Mathieu Blanchet

Gary Scavone

MUMT306

7 December 2020

Final Project: The HRMNZR

Objective:

The main objective of this project was to create a Max/MSP patch capable of harmonizing any given note in any given key played from a MIDI controller. The user would be able to choose which key they are playing in via a drop-down menu, toggle between major or minor modes, then play a bassline on a MIDI keyboard which would be automatically harmonized by the patch. Additional functionalities of this patch include the ability to introduce ninths in the harmonies via another toggle button and the ability to choose between five different voicing combinations.

Methodology:

Initially, it was quite easy to create a Max/MSP patch where harmonies were generated from incoming MIDI data from a USB MIDI keyboard. Using a "+" object, the received MIDI pitch would be sent to the right inlet and a message box of specified value would be sent to the left inlet Figure 1). Each time the user would plays a note, a minor third would be heard above the played pitch. At first, the patch was only designed to work in C major or minor in order to ensure that a working patch would be achieved. A sub patcher called "notes" was then created in order to compare the incoming pitch with different select objects, which would identify the note name of the played note. This meant if pitch 60 was played, then the sub patcher would identify it is a C and output a bang from the first outlet if the patcher (Figure 2).

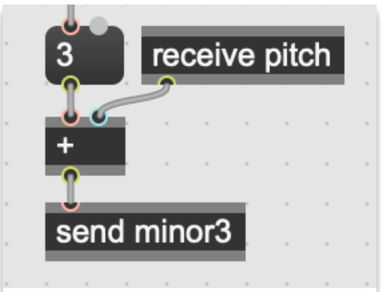
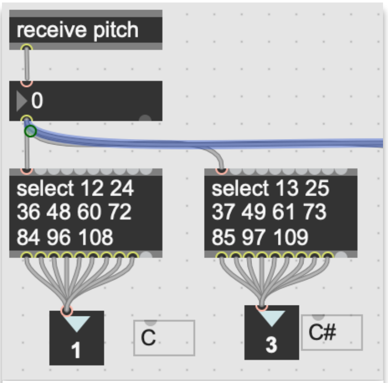


Figure 1, Basic adding structure


receive pitch
0
select 12 24 36 48 60 72 84 96 108
select 13 25 37 49 61 73 85 97 109
1
C
3
C#
Figure 2, Notes sub patcher

From there, the bang would go through the correct message boxes to create the appropriate harmonies. For example, if a C was played in C major, then a major third, perfect fifth, and major seventh would be banged and sent to the noteout object. It then became a matter of sending the correct bangs to their respective harmonies. Below is a table showing which harmonies are played for each scale degree:

Scale Degree	Harmonies		Chords Formed
1	Major: 1, 3, 5, 7	Minor: 1, b3, 5, b7	Major 7th, Minor 7th
b2	Major: 1, 3, 5	Minor: 1, 3, 5	Major Triad
2	Major: 1, b3, 5, b7	Minor: 1, b3, #4, b7	Minor 7th, Half Diminished
b3	Major: 1, 3, 5, 7	Minor: 1, 3, 5, 7	Major 7th
3	Major: 1, b3, 5, b7	Minor: 1, b3, #4, b7	Minor 7th, Half Diminished
4	Major: 1, 3, 5, 7	Minor: 1, b3, 5, b7	Major 7th, Minor 7th
#4	Major: 1, b3, #4	Minor: 1, b3, #4	Diminished Triad
5	Major: 1, 3, 5, b7	Minor: 1, 3, 5, b7	Dominant 7th
b6	Major: 1, 3, 5, 7	Minor: 1, 3, 5, 7	Major 7th
6	Major: 1, b3, 5, b7	Minor: 1, b3, #4, b7	Minor 7th, Half Diminished
b7	Major: 1, 3, 5, 7	Minor: 1, 3, 5, 7	Major 7th
7	Major: 1, b3, #4	Minor: 1, b3, #4	Diminished Triad

For scale degrees whose formed chords alter between the major and minor mode, a gswitch object was used to send the received bang to either the major or minor thirds and sevenths, as well as the perfect fifth or tritone. Using a toggle object, the user could activate these gswitch objects and toggle between the major and minor modes.

The biggest challenge was to incorporate the functionality where the user selects which key they want to play in. Using a dropdown menu, an outputted value would be sent to a select object which would bang the respective chosen key. In other words, if the user selected the key of D, then an output value of 2 would be sent to the select object, which would then output a bang out of inlet 3 and in turn send a bang to an

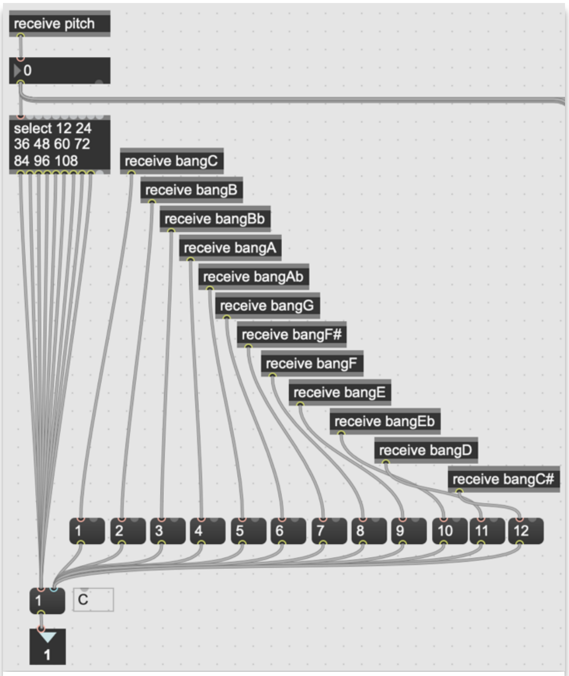


Figure 3, Notes sub patcher improved

appropriately labeled send object. Each key has a respective send object. In order to change the key of the patch, a message box system was set up in the “notes” sub patcher where 12 numbered message boxes would receive different bangs depending on the chosen key (Figure 3). These message boxes would set the scale degree of each note in the octave. For example, if the key of F was chosen, then F would become 1, G would become 3, C would become 7, and so on.

At this point, the Max/MSP patch was able to read which key was being chosen and attribute the appropriate scale degree to each chromatic note of the octave, play suitable harmonies for any scale degree, as well as toggle between major and minor tonalities for any given key.

The next step was to incorporate the voicing functionality. In a new sub patcher named “harmonies”, bangs from five different buttons available to the user were received. Five different voicing combinations were created using a range of two octaves. Here is a table of the harmonic order for any given voicing available in the patch:

Voicing	**Harmonic order**
Open Voicing	1, 5, (9), 3, 7
Closed Voicing	1, 3, 5, 7, (9)
Octave Higher	1, [3, 5, 7, (9)] one octave above 1
Jazz Voicing I	1, 5, 7, 3, (9)
Jazz Voicing II	1, 7, (9), 3, 5

In the “harmonies” sub patcher, depending on which voicing was chosen, appropriate message boxes are banged to a send object which sets the harmonies in the right octave (Figure 4).

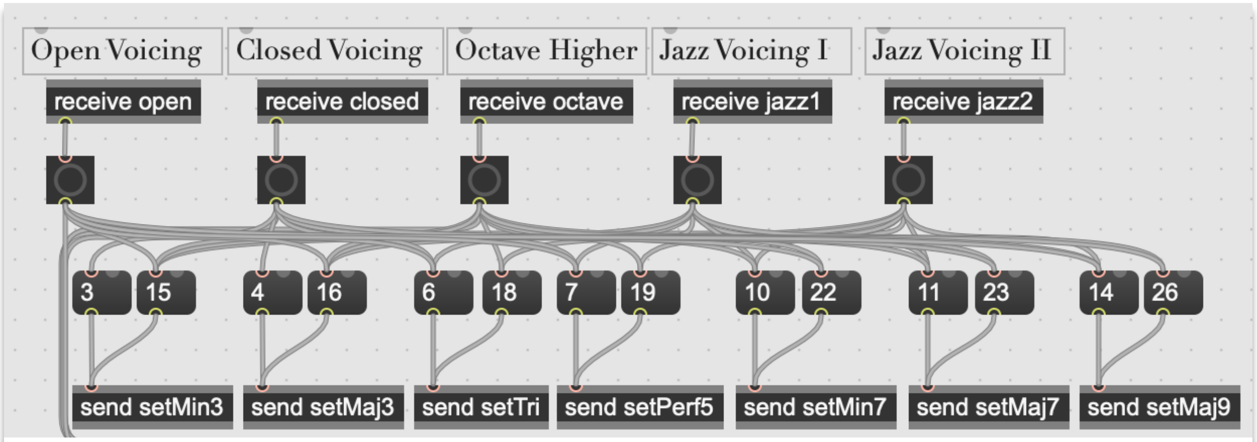


Figure 4, “harmonies” sub patcher

Additionally, a message box containing the respective voicing name is banged to another send object in order to display the chosen key to the user.
Finally, an extended harmony toggle was implemented which adds a major ninth harmony to most chords in the scale. This was done by using another gswitch object to toggle between sending or blocking the bang from getting through to the major ninth harmony.

Results:

The resulting Max/MSP patch achieves every objective that was set. Using presentation mode, the patch features a user-friendly GUI where one can chose a key, toggle between major and minor modes, chose between five different voicing combinations, and add extended harmonies. Using a USB MIDI controller or MIDI software such as MidiKeys, the user can press a single note and generate suitable harmonies for the chosen key. The harmonies are only heard when the user presses a note, making this a fun and intuitive tool to easily create new chord progressions.