Soroboro

An Honors Thesis Project by

Alexander Dupuis

Concentrating in Electronic Music & Multimedia

Brown University Music Department

Thesis Documentation

April 18, 2010

Faculty Advisor: Joseph Rovan

Summary

Soroboro is a piece for bass clarinet, live audio processing and interactive graphics using Max/MSP/Jitter. The clarinet acts as the source for the electronically manipulated sound, and controls different parameters of the audiovisual processing, hopefully leading toward a unified visual and sonic environment. The piece was premiered on Saturday, April 17th of 2010 in Brown's Grant Recital Hall, with Amy Advocat as the bass clarinetist.

Goals

As stated in my thesis proposal, one of the initial goals of the project was to write a series of shorter pieces for three instruments. However, almost immediately after beginning to compose the first movement for bass clarinet, I decided that writing a longer piece for bass clarinet would not only be logistically easier for me, but would be more satisfying compositionally. The bass clarinet is quite possibly my favorite wind instrument, and a longer form for the piece allowed me to utilize more of the instrument's potential within a cohesive work than I would have been able to in one movement.

Conceptually, I wanted the processing to reflect the Ouroboros, the snake that eats itself. I decided to create audio and visual processes that, given an appropriate input, processed the sounds or images and fed the altered results back into themselves, leading to constantly morphing textures. Though these modules were adapted to specific purposes within the piece, I now can modify them for use in future projects simply by inserting different processes into the feedback loop.

In terms of the technical and aesthetic goals, I wanted a piece with a clear, sophisticated relationship between the live performance, the electronic sound and the processed visuals. Though the piece would move between different sections, I wanted the overall feel to be of a unified piece with variations, with the written part, audio and visuals progressing naturally. Finally, I wanted a reliable piece that would lend itself to future performances by limiting unnecessary technical requirements, allowing others to perform the piece without specialized knowledge of Max/MSP/Jitter.

Composition

After deciding to compose a single piece for bass clarinet, I mapped out the composition into five stages. Each section would have a focal pitch, which I ended up structuring as the first half of an octatonic scale: G – G♯ – B♭ – B♮ – G, with the last focal G as a recapitulation of sorts. Each section has a unique combination of audio and visual processes, as well as distinct source material for the visuals (with the exception of the first and last movements, which share the same source with slightly different processes).

The first movement begins on a G4 (as written for bass clarinet), and slowly expands upward. I imagined a foggy, subdued sonic world that is initially still and calm, but becomes murkier and more turbulent as the clarinet strays from the initial note. The audio processing is intended to strengthen that idea, roughly matching the pitch of the clarinet, but becoming slightly muddier as the clarinet's phrases move higher in pitch and become more rapid. The choice to

move the pitch in an upward direction was a conscious one: though the bass clarinet excels in the low register, I wanted to totally avoid that area for the first movement, building up the tension as the clarinet rises higher and higher until that tension can be released at the start of the second movement.

The low $G_{\sharp}$ does come in at the start of the second movement, and functions as a pedal throughout. The melody is still carried out in the middle to upper register of the clarinet, driven by melodic fragments that are expanded and punctuated with the low $G_{\sharp}$ pedal. The tension builds within this movement as the low notes morph into multiphonics, with the clarinetist singing into the instrument to add further distortion to the timbre. At the height of the intensity of the multiphonics, a distorted beat created from snippets of the live clarinet sound fades in, setting up the transition into the third movement.

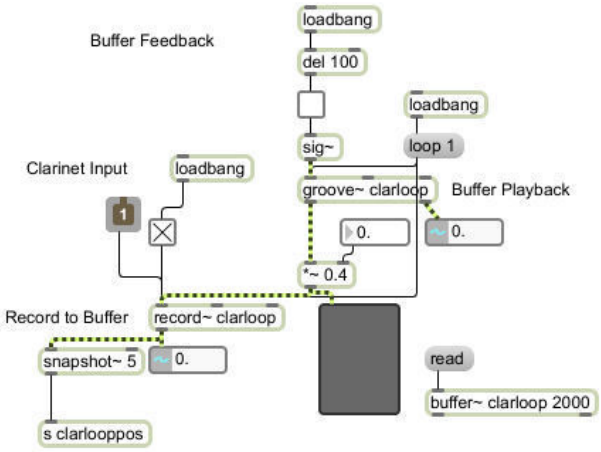
The third movement is the only movement in the piece governed by a strict tempo and beat. While there is an underlying rhythmic structure to the phrases in the other movements, their uneven rhythmic divisions push the performer towards more fluid playing that is almost improvisatory in feel. In contrast to that, the third movement aims to have a tighter, groovier feel. Though I enjoy driving rhythms in music, my rhythmic processing of live sources has often disappointed me in the past, with the end result generally being fairly static beats with thin sound. For this piece, I managed to get a more full sound through processing and compression, and also was able to map the clarinet's pitch and amplitude to real-time processes that allowed the beat to morph over time to reflect changes in the clarinetist's performance.

After the intensity of the previous three movements, the fourth movement offers a respite from extremes of pitch, volume and rhythm, though it is still contains foreboding undertones. I wanted a ghostlier sound, and wrote large tremolos that, while executable from a fingering standpoint, are too large to facilitate full sounding of both notes. This results in a sound that, while pitched, is more breathy and fluttering than a typical clarinet note. The fluttering characteristics are amplified by the processing, which focuses on the upper end of the spectrum using interactive equalization, bringing out the clicks and higher notes.

The tremolos gradually diminish in size, until they come back to focus on the original G note. Thus, the conceptual snake has come around to find itself at the end, through the restatement of the pitches as well as some of the motivic elements from the first movement. This time, the G is played in the lower register, signaling the end of the piece, though the ending multiphonics give the piece an air of unease even as it draws to a close.

Audio Processing

The main audio processing module consists of a continuously recorded buffer that feeds back into itself. The clarinet input is recorded to a buffer at the same time that the buffer's contents are scaled down in amplitude and recorded back to the buffer as well. In this way, the buffer acts almost like a delay with feedback, taking in new input while gradually fading out the previous sound through the feedback process. The crucial difference in this

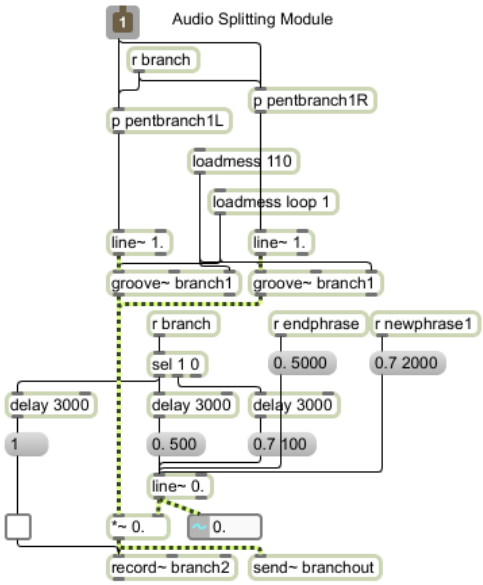


module is that the buffer's contents can be modified in a much more flexible manner; small sections of the buffer can be played at any time, can be reversed, and can be altered in pitch by stretching or contracting the amount of time they take to play back. These pitch shifts can also be undone by fft-based pitch shifting, a process which allows the option to stretch the amount of time a sample takes to play back without altering its pitch.

Though this module gave me a buffer of gradually decaying clarinet input that I could draw on, I still needed to devise the best way to access the data. The main issue with a continuously recorded buffer is that the virtual playback head should not over cross the virtual recording point. If this happens, the playback will cross over a discontinuity between the samples, resulting in a click or pop, which for my piece would be aesthetically out of place with the smooth effects that are generally desired. Compounding the issue is the fact that I wanted to play back the buffer's contents at different speeds; I had to factor in the possibility that a fast playback might overtake the recording point and cross it, and that slower playback could allow the recording point to pass the playback head. Therefore, I monitored the position of the recording head (using the "snapshot" object in the above figure) and used that data as an offset for the starting playback position, taking into account the speed of the desired playback so that the recording point would not interfere and cause a discontinuity.

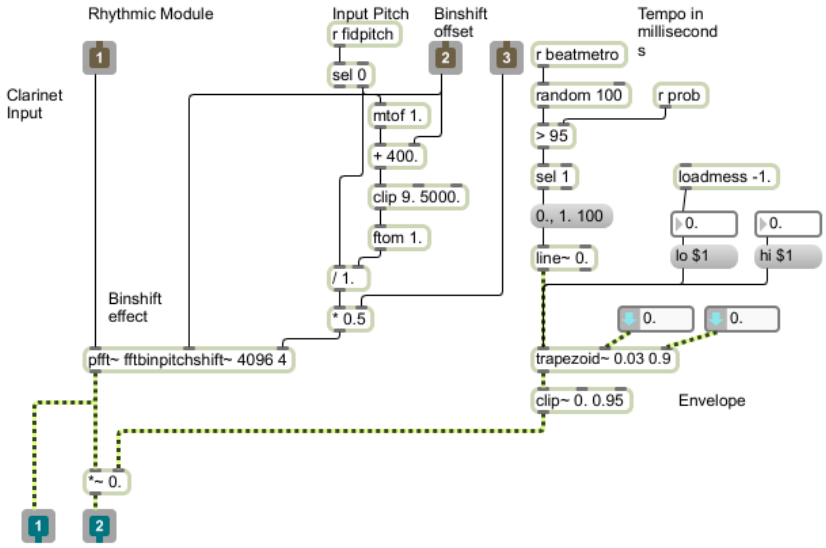
This system is used in the piece to create washes of sound from the clarinet input. For each section, I chose different speeds for the clarinet input to be played back at, depending on the desired pitch shifts and time stretches, and filtered the results. The filtering was done with resonant filters mapped to the third and fourth partials of the input clarinet pitch, resulting in a generally high-pitched sound that related to the pitch of the current clarinet note. In this way, the processing is kept from becoming too thick and overpowering, especially during more chromatic parts of the piece, when the actual contents of the buffer become highly dissonant from the chromatic notes feeding back onto each other.

The other ambient audio processing module, which appears in the fourth movement, involves "splitting" the clarinet's audio based on phrasing. When the clarinet starts a new phrase (by playing after a rest of sufficient length) the module records the initial note of the phrase and begins to repeatedly play it back. When the processing detects a second attack, the note is sent to two separate pitch shifters, one of which shifts the note up and the other down a pentatonic interval (either a minor third or major second). These two new notes are recorded back into the buffer and looped, until a third attack comes, splitting those two notes up and down into four new notes. This multiplication continues until the performer stops their current phrase, which resets the effect. Just like the first module, this effect could quickly become extremely dissonant and unclear, so the output is sent to a set of tuned delays that resonate the audio at specific pitches, using the C# major pentatonic scale for the fourth section and the F major pentatonic scale for the fifth movement.



The last main audio processor is the rhythmic engine. The input clarinet sound is sent through a pair of modules that analyze the audio to get its sinusoidal components, and shift those

sine waves up a specified number of hertz. This disrupts the relationships between the sinusoids: for pitched elements, the components' frequencies are usually close to being multiples of the hertz of the fundamental frequency, but offsetting all frequencies causes the components to lose this relationship. The input signal is also analyzed for its original fundamental frequency, and an fft pitch shifter attempts to restore the fundamental to its original pitch after the frequency shifting. This should, in theory, result in a pitch with the same fundamental as the clarinet, but with disrupted component relationships retained from the offset process.



This altered sound is then gated with an envelope that is randomly triggered by a metronome. The output of the enveloped sound is run through precisely timed delays, resulting in a rhythmic beat derived from the actual clarinet input. The output of this process is run through a degrade~ object with its sampling rate inversely related to the input pitch; the higher the clarinet plays, the lower the sampling rate, and the more distorted the signal gets. This adds bite to the upper end of the sound, and also causes the sound of the processing to shift throughout the movement as the clarinet's pitch moves higher. The beats also trigger low frequencies sine waves to add punch to the low end, and a compressor is used to even out the sound. The result is an interactive rhythmic engine that still maintains a fairly full sound, even without the use of non-realtime audio.

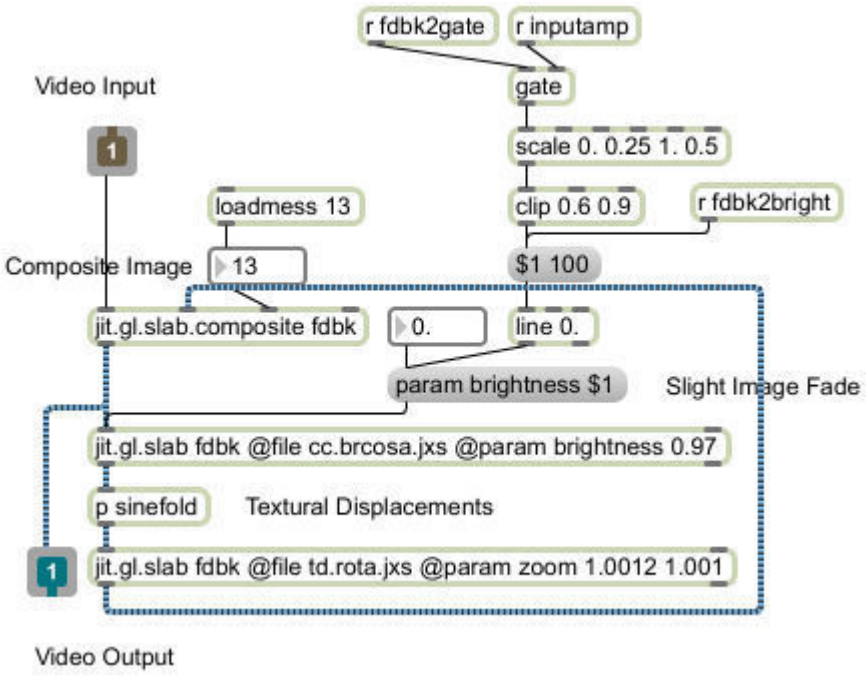
Graphics Processing

My approach to the video processing was to create processes that could have interesting interactive relationships to the audio while maintaining acceptable levels of picture quality and reliability. Reliability is a key issue on several levels; the program must be able to run the graphics with as little lagging and frame dropping as possible, the threat of the graphics totally stopping in a live performance must be extremely low, and the performer should be able to get the proper range of responses out of the system without the need for excessive calibration of the sound equipment and video parameters. While I did not manage to create a system that runs perfectly, the lags are minimal and can be improved in future versions without causing the quality of the effects to suffer.

My first decision in designing the graphics portion was to use only openGL shaders in Jitter for the manipulations. The shaders operate on the graphics card rather than the CPU; this allows the audio processing (which runs on the CPU) and the video processing to operate separately and not slow each other down. Just as most of the audio processing centers on feedback loops, the shaders are configured in feedback loops as well. The input image is entered into the left inlet of a composite shader that combines two images by keeping the brightest pixels

from each. The brightness of the composite output is lowered slightly (the rough equivalent of scaling audio amplitude down in a feedback loop) and the image is texturally displaced in some manner. The result is then sent to the right inlet of the compositor, resulting in textural distortions of the input that morph over time and can be controlled by audio parameters.

My distortion of choice for the project was a sine wave displacement shader. The sine wave's frequency, amplitude, center and phase can be altered, allowing numerous opportunities for the mapping of audio parameters. The variety of effects that can come from sine wave feedback are varied; neat blocks or lines of solid colors, glossy squares, water rippling illusions, shuddering effects and grainy distortion are all possible depending on the settings. I used two of these sine wave displacement loops, one after the other, which in combination are able to do the bulk of my processing work. One sine wave is usually controlled by the input, and the other by the processed audio, allowing both components to alter the appearance of the visuals. The audio amplitude is usually mapped to the input amplitude, though oftentimes inversely, and the scaling of the mapping changes through the different sections. Attacks in the audio sometimes trigger changes in position and frequency, and pitch changes can be used to rotate the image.



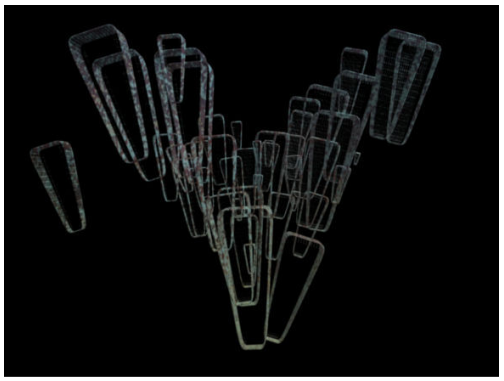
The input audio generally controls the brightness of the input source files, which are short looping animations that I created in the 3d graphics program Blender. The one exception is the rhythmic third movement, where the input brightness of an input video of colored squares is controlled by the rhythmic engine. The engine is driven by the input sound, though, so the clarinetist still has indirect control over that aspect. The section also has fixed sine wave displacements, with the visual changes derived from changes in the scale and position of the pulsing squares, which are mapped to changes in clarinet amplitude and pitch.

The development of the processing influenced my decisions in the creation of movie inputs for the graphics. Since the compositing combines the brightest parts of two images, the sources should be created on black backgrounds. Excessively complex portions of images are useless, since the feedback process washes out details that are too tiny. Also, since the processing does, for the most part, provide textural changes without creating many interesting alterations in size or location, those elements need to already be added to some of the input videos. Any masking, crossfading or non-feedback distortions should also be done ahead of time to make the processing as efficient as possible. Finally, movie files should be large enough to retain their quality, but small enough to load with little to no lag in performance.

Though slight amounts of lag remain during the loading of particular videos, I believe that I can still greatly reduce the file sizes of the problematic videos without adversely affecting their quality. I am otherwise satisfied with the relationship between sound and image in my piece, with the audio given room to have a range of effects without requiring an exact setup of the levels and calibration of the components. This is a significant step towards having a piece that can be performed without my direct guidance, provided that the performer has the software and the appropriate video files. Though complicated and intricate works can be a lot of fun to make, they're more likely to be played by other people if they are relatively easy to set up and run without requiring an understanding of the technical workings of the software.

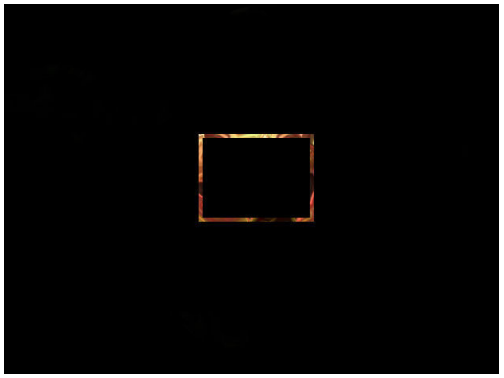
Still examples of the input movie files:

Movements 1 and 5:

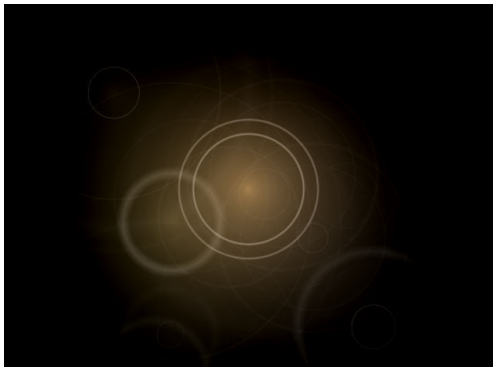


Movement 2:

Movement 3:



Movement 4:

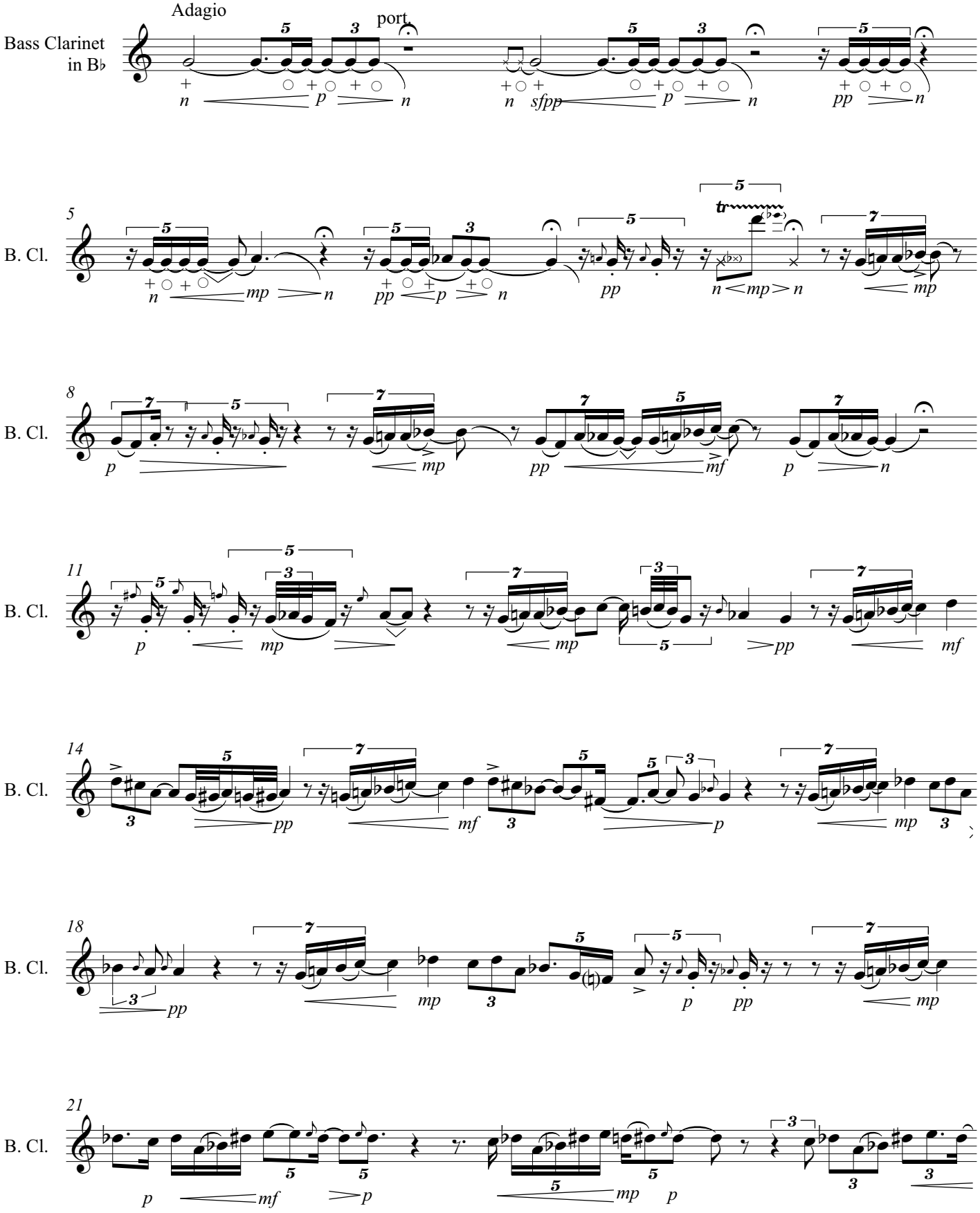


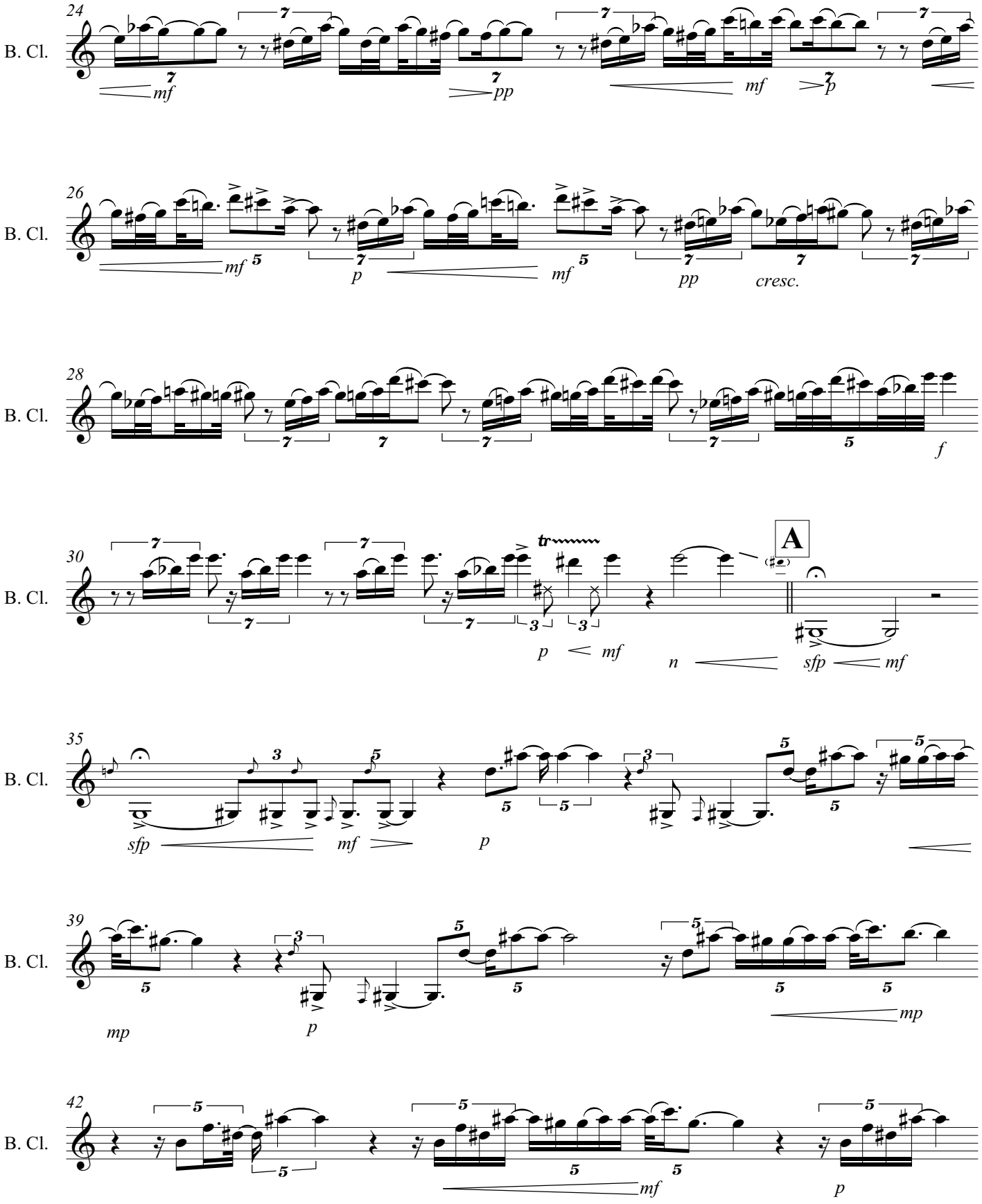
Conclusion

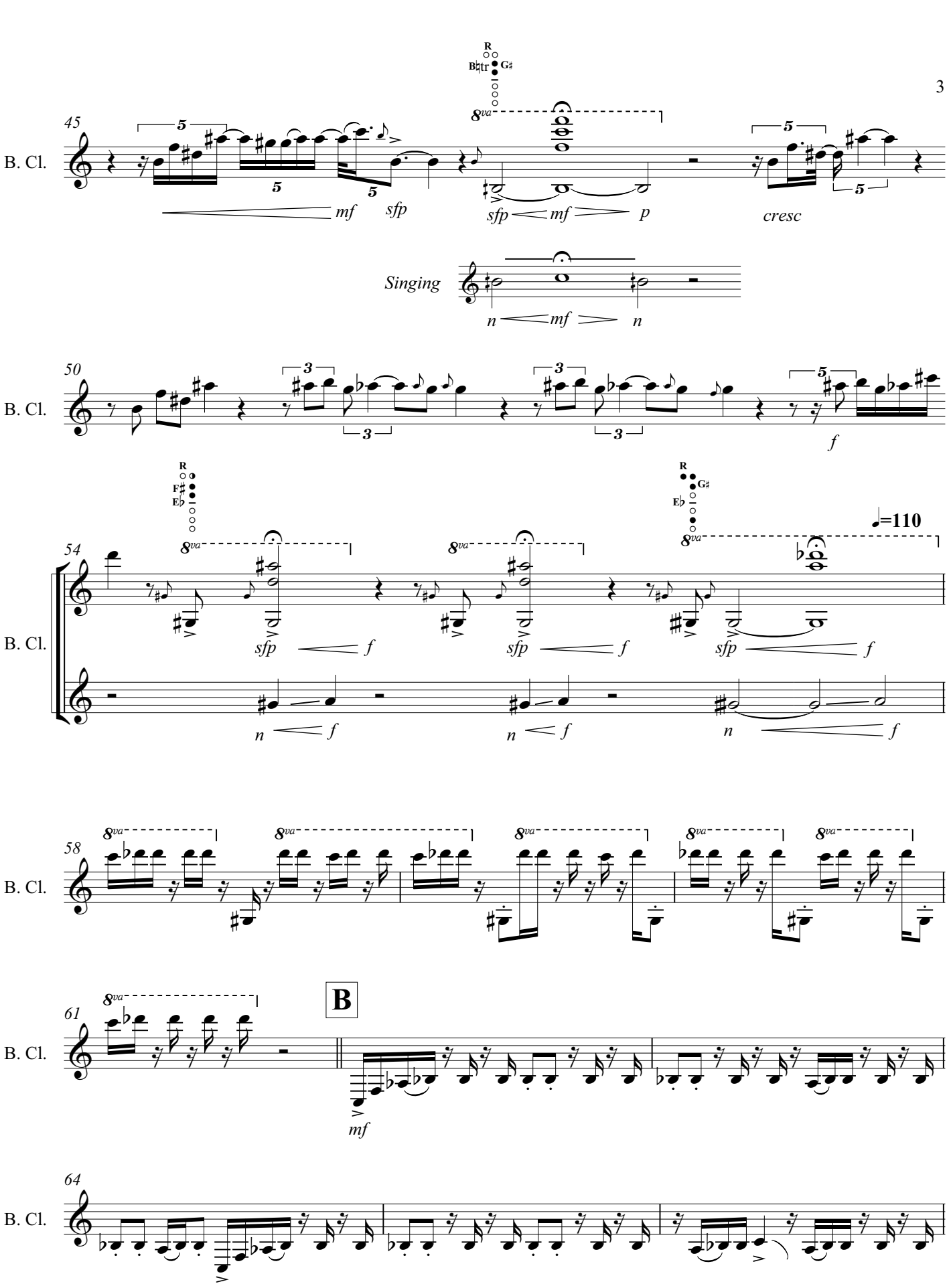
I am pleased with the outcome of the content of my piece. I feel that I managed to create reasonably clear audio and visual processes that complement each other well while adhering to the conceptual goal of processing through feedback. The piece ran fairly smoothly in the rehearsals and performance, with a minimum of technical issues. The one issue that remains is the occasional visual lag, which can be fixed with more efficient video files. Otherwise, I was able to satisfy my own artistic and technical goals, and have created a number of systems that can be used as the technical basis for future pieces.

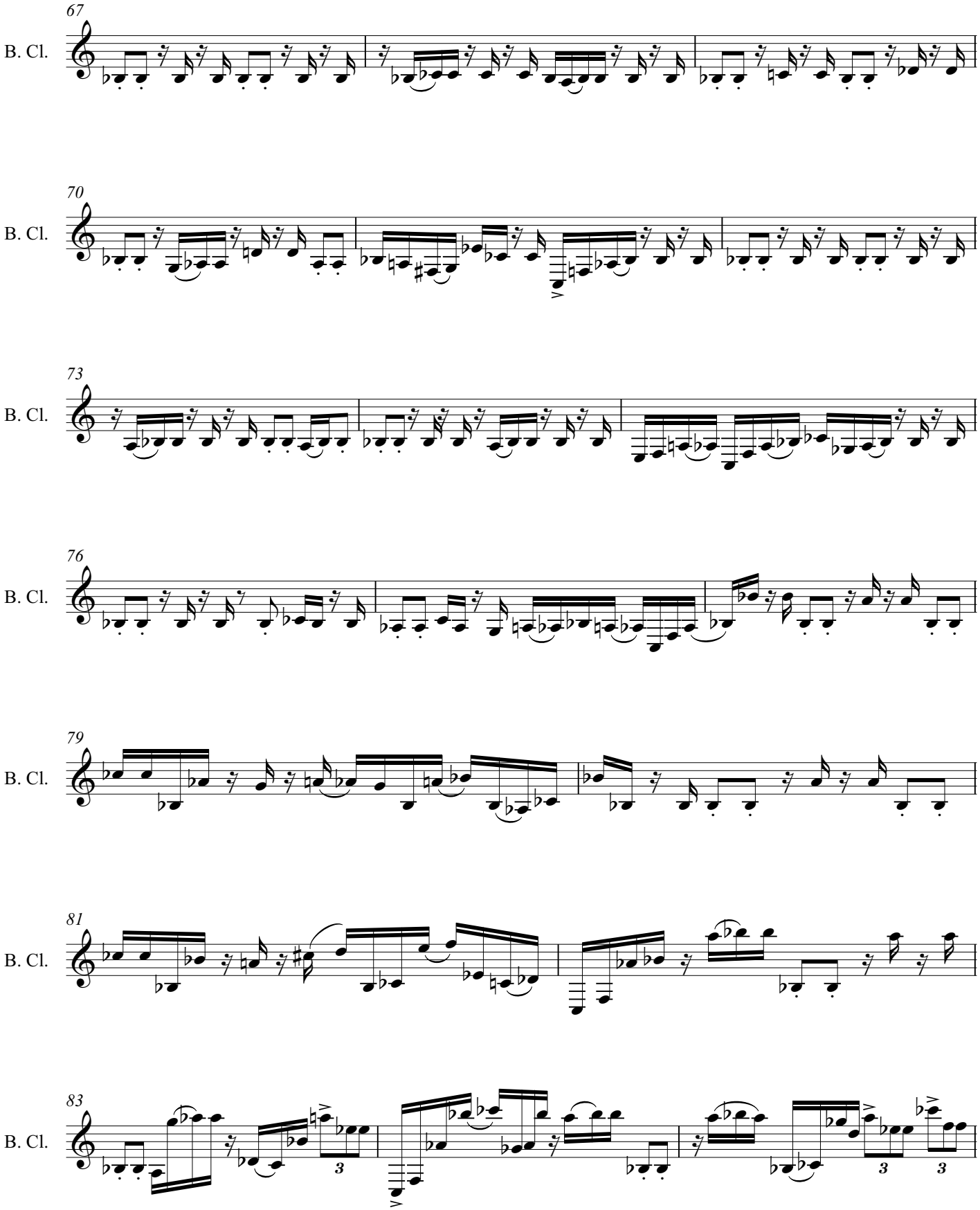
Soroboro

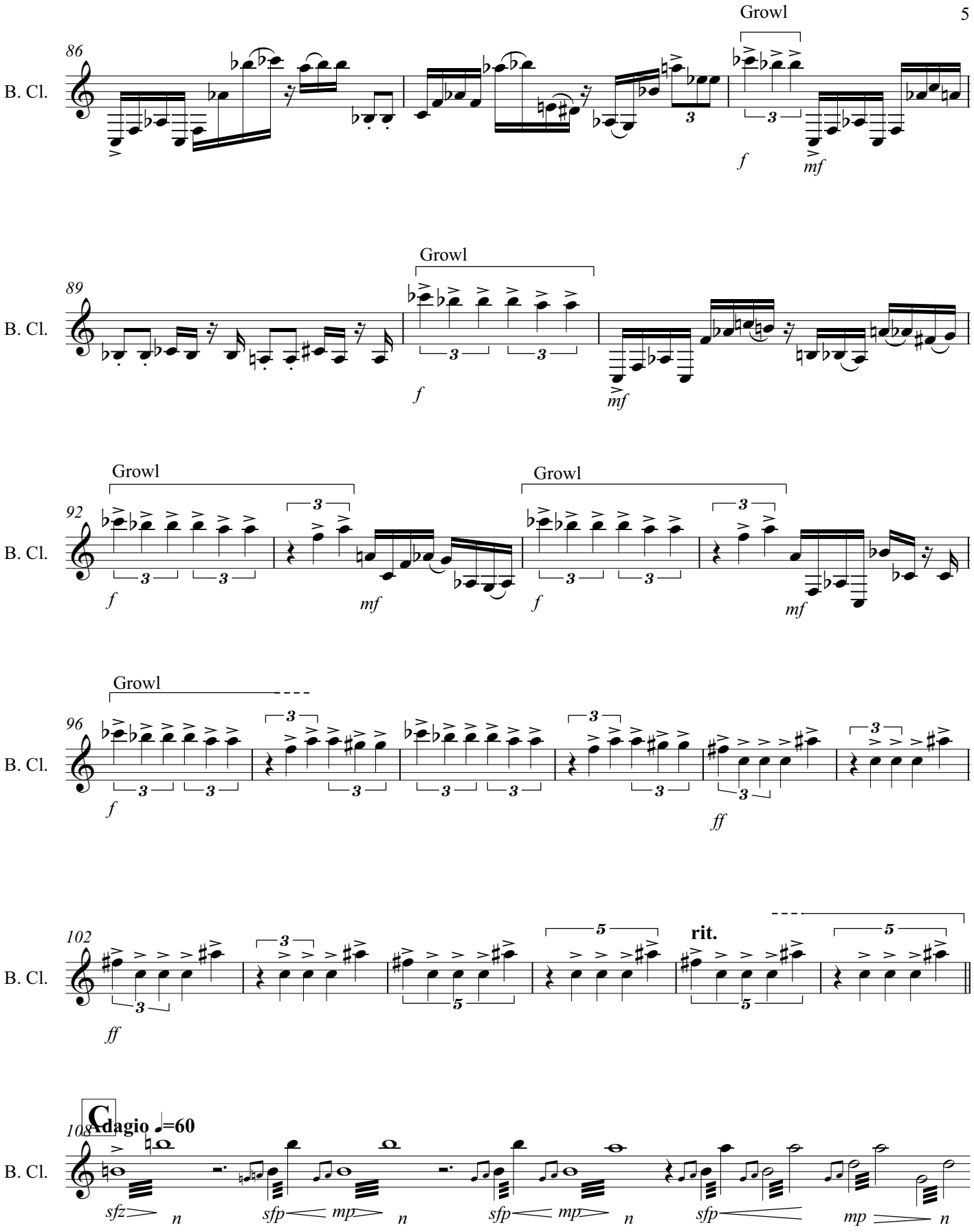
Alexander Dupuis

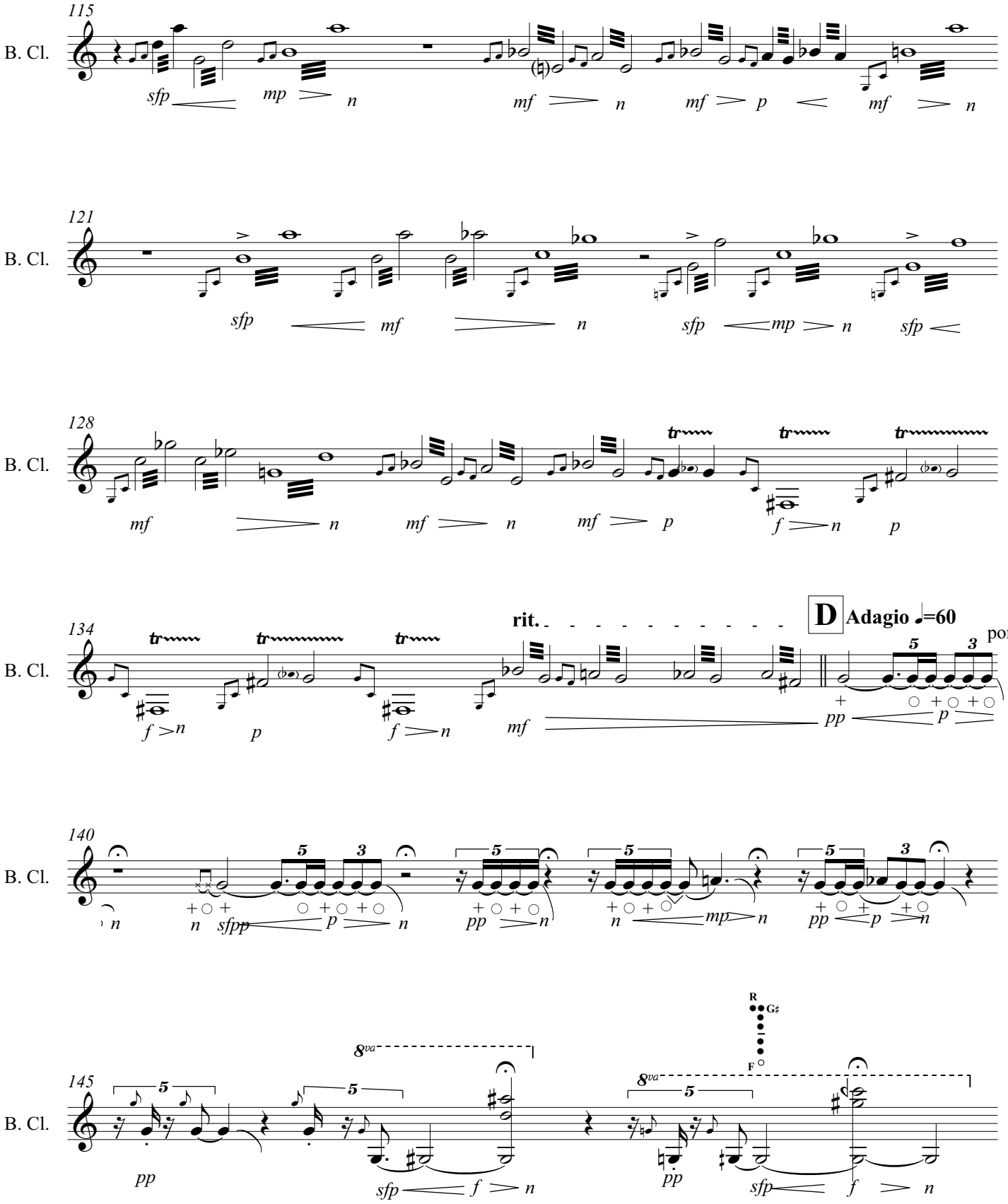


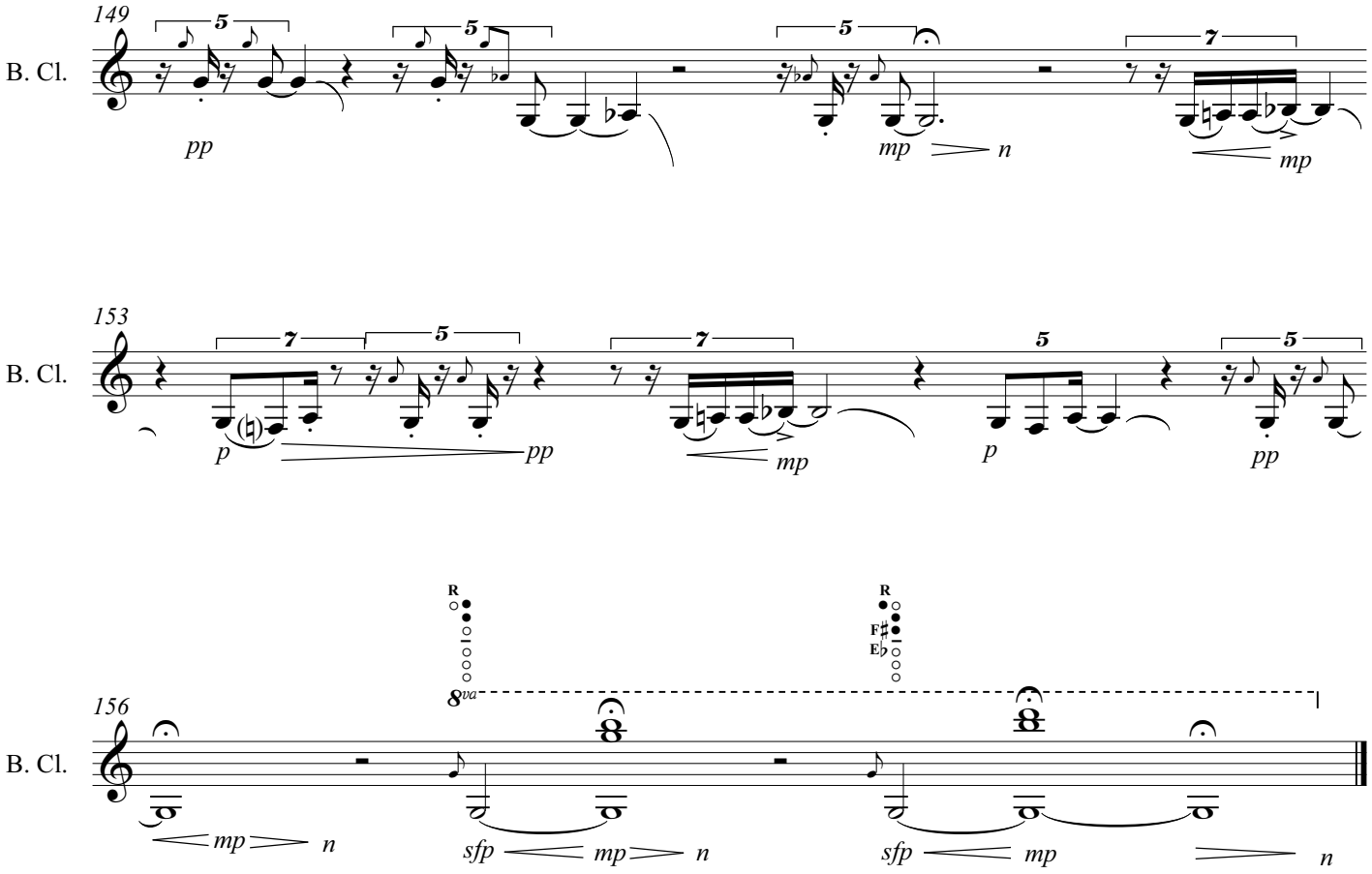

24
B. Cl.
mf
pp
mf
p
26
B. Cl.
mf
p
mf
pp
cresc.
28
B. Cl.
f
30
B. Cl.
A
p
mf
n
sfp
mf
35
B. Cl.
sfp
mf
p
39
B. Cl.
mp
p
mp
42
B. Cl.
mf
p


45
B. Cl.
mf
sfp
sfp
mf
p
cresc
Singing
n
mf
n
50
B. Cl.
f
54
B. Cl.
sfp
f
sfp
f
sfp
f
n
f
n
f
n
f
♩=110
58
B. Cl.
61
B. Cl.
B
mf
64
B. Cl.


67
B. Cl.
70
B. Cl.
73
B. Cl.
76
B. Cl.
79
B. Cl.
81
B. Cl.
83
B. Cl.


B. Cl.
86
Growl
f
mf
89
Growl
f
mf
92
Growl
Growl
f
mf
f
mf
96
Growl
f
ff
102
rit.
ff
C
Adagio ♩=60
108
sfz
n
sfp
mp
n
sfp
mp
n
sfp
mp
n


115
B. Cl.
sfp
mp
n
mf
n
mf
p
mf
n
121
B. Cl.
sfp
mf
n
sfp
mp
n
sfp
128
B. Cl.
mf
n
mf
n
mf
p
f
n
p
134
B. Cl.
f
n
p
f
n
mf
rit.
D
Adagio ♩=60
pp
p
port.
140
B. Cl.
n
n
sfpp
p
n
pp
n
n
mp
n
pp
p
n
145
B. Cl.
8va
pp
sfp
f
n
R
G♯
F
8va
pp
sfp
f
n


149
B. Cl.
pp
mp
n
mp
153
B. Cl.
p
pp
mp
p
pp
156
B. Cl.
8va
R
R
mp
n
sfp
mp
n
sfp
mp
n

Soroboro

Alexander Dupuis

Performance Notes

In all movements save the third, accidentals continue to modify notes until the next notated change. Courtesy accidentals are included at times when the modification may be unclear. The third movement contains bar lines and follows normal conventions for accidentals.