Hiraku - 開曲

06/02/2019

—

Davide Gioiosa

Davide Dal Cortivo

Milano (MI)

Owners and Rights

The project was entirely realized by **Davide Gioiosa** and **Davide Dal Cortivo**, graduates in Computer Engineering and students of Music and Acoustics Engineering at *Politecnico di Milano*. To them belong all the rights of the application.

Goals

Have you ever thought about join a live performance with an artist adding in a song the riff that you have in mind? Or take part of a session in another part of the world sending your ideas?

HIRAKU, is a project created and based on the idea of communication with music through the use of internet. Started from a first idea of Davide Gioiosa, HIRAKU (**開曲**) means **open melody** or **open composition**, representing the idea music creation with shared ideas. It has been developed by a team of two members: **Davide Gioiosa** and **Davide Dal Cortivo**.

The web-application has the main goal of connect people's ideas, in whatever place they are, and create new music and infinites performances scenarios.

Specifics

The user interface is written in HTML and CSS, the logic in Javascript. This project includes several external libraries, that are:

1. **jQuery**

   to use particular Javascript constructs
   https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js
2. **AudioContext monkeypatch**

   for more browser compatibility with AudioContext specifics
   https://cwilso.github.io/AudioContext-MonkeyPatch/AudioContextMonkeyPatch.js
3. **Bootstrap Notify**

   for the graphics in the notifications
   https://cdnjs.cloudflare.com/ajax/libs/mouse0270-bootstrap-notify/3.1.5/bootstrap-notify.min.js

4. **WebAudioFont**

   where instruments come from
   https://surikov.github.io/webaudiofont/npm/dist/WebAudioFontPlayer.js

5. **Animate.css**

   for the animation of notifications
   https://cdnjs.cloudflare.com/ajax/libs/animate.css/3.2.3/animate.min.css

In addition, a part of Bootstrap Glyphicons library (for notification icons) and an extra font (Roboto Condensed) are also be imported.

Particular attention must be given to next topics included in the project:

1. **Firebase**

   used for his real time services

2. **Dropbox**

   where loops come from

3. **Midi support**

   it gives the possibility of connect a midi instrument like a keyboard

4. **Touch support**

   for a mobile experience

5. **Web Worker**

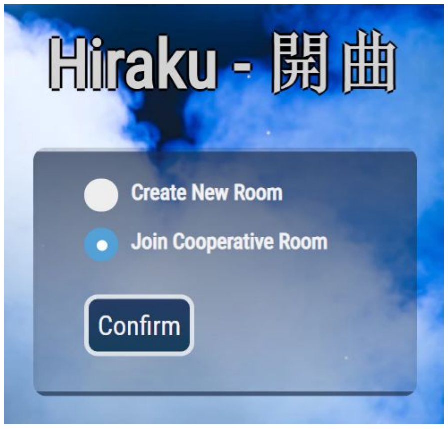
   used to create a metronome that will be independent of the main thread

Room: Create / Join

The program present an initial screen with the choice between two options:

1. **Create New Room**

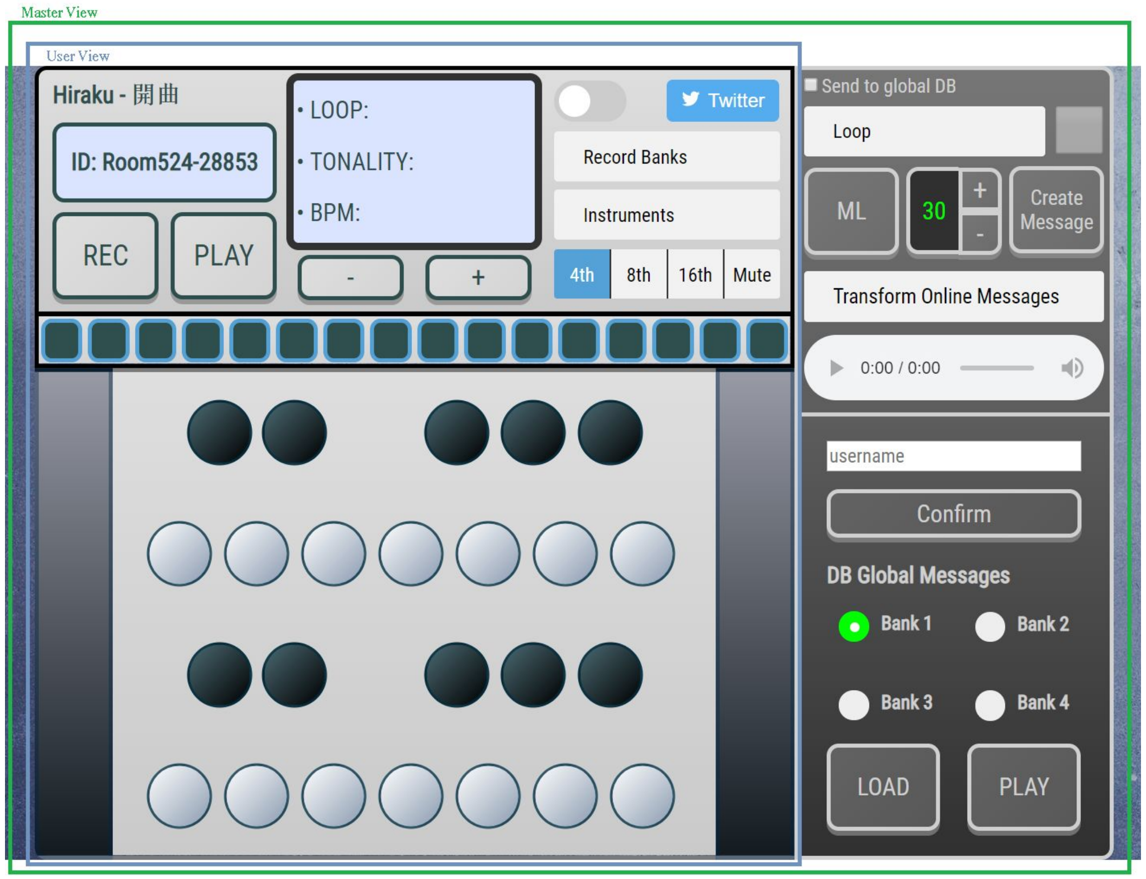
   The option leads to the insertion of the nickname and the creation of a new room in which the user is the owner and Master. The room will be allocated into the database with an unique *ID* and will contain information about musical messages, loop selected and datas about time



synchronization. Then the user access to the application, which provides the possibility to use the keyboard and the control panel of room's settings. Using these tools the user will create and control his *music collaborative session online*.

2. **Join Cooperative Room**.

The option leads, like the previous one, to the choice of the nickname and is followed by the insertion of the ID of the room in which it wants to participate, which must be provided by the master. Subsequently, the user accesses the Guest interface containing the keyboard, through which he can communicate and compose messages to be sent to the master. In addition, the room's control parameters will be managed by the owner of the room and changed in real time on the screen of the user (for example the loop selected).



Master and User Views in Hiraku

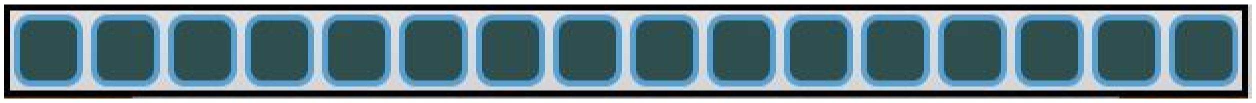
Concept of Musical Message

HIRAKU introduces and is based on the concept of **musical message** (**MM**). The program, after pressing the rec button, processes what is played by the user on the keyboard-interface and at the end of the recording generates, through a conversion algorithm, a message. This will be the transcription of the performed performance, ie a music sheet that can be translated by the program in order to be played.

This encoding / decoding technique allows to record and stored the messages, the listening and the sending of a music composition performed on the application.

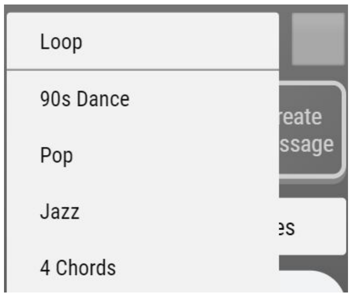
Graphic responsive - Progress Bar

This bar represents the length of a loop, expressed in pulses. The squares turn on one by one following the rhythm and the "on-color" changes related to the time signature of the loop. For example, if we have a loop composed by 4 beats and a 5/4 time signature, we will have a bar composed by 5*4 squares, and the "on-color" changes every 5 squares. It's created dynamically in Javascript when a new loop is selected and it's controlled by a Web Worker.



Loop characteristics and buffer Audio - Tonality Limitations and Pro version

Loops have been created with FL Studio and some VST. They are stored in a Dropbox account and, when one of them is selected, it will be decoded in an audio buffer ready to be played. The attribute "loop" is setted true in order to obtain a "loop" effect.

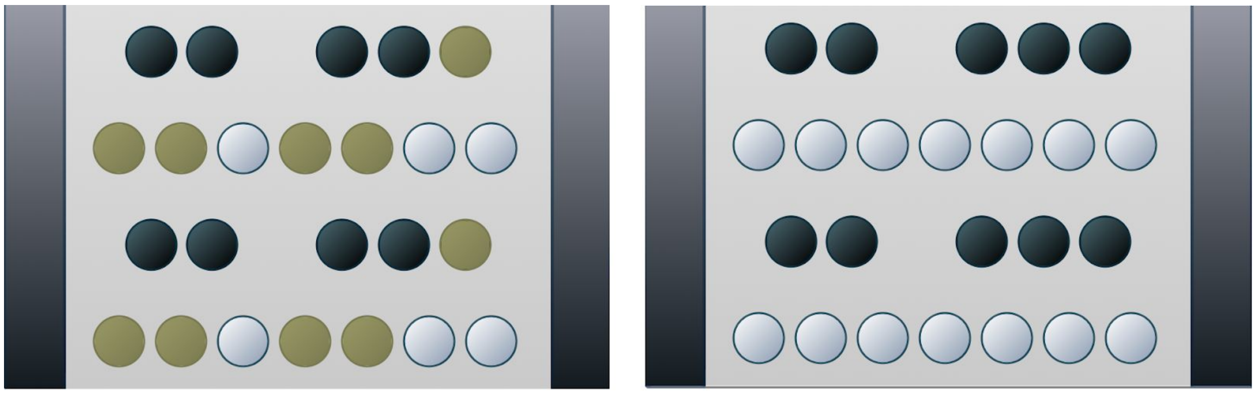


FL Studio / VST links
https://www.image-line.com/flstudio/
https://refx.com/
https://www.native-instruments.com/en/products/komplete/samplers/kontakt-6/

A loop contains a lot of information saved on Firebase (duration, tonality, time signature, bpm etc…). This information is very important, because it modifies several aspects in the program such as the progress bar.

This program isn't designed only for musicians, so the tonality of a loop puts restrictions on playable notes. This is to limit wrong notes during a registration on a particular loop. If you are a musician don't worry, you can switch to "PRO" mode and all notes will be available again.

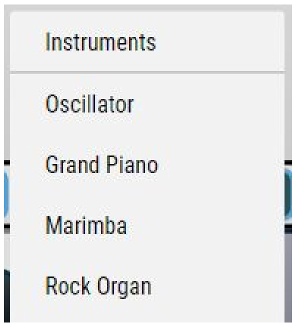
A function calculates the degrees of the ionian scale of a particular tonality to decide the playable notes. If the tonality is minor, it refers to his relative major for the calculus.



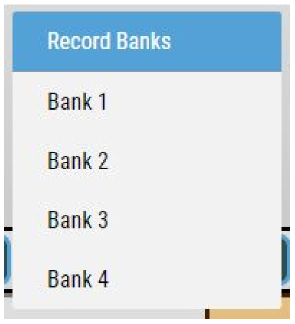
Example of difference of Tonal Mode (E in this case) and Pro Mode on keyboard.

Instruments

Instruments are imported from WebAudioFont library except for the Oscillator that was created directly in Javascript. We have chosen them thought to be played in a solo performance on a loop, so they had to be versatile with a pleasant middle-high register sound. You can play only from C3 to B6 on the PC keyboard, but you can extend the range from A0 to C8 (like a piano range) with the connection of an external midi controller.



Master Records, Local Banks, Metronome, Worker



The "REC" button starts a registration. The length of a registration is equal to the length of the selected loop and it starts at the beginning of the loop. The registration is sent to the master and saved in one of four record Banks, the one that is selected. You can listen your registration with the “PLAY” button or overwritten it with a new registration on the same record Bank. A recorded message is modeled with an array of objects having a structure “note, frequency, duration”.

A user guest can record only when a loop is playing, a master also when it’s not. With the “check” on “Send to global DB”, a master can send a recorded message to global DB that can be loaded and played by another master at any time.

Metronome and Web Worker

The metronome is managed by a Web Worker that schedules in the future the “ticks” that will be played with an oscillator. You can play 4th notes, 8th notes, 16th notes or you can turn off it with the “Mute” button. Web Worker controls also the illumination of the squares in the progress bar.

We have chosen to use a Web Worker in order to obtain the better possibile precision of the “ticks”, because this program is based on record messages played on a loop.

Functions related to metronome and Web Worker were taken from the follow link, where there’s also a tutorial that explains how you can create a very precise metronome. Obviously, the functions have been modified for our purposes.
https://www.html5rocks.com/en/tutorials/audio/scheduling/

User synchro-records with master, metronome-Worker and local banks

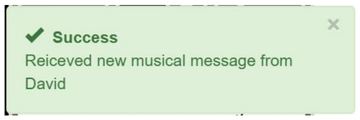
After several attempts in order to synchronize recorded messages with loops, a very good result has been achieved. Firstly, we tried to execute any action in a single thread, but it caused lag: the continuous update of the progress bar became less precise over time. Moreover, recorded messages are played in real time on a loop and it also caused lag.

Subsequently, we decided to create an audio type Blob from a recorded message (see “Blob-mp3 of MM Received” section for more information) and manage the metronome and the progress bar with a Web Worker (see “Metronome and Web Worker” section for more information). These changes increase considerably the synchronization between recorded messages and loops.

MM, send and reception with Firebase

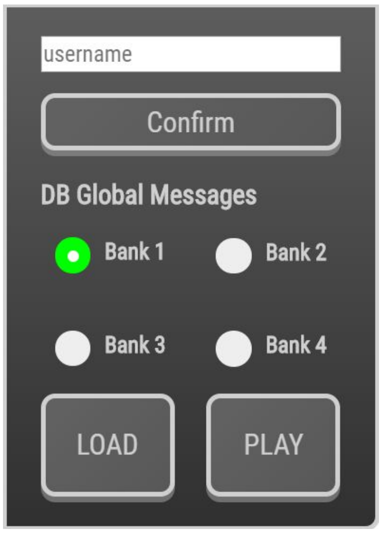
When room's master or an user logged into the room completes the creation of a musical message on the selected loop, the MM is send directly on the database and received in realtime by the master, that is listening on database room's updates. The operations are notified in the application with notifications (which will be discussed later). Once received, the musical messages are decoding by the translation algorithm and ready to be played.

Success
Musical Message sent to the database


Success
Reicevend new musical message from David

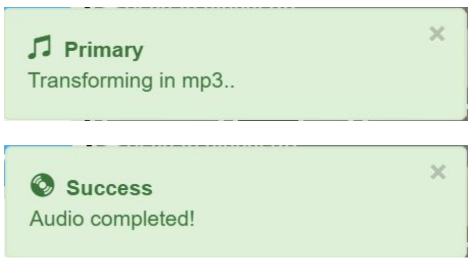
MM Global

As mentioned in a previous paragraph, Room's Master have also the possibility to send a musical message created to a global database container. This operation will permit to every other master of other rooms to get and play the global uploaded MM. In fact in the panel controls of the room exists the database's Global Messages Area: the user inserts the username of the owner of the message that wants to load in the application, choose the bank where the message was stored (one of four) related to the account searched, loads it and then the MM can be played applying the decoding and translation algorithm. This function allows a global communication channel between users with the use of MM.



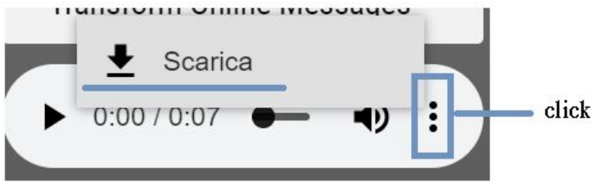
Blob-mp3 of MM Received

Once the master has decided to perform the messages that he has received, he will hit the button *"Transform Online Messages"*. Starting from the collection of all MM received, the web-application will play them one by one internally, in a Media Recorder, saving the informations in chunks with whom is created an audio type Blob. From that it will be created an audio with the url containing the informations.



This operation provides to create an audio with all the MM that will be played simultaneously to the song loop on which they were recorded. The choice of creating this audio instead of playing the messages under the loop was made to maintain the delay at the lowest level possible. After the conversion has been completed, the master will receive a notification and all will be ready for the performance.

This conversion of messages adds a new feature to the web-app, that is the possibility to download the performed message in mp3 format (using audio/mpeg conversion in blob), allowing the use of the records also outside the application.



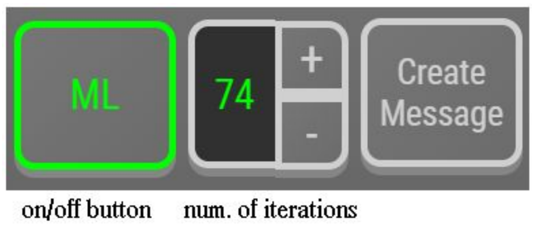
Play loop synchronized with MM received

After the creation of an audio type Blob, loop and messages are ready to be played together. So when the loop is restarted, the audio containing the messages will be played in synchro creating the effect of a collaborative improvisation on a music piece.

Markov-Chain and Bot Riff Creation

One of the main features of *Hiraku* is the **Machine Learning Algorithm** which allows the application to achieve the ability of creating new musical messages inferring from the MM that has received from the users that have participated to the *join session,* clustering all the received informations.

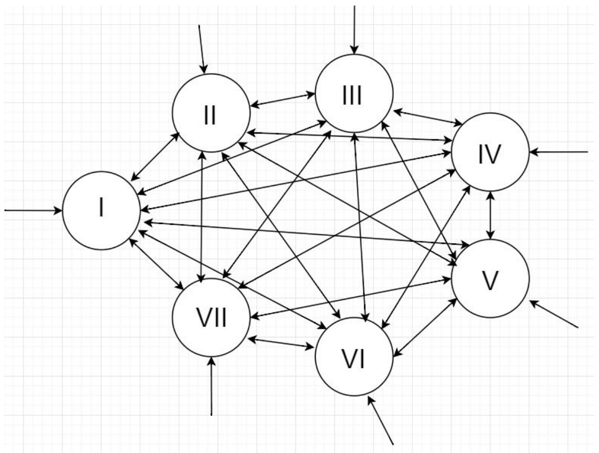
The algorithm has been created entirely starting from 0 and its reasoning creation process is based on Markov-Chain principles.

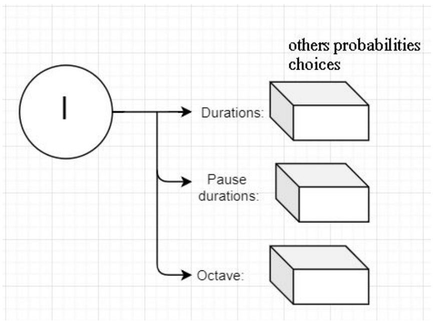


If the Machine Learning button is active, all no pro MM received will feed the Markov Chain structure in the program. The data structure is based on informations about notes grades, intervals, octave changes, time durations, pauses and the algorithm provides to create probabilistic branches based on all the information contained in the messages. The Markov Chain keep updating whenever a new message arrives, and the more informations it has, the more it can extract an interesting style of playing.

After setting the number of iterations that the algorithm will do into the MC, clicking the *'Create Message'* button, it will be constructed a new MM based on the probabilistic information path and at the end applying duration cuts to allow the correct fit of the message. Then, the new computer-created MM is added in the queue with the user's ones, ready to be converted into the audio mp3.

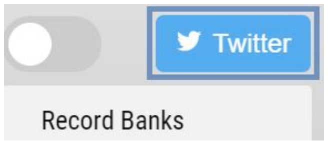
In the graphics is showed the Markov Chain structure implemented into the program: is based on the first level on a probabilistic choice between notes grades sequences - in the first picture every arc that starts from a Note Grade has his own probability, and the sum of all the probabilities arcs that starts from that note is 1. Then there are further levels on the other informations explained before, such as duration, pauses and octaves changes, with their own probabilities. Having this structure the bot riff generator receives the number of iterations to make into the structure creating a musical message based on all the previous MM received by the user. The result is that the master can have an infinite message-generator that can be used to improve his live performance, in addition to the message that can receive from other users.

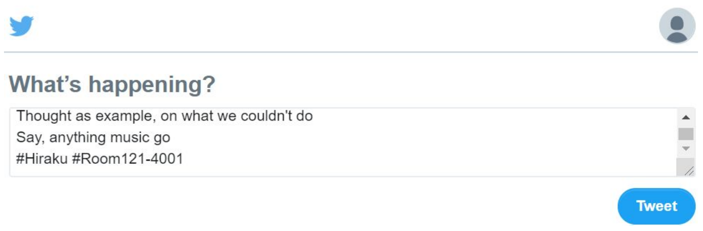




Twitter text conversion, Python Crawler

A further feature of *Hiraku* is the possibility to share the MM created on Twitter in a text-version format clicking on the button on the top right of the keyboard.





Thought as example, on what we couldn't do
Say, anything music go
#Hiraku #Room121-4001

In fact after the creation and submission of the message played on the keyboard, there is an algorithm that converts locally the MM created in text-lyrics, using an hashmap association with words: informations such as note-duration are the keys to log into the structure composed of texts belonging to famous musical songs (such as *Red Hot Chili Peppers, Dire Straits…*). The relation returns a word for every note and with the combination of all of them a pseudo music text is created.

This feature permits to store into Twitter timeline all the messages received in a session, creating a memory record of the performance.

Furthermore this option was designed to allow the possibility to send a MM for a master without using the *Hiraku* application directly. In fact an user could send a text message on Twitter with a final hashtag *#hiraku* and through the use of a python program running on a room master computer, the owner of a room can receive MM also in this format. At the moment the python is completed, and then this conversion feature will be one of the future updates of the program.

Error check - Notifications

Every important action is notified by an alert message on the top-right of the page. The notification graphic has been realized thanks to Bootstrap library. It wasn't completely imported, because it would have changed a lot of graphic already implemented. So, the CSS part of Bootstrap related to alert message was simply copy-pasted in the CSS of the program. A couple of Glyphicons has been imported and used in the notifications. They represent the type of interested notification (warning, message, info etc…). The animations have been realized with Animate.css.

Conclusions

Hiraku was designed as a scalable application and contains already feature that will be implemented in the future in the next updates. We would like to thank our university professors for the acquired knowledge and support during the development.

Our next goal is to extend the program with other ideas that we have in mind, and present our project in musical events (an opportunity can be the *FestiValle* in august 2019).

Project Link: http://hiraku.surge.sh/

The Hiraku project team,

Davide Gioiosa

Davide Dal Cortivo