SERENA FELDBERG
SANTIAGO CARRILLO
MICHELLE BEREZOUSKY
MARTINA GRÜNEWALD

TP FINAL

MUSICALIZADOR

PENSAMIENTO COMPUTACIONAL

UNIVERSIDAD DE SAN ANDRÉS

JULIO 2022

Objetivos del trabajo

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

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

Diseños del programa

En la primera parte del programa se encuentra desarrollada la sintetización de las notas. Esto se implementa a partir de dos diferentes clases, las cuales son Instrument y Note.

Por una parte Note permite leer el archivo que contiene la partitura, y de esta manera almacena los datos para su futuro uso. Tiene diferentes métodos a parte de la lectura del archivo que permiten obtener la duración total de la canción y cuantas notas tiene la partitura.

Por otro lado, la clase Instrumento también lee el archivo que contiene la información del instrumento. A partir de esto, se almacenan los datos y se desarrollan todos los métodos necesarios para la suma de los armónicos y la modularización de la nota. Es decir, en un principio, la clase tiene algunos métodos básicos para obtener distintos datos específicos como get_harmonics, get_amplitude o get_fn_info. Esta última lo que hace es obtener los datos de la parte específica de la nota, esto es ataque, sostenido o decaimiento. Las funciones principales de esta clase son gen_tone que se encarga de hacer la suma de los armónicos y gen_mod se encarga de modularizar la nota. Para su modulación hay un diccionario que contiene todas las funciones matemáticas posibles que puedan ser llamadas. Una vez implementadas estas funciones, el método gen_track tiene como función juntar todas las notas en un mismo array, formando el track principal. Por último la función de audio_wav genera un archivo en formato wave con el track generado.

Algunos datos como las funciones matemáticas o el valor equivalente de las frecuencias escritas en formato inglés se definieron en archivos separados.

Para probar que el código mencionado anteriormente funcionaba de manera correcta, se implementaron dos archivos de test. Uno por cada clase principal, las cuales son Instrument y Note. Dentro de los archivos de test, se encuentran varias funciones para evaluar el comportamiento de la

mayoría de las funciones de dichas clases. Algunos de los test son para corroborar que efectivamente los resultados obtenidos son los esperados, y otros de los test evalúan que los valores, por ejemplo, no sean negativos o que sean distintos, dependiendo el caso.

Luego, en relación a la segunda parte del trabajo lo organizamos de la siguiente manera; con la partitura convertida por el archivo 'midi2score.py', utilizando parser en el archivo xylophone_server.py, definimos como xylonote a la lista generada por la función xylonote_list, la cual lee la partitura en el formato pedido y descarta las notas que el instrumento físico no admite, acto seguido, con las que sí admite las transforma en un objeto de clase XyloNote y las almacena en una misma lista.

Habiendo dicho eso, el programa inicializa el servidor como un objeto de clase XyloClient, carga la lista con las xylonotas y las reproduce.

Reseñas sobre problemas encontrados

En un principio, cuando armamos el track final tuvimos varios problemas porque las dimensiones de los arrays no eran iguales, por ende no podemos sumarlos y tampoco multiplicarlos. El problema principal de este error surge porque no sabíamos utilizar los numpy arrays con tanta facilidad, sino que gran parte del trabajo consistió en entender el funcionamiento del módulo numpy.

Otro de los problemas fue la saturación del audio, dado que no arrancamos cada nota en 0 sino que cuando realizamos la suma de los armónicos usábamos el tiempo de inicio de la nota como tiempo de inicio para generar el array. Por ende, se escuchaba todo superpuesto y no cobraba sentido. Además, no sabíamos que la duración de la nota no incluye el tiempo de decaimiento, entonces ese fue un error que se nos acumuló a que suene mal el audio. Cuando corregimos esos errores, debimos buscar una amplitud para multiplicar al array que sea correcta para que el audio no se saturara.

Seguido a ello, la parte de modularización de la nota, en especial la parte de decay fue otro problema, porque al trabajar con numpy arrays, algunas de las funciones matemáticas tuvieron que ser redefinidas. A lo largo del desarrollo de esta parte nos surgieron algunos Runtime Warnings que tenían que ver con cómo estábamos calculando las funciones, dado que estábamos evaluando el array entero y no solo en la sección que necesitábamos.

Finalmente, un error importante fue el poder comprender el uso de parser en el código, puesto que en un principio intentábamos pasar las listas con las XyloNotes como argumentos en lugar de generar el código en la misma función main(). Asimismo, poder establecer el MockServer y manejar dos terminales en principio nos generó errores.

Indicaciones para ejecutar correctamente el programa

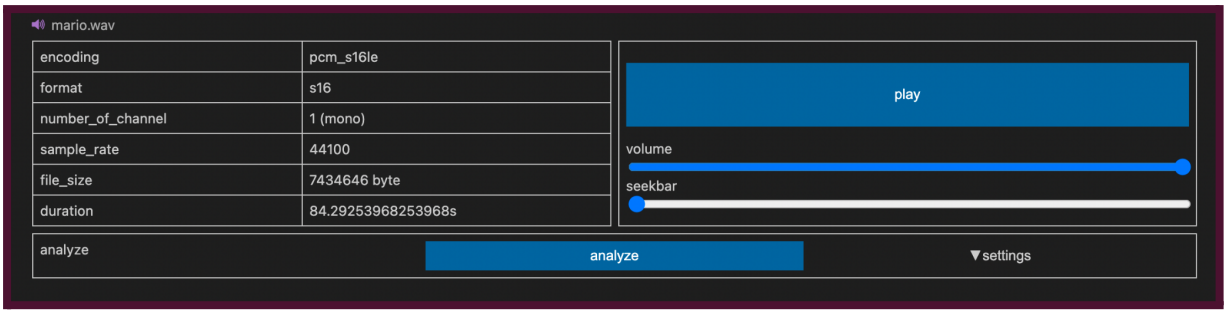
En primer lugar, para que el programa sea ejecutable, el usuario debe contar con un dispositivo con un intérprete de python. Una vez asegurado esto, el usuario debe clonar el repositorio en una ubicación de la computadora a usar. Después de esto, el usuario debería leer el archivo README.md para tener en cuenta los requisitos del proyecto y cómo correr ciertas partes del código.

Para ejecutar el programa se debe contar con las carpetas correspondientes como también con un archivo midi y otro de texto donde se guardarán las partituras. Además, se necesita el archivo del instrumento.

Dicho eso, es necesario posicionarse sobre la carpeta "convert_midis" y utilizar la consola para obtener las partituras de la canción. Aquí el camino se divide en dos, para obtener su versión 'digital' se debe volver a usar la consola, esta vez posicionado sobre 'src', se le pasa la partitura obtenida, el instrumento, la frecuencia, y el nombre donde se escribirá el archivo wav. Dados unos minutos se creará el archivo y podrá reproducirse correctamente.

Por otro lado, si la intención es pasar el archivo al xilofón metalizado, nos debemos posicionar en 'server_load' y pasar por la consola el archivo con las partituras junto al IP al que nos queremos conectar.

Resultado de ejecución



Resultado de ejecución normal cuando se siguen los pasos estipulados. Archivo .wav

Bibliografía

Newest “python” questions. (s/f). Stack Overflow. Recuperado el 6 de julio de 2022, de https://stackoverflow.com/questions/tagged/python