Musicalizador

Pensamiento Computacional

Trabajo Práctico Final

1. Objetivos

Los objetivos del presente trabajo se encuentran divididos en dos partes:

1. Realizar un programa que permita, a partir de una partitura, sintetizar las notas predefinidas.
2. A partir de una partitura, enviar las notas que se quieren tocar a un instrumento real. En este caso, se tratará de un metalófono.

2. Alcance del Trabajo Práctico

Mediante el presente trabajo se busca que el estudiante adquiera y aplique los conocimientos sobre los siguientes temas:

- Fundamentos de Python.
- Modularización del proyecto.
- Clases y funciones.
- Archivos de texto.
- Comunicación con un sistema físico real.
- Traducción de un problema a un modelo computacional.
- Buenas prácticas de desarrollo.

3. Introducción

3.1. El Sonido

La percepción que sentimos como sonido se debe a la propagación de ondas en el aire que es medida por nuestros oídos. En los sonidos podemos distinguir una intensidad, por la cual hay sonidos más fuertes que otros; un tono, por el cual hay sonidos más agudos y sonidos más graves; y un timbre, por el cual no todas las cosas que tienen el mismo tono suenan igual. Supongamos una onda de 440 Hz que oscila durante 27 milisegundos (Fig. 1). Esta onda es sencilla de concebir matemáticamente, pero es imposible de obtener por medios físicos. Una onda física no podría comenzar a oscilar instantáneamente, ni tampoco podría atenuarse inmediatamente. Si oyéramos una onda de esas características, percibiríamos un chasquido al comienzo y al final de la misma. Una onda producida por un elemento físico tendrá un incremento paulatino hasta su máximo producido por el **ataque** y su entrada en resonancia, irá apaciguándose por disipación de energía durante el **sostenido**, y luego tardará unos instantes en volverse a poner en reposo durante el **decaimiento** (Fig. 2).

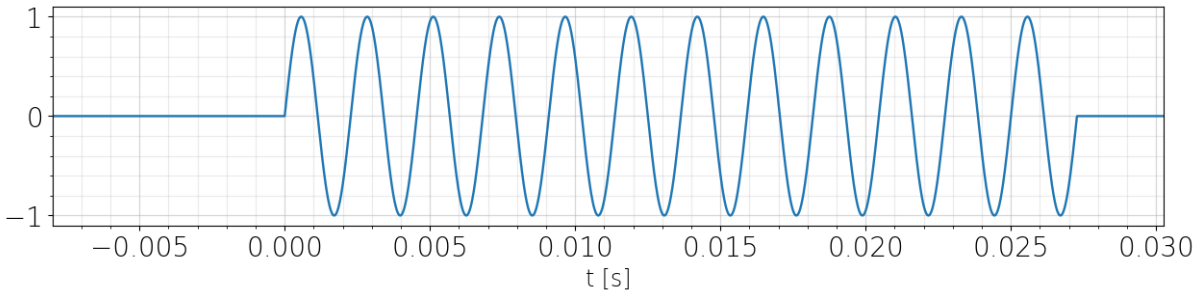

Figure 1: Onda sinusoidal de 440 Hz de 0.027 segundos de duración.

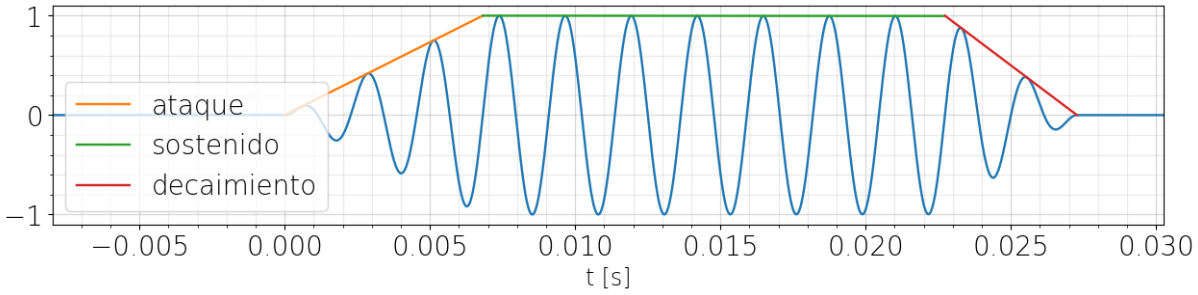

Figure 2: Ataque 0 s < t < 0.007 s; Sostenido 0.07 s < t < 0.023 s; Decaimiento t > 0.023 s

Hasta ahora tenemos una onda pura y, como podría anticiparse, tampoco las ondas puras son lo que abunda en la naturaleza. La mayor parte de los dispositivos, además de vibrar con la frecuencia fundamental que percibimos como tono vibran en múltiplos de la misma, llamados armónicos o en frecuencias parásitas. Ni siquiera instrumentos como un diapasón son capaces de entregar un sonido puro, con una oscilación de una sola frecuencia. Las ondas generadas por un instrumento oscilarán a una frecuencia dada por su *tono*, y generarán resonancias en diferentes armónicos con menor amplitud (Fig. 3). Juntando el *tono*, dado por la frecuencia fundamental; la *intensidad*, modulada por el **ataque** – **sostenido** – **decaimiento** y el *timbre*, dado por la adición de armónicos; una onda de 440 Hz producida por un instrumento real, tendría un aspecto similar al de la Figura 4.

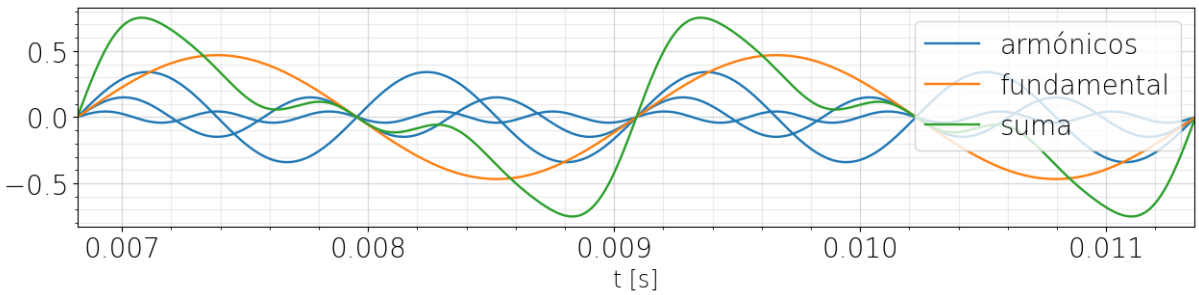

Figure 3: Armónimos de una frecuencia fundamental de 440 Hz y su suma. Esta onda podría imitar el timbre de un piano.

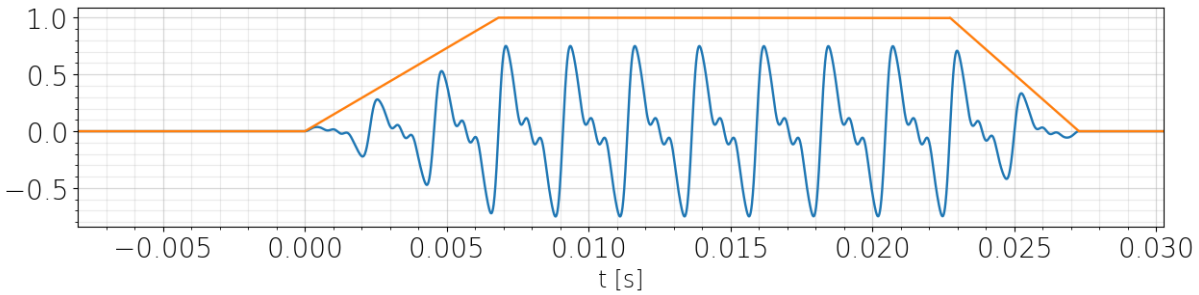

Figure 4: Onda de 440 Hz sintetizada en tono, intensidad y timbre.

3.2. La Escala Musical

La música se genera en base a sonidos. Ahora bien, para que la misma suene agradable al oído, se construyen escalas que fijan qué tonos están permitidos. Todas las escalas se construyen en base a particiones de la octava, y luego se replican en las octavas subsiguientes. Se considera que hay una octava entre dos notas donde una duplica la frecuencia de la otra.

Las escalas occidentales son dodecafónicas, se construyen en base a 12 particiones equidistantes de una octava. Además, se define que la nota La de la cuarta octava tiene una frecuencia de 440Hz. Las notas de la escala son Do, Do♯ (o Re♭), Re, Re♯ (o Mi♭), Mi, Fa, Fa♯ (o Sol♭), Sol, Sol♯ (o La♭), La, La♯ (o Si♭) y Si, en ese orden. Por razones de simplificar la notación, introduciremos el sistema de notación musical anglosajón o cifrado inglés. En el mismo a cada nota se le asigna una letra entre la A y la G, comenzando por la nota La y continuando cíclicamente a la octava siguiente. Sabiendo que hay 12 notas, equidistantes exponencialmente, que el La de la cuarta se fija en 440 Hz, que las octavas avanzan duplicando, y el nombre de las notas, podemos reconstruir la frecuencia para cualquier nota dada. La Tabla 1 muestra un ejemplo de ello.

Table 1: Frecuencias para la cuarta octava (afinación 440).

Notación franco-belga	Notación inglesa	Frecuencia (Hz)
si3	B4	493.883
la♯3/si♭3	A♯4/B♭4	466.164
la3	A4 (la 440)	440.000
sol♯3/la♭3	G♯4/A♭4	415.305
sol3	G4	391.995
fa♯3/sol♭3	F♯4/G♭4	369.994
fa3	F4	349.228
mi3	E4	329.628
re♯3/mi♭3	D♯4/E♭4	311.127
re3	D4	293.665
do♯3/re♭3	C♯4/D♭4	277.183
do3	C4 (do central)	261.626
si2	B3	246.942
la♯2/si♭2	A♯3/B♭3	233.082
la2	A3	220.000

3.3. El Formato WAVE

El formato WAVE forma parte de la especificación RIFF de Microsoft. El mismo permite almacenar en modo raw (sin compresión) muestras de audio. Un archivo WAVE es un archivo binario compuesto por un preámbulo, y dos chunks. El primero de ellos es de tamaño fijo, y contiene la información acerca de las características del track; mientras que el segundo contiene las muestras de audio que componen al mismo. Para este trabajo se utilizará el formato PCM (esto es, sin compresión), 16 bits de resolución y formato monoaural. El contenido binario de un archivo WAVE de estas características es el de la Figura 5. Tanto los números de 16 bits como de 32 en el formato WAVE, se almacenan en el archivo en formato little endian.

Table 2: Contenido binario de un archivo WAVE PCM monoaural de 16 bits.

Campo	Significado	Tipo	Valores
ChunkId	Identifica un archivo RIFF	`str[4]`	Vale siempre `"RIFF"`
ChunkSize	Tamaño del archivo	`uint` 32 bits	$36 + 2n$
Format	Identifica un WAVE	`str[4]`	Vale siempre `"WAVE"`
SubChunk1ID	Encabezado chunk1	`str[4]`	Vale siempre `"fmt "`
SubChunk1Size	Tamaño chunk1	`uint` 32 bits	Vale siempre 16
AudioFormat	Formato de audio	`uint` 16 bits	Vale 1
NumChannels	Número de canales	`uint` 16 bits	Vale 1
SampleRate	Tasa de muestreo	`uint` 32 bits	8000, 44100, etc.
ByteRate	Tasa de bytes	`uint` 32 bits	2 $\times$ SampleRate
BlockAlign	Alineación de bloque	`uint` 16 bits	Vale 2
BitsPerSample	Bits por muestra	`uint` 16 bits	Vale 16
SubChunk2ID	Encabezdo chunk2	`str[4]`	Vale siempre `"data"`
SubChunk2Size	Tamaño chunk2	`uint` 32 bits	$2n$
Data	Muestras	`int[n]`	Secuencia de n muestras

4. Desarrollo del TP

El trabajo práctico se desarrollará en varias partes.

4.1. Sintetizador

En una primera instancia realizar una síntesis sobre un archivo WAVE en base a la configuración de un sintetizador y una secuencia de notas. El tipo de sintetizador deberá ser parametetrizable acorde a los distintos tipos de sintetizadores que se presentan en la sección síntesis.

La sintaxis de la ejecución será:

```
$ ./sintetizador [-f <frecuencia>] -i <instrumento> -p <partitura> -o <
    audio.wav>
> sintetizador [-f <frecuencia>] [-i <instrumento>] -p <partitura> -o <
    audio.wav>
```

Table 3: Parámetros del ejecutable.

Parámetro	Descripción
-p	Nombre del archivo de entrada que describe la partitura
-i	Nombre del archivo de entrada que describe un instrumento
-o	Nombre del archivo de salida (archivo WAVE a generar)
-f	Frecuencia de muestreo. Una entre: 8000, 9600, 11025, 12000, 16000, 22050, 24000, 32000, 441000, 48000, 88200, 96000 (48000 si se omite el parámetro)

4.2. Comunicación con un Metalófono

En la segunda parte, se deberá hacer otro ejecutable que se comunique con el metalófono. El mismo leerá la misma partitura, y le enviará por un canal de comunicación las notas que el metalófono debe tocar.

La sintaxis de ejecución será:

```
$ ./metalofono -p <partitura.txt> -d <dispositivo>
> metalofono -p <partitura.txt> -d <dispositivo>
```

4.3. Partitura

El archivo de partitura, será un archivo de texto donde cada línea representará una nota a ser ejecutada. Cada línea contendría un tiempo de inicio de la ejecución de la nota, la nota a interpretar y la duración. Todos los tiempos se especificarán en segundos. Las notas se especificarán usando el sistema de notación musical anglosajón, utilizando el sufijo “s” para notas sostenidas (♯) y “b” para notas bemoles (♭) y su octava correspondiente.

Por ejemplo:

```
0   A4  .5
.5  Bb4 .5
1   B4  .5
1.5 C4  .5
2   Cs4 .5
2.5 D4  .5
```

Describe una secuencia de semitonos de medio segundo de duración, desde el La hasta el Re de la cuarta octava. La duración de la ejecución del fragmento será de 3 segundos.

Se debe nota que, al indiciar el inicio de las notas únicamente, es posible solaparlas, por lo que podría sonar más de una nota en cada instante de tiempo, dando así la posibilidad de sintetizar acordes.

En el archivo se desconoce la cantidad de notas a ejecutar, y no es requisito que el mismo venga ordenado cronológicamente (tampoco hace falta).

4.4. Configuración del Sintetizador

El archivo de sintetizador, será un archivo de texto plano con información sobre los armónicos, y las envolventes para modular la amplitud.

El archivo de sintetizador estará dividido en cuatro partes:

- **Armónicos**: donde se especificará el número de armónicos, y luego pares de múltiplo e intensidad.
- **Ataque**: donde se especificará el tipo de ataque, y luego los parámetros.
- **Sostenido**: donde se especificará el tipo de sostenido, y luego los parámetros.
- **Decaimiento**: donde se especificará el tipo de decaimiento, y lueog los parámetros.

Los moduladores de amplitud serán alguno de los de la Tabla 3.

Por ejemplo, un archivo de sintetizador podría tener este contenido:

```
8
1 0.577501
2 0.577501
3 0.063525
4 0.127050
5 0.103950
6 0.011550
7 0.011550
8 0.011550
TRI 0.05 0.03 1.3
CONSTANT
INVLINEAR .02
```

En él se define que hay 8 armónicos en octavas, un ataque `TRI` de duración 0.05 segundos, un sostenido `CONSTANT` y un decaimiento `INVLINEAR` de duración 0.02 segundos.

Table 4: Moduladores de amplitud. Para su uso en A (Ataque), S (Sostenido), D (Decaimiento).

Nombre	Ecuación	Parámetros	Uso
`CONSTANT`	$f(t) = 1$	–	S
`LINEAR`	$f(t) = \frac{t}{t_0}$	t_0	A
`INVLINEAR`	$f(t) = \max\left\{1 - \frac{t}{t_0}; 0\right\}$	t_0	S, D
`SIN`	$f(t) = 1 + a\sin(ft)$	a, f	S
`EXP`	$f(t) = e^{\frac{5(t-t_0)}{t_0}}$	t_0	A
`INVEXP`	$f(t) = e^{\frac{-5t}{t_0}}$	t_0	S, D
`QUARTCOS`	$f(t) = \cos\left(\frac{\pi t}{2t_0}\right)$	t_0	S, D
`QUARTSIN`	$f(t) = \sin\left(\frac{\pi t}{2t_0}\right)$	t_0	A
`HALFCOS`	$f(t) = \frac{1+\cos\left(\frac{\pi t}{t_0}\right)}{2}$	t_0	S, D
`HALFSIN`	$f(t) = \frac{1+\cos\left(\pi\left(\frac{t}{t_0}-\frac{1}{2}\right)\right)}{2}$	t_0	A
`LOG`	$f(t) = \log_{10}\left(\frac{9t}{t_0}+1\right)$	t_0	A
`INVLOG`	$f(t) = \begin{cases} \log_{10}\left(\frac{-9t}{t_0}+10\right) & t < t_0 \\ 0 & t \geq t_0 \end{cases}$	t_0	S, D
`TRI`	$f(t) = \begin{cases} \frac{ta_1}{t_1} & t < t_1 \\ \frac{t-t_1}{t1-t_0} + a_1 & t > t_1 \end{cases}$	t_0, t_1, a_1	A
`PULSES`	$t' = \frac{t}{t_0} - \left\lfloor \frac{t}{t_0} \right\rfloor$, $f(t') = \min\left\{\left\lvert \frac{1-a_1}{t_1}(t' - t_0 + t_1)\right\rvert + a_1\right\}$	t_0, t_1, a_1	S

4.5. Síntesis

4.5.1. Sintetizador Aditivo

La síntesis del sonido se realizará para cada nota por separado, realizando una suma sobre el audio de las demás notas.

Dada una nota que comienza en el instante t_0, de frecuencia f y duración d, con un sintetizador con una serie de n armónicos donde están definidos sus múltiplos M_i, y sus intensidades I_i, además de sus funciones de ataque f_a, sostenimiento f_s, y decaimiento f_d, se procede como sigue.

El timbre del instrumento se obtiene de computar la siguiente expresión:

$$y(t) = \sum_{i=1}^{n} I_i \sin\left(2\pi f M_i (t - t_0)\right)$$

Esto nos da una señal infinita, la misma será ajustada según la modulación de amplitud. La modulación de amplitud se compone por la concatenación de ataque, sostenido y decaimiento.

Antes de explicar cómo se combinan, vale observar que todas las funciones para ser usadas como ataque, varían su amplitud de 0 a 1 en un intervalor de tiempo t_a, todas las funciones de decaimiento recorren el camino inverso de amplitud decreciente de 1 a 0 en un tiempo t_d, y todas las funciones de sostenido comienzan en 1 y son siempre positivas. Esto es importante para comprender cómo se las empalma.

Una nota no puede durar nunca menos que el tiempo de ataque t_a, y una vez finalizada la nota, se ejecuta el decaimiento, de tiempo t_d. Luego, la función de modulación estará dada por:

$$m(t) = \begin{cases} f_a(t - t_0) & t_0 < t < t_0 + t_a \\ f_s(t - (t_0 + t_a)) & t_0 + t_a < t < t_0 + d \\ f_s(t_0 + d) f_d(t - (t_0 + d)) & t_0 + d < t < t_0 + d + t_d \\ 0 & \text{para todo otro } t \end{cases}$$

Observar que la misma es una función continua.

Finalmente, la amplitud de la nota estará dada por:

$$a(t) = A y(t) m(t)$$

Esta función será distinta de cero como mucho en el intervalo $t_0 < t < t_0 + d + t_d$. La constante A debe ser elegida convenientemente para darle el volumen a nuestro instrumento; al elegirla notar que un valor pequeño hará que al truncar sobre 16 bits pierdan muchos valores intermedios, y notar que un valor muy grande puede hacer que la adición de varias notas genere un desbordamiento (*overflow* inglés), lo cual en el audio se va a traducir en una saturación del sonido.

El único paso que resta para llevar estas notas a un archivo WAVE es muestrear estas ondas según la frecuencia dada. Dicha discretización debe hacerse en tantas muestras por segundo como las indicadas en la frecuencia de muestreo f_m, con t variando intervalos $1/f_m$.

4.6. Metalófono

Una parte importante del trabajo es interactuar con un instrumento real. En este caso, estaremos introduciendo un metalófono. En particular, este metalófono se encuentra robotizado, es decir, que cada tecla tiene un servo encargado de mover el martillo y tocar la nota según se le indique. Hay 30 notas y 30 servos los cuales se podrán comandar a través de un pequeño pedazo de código (provisto por los docentes) que traducirá las notas en pulsos que llegarán a cada uno de los servos.

5. Restricciones

El trabajo final está sujeto a las siguientes condiciones:

1. El trabajo se realiza en grupo de exclusivamente 4 integrantes como máximo.
2. El proyecto debe reflejar todas las buenas prácticas hechas durante todo el semestre.
    1. Modularización en funciones.
    2. Las funciones deben tener un comportamiento único, *sin efectos secundarios*.
    3. Sin código global, excepto en casos especiales como el de constantes.
    4. Modularización en archivos.
    5. Nombrar los archivos con la convención correspondiente, minúscula y guiones bajo.
    6. El código completamente en inglés, incluso los comentarios.
    7. Nombrar las funciones y variables correctamente, de tal manera que sea claro qué hacen.
    8. No debe exisitir ningún tipo de excepción, e.g., `ValueError`, `IndexError`, etc.
    9. Toda clase/función debe tener tests. Se sugiere utilizar `pytest`.

Si bien no se considera un requisito la definición y uso de clases, sí se valorá positivamente su utilización. En ese caso solo debe haber funciones aisladas en casos especiales.

6. Entrega del Trabajo Práctico

Se debe realizar una entrega digital a través del campus de la materia, compuesta por un único archivo comprimido con todos los contenidos del trabajo.

El nombre de dicho archivo debe cumplir el siguiente formato:

```
tp2_legajoA_legajoB_legajoC_legajoD.zip
```

donde `legajoX` es el legajo de cada integrante del grupo (ordenados de menor a mayor) y `.zip` es la extensión del archivo comprimido (también puede ser `.zip`, `.tar`, `.gz`, o `.tar.gz`, `.rar`).

A su vez, el archivo comprimido debe contener los siguientes elementos:

1. El código fuente en varios archivos de python (.py).
2. La documentación del desarrollo **en formato PDF**, que contiene los siguientes items:
    1. Carátula del trabajo práctico, nombre y apellido del alumno y dirección de correo electrónico.
    2. Objetivos del trabajo.
    3. Diseño del programa. Esto incluye una explicación sobre cómo funcionan las distintas partes del programa y las alternativas de diseño que fueron consideradas.
    4. Reseña sobre problemas encontrados y soluciones.
    5. Indicaciones para ejecutar correctamente el programa y las pruebas.
    6. Resultados de ejecuciones, incluyendo capturas de pantalla (como imagen o texto), bajo condiciones normales e inesperadas de entrada.
    7. Bibliografía.

Nota: el informe debe estar redactado en *correcto* castellano.

7. Bibliografía

[1] Beranek, Leo & Mellow, Tim & Zuckerwar, Allan (2013). Chapter 1 - Introduction and terminology. Acoustics: Sound Fields and Transducers. The Journal of the Acoustical Society of America. 134. 2337. 10.1121/1.4816547.

[2] Kabal, Peter (2017). Audio File Format Specifications. http://www-mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/WAVE.html.

[3] Lynn, Benn. The Karplus-Strong Algortihm. https://crypto.stanford.edu/~blynn/sound/karplusstrong.html.