Antescofo

Matthaeus Mayr, Amrita Newton, Hannah Schmidbauer, Leonard Trommler

July 2015

University of Salzburg

This project was part of an internship program lead by Professor Christoph Kirsch and Professor Ana Sokolova, assisted by Martin Aigner.

The internship was funded by the FFG (Austrian Research Promotion Agency): Project Sommerpraktika 2015

Contents

1 Introduction

Diving into the world of Antescofo can be quite a feat. Therefore we set up an overall idea of what we'd like to accomplish as well as goals under that task. We want to create a program that will allow a computer to create live accompaniment for a singer. Now, of course, this would probably take more time than we had to refine and make perfect, however, I'd say we got pretty close. To do this, we needed to, first, learn as much as we could about Antescofo and PureData, second, create a program on PureData that allows us to play and follow a track, third, create a backing track, and finally, test, test, test! This document shows just a snippit of our work and the trouble shooting we came across in the process.

2 Pure Data

Our project is programmed in a visual programming language called PureData, using ”Messages” sent between different objects, one of the most important messages is the ”bang” message (it triggers something). It’s open source so everybody can use it. We worked with a MAC operating system in order for the program to run stably. There were some problems with the audio settings, because it always activated all signal channels and we just needed two of them.

Another problem was, that it often crashed by typing certain commands. So we had to copy complete files and edit them that we could use that commands.

3 Getting Used to Antescofo

After getting used to the programming language we started testing Antescofo itself. Our first song was "Alle meine Entchen" because it's simple to play. We had a GUI called Ascograph (see picture 4) which we used to see if Antescofo followed us. At first we tried clapping and whistling to test if it reacts. The main problems were that it skipped notes and couldn't really follow us if we changed the speed. We had to adjust the sensibility of the microphone because it reacted to every background noise. For playing with the flute, we changed the octave to the one we played in. It worked way better than before and we could play faster or slower and Antescofo still followed. With a MIDI-keyboard we recorded a simple .wav-file to back our playing. But we had to be careful because if we made the breaks too long, Antescofo continues playing and doesn't synchronize anymore.

4 Singing with Antescofo

As part of both the performance aspect as well as the regular testing of our project, we spent much of our time attempting to use live instruments rather than those hooked up to the computer. Therefor, I ended up spending a lot of my time singing along with the Antescofo.

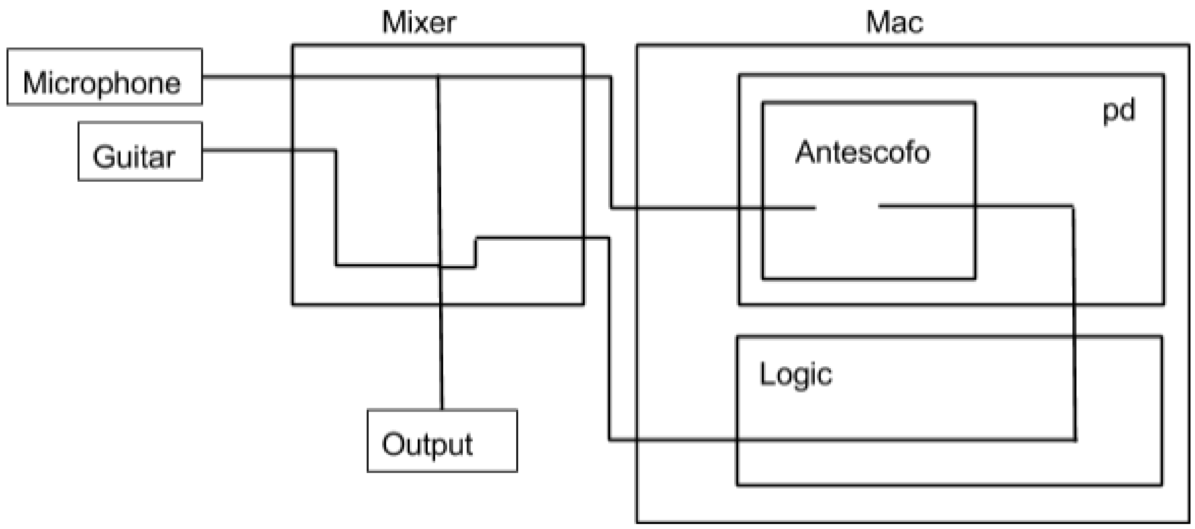
Although we had previously corrected many of the live sound related issues, by the time singing came around, we discovered a myriad of other problems. For example, what would pick the voice up (i.e. the computers mic vs. a mic attachment)? Would the computer's system even be sensitive enough to pick up a voice from a standing distance away? We tested this with a few computers, and as expected, the newer computers were far more sensitive and dealt with the voice far more accurately on Ascograph. Still, a mic was more consistent and had better sound for recording. So, we rolled out the cords and mixers and created an elaborate hook up.

The next question was rhythmic. Antescofo can pick up tune very accurately, but it's rhythmic knowledge is still growing. This means I had to sing with a metronome in my ear. The BPM of the metronome reflected that which had been in the original Ascograph script as well as the one set in Pure Data. Due to a slight lag, this made listening to the program and singing simultaneously quite difficult. Two different rhythms can be difficult, but having a backing track at the wrong rhythm, definitely gives you a headache. This problem may cause work to discontinue for the day due to a man down, and, let me tell you, no body wants that. Therefor, headphones for everyone, excluding the singer, came in handy. This also cleared any issues the program might have with hearing itself.

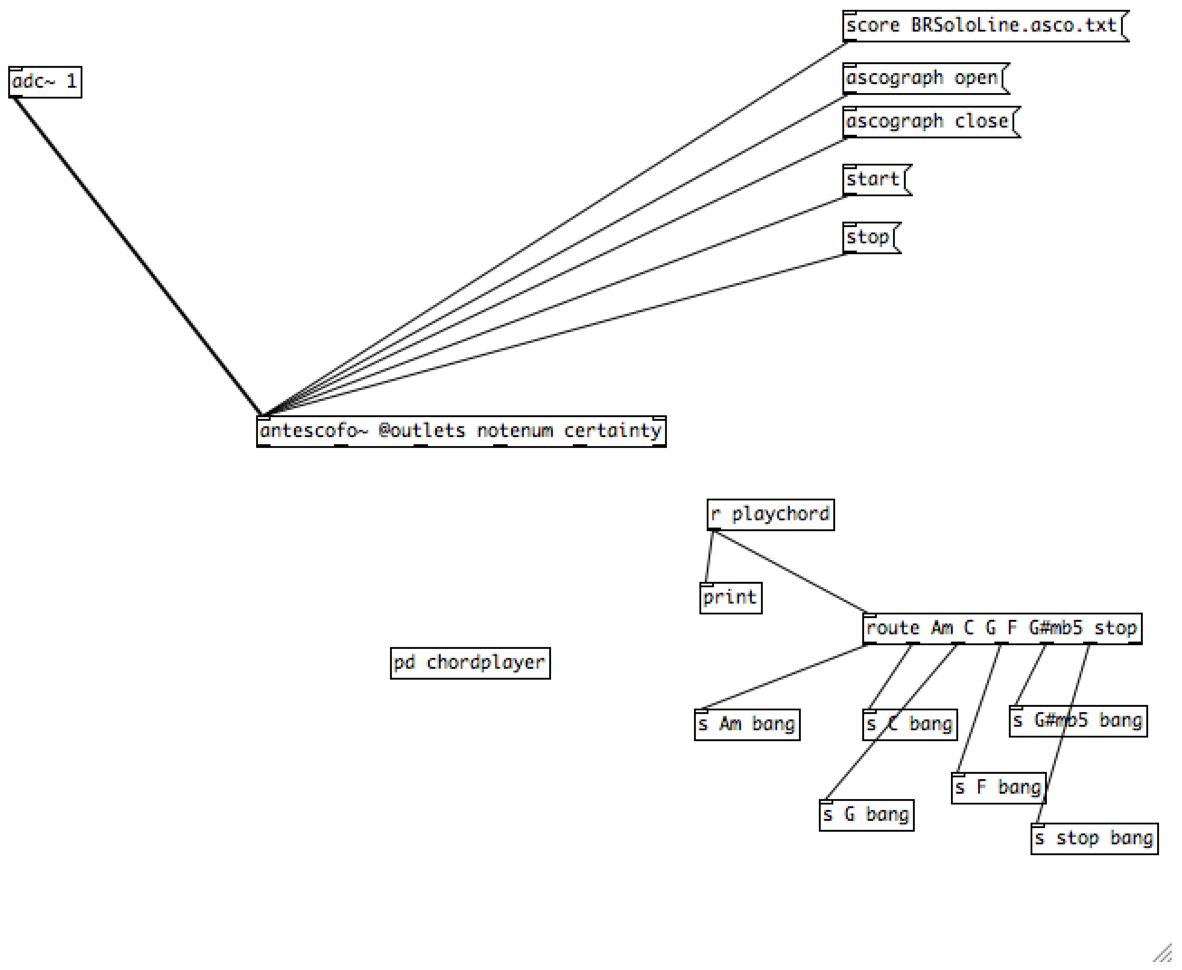
However, this also brought up the question: what if the singer went off pitch during the course of the program because they couldn't hear the back track? I solved this issue by continuously playing the tonic throughout singing the song. Later, during the recording, we solved the problem with live guitar accompaniment.

This definitely wasn't ideal, but hopefully, eventually, the lag will be close to zero, giving the singer an ideal karaoke track. At least, that's the function we had hoped to achieve. Maybe with a little refinement and more time, we will soon come to that.

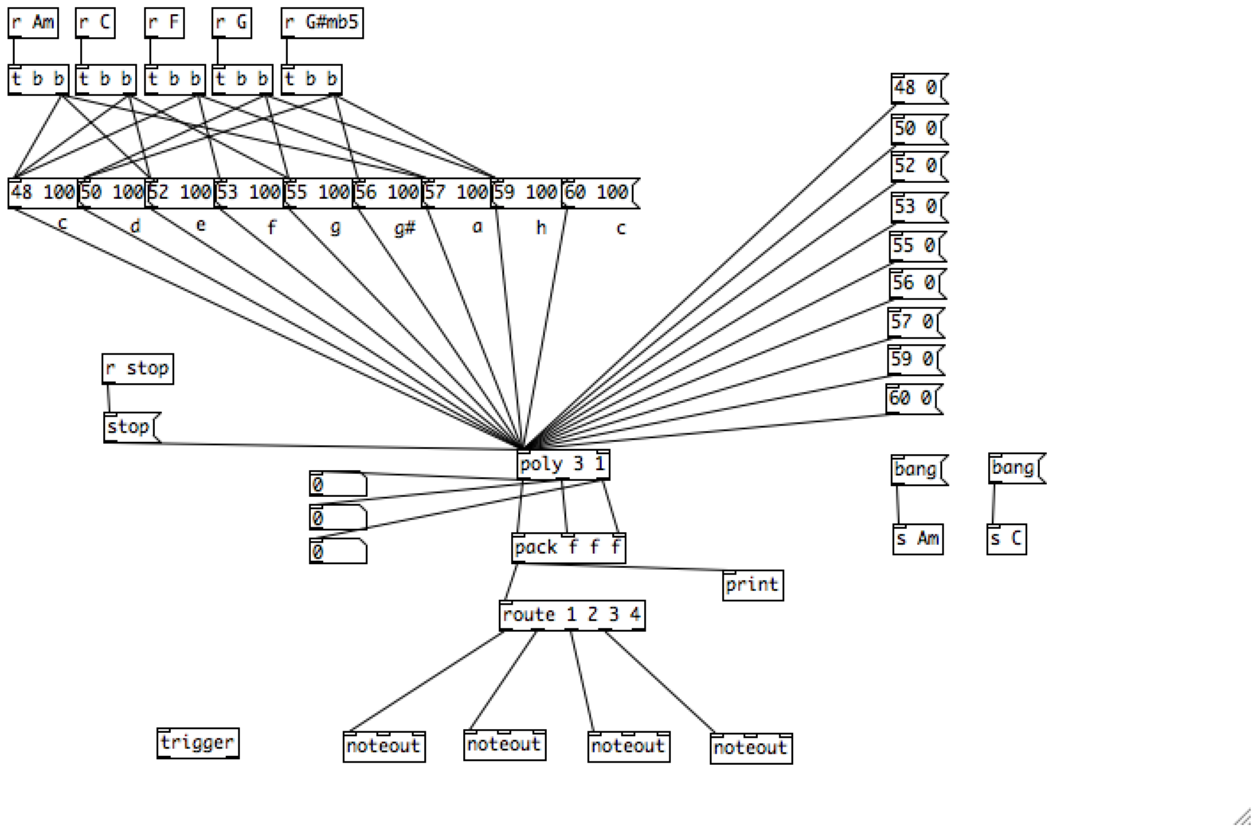
5 Our Final Setup



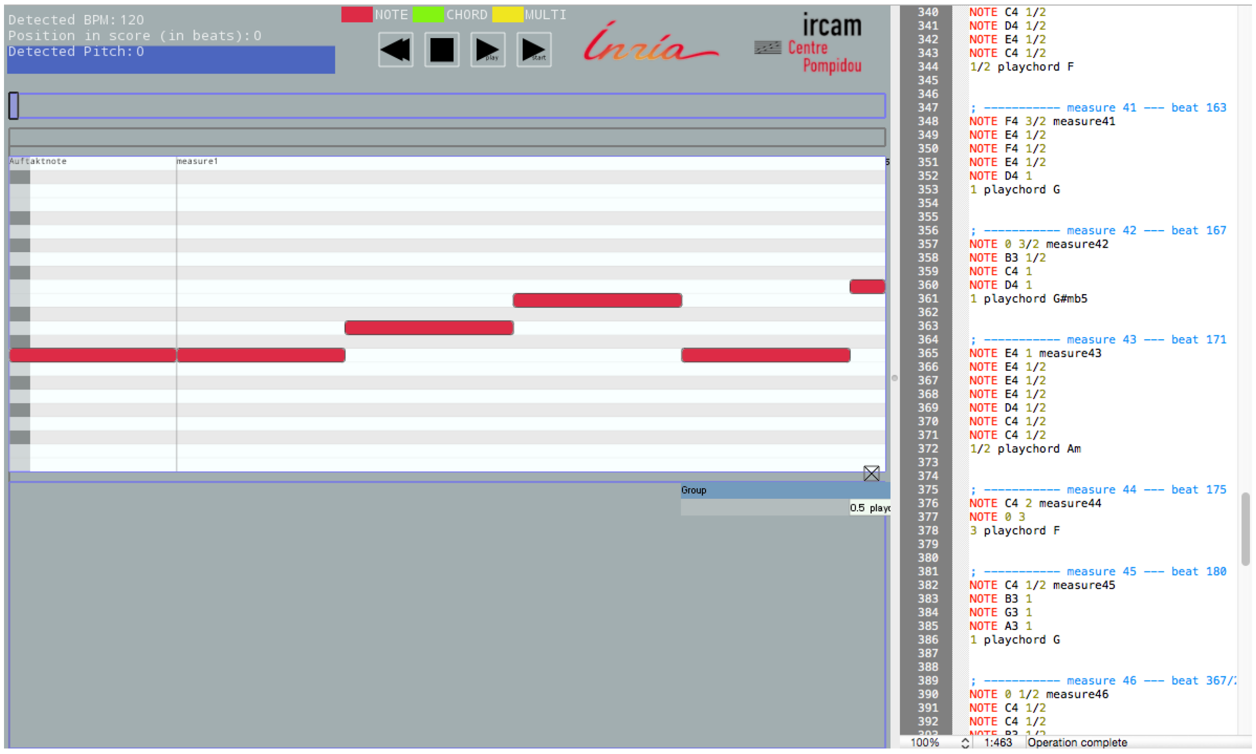
Picture 1: This Picture shows our Setup. The microphone is used to record the singing, whereas the guitar only serves to stay in the rythm. The signal of the microphone is sent by the mixer to Pd, where it will be processed by Antescofo directly. Antescofo sends the name of the chord, for example "Am", to Pd, where it is created by the subpatch "chordplayer", shown later. This chord, three midi notes, is played by Logic and gets to the standard output, which is connected to the mixer. The mixer generates one audio signal out of three (the microphone, the guitar and Logic) and sends it to an ouput, where we can connect headphones, speakers or another Mac, which records the sound.



Picture 2: This is our main patch. It contains an audio input element (adc), some Antescofo control elements (the score element, which tells Antescofo what score to use, the Ascograph open and close elements, which controls the Ascograph (Picture 4), and the start and stop elements, which activate and deactivate the score following. The elemens in the bottom right corner convert the note name sent by Antescofo (see also the part of our score visible in picture 4) into a ”bang” sent to the subpatch over the right ”channel”. The single element is our subpatch ”chordplayer”, which creates the midi chords.



This is the "chordplayer" subpatch. At the top left are elements that receive the "bang". These "bangs" trigger messages with the midi note number and the volume of the note or, in case of stop, a stop message. In the mid of this subpatch is the poly element, an element which can pack a specific amount of notes into one chord. Later, we split up this chord into the single notes, because Logic can't recognize the chords. So the only reason we use this element is that it accepts a stop command to stop the chord. The elements at the bottom are midi outputs to send the midi notes to Logic.



```
NOTE C4 1/2
NOTE D4 1/2
NOTE E4 1/2
NOTE C4 1/2
1/2 playchord F


; ----------- measure 41 --- beat 163
NOTE F4 3/2 measure41
NOTE E4 1/2
NOTE F4 1/2
NOTE E4 1/2
NOTE D4 1
1 playchord G


; ----------- measure 42 --- beat 167
NOTE 0 3/2 measure42
NOTE B3 1/2
NOTE C4 1
NOTE D4 1
1 playchord G#mb5


; ----------- measure 43 --- beat 171
NOTE E4 1 measure43
NOTE E4 1/2
NOTE E4 1/2
NOTE E4 1/2
NOTE D4 1/2
NOTE C4 1/2
NOTE C4 1/2
1/2 playchord Am


; ----------- measure 44 --- beat 175
NOTE C4 2 measure44
NOTE 0 3
3 playchord F


; ----------- measure 45 --- beat 180
NOTE C4 1/2 measure45
NOTE B3 1
NOTE G3 1
NOTE A3 1
1 playchord G


; ----------- measure 46 --- beat 367/
NOTE 0 1/2 measure46
NOTE C4 1/2
NOTE C4 1/2
```

This picture shows the "Ascograph", Antescofos graphic interface. It contains information about the velocity of the music, the midi note number and the position in the score, as well as a graphic representation, some buttons to control Antescofo and the score on the left. In the score, you can see some of the commands. The NOTE commands with the midi number and the length in beats tell Antecofo the score to follow. An action like "0.5 playchord Am" is triggered as soon as the note in the line above is played. This action for example sends, after a delay of 0.5 beats, the message "Am" over the channel "playchord" to PureData. This message is processed in the main patch and the subpatch to play the chord "Am", as explained in picture 2 and 3.