Web Audio Project : Simple CataRT

To conclude this introduction to the Web Audio API you will implement a simple version of CataRT (concatenative real-time sound synthesis system). Here is a brief explanation from the original DAFx publication :

> *"[CataRT] plays grains from a large corpus of segmented and descriptor-analysed sounds according to proximity to a target position in the descriptor space. This can be seen as a content-based extension to granular synthesis providing direct access to specific sound characteristics."* (Schwarz et al., 2006) .

In our case, the descriptor plane is a 2D plane where every sound blocks are scattered according to two sound descriptors :

- ➔ the Zero-Crossing Rate (ZRC) on the X axis;
- ➔ the Audio Power (RMS) on the Y axis.

For a detailed explanation of the descriptors, see Geoffroy Peeters Cuidado article.

You will find the project folder in the Git repository. Don't forget to install the dependencies with *npm install* and start the server with *npm run dev*.

Blocks analysis (14 points)

Q1.1 getTimes (3 points)

Code the *getTimes* function and store the blocks starting times in the *data* object.

Q1.2 rms (4 points)

Code the *rms* function and store the blocks RMS values in the *data* object.

Q1.3 zeroCrossing (4 points)

Code the *zeroCrossing* function and store the blocks ZCR values (expressed in Hz) in the *data* object.

Q1.4 normalize (3 points)

Code the *normalize* function and store the blocks normalized descriptor values in the *data* object.

Find the closest grain from the user's position (3 points)

Q2 findStartTimeFromGuiPosition

findStartTimeFromGuiPosition returns the time of the closest block from the user position in the GUI. Code this function.

Concatenative engine (3 points)

Q3 ConcatEngine

Implement the concatenative engine. The concatenative engine is very (very) similar to the granular engine, except that the position in the buffer is determined by the position of the user in the 2D space (using the *findStartTimeFromGuiPosition* function). The sound must be played only if the user is maintaining the mouse button on the interface. Use *console.log* to understand how the GUI works (the interaction between the interface and *globals.guiPosition* is already implemented).

Finally, create the scheduler, create the engine and add it to the scheduler.

The script *index.js* must be sent before **January 23, 2022 at 23:59** to Benjamin.Matuszewski@ircam.fr and Victor.Paredes@ircam.fr. You must comment your code to make it understandable by others.