePath();
121                             arraycomponer2 += "RE' - ";
122                             break;
123                     case "mi'":
124                             espacio2+= 50;
125                             ctx3.beginPath();
126                             ctx3.fillStyle = "black";
127                             ctx3.strokeStyle="black";
128                             ctx3.arc(espacio2, 47, 8, 0, 2 * Math.PI, false);
129                             ctx3.fill();
130                             ctx3.stroke();
131                             ctx3.closePath();
132                             arraycomponer2 += "MI' - ";
133                             break;
134                     }//cierro switch
135             }//cierro bucle for
136 }
137
138
139 function guardar() {
140
141 }
142
143
144
145 var detenido = false;
146
147
148 function pausar() {
149             if (detenido) {
150                     detenido = false;
151                     document.getElementById("btnpausar").value = "Pausar";
152                     var valorj = j;
153                     reproducirMelodia(valorj);
154
155             }
156             if(!detenido){
157                     detenido = true;
158                     document.getElementById("btnpausar").value = "Continuar";
159             }
160 }
161
162 function detener(){
163                     detenido = false;
164                     console.log("Deteniendo...");
165                     ctx3.lineWidth = 0;
166                     ctx3.strokeStyle = "white";
167                     ctx3.fillStyle = "white";
```

```
168                     ctx3.beginPath();
169                     ctx3.fillRect(espaciovertical-5, 0, 12, 12);
170                     ctx3.closePath();
171                     ctx3.fill();
172
173
174                     ctx3.strokeStyle = "white";
175                     ctx3.fillStyle = "white";
176                     ctx3.beginPath();
177                     ctx3.fillRect(espaciovertical-5, 135, 12, 5);
178                     ctx3.closePath();
179                     ctx3.fill();
180                     document.getElementById('sonandonota').innerHTML = ' ';
181                     document.getElementById("btnpausar").value = "Pausar";
182
183                     ctx.clearRect(0, 0, canvas.width, canvas.height);
184                     ctx.drawImage(img2, 0, 0);
185         clearInterval(i);
186                     espaciovertical = 60;
187
188           document.getElementById("btnreproducir").disabled=false;
189           document.getElementById("btnpausar").disabled=true;
190           document.getElementById("btndetener").disabled=true;
191           document.getElementById("tempo").disabled=false;
192
193
194 }
195
196
197           var espaciovertical = 60;
198           var botondetener = false;
199 function reproducirMelodia(valorj){
200
201
202           if(tempoS == 0) {
203                     alert("Selecciona un tempo musical.");
204           }
205           else {
206
207           document.getElementById("btnreproducir").disabled=true;
208           document.getElementById("btnpausar").disabled=false;
209           document.getElementById("btndetener").disabled=false;
210           document.getElementById("tempo").disabled=true;
211           var counter = 0;
212           var j = 0;
213           var espacio3 = 60;
214
215
216
217
218           //BORRAR y DIBUJAR LIMPIO
219           ctx.clearRect(0, 0, canvas.width, canvas.height);
220           ctx.drawImage(img2, 0, 0);
221
222           i = setInterval(function(){
223                     if(botondetener == true){
```

```
224                     espaciovertical = 60;
225                     clearInterval(i);
226                     ctx3.lineWidth = 0;
227                     ctx3.strokeStyle = "white";
228                     ctx3.fillStyle = "white";
229                     ctx3.beginPath();
230                     ctx3.fillRect(espaciovertical-5, 0, 12, 12);
231                     ctx3.closePath();
232                     ctx3.fill();
233
234
235                     ctx3.strokeStyle = "white";
236                     ctx3.fillStyle = "white";
237                     ctx3.beginPath();
238                     ctx3.fillRect(espaciovertical-5, 135, 12, 5);
239                     ctx3.closePath();
240                     ctx3.fill();
241                     document.getElementById('sonandonota').innerHTML = ' ';
242
243                     ctx.clearRect(0, 0, canvas.width, canvas.height);
244                     ctx.drawImage(img2, 0, 0);
245                     console.log("hola");
246
247                     botondetener = false;
248                     detenido = false;
249                     espaciovertical = 60;
250
251             }
252
253                     // do your thing
254                     if(!detenido && !botondetener){
255                     document.getElementById('sonandonota').innerHTML = 'Sonando nota:
256 '+ arrayDeNotas[j].toUpperCase();
257                     document.getElementById(arrayDeNotas[j]).click();
258                     espaciovertical+= 50;
259
260
261                     ctx3.lineWidth = 0;
262                     // Color de línea
263                     ctx3.strokeStyle = "white";
264                     // Color de relleno
265                     ctx3.fillStyle = "#AB47BC";
266                     ctx3.beginPath();
267                     // Nos movemos a la esquina superior izquierda
268                     ctx3.moveTo(espaciovertical, 5);
269                     // Dibujamos una línea hacia abajo
270                     ctx3.lineTo(espaciovertical-5, 0);
271                     // Desde el fin de esa línea,
272                     // dibujamos una hacia la derecha
273                     ctx3.lineTo(espaciovertical+5, 0);
274                     ctx3.closePath();
275                     // Y dejamos que JS cierre nuestro dibujo
276
277                     ctx3.fill();
278
279                     ctx3.beginPath();
```

```
280                     // Nos movemos a la esquina superior izquierda
281                     ctx3.moveTo(espaciovertical, 135);
282                     // Dibujamos una línea hacia abajo
283                     ctx3.lineTo(espaciovertical-5, 140);
284                     // Desde el fin de esa línea,
285                     // dibujamos una hacia la derecha
286                     ctx3.lineTo(espaciovertical+5, 140);
287                     ctx3.closePath();
288                     // Y dejamos que JS cierre nuestro dibujo
289
290                     ctx3.fill();
291
292                     if( espaciovertical > 110) {
293                     ctx3.lineWidth = 0;
294                     ctx3.strokeStyle = "white";
295                     ctx3.fillStyle = "white";
296                     ctx3.beginPath();
297                     ctx3.fillRect(espaciovertical-56, 0, 12, 12);
298                     ctx3.closePath();
299                     ctx3.fill();
300
301
302                     ctx3.strokeStyle = "white";
303                     ctx3.fillStyle = "white";
304                     ctx3.beginPath();
305                     ctx3.fillRect(espaciovertical-56, 135, 12, 5);
306                     ctx3.closePath();
307                     ctx3.fill();
308                     }
309
310
311
312                     var nota = arrayDeNotas[j];
313                     switch (nota) {
314                     case 'do':
315                     //BORRAR y DIBUJAR LIMPIO
316                     ctx.clearRect(0, 0, canvas.width, canvas.height);
317                     ctx.drawImage(img2, 0, 0);
318                     //FLAUTA-AGUJERO DETRAS
319                     ctx.beginPath();
320                     ctx.fillStyle = "red";
321                     ctx.strokeStyle="black";
322                     ctx.arc(37, 173, 3, 0, 2 * Math.PI, false);
323                     ctx.fill();
324                     ctx.stroke();
325                     ctx.closePath();
326                     //FLAUTA-PRIMERO
327                     ctx.beginPath();
328                     ctx.fillStyle = "red";
329                     ctx.strokeStyle="black";
330                     ctx.arc(60, 170, 3, 0, 2 * Math.PI, false);
331                     ctx.fill();
332                     ctx.stroke();
333                     ctx.closePath();
334                     //FLAUTA-SEGUNDO
335                     ctx.beginPath();
```

```
336                     ctx.fillStyle = "red";
337                     ctx.strokeStyle="black";
338                     ctx.arc(60, 194, 3, 0, 2 * Math.PI, false);
339                     ctx.fill();
340                     ctx.stroke();
341                     ctx.closePath();
342                     //FLAUTA-TERCERO
343                     ctx.beginPath();
344                     ctx.fillStyle = "red";
345                     ctx.strokeStyle="black";
346                     ctx.arc(60, 218, 3, 0, 2 * Math.PI, false);
347                     ctx.fill();
348                     ctx.stroke();
349                     ctx.closePath();
350                     //FLAUTA-CUARTO
351                     ctx.beginPath();
352                     ctx.fillStyle = "red";
353                     ctx.strokeStyle="black";
354                     ctx.arc(60, 243, 3, 0, 2 * Math.PI, false);
355                     ctx.fill();
356                     ctx.stroke();
357                     ctx.closePath();
358                     //FLAUTA-QUINTO
359                     ctx.beginPath();
360                     ctx.fillStyle = "red";
361                     ctx.strokeStyle="black";
362                     ctx.arc(60, 268, 3, 0, 2 * Math.PI, false);
363                     ctx.fill();
364                     ctx.stroke();
365                     ctx.closePath();
366                     //FLAUTA-SEXTO
367                     ctx.beginPath();
368                     ctx.fillStyle = "red";
369                     ctx.strokeStyle="black";
370                     ctx.arc(61, 293, 3, 0, 2 * Math.PI, false);
371                     ctx.fill();
372                     ctx.stroke();
373                     ctx.closePath();
374                     //FLAUTA-SEPTIMO
375                     ctx.beginPath();
376                     ctx.fillStyle = "red";
377                     ctx.strokeStyle="black";
378                     ctx.arc(57, 322, 3, 0, 2 * Math.PI, false);
379                     ctx.fill();
380                     ctx.stroke();
381                     ctx.closePath();
382                     break;
383
384                     case 're':
385                     //BORRAR y DIBUJAR LIMPIO
386                     ctx.clearRect(0, 0, canvas.width, canvas.height);
387                     ctx.drawImage(img2, 0, 0);
388                     //FLAUTA-AGUJERO DETRAS
389                     ctx.beginPath();
390                     ctx.fillStyle = "red";
391                     ctx.strokeStyle="black";
```

```
392                     ctx.arc(38, 173, 3, 0, 2 * Math.PI, false);
393                     ctx.fill();
394                     ctx.stroke();
395                     ctx.closePath();
396                     //FLAUTA-PRIMERO
397                     ctx.beginPath();
398                     ctx.fillStyle = "red";
399                     ctx.strokeStyle="black";
400                     ctx.arc(60, 170, 3, 0, 2 * Math.PI, false);
401                     ctx.fill();
402                     ctx.stroke();
403                     ctx.closePath();
404                     //FLAUTA-SEGUNDO
405                     ctx.beginPath();
406                     ctx.fillStyle = "red";
407                     ctx.strokeStyle="black";
408                     ctx.arc(60, 194, 3, 0, 2 * Math.PI, false);
409                     ctx.fill();
410                     ctx.stroke();
411                     ctx.closePath();
412                     //FLAUTA-TERCERO
413                     ctx.beginPath();
414                     ctx.fillStyle = "red";
415                     ctx.strokeStyle="black";
416                     ctx.arc(60, 218, 3, 0, 2 * Math.PI, false);
417                     ctx.fill();
418                     ctx.stroke();
419                     ctx.closePath();
420                     //FLAUTA-CUARTO
421                     ctx.beginPath();
422                     ctx.fillStyle = "red";
423                     ctx.strokeStyle="black";
424                     ctx.arc(60, 243, 3, 0, 2 * Math.PI, false);
425                     ctx.fill();
426                     ctx.stroke();
427                     ctx.closePath();
428                     //FLAUTA-QUINTO
429                     ctx.beginPath();
430                     ctx.fillStyle = "red";
431                     ctx.strokeStyle="black";
432                     ctx.arc(60, 268, 3, 0, 2 * Math.PI, false);
433                     ctx.fill();
434                     ctx.stroke();
435                     ctx.closePath();
436                     //FLAUTA-SEXTO
437                     ctx.beginPath();
438                     ctx.fillStyle = "red";
439                     ctx.strokeStyle="black";
440                     ctx.arc(61, 293, 3, 0, 2 * Math.PI, false);
441                     ctx.fill();
442                     ctx.stroke();
443                     ctx.closePath();
444                     break;
445
446                     case 'mi':
447                     //BORRAR y DIBUJAR LIMPIO
```

```
448                     ctx.clearRect(0, 0, canvas.width, canvas.height);
449                     ctx.drawImage(img2, 0, 0);
450                     //FLAUTA-AGUJERO DETRAS
451                     ctx.beginPath();
452                     ctx.fillStyle = "red";
453                     ctx.strokeStyle="black";
454                     ctx.arc(38, 173, 3, 0, 2 * Math.PI, false);
455                     ctx.fill();
456                     ctx.stroke();
457                     ctx.closePath();
458                     //FLAUTA-PRIMERO
459                     ctx.beginPath();
460                     ctx.fillStyle = "red";
461                     ctx.strokeStyle="black";
462                     ctx.arc(60, 170, 3, 0, 2 * Math.PI, false);
463                     ctx.fill();
464                     ctx.stroke();
465                     ctx.closePath();
466                     //FLAUTA-SEGUNDO
467                     ctx.beginPath();
468                     ctx.fillStyle = "red";
469                     ctx.strokeStyle="black";
470                     ctx.arc(60, 194, 3, 0, 2 * Math.PI, false);
471                     ctx.fill();
472                     ctx.stroke();
473                     ctx.closePath();
474                     //FLAUTA-TERCERO
475                     ctx.beginPath();
476                     ctx.fillStyle = "red";
477                     ctx.strokeStyle="black";
478                     ctx.arc(60, 218, 3, 0, 2 * Math.PI, false);
479                     ctx.fill();
480                     ctx.stroke();
481                     ctx.closePath();
482                     //FLAUTA-CUARTO
483                     ctx.beginPath();
484                     ctx.fillStyle = "red";
485                     ctx.strokeStyle="black";
486                     ctx.arc(60, 243, 3, 0, 2 * Math.PI, false);
487                     ctx.fill();
488                     ctx.stroke();
489                     ctx.closePath();
490                     //FLAUTA-QUINTO
491                     ctx.beginPath();
492                     ctx.fillStyle = "red";
493                     ctx.strokeStyle="black";
494                     ctx.arc(60, 268, 3, 0, 2 * Math.PI, false);
495                     ctx.fill();
496                     ctx.stroke();
497                     ctx.closePath();
498                     break;
499
500                     case 'fa':
501                     //BORRAR y DIBUJAR LIMPIO
502                     ctx.clearRect(0, 0, canvas.width, canvas.height);
503                     ctx.drawImage(img2, 0, 0);
```

```
504                     //FLAUTA-AGUJERO DETRAS
505                     ctx.beginPath();
506                     ctx.fillStyle = "red";
507                     ctx.strokeStyle="black";
508                     ctx.arc(38, 173, 3, 0, 2 * Math.PI, false);
509                     ctx.fill();
510                     ctx.stroke();
511                     ctx.closePath();
512                     //FLAUTA-PRIMERO
513                     ctx.beginPath();
514                     ctx.fillStyle = "red";
515                     ctx.strokeStyle="black";
516                     ctx.arc(60, 170, 3, 0, 2 * Math.PI, false);
517                     ctx.fill();
518                     ctx.stroke();
519                     ctx.closePath();
520                     //FLAUTA-SEGUNDO
521                     ctx.beginPath();
522                     ctx.fillStyle = "red";
523                     ctx.strokeStyle="black";
524                     ctx.arc(60, 194, 3, 0, 2 * Math.PI, false);
525                     ctx.fill();
526                     ctx.stroke();
527                     ctx.closePath();
528                     //FLAUTA-TERCERO
529                     ctx.beginPath();
530                     ctx.fillStyle = "red";
531                     ctx.strokeStyle="black";
532                     ctx.arc(60, 218, 3, 0, 2 * Math.PI, false);
533                     ctx.fill();
534                     ctx.stroke();
535                     ctx.closePath();
536                     //FLAUTA-CUARTO
537                     ctx.beginPath();
538                     ctx.fillStyle = "red";
539                     ctx.strokeStyle="black";
540                     ctx.arc(60, 243, 3, 0, 2 * Math.PI, false);
541                     ctx.fill();
542                     ctx.stroke();
543                     ctx.closePath();
544                     break;
545
546                     case 'sol':
547                     //BORRAR y DIBUJAR LIMPIO
548                     ctx.clearRect(0, 0, canvas.width, canvas.height);
549                     ctx.drawImage(img2, 0, 0);
550                     //FLAUTA-AGUJERO DETRAS
551                     ctx.beginPath();
552                     ctx.fillStyle = "red";
553                     ctx.strokeStyle="black";
554                     ctx.arc(38, 173, 3, 0, 2 * Math.PI, false);
555                     ctx.fill();
556                     ctx.stroke();
557                     ctx.closePath();
558                     //FLAUTA-PRIMERO
559                     ctx.beginPath();
```

```
560                     ctx.fillStyle = "red";
561                     ctx.strokeStyle="black";
562                     ctx.arc(60, 170, 3, 0, 2 * Math.PI, false);
563                     ctx.fill();
564                     ctx.stroke();
565                     ctx.closePath();
566                     //FLAUTA-SEGUNDO
567                     ctx.beginPath();
568                     ctx.fillStyle = "red";
569                     ctx.strokeStyle="black";
570                     ctx.arc(60, 194, 3, 0, 2 * Math.PI, false);
571                     ctx.fill();
572                     ctx.stroke();
573                     ctx.closePath();
574                     //FLAUTA-TERCERO
575                     ctx.beginPath();
576                     ctx.fillStyle = "red";
577                     ctx.strokeStyle="black";
578                     ctx.arc(60, 218, 3, 0, 2 * Math.PI, false);
579                     ctx.fill();
580                     ctx.stroke();
581                     ctx.closePath();
582                     break;
583
584                     case 'la':
585                     //BORRAR y DIBUJAR LIMPIO
586                     ctx.clearRect(0, 0, canvas.width, canvas.height);
587                     ctx.drawImage(img2, 0, 0);
588                     //FLAUTA-AGUJERO DETRAS
589                     ctx.beginPath();
590                     ctx.fillStyle = "red";
591                     ctx.strokeStyle="black";
592                     ctx.arc(38, 173, 3, 0, 2 * Math.PI, false);
593                     ctx.fill();
594                     ctx.stroke();
595                     ctx.closePath();
596                     //FLAUTA-PRIMERO
597                     ctx.beginPath();
598                     ctx.fillStyle = "red";
599                     ctx.strokeStyle="black";
600                     ctx.arc(60, 170, 3, 0, 2 * Math.PI, false);
601                     ctx.fill();
602                     ctx.stroke();
603                     ctx.closePath();
604                     //FLAUTA-SEGUNDO
605                     ctx.beginPath();
606                     ctx.fillStyle = "red";
607                     ctx.strokeStyle="black";
608                     ctx.arc(60, 194, 3, 0, 2 * Math.PI, false);
609                     ctx.fill();
610                     ctx.stroke();
611                     ctx.closePath();
612                     break;
613
614                     case 'si':
615                     //BORRAR y DIBUJAR LIMPIO
```

```
616                     ctx.clearRect(0, 0, canvas.width, canvas.height);
617                     ctx.drawImage(img2, 0, 0);
618                     //FLAUTA-AGUJERO DETRAS
619                     ctx.beginPath();
620                     ctx.fillStyle = "red";
621                     ctx.strokeStyle="black";
622                     ctx.arc(38, 173, 3, 0, 2 * Math.PI, false);
623                     ctx.fill();
624                     ctx.stroke();
625                     ctx.closePath();
626                     //FLAUTA-PRIMERO
627                     ctx.beginPath();
628                     ctx.fillStyle = "red";
629                     ctx.strokeStyle="black";
630                     ctx.arc(60, 170, 3, 0, 2 * Math.PI, false);
631                     ctx.fill();
632                     ctx.stroke();
633                     ctx.closePath();
634                     break;
635 
636                     case "do'":
637                     //BORRAR y DIBUJAR LIMPIO
638                     ctx.clearRect(0, 0, canvas.width, canvas.height);
639                     ctx.drawImage(img2, 0, 0);
640                     //FLAUTA-AGUJERO DETRAS
641                     ctx.beginPath();
642                     ctx.fillStyle = "red";
643                     ctx.strokeStyle="black";
644                     ctx.arc(38, 173, 3, 0, 2 * Math.PI, false);
645                     ctx.fill();
646                     ctx.stroke();
647                     ctx.closePath();
648                     //FLAUTA-SEGUNDO
649                     ctx.beginPath();
650                     ctx.fillStyle = "red";
651                     ctx.strokeStyle="black";
652                     ctx.arc(60, 194, 3, 0, 2 * Math.PI, false);
653                     ctx.fill();
654                     ctx.stroke();
655                     ctx.closePath();
656                     break;
657 
658                     case "re'":
659                     //BORRAR y DIBUJAR LIMPIO
660                     ctx.clearRect(0, 0, canvas.width, canvas.height);
661                     ctx.drawImage(img2, 0, 0);
662                     //FLAUTA-SEGUNDO
663                     ctx.beginPath();
664                     ctx.fillStyle = "red";
665                     ctx.strokeStyle="black";
666                     ctx.arc(60, 194, 3, 0, 2 * Math.PI, false);
667                     ctx.fill();
668                     ctx.stroke();
669                     ctx.closePath();
670                     break;
671 
```


```
672                     case "mi'":
673                     //BORRAR y DIBUJAR LIMPIO
674                     ctx.clearRect(0, 0, canvas.width, canvas.height);
675                     ctx.drawImage(img2, 0, 0);
676                     //FLAUTA-MEDIO AGUJERO DETRAS
677                     ctx.beginPath();
678                     ctx.fillStyle = "red";
679                     ctx.strokeStyle="black";
680                     ctx.arc(37, 173, 3, 0, 1 * Math.PI, false);
681                     ctx.fill();
682                     ctx.stroke();
683                     ctx.closePath();
684                     //FLAUTA-PRIMERO
685                     ctx.beginPath();
686                     ctx.fillStyle = "red";
687                     ctx.strokeStyle="black";
688                     ctx.arc(60, 170, 3, 0, 2 * Math.PI, false);
689                     ctx.fill();
690                     ctx.stroke();
691                     ctx.closePath();
692                     //FLAUTA-SEGUNDO
693                     ctx.beginPath();
694                     ctx.fillStyle = "red";
695                     ctx.strokeStyle="black";
696                     ctx.arc(60, 194, 3, 0, 2 * Math.PI, false);
697                     ctx.fill();
698                     ctx.stroke();
699                     ctx.closePath();
700                     //FLAUTA-TERCERO
701                     ctx.beginPath();
702                     ctx.fillStyle = "red";
703                     ctx.strokeStyle="black";
704                     ctx.arc(60, 218, 3, 0, 2 * Math.PI, false);
705                     ctx.fill();
706                     ctx.stroke();
707                     ctx.closePath();
708                     //FLAUTA-CUARTO
709                     ctx.beginPath();
710                     ctx.fillStyle = "red";
711                     ctx.strokeStyle="black";
712                     ctx.arc(60, 243, 3, 0, 2 * Math.PI, false);
713                     ctx.fill();
714                     ctx.stroke();
715                     ctx.closePath();
716                     //FLAUTA-QUINTO
717                     ctx.beginPath();
718                     ctx.fillStyle = "red";
719                     ctx.strokeStyle="black";
720                     ctx.arc(60, 268, 3, 0, 2 * Math.PI, false);
721                     ctx.fill();
722                     ctx.stroke();
723                     ctx.closePath();
724                     break;
725             }//cierro switch-case
726                     j++;
727                     counter++;
```

```
728
729     if(j == arrayDeNotas.length-1) {
730
731           var ejecT = setTimeout(fin, 1000);
732
733           function fin(){
734                     console.log("he terminado");
735                     ctx3.lineWidth = 0;
736                     ctx3.strokeStyle = "white";
737                     ctx3.fillStyle = "white";
738                     ctx3.beginPath();
739                     ctx3.fillRect(espaciovertical-5, 0, 12, 12);
740                     ctx3.closePath();
741                     ctx3.fill();
742
743
744                     ctx3.strokeStyle = "white";
745                     ctx3.fillStyle = "white";
746                     ctx3.beginPath();
747                     ctx3.fillRect(espaciovertical-5, 135, 12, 5);
748                     ctx3.closePath();
749                     ctx3.fill();
750                     document.getElementById('sonandonota').innerHTML = ' ';
751
752                     ctx.clearRect(0, 0, canvas.width, canvas.height);
753                     ctx.drawImage(img2, 0, 0);
754         clearInterval(i);
755                     espaciovertical = 60;
756
757           document.getElementById("btnreproducir").disabled=false;
758           document.getElementById("btnpausar").disabled=true;
759           document.getElementById("btndetener").disabled=true;
760           document.getElementById("tempo").disabled=false;
761
762           }
763
764
765
766     }
767                     }
768 }, tempoS);
769
770
771           }//cierro else del tempo musical
772 }
```

Flauta dulce – Script JavaScript editor – **editar.js**

```
function borrar(){
          var ctx = document.getElementById("canvas-componer").getContext("2d");
          ctx.clearRect(0, 0, canvas.width, canvas.height);
          ctx.drawImage(img3, 0, 0);
          document.getElementById('ultimanota').innerHTML = '';
          document.getElementById('labelmelodia').innerHTML = '';
          arraycomponer = "";
          espacio = 60;
          contador = 0;
}

function deshacer(){

          var ctx = document.getElementById("canvas-componer").getContext("2d");

          console.log(arraycomponer);
          var arrayDeNotas = arraycomponer.split(" - ");
                    console.log(arrayDeNotas);
                    borrar();
          for (var i=0; i < arrayDeNotas.length-2; i++) {
                    var nombrenota = arrayDeNotas[i].toLowerCase();
                    switch(nombrenota) {
                              case "mi'":
                              drawCoordinates(500,41);
                              break;
                              case "re'":
                              drawCoordinates(500,51);
                              break;
                              case "do'":
                              drawCoordinates(500,61);
                              break;
                              case "si":
                              drawCoordinates(500,71);
                              break;
                              case "la":
                              drawCoordinates(500,81);
                              break;
                              case "sol":
                              drawCoordinates(500,91);
                              break;
                              case "fa":
                              drawCoordinates(500,100);
                              break;
                              case "mi":
                              drawCoordinates(500,105);
                              break;
                              case "re":
                              drawCoordinates(500,115);
                              break;
                              case "do":
                              drawCoordinates(500,125);
                              break;
                    }
                    }

```

```
56      }
57
58      function getPosition(event){
59          var rect = canvas2.getBoundingClientRect();
60          var x = event.clientX - rect.left;
61          var y = event.clientY - rect.top;
62
63          drawCoordinates(x,y);
64      }
65                var contador = 0;
66
67      function drawCoordinates(x,y){
68                var ctx = document.getElementById("canvas-componer").getContext("2d");
69
70                var pentagramalleno = false;
71                if (contador == 19) {
72                alert("Has completado el pentagrama.");
73                pentagramalleno = true;
74                }
75                if(y > 40 && y < 135 && x > 100 && x < 1000 && !pentagramalleno)/*DENTRO DEL
76      PENTAGRAMA*/ {
77                ctx.fillStyle = "black";
78                ctx.strokeStyle="black";
79          ctx.beginPath();
80                if( y > 40 && y < 50) {//miS
81                espacio += 50;
82          ctx.arc(espacio, 47, pointSize, 0, Math.PI * 2, false);
83                arraycomponer += "MI' - ";
84                document.getElementById('ultimanota').innerHTML = "MI'";
85                document.getElementById('labelmelodia').innerHTML = arraycomponer;
86                contador++;
87                }
88                if( y > 50 && y < 60) {
89                espacio += 50;
90          ctx.arc(espacio, 55, pointSize, 0, Math.PI * 2, false);
91                arraycomponer += "RE' - ";
92                document.getElementById('ultimanota').innerHTML = "RE'";
93                document.getElementById('labelmelodia').innerHTML = arraycomponer;
94                contador++;
95                }
96                if( y > 60 && y < 68) {
97                espacio += 50;
98          ctx.arc(espacio, 65, pointSize, 0, Math.PI * 2, false);
99                arraycomponer += "DO' - ";
100               document.getElementById('ultimanota').innerHTML = "DO'";
101               document.getElementById('labelmelodia').innerHTML = arraycomponer;
102               contador++;
103               }
104               if( y > 68 && y < 78) {
105               espacio += 50;
106         ctx.arc(espacio, 72, pointSize, 0, Math.PI * 2, false);
107               arraycomponer += "SI - ";
108               document.getElementById('ultimanota').innerHTML = 'SI';
109               document.getElementById('labelmelodia').innerHTML = arraycomponer;
110               contador++;
111               }
```

```
112             if( y > 78 && y < 86) {
113             espacio += 50;
114         ctx.arc(espacio, 82, pointSize, 0, Math.PI * 2, false);
115             arraycomponer += "LA - ";
116             document.getElementById('ultimanota').innerHTML = 'LA';
117             document.getElementById('labelmelodia').innerHTML = arraycomponer;
118             contador++;
119             }
120             if( y > 86 && y < 95) {
121             espacio += 50;
122         ctx.arc(espacio, 88, pointSize, 0, Math.PI * 2, false);
123             arraycomponer += "SOL - ";
124             document.getElementById('ultimanota').innerHTML = 'SOL';
125             document.getElementById('labelmelodia').innerHTML = arraycomponer;
126             contador++;
127             }
128             if( y > 95 && y < 101) {
129             espacio += 50;
130         ctx.arc(espacio, 99, pointSize, 0, Math.PI * 2, false);
131             arraycomponer += "FA - ";
132             document.getElementById('ultimanota').innerHTML = 'FA';
133             document.getElementById('labelmelodia').innerHTML = arraycomponer;
134             contador++;
135             }
136             if( y > 101 && y < 111) {
137             espacio += 50;
138         ctx.arc(espacio, 106, pointSize, 0, Math.PI * 2, false);
139             arraycomponer += "MI - ";
140             document.getElementById('ultimanota').innerHTML = 'MI';
141             document.getElementById('labelmelodia').innerHTML = arraycomponer;
142             contador++;
143             }
144             if( y > 111 && y < 121) {
145             espacio += 50;
146         ctx.arc(espacio, 116, pointSize, 0, Math.PI * 2, false);
147             arraycomponer += "RE - ";
148             document.getElementById('ultimanota').innerHTML = 'RE';
149             document.getElementById('labelmelodia').innerHTML = arraycomponer;
150             contador++;
151             }
152             if( y > 121 && y < 135) {
153             espacio += 50;
154         ctx.arc(espacio, 125, pointSize, 0, Math.PI * 2, false);
155             arraycomponer += "DO - ";
156             document.getElementById('ultimanota').innerHTML = 'DO';
157             document.getElementById('labelmelodia').innerHTML = arraycomponer;
158             contador++;
159             }
160             ctx.stroke();
161         ctx.fill();
162             }
163             else{
164                         alert("Haz click dentro del pentagrama y en la posicion de una nota
165     valida.");
166             }
167     }
```

```
168
169     function enviarreproductor(){
170               borrarreproductor();
171               var response = arraycomponer.toLowerCase();
172               var x = 1;
173               cargarpentagrama(response,x);
174               document.getElementById('sonandomelodia').innerHTML = 'ComposiciÃ³n: '+
175     arraycomponer2;
176               document.getElementById('sonandonota').innerHTML = ' ';
177               borrar();
178
179
180                         detenido = false;
181                         console.log("Deteniendo...");
182                         ctx3.lineWidth = 0;
183                         ctx3.strokeStyle = "white";
184                         ctx3.fillStyle = "white";
185                         ctx3.beginPath();
186                         ctx3.fillRect(espaciovertical-5, 0, 12, 12);
187                         ctx3.closePath();
188                         ctx3.fill();
189
190
191                         ctx3.strokeStyle = "white";
192                         ctx3.fillStyle = "white";
193                         ctx3.beginPath();
194                         ctx3.fillRect(espaciovertical-5, 135, 12, 5);
195                         ctx3.closePath();
196                         ctx3.fill();
197                         document.getElementById('sonandonota').innerHTML = ' ';
198                         document.getElementById("btnpausar").value = "Pausar";
199
200                         ctx.clearRect(0, 0, canvas.width, canvas.height);
201                         ctx.drawImage(img2, 0, 0);
202               clearInterval(i);
203                         espaciovertical = 60;
204
205               document.getElementById("btnreproducir").disabled=false;
206               document.getElementById("btnpausar").disabled=true;
207               document.getElementById("btndetener").disabled=true;
208               document.getElementById("tempo").disabled=false;
209
210     }
211
212     function descargar(){
213               descargarArchivo(generarXml(arraycomponer), 'melodia-flauta.xml');
214               //Genera un objeto Blob con los datos en un archivo XML
215               function generarXml(arraycomponer) {
216                         var texto = [];
217                         var arrayDeNotas = arraycomponer.split(" - ");
218                         texto.push('<?xml version="1.0" encoding="UTF-8" ?>\n');
219                         texto.push('<notas>\n');
220                         for (var i=0; i < arrayDeNotas.length-1; i++) {
221                         texto.push('<nota
222     nombre="'+arrayDeNotas[i].toLowerCase()+'">'+arrayDeNotas[i].toLowerCase()+'</nota>\n');
223                         }
```

```
224                         texto.push('</notas>\n');
225                         return new Blob(texto, {
226                         type: 'application/xml'
227                         });
228             };
229
230             function descargarArchivo(contenidoEnBlob, nombreArchivo) {
231                var reader = new FileReader();
232                         reader.onload = function (event) {
233                                  var save = document.createElement('a');
234                                  save.href = event.target.result;
235                                  save.target = '_blank';
236                                  save.download = nombreArchivo || 'archivo.dat';
237                                  var clicEvent = new MouseEvent('click', {
238                                           'view': window,
239                                                    'bubbles': true,
240                                                    'cancelable': true
241                                  });
242                                  save.dispatchEvent(clicEvent);
243                                  (window.URL ||
244     window.webkitURL).revokeObjectURL(save.href);
245                         };
246                         reader.readAsDataURL(contenidoEnBlob);
247             }
248     }
```