Vocal Theremin

Or "the Mongolian throat singing machine"

MUMT306 Final Project by Boyu Deng
260656496

Introduction & Objectives

In 1928, an electronic musical instrument that requires no physical contact to play was invented, later named after its creator, the Soviet inventor: Léon Theremin[1]. The instrument has two major parameters that can be controlled with distance sensing antennas: pitch and volume.

Over the century, the development of human-instrument interface has been pursuing intuitiveness, ease of use and functionality. But as electronics are industrialized and controller components are modularized and standardized, more musicians prefer to customize their own interface from start.

In this project, the objective is to purpose a blueprint for a Theremin-like control that utilizes Arduino Uno controller kit, which bridges the interactive input to a digital-domain vocal synthesizer, built with Max/MSP.

The original goal for this instrument is to recreate the sound of a vocoder controlled by a keyboard or a synthesizer. Since synthesizing the sound of vocal is intrinsically different than manipulating parameters of an input vocal, the recreation of the sound would be entirely different from just extracting vowels from a real voice; in addition, the parameters such as formants and vowel names are taken from a generic example of vowel formants.[2]

Instrument Design

For "Vocal" part, the instrument would require a base for constructing a vocal-like sound. One Max/MSP patch is used for all components involved in sound generation. In the aspect of sound analysis, the sound should compose a rich frequency spectrum to be manipulated on. In the design of this vocal synthesizer, the sound is constructed with the oscbank~ object by Max signal processing (Fig. 1). Oscbank~ object uses multiple sine wave generators at the same time, each with its own frequency and amplitude.

In this scenario, only 128 partials are used; the concern about using higher numbers of partials or template complex waves (anti-aliased square or sawtooth) is that the higher harmonics impose an unnatural acoustic representation of voice, which result in a synth-like sound rather than vocal-like. The resulting spectrum is visualized below(Fig. 2).

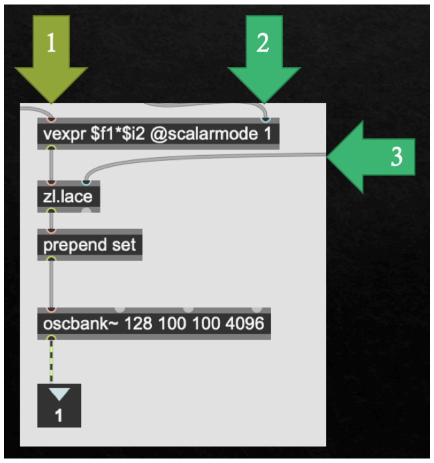

Figure 1: Constructing the canvas for vocal synthesizer
The 'oscbank~' object takes input as "set fff vvv fff vvv fff vvv…" where "fff" is the frequency of a partial, and "vvv" is the amplitude of that partial defined by the frequency in front. Entry 1 takes a fundamental frequency, while entry 2 is a list of 128 multipliers (1, 2, …) that is scaled with the fundament with 'vexpr' object. The list of 128 frequencies are then laced with a list of reciprocals of the multipliers (1., 0.5, 0.33, …) and is prepended with message 'set' to instruct the output of 'oscbank~'.

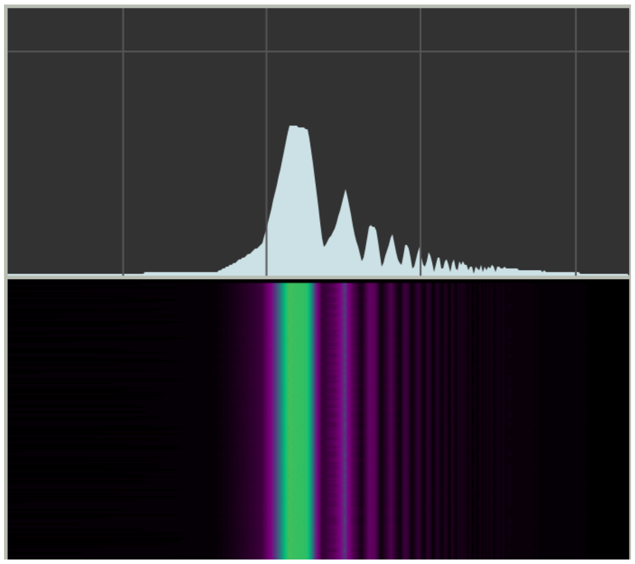

Figure 2: Output of oscbank~ at f=162Hz, no vibrato
Spectroscope(top) and sonogram(bottom) are representation of the raw sound generated from oscbank~ when the fundamental frequency input in figure 1 is at 162Hz. The horizontal axis displays a logarithmically scaled frequency domain, the vertical axis displays amplitude of corresponding frequency.

The approach for compiling the series of frequencies with a single fundamental has some advantages. One being its unified control over the pitch; based on this perk, a low frequency oscillator is placed as a frequency-modulating *vibrato* unit for the voice.

To emulate the sound generated when resonating in a human throat, two formants can be derived from properties of the resonance of the "imaginative throat". In this case, a generic model of vowels and their corresponding formants is used (Fig. 3)[2] to build the vocal.

The method of shaping the spectrum into one with desired peaks involves reson~ object by MSP: two reson~ objects are in parallel, where each takes input as the frequency of an individual formant peak. The resulting outputs are halved in amplitude, then combined into a single output.

For example, a resulting spectrum of the canvas being shaped into the "œ" vowel is shown below (Fig. 4).

Vowel (IPA)	Formant 1 (Hz)	Formant 2 (Hz)	Δf: F2-F1 (Hz)
a	850	1610	760
ɑ	750	940	190
ɒ	700	760	60
e	390	2300	1910
ɛ	610	1900	1290
ɤ	460	1310	850
i	240	2400	2160
o	360	640	280
ø	370	1900	1530
œ	585	1710	1125
ɶ	820	1530	710
ɔ	500	700	200
u	250	595	345
ɯ	300	1390	1090
ʌ	600	1170	570
y	235	2100	1865

Figure 3: The vowels used in the construction of vocal synthesizer[2]
The table displays each IPA letter represented by a vowel. The formant 1, 2 indicates the formant peaks that corresponds to that vowel. Δf shows the difference in frequency between the two formant peaks.

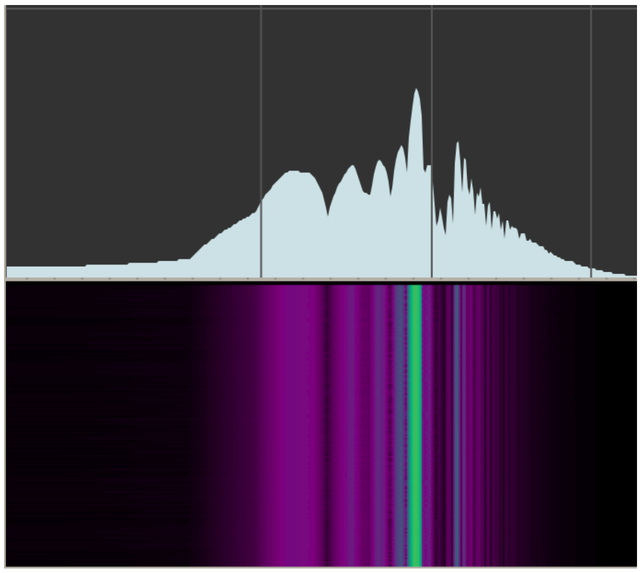

Figure 4: Vowel shaping to œ at f=162Hz, no vibrato
The signal is taken from the example showed in figure 2, passed through two reson~ objects (each resonates at 585, 1710Hz) in parallel, then combined into a single output.

The problem of continuity emerges as the formant peak values of vowels are discrete; unlike the schematic of what figure 3 displays, a voice that pronounces vowels will have smooth transitions between the vowels, especially in the context of this project: a basso continuo singing of series of vowels. One solution that stayed in the final version of this vocal synthesizer is by using ‘pattr’ and ‘pattrstorage’ objects.

Unlike ‘preset’, ‘pattrstorage’ allows a float value to interpolate between stored instances of snapshots. For example, a float number 1.50 as the input of ‘pattrstorage’ will result in the ‘pattr’ connected objects to have values stored between instance 1~2.

To play the instrument, one needs to simultaneously control 5 parameters, mentioned previously above: pitch/frequency, velocity/amplitude, vowel, vibrato rate and depth. The choice of controller to use may directly related to how difficult it is to learn and play the instrument. In fact, four controllers are mapped to the Arduino board to provide inputs; while each control method is intuitively related to the property of the parameter it is mapped to.

For pitch control, an ultrasonic sensor is mapped to the fundamental frequency of the synthesizer. The linearity of the values that ultrasonic sensor gives fits the conventional impression of pitch being on a linear scale; moreover, ultrasonic sensor is highly accurate due to its short wavelength, both its accuracy and the contact-free style of manipulating pitch is perfect for simulating the Theremin. The sensor takes a digital input(Trigger) and provides a digital output(Echo) when powered under 5V; if the digital input is set to “HIGH”, the ultrasonic unit turns on and triggers a pulse until the input is set to “LOW”; when the ultrasonic sensor receives the returning pulse, the “Echo” pin outputs a digital value “HIGH/LOW”. With the help of a pulsein() function, the duration between send and receive can be quantified, hence the value from pulsein() is in proportion to the distance between sensor and the reflective object.

To control velocity/amplitude of the synthesizer, a potentiometer is used.

For interpolating vowels, a photoresistor is used to provide a continuous float output. The reason of using a photoresistor is that, it requires no contact, but most importantly, it is intuitive to the user. Changing the environmental luminosity will alter the resistance of the sensor hence its output, much alike changing the timbre of a sound by hindering or resonating its sound source (eg. covering a speaker with hands, listening to a sound from a tube or a cone).

For modulation of vibrato, a 2-axis joystick is applied. Since the joystick both physically and digitally resets to [0,0] if unattended, the parameter output from this component is taken the absolute value of its original output.

All of the quantified controller values are sent as a list through USB serial connection, the circuit diagram and connections are shown below(Fig. 5).

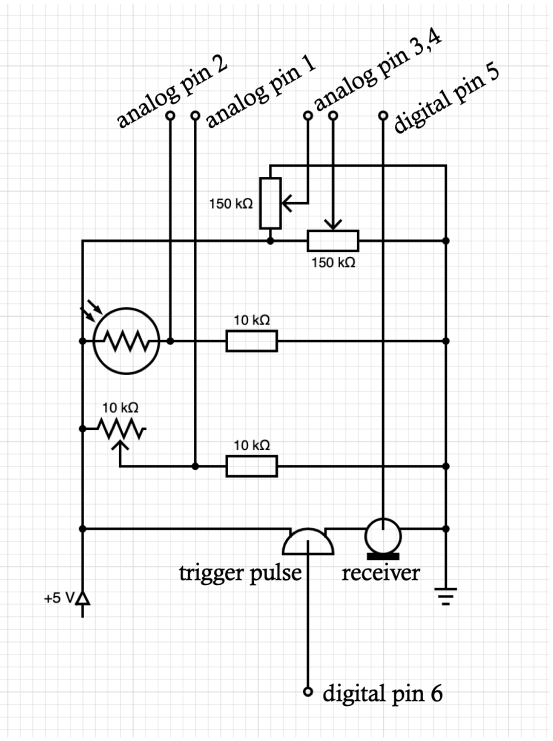

Figure 5: Circuit diagram of Arduino control of the Vocal Theremin
The circuit is originally constructed on a breadboard, using a single source of +5V power source and ground. Pins are connected to the Arduino board accordingly, the two variable resistor for analog pin 3, pin 4 are from the joystick.

<u>Discussion & Improvement</u>

There are some challenging problems that emerged during the development of the Vocal Theremin:

The ultrasonic sensor, despite being the perfect candidate of a straightforward control, has some significant drawbacks due to its intrinsic property. Ultrasound has a short wavelength, which means it is sensitive not only to the distance of the reflective surface, but also the angle that it is place at the reflective event; using bare hand to control the ultrasonic sensor results in an output that is tested to be massively unstable. This is solved in two steps; using an augmentation to the hand, for example, attaching a flat plastic board to the hand, allows even reflection and good response for the ultrasonic echo input; the other step is using 'rampsmooth~' object to tame the signal into having smaller fluctuations and smoother curves.

Some improvements can be made with this instrument:

The potentiometer for velocity control is a stable source of constant output, but it could be replaced by a piezoelectric device for a more intuitive relation.

The synthesis part of the instrument can use some more tweak. The result is not far from a vocoder-like recreation, but it is still quite robotic and sounds unrealistic. The other approach beside synthesize vowels from scratch, is to extract vowels from a recording, even a real-time audio input, which would require Fourier transform and heavy signal processing.

<u>Appendix</u>

FP.maxpat – the Max patch that contains the synthesis part of Vocal Theremin

ffff.json – autoloaded .json file that stores pattrstorage information in FP.maxpat

FP.ino – the Arduino code uploaded onto the Arduino Uno board

CD.png – the circuit diagram outlines the connection of controller components and Arduino Uno board

Demo.mp3 – a short recording a performance using the Vocal Theremin

Reference:

1. Glinsky, A., *Theremin: Ether Music and Espionage*. 2000: University of Illinois Press.
2. Catford, J.C., *A Practical Introduction to Phonetics*. 1988: Oxford University Press.