HARMONIC COMPLEXIFIER

Nicolò Chillè

Rocco Scarano

Michele Murciano

Main idea:

- Develop a sort of didactic tool that allows inexperienced musicians to generate intricate chord sequences
- Add up to 5 level of complexity to a commonly structured 4-chords sequence, given as input

Features:

- Six possible "complexifiable" sequences
- Seven possible chord types as input
- Preset selection
- Level of complexity selection
- Chord sequence playback
- 2 different playback patterns
- 3 levels of playback speed
- Guitar tablature for the outputted chords
- Midi file export of the "complexified" sequence

Complexifying the sequence:

1. Find the root of the sequence
2. Find the most similar sequence
3. Complexify the sequence

Root finding:

- **Take all the chords from the input**
- **Check if one of the notes is the root of the sequence**
- *I.E: at least 3, belong to the key of that note*
- **If none of the notes is the root, check all the other possible notes**
- **If a chord isn’t in scale, it is pointed out to the user**

Sequences:

- I II V I
- I IV V I
- I V IV I
- I VI V I
- I VI II V
- I V VI IV

The program can also work with slightly different sequences, reducing them to one of these with which it can work best.

The sequence can be entered either by inserting the key and the preset, or by having it recognized by the program itself from a circle of chords.

Levels of Complexity:

Substitutions are cumulative

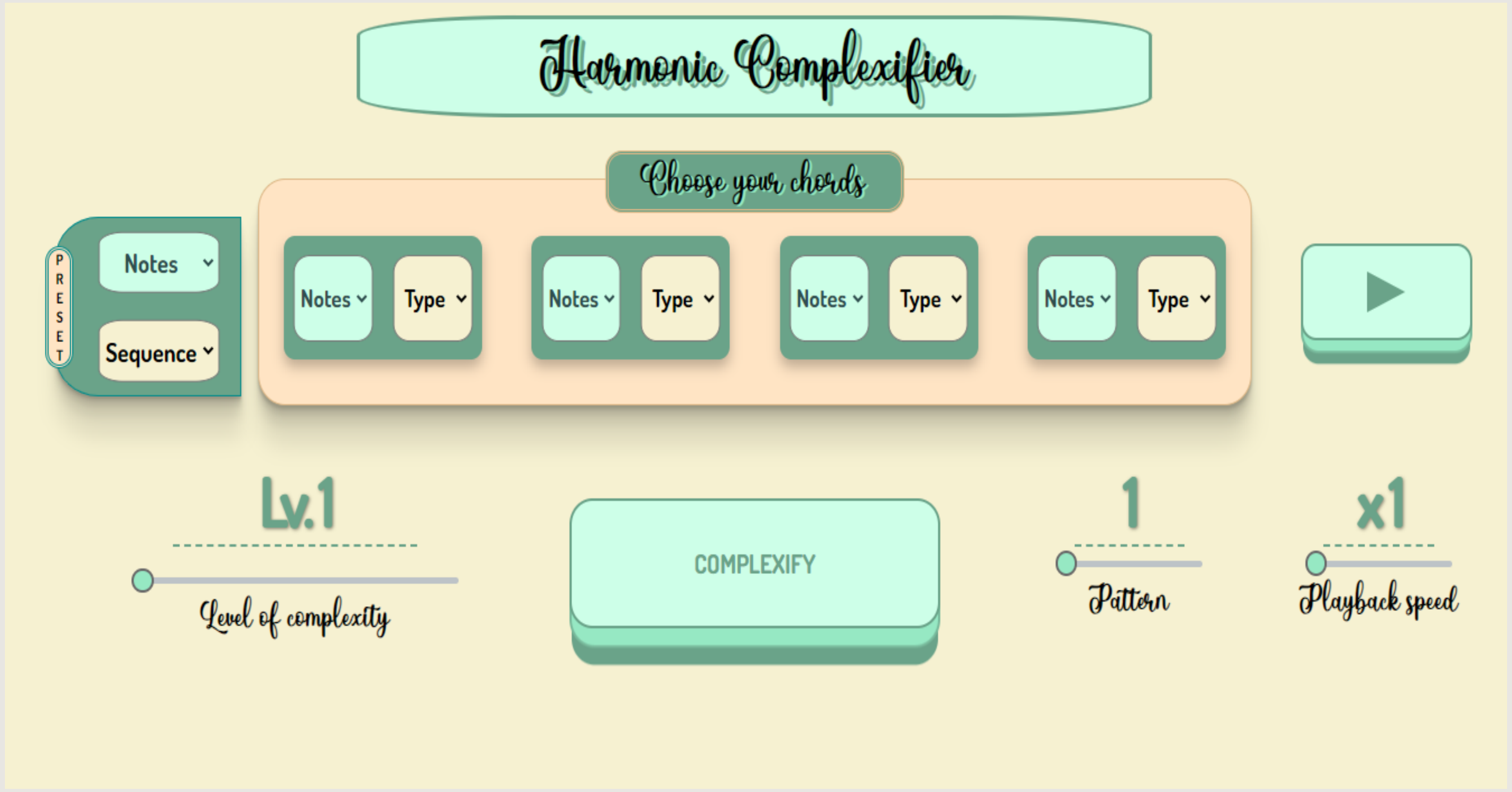
1. **Transition from triads to tetrads**
2. **Relative minor substitution**
3. **Secondary dominant substitution**
4. **Parallel minor substitution**
5. **Tritone substitution**

Audio playback:

- Implemented using *"Howler.js"* audio library
- Sound source created by using *".wav"* audio files
- Each file plays a chord for the duration of one quarter
- Concatenated according to the chord actual duration
- Played back in a loop using the *"setTimeout()"* function, with the chosen speed

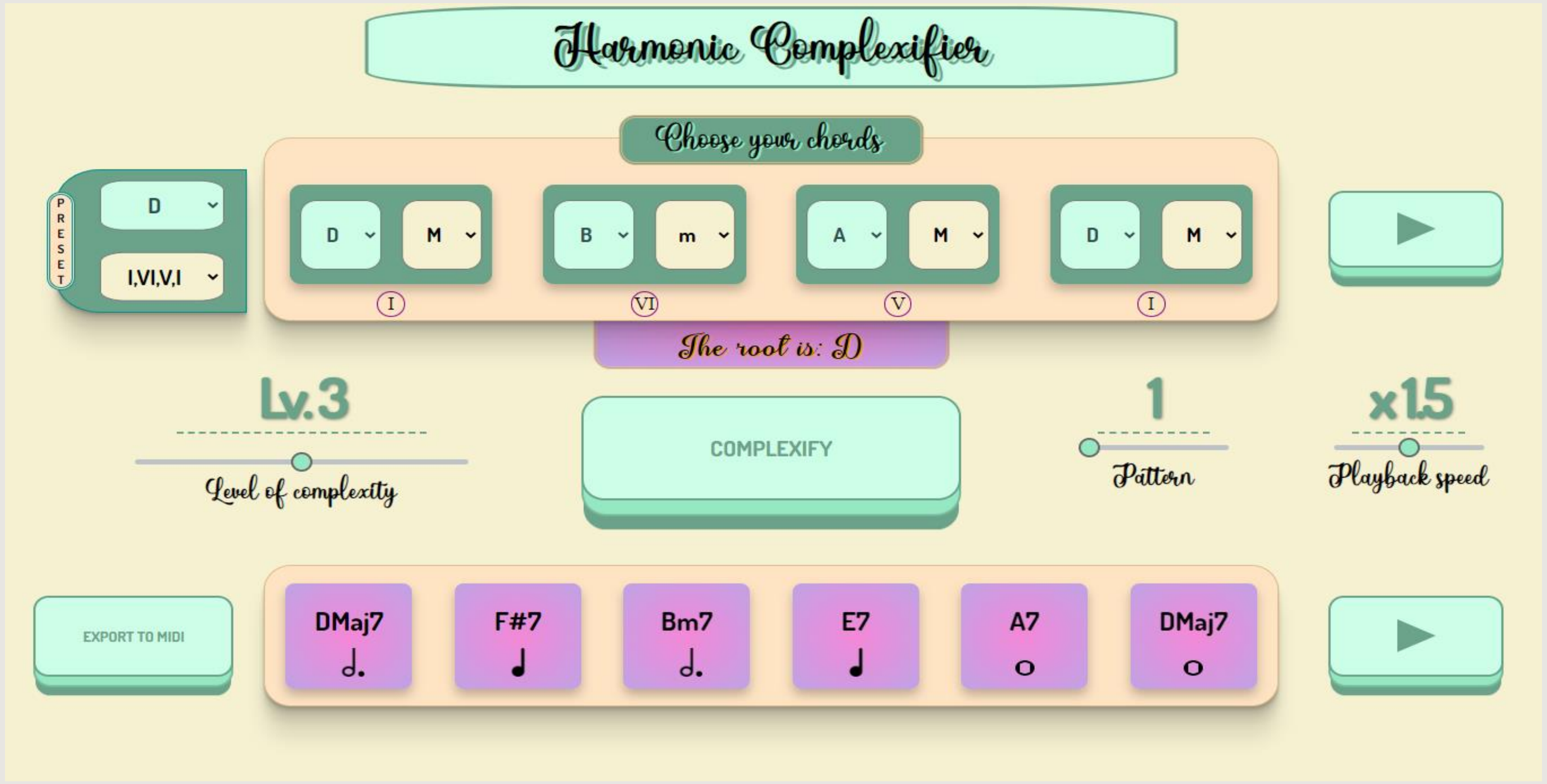
Graphical User Interface

- Initial state



Graphical User Interface

- After user interaction



PRESET: D, I,VI,V,I

D M	B m	A M	D M
I	VI	V	I

The root is: D

Lv.3 Level of complexity

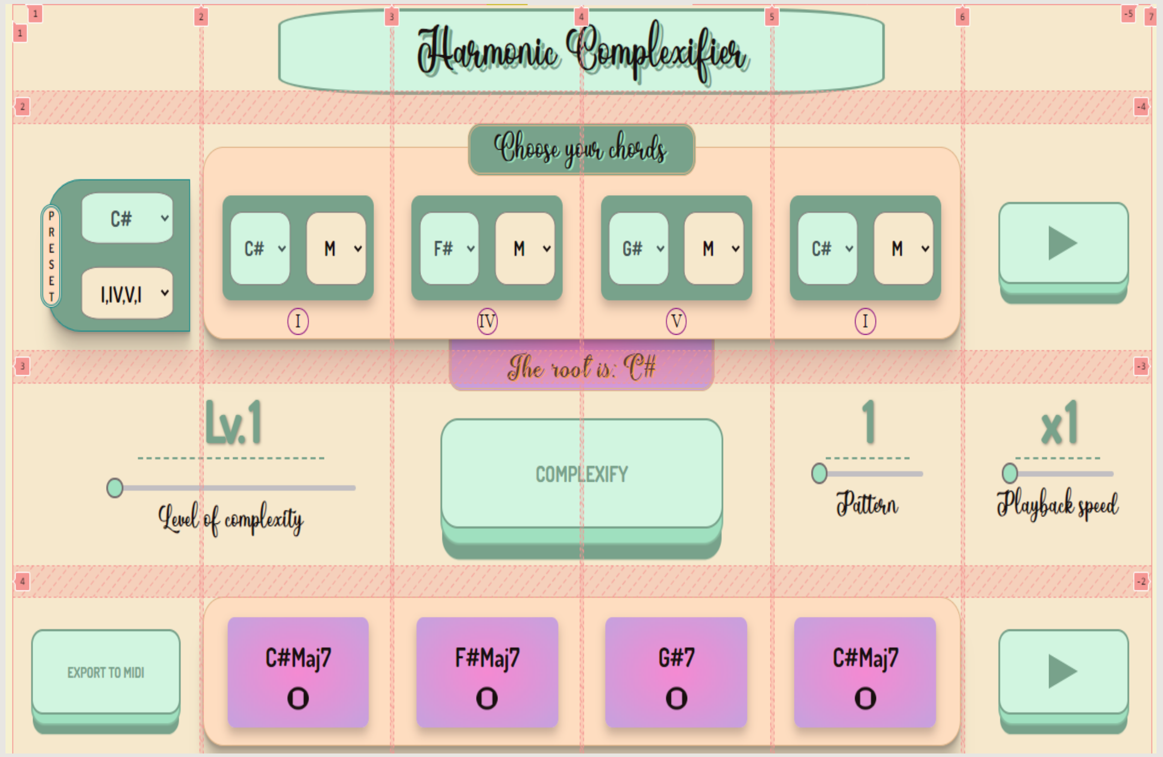
1 Pattern

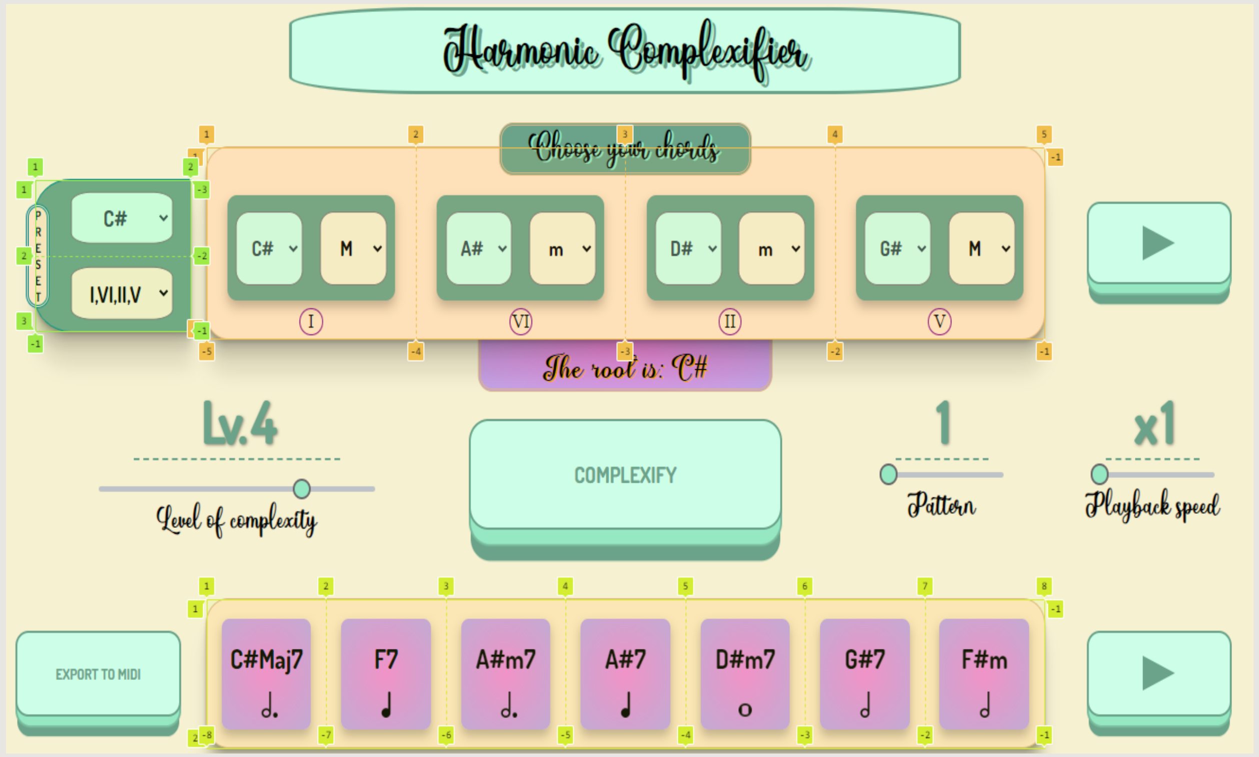
x1.5 Playback speed

DMaj7	F#7	Bm7	E7	A7	DMaj7
𝅗𝅥.	♩	𝅗𝅥.	♩	𝅝	𝅝

Graphical User Interface

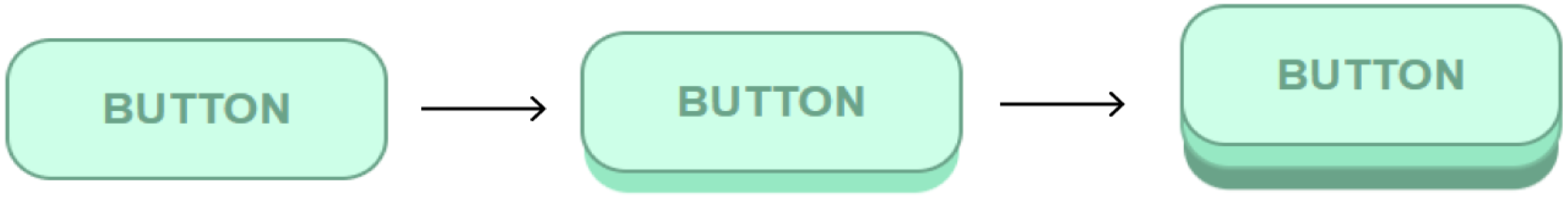
- CSS Grid Layout





Graphical User Interface

- **3D Button implementation**



```
.big-button
```

```
.big-button
+
.big-button::before{
   ...
   transform: translate3d(0, 0.55em, -1em);
}
```

```
.big-button
+
.big-button::before{
   ...
   transform: translate3d(0, 0.55em, -1em);
   box-shadow: 0 0 0 2px var(--colorB),
                        0 0.55em 0 0 var(--colorA);
}
```

Midi Export:

- **Implemented using** *"midi-writer-js"* **library**
- **Extract, for each chord of the output sequence, the notes**
- **Create a noteEvent of its duration**
- **Add it to the midi track**
- **Once done, generate a download url of the midi track**

Conclusions

- Successfullyin guides a beginning musician in music theory
- Encourages him to evolve and vary using the level of complexity and transformations he likes best.
- Helpful to understand and learn new chords, by using the guitar tablature and the midi export function

Ideas

- Further increase the sequences with which the program performs best
- Increase the number of musical instruments with which you can hear the result
- Being able to visualize how to play the output chords with different musical instruments