BoiteARItmac

Julien Calabrese
Elise Gondange
Léo Genot

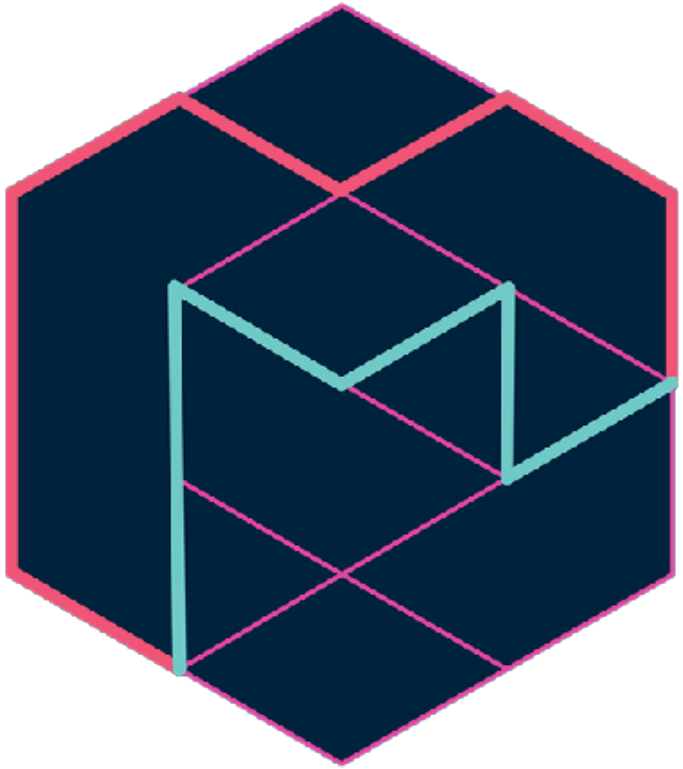
Esthétique Algorithmique

PLAY

I) Concept

First, we had to choose which library we'd like to use. As we did lots of examples with images on mobileNet, we had no idea about what we could do that was interesting and different enough from the example we did in courses. Then, we easily understood that magenta.js was about music creation.

Some of us are musicians and used to compose songs on computer-generated music software like FL Studio or Pro Tools. One important aspect of the music creation process is to make drums. The composition sequence depends on the style of music that you are creating, but each style has classicals drum patterns that are used quite often. In fact, producers are just making some variations into those patterns.
Because of this, we've asked ourselves if we could make a tool which is using some classic drum patterns to create variations of them.
Thanks to magenta, we have decided to create the BoiteARItmac, which is creating 3 different styles of drums, which are the most used by music producers on CGM softs : Electro, trap, and two-step (equally called UK Garage).



II) Algoritmic approach

The first step was to find an AI that was able to play a drum pattern following a pre-saved drum pattern. Once we found it, we added some elements to its practice to perfect it.
This AI was created to follow a single pattern at a single tempo. This pattern wasn't in the styles that we wanted.
So we had to create the 3 patterns that we wanted, to let our AI play with it. Then, we had to create a switch between those patterns and the different tempos that they are using. When the user clicks on a style button, the IA plays this style with its typical tempo.

BPM switch

```
function playPattern(pattern) {
    switch (seedPat) {
        case seedPattern:
            bpm = '8n';
            break;
        case seedPattern2:
            bpm = '10n';
            break;
        case seedPattern3:
            bpm = '16n'
            break;
        default:
    }
    sequence = new Tone.Sequence(
        (time, { drums, index }) => {
            drums.forEach(d => {
                drumKit[d].start(time)
            });
        },
```

One of our three pre-saved patterns

```
var seedPattern = [
    //Pattern Electro
    [0, 2, 5],
    [2],
    [0, 2, 6, 3],
    [2],
    [0, 2],
    [1, 2],
    [0, 1, 2, 6],
    [2],
    [0, 2],
    [2],
    [0, 2, 6, 3],
    [2],
    [0, 2],
    [1, 2],
    [0, 1, 2, 6],
    [2],
];
```

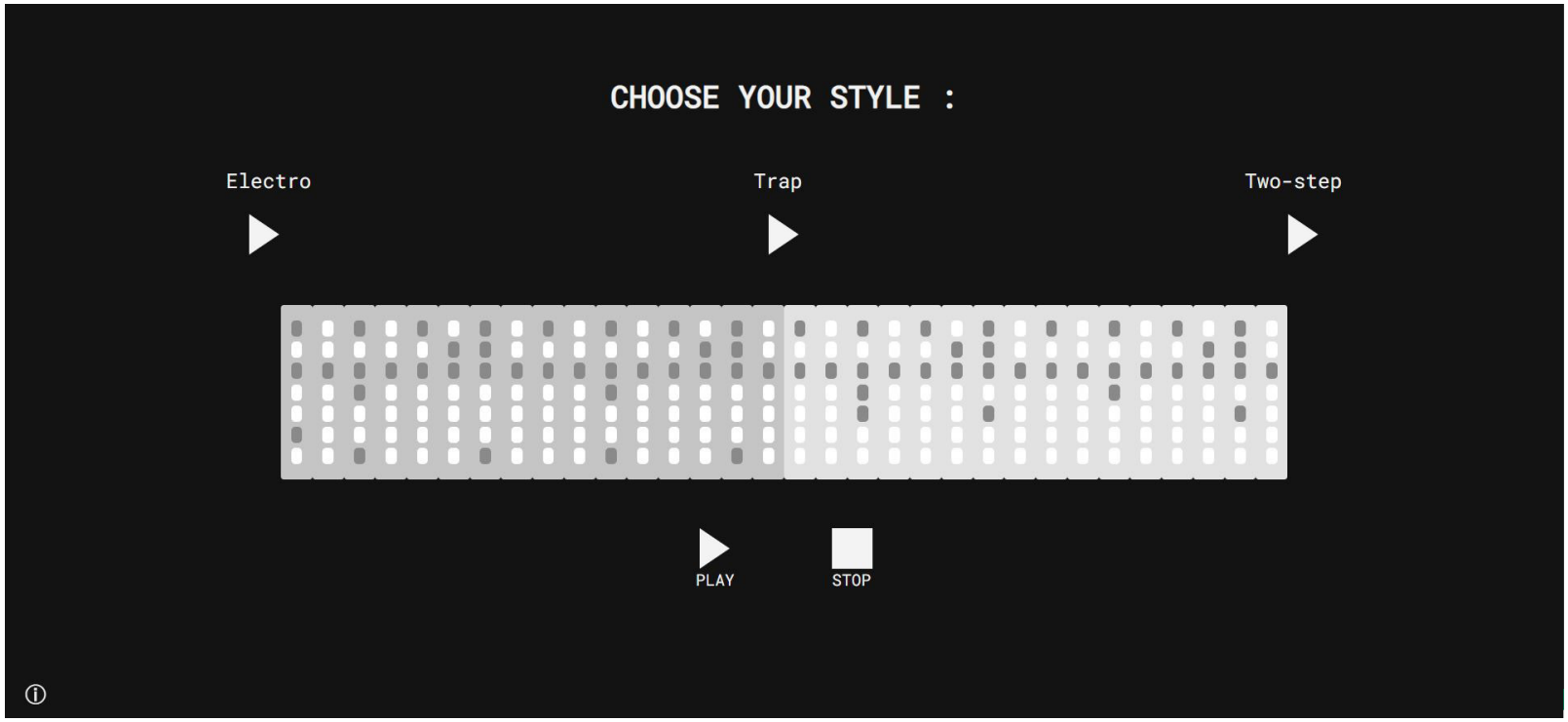
AI practice to create variations

```
const midiDrums = [36, 38, 42, 46, 41, 43, 45];
const reverseMidiMapping = new Map([
    [36, 0],
    [35, 0],
    [38, 1],
    [27, 1],
    [28, 1],
    [31, 1],
    [32, 1],
    [33, 1],
    [34, 1],
    [37, 1],
    [39, 1],
    [40, 1],
    [56, 1],
    [65, 1],
    [66, 1],
    [75, 1],
    [85, 1],
    [42, 2],
    [44, 2],
    [54, 2],
    [68, 2],
    [69, 2],
    [70, 2],
    [71, 2],
    [73, 2],
    [78, 2],
    [80, 2],
    [46, 3],
    [67, 3],
    [72, 3],
    [74, 3],
    [79, 3],
    [81, 3],
    [45, 4],
    [29, 4],
    [41, 4],
    [61, 4],
    [64, 4],
    [84, 4],
    [48, 5],
    [47, 5],
    [60, 5],
    [63, 5],
    [77, 5],
    [86, 5],
    [87, 5],
    [50, 6],
    [30, 6],
    [43, 6],
    [62, 6],
    [76, 6],
    [83, 6]
]);
```

III) Aesthetics

Our goal was to create a really simple tool to use and understand for music creators with CGM software. We were hugely inspired by the drums rack in the famous software, FL studio. Thanks to its simple aesthetic, users can see which instrument is being played in the pattern.
To make a difference between the pattern that we have saved and the pattern which is created by the AI, their colors are different. The AI's one is lighter than the classical one.
There is a total of 32 times played, 16 that we pre-saved, and 16 created by the AI.
Our entire web page is dark, and each important elements are lighter, to make the UI fast and easy to use.

FL Studio drum patterns aethetic



BoiteARItmac aesthtics

IV) Documentation

We divided tasks between us. Julien would work on the music and melody aspect of the program, ensuring that everything that was being played was actually sounding good and perfectly timed. He created the different patterns, notes/ pitch matrix, and setting the perfect BPM according to the drums style. He also linked real music assets to patterns.
Elise worked on the visual aspects of the UI and On integrating the AI and DrumRNN in the code. Having a very basic and yet easy-to-use UI was our main objective, we wanted the user to have really quick access to generated drum patterns to help in his creation process.
And Léo worked on linking Julien's and Elise's work, making sure that the perfect Julien's patterns were actually being played and that the AI-generated a new sequence from those patterns.

We encountered a lot of problems and bugs in this project. The main difficulty was the almost non-existent Magenta documentation, we had to rely on other people's sources, which we wanted to avoid to create uniqueness in our code.
The other major bug was the fact that the AI is server and promises based programming, meaning that a lot of Js functions arent easily applicable to it, for example, once the pattern is generated it is almost impossible to generate a new one without stopping the actual sound playing, which makes the user interactions a bit chunky.
And the last issue, which is related to the lack of documentation, was the fact that we actually had a very hard time building the notes variations matrix, we needed to fill it by hand and ear to have a good sounding sequence because it is never ever explained anywhere how to build this type of matrix.

Thank you for your attention, access to the code: https://github.com/leogenot/MagentaMusicAI
If you incounter problems to make the project works with GitHub, here is a demo on Youtube: https://youtu.be/VCN_g7Di6wA