UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Desarrollo de herramienta para generación procedural de pistas musicales

Trabajo de graduación presentado por Jorge Andrés Pérez Barrios para optar al grado académico de Licenciado en Ingeniería en Ciencias de la Computación y Tecnologías de la Información

Guatemala,

2022

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

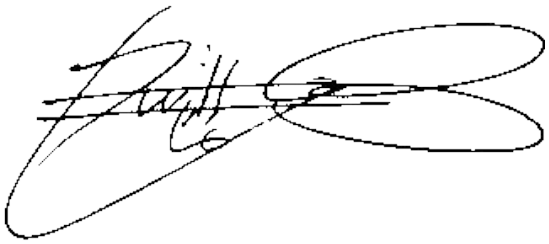
Desarrollo de herramienta para generación procedural de pistas musicales

Trabajo de graduación presentado por Jorge Andrés Pérez Barrios para optar al grado académico de Licenciado en Ingeniería en Ciencias de la Computación y Tecnologías de la Información

Guatemala,

2022

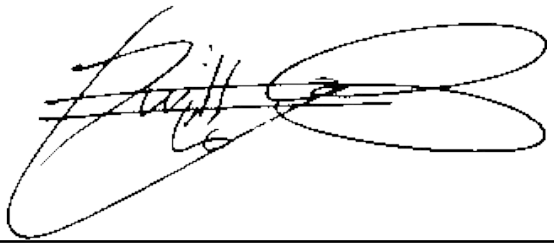
Vo.Bo.:



(f) _____

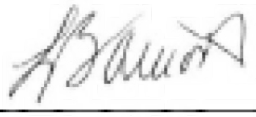
Ing. Christopher Darwin Berganza Ispache

Tribunal Examinador:



(f) _____

Ing. Christopher Darwin Berganza Ispache



(f) _____

Ing. Douglas Leonel Barrios Gonzalez

(f) _____

Lic. Diego Raúl Guzman Verbena

Fecha de aprobación: Guatemala, 8 de diciembre de 2022.

Índice

Lista de figuras

Lista de cuadros

Resumen

Hacer música para un videojuego o un "stream" es un proceso bastante complicado, ya que esta es una de las partes más importantes dentro de un juego debido a que esta ayuda a la psicología que este quiere transmitir. El compositor requiere de bastantes conocimientos tanto teóricos como prácticos para poder componer una canción que haga coherencia y, más importante, que transmita los sentimientos que esa escena quiera representar. Y aun así toma bastante tiempo componer una pista musical que se adapte al estilo del videojuego o "stream", llegando a tomar meses o incluso años. Otra de las complejidades de adquirir música para un creador de contenido es el tema de derechos de autor, ya que autores muy comerciales pueden tomar acciones legales si se descubre que un individuo u organización utilizan su música; siendo la única opción, para un creador de contenido independiente, descargar música libre de regalías, pero esta música pueda que no se adapte a su contenido. Es por eso por lo que este trabajo tiene como objetivo la realización de una herramienta "open source" escrita en Python y utilizando el marco de trabajo SCAMP, que puede utilizarse para generar música automática y proceduralmente, libre de regalías a pagar, con el fin de ayudar a pequeños equipos de desarrollo de videojuegos o creadores de contenido independientes que se les dificulte o no tengan la posibilidad de crear música por su cuenta. Esta herramienta genera música de tres sensaciones o "moods" diferentes (agresivo, épico y melancólico), pertenecientes a subgéneros del metal. Se llegó a la conclusión que es posible la creación de una herramienta open source que genera música computacional proceduralmente, de manera casi instantánea; personalizable y escalable a diferentes "moods" de un género musical e incluso otros géneros musicales, así como conectar esta herramienta a una estación de trabajo de audio digital.

Abstract

Making music for a video game or a stream is quite a complicated process, since this is one of the most important parts of a game, for it helps the psychology that it wants to convey. The composer requires a lot of theoretical and practical knowledge to be able to compose a song that makes coherence and, more importantly, that transmits the feelings that the scene wants to represent. And yet it takes a long time to compose a music track that suits the style of the video game or stream, taking months or even years. Another of the complexities of acquiring music for a content creator is the issue of copyright, since a commercial author can take legal actions if an individual or organization is found to be using their music; being the only option for an independent content creator to download royalty-free music, but this music may not suit their content. That is why this work aims to make an open-source tool, written in Python and using the SCAMP framework, which can be used to generate royalty-free music automatically and procedurally, to help small game development teams or independent content creators who have difficulty or are unable to create music on their own. This tool generates music with three different sensations or "moods" (aggressive, epic, and melancholic), belonging to subgenres of metal. It was concluded that it is possible to create an open-source tool that generates computational music procedurally, almost instantaneously; customizable, and scalable to different moods of a musical genre and even other musical genres, as well as connecting this tool to a digital audio workstation.

CAPÍTULO 1

Introducción

La composición de música para cualquier medio audiovisual es vital para el sentimiento que este quiere transmitir. Sin embargo, este es un proceso complejo que necesita de conocimientos en su teoría y mucha creatividad en su práctica, ya que esta puede estar compuesta por cualquier tipo de sonidos ya que se pueden utilizar de cualquier tipo para componer una canción, estos siempre sonaran musicales para el oído humano, siempre y cuando lleven una secuencia rítmica. Sin embargo, se utilizan sonidos que se han pensado que son específicamente musicales, ya que si las composiciones que se hiciesen con sonidos aleatorios sería totalmente caótica, en vez de ser una composición con ritmo y coherencia (Hewitt, 2008). Estos sonidos musicales son el resultado de ondas de sonido que son vibraciones que causan disturbios en la atmósfera, causando cambios en la densidad y presión del aire (Collins, 2008). Es el manejo de estas vibraciones lo que toma tanto tiempo en controlar para que suenen de manera rítmica y melódica, ya sea en instrumentos físicos como en sintetizadores de computadora por medio de un oscilador. Por otra parte, la generación procedural de contenidos ("Procedural Content Generation", PCG en inglés) permite a un programador crear cualquier tipo de contenido que se quiera, así como arte visual como mapas y terrenos para videojuegos en dos y tres dimensiones, la generación de texturas; y contenido auditivo como sonidos ambientales o música, siempre y cuando esta generación esté basada en algoritmos siguiendo reglas impuestas por el programador y el uso de números pseudoaleatorios. (Green, 2016)

Este proyecto consiste en la generación procedural de música, utilizando toda la teoría musical para crear una herramienta "open source" escrita en Python y utilizando SCAMP como marco de trabajo, que puede utilizarse para generar música procedural, en instantes, libre de regalías a pagar, personalizable con el fin de ayudar a pequeños equipos de desarrollo de videojuegos o creadores de contenido independientes que se les dificulte o no tengan la posibilidad de crear música para su contenido. La música que se planteó componer para que esta herramienta genere es del género del metal, más específicamente, tres diferentes "moods" (o sensaciones) que este género puede proveer. Estos "moods" son: agresivo ("death

metal"), épico ("djent") y el melancólico ("melodic death metal"). Para componer una pista musical, se utilizó la estructura básica de una canción: introducción, versos, coros y outro, siendo cada parte de la estructura un bloque de bloques generado a partir de un ritmo fundamental, al que se le pueden añadir ritmos preestablecidos.

Así como se mencionó antes, el objetivo de esta herramienta es proveer una librería "open source" para que pequeños estudios de desarrollo de videojuegos y creadores de contenido puedan obtener música de manera gratuita, sin riesgos legales de "copyright" si es que estos no poseen el equipo y conocimientos necesarios para componer una canción por su cuenta. Esta herramienta también provee la libertad al usuario de poder definir el "mood" que quiere transmitir la canción, la escala, la clave y el tempo; así como poder modificar el código fuente para establecer sus propias reglas musicales y ritmos preestablecidos.

CAPÍTULO 2

Justificación

Se dio la tarea de realizar este proyecto, ya que para la creación de un videojuego, se necesita escenificar el ambiente al jugador por medio de un elemento musical. Es por esto que se necesita de un compositor o productor de música para darle el efecto deseado según los sentimientos que cada escena desea transmitir. Sin embargo, para los estudios indie de videojuegos que aún no tienen un compositor o alguien que sepa de música, es una tarea complicada que puede, ir incluso, en contra del tiempo esperado para el desarrollo de un demo o videojuego ya que necesitan crear pistas muy largas y recurrentes. Es por esto por lo que se plantea la idea de este proyecto, para hacer una herramienta, fácil de usar, accesible a cualquier persona para crear pistas musicales aleatorias (proceduralmente). Inclusive, a creadores de otro tipo de contenido, así como "podcasts", "streaming", etc., quienes necesiten de música ambiental o música de fondo, libre de derechos de autor que no afecten sus gastos o personas particulares que solo quieren escuchar algo nuevo. En el anexo 10.1 se pueden encontrar las encuestas realizadas a los usuarios finales.

CAPÍTULO 3

Objetivos

3.1. Objetivos generales

- Componer una pista musical aleatoriamente.
- Diseñar y programar un generador procedural de música capaz de componer pistas para un género musical en específico.

3.2. Objetivos específicos

- Crear un algoritmo que se adapte a las reglas del género musical.
- Adaptar el algoritmo a los "moods" del género musical escogido.
- Crear una interfaz gráfica para que el usuario pueda interactuar con el generador.

CAPÍTULO 4

Marco teórico

4.1. Teoría musical

4.1.1. Ruido y sonidos musicales

La música está compuesta totalmente por sonidos. Así como lo describe Michael Hewitt, realísticamente se pueden utilizar cualquier tipo de sonidos para componer una canción. Desde sonidos de naturaleza como lluvia, las olas del mar en la playa, sonidos de animales incluso sonidos de maquinas y charlas. Así mismo los sonidos de percusión. Ya sean sonidos de rasguños, golpes, o sacudidas a diferentes objetos; estos siempre sonaran musicales para el oído humano, siempre y cuando lleven una secuencia rítmica. Sin embargo, si la musica estuviera compuesta únicamente por ruidos, las composiciones y la audiencia serían muy limitados. Es por eso que se utilizan sonidos que se han pensado que son específicamente musicales. (Hewitt, 2008)

Los sonidos musicales que escuchamos son el resultado de las ondas de sonido. Estas ondas no son más que vibraciones que causan disturbios en la atmósfera. Las ondas se propagan a través del aire así como cambian conforme a la presión. Esencialmente cuando las partículas del aire colisionan entre sí para causar regiones de alta o baja densidad: mientras más partículas de aire, más densidad y presión (Collins, 2008). Cuando estos sonidos son caóticos y confusos, se les llama ruidos. Estos ruidos no producen tanto placer al ser escuchados. Pero cuando algún objeto (generalmente instrumentos musicales) produce sonidos regulares, ordenados y con patrones en las ondas, estos sonidos producen placer. Y combinados juntos producen música, en lugar de ruido.

4.1.2. Periodicidad

Al haber un sonido, las partículas del aire vibran de una manera repetitiva a un ritmo. Este comportamiento físico, de repetirse de manera constante, muchas veces por segundo, es llamado una oscilación. La vibración periódica de un objeto es caracterizado por el periodo que se mide por el largo de una repetición individual en el tiempo. Así mismo, existe la frecuencia de vibración, que es el numero de repeticiones por segundo. Esta frecuencia de vibración es medida en Hertz (Hz). Sin embargo, el periodo es inversamente proporcional a la frecuencia, por lo que una oscilación de 100 Hz tiene un período de 1/100 segundos. (Collins, 2008)

Existe un método para analizar la periodicidad de estas ondas de sonido. Este método es el análisis de Fourier, el cual permite analizar cualquier onda perfectamente periódica y desglosarla en su oscilación más básica. Esta observación puede ser simplificada, así como lo describe Nick Collins, con el sistema físico oscilatorio más simple, una masa cualquiera que es afectada por una fuerza que se restaura. Este sistema puede ser un objeto de masa x unido a un resorte sin fricción. El resultado del análisis de Fourier sobre este sistema nos da como resultado la función más básica de forma de onda que se puede encontrar: la función sinusoidal. El sonido de esta forma de onda es el sonido más puro de una nota, similar al sonido que produce un diapasón y puede ser sintetizado y reproducido por una computadora, llamado oscilador, por la sencillez de su ecuación y no puede simplificarse. De esto se discutirá más a detalle en la sección 4.1.6 que tratará sobre la síntesis.

Según la teoría de Fourier, cada una de las funciones sinusoidales pueden desglosarse en componentes sinusoidales. Cada componente esta conformado por una frecuencia y una fase, la cual es la posición inicial dentro de un solo ciclo, al igual que la amplitud, la cual muestra cual fuerte es el componente. Además, la frecuencia mas baja que un componente puede llegar es llamada la frecuencia fundamental. La frecuencia del resto de componentes están dadas por los múltiplos enteros de la frecuencia fundamental, llamadas las series armónicas. Así que los componentes de una onda con frecuencia fundamental de 100Hz, tendrán una frecuencia de 100, 200, 300, etc. Estos múltiplos son llamados armónicos o sobre tonos, contando desde la frecuencia del segundo armónico (el doble de la frecuencia fundamental).

4.1.3. Tono

La frecuencia de una nota musical es un atributo psicoacústico del sonido, el cual es subjetivo ya que es la sensación principal causada por la estimularon de una nota cuando entra al oído. El rango aproximado de frecuencias audibles es de 20Hz hasta 20,000Hz (Dodge y Jerse, 1997). La frecuencia de una nota musical se le llama tono. Este tono se refiere a cuán alta o cuán baja es la nota. En otras palabras, es la percepción del oído de la longitud de onda cuando un sonido es producido. Tonos bajos tienen una longitud de onda relativamente mayor que tonos mas altos. El rango de frecuencias generalmente usado en la música es de aproximadamente siete octavos de sonido; lo que equivale al rango de notas que se pueden encontrar en un piano de cola para conciertos. Este rango, aproximadamente, es desde los 30Hz hasta los 5KHz. Los rangos de los demás instrumentos musicales están ubicados dentro de este rango; cada rango es característico de ese instrumento. (Hewitt, 2008)

Las notas producidas por los instrumentos es típicamente estable en sus frecuencias.

Pero para que varios instrumentos puedan trabajar entre sí, es indispensable afinarlos a la misma frecuencia. De esta manera, una nota La de un instrumento es idealmente la misma nota La que puede tocar otro instrumento. La frecuencia más común, o el tono estándar, para afinar los instrumentos para la nota La es de 440Hz. Otro estándar musical es la nota Do central. Esta nota se usa como punto de referencia para decidir si una nota debe de ir en clave de sol, para notas con mayor tono o clave de fa, para notas con tono más bajo.

4.1.4. Amplitud

Así como se mencionó en la sección 4.1.2, uno de los atributos de las ondas de sonido es la amplitud. Este determina la intensidad que una nota. También llamado volumen, este muestra qué tan fuerte o suave es un sonido, dependiendo de la altura de la onda. Esto es así ya que una onda con mayor altura, o mayor amplitud, lleva consigo más energía; a diferencia de una onda con menor altura, llevará consigo menos energía. Consecuentemente, un incremento en la amplitud en un sonido es registrado como un mayor volumen y viceversa. Los niveles de volumen pueden medirse con decibeles (dB), siendo 0dB silencio. Esta medición es una función logarítmica, ya que el oído humano trabaja aproximadamente de esta manera. Los decibeles no son más que la conversión de la amplitud a un nivel de referencia. La ecuación estándar es la siguiente:

$$dB = 10log_{10}\left(\frac{input}{referencia}\right)$$

Donde la variable input es el poder de la señal, el cual es el cuadrado de la amplitud y se utiliza un logaritmo en base 10. La medición de los decibeles siempre es con base en una referencia. (Collins, 2008)

4.1.5. Timbre

Antes de definir qué es el timbre y sus propiedades, hay que definir que es el espectro y los diferentes filtros que puede tener una onda de sonido. El espectro de sonido es la representación de un sonido, generalmente se toma una porción de todo el sonido, en términos de la cantidad de vibración en función de cada frecuencia individual. Este es usualmente presentado como una gráfica en la que se muestra ya sea el poder o la presión (eje Y) en función de la frecuencia (eje X). El poder o presión están medidos en decibeles y la frecuencia esta medido en vibraciones por segundo, Hertz (University New South Wales, s.f.). Para medir y graficar correctamente el espectro de sonido se utiliza el espectrograma, el cual es una gráfica en el tiempo que indica la cantidad de energía de un sonido en los diferentes componentes de su frecuencia (armónicos). Esto se hace haciendo un análisis de Fourier en cada armónicos sobre el tiempo.

Ahora bien, habiendo definido qué es un espectrograma y cómo es que se analiza, se puede definir qué es el timbre: una propiedad del sonido la cual permite distinguir un sonido de, por ejemplo, una guitarra de una flauta, aunque estén tocando la misma frecuencia a la misma intensidad. Para poder entender por qué estos dos instrumentos tienen una diferencia

timbral, se debe considerar cómo es que está compuesto un tono musical. Hewitt hace la siguiente analogía: al tocar una cuerda de una guitarra, esta vibra a cierta velocidad, la cual define la frecuencia y por consecuente, el tono de una nota. Sin embargo, la cuerda no solo vibra en toda su longitud, sino que también lo hace a lo largo de todas las fracciones de su largo. A estas fracciones se les llama modos de vibración y cada modo tiene su propia frecuencia característica.

A estos modos también se les llama armónicos, de la misma manera que se discutió en la sección 4.1.2, son los múltiplos enteros de la frecuencia fundamental. Esta frecuencia fundamental es importante ya que es la que determina el tono de una nota. Pero a demás de escuchar la frecuencia fundamental, también se pueden escuchar los demás armónicos en simultáneo, acompañándola. Y, así como su nombre les indica, corresponden a las vibraciones de las fracciones de la cuerda: el segundo armónico corresponde a las vibraciones de las mitades de la cuerda, el tercer armónico corresponde a las terceras partes de la cuerda y así sucesivamente hasta extenderse al infinito, donde cada fracción contribuye con sus propias frecuencias, conformando así el tono musical.

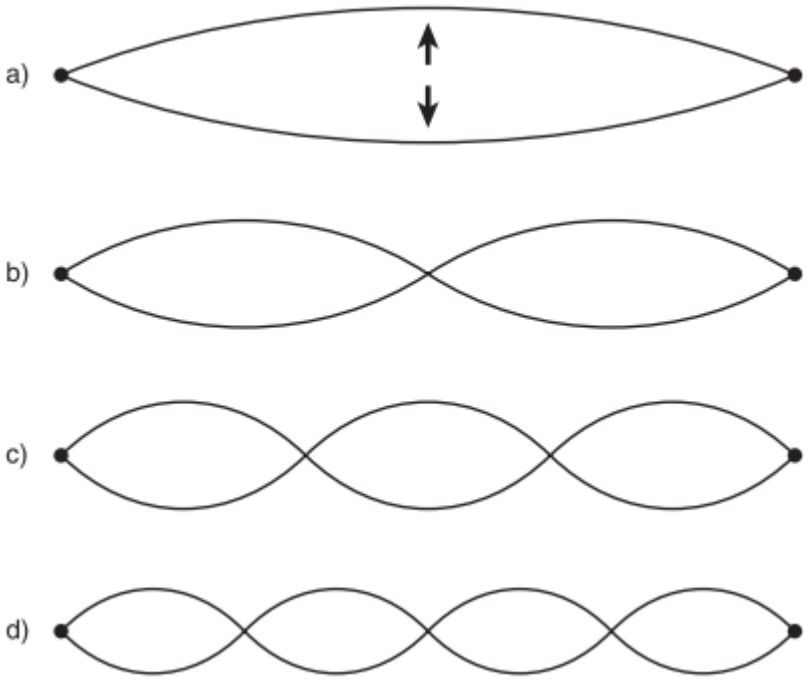


Figura 1. Representación de los primeros cuatro armónicos en una cuerda finita. (Hewitt, 2008).

Así como se ve en la Figura 1, la frecuencia fundamental es representada por la cuerda a, el segundo armónicos por la cuerda b y así sucesivamente. Estos modos contribuyen a la percepción del timbre del tono de dicho instrumento. Es más como una ilusión auditiva igual a la ilusión que crea el cerebro cuando se ve fijamente a un monitor: los pequeños píxeles están separados unos de otros, conformados únicamente por los colores rojo, azul y verde. Pero el cerebro los combina dando así la ilusión de ver colores como el amarillo, morado, blanco, etc. Así mismo, otros instrumentos musicales producen tonos ricos en dichos armónicos, dada la característica mecánica de vibración de dicho instrumento.

Estos armónicos son de gran importancia al momento de componer una melodía, ya que

estos son los que dictan que notas armonizan con otras. En otras palabras, las notas que tienen los mismos armónicos pueden combinarse, así el resultado es armonía en lugar de caos. A estas combinaciones de notas armónicas se les llama acordes.

4.1.6. Síntesis

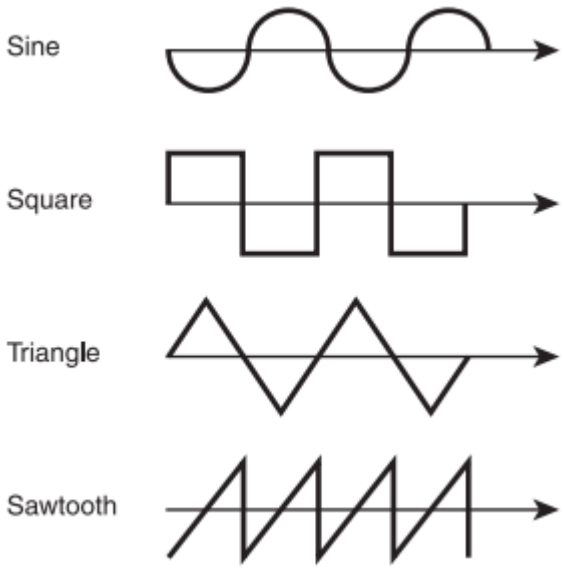


Figura 2. Gráficas de las cuatro formas de onda. (Hewitt, 2008).

Así como se discutió en la sección 4.1.2, el oscilador es el componente de una computadora o sintetizador que produce los sonidos. Sin embargo el oscilador no solo es capaz de reproducir una frecuencia, sino que también es capaz de reproducir sus contenidos armónicos. Esto quiere decir que el oscilador es capaz de producir una nota musical imitando las complejas formas de onda características del timbre de los diferentes tipos de instrumentos musicales, como violines, flautas, pianos, instrumentos de cuerda, de percusión, etc. Esto lo hace con las diferentes formas de onda que puede producir. La Figura 2 muestra las diferentes formas de onda que un oscilador puede sintetizar. En el anexo 10.2 se pueden encontrar los espectros de cada una de las formas de onda.

Así como se discutió con anterioridad, la forma de onda más sencilla es la forma sinusoidal, ya que esta solo contiene un armónicos. La Figura 41 muestra el espectro de una onda sinusoidal sobre la nota do, cinco octavos arriba del centro. Sin embargo también existen otras formas de onda con las que el oscilador puede sintetizar los diferentes instrumentos. Uno de ellos es la forma de onda cuadrada, la cual enfatiza los armónicos impares, para poder dar sonidos vacíos, similares a tonos producidos por bajos. La Figura 42 muestra el espectro de una forma de onda cuadrada.

Al igual que la forma cuadrada de onda, la forma de onda triangular también contiene armónicos impares, pero solo algunos. Esta forma de onda da sonidos más limpios que la cuadrada, por lo que puede imitar el sonido de instrumentos como la flauta. La Figura 43 muestra el espectro de esta forma de onda. Por último, está la forma de onda más compleja:

la forma de onda de sierra. Esta forma de onda contiene casi todos los armónicos de una nota. Esta puede imitar el timbre de instrumentos como instrumentos de cuerda e instrumentos de viento de metal. La Figura 44 muestra el espectro de la onda de sierra. También existe la forma de onda semi-sinusoidal, pero no es considerada una forma de onda básica.

4.1.7. MIDI

Durante la década de 1970 empezó la carrera comercial de los sintetizadores, siendo las compañías "Yamaha", "The Roland Corporation", "Korg", "Oberheim" y "Octave Electronics", entre otros, los pioneros en comercializar sintetizadores análogos y digitales. Estos sintetizadores utilizan voltajes para representar las frecuencias de una nota. Sin embargo, a principios de los 80´s hubo un gran problema a raíz de esta explosión de manufactura, ya que los productos de una compañía eran totalmente incompatibles con los sintetizadores de otra. Esto fue debido a las muy diferentes leyes de voltaje a tono, forzando a los fabricantes a experimentar con interfaces de control digital que trataban de resolver este problema. Sin embargo, debido a la incompatibilidad de dispositivos de control de voltaje, algunos dispositivos representaban las señales correctamente, pero en otros dispositivos había algún tipo de respuesta o incluso no había ninguna.

Dave Smith, representante de "Sequential Circuits", tuvo la iniciativa en plantear la idea de un protocolo estándar, quien junto a Kakhashi, Oberheim y Chet Wood presentaron la propuesta de una Interfaz Universal de Sintetización (USI). Pero en 1982, representantes de varias compañías hicieron mejoras a la propuesta de Wood en la Asociación Nacional de Comerciantes de Música (NAMM). Sin embargo muchos otros representantes protestaron en contra de esta nueva propuesta ya que el protocolo restringía la información que podía llevar: era una simple transmisión de apagado/encendido de notas/eventos ya que debería de ser capaz de transportar más información. En septiembre de ese mismo año, varias compañías japonesas, junto con "Sequential Circuits", finalizaron el borrador del nuevo protocolo final, que llevaría el nombre de "Musical Instrument Digital Interface" (MIDI), en forma de un artículo en la revista "Keyboard", escrito por Robert Moog. y en 1983 en una junta de la NAMM, se hizo una demostración de la idea formalizada de este protocolo y a finales de ese mismo año, muchos de los fabricantes de sintetizadores incluían una interfaz MIDI como un estándar de comunicación. (Manning, 2013)

Como este protocolo es una interfaz de comunicación entre *hardware* y *software* que, al igual que una computadora se comunica con impresoras, discos duros o hasta el internet; MIDI necesita un *hardware* especial para la traducción de datos. Sin embargo, el diseño físico y el cómo la información MIDI es representada e interpretada es cuestión de cada fabricante. En el caso de sintetizadores digitales, la interfaz se requiere únicamente para transferir información de un formato digital a otro. Pero en el caso de sintetizadores análogos, se requiere una incorporación de una traducción de un formato digital a análogo y viceversa, ya que se requiere traducir información MIDI a control de voltajes y viceversa.

MIDI es un protocolo de comunicación en el cual la información es convertida en un solo flujo de bits transmitida entre dispositivos a través de un solo cable. A pesar de esto, el protocolo MIDI tiene sus diferencias con las interfaces de computadora, ya que estas miden interrupciones en un flujo de corriente, mientras que el MIDI detecta pulsos de voltaje.

Pero convertir un modo a otro afortunadamente solo requiere de un circuito sencillo, lo que lo hace capaz de trabajar con una computadora. Pero la adición de aisladores ópticos es necesario para proteger los equipos para evitar bucles de tierra entre equipos o la transmisión accidental de picos altos de voltaje que podrían dañar un dispositivo.

Existen tres tipos de puertos MIDI: MIDI IN, MIDI OUT y MIDI THRU. Los primeros dos se utilizan par mandar o recibir información MIDI, mientras que el puerto MIDI THRU se utiliza para hacer una confección entre un dispositivo a otro. Por ejemplo, conectar tres sintetizadores para realizar una comunicación circular, se necesita este puerto, ya que en la especificación de este protocolo, no señala que es un requisito que un puerto MIDI IN escuche las señales del puerto MIDI OUT.

Por otra parte, la naturaleza de los datos transmitidos por los puertos puede variar con respecto al contexto, ya que existen varios códigos que controlan el funcionamiento de un sintetizador. Estos son usados para transmitir información de una nota o evento individual. Estos códigos consisten en una serie de tres bytes, conteniendo tres ítem de información. El primer byte es conocido como la señal de estado o instrucción es el que indica el inicio de un nuevo mensaje MIDI e identifica su propósito. El segundo identifica el tono de una nota. Este lo hace con un simple número en un rango desde 0 hasta 127, siendo la nota do de la escala natural mayor el número 60 mientras que la nota la de concierto es nueve semitonos arriba, correspondiendo el número 69. El tercer byte está reservado para la información de velocidad del teclado, el cual denota el volumen basado en la proposición que mientras más rápido se toca una nota en el teclado, más fuerte sonará. Es por eso que se necesita cuantificar esta velocidad a un valor en el rango de 0 a 127 (Manning, 2013). Esto funciona tanto para sintetizadores monófonos como para sintetizadores polífonos.

4.1.8. Nota Do y las octavas

A lo largo de un teclado MIDI existen siete notas blancas y cinco notas negras agrupadas en pares y en tríos intercaladas en estas notas blancas. A lo largo de este trabajo se hablará de la escala de do mayor, por practicidad del mismo, por lo que la primer nota blanca a la izquierda del par de notas negras de esta escala es llamada la nota do o C en inglés. Luego le siguen las notas re, mi, fa, sol, la y si; también llamadas D, E, F, G, A, B en inglés. Dependiendo del rango de la cantidad de notas que tenga un teclado, se puede ver que hay múltiples notas do a lo largo de dicho teclado. Esto es así porque, según lo que se discutió en la sección 4.1.2, cada nota genera sus propios armónicos, los cuales son múltiplos enteros del fundamental. Entonces una nota A_1, cual tiene una frecuencia fundamental de 220Hz, su segundo armónico será la nota A_2 con 440Hz. Entonces se dice que la nota A_2 es una octava arriba que la nota A_1. Lo mismo pasa con las siguientes notas. El término "octava" se refiere a que está ocho notas blancas arriba, su segundo armónico. Así como se muestra en la Figura 7, donde se ve la nota do en el teclado junto a su octava. Estas notas se nombran de la misma manera ya que contienen los mismos armónicos y suenan igual; son virtualmente la misma nota.

Las notas negras también forman parte de la escala de do mayor, pero estas son alteraciones de las notas blancas y están un semitono separadas de las blancas y las blancas están separadas por un tono (o dos semitonos), a excepción de los pares de notas E y F así como

B y C . Dependiendo de la nota que se esté tocando, las notas negras obtienen su nombre. Por ejemplo, la nota negra que está entre C y D, si se está tocando C y se sube un semitono obtiene el nombre de C sostenido (representada por $C\sharp$), ó si se está tocando la nota D y se baja un semitono, obtiene el nombre de D bemol ($D\flat$).

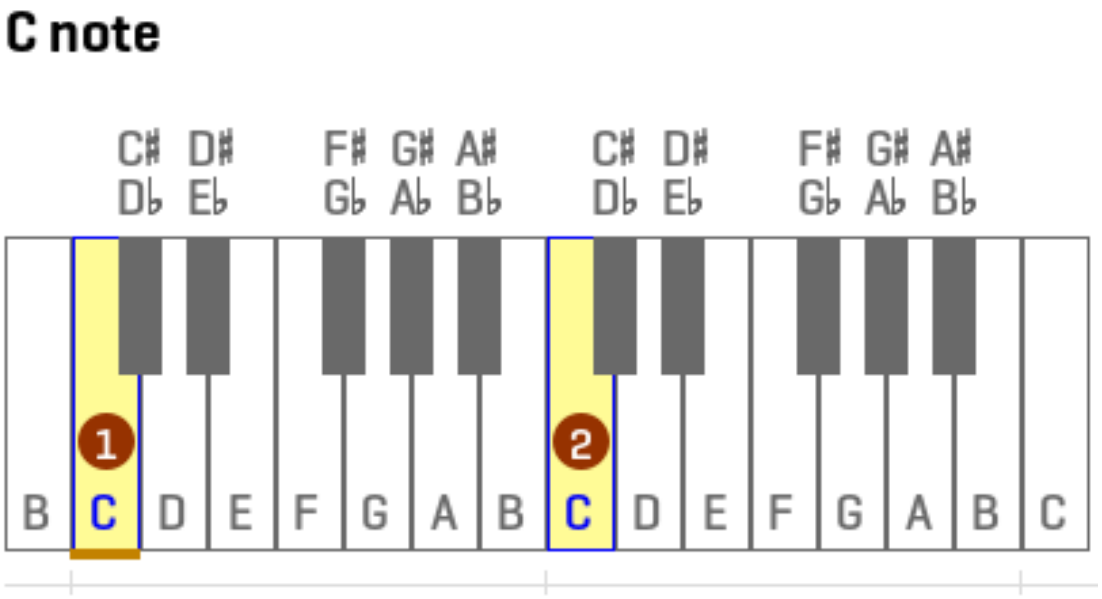


Figura 3. Teclado MIDI con una nota do natural y otra una octava arriba. (Veler Ltd, 2022).

4.1.9. Escalas y acordes

Una escala es una sucesión de notas, generalmente a lo largo de una octava y se repite en las demás octavas. Las notas que se escogen en estas escalas son escogidas con propósito de melodía. Existen diferentes tipos de escalas, sin embargo por el alcance de este trabajo sólo se discutirá sobre las escalas utilizadas dentro de la música occidental: las escalas cromáticas, mayores y menores y las pentatónicas. Cuando se habla de escalas mayores y menores se refiere al sentimiento que las notas quieren expresar, ya que están arregladas de tal manera para que las escalas mayores muestran una sensación de alegría y calidez así como la canción *Imagine* de John Lennon, mientras que las escalas menores muestran sensaciones oscuras y tristes como la canción *Things We Said Today* de "The Beatles".

Cada una de las escalas tiene una nota que define la distancia o grado de las notas y es la que le da nombre a las escala. Por ejemplo, la escala natural de do tiene como nota fundamental la nota C, ya que esta es la que aparece de primero en la escala. Lo mismo pasaría si se cambia la nota fundamental C por D. Las notas de la escala serían D, E, F sostenido, G, A, B y C sostenido; generando así la escala mayor de D. F y C son notas sostenidas ya que esta escala tiene una estructura que es la que le da forma a las distancias entre las notas. Esta estructura es tono, tono, semitono, tono, tono, tono, semitono. Esto quiere decir que la distancia entre la primer y segunda nota son dos notas, entre la tercera y la cuarta son dos notas, entre la cuarta y la quinta es una nota y así sucesivamente. Todas las escalas tienen sus propias formas o intervalos entre notas, lo que le da la característica a dicha escala. Las escalas menores son la contra parte de las mayores: tienen los mismos

intervalos que las mayores, pero estas están generalmente desfasadas un intervalo de notas para que su tercer intervalo sea menor, o una tercera menor entre la primer y tercera nota. Caso contrario de las escalas mayores: tiene una tercera mayor entre la primer y tercera nota.

Para añadirle armonía a una escala, se selecciona un conjunto de dos, tres, cuatro, hasta siete notas, todas pertenecientes a la misma escala pero pueden variar entre octavas. A estos conjuntos de notas se les llama acordes. Estos acordes tienen la característica que comparten ciertos armónicos entre sus notas, lo que las hace sonar de manera agradable y no caen en caos. Se pueden formar acordes de todas las doce notas pertenecientes a una escala y, así como las escalas, cada acorde tiene una nota fundamental que es la primer nota que aparece en este. Esta nota fundamental es la nota que le da la tonalidad al acorde y constituye la referencia para sus intervalos. La manera que se construyen los acordes es seleccionando la primer nota, la fundamental, y contar la tercera y quinta notas relativas con respecto a la fundamental. Otro método para construir los acordes de una escala es por medio del círculo de quintas, como se ve en la Figura 8. De esta manera, si se quiere encontrar el acorde de C en la escala natural de C, se seleccionan las notas C, E, G. El Cuadro 1 muestra una representación de todos los acordes de las notas en la escala de C mayor. Cada acorde es representado por un número romano. El séptimo acorde de la escala es diferente a los otros, ya que este acorde es disonante por lo que se le llama acorde disminuido. No se profundizará sobre este acorde ya que queda fuera del alcance de este trabajo.

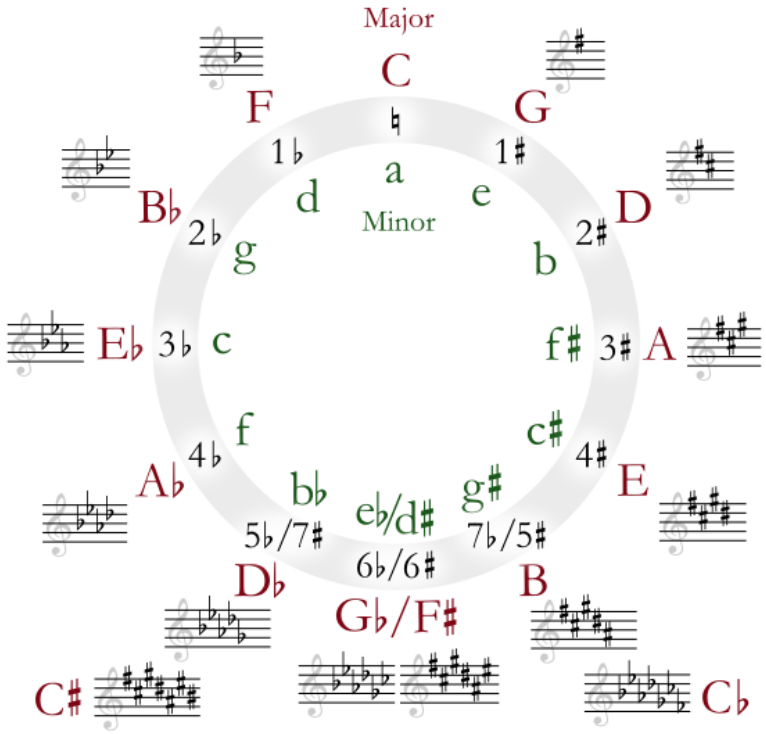


Figura 4. Circulo de quintas para las escalas naturales mayor y menor.

Cuadro 1: Acordes de la escala mayor de C

I	II	III	IV	V	VI	VII
C	**D**	**E**	**F**	**G**	**A**	**B°**
C	D	E	F	G	A	B
E	F	G	A	B	C	D
G	A	B	C	D	E	F

A pesar de que existen varios acordes para una escala, sólo se utilizan unos cuantos de estos acordes para dar así la sensación que el compositor desea. Siendo estos los acordes: I, IV, V y VI (I, IV y V para las escalas de "blues"). Así mismo, una sucesión aleatoria de estos acordes se le llama una progresión y puede variar comúnmente entre 8, 12 y 16 compases.

4.1.10. Duración, ritmo y tempo

Dejando a un lado las frecuencias y los tonos de las notas, cada nota tiene una duración dentro de la dimensión del tiempo. Se han cuantificado las diferentes duraciones que una nota puede tener. Para entender estas duraciones, se necesita comprender qué es un compás, el cual es una métrica musical que mide cuántas unidades de tiempo (figuras musicales) tiene dicho grupo. En palabras simples, es cuántas duraciones de notas puede contener un compás con un límite preestablecido. El límite más comúnmente utilizado es el compás de 4/4.

Tomando como base este compás, la nota que más duración puede tomar es llamada la nota redonda. Esta nota ocupa todo el compás, y es representada por 𝅝. Luego le sigue la nota blanca, que es la mitad de la duración de la redonda y es representada por 𝅗𝅥. Las notas negras son la mitad de las notas blancas, o, 1/4 de la duración de una redonda: ♩. Luego siguen las corcheas que es 1/8 de una nota redonda, ♫, las semicorcheas que son 1/16 de una redonda, 𝅘𝅥𝅯y las fusas que son 1/32, 𝅘𝅥𝅰. Existen otras notas con mayores duraciones que las redondas, así como notas con menor duración que las fusas, pero estas caen en desuso. Así mismo, los silencios acompañan a las notas, cada uno con las mismas duraciones, representados por 𝄻, 𝄼, 𝄽, 𝄾, 𝄿y 𝅀respectivamente. La duración de una nota se puede calcular cuantitativamente ya que cada duración es la mitad de una fracción de una nota redonda, siendo la fracción un exponente de dos; teniendo la siguiente ecuación:

$$d = \frac{1}{2^n}$$

Siendo 2^n la fracción de una nota redonda.

Una sucesión de duraciones de notas y silencios, con cierta repetición es llamado el ritmo. Este es fundamental dentro de la música, ya que si no existiera el ritmo, no existiría la música. Sin embargo hay un fenómeno que indica el largo de una de las notas anteriormente mencionadas y, por ende, el largo de un ritmo. Este fenómeno se conoce como el tempo, el cual es la velocidad en que una nota se pulsa. Es el determinante que indica cuántas pulsaciones ocurren por minuto, o en palabras más sencillas, cuántas notas negras se tocan por minuto.

4.2. Generación procedural

4.2.1. ¿Qué es la generación procedural?

Para comprender qué es la generación procedural de contenidos (de ahora en adelante PCG, por sus siglas en ingles), Dale Green utiliza la analogía de comparar la carga de algún archivo multimedia (ya sea alguna imagen, audio o vídeo) desde almacenamiento a memoria, con la compra de algún mueble (una mesa, silla, etc.). Cuando se carga alguno de estos archivos, la computadora los carga a memoria como un todo. Tal vez carga los vídeos por pedazos más cortos, pero ese archivo ya existe como un todo dentro de almacenamiento. Lo mismo pasa cuando se compra algún escritorio o silla: obtenemos un mueble como un todo. Pero ahora que pasa si en vez de comprar un mueble previamente armado, sino uno que viene por piezas e instrucciones para armar en casa. En vez de obtener un mueble como un todo, se obtienen las piezas por separado las cuales, con ayuda de las instrucciones, se puede armar. O incluso, se podría armar agregándole otras piezas o instrucciones deseadas para darle un toque mas personalizado. (Green, 2016)

Ahora bien, en el contexto de la computación, estos procesos de crear algo a partir de procedimientos o algoritmos es llamado como generación procedural. La mesa fue creada a partir de seguir el algoritmo que dictaban las instrucciones. Este mismo principio puede ser aplicado para cualquier campo: creación de niveles de videojuegos, imágenes, musica, vídeos, texto, etc. Por lo tanto podemos definir la PCG de contenido como el proceso de crear dicho contenido con algoritmos. Estos algoritmos, si no son modificados, siempre darán el mismo resultado a partir del mismo input. Esto se debe a que las computadoras son deterministas. Sin embargo, la PCG no es inherentemente aleatoria. Se necesita de un elemento aleatorio para que el contenido sea diferente y dinámico en cada generación.

4.2.2. Generación pseudoaleatoria

Números pseudoaleatorios

Con números aleatorios, cada generación procedural puede añadir un elemento impredecible a su resultado, lo que lo hace una experiencia única. Sin embargo, es imposible que una computadora genere números completamente aleatorios. Esto es por el mismo principio determinista que se mencionó con anterioridad: con el mismo algoritmo, dos computadoras retornarán el mismo resultado con el mismo input, por lo que no se puede esperar que actúen de manera impredecible (Green, 2016). Si este fuera el caso, las computadoras necesitarían de *hardware* especial, junto con un gran poder de procesamiento, ya que es muy complejo de simularlo. Es por eso que las computadoras solo pueden generar números pseudoaleatorios, lo que significa que no son aleatorios completamente. Estos son el resultado de ecuaciones matemáticas y algoritmos complejos que calculan cadenas de números y pueden ser resueltos con anterioridad. Esto los hace tener un buen rendimiento al momento de calcular un número y no depender tanto del *hardware* (Watkins, 2016). Pero cada uno de estos generadores esta construido diferente. Unos son mas complejos que otros, lo que los hace perfectos para tareas simples como videojuegos y simulaciones sencillas. Pero algoritmos más complejos tienen usos en la criptografía, donde el output no puede ser dado por patrones.

Semillas

Así mismo, cada uno de estos generadores puede utilizar semillas, "seeds" para generar una secuencia de estas cadenas, que eventualmente se repiten. Estas semillas pueden ser impuestas por el usuario o por otros medios pseudoaleatorios como por ejemplo, utilizar la fecha y hora actuales de la computadora. Una semilla es, en otras palabras, el punto de inicio de un algoritmo. Son utilizadas principalmente para, por ejemplo, recrear una secuencia que se utilizó con anterioridad y se borró de memoria. Pero así como se mencionó, estas semillas pueden repetirse. Para evitar este problema, se tienen que tomar en cuenta múltiples factores, como el tamaño de la semilla, los valores y el rango de la secuencia. De esta manera es que se puede dar ese elemento impredecible a la PCG, ya que los números pseudoaleatorios son los que dirigen las decisiones. Es imprescindible minimizar las decisiones de detalles menores, así como generar bosques colocando arboles uno por uno aleatoriamente, tarea que gastaría mucho tiempo si se hace manualmente.

4.2.3. Uso y aplicaciones de la generación procedural de contenido

Habiendo conocido los conceptos de la PCG y la aleatoriedad que existe en ellos para crear entornos dinámicos y únicos, se debe de saber para qué se utiliza esta generación pseudoaleatoria. A continuación se concretaran las formas más comunes de su utilización.

Espacio y memoria

La principal de las razones por las que se utiliza la PCG es por su robustez, adaptabilidad y tamaño. Este último es uno de los mayores beneficios de la PCG. Se puede tomar como ejemplo el videojuego "Rogue". En la época que salió este videojuego (1980), las computadoras no eran tan potentes a nivel de almacenamiento de memoria y procesamiento. Sin embargo, estas computadoras solo necesitaban utilizar la cantidad de procesamiento y memoria necesaria para cargar los niveles (en ASCII) de "Rogue", ya que estos eran calculados algorítmicamente en vez de directamente cargados a memoria (Watkins, 2016). Esto hacía el juego muy ligero. Siguiendo la analogía de Green, esto haría que la caja donde viene la mesa fuera mucho más ligera y fácil de transportar.

Arte visual

Otro de los usos mas prominentes de la utilización de PCG es la creación de mapas y terrenos dentro de los videojuegos. Se utilizan algoritmos junto con la función matemática del Ruido de Perlin como factor aleatorio para generar mapas sencillos en 2D para mundos en 3D. Así mismo, similar a la generación de mapas, la PCG tiene dentro de sus usos la creación de texturas. Estas también utilizan ruido, como el Ruido de Perlin, pero utilizando diferentes patrones y ecuaciones que controlan mejor ese ruido para crear patrones reconocibles. (Green, 2016)

Arte musical

Así como se mencionó que la PCG puede utilizarse para crear imágenes, las cuales son una forma de arte, la generación procedural también puede combinarse con otro tipo de arte: sonidos y la música. A pesar de que es poco común, puede utilizarse para crear

sonidos dentro de los videojuegos, con la manipulación de sonidos existentes, ya que se pueden espacializar. Sin embargo, también se puede crear musica procedural, pero este proceso requiere de muchos conocimientos teóricos para generarlos proceduralmente. No solo se requiere saber cómo generar contenido por medio de procedimientos aleatorios, sino que también se necesita saber de teoría de la música para poder aplicarlos en este campo para poder generar una melodía coherente.

CAPÍTULO 5

Metodología

5.1. Herramientas utilizadas

5.1.1. Suite for Computer-Assisted Music in Python (SCAMP)

Para la construcción del prototipo se utilizó el lenguaje de programación Python, en la versión 3.10.7, junto con la librería random nativa. Se eligió utilizar este lenguaje de programación ya que es versátil y tiene una gran variedad de librerías open source hechas por la comunidad para la comunidad. Una de las librerías utilizadas que se seleccionó es la librería "Suite for Computer-Assisted Music in Python" (SCAMP). Esta librería, así como su creador la describe, es un marco de trabajo que tiene como finalidad actuar como un centro de control que conecta al compositor con otros recursos para la creación de pistas musicales, siempre manteniendo un modelo mental tan abierto como sea posible. Sin embargo, SCAMP no ofrece utilidades, como kits de herramientas generativos o modelos escalares/armónicos, puesto que el compositor las tiene que importar de herramientas de terceros o él mismo las tiene que crear. Este marco de trabajo es muy sencillo de utilizar, ya que cuenta con un objeto principal llamado Session, el cual es como una estación de trabajo de audio digital (DAW en inglés) ya que cuenta con un pequeño sintetizador MIDI que lleva la cuenta de diferentes instrumentos construidos internamente al igual que permite transmitir notas MIDI a un DAW a través de un cable MIDI; permite manejar el flujo del tiempo ya que el objeto "Session" es un objeto que hereda del objeto "Clock" de Python permitiendo manejar el tiempo y ajustar tempos, así como permite grabar audio y transcribir las notas reproducidas por los diferentes instrumentos, guardarlas e imprimirlas en una partitura. (Evanstein, 2019)

Así mismo, SCAMP se inicia asignando el objeto "Session" a una variable y desde esa variable se puede trabajar lo anteriormente mencionado. Para asignar un instrumento, se asigna el atributo `new_part()` de este objeto a una variable, recibiendo el nombre del instrumento entre comillas. Así mismo también se puede asignar un instrumento para transportar

información MIDI para posterior usa a un DAW mediante un cable MIDI, asignando el atributo `new_midi_part()`, recibiendo como parámetros un nombre descriptivo para el instrumento y el número de puerto que transportará información. Los puertos que se utilizan para cada instrumento son: puerto 1 para los instrumentos de batería, puerto 2 para el bajo y puerto 3 para la guitarra. Para hacer que SCAMP reproduzca una nota, se hace por medio del atributo `play_note()` pero desde la variable donde se asignó el instrumento, recibiendo tres parámetros: la nota en número MIDI, el volumen en decimales entre 0 a 1 y la duración en decimales entre 0 y 1. Para reproducir un acorde, se utiliza el atributo `play_chord()` que recibe los mismos parámetros pero recibiendo una lista de notas en vez de sólo un número. Para reproducir múltiples instrumentos a la vez, se hace por medio del atributo `fork()` de "Session", y para imprimir la partitura se hace por medio del atributo `stop_transcribing().to_score().show()`. Es por estas ventajas, versatilidad y sencillez de utilización que se escogió SCAMP como marco de trabajo para el prototipo de este trabajo. Así mismo, se utilizó la librería TKinter de Python para la creación de la pequeña interfaz gráfica.

5.1.2. Estación de Trabajo de Audio Digital (DAW)

Para complementar lo que se hizo con SCAMP, se decidió trabajar con un DAW profesional, ya que estos permiten el uso de sintetizadores de múltiples instrumentos integrados, al igual que la instalación de "plugins", así como "plugins" para instrumentos y grabar audio. Se escogió el DAW "Ableton Live", ya que este permite todo esto y recibir entradas MIDI. Adicionalmente, se instaló, a "Live", el "plugin" en formato VST2 la batería "MT PowerDrumKit" para sintetizar las notas de cada instrumento de batería generadas. Así mismo, se instaló el "plugin" de amplificador "Emissary" de "Ignite Amps" junto con "NadIR", el cual es un convólver de respuesta de impulso dual (IR) de latencia cero, diseñado para usarse como un simulador de gabinete para amplificadores de guitarra y bajo (Ignite Amps, 2019).

5.2. El algoritmo

El algoritmo creado en este trabajo es un algoritmo propio, el cual se trabajó en diferentes etapas: estructura de la canción, ritmo fundamental, intro, versos, coros y outro. Este algoritmo genera pistas musicales totalmente nuevas, proceduralmente, ya que se utiliza una serie de reglas para que las partes y notas generadas aleatoriamente tengan una estructura musicalmente coherente, así como lo explica Dennis Martensson en su video "How I Procedurally Generate Djent Music" (https://youtu.be/p-lNcpb5ZwE), de quien se tomó inspiración para estructurar las canciones. La Figura 5 muestra un diagrama de flujo de cómo se comporta el algoritmo al momento de generar una canción. El género de las pistas que este algoritmo genera es el metal (subgénero del rock) pero se realizaron varias encuestas sobre las sensaciones que debería de generar sobre las pistas para poder trabajarlas en videojuegos. Las sensaciones o "moods" que más se utilizan en la creación de videojuegos, pertenecientes y adaptables al metal, son: "mood" agresivo para una escena e acción, "mood" melancólico para una escena triste o melancólica y un "mood" épico para dar ese sentimiento de alegría épica para escenas de acción o una escena más alegre. Este algoritmo es adaptable a los tres "moods" ya que estos llevan la misma estructura de generación gracias a que pertenecen al mismo género musical. Solo se deberá especificar el "mood" deseado y el algoritmo trabaja

sobre parámetros para acomodar las sensaciones.

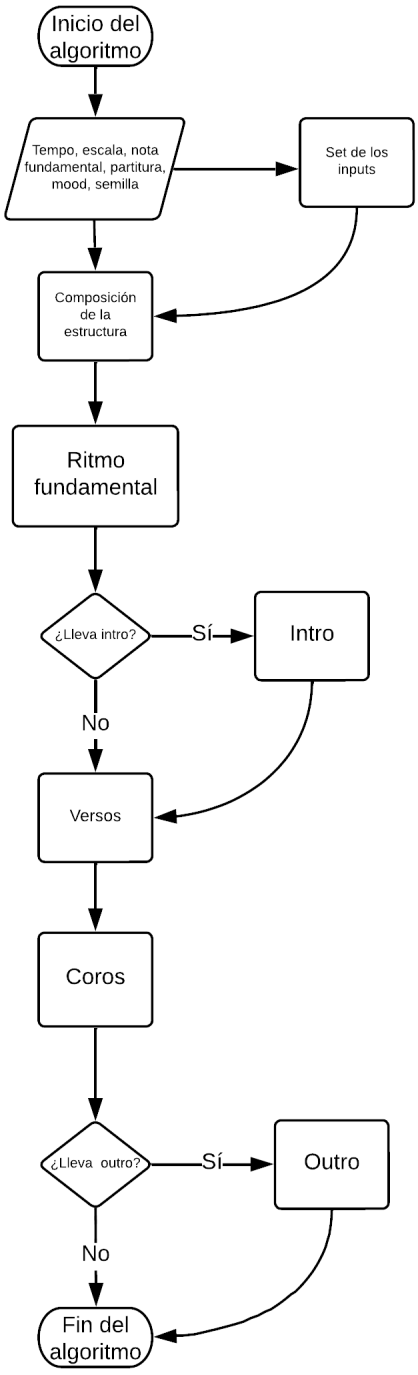


Figura 5. Diagrama de flujo de la generación de una canción.

Para la generación de las semillas, se aleatoriza un número con ayuda del atributo "maxsize()" del módulo sys de Python. Este atributo señala el valor máximo que un tipo

de dato “Py_ssize_t” puede contener. Es decir, es el tamaño máximo de objetos “string” y listas que el puntero de Python puede llegar a asignar. Este número es dependiente de la arquitectura de la plataforma en la que se esté corriendo el programa. Luego de obtener este número máximo, se divide dentro de 8000, para así obtener un número con un máximo de 15 caracteres, con el fin de no generar un “overload” al momento de correr el programa en arquitecturas no tan potentes.

Para la síntesis de los instrumentos de percusión, se utilizó el estándar MIDI para cada uno de estos instrumentos. Esto quiere decir que un único número MIDI está mapeado para un sólo instrumento. Este estándar se puede representar por la siguiente tabla:

Número MIDI	Instrumento
36	Bass Drum
37	Snare Rim
38	Snare
41	Floor Tom
42	Closed Hi Hat
44	Hi Hat Foot
45	Low Tom
48	High Tom
46	Open Hi Hat
49	Crash
51	Ride Body
52	China Cymbal
53	Ride Bell
55	Splash Cymbal
58	Crash + Hold

Cuadro 2: Mapeo de números MIDI a instrumentos de batería.

5.2.1. Estructura de la canción

Esta es la fase en la que se decide qué partes va a llevar la nueva pista. Hay que recordar las partes más comunes que una canción puede llevar: introducción, primer verso, coro, segundo verso, segundo coro y algunas veces llevan outro. Esta no es una estructura estándar para componer una canción, ya que estas partes pueden variar, dependiendo del compositor, o incluso se pueden hacer modificaciones y añadir nuevas partes. De este modo se crea la estructura de la canción a generar. De primero el algoritmo decide con una probabilidad del 50 % si la canción debe de llevar una introducción. Caso contrario, solo omite añadir esta parte. Luego viene el primer verso, obligatorio, al igual que el primero coro. El segundo verso tiene la característica que puede ser el mismo que el primer verso, o, una variante diferente con otro ritmo parecido. Lo mismo pasa con el segundo coro: puede se igual que el primer coro, así como una variante más melódica o puede ser completamente diferente al primer coro. Por último, existe una probabilidad del 50 % de que el algoritmo genere un outro si el “mood” es agresivo.

5.2.2. Ritmo fundamental

Esta es la parte más importante de la canción, ya que es la parte que define el ritmo tanto de los instrumentos de percusión, como los instrumentos de cuerda. Para generar este ritmo se generan números enteros aleatorios entre 2 y 5 para el "mood" agresivo, 2 y 4 para el "mood" épico al igual que para el "mood" melancólico. Cada uno de estos números servirá para generar la duración de cada una de las notas, siendo estos exponentes de dos, así como se discutió en la sección 4.1.10. Luego, para normalizar las notas a un número que SCAMP pueda leer, esta fracción se divide dentro de 0.25. SCAMP puede leer cualquier número con punto flotante como parámetro de duración de una nota al momento de reproducirla, pero se trabajó de esta manera para estandarizar las duraciones de las notas y apegarse lo más posible al alcance del proyecto que es generar canciones proceduralmente con las normas de la música occidental.

Luego de generar una nota, se verifica esta duración y añade el número de estas mismas notas faltantes para completar una nota negra: si genera una corchea se añade otra corchea, si genera una semicorchea se añaden otras tres y así sucesivamente para que estas notas den una suma de una negra. La aleatorización de las notas se hace por cada compás hasta que este de una suma de cuatro negras entre todos sus compases. Este proceso se hace compás por compás hasta que se llega al límite de cuatro compases que luego se copian y pegan para formar ocho compases para generar repetición dentro del ritmo fundamental de la canción. Este ritmo es el que el bombo y la guitarra deben de seguir, ya que el bombo es el que acentúa las notas de la guitarra dentro del género del metal, por lo que se copia y se guarda para cada uno de estos instrumentos, incluido el bajo. Sin embargo, se reducen las fusas del ritmo de los instrumentos de cuerda para que el bombo tenga una acentuación más notable.

Después de guardar el ritmo en estos instrumentos, se genera una decisión con un 50 % de probabilidad para que el algoritmo genere el ritmo que debería de seguir el redoblante o "snare": un valor de 0 para un redoblante en doble tiempo (se toca una vez cada dos negras) o un valor de 1 para un redoblante a un tiempo (cada tercer negra del compás). Esto le da una probabilidad de que el ritmo fundamental se escuche más o menos energético, así como más o menos pesado. Para la generación de los platillos de contratiempos, "hi-hat", simplemente se tocará cuatro veces por compás o, cada negra. El remate o "cimbal", se silencia a las mismas notas que el contratiempo.

Luego de esto se decide, de manera aleatoria la escala y la progresión de acordes que deben de llevar los instrumentos de cuerda. Para la escala, se verifica cuál es el "mood" que el usuario desea. Para "mood" épico, se escogen al azar las escalas mayores y para un "mood" agresivo o melancólico se escogen las escalas menores. Luego, dependiendo de la escala escogida, se generan los acordes para cada una de las notas siguiendo la estructura de los acorde: para una nota tónica, se escoge la tercera y la quinta de dicha nota. Luego se elige una estructura para la progresión de los acordes, dependiendo de la escala escogida, y se aleatoriza esta estructura con las siguientes reglas para cada uno de los ocho compases del ritmo fundamental: para el primer y último compás, se debe de tocar el primer acorde. Para los compases 2 a 4, existe un 50 % de probabilidad de que se escoja el primer acorde de la progresión y un 25 % de probabilidad de escoger la segunda posición de la progresión (si es que existe) y otro 25 % de probabilidad para escoger la tercer posición (si existe). Para los compases 5 a 7, se escoge de manera uniforme las posiciones de la progresión. En caso de que la estructura de la progresión sólo tenga un valor, la progresión únicamente será de

ese mismo valor para todos los compases.

5.2.3. Intro

Para la introducción, si la decisión del algoritmo lo permite, puede dividirse en dos decisiones: un 50 % para que mantenga el ritmo fundamental, dando una pequeña introducción un poco mas melódica de qué tratará la canción, reduciendo las notas de los instrumentos de cuerda, pero manteniendo el ritmo del bombo. Y un 50 % de probabilidad de que la otra decisión sea una introducción de tipo ambience, donde solo se tocará la guitarra, utilizando el ritmo fundamental como base y silenciando los demás instrumentos, siendo el este ritmo reducido para generar la mayor cantidad de notas blancas posible y asimilar el ambience.

5.2.4. Versos

Para generar los versos, de primero se llama una función de decisión de la estructura del verso. Esta función, primero, escoge de manera aleatoria cuatro posibles ritmos de batería previamente estructurados como "presets" y los añade a una lista. Estos ritmos son ritmos comúnmente escuchados dentro del metal, que se "grabaron" previamente para darle cambios de ritmos a los versos y así evitar la monotonía de escuchar un solo ritmo y hacer que el oyente muestre interés al haber una variación y así "romper" las reglas de la composición, que también pueden agregarse más o modificar los existentes. Esta es una técnica comúnmente utilizada dentro de la generación procedural ya que no todo tiene que ser aleatorio. Pueden haber pequeñas partes ya construidas que el programa o algoritmo puede seleccionar para evitar complejidades y darle una sensación de orden a la aleatorización. Un ejemplo de esto es en el videojuego "Minecraft": el mundo y terreno son completamente aleatorizados, pero se crearon estructuras complejas como casas y otras edificaciones con anterioridad ya que estas estructuras son muy complejas de programar para que el algoritmo las genere desde cero, pero al tenerlas como "presets" sólo requiere de una ubicación compatible con dicho edificio y el algoritmo sólo se encarga de ponerlo en su lugar.

Luego de seleccionar los posibles nuevos ritmos, se seleccionan de manera aleatoria estos ritmos, junto con el ritmo fundamental, para así estructurar el verso. Luego de tener la estructura, se observa el largo de esta estructura para buscar repetición: si el largo es mayor a ocho ritmos, no habrá repetición pero si es menor a ocho ritmos, puede existir un 33 % de probabilidad de que se repitan estos ritmos de manera secuencial, otro 33 % de probabilidad de repetir estos ritmos pero en un orden inverso y otra probabilidad del 33 % que no haya repetición. Luego el algoritmo itera sobre la estructura y añade cada uno de los instrumentos. Se revisa que el número de compases y se ajustan para que este número sea un múltiplo de ocho entre 16 y 32 compases ya que se está trabajando con una progresión aleatoria de ocho compases, que se discutió en la sección 5.2.2. Para el ritmo de los instrumentos de cuerda, si el "mood" seleccionado es agresivo, se revisa un booleano para un cambio de ritmo y, si este es verdadero, se selecciona uno de dos tipos de ritmos de "power chords": uno intermitente en los compases 1, 2, 5 y 6 (representados con tres semicorcheas y un silencio de corchea, en todo el compás); y otro más estable en los compases 1, 2, 3 y 4 (representados por corcheas en todo el compás).

5.2.5. Coros

Los coros están formados por una decisión principal entre tres posibles resultados: un coro con un ritmo totalmente nuevo, un coro que mantiene el ritmo fundamental o un coro que es una modificación del fundamental para darle un toque más melódico. Si la decisión fue un ritmo totalmente nuevo, se llama a la función que genera el ritmo fundamental pero pasándole un parámetro nuevo que indica que la parte de la canción que la está llamando es el coro. Este parámetro modifica los exponentes de la duración de las notas para generar desde notas negras hasta generar corcheas, o en caso de un "mood" melódico, generar desde notas blancas hasta corcheas. Luego de haber generado el nuevo ritmo, se reducen las notas de los instrumentos de cuerda para el "mood" agresivo. Esta reducción hace que todos los pares de una misma nota se formen en una sola nota con un grado de duración mayor. Por ejemplo, dos semicorcheas formarán una corchea y así sucesivamente. Después de haber reducido estas notas, se toman los acordes tónicos de la progresión para la melodía con la guitarra. Por otro lado, si la decisión del ritmo es mantener el ritmo fundamental, únicamente se reducen las notas de la guitarra y el bajo y se mantiene el ritmo del bombo. Si la decisión es un ritmo melódico, se reducen tanto las notas de los instrumentos de cuerda como del bombo. Para las tres decisiones se mantienen los acordes tónicos de la progresión para la melodía. Para los "moods" épico y melancólico, se escogieron compases con corcheas ya que los tempos en estos "moods" no son tan rápidos como en el agresivo, por lo que podrían sonar muy blandas las notas fundamentales.

5.2.6. Outro

Para la generación del outro, si es que el algoritmo escogió esta opción, se utiliza una técnica que se encuentra en canciones tanto del metal y del punk (subgéneros agresivos de los mismos) llamada como "breakdown", y se utiliza para evitar la composición tradicional de verso-coro-verso. Este es un bajón en el tempo de la canción para dar una sensación más agresiva y pesada en los instrumentos. Para dar esta sensación, de primero, se copian los ritmos y notas de cada instrumento pertenecientes al primer verso, ya que comúnmente se toma esta parte para hacer el "breakdown". Luego se modifican los primeros dos compases de cada instrumento para silenciarlos exceptuando la guitarra, a la cual se le añaden dos notas redondas, una en cada uno de estos dos compases: en el primero para reproducir en todo el compás el último acorde que se tocó en el coro y en el segundo compás para silenciar la guitarra. Se hace esta modificación para añadir suspenso para que el oyente esté al pendiente de la canción y, así, sorprenderlo con esta técnica. Luego se acorta el largo de los compases a la mitad de los compases del verso. De esta manera, al momento de reproducir la canción, el reproductor podrá saber cuándo bajar el tempo, ya que el indicador puede ser cualquiera de las dos notas redondas de la guitarra. Esta parte de la canción será la mitad del total de compases que el primer verso.

5.2.7. Instrumentos de cuerda

Para generar las notas de los instrumentos de cuerda, guitarra y bajo, se trabajan con funciones en común entre todas las partes de la canción. Es decir, tanto la introducción, versos y coros llaman a estas funciones para generar las notas y acordes. Para generar las

notas del bajo, se itera sobre el ritmo que se desea dependiendo de la parte de la canción. Luego se inicia un contador para contar los compases que se están iterando, que corresponden a los compases de la progresión anteriormente generada. Luego de esto, se añade la nota tónica del acorde de la posición respectiva al compás de la progresión. Para los compases 2 y 6, en vez de la tónica, se añade su tercera. Y para los compases 3 y 7, existe una probabilidad del 50 % en que se añada la tónica o la quinta. Esto a cada una de las notas de los compases del ritmo del bajo.

Para las notas y acordes de la guitarra, de igual manera que para el bajo, se itera sobre las notas por cada compás del ritmo de la guitarra. También se inicia un contador de compases para llevar un control de la progresión. Sin embargo, aquí es donde termina la similitud con el bajo, ya que cuando se encuentra una semicorchea o una fusa, se añade el "power chord" (la tónica y su quinta) del primer acorde a dicha nota. Caso contrario, se añade una nota aleatoria dentro de las tres notas que tiene el acorde correspondiente al compás de la progresión (ya sea la tónica, su tercera o su quinta). Para el caso de que se estén construyendo las notas para los coros, en vez añadir un "power chord" o una sola nota aleatoria, se añaden los acordes enteros para así hacer los coros más melódicos, trabajando con acordes.

5.3. Hardware utilizado

Para obtener los resultados de esta herramienta musical, se utilizó la computadora con las siguientes especificaciones de *hardware*:

- Procesador: Intel Core i7 11700K 16CPUs 3.6GHz
- RAM: 32GB 2400 MHz
- 1.74GB SSD
- Motherboard: TUF Gaming Z590-Plus WiFi
- Controlador de Audio: THX Spacial - Synapse

CAPÍTULO 6

Resultados y discusión

6.1. Estructura de las canciones generadas

Los resultados obtenidos del programa creado para este proyecto se guardan en archivos en formato JSON, ya que son dos diccionarios: uno para los instrumentos de cuerda y otro para los instrumentos de percusión. Estos a su vez, están divididos en otros diccionarios que guardan la duración de cada instrumentos y, en el caso de la guitarra y el bajo, guardan también las notas musicales de cada uno de estos. Estos resultados, que se mostrarán a continuación, son acerca de los tres "moods" que se mencionaron con anterioridad: agresivo, épico y melancólico. Se utilizó la semilla 4875434493137809954 para generar una canción por "mood". Se utilizó esta única semilla para los tres "moods" para así poder tener un punto de comparación entre las tres canciones y verificar si tienen similitudes en las notas y sus duraciones. La escala generada por esta semilla, para los "moods" agresivo y melancólico, fue la escala de "blues" menor, ya que estos dos comparten las escalas menores, en el modo de A0, pero una octava más baja que A1 (natural). Para el "mood" épico, se generó la escala de "blues" mayor con el mismo modo en A0.

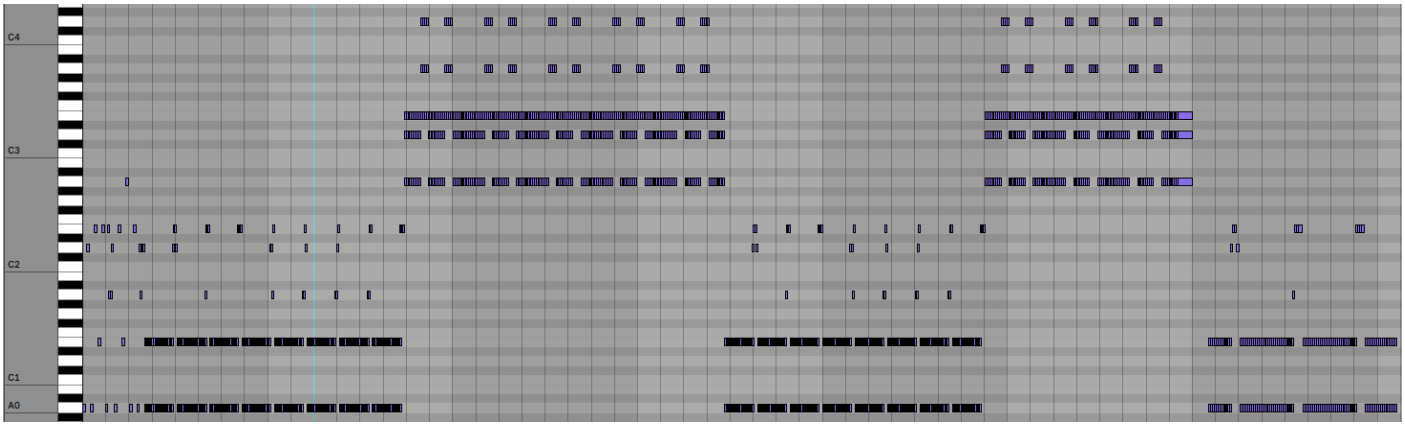


Figura 6. Notas de guitarra para el “mood” agresivo.

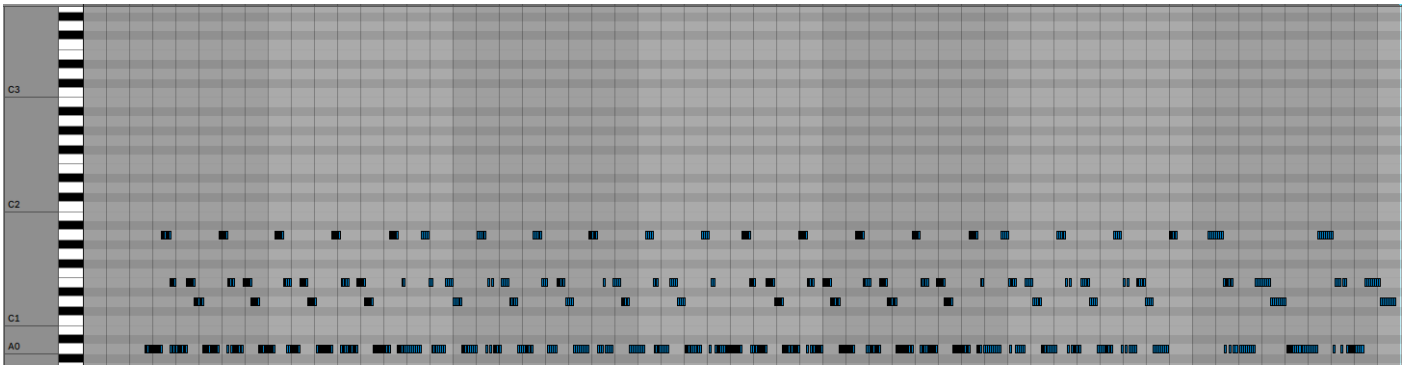


Figura 7. Notas de bajo para el “mood” agresivo.

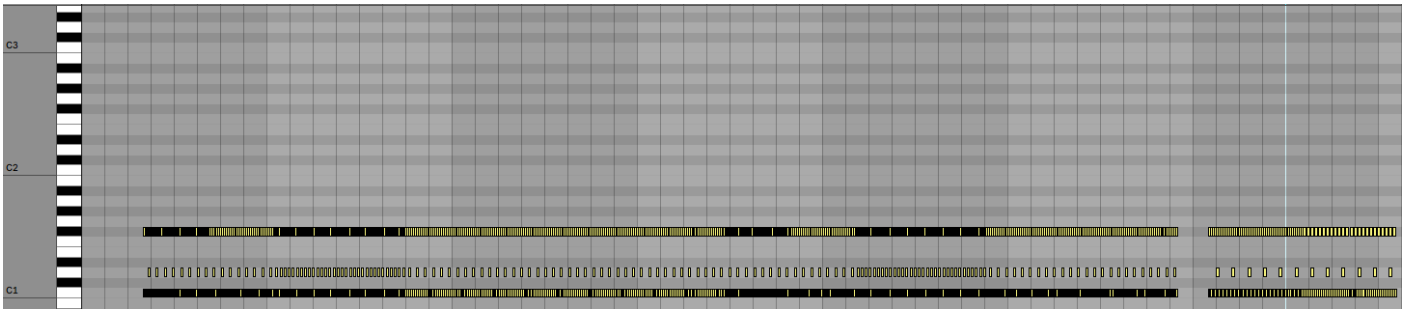


Figura 8. Notas de batería para el “mood” agresivo.

Dentro de “Ableton Live” se escogieron los colores morado para representar las notas de las guitarras, azul para las notas de los bajos y café para las notas de las baterías en los tres “moods”. Para poder identificar las notas, Live las representa por octavas de C, siendo A0 el tono fundamental en los tres “moods” (la nota que aparece hasta abajo) y se pueden identificar las siguientes octavas de C junto a las teclas de piano correspondientes. Y para las duraciones de las notas, mientras más coloridas se ven las notas en las figuras, significa que esas notas duran más tiempo: son notas blancas, negras o corcheas; mientras que pasa lo contrario con las semicorcheas y las fusas, que se ven líneas más os-

curas ya que estas están más juntas. Por lo tanto, las figuras 6, 7 y 8 corresponden a la guitarra, bajo y batería (siendo la línea de arriba las notas para el contratiempo, la del medio para el redoblante y la última para el bombo), respectivamente, para el "mood" agresivo (pista reproducible desde https://on.soundcloud.com/ypKMq) . Las figuras 9, 10 y 11 corresponden a los instrumentos del "mood" melancólico (pista reproducible desde https://on.soundcloud.com/A5F3t) y las figuras 12, 13 y 14 corresponden al "mood" épico (reproducible desde https://on.soundcloud.com/Apgu5).

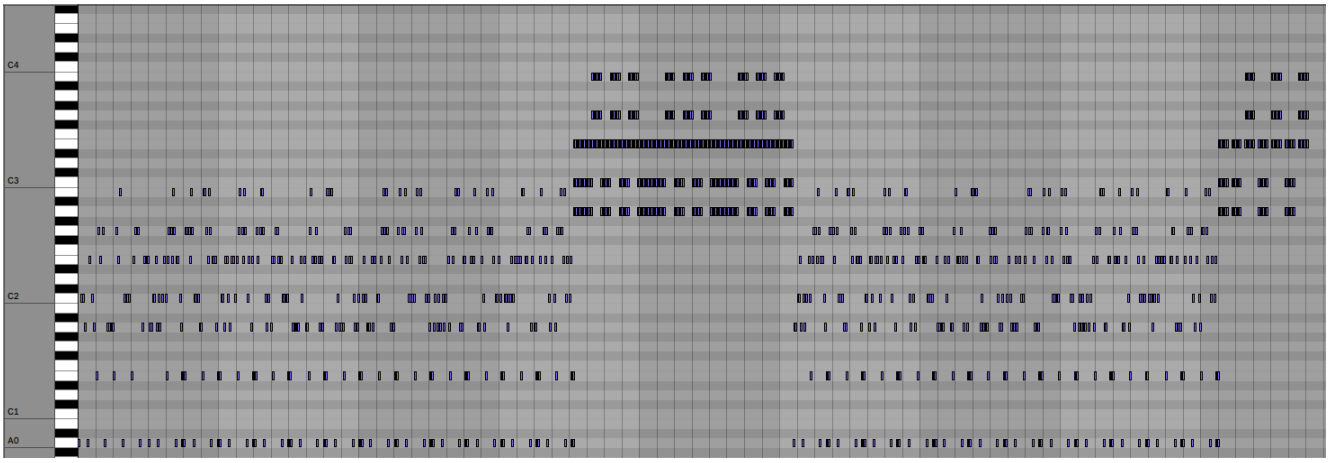


Figura 9. Notas de guitarra para el "mood" melancólico.

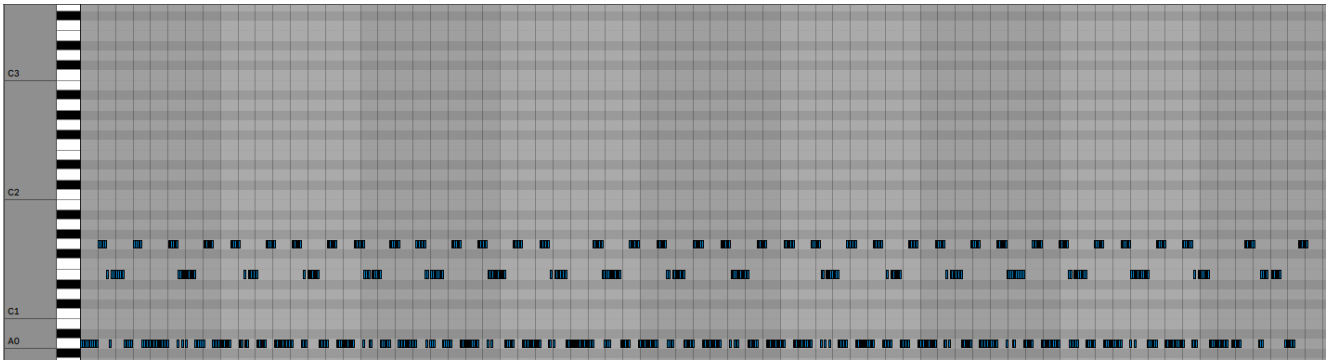


Figura 10. Notas de bajo para el "mood" melancólico.

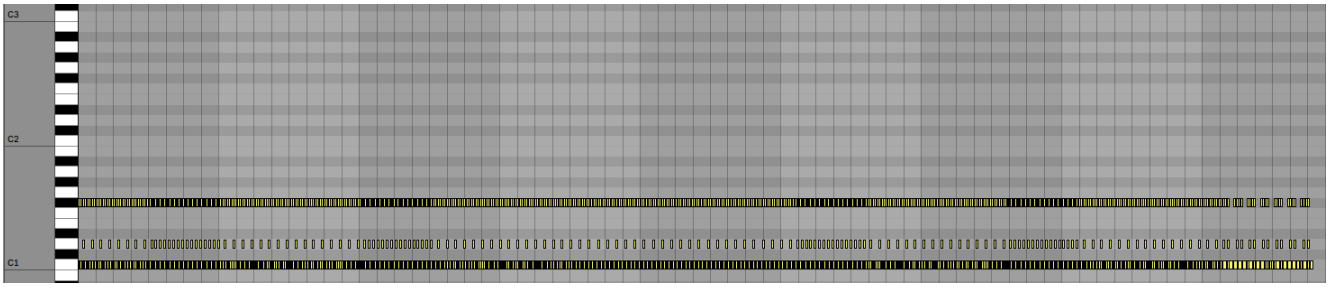


Figura 11. Notas de batería para el "mood" melancólico.

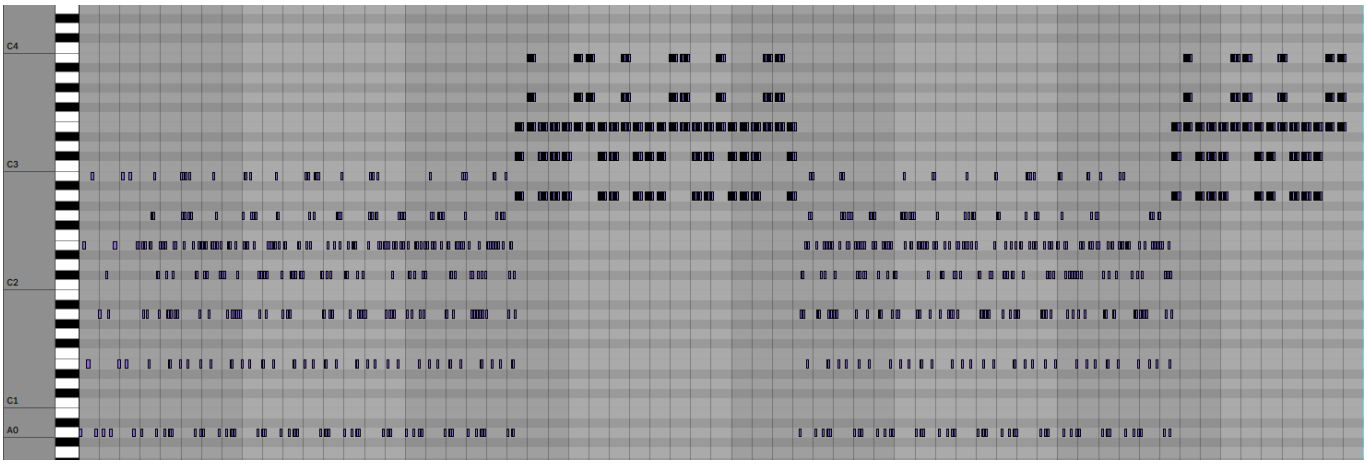


Figura 12. Notas de guitarra para el “mood” épico.

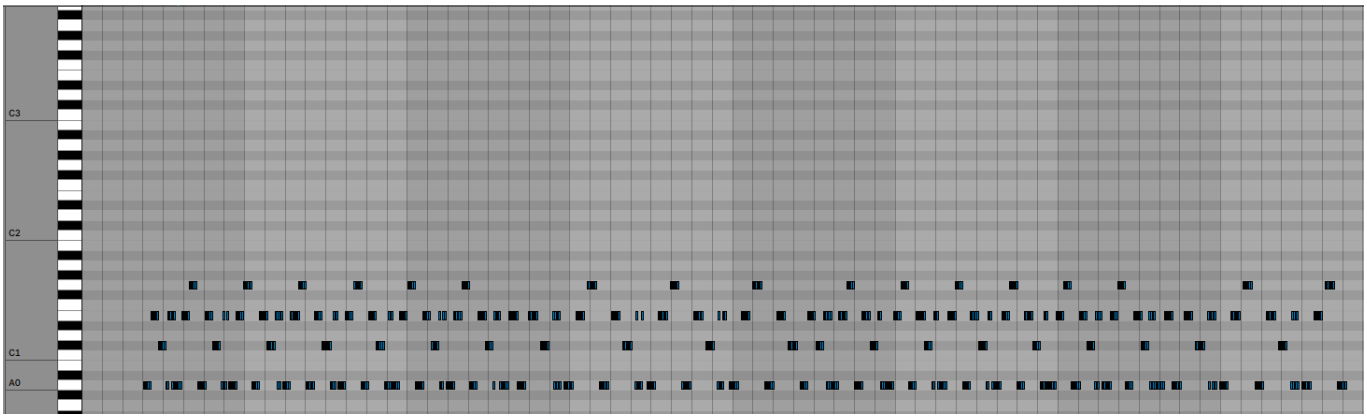


Figura 13. Notas de bajo para el “mood” épico.

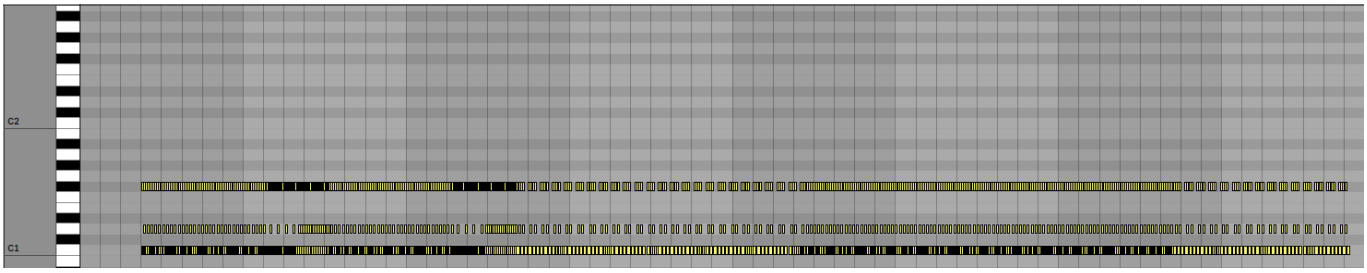


Figura 14. Notas de batería para el “mood” épico.

Tomando las figuras 6, 9 y 12; correspondientes a las notas de guitarra para los tres “moods” como referencia, se puede observar que las tres sensaciones tienen la misma estructura: los primeros ocho compases pertenecen a la introducción tipo “ambience”. Luego, las secciones dos y cuatro de las canciones son los versos que contienen las mismas notas tanto en ambas secciones, como en las tres sensaciones, exceptuando el “mood” agresivo ya que el algoritmo escogió el uso de “power chords” para este verso. Estas “power chords” tienen la

característica que están compuestas por corcheas, que se obtienen comparando todo el compás y, si el algoritmo encuentra que todas las notas del compás son corcheas, las convierte en acordes. De lo contrario, Seguidamente, las secciones tres y cinco corresponden a los coros. Estos coros utilizan el ritmo fundamental. Sin embargo, a pesar de que la semilla no generó las mismas progresiones para cada "mood", son las mismas notas y los mismos acordes. Se puede observar que en los tres coros hay una mayor presencia de corcheas para los "moods" épico y melancólico y una mayor presencia de notas blancas y negras en el "mood" agresivo. Por otro lado, la sexta parte de la canción en el "mood" agresivo, el outro, se puede observar que son las mismas notas que en los versos, pero hay una mayor presencia de notas más largas que en estos ya que se bajó el tempo en esta parte.

6.2. Reproduciendo y sincronizando los instrumentos

Una de las incógnitas que pueden surgir a lo largo de este informe es la manera en cómo están sincronizados los instrumentos, tanto en la reproducción de sus notas con sus duraciones, como en la coordinación con todos los demás instrumentos ya que al momento de reproducir cada canción, se deben de reproducir todos los instrumentos por separado, pero al mismo tiempo, en el mismo tiempo. Para esta sincronización y evitar desface ente los instrumentos, se utilizó el paralelismo utilizando el atributo `fork()` de la clase `Session`, anteriormente mencionado. Este atributo genera un proceso paralelo que se ejecuta sobre un hijo de la clase `Clock`, así puede SCAMP reproducir dos o más instrumentos en el mismo tiempo. Al principio, se había pensado hacer una función por instrumento, pasándole todos los compases a cada función para que así cada instrumento iterara sobre sus compases y después sobre sus notas. Sin embargo esto generaba problemas en ciertas partes de las canciones, a pesar de que cada instrumento se tocaba en su respectivo tiempo (nunca hubo ningún instrumento desfasado en el tiempo), habían problemas de silencios omitidos o, en algunos casos, un instrumento repetía un verso cuando los demás instrumentos ya habían terminado de tocar todos los compases. Para solucionar esto, se hizo que el proceso padre (el del reproductor) fuera el que iterase sobre todos los compases de la canción y los procesos hijos (las funciones de los instrumentos) iterasen solamente sobre las notas de cada compás. Esto hizo una mejora en el rendimiento del programa y, ahora sí, todos los instrumentos tocan sus respectivos compases en orden, sin omitir ningún silencio y terminan al mismo tiempo. La Figura 15 muestra el comportamiento de los procesos de la herramienta al momento de generar y sintetizar una canción.

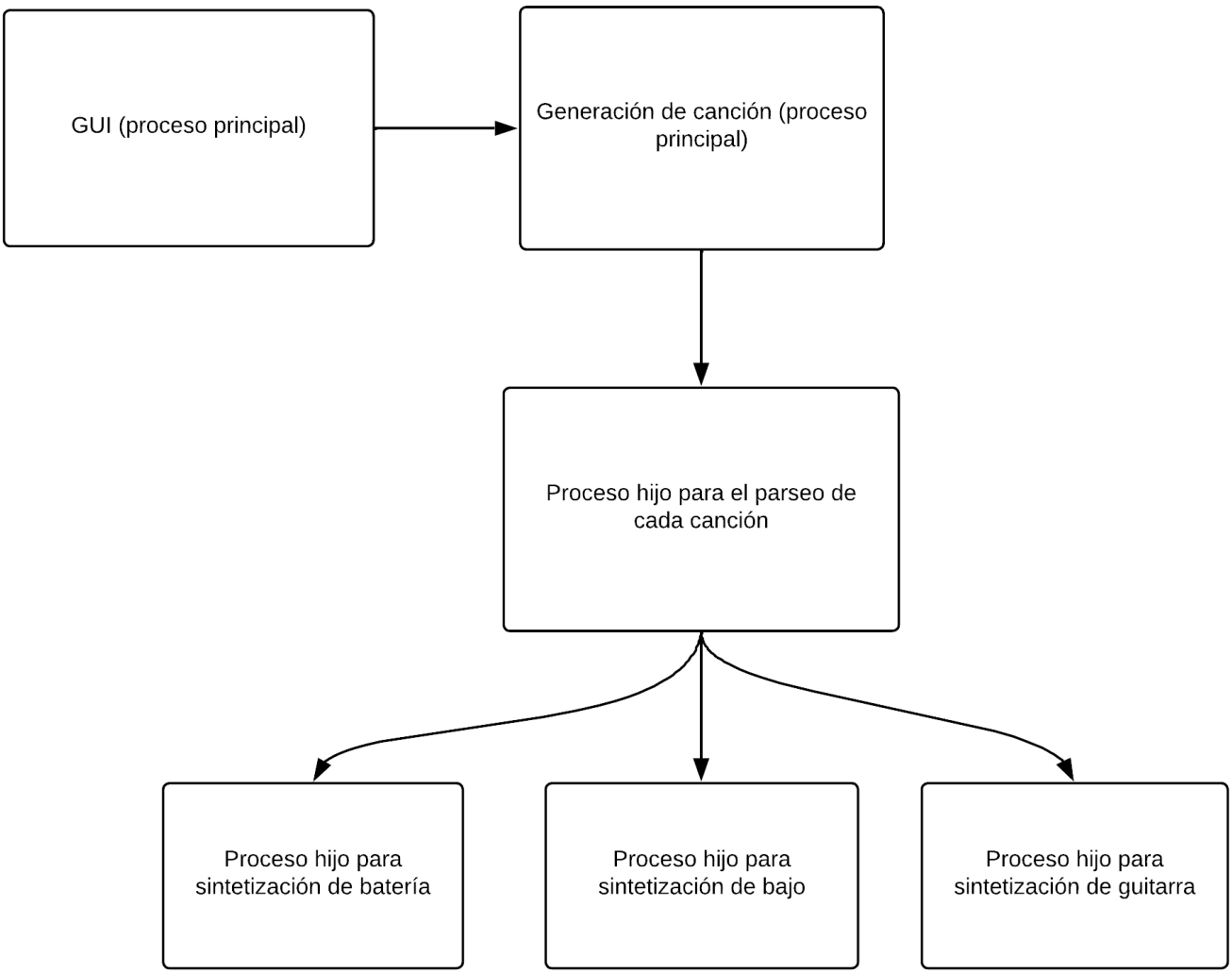


Figura 15. Notas de bajo para el "mood" agresivo.

Tomando los segundos ocho compases del primer verso del "mood" agresivo como referencia, la Figura 16, pueden separarse en dos tablas para comprobar esta coordinación con sus notas, así como se ven en las tablas 3 y 4, que pertenecen a la duración de las notas y a las notas MIDI respectivamente. En el Cuadro 3 se puede observar la presencia de la duración de los "power chords" (en corcheas) que el algoritmo escogió en base a la semilla. También se puede observar en el Cuadro 4 que las notas vienen en pares de dos notas, acordes, siendo la primera la tónica de ese acorde y la siguiente su quinta. En los compases 4 y 8, se pueden observar que las últimas cuatro notas están sueltas: no son "power chords", a pesar de ser también corcheas, ya que así como se discutió en la metodología, solo los compases con corcheas en todas sus notas son "power chords".

Compás	Duración de las notas							
1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
2	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
3	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
4	0.25, 0.25	0.25, 0.25	0.25, 0.25	0.25, 0.25	0.5	0.5	0.5	1.0
5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
6	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
7	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
8	0.25, 0.25	0.25, 0.25	0.25, 0.25	0.25, 0.25	0.5	0.5	0.5	1.0

Cuadro 3: Duración de las notas en el primer verso del "mood" agresivo para la guitarra.

Compás	Notas MIDI							
1	33, 40	33, 40	33, 40	33, 40	33, 40	33, 40	33, 40	33, 40
2	33, 40	33, 40	33, 40	33, 40	33, 40	33, 40	33, 40	33, 40
3	33, 40	33, 40	33, 40	33, 40	33, 40	33, 40	33, 40	33, 40
4	33, 40	33, 40	33, 40	33, 40	50	52	52	50
5	33, 40	33, 40	33, 40	33, 40	33, 40	33, 40	33, 40	33, 40
6	33, 40	33, 40	33, 40	33, 40	33, 40	33, 40	33, 40	33, 40
7	33, 40	33, 40	33, 40	33, 40	33, 40	33, 40	33, 40	33, 40
8	33, 40	33, 40	33, 40	33, 40	45	52	52	52

Cuadro 4: Notas MIDI obtenidas en los primeros ocho compases del primer verso agresivo para la guitarra.

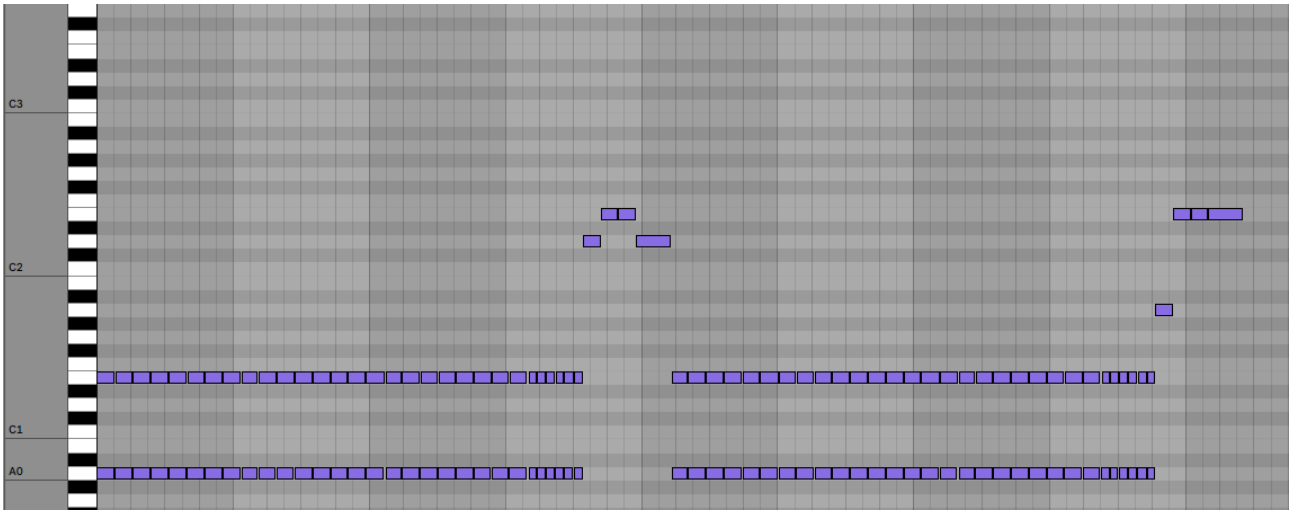


Figura 16. Notas de guitarra del primer verso para el "mood" agresivo..

Compás	Duración de las notas							
1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
2	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
3	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
4	0.25, 0.25	-0.25, 0.25	0.25, 0.25	0.25, 0.25	0.5	0.5	0.5	1.0
5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
6	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
7	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
8	0.25, 0.25	-0.25, 0.25	0.25, 0.25	0.25, 0.25	0.5	0.5	0.5	1.0

Cuadro 5: Duración de las notas en el primer verso del "mood" agresivo para el bombo.

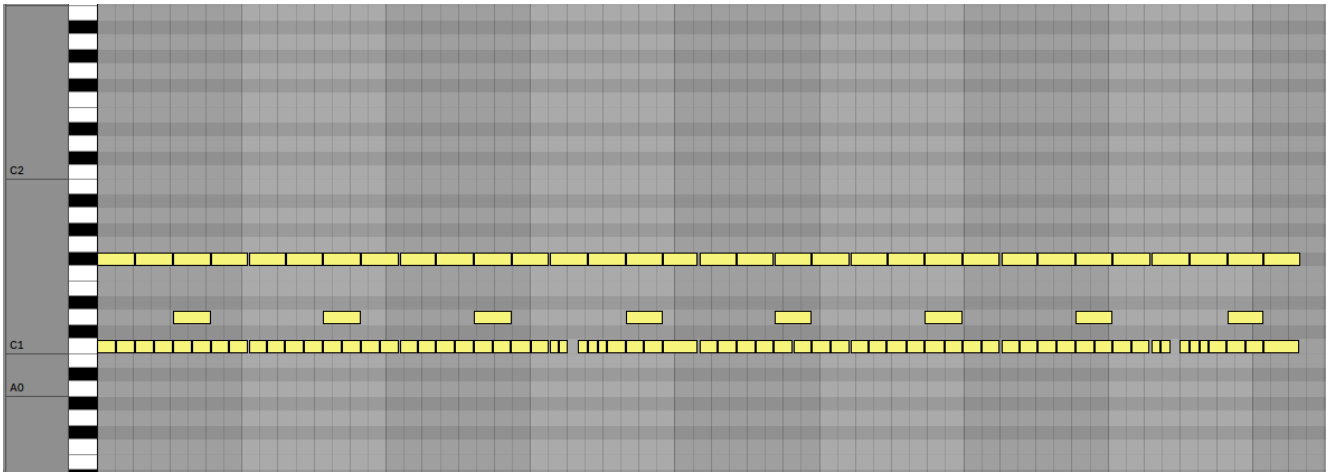


Figura 17. Notas de batería del primer verso para el "mood" agresivo..

Así mismo, como se observa en la Figura 17 y el Cuadro 5, los cuales pertenecen a las notas del bombo de la batería para los mismos compases anteriormente mencionados, se puede observar que pertenecen a las mismas duraciones que las notas de de de guitarra. Por lo tanto, se puede comprobar que tanto la batería como la guitarra van sincronizados en el mismo tiempo. Otro ejemplo de esta sincronización es en los compases 135 al 142, de la misma canción, ya que el 135 y el 136 pertenecen a los últimos dos compases del último coro, el 137 y 138 pertenecen a los primeros dos compases del outro, en los que se hace esa transición silenciando todos los instrumentos en ambos compases exceptuando a la guitarra en el compás 136, como se discutió en la metodología. Por ello se sacaron las figuras 18 y 19, pertenecientes a la guitarra y batería, respectivamente, para demostrar nuevamente esa sincronización en el tiempo. De igual manera se puede ver en los cuadros 6 y 7 las duraciones de cada nota. En estos cuadros se puede observar que los primeros dos compases de la guitarra son las mismas notas que del bombo, pero reducidas. En los siguientes dos compases se silencia completamente el bombo pero la guitarra tiene dos notas redondas, la segunda silenciada. Los compases 139 al 142 son una copia de los primeros cuatro compases del verso, pero se puede notar que en el bombo, no es el ritmo fundamental, ya que este es uno de los ritmos preescritos.

Compás	Duración de las notas							
135	1.0		1.0		1.0		1.0	
136	0.5	0.5	0.5	1.0		0.5	1.0	
137	4.0							
138	-4.0							
139	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
140	0.25, 0.25	0.25, 0.25	0.25, 0.25	0.5	0.5	0.5	1.0	
141	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
142	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

Cuadro 6: Duración de las notas en el primer verso del “mood” agresivo para la guitarra.

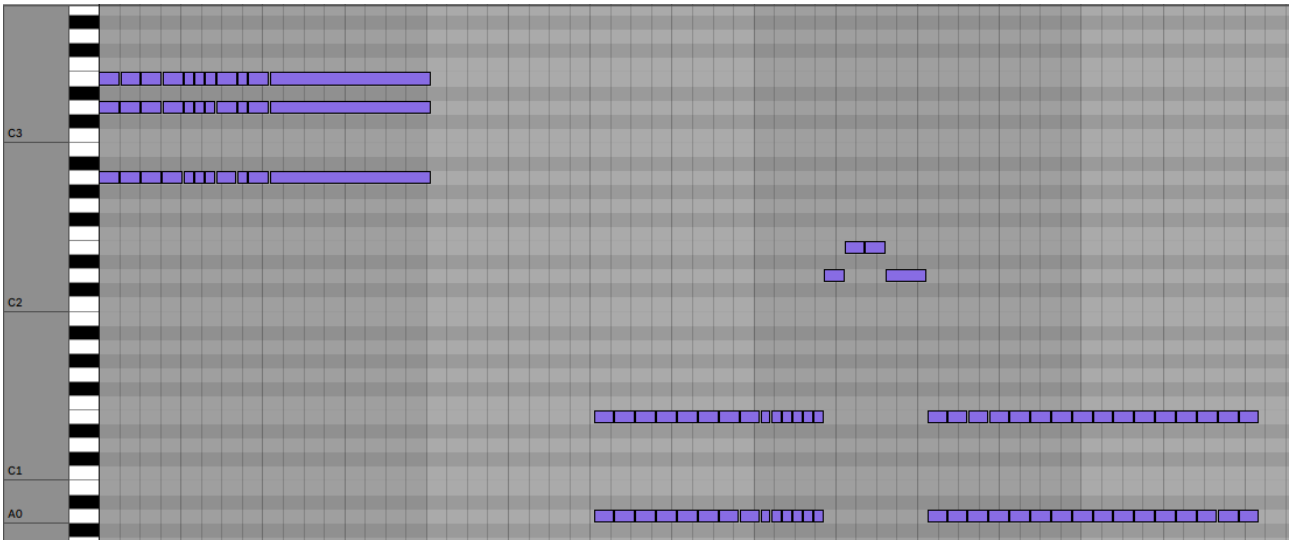


Figura 18. Duración de las notas de guitarra de los compases 135 al 142 para el “mood” agresivo..

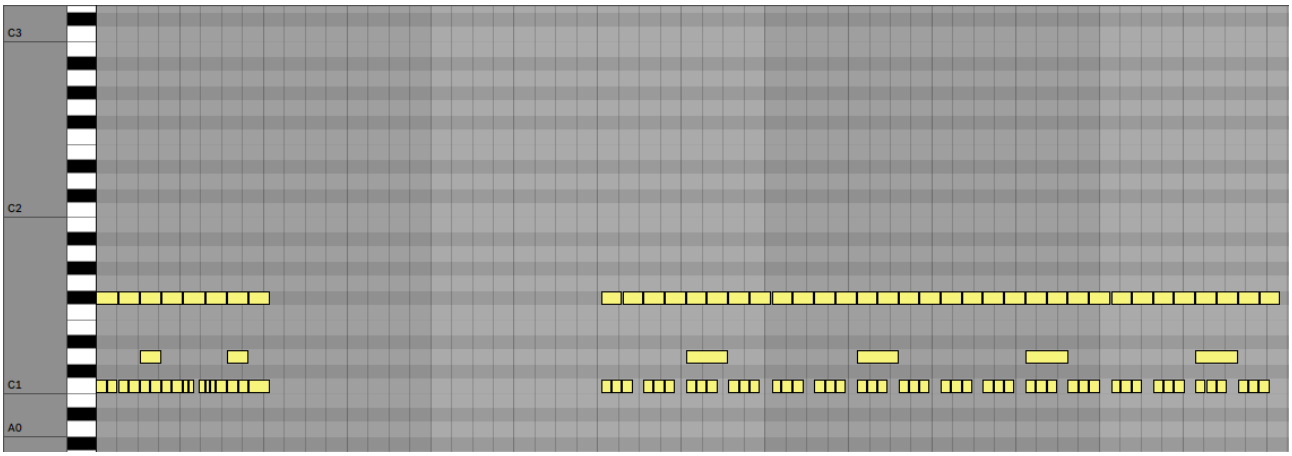


Figura 19. Duración de las notas de batería de los compases 135 al 142 para el “mood” agresivo..

Compás	Duración de las notas							
135	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
136	0.25, 0.25	-0.25, 0.25	0.25, 0.25	0.5	0.5	0.5	1.0	
137	-1.0		-1.0		-1.0		-1.0	
138	-1.0		-1.0		-1.0		-1.0	
139	0.25, 0.25	0.25, -0.25	0.25, 0.25	0.25, -0.25	0.25, 0.25	0.25, -0.25	0.25, 0.25	0.25
140	0.25, 0.25	0.25, -0.25	0.25, 0.25	0.25, -0.25	0.25, 0.25	0.25, -0.25	0.25, 0.25	0.25
141	0.25, 0.25	0.25, -0.25	0.25, 0.25	0.25, -0.25	0.25, 0.25	0.25, -0.25	0.25, 0.25	0.25
142	0.25, 0.25	0.25, -0.25	0.25, 0.25	0.25, -0.25	0.25, 0.25	0.25, -0.25	0.25, 0.25	0.25

Cuadro 7: Duración de las notas en el primer verso del "mood" agresivo para el bombo.

6.3. Generación de partituras

La generación de las partituras fue un problema sencillo de resolver, así como se discutió en la metodología, se utilizaron los atributos `start_transcribing` antes de reproducir los instrumentos y `stop_transcribing` después de haberlos reproducido. A pesar de que SCAMP hace un intento de poner cada nota de manera realista, no puede distinguir entre un instrumento en clave de sol y uno en clave de fa, por lo que los interpreta con la misma clave, así como se ve en la Figura 20. Sin embargo, a pesar de tener problemas de interpretación, SCAMP sí puede interpretar bien las posiciones y duraciones de las notas en el pentagrama, al igual que marca el tempo correctamente, al igual que puede separar dos pentagramas, uno debajo del otro, de cada instrumento que se le pase. No se incluyeron los instrumentos de percusión en la generación de las partituras, ya que SCAMP no puede generar una partitura de percusión, ya que esa es diferente.

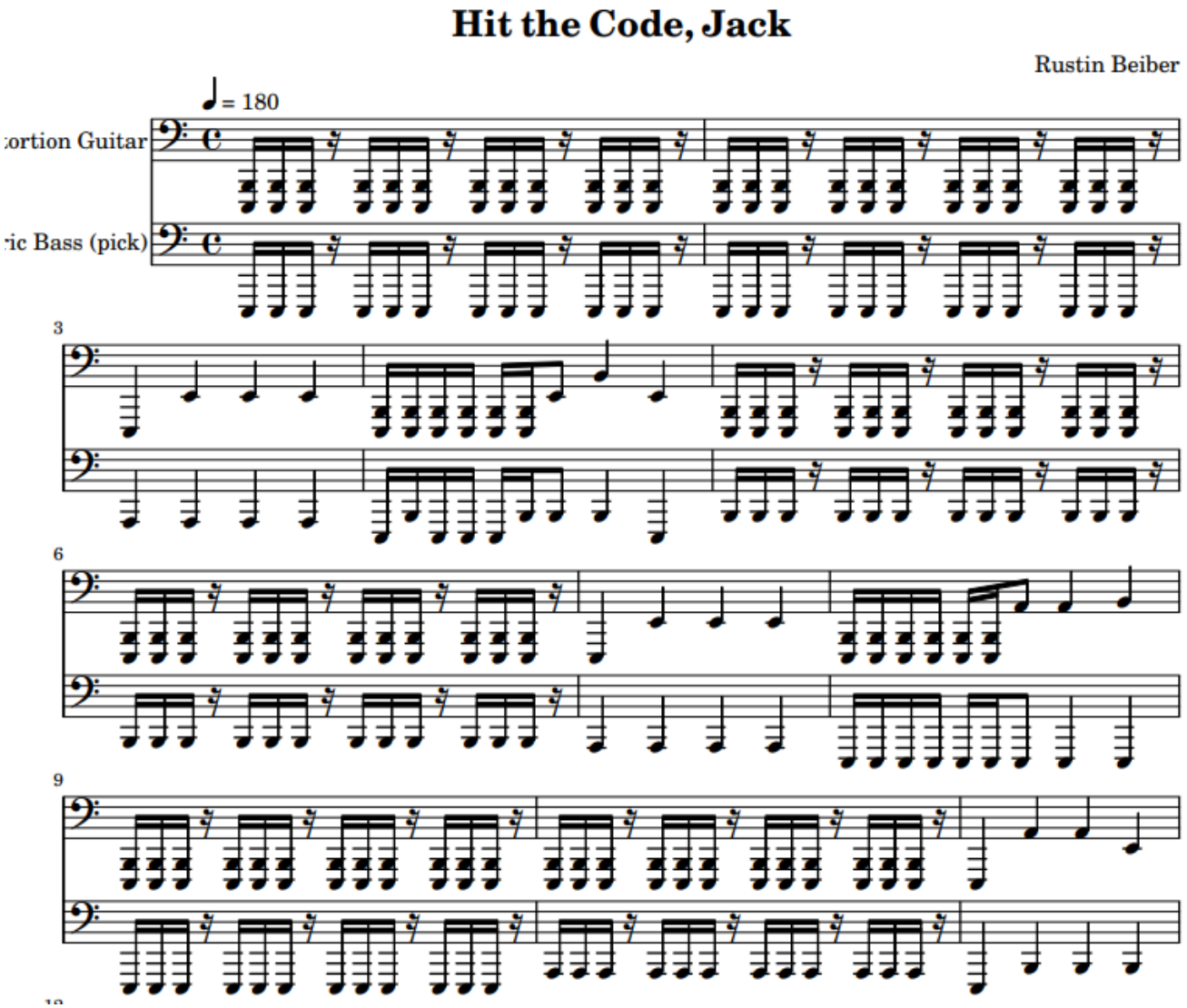


Figura 20. Partitura generada para canción de “mood” agresivo y semilla 1999..

6.4. Correcciones al algoritmo

Se hicieron una serie de pequeñas correcciones al algoritmo. Estas correcciones no fueron cambios importantes en la lógica del algoritmo, como lo fuera reestructurar toda la canción. Sino fueron pequeños cambios como reducciones al número de compases en los versos y coros, al igual que cambios en las duraciones de las notas de los coros. Esto se puede ver reflejado comparando las figuras 12, 13 y 14 con las figuras 21, 22 y 23, siendo estas últimas figuras de la guitarra, bajo y batería, respectivamente, pero para una nueva canción en “mood” épico generada con la semilla 224508386959501 (reproducible desde https://on.soundcloud.com/Ux95x). En estas se puede observar que el largo de los versos y los coros ha disminuido.

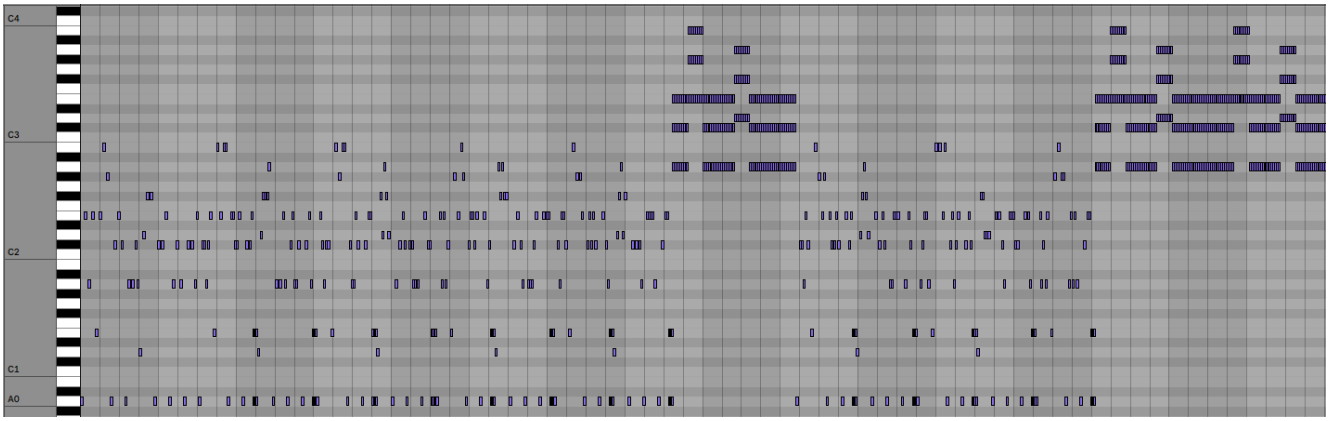


Figura 21. Duración de las notas de guitarra para el nuevo “mood” épico..

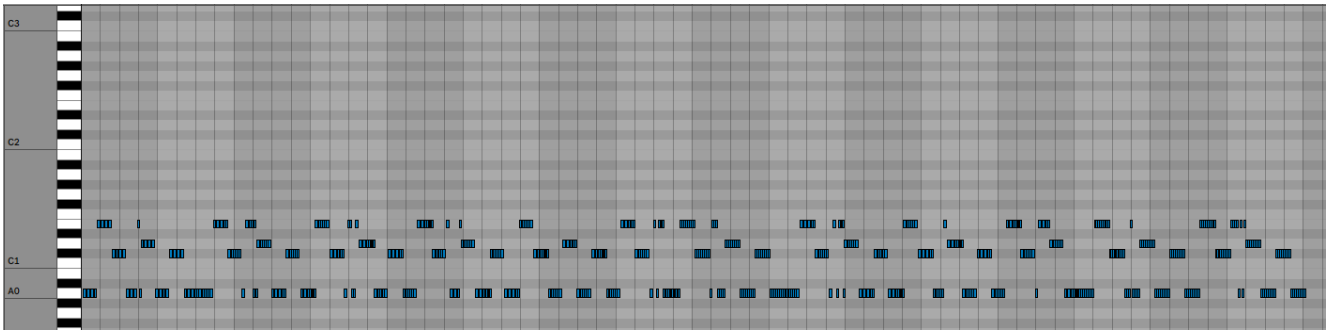


Figura 22. Duración de las notas de bajo para el nuevo “mood” épico..

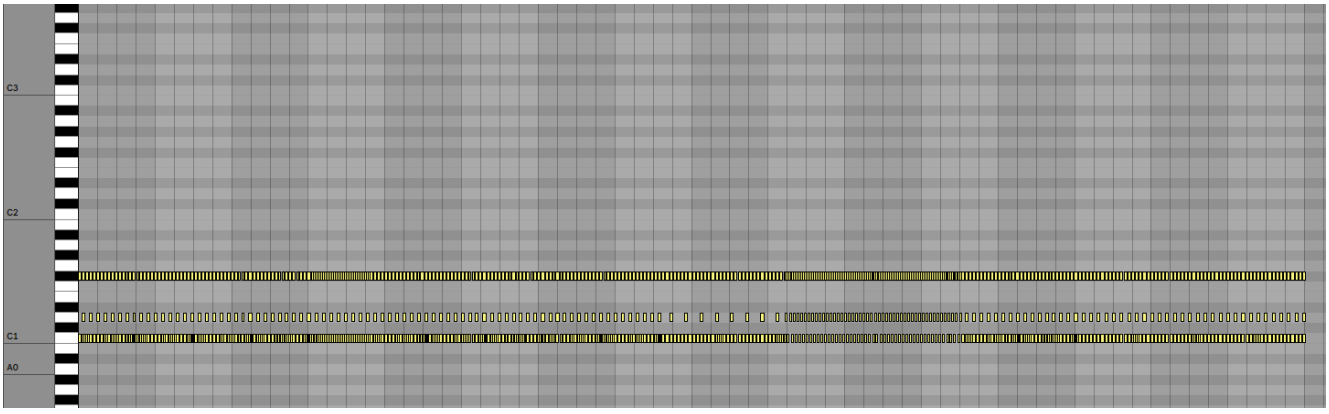


Figura 23. Duración de las notas de batería para el nuevo “mood” épico..

6.5. Rendimiento

Para el cálculo del rendimiento de la generación de cada canción y de la reproducción de la misma tanto con el sintetizador nativo de SCAMP, como la grabación de los instrumentos en “Ableton Live”, se utilizó la librería “psutil”, la cual es una librería multiplataforma utilizada para el monitoreo de procesos en ejecución y utilización del sistema en Python. Se

utiliza principalmente para la supervisión, limitación y gestión de los recursos y procesos en ejecución, así como la creación de perfiles. Para ello se monitorearon dos métricas principales: el uso del CPU (en porcentaje) y la utilización de memoria (en Mbytes), con la ayuda de los atributos `psutil.cpu_percent()` y `psutil.Process(os.getpid()).memory_info().rss`, respectivamente. Así mismo, se tomó el tiempo que tomó la generación de una canción con ayuda del atributo `time()` de la librería `time`.

La generación de las notas de cada canción dieron como resultados: un tiempo de ejecución promedio de un décimo de segundo, así como se ve en la Figura 24. También se puede observar que tomó un 16 % del porcentaje de CPU, pero esto no es del todo cierto, ya que este uso de CPU fluctúa constantemente. Ha dado resultados variantes, desde 8 % hasta un 31 %, por lo que se puede decir que esta métrica no es dependiente del proceso. Por último, se puede observar el uso de memoria desde que se inició el proceso hasta que se generó y guardó la canción. Esto dio como resultado un uso de memoria de menos de 0.20Mb, lo que significa que el proceso de generación no toma una porción de memoria significativa.

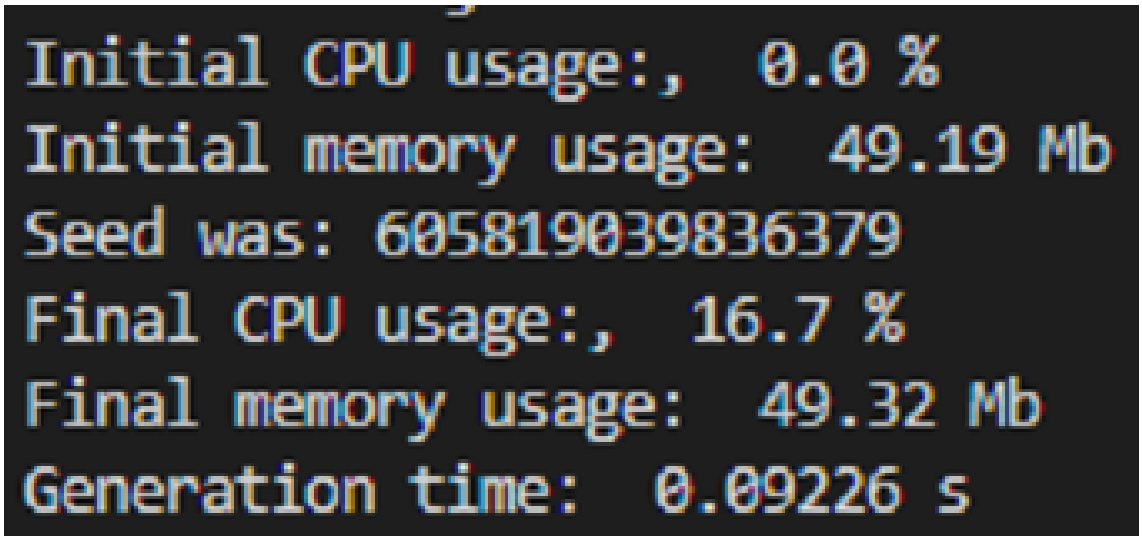


Figura 24. Métricas de porcentaje de CPU, utilización de memoria y tiempo de ejecución para la generación de una canción..

Por otra parte, al momento de la reproducción, se capturaron las métricas de la reproducción desde la sintetización nativa de SCAMP (no conectando el programa a "Ableton Live"). Esto dio como resultado lo que se muestra en la Figura 25. De igual manera que en la generación de la canción, el uso de CPU no es dependiente del proceso, ya que estos resultados también fluctúan. Sin embargo, si hubo un cambio en la utilización de la memoria, ya que esta sintetización de toda la canción toma un promedio de 58.04Mb. A pesar de haber obtenido un resultado mayor al de la generación, no toma una parte relevante de la memoria, por lo que se puede decir que la sintetización nativa y el modo en que se están iterando las canciones están bien optimizadas. No se tomó el tiempo de ejecución de esta parte del programa, ya que este tiempo de ejecución depende puramente del largo de la canción que se haya generado, por lo que no es relevante en la toma de estas métricas. De igual manera, se tomaron las métricas para la sintetización con "Ableton Live", como se muestra en la Figura 26. En esta sintetización y grabación de los instrumentos, hubo una

mejora en el rendimiento del uso de la memoria de un 48 %, con un uso de 28Mb de memoria en promedio. Esto se debe a que Python no tiene que hacer el trabajo de sintetización, ya que Python solo le pasa los parámetros a "Live" y este solo se encarga de sintetizar los instrumentos. Por ende, esto significa que todos los procesos, incluidos los subprocesos de reproducción de los instrumentos están bien optimizados para cualquiera de los usos que se le den a esta herramienta de generación de música procedural.

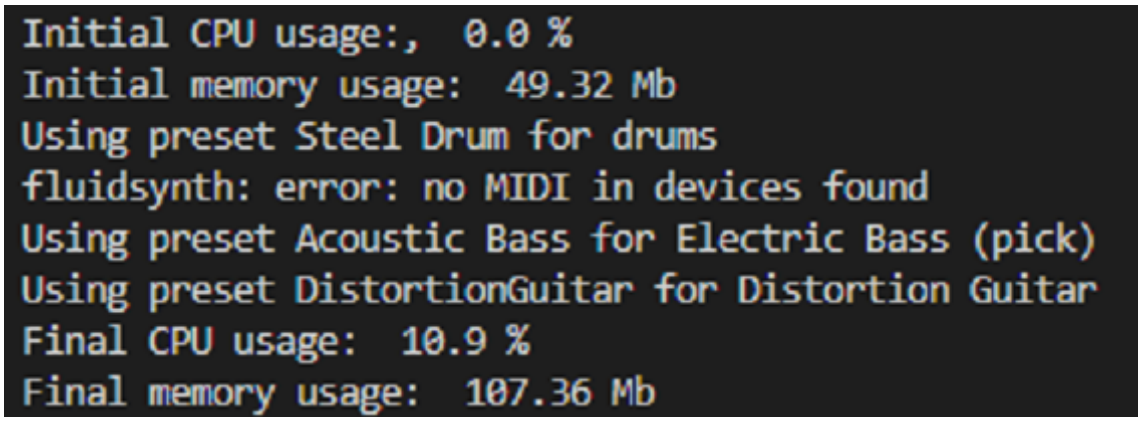


Figura 25. Métricas de porcentaje de CPU, utilización de memoria reproducción nativa de una canción..

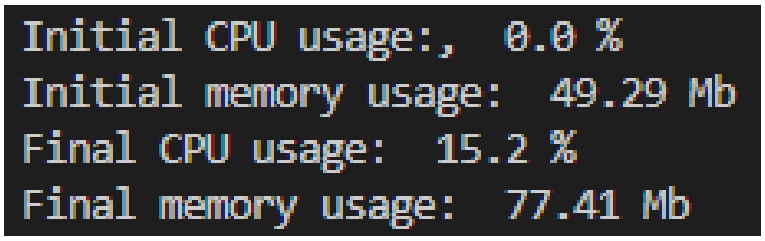


Figura 26. Métricas de porcentaje de CPU, utilización de memoria reproducción desde Ableton Live de una canción..

CAPÍTULO 7

Conclusiones

1. Se cumplieron los objetivos principales de de generar una pista musical aleatoria y diseñar y crear un programa generador de música procedural capaz de componer pistas para varios "moods" de un género en específico.

2. Se cumplió el objetivo de crear un algoritmo que se adapte a las reglas del género musical y a sus "moods".

3. Se pudo crear dicho algoritmo para que fuera lo más liviano posible, optimizando tanto la generación de una pista musical, como en la reproducción y sintetización de dicha canción.

4. Se pudieron generar las sensaciones que se querían transmitir mediante los "moods" del género musical.

5. Se pudo crear una herramienta compatible con estaciones de trabajo de audio digital capaz de generar canciones generadas proceduralmente para usuarios que necesiten música libre de "copyright".

6. Se pueden generar las partituras de las canciones generadas, aunque cuenten con pequeños errores, sí se puede distinguir las posiciones y duraciones de las notas.

CAPÍTULO 8

Recomendaciones

8.1. Equipo recomendado

8.1.1. Mínimo

- Sistema Operativo Windows/Linux/MacOS
- Procesador Intel Core i3 o AMD multi núcleos.
- 4Gb de memoria RAM.
- 12Mb de espacio de almacenamiento.
- Tarjeta y controladores de sonido nativos de Windows/Linux.

8.1.2. Recomendado (Ableton Live)

- Sistema Operativo Windows/MacOS
- Procesador Intel® Core™ i5 o AMD multi núcleos.
- 8Gb de memoria RAM.
- Pantalla con resolución de 1366x768.
- Al menos 8Gb de espacio de almacenamiento.
- Hardware de audio compatible con ASIO para un mejor rendimiento de audio.

8.2. Otras recomendaciones

- Una mayor experiencia en composición musical puede llegar a tener resultados más complejos con otros moods e incluso integrar otros géneros musicales.
- Utilizar esta herramienta con un DAW para sacar mejor provecho de todas las funcionalidades.

CAPÍTULO 9

Bibliografía

Collins, N. (2008). *Introduction for Computer Music*. Wiley.

Dodge, C., & Jerse, T. A. (1997). *Computer Music Synthesis Composition and Performance*. Schimer.

Evanstein, M. (2019). *SCAMP: Suite for Computer-Assisted Music in Python* (Tesis de Maestría). University of California Santa Barbara.

Green, D. (2016). *Procedural Content Generation for C++ Game Development*. Packt Publishing.

Hewitt, M. (2008). *Music Theory for Computer Musicians*. Cengage Learning.

Ignite Amps. (2019). *NadIR*. https://www.igniteamps.com

Manning, P. (2013). *Electronic and computer music*. Oxford University Press.

University New South Wales. (s.f.). *What is a Sound Spectrum?* http://www.phys.unsw.edu.au/jw/sound.spectrum.html

Veler Ltd. (2022). *C note*. https://m.basicmusictheory.com/c-note

Watkins, R. (2016). *Procedural Content Generation for Unity Game Development*. Packt Publishing. https://books.google.com.gt/books?id=8GwdDAAAQBAJ

CAPÍTULO 10

Anexos

10.1. Resultados a encuestas realizadas a usuarios finales

Enlace a la encuesta: https://forms.gle/ZKUGMLDRFVtPauwy8

Figura 27. Primera pregunta de la encuesta.

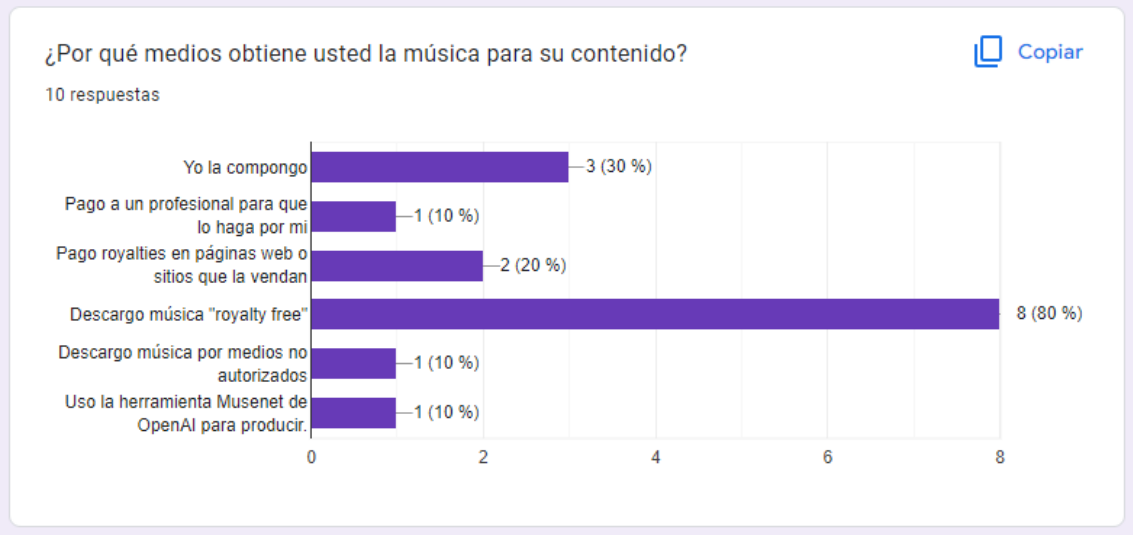


Figura 28. Segunda pregunta de la encuesta.

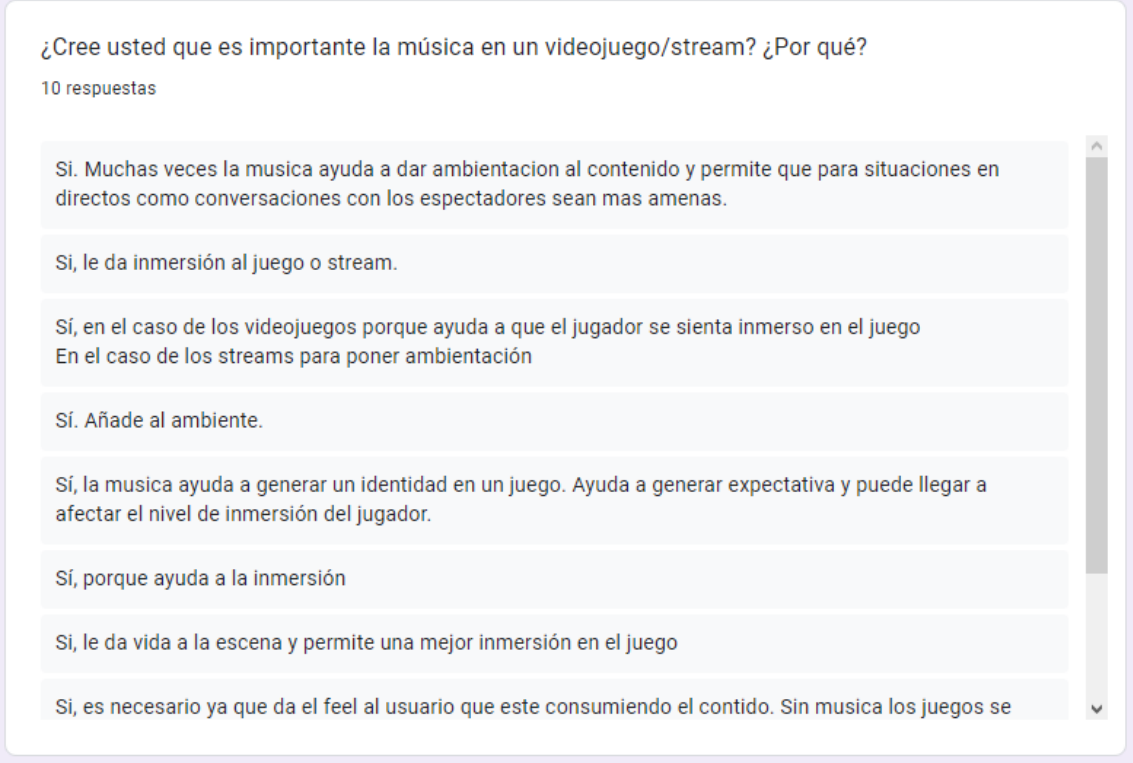


Figura 29. Tercera pregunta de la encuesta.

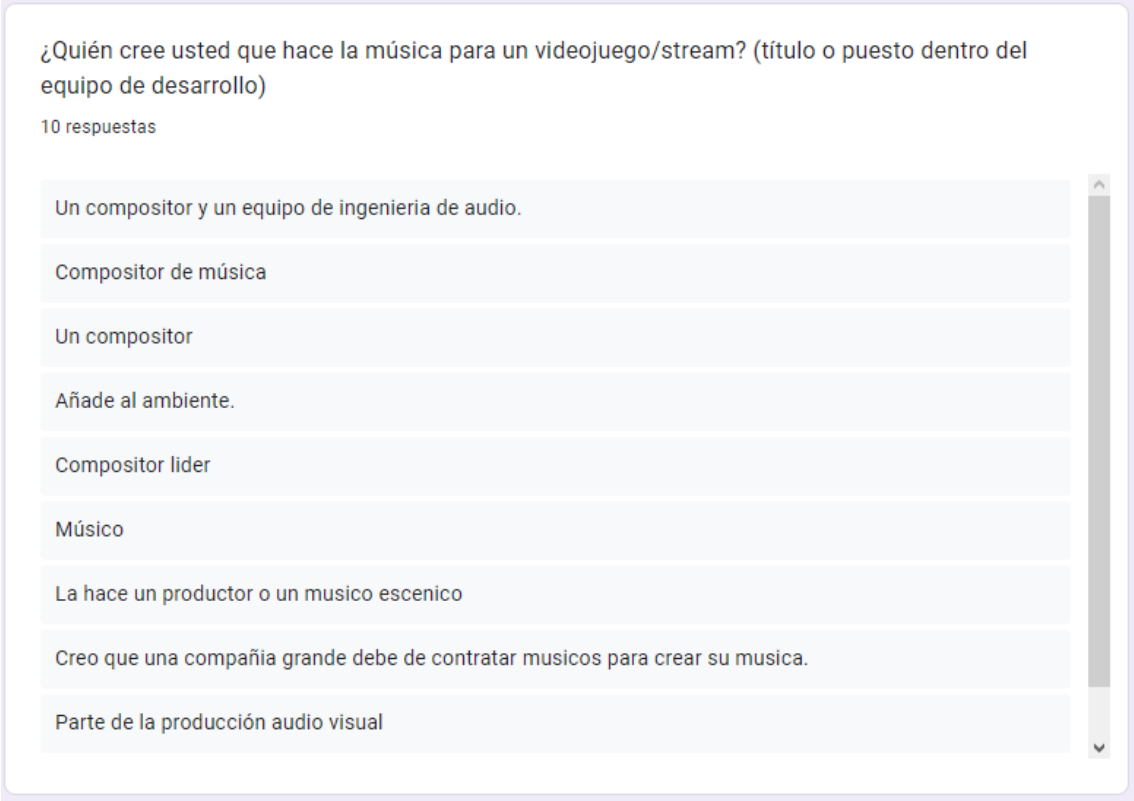


Figura 30. Cuarta pregunta de la encuesta.

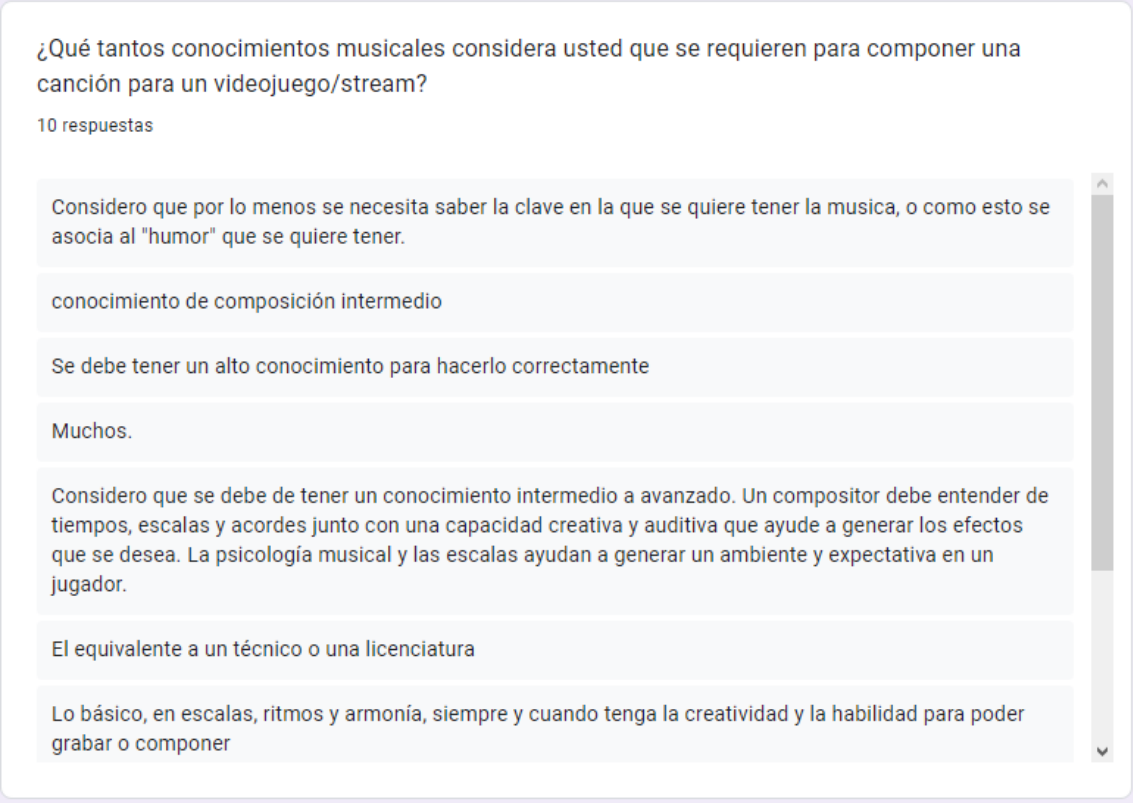


Figura 31. Quinta pregunta de la encuesta.

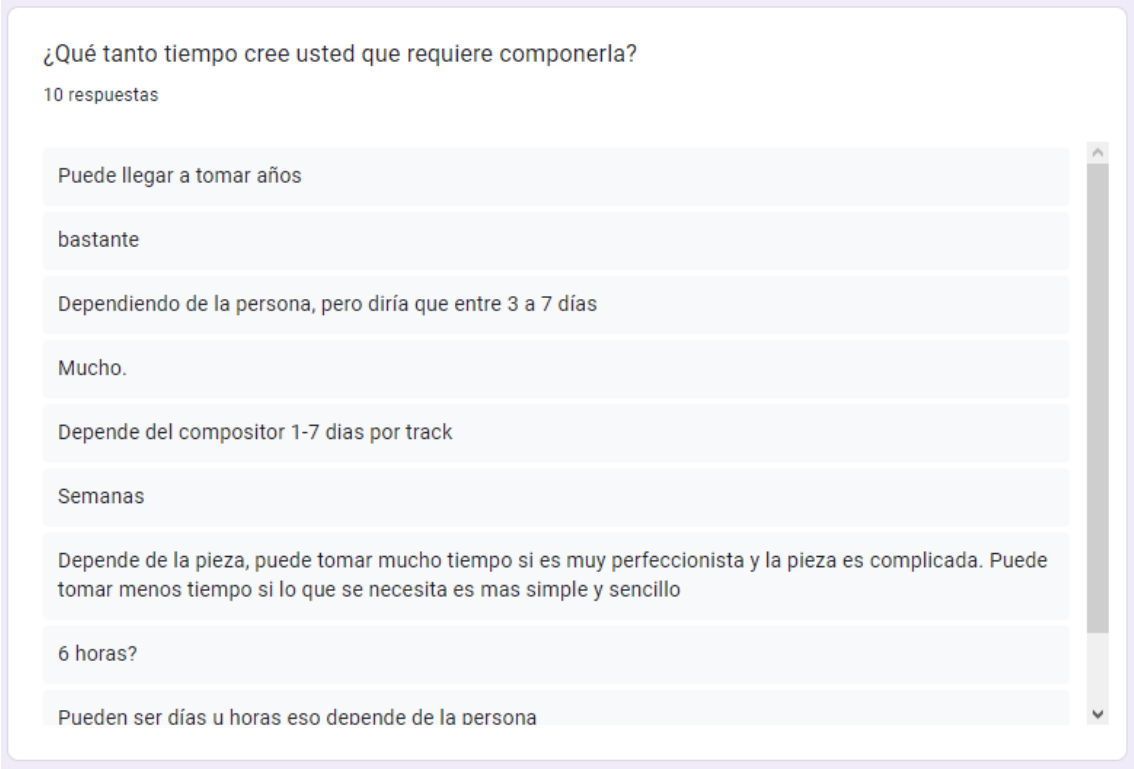


Figura 32. Sexta pregunta de la encuesta.

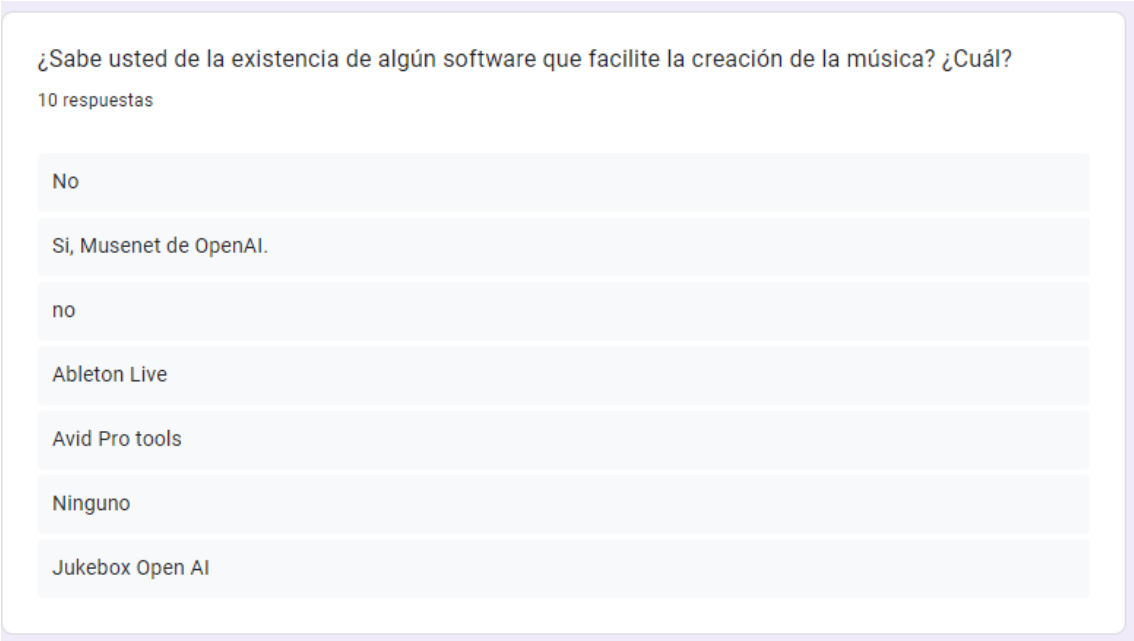


Figura 33. Séptima pregunta de la encuesta.

Figura 34. Octava pregunta de la encuesta.

Figura 35. Novena pregunta de la encuesta.

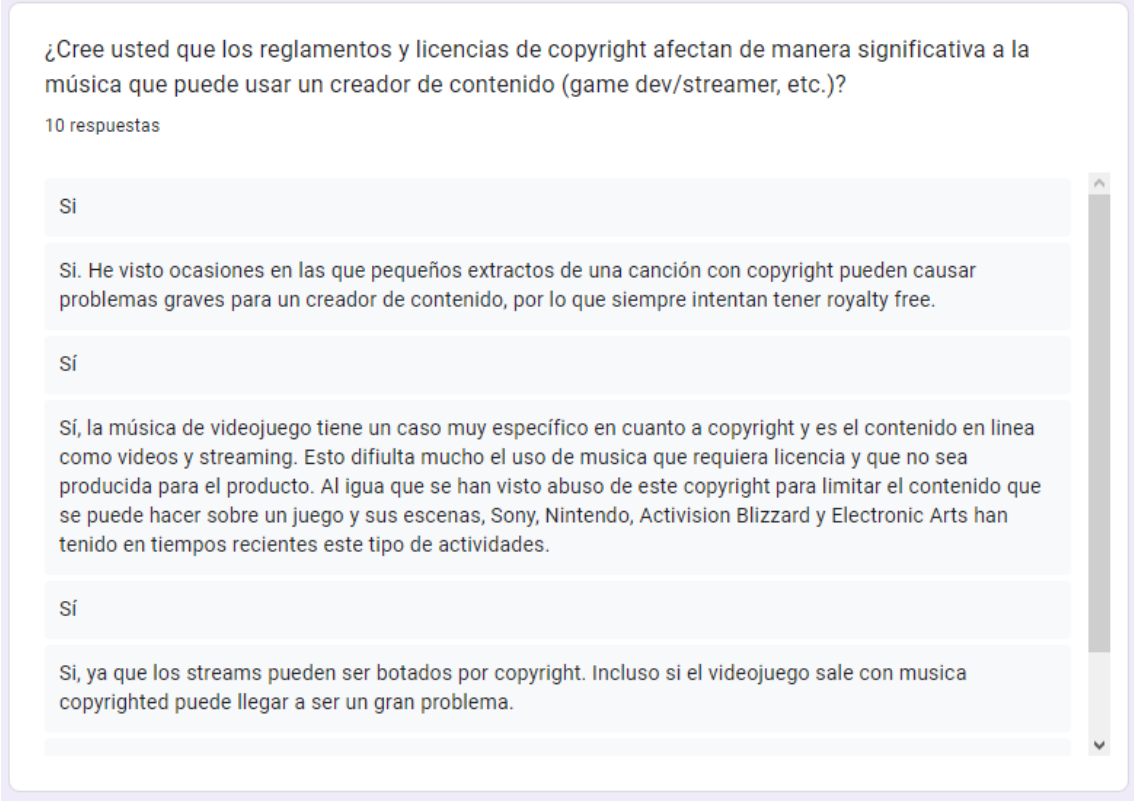


Figura 36. Décima pregunta de la encuesta.

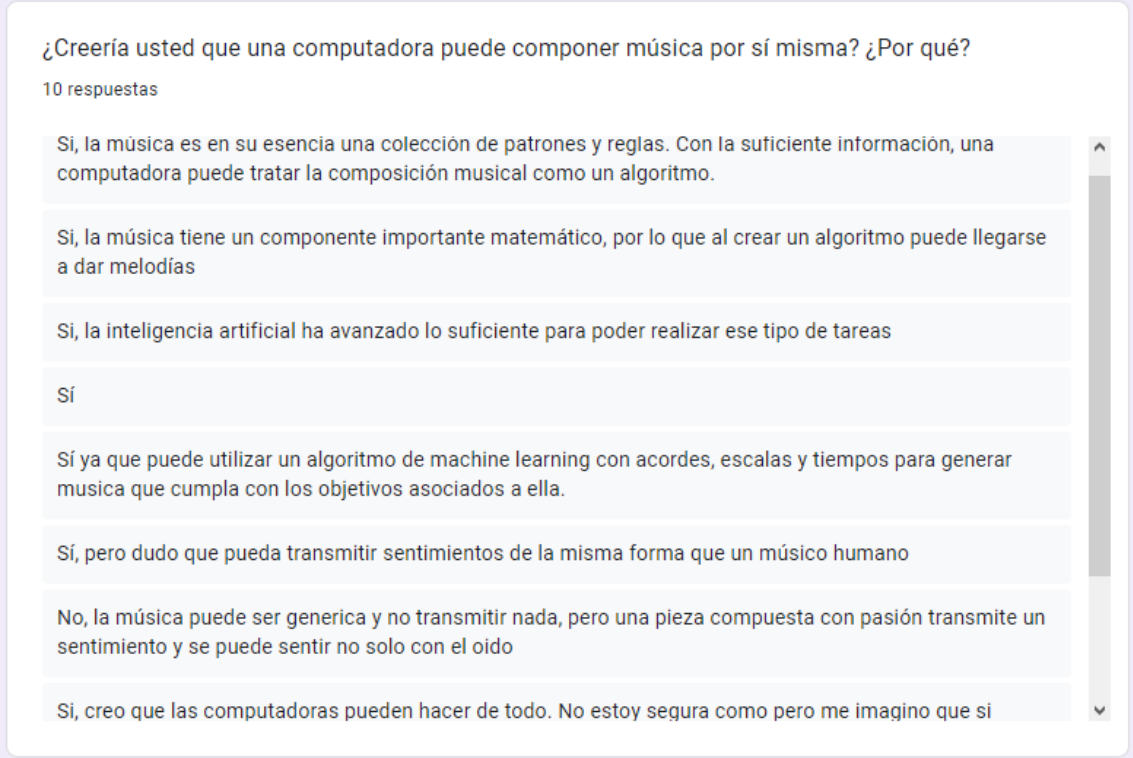


Figura 37. Décima primera pregunta de la encuesta.

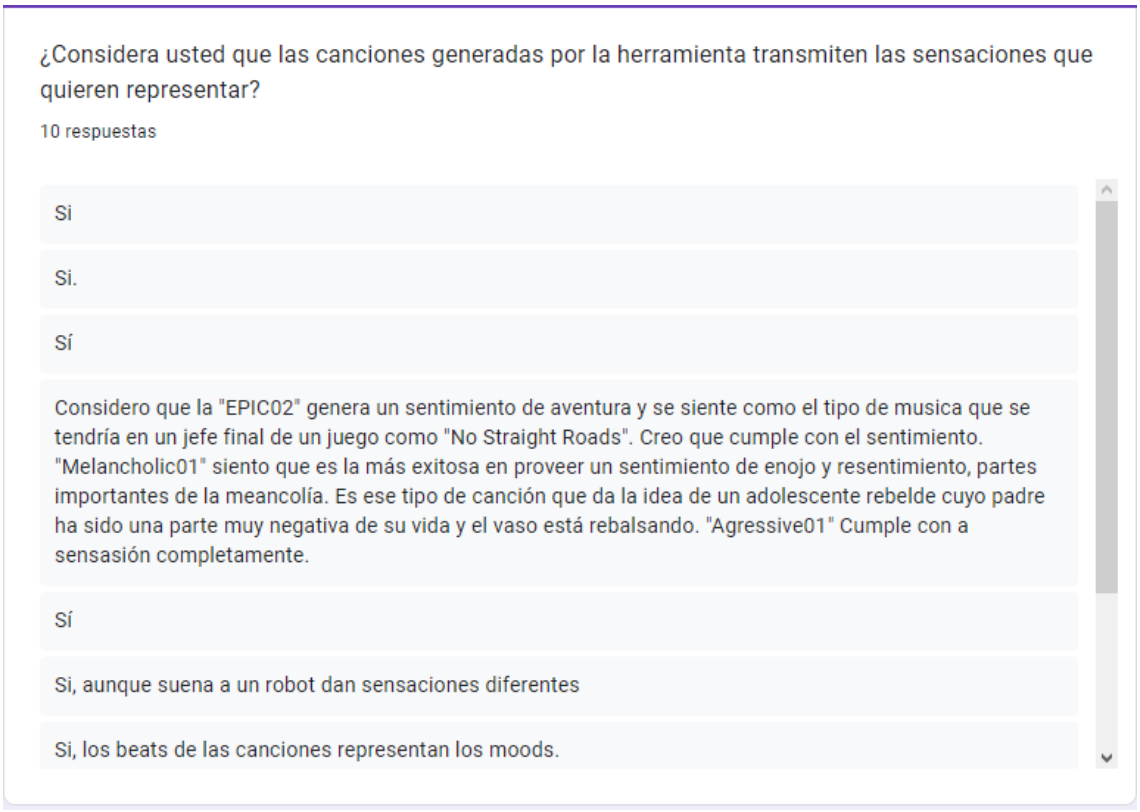


Figura 38. Décima segunda pregunta de la encuesta.

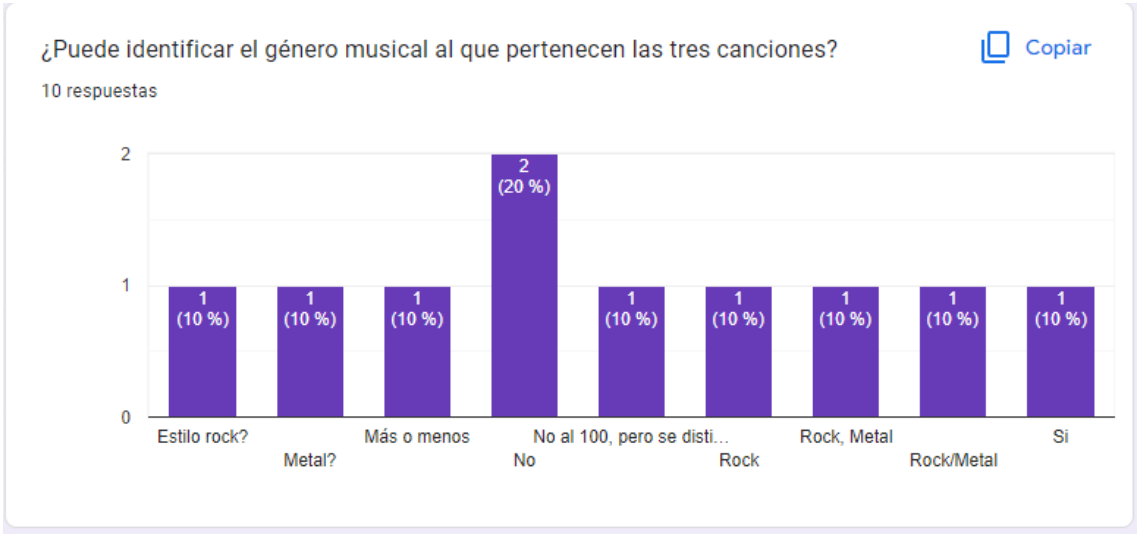


Figura 39. Décima tercera pregunta de la encuesta.

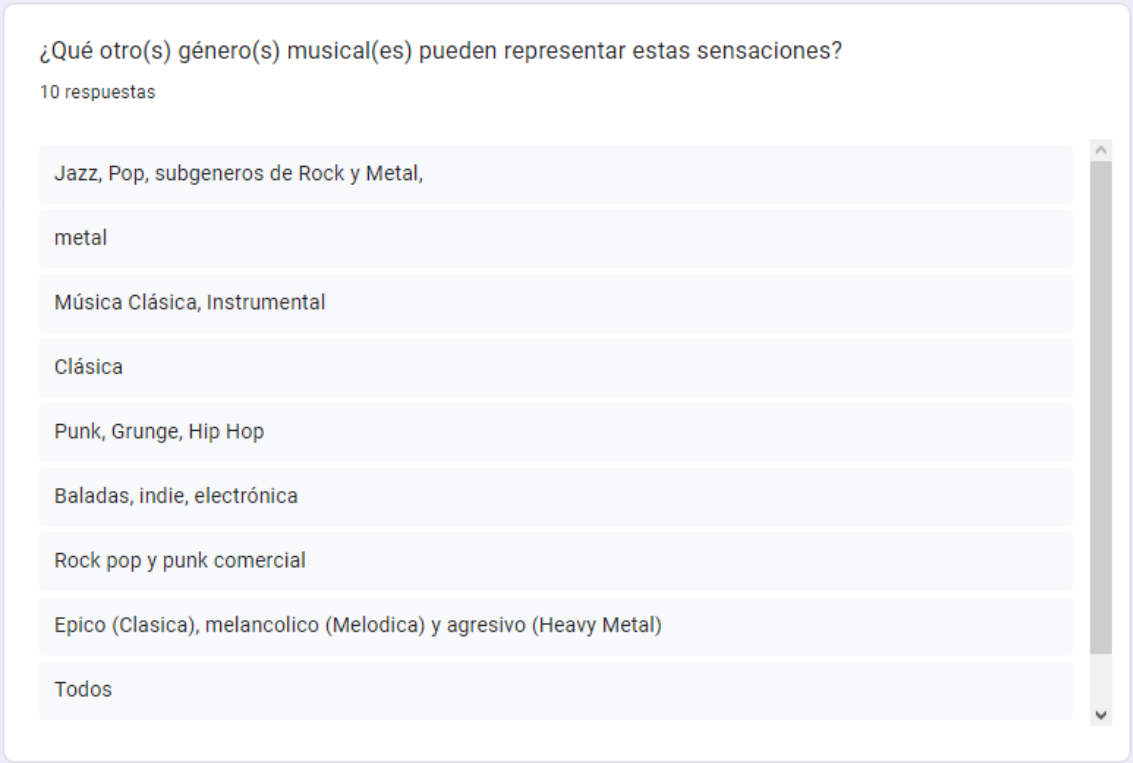


Figura 40. Décimo cuarta pregunta de la encuesta.

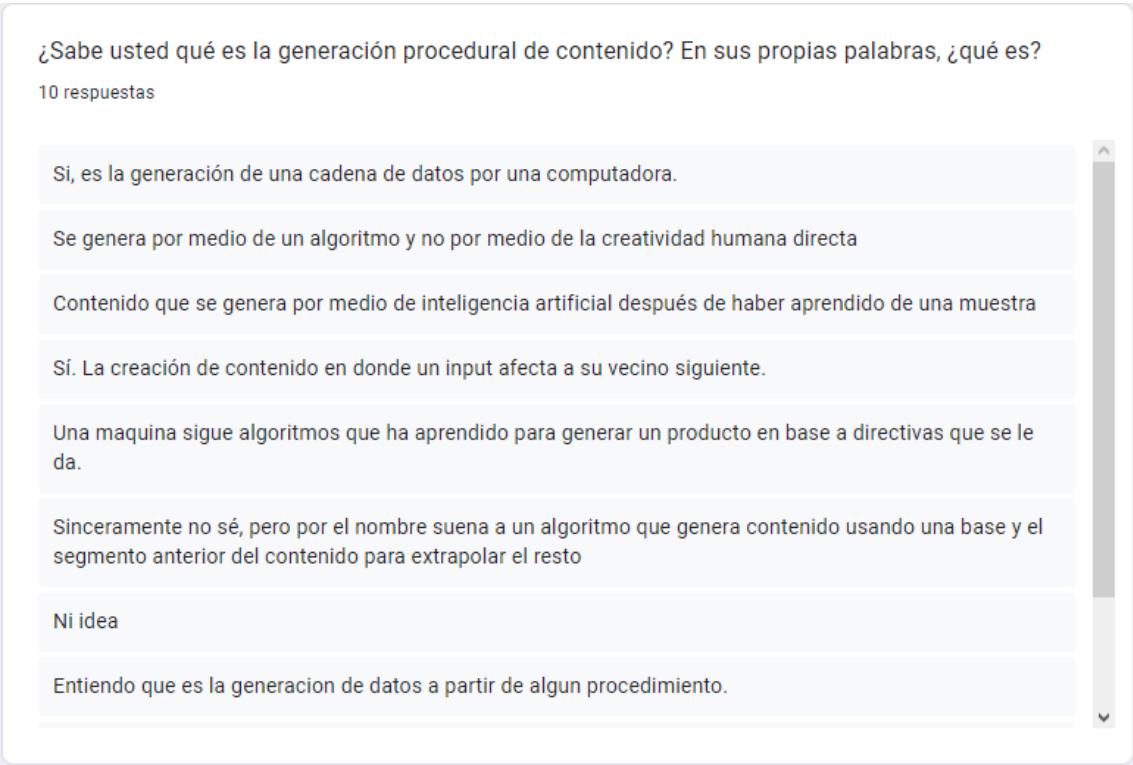


Figura 41. Décimo quinta pregunta de la encuesta.

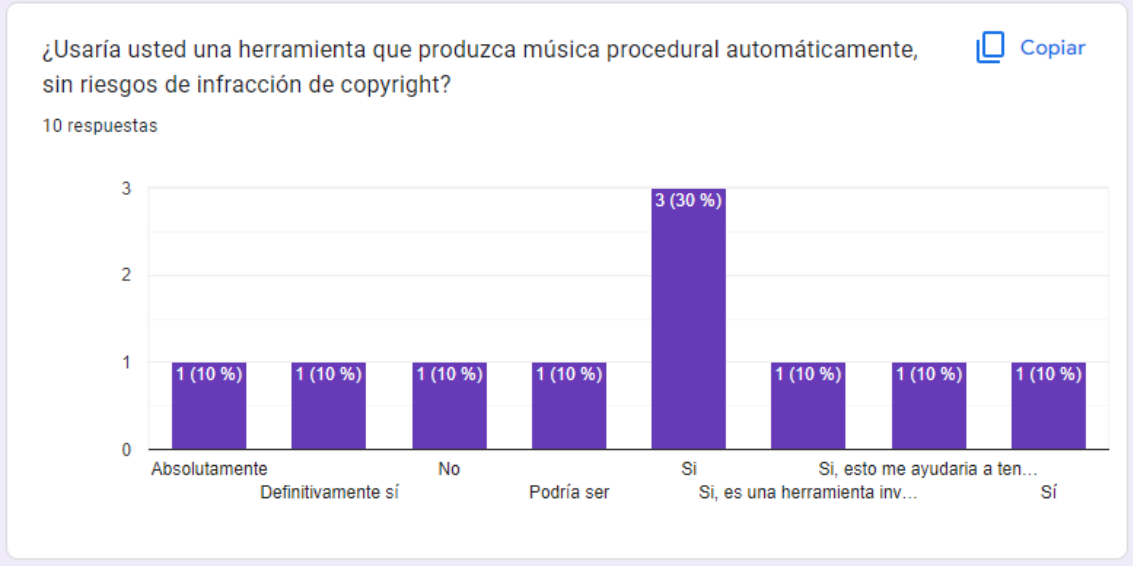


Figura 42. Décimo sexta pregunta de la encuesta.

10.2. Gráficas de espectros de onda

Figura 43. Gráfica del espectro de la forma de onda sinusoidal.

Figura 44. Gráfica del espectro de la forma de onda cuadrada.

Figura 45. Gráfica del espectro de la forma de onda triangular.

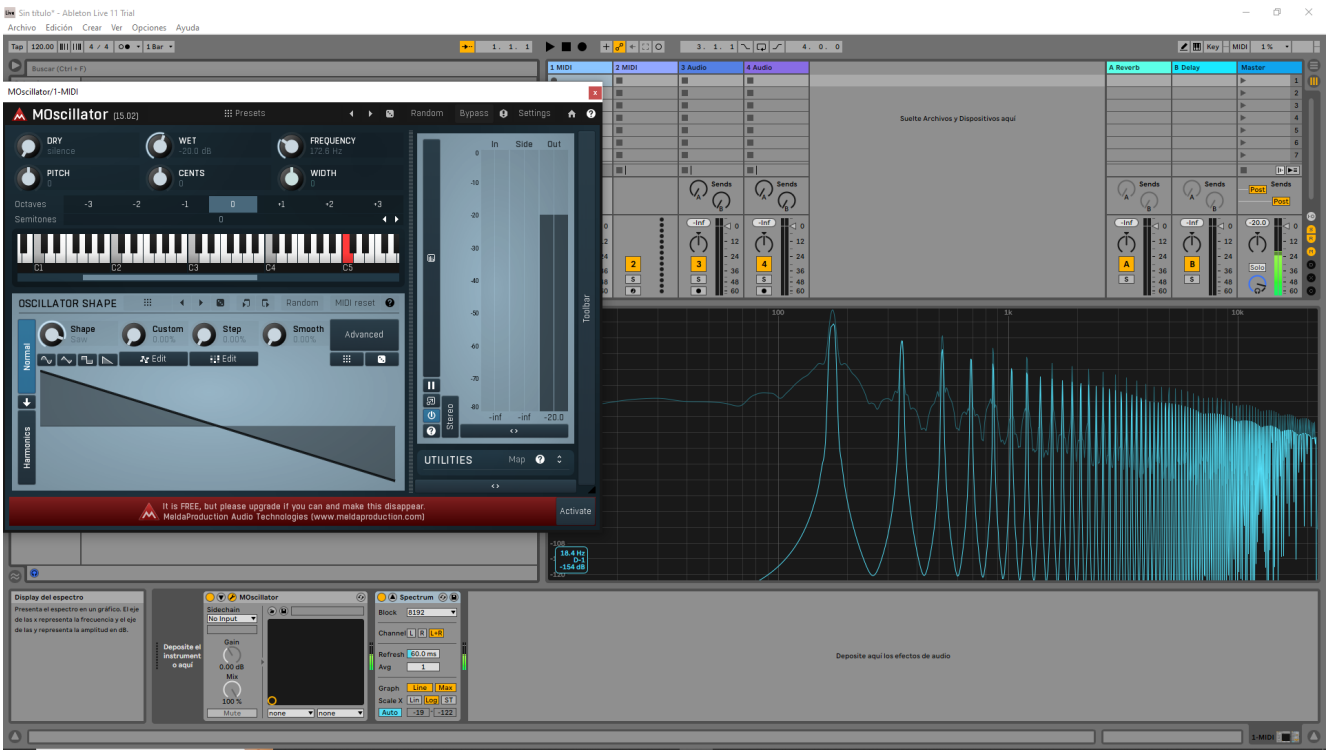


Figura 46. Gráfica del espectro de la forma de onda de sierra.

10.3. Enlace al repositorio de GitHub

https://github.com/DaronP/Tesis_Sandbox