Bilkent University

Department of Computer Engineering

Senior Design Project

microgliss: a microtonal editing tool for world music

Project Specifications Report

Group Members: Artun Cura, Sonat Uzun, Orkan Öztrak

Supervisor: Vis. Prof. Dr. Fazlı Can

Jury Members: Assoc. Prof. Dr. Çiğdem Gündüz Demir
Asst. Prof. Dr. Can Alkan

Innovation Expert: Burcu Coşkun Şengül

Project Specifications Report
October 12, 2020

Contents

Project Specifications Report

Microgliss: a microtonal editing tool for world music

1 Introduction

Microgliss is the new cutting-edge note editor and synthesizer[1] that allows you to write microtonal music[2] using an intuitive node and edge-based workflow. It will allow users to write music in any tuning system or independent from any tuning system. It will allow users to add glissandos[3] between any note and edit these glissandos with further settings. Microgliss is designed by keeping in mind the growing interest in microtonal music around the globe and common limitations of existing midi/note editors and synthesizers.

This product will be developed for musicians. To understand the details about the product, you need to have a basic knowledge about the terms which will be used during the discussion of the project. Direct definitions are given through the report but a more general knowledge will be given in the next section.

1.1 Basic definitions and a brief explanation about the domain

Nowadays, people use programs which are called Digital Audio Workstation (DAW) for recording, editing and producing music. These programs are designed specifically for this purpose and optimized according to the needs of musicians. They let users add different layers of sound, which might be recordings or digital instruments, and you can edit these tracks by using different channels. Tracks are routed to a digital mixer, a device which lets you change the volume of a track, add different effects and signal processors. Every DAW contains some basic plugins for audio editing and production, but the user is also free to use 3rd party VST[4] plugins from external sources.

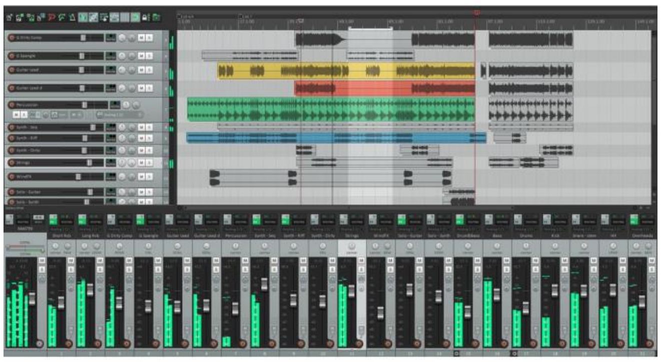

Figure 1 An example DAW, Reaper [1]

[1] An electronic musical instrument that generates audio signals.
[2] Music that composed of intervals which are smaller than a semitone. This term also includes any tuning system which differs from the western 12-tone tradition.
[3] Sliding from one note to another seamlessly.
[4] A plug-in format for a digital audio workstation.

Also, in addition to the recorded audio, a user can use virtual instruments. Every DAW provides a system called MIDI[5] piano roll for note editing. In these systems, the vertical axis represents the pitch of the note while the horizontal axis represents time. In the figure below, the green lines are the notes with a start time, duration and end time.

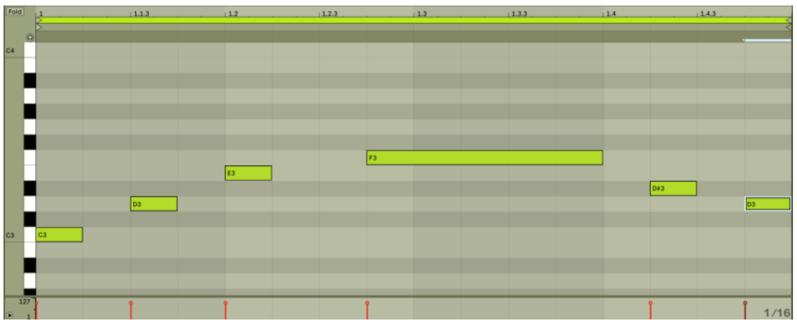

Figure 2 An example of a Midi Piano Roll

In these editors, available pitches are pre-determined according to the 12-tone western musical system. 12 tone musical system based on dividing the distance between two consecutive same lettered notes, an octave, to 12 almost equal spaces. Every space is called a semitone, and Figure 2, every semitone is shown with a clear separation line, as if it is a piano, in the figure.

This shows us that in traditional MIDI editor systems, every written note must be separated, on a different line, and their frequencies are already set.

1.2 Details about our project

As it is stated in the previous section, DAW's let users use the 12-tone system in the piano roll. But in 2020, this traditional system should not be a constraint. Microgliss lets users work with a freer workflow and eliminate these restrictions.

Microgliss will have two separate modules which can be accessed through one window: A microtonal note editor and a synthesizer to play these notes. In the microtonal editor, the user will be able to add notes, edit notes and connect them. Every note will be represented as a node, and their connections will be represented as edges. Nodes will be placed on an infinitely(almost) zoomable field. This will provide users to add notes without definite frequency restrictions.

[5] Musical Instrument Digital Interface

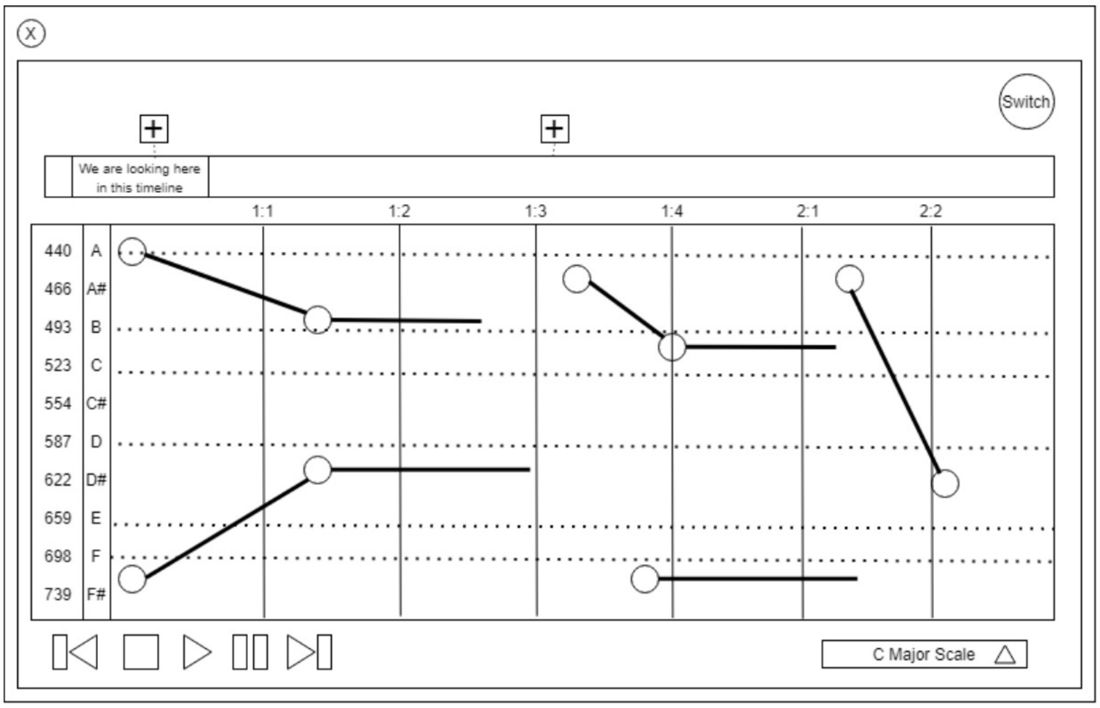

Figure 3 Preliminary Editor UI Design

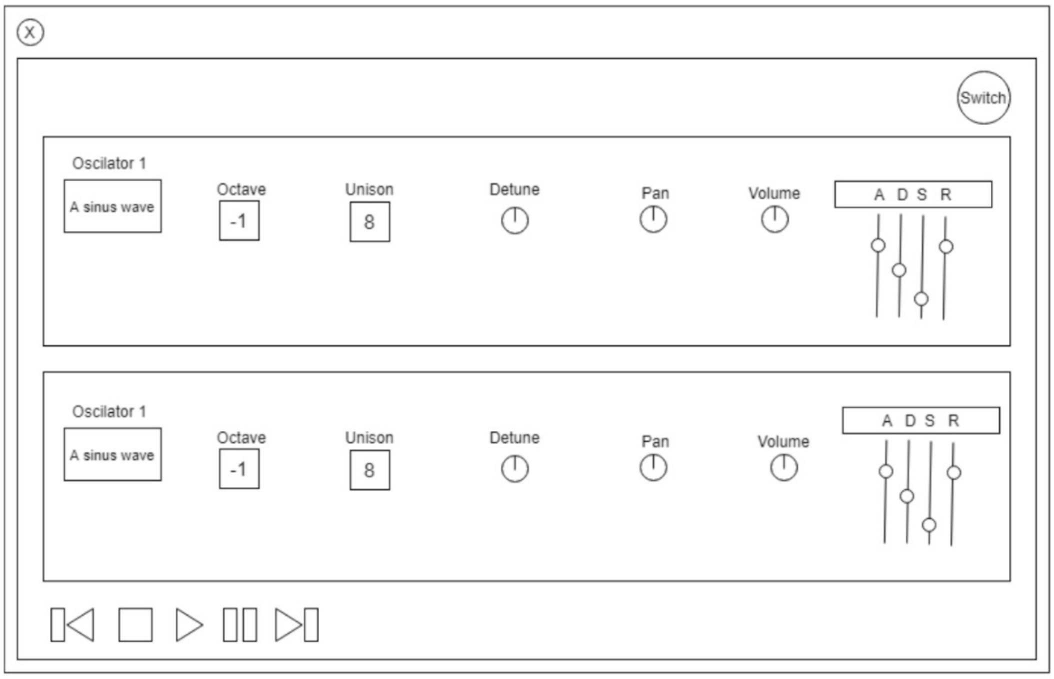

Figure 4 An Example Synthesizer UI Design

In the synthesizer part, users will be able to change the sound of the instrument according to their needs. All capabilities of the project will be discussed in the functional requirements section.

Microgliss can be put directly into the DAW’s audio channel as a VST plugin. This will provide a way to eliminate traditional midi editor constraints. An editor which encompasses the entire timeline of a project is uncommon in VST’s, but we know that it is possible because we know VST’s can be aware of the playback position of the project. Also, there is a very popular VST that already has an editor that encompasses the entire timeline called Melodyne.[2] It is used for editing individual notes in an existing audio recording (a problem that is different from ours).

1.3 What is special about our idea?

The innovative and interesting part of the program is the note editor. Editing capabilities of our editor is beyond any existing tool. There are other microtonal editors out there but they are not as flexible as Microgliss. Generally, they enforce a grid (a tuning system), but using a grid should be a preference of the user, which also they might prefer not to. Having it as the only choice is a restriction which will be eliminated with our product.

They also don't allow individual note glissandos except for Bitwig Studios internal MIDI editor. Our product has a much different and much more flexible proposed workflow and can be used in DAW’s other than Bitwig studio. [3]

2 Constraints

2.1 Compatibility

Professional music applications can be successful in terms of widespread usage and production of culturally significant pieces in two ways:
1. If they can be integrated into existing workflows of professional musicians easily.
2. If they are so crucial and special in their sound or functionality that musicians will be willing to use them even when they aren’t smoothly integrated to their existing workflows.

Even when a project is unique in its capabilities having it compatible with existing workflows will increase in its popularity, usage and therefore it’s capacity of being used in culturally significant works of art. Thus we need to take compatibility into account and make it available for any kind of device and program.

Today almost all music and more specifically electronic music goes through are made using DAW's, thus our program needs to be compatible with DAW's. For this reason, we plan to export our project in VST format which puts a constraint on the tools we can use and functionalities we can add to our program. We are planning to use the C++ language and JUCE framework, because they are trusted and popular development tools for developing VSTs and it is easier for us to export in other formats as well. Also, JUCE is free until a high threshold earning value, which is appropriate for us.

2.2 Development

The editor is the core of the program. But we also need to build a synthesizer to play the data supplied by the editor. We will keep our focus on the editor to make it the best it can be in our limited time frame and will go with a simple design for the synthesizer.

2.3 Cultural

We will take into consideration all the musical cultures to the extent of our capacity while we are deciding on the details of our editor. For example, in Indian ragas (which is similar to a western scale) there are glissandos between specific notes, which is something you can do with our editor. If in some way we can make it easier to write in that raga, we'll do it.

3 Professional and Ethical Issues

Our project will be inclusive for any kind of world music, Being culturally sensitive when designing such a tool is important. If project such as ours can include as much as they can from different musical traditions, these traditions can be vitalized, thrive and escape the fate of being forgotten in this modern era. The world has a multitude of rich musical cultures and we believe that these cultures should be engraved into the tools we use while creating music.

But we shouldn't merely focus our attention on recognition and replication of the ideas in these musical cultures. We should encourage and allow innovation and creativity with our design and unlock new creative potentials with our design. We are not only recognizing past and existing cultures. We are also designing the tools for the music of the future.

It is also important to be respectful to both needs and capabilities of both beginner and professional users. We will design our application to be easy to use and powerful as much as we can. We plan to ask other people to test our program thus we can serve the best user experience for any level of musician.

4 Requirements

4.1 Functional Requirements

The main controller device for the following actions will be the mouse and the basic keyboard buttons such as shift, alt and ctrl. We will use UI elements, such as sub-windows with editable parameters and tools to aid the mouse in the editing process if necessary. This will enhance the workflow for the user. As discussed in the previous sections, our program will have two main modules which are note editor and a synthesizer. Functional requirements of these modules will be analyzed within their title.

4.1.1 Note editor

Controlling the timeline:

- The user will be able to move the pitch axis up or down.
- The user will be able to move the time axis left or right.
- The user will be able to zoom in or out to the timeline.

Adding Notes (Nodes):

- The user will be able to add a note in any time and any frequency value. These values might be snapped to a certain time and frequency values if the snapping function is enabled.

Selecting Notes (Nodes):

- The user will be able to select a note by clicking on a note.
- The user will be able to select multiple notes.

Editing Notes (Nodes):

- The user will be able to edit the nodes’ frequency by dragging the note up or down.

Adding Glissandos (Edges):

- The user will be able to add different glissandos (linear, curved etc.) between notes by selecting multiple notes, making an interaction which will open a window. select glissando type, and selecting the desired glissando type.

Editing Glissandos:

- The user will be able to adjust the curvature of the glissando and add vibrato by selecting at least two notes, making an interaction which will open a window for editing of the glissando type.

4.1.2 Synthesizer

In most software and hardware synthesizers, there are editable parameters that modify the sound output. Our programs own synthesizer will have editable parameters as well and have a similar workflow to existing synthesizers for editing them.

Editing Parameters:

- The user will be able to edit dropdown type parameters by clicking on the parameter and selecting the desired option.
- The user will be able to edit the knob type parameters.

4.2 Non-functional requirements

Scalability:

- As we discussed earlier, the note editor is the core of the program, though there could be many other synthesizers that could work with the editor. For this reason, we will build the program in a way that this synthesizer part can be replaced without touching the code of the core editor. It is compelling to imagine that with the evolution of new protocols and their integration to DAW’s many synthesizers can work with our editor. For the time being, we will look into the OSC protocol to see if we can use it in any way to communicate with other instruments.

Usability:

- Our program is mainly an editor program. We want It to be both fast to write in notes and modify the notes that are already there. We will supply the generic tools for mass editing midi notes like selecting multiple notes, deleting, copying, pasting and editing them. We can also supply our new tools for inputting and editing many notes in a short amount of time. In the end, we wish the program to be as fast to use as the traditional midi roll piano notation. We can ask people to test our program to check if this is the case.

- We have based our design on the midi piano roll which is also similar to sheet music, thus it should be easy for people who are familiar with the piano roll editor or sheet music, to learn using our program.
- Our program should also have a modern UI and should work with multiple resolutions. This is given for UI’s in most domains but many VST's don't have a scalable UI.

- Our program should have a scalable note editor so that we can see the section of the timeline we want to see as zoomed in and zoomed out as we want. Both vertically and horizontally.

5 References

[1] "Reaper" https://www.reaper.fm/ [Accessed 12 October 2020]

[2] "What is melodyne?" https://www.celemony.com/en/melodyne/what-is-melodyne [Accessed 12 October 2020]

[3] "Bitwig Studio" https://www.bitwig.com/ [Accessed 12 October 2020]