Università
della
Svizzera
italiana

**Software
Institute**

RUSE

Influencing Live Concerts with Real-Time User Feedback

Andrea Brites Marto

June 2021

Supervised by
Prof. Dr. Michele Lanza

Co-Supervised by
Dr. Roberto Minelli

BACHELOR PROJECT

Abstract

In a live electronics music performance, the artist plays the music and wants to give a great experience to his audience. In it, the artist is not aware of what is going on in the audience's mind, and, at the same time the spectators can not provide their feeling to the composer. The goal of the project is to develop a system where the spectators of a live music performance can participate actively by providing real-time feedback to the artist. To represent the general perception of the concert we introduced the *feedback parameters* such as slow vs. fast and consonant vs. dissonant. By using a web application the audience expresses their opinion on six different perceptual music parameter pairs. We implemented this approach in RUSE, a web application which aggregates all the data and presents them, with an intuitive visualization, to the artist who can influence in real-time the sound synthesis and compositional processes. Furthermore, the data is available after the live performance to allow further analyses. As of now, the project is a real and working service available at `https://ruse.si.usi.ch`.

Contents

Chapter 1

Introduction

In a live event, which is a live electronics music performance or a music concert, there are two actors involved, the artist who plays the music to give the best possible experience, and, the audience who listen and enjoy the music. In it, the artist has not a real-time interaction with the audience and does not know what is going on in their minds. Concurrently, the spectators are not able to express their feeling or their opinion in any moment of the event. For this reason, in a live performance, musicians have an active role, since they manage the event and can make decisions about it, while the audience have a passive role since they can not make decision and change anything. Therefore, the artist decides on her own without knowing what goes on in people minds during the live event. Understanding the current feeling of the audience would make them more as an active role since the artist would be able to adapt or evolve the concert by accomodating their desires. To understand and to describe the general perception during a live event we studied and defined what we call *feedback parameters* by taking into account the musical technical aspects and the feelings involved in it.

RUSE is a tool that allows the audience to express their feelings, their opinion on six different couples of parameters such as slow vs. fast and consonant vs. dissonant, during a live performance. In turn, the artist is able to alter the concert in real-time according to the results received. Moreover, the real-time concept is very important, since the feelings are different in every moment and they change constantly and subjectively, it is crucial to provide this real-time interaction between artist and audience.

The goal of the project is to give the audience a more active role during a live music performance by providing their feelings to the artist who can evolve the concert accordingly. More specifically, the artist is able to control and ask the feedback to the audience in any moment i.e, to receive real-time feedback. As a result, we give a peer-to-peer situation where a live event become a participative event for the audience by bringing them at the same level of participation as the artist [1, 2]. We developed RUSE, as a web-application, to gather feedback from the audience and to represent the data to the artist in real-time. In addition, RUSE is compatible with Max [3] software, a music development environment, that allows the composer to receive the data and to control or ask the feedback directly from the software.

The project was developed jointly with the assistance of Danilo Gervasoni, **Conservatorio della Svizzera italiana**, a Master of Art in Composition and Theory student at the Conservatory of Lugano who did take care of the musical and compositional aspects of the project.

Chapter 2

Gathering Real-time Feedback

User feedback is an important component when we need to improve, modify, learn something. Given the goal of the project we have to gather feedback and present it in real-time, which is not the normal feedback from surveys where it is analyzed afterwards, but it is a continous update and analysis of the data while the live event is in progress such that the artist is able to understand the audience's feeling and make the decisions in real-time.

To get the feedback we also need to provide a simple and a captivating system to make people comfortable by giving it. Furthermore, the model to present the data is as much coincise as possible by considering technical musical aspects and the audience perception to allow a fast and simple interpretation. In this section we describe in details the idea behind the feedback system and model to gather and represent data intuitively along with the choice of the parameters for the actual version of RUSE.

2.1 How to gather real-time feedback intuitively

A music live event can arise many feelings and can be analyzed by many metrics, those are what we call *feedback parameters*. We studied the parameters in a way that we combine technical aspects and the feelings related to the event such as tempo, slow, fast, or monodic. To achieve a simple visualisation we define a pair composed by two parameters which is assigned to an axis, where we have a **positive** parameter, with value 1, and a **negative** with value -1.

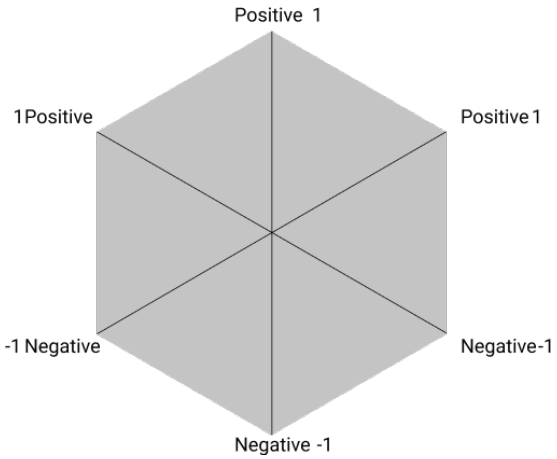

FIGURE 2.1: Feedback input prototype

As we can see from the Figure 2.1 there are three axes and six parameters, where each axis is composed by a positive parameter and a negative one which respectively have the bounds 1 and -1. In order to represent this schema, we have a hexagon as the general form of the feedback input. In this manner, the user is able to move a cursor along these axes and provide the feedback every time the cursor change

position. The meaning of **positive** and **negative** parameters is given by their intensity. For instance, the sound can have high and low frequency or can be fast and slow, in that sense, a positive parameter is when it is related to an high intensity value such as fast, chaotic, polyphonic, while for negative we have the opposite, it can be slow, regular, or monodic.

This model helps the user to have a better perception of its global feedback trend since by pairing the parameters in positive and negative we can have a spatial division on the hexagon. More specifically, we assign a color to each hexagon's area i.e., the positive and negative area, to improve this concept visualization. As shown in Figure 2.2 the negative parameters are in the lower half, while the positive are in the upper half. For example, a user can easily see if its cursors are in *red* or in the *blue* zone.

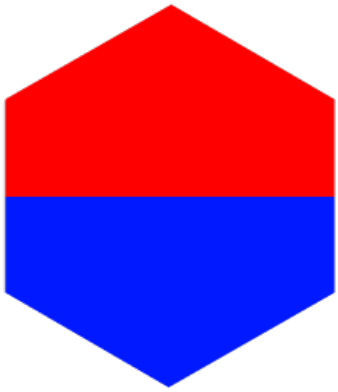

FIGURE 2.2: Global trend feedback perception

2.2 How to represent data intuitively

While being gathered from the audience, the data must be represented such that it gives the idea of what is going on audience mind with the aim to proceed to improve the live performance. The feedback is presented as a mean value of every axis, as already suggested in Figure 2.1 we have three mean values for each hexagon. Moreover, to provide the real-time interaction, every mean value is updated simultaneously, or on following artist's demand. Specifically, when a composer is performing is not always able to read, understand and adapt the current track concurrently, therefore the feature we call *interval feedback* come in help to this situation where the artist can choose an interval in which those values are updated and the results are computed only conseidering this interval in order to better organize the next moves. For example we can decide that every thirty seconds we want to see the mean values because we need to adapt the track we are playing and in a second time we want them at every fifty seconds, or we can just choose to receive the feedback as it comes from the audience.

Considering the real-time feedback it is useful to have a certain control of the audience feedback input, where the artist needs to monitor only specific parameters in specific situation. To give an example, during a live performance, there are specific moments in which the artist already knows what is the programmed move and by knowing the general perception on specific feedback beforehand, can help to adjust the track. More specifically, there is the possibility to ask specific feedback to the audience, where the artist can enable and disable some parameters such that the data received is more focused on what is needed. As a result of the interval feedbacks and parameters locking we avoid to influence the data from different moments during the live event and we allow the artist to customize the feedback on the fly.

2.3 Parameters in RUSE

In the current version of RUSE, there are two sets of parameters where each set is represented by a hexagon. These sets are assigned to two main categories, defined by considering genres aesthetics and domains of music perception, as follows:

- Structural
- Timbrical

The **Structural** group represents the parameters that describe the rhythm and the general structure of the performance such as the ambient and the drone. On the other hand, the **Timbrical** parameters, focus more on the musical notes where we can check the noise and the glitch. In this manner we can manage the microformal i.e., a formal development of the timbre and sound, and macroformal i.e., the vertical (sounds chordal) and horizontal (the timbre's sequence over time) organization, variations. Consequently, we defined twelve paramenters by pairing and arranging them in their respective group mentioned above.

Set	Negative	Positive
Structural	Slow	Fast
	Regular	Chaotic
	Monodic	Polyphonic
Timbrical	Dissonance	Consonance
	Low Pitch	High Pitch
	Dark Timbre	Bright Timbre

TABLE 2.1: RUSE parameters

The first pair, Slow - Fast, controls the speed of the rhythm wheather it is slow or fast. The Regular - Chaotic defines the rhythm regularity i.e., when elements are repeated exactly in an evenly spaced arrangement, and, the Monodic - Polyphonic is the pair that manages the number of notes which increases the synthesis voices in action therefore it checks if there are too many different notes or too few. The Consonance - Dissonance describes the level of the amount of sidebands in the FM spectrum [4] and the harmonic ratio between them, as well as to the frequency deviation on the fundamental. The Low Pitch - High Pitch establishes the fundamental frequency i.e., the perceived musical pitch of a note. The Bright Timbre - Dark Timbre defines the timbre of a note which is the sound quality perceived of a musical note. In Table 2.1 we summed up the parameter pairs with their respective groups.

Chapter 3

Project development

In this chapter we give a general overview of the project development by illustrating the architecture with the technologies used, a complete use case scenario, and, the user interface explained in details.

3.1 System Architecture

In this section we cover the application architecture, the technologies used to develop the application and, the deployment.

3.1.1 Architecture

The application architecture follows a frontend and backend model. More specifically, there are two frontend clients, the Audience clients i.e., the clients assigned to the audience which allow them to provide feedback, and, the Artist client which provides an interface to the artist in order to monitor the results and to manage the feedback inputs. At the creation of an event, the system creates two WebSocket channels, one between the artist client and the server, while the second between the audience clients and the server such that there is a continuous communication. RUSE uses sockets as communication mechanism, in which every action is mapped to an event. When the server receives data, it sends them to the Engine i.e., the part of the backend responsible to do all the computation with the data, which takes care to update the current event and then returns the data to the Web server such that frontend clients can be updated. The Figure 3.1 depicts the global workflow of the application.

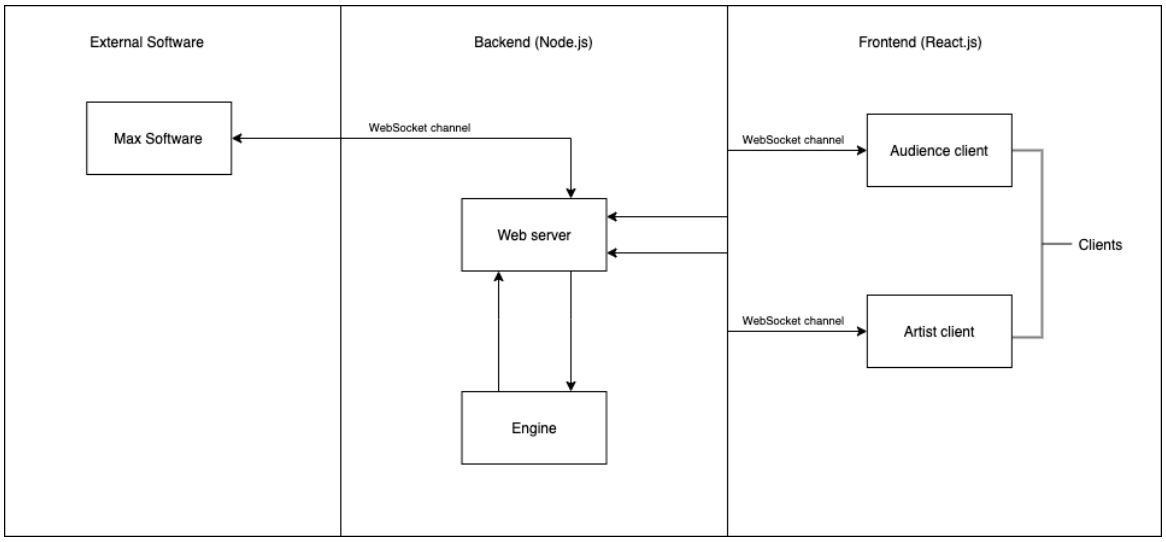

FIGURE 3.1: Application architecture

In addition, there is the possibility to use an external software, the Max software i.e, a software developed by Cycling '74 [3] which provides an environment to create a dedicated software or a musical and multimedia application in real-time. As shown in the Figure 3.1, on the left, Max is able to establish a WebSocket channel with the Web server and to access the same features as the artist client does, to communicate with the server and audience clients.

3.1.2 Technologies

The application is written in JavaScript. In particular, the back-end is made in Node.js [5], using the Express.js [6] and Socket.io [7] frameworks while the front-end is made using React.js along with Material UI [8] framework. For the development we used a Git repository and for the deployment we used Docker [9].

In order to make the development of some features easier, it is worth to mention some of the many libraries used. For example, the language support, specifically the Swiss languages and English, we used *i*18*n* [10] JS library.

3.1.3 Deployment

RUSE has been deployed on a VM managed by the Software Institute. For the deployment we used Docker [9] where we have two docker files, one for the frontend (Listing 3.1) and another for the backend (Listing 3.2) to build the docker images. By having the docker images we then use a docker-compose (Listing 3.3) file which allows us to run the docker images, and therefore to run the frontend and backend, by using only one docker file. In Listing 3.1 you can find the frontend docker file, while in Listing 3.2 the backend docker file and Listing 3.3 illustrates the docker-compose file.

```
FROM node:latest
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
COPY package.json /usr/src/app/
RUN yarn install
COPY . /usr/src/app
EXPOSE 3000
CMD ["yarn", "start"]
```

LISTING 3.1: Frontend Dockerfile

```
FROM node:latest
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
COPY package.json /usr/src/app/
RUN npm install
COPY . /usr/src/app
EXPOSE 5234
CMD ["node", "server"]
```

LISTING 3.2: Backend Dockerfile

```
version: '3.8'
services:
  ruse_backend:
    container_name: ruse_backend
    image: gitlab.reveal.si.usi.ch:60090/students/2021/andrea-brites-marto/
       ruse-backend:latest
    ports:
      - "5234:5234"
  ruse_frontend:
    container_name: ruse_frontend
    image: gitlab.reveal.si.usi.ch:60090/students/2021/andrea-brites-marto/
       ruse-frontend:latest
    tty: true
    ports:
```

```
  - "3000:3000"
environment:
  - REACT_BACKEND_HOSTNAME=ruse_backend
links:
  - ruse_backend
depends_on:
  - ruse_backend
```

LISTING 3.3: Docker-compose, `docker-compose.yml`

3.2 Use cases

As we can see from Figure 3.2, the system is composed by two actors, the audience user and the artist user. First of all, the artist creates and starts the live event i.e, a live electronic music performance, with the action *Create Event* that establishes the webSocket channel between artist and server and provides an event code to share with the spectators or whoever want to join the event. Consequently, the audience users can use the event code shared by the artist to join the live event (*Join Event*) which, also here, the action establishes communication between audience client and server.

From here on, artist and audience can communicate continously, and, while the event is on, the artist has three possible actions. The first one is *Ask Specific Feedback*, which allows to ask only specific parameters to the audience by disabling or enabling, on the audience clients, the cursors on the hexagon axes. The second action is the *Set Interval Feedback*, where the artist allows the spectators to provide feedback for a given interval time and the cursor's starting position and when the time elapses the results are displayed. The third action is *Real-time Feedback* which when used, the results are returned in real-time and everytime a feedback is provided, they are shown to the artist without waiting time constraints such as interval time. When an audience user votes (*Provide Feedback*), the event data is updated (*Update Event Data*) and then we aggregate the data (*Aggregate Results*) such that the artist receives the results based on the triggered action.

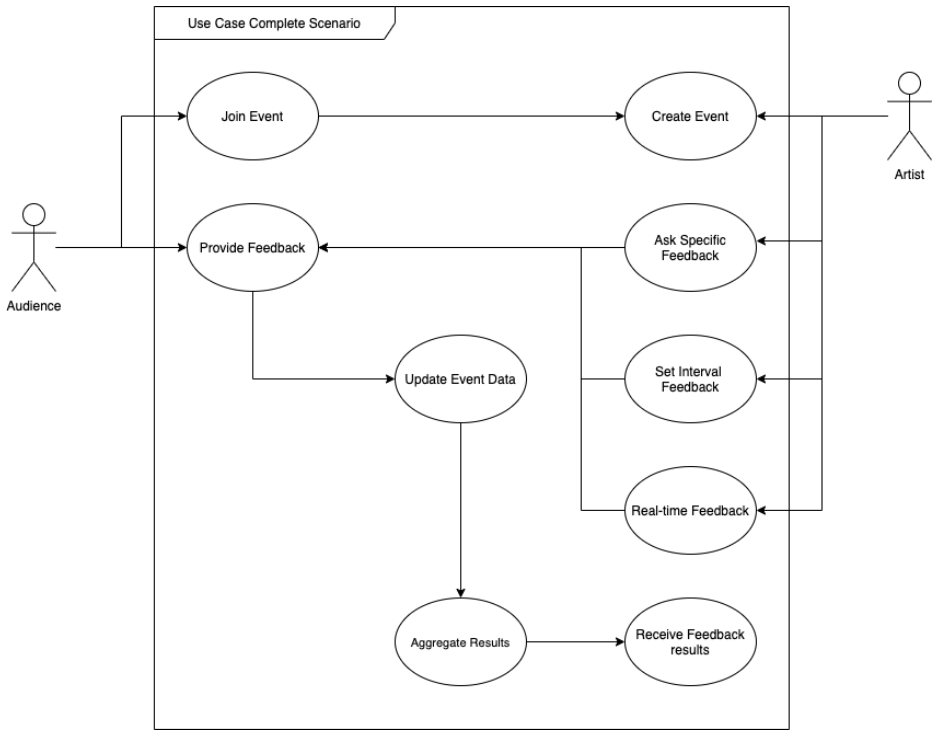

FIGURE 3.2: Use case complete scenario

3.3 User Interface

Figure 3.3 summarizes the navigation structure, where we have four main components. When visiting the page, the user is presented with the MainPage where the user can join or start a live event. In the first case the user must provide an event code in order to join the live event which will be redirected to the AudienceView while in the other case the user can freely start a new live event and will be redirected to the ArtistView. The AudienceView and ArtistView are accessed from the MainPage and are the views assigned respectively to the audience and the artists. In addition, we also have a CreditPage where we show the people who contributed in this project.

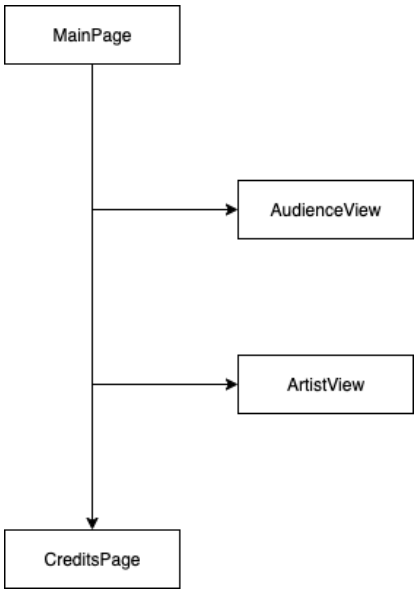

FIGURE 3.3: Navigation Structure

3.3.1 Main page View

In Figure 3.4 we can see the splash page of the application, which introduce users with the project logo on the top followed by a introductory description of RUSE. The MainPage also provides two components below the text presentation, where the first one, on the left, allows the users to provide an event code to join an event while the second one, on the right, allows the artists or composers to create and start a new event.

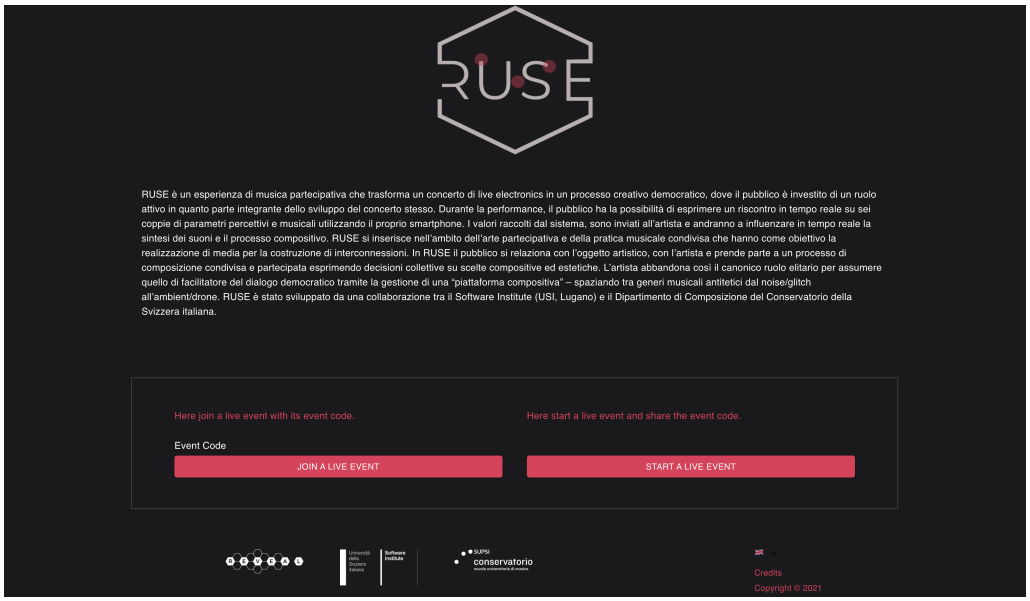

FIGURE 3.4: The Main Page (Desktop version)

3.3.2 Artist View

The Artist View is the component assigned to the composer of the event. In it, the artist is able to perform the actions described in Figure 3.2 and to monitor the parameter ratios during the performance. More specifically, a parameter ratio is the mean of the feedback values for each axis provided by the audience and is represented by a heatmap bar, as we can see in Figure 3.5. In turn, every bar component has an indicator, the white vertical line that shows the parameter ratio, to see at a glance whether the audience tends towards an extreme or the other. The gauges are also provided with two buttons, which are located next to each heatmap. The outer one, when clicked, emits a socket event which asks the server to upload the parameter value chosen by the user and update the parameter ratio in real-time. Additionally, each hexagon axis has a boolean state, which indicates if a cursor on that axis is visible or not. The other button can alter the state of the axis, namely activating or deactivating that cursor, by simply clicking on it. Besides observing the feedback of the audience, the artist is able to influence the Audience View in real-time. For instance, she can activate (or deactivate) any of the axes in real-time. The Audience View will reflect those changes and enable (or prevent) the audience to express its feedback on those axes.

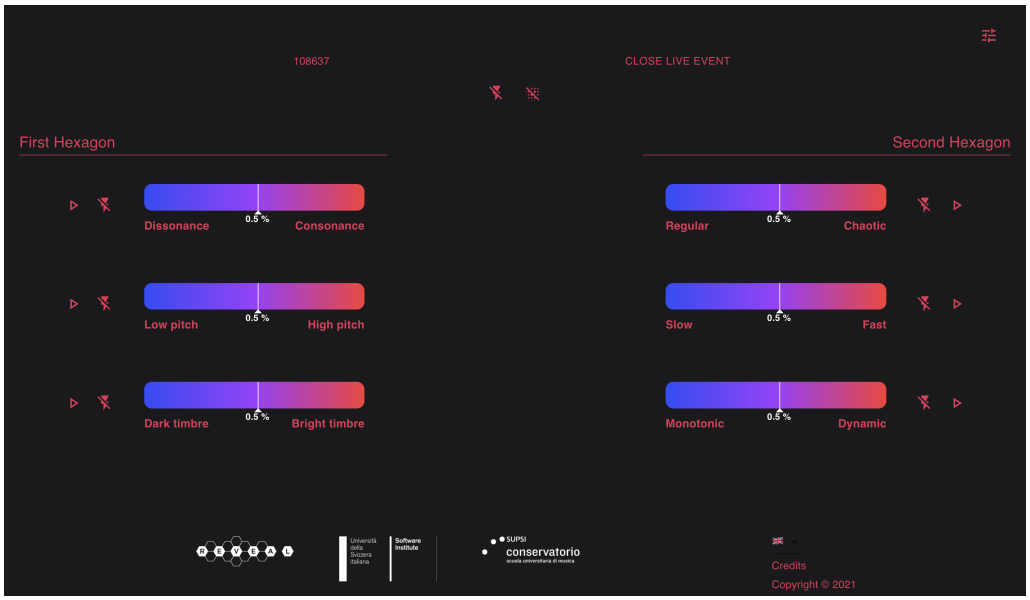

FIGURE 3.5: The Artist View (Desktop version)

In this view is possible to customize range and unit values, the time interval and the initial position of the cursor of the next interval. For example, in Figure 3.6 we can see that all bars are in percentages and from 0 to 1, we can modify such that we have a range from 0 to 3,000 with Hz as unit value. The values are then directly converted based on the settings. The time interval is in seconds and by default is set to 3,000 seconds.

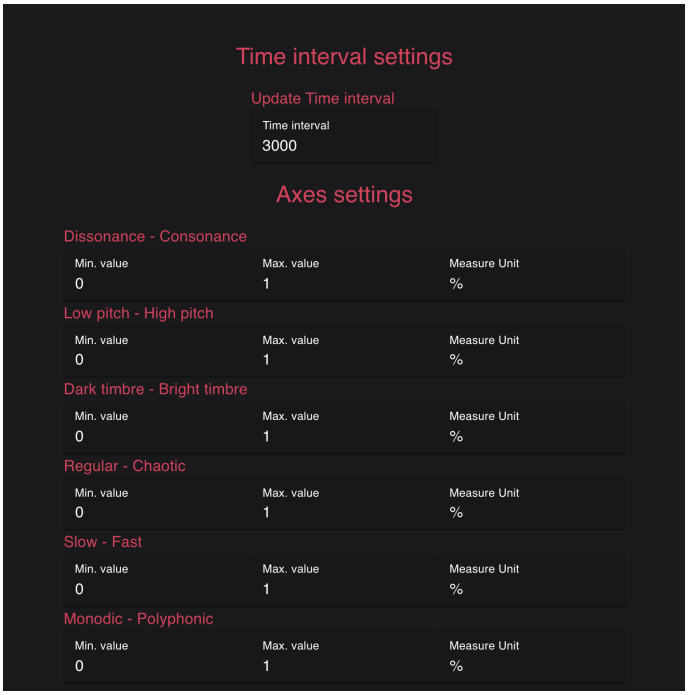

FIGURE 3.6: Artist's settings View (First Part)

In Figure 3.7, is shown the second part of the settings, where there is the export data feature which allows to download the entire history feedbacks provided by the audience. RUSE also provides means for the artist to gather feedback from the audience only on specified time intervals. To do so, the artist can start

a timer and by specifying for the required axes the starting position of the cursor. During this time interval, only the axes needed are enabled and RUSE will collect the data from the audience. When the timer elapses, the Artist View will show the opinion of the public during this time frame in the corresponding heatmap bars we explained above.

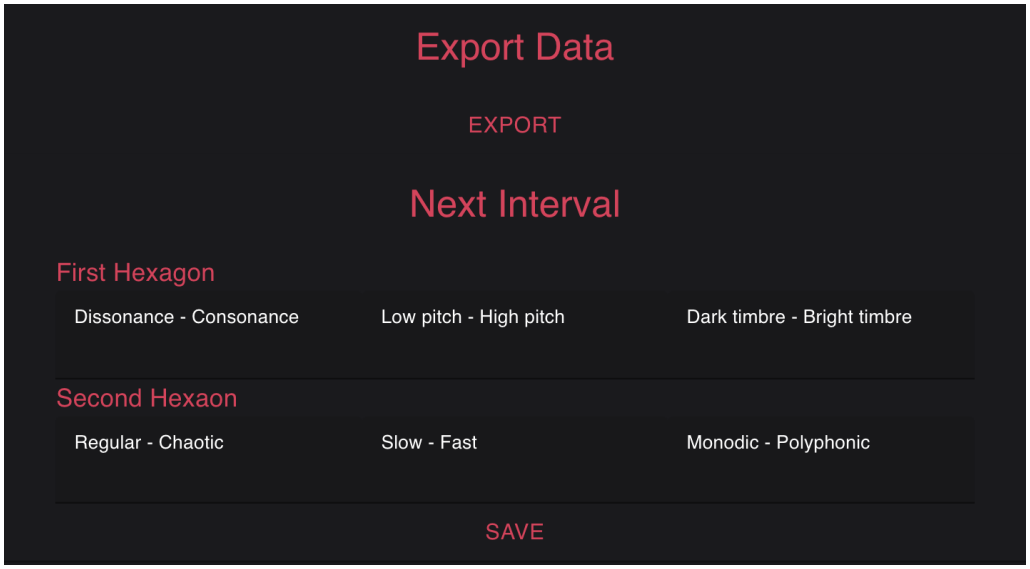

FIGURE 3.7: Artist's settings View (Second Part)

3.3.3 Audience View

The AudienceView is used by the audience to provide feedback and is accessed by providing an event code. In the current version of RUSE, we defined six pairs of parameters, lying on opposite ends of a given spectrum (e.g., slow / fast). In the Audience View, participants are shown two hexagons to express their feedback on the aforementioned parameters. Each pair of parameters is positioned on diametrically opposite vertices. **Positive** values are placed in the upper half of the axes and colored in red and the **negative** ones in the lower half and colored in blue. On each of the three diagonals formed by joining diametrically opposite vertices, users can express their opinion, i.e., by dragging around a slider. Whenever a user expresses a feedback, RUSE records the actual value and a timestamp. The overall feedback of the audience is computed by aggregating all the preferences of the single users. In Figure 3.8 we can see an example of two sets of parameters (i.e., two hexagons), where the right one has three cursors, while the other one on the left has only one since the artist asked the audience to vote on this configuration.

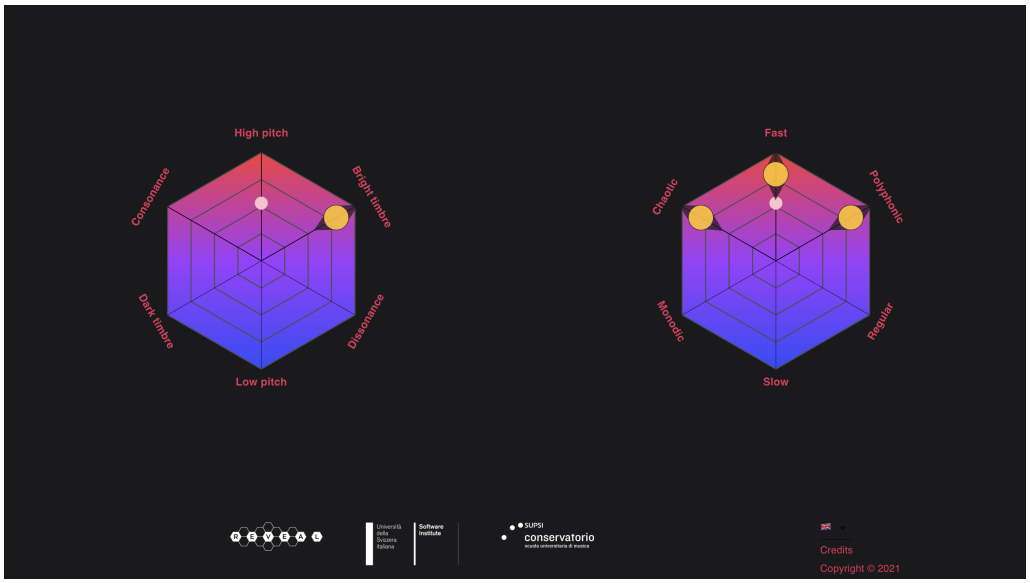

FIGURE 3.8: The Audience View (Desktop version)

Chapter 4

Testing

The web application has been continously tested during its development. To improve and test RUSE we organized two main meetings at Palazzina DR, in Lugano i.e., a branch of Conservatorio della Svizzera Italiana, with expert people on the field such as **Roberto Minelli**, **Danilo Gervasoni** and **Alberto Barberis**. The first one was on the 16th of April, while the second on the 3rd of May. During these days we tested if RUSE is able to provide the necessary data to produce music and the global workflow in a real environment where, thanks to Conservatorio della Svizzera Italiana, we were able to simulate a simple concert.

In the first test day, RUSE has been tested on the main features such as the communication between the audience and artists clients with the server, the computation of the results presented to the artist and that the views (audience and artist) behave correctly. We used Max software in order to produce and modify on the fly the music while receiving the feedback from RUSE. In the following Figure 4.1 you can see some shots of the day.

FIGURE 4.1: Some shots of the testing day on the 16th of April

The application was working very well, but we discovered some feature to improve:

- Simplify the Audience View and make it responsive (desktop/mobile)
- Introduce the concept of time intervals
- Change from one only event to multiple events at the same time

In the second meeting, on the 3rd of May, most of the features were working and we focused on improving the time interval feature. At the beginning the idea was just to gather data from an interval time, to have a precise feedback on that moment i.e., a mean value of the parameters based on the interval time and not on the total time passed, but we found out that this concept can be improved by adding the possibility to also specify which parameters can be voted on the said interval time because it allows the artist to make a minimal but significant variation.

Chapter 5

Conclusion & future improvements

In this project, we achieved the main goal, a system in which the audience during a live electronic music event is able to participate actively and on the other hand the artist can influence and evolve the performance accordingly to the feedback received. We studied an intuitive model to gather data along with the choice of the parameters to efficiently improve this experience.

Despite RUSE is a working web-application with all the necessary functionalities, it can still be extended with some improvements in the future. A meaningful extension to implement would be to configure a database and save the data in it, since as of now the application does not save the data in a database, but it only allows to export the data after the concert is concluded. In RUSE, the hexagon's parameter are represented by a json object and they can be changed by modifying the file, therefore the customization of the hexagons does not have a proper interface, with a dedicated component to customize the hexagon from the interface would improve the user exeprience. In addition, by having a database, the possibility to analyse data afterwards would be easier and therefore it would be interesting to develop a more sofisticated algorithm to visualize the results after the concert.

By exploring the concept of giving an audience a participative and active role during a live performance, RUSE is a great tool, it can be used in many ways such as live electronic concert, to do social experiments on how an audience react during a live performance, even to influence live streamings. I am really satisfied with the results of this project and I am convinced that RUSE has a great potential, I am also very happy that the project is a real working web-application and is currently available at `https://ruse.si.usi.ch/`.

Acknowledgements

I would like to thank Prof. Dr. Michele Lanza for allowing me to work on this project, for his guidance and teaching during the development. I also want to express my gratitude to Dr. Roberto Minelli for guiding and encouraging me, as well as for his great help in organizing the project and his continuous feedback and suggestion on the project. I am grateful to the Software Institute and Conservatorio della Svizzera Italiana for the collaboration, especially for Danilo Gervasoni, who assisted me in the technical musical aspects, and, Alberto Barberis for his experience on the field. I also want to thank my family and my dear friends, Sasha and Stefano, for their patience and moral support during this semester.

Bibliography

[1] K. Rutten, "Participation, art and digital culture, critical arts," 2018.

[2] V. Gourdarzi, A. Gioti, G. Lepri, and F. Morreale, "Exploring participatory sound art," 2019.

[3] C. '74, "Cycling '74." https://cycling74.com/, 2021.

[4] K. Seymour, *"Frequency Modulation". The Electronics Handbook*. Taylor & Francis Group, 2005.

[5] O. Foundation, "Documentation | node.js." https://nodejs.org/it/docs/, 2021. Accessed on 03.05.2021.

[6] O. Foundation, "Express 4.x - api reference." http://expressjs.com/en/4x/api.html, 2021. Accessed on 05.04.2021.

[7] Socket.IO, "Introduction | socket.io." https://socket.io/docs/v4, 2021. Accessed on 03.05.2021.

[8] Material-UI, "Material ui: A popular react ui framework." https://material-ui.com/, 2021. Accessed on 03.05.2021.

[9] D. Inc., "Docker documentation | docker documentation." https://docs.docker.com/, 2021. Accessed on 27.04.2021.

[10] i18next community, "Introducition | i18next documentation." https://www.i18next.com/, 2021. Accessed on 12.04.2021.