RandoMIDI – Algorithmic Composition Engine

This application has been built to be used to generate and control MIDI data. The building of the patch has utilised a range of Max 7 objects that allow the user to break down and influence an aleatoric composition process, creating an algorithmic composition engine patcher. The patcher has been built using techniques covered in taught sessions and Max 7 help patches/tutorials. Submitted in this assignment is the following; a Max patcher, a Max application, a Max for Live device and a written commentary and conclusion of the building and completion of the submitted materials.

Overview of Algorithmic Compositor Features

The completed patcher, shown in *Figure 1* (below) going by the name of *RandoMIDI,* offers the user a variety of controls and parameters to influence the aleatoric composition process. The user is able to control the composition of a drum kit and three synthesizers, influencing the genre of the composition, the rate at which they compose, the key and scale of the composition and more. Since the patch is MIDI based, and utilises the operating system's in-built AU DLS Synthesizer, the patch is primarily self-contained and ready to use in an instant.

Figure 1 – User interface of the 'RandoMIDI' Algorithmic Compositor

How to Use the Algorithmic Compositor

The user can start and stop the composition process at any time by using the Toggle button to the left of the interface, found next to the Tempo value, which can also be changed by the user at any time. The user is provided with a visual response of the start/stop status and the current tempo, beat and tick count.

Upon starting the patch, the synth sounds will be played and recorded, but in order for the drums to be heard, the user must operate the Genre menu to the left of the patch, selecting a desired genre for the drums to be played in, which includes Techno, House, Electro and Trance. By default, the patch opens with the genre set to 'Off', but the user can change the genre of the drums at any time.

The Roll menu allows the user to add a roll/loop effect to the drums and the synth sounds. The Roll menus default setting is 'Off', but the user can operate the Roll menu at any time, choosing a fraction for which the sequence will be looped at.

The drum sequencer has two further menus; Pattern and Mute. The Pattern menu can be operated to change how often the pattern will be randomised. Each time the pattern is randomised, the sequence changes within the selected genre. The user can use the Pattern menu to set how often (in Bars) the pattern will randomise, or the user can turn the randomisation off. The user may also randomise the pattern at any time using the Randomise Pattern button, found below the Pattern menu. The Mute menu allows the user to set how often (in Bars) the individual drum instruments will be muted and unmuted randomly. Like the Pattern menu, the user is also able to turn the Mute randomisation off by selecting the 'Off' setting. In the lower of the interface, a Mute menu is seen, which is also operated by the Mute menu to give the user a visual feedback of the randomisation. The user may mute and unmute individual drum parts at any time using the lower Mute menu. Found next to the Randomise Pattern button, the user can operate the Swing dial to increase or decrease the swing amount in the sequence, influencing the drums and the synth sounds with a clean swing effect.

The user is able to choose the signal path for the MIDI data generated by the drum sequencer using the Drums MIDI Out menu, found to the left of the interface below the Tempo marker. By default, this menu is set to 'AU DLS Synth 1', which uses the in-built operating systems synthesizer. Found below the lower Mute menu is the Randomise menu. Here, the user can use the dials assigned for each individual drum instrument to further randomise its pattern, choosing to add discreet input from the randomiser or to max it out and go wild. These dials also give the user a visual response of when each drum instrument is played, by flashing the LED in the centre of the dial whenever each instrument is hit. The user can choose to allocate the randomisation effect to be pre-mute or post-mute by using the Pre-Mute and Post Mute buttons to the left of the Randomise menu. By selecting the Pre-Mute option, the randomisation will only occur when the instrument is unmuted. By selecting the Post-Mute option, the randomisation will occur at all times, even when the instrument is muted.

To the upper left of the interface, the user is given the controls for the synth sounds, which include Bass, Lead and String sounds. The three synth sounds are given their own menu operating in bars, much like the drum sequencer. The user can select how often in bars the patch will randomise the voice and the sequence of each instrument using these menus. Much like the drum instruments, the user is also given a visual response of the sequence with the LED's found under each instruments bar menu, buttons to mute and unmute each instrument and a MIDI Out destination, which is set to 'AU DLS Synth 1' by default. However, for each instrument the user is also given a menu for the voice of the synth, which can be operated at any time by the user and by the instruments Bar menu.

To the right of the synth interface, the user can change how often the key of the sequence is changed and how often the synths mute settings randomise. The user may change the key of the sequence by operating this Key menu and by operating the lower Key menu, found just below. The keyboard gives the user a visual response of the selected key, chosen by the Key randomisation or by clicking a key on the keyboard to set a new key. The user can operate the MIDI In menu to the left of the keyboard to select a MIDI input to influence the key of the piece, allowing the user to use a MIDI keyboard or other devices to change the sequencers key.

Found below the Key menu is a Scale menu, allowing the user to select from a range of ten scales for the user to choose, influencing the scale at which the random synth sequences will be played in.

To the far right of the interface, the user can operate the Preset menu, allowing the user to recall a range of 6 presets to explore.

Table of Artistic Goals and Technical Realisation

Artistic Goal	Technical Realisation	Ref. No:
Patch opens in presentation mode displaying only user interface objects with a screen size suitable for a 13" non-widescreen monitor	For this patch to open in presentation mode, the 'Open in Presentation' parameter in the Patcher Inspector was toggled. To utilise presentation mode, the appropriate objects that were chosen for the user interface were toggled to be displayed in Presentation Mode, arranged appropriately for a 13" non-widescreen monitor to provide a suitable interface for the user.	1
Use Patcher Objects	This patch was built with the use of patcher objects, helping to organise the objects in patcher view. [Patcher] objects were used to create a self-contained patcher, along with [inlet] and [outlet] objects to provide access to the patch for external objects. By appropriately grouping objects in to patchers, the signal flow can be easier to follow and the objects can be grouped appropriately.	2
Replicate drum sequences from 4 genres	This patch starts with the drum sequence generator; the user is able to choose a genre of drums for the patcher to generate; Techno, House, Electro or Trance. In order for the patcher to generate interesting and unique drum sequences for these genres, a range of sequences were composed for each instrument, such as the kick, snare, hi-hat etc. following the specified genre. When the user alters the genre parameter of the patch using the [textbutton] objects shown in *Figure 2*, the patcher alternates the [counter] objects signal flow to the appropriate patcher for the selected genre, shown in *Figure 3*. Found in each genre patcher is a range of patcher objects for each drum instrument, containing sequences for the appropriate instrument in the selected genre. For the sequences to be played, a [tempo] object is used to output number values of different ranges to a specified tempo value, which is defined by the user via an integer object in the user interface. To start and stop the tempo object, a [key] object, [select 32] object and a [toggle] object trigger the start and stop of the tempo object at the press of the space bar. The toggle object is also shown in the user interface, as it can be triggered but it also shows a visual response of the start and stop status of the tempo object.	2, 12

Roll feature	A roll feature was added to the patch to add an interesting effect for the user to play with. The roll uses [gate] objects to alternate the value of the metro [counter] object, displayed in *Figure 5*. As the gate object counts the 16 ticks produced from the [metro] object, the [gate] objects re-trigger the pattern with different values using appropriately numbered [message] objects. As the gate object counts the 16 ticks, the [message] objects interpret different values from the ticks, producing reduced, or fractioned versions of the value of the tick.	2, 20
Pattern and Mute Randomiser	The patch is able to randomise the patterns for each drum instrument and randomise the mute of each instrument. Found in each drum genre patcher object is a series of [random] objects, as shown in *Figure 4*. These [random] objects receive a bang specified by the [counter] objects in the [p barcount] patcher, shown in *Figure 6*, which is specified by the [textbutton] object series in *Figure 7*. Selecting a [textbutton] object from *Figure 7* will alter the signal flow of the [p barcount] patcher in *Figure 6* by altering the [gswitch2] objects to allow the signal to progress from the appropriate counter whilst closing the signal from the other counters. The signal from the [counter] object, which counts the relevant number of bars specified by the [textbutton] objects in *Figure 7* then triggers the [random] objects shown in *Figure 4.* These [random] objects send newly generated number values to the [patcher] objects for each percussion instrument, seen in *Figure 4*. These values travel through a [select] object in each instrument [patcher] object, selecting a pattern from the range of pre-sequenced patterns for each instrument, which can be seen in *Figure 8.* When the [random] objects are triggered in *Figure 4*, each instrument changes in pattern, or sometimes uses the same pattern as it may receive the same number. This is a simple and effective way to compose new drum sequences in an instance. After travelling through a large amount of [patcher] objects and [gswitch] objects, the signal from the sequence patterns eventually reaches a [makenote] object, which creates a note to be sent to a [noteout] object. This sends a note to the patchers default synth, the AU DLS Synth. Each note of the drums has a different pitch, specified by the [makenote] object, relating to the instrument of the drum kit.	2, 3, 4, 7, 13, 14

	To add further variation to the compositional process of the patch, the user is able to mute and unmute individual drum parts, but is also able to specify these parts to be muted randomly at specified intervals, similar to the process used in randomising the pattern. A duplicate [p barcount] object is used, just like the patcher shown in *Figure 6,* which is also operated by a duplicate series of [textbutton] objects as shown in *Figure 7*. The [counter] objects send their signal to a [p randommute] object, shown in *Figure 9*. This [patcher] object uses a strip of [random 4] objects to generate a random number between 0 and 3 for each individual instrument. Next, each [random 4] object is followed by a [select 1 2] object, which selects the values 1 and 2 from the newly generated number. If the number happens to be a 1 or a 2, the [select 1 2] object will send a bang to the [message] object with 1 in, which will unmute the instrument. However, if the [select 1 2] object receives a 3 or a 0 from the [random 4] object, the [select 1 2] object will send a bang to the [message] object with 0 in, which will mute the instrument.	
Randomise pattern button	The user is given the option to randomise the pattern at any time, without having to rely on the bar count feature to randomise the pattern. A [button] object is used in the interface, which sends a bang to the [p genre] object, travelling through the [patcher] objects via their various [inlet] objects and finally reaching the [random] objects in each instrument [random] object series, shown in *Figure 4*. Using the *Randomise Pattern* [button] object will send a bang straight to the [random] objects, providing the user with a fast and instant pattern randomisation feature.	3, 13, 14
Swing Feature	The user is granted with a Swing feature, allowing the user to adjust the swing of the rhythm. The [p swing] object, shown in *Figure 10,* captures the ticks from the [metro] object, splitting the values in to two parts; the odd numbers and the even numbers, since swing is an effect that applies to the even numbers of the timing value. Using [/] objects to express the formula (60,000 / Tempo = m/s) to calculate the gap between each tick in m/s for each tempo, the [p swing] object calculates an appropriate range of time in m/s for a swing effect to take place in. The [p swing] objects uses a line of [delay] objects to delay the time of each even tick, based upon the value set by the user using a dial and the calculated range for the dial to operate in. By adjusting the value of the Swing [dial] in the device, the [p swing] patcher adds a relevant delay to the even ticks, creating a groovy swing effect for the user to play with.	2, 13

Drums MIDI Out	The user is able to specify where the signal from the drum sequence generator is to be sent. By default, it is sent to the in-built AU DLS Synth, which generates drum sounds from the operating systems MIDI instrument. However, the user may want to send this signal elsewhere, such as to a DAW or hardware interface/instrument. Shown in *Figure 11,* a [midiinfo] object is used at the start of the patch to fill the [umenu] object with information of the MIDI output availabilities. The user is able to use the [umenu] object to select a desired MIDI output. The [umenu] object then sends the selected preference to the [noteout] object of each drum instrument, altering the MIDI destination for the drums as a whole.	9
Mute Kit Parts	As well as using the [p beatcount] object to mute drum parts at selected intervals, the user can also use the [textbutton] object in the interface to mute and unmute each part. By using the signal from the random mute system to toggle the interfaces [textbutton] object, it also provides further visual information for the user, avoiding confusion as the patch mutes and unmutes random parts. These [textbutton] objects, shown in *Figure 12* can be operated by both the random mute patcher object and the user in the user interface. These [textbutton] objects toggle a [gswitch2] object for each instrument, which can open or close the signal flow for each instrument.	13
Randomise individual drum parts	The user is able to add randomisation to each drum part, with the ability to increase and decrease the intensity of the randomisation. The [p randomization] object makes use of the tick counter, received via an inlet and sent to a [random 32] object. Each tick, the [random 32] object generates a random number value between 0 and 31, shown in *Figure 13*. The [dial] object is operated by the user to increase and decrease the intensity of the randomisation. Before being sent to the [p randomization] object, the dials operated by the user are inverted, using a subtraction object (shown as [!- 32] in *Figure 13*), which reverses the inlets by a given value, which in this case is 32. Since the lowest value of the dial is now 32, a [sel 32] object is used shown in *Figure 13* to cut off the randomisation signal flow when the dial is at its lowest, or off. Once the dial is no longer at 32, the signal flow opens, allowing the randomisation to be heard. As the value of the dial increases, the value from the invert object reduces, from 32 to 1. This value is then sent to the Number Range inlet of the [random 32] object. Effectively this means that as the dial increases in value, the range of the [random] object decreases,	2, 3, 13, 20

	which in turn increases the chance of the object sending a bang. The bangs generated by the random objects are sent to the noteout objects, creating notes alongside the notes generated by the sequences earlier in the patch. The user is able to specify this effect to be either post-mute or pre-mute. Shown in *Figure 15,* LED's are used to enable the user to specify pre-mute or post mute, which in turn toggles the [gwitch2] objects visible in *Figure 15* to alternate the signal flow. If the user specifies post-mute, the signal from the randomiser will travel through the same switch as the mute toggles. If the user specifies pre-mute, the signal from the randomiser will travel straight to the noteout objects for each drum instrument.	
Bass, lead, strings, randomised voice, record and playback	The user is able to introduce some synth sounds to the patch, including bass, lead and string sounds by programming the series of [textbutton] objects in the interface to specify how often each instrument pattern changes and mutes, similar to the drum kit. Since the bass, keys and strings are generated in a very similar manner, the system for creating the keys will be looked at primarily. Looking at the signal flow for the keys, after passing through the [p barcount] system, which uses the [textbutton] objects and a [counter] object to count a number of bars specified by the user, the [p keys] object, displayed in *Figure 16*, uses an [mtr] object to record MIDI date and play it back. Since recording with the [mtr] object stops the signal flow, the recording system alternates the signal with [gswitch2] objects, so that when the [mtr] object records, a duplicate of the signal is sent to the noteout system to be played. Once the signal is recorded and playback has begun, the signal is switched from the direct routing to the playback from the [mtr] object. In order to generate pitch, the [metro] object sends its ticks to the [p scale] object in the [p keys] object. The signal is then sent to a generative system in the [p keys] object displayed in *Figure 17*. Each tick bangs the random 20 object, sending a random number value to the [select] object. Although the [select] object has 20 variables, there are only 8 buttons. These buttons generate a number sequence of a pitched scale, with the example in *Figure 17* being a major scale. These numbered [message] objects are linked to more than one outlet, some more than others. This means that some notes have a higher or lesser chance of being played.	2, 3, 4, 5, 9, 24

	The randomly generated number/pitch sequence is then sent to the record system via a [send] object, displayed in *Figure 17*. Much like the pitch generation system, the velocity and duration system, shown in *Figure 18* start with [random] objects receiving bangs from the [metro] object. Each bang generates a random number, which travels through a [select] object to trigger the [message] objects for each parameter. Like the pitch generation system, some messages are more likely to be banged than others. Looking at *Figure 18*, the most likely message to be banged is the value of 0. Since a velocity of 0 is effectively silence, this acts as a musical rest, adding a break to the sequence. The values generated by the pitch, velocity and duration are sent individually to the recording system using [send] objects. When using the AU DLS Synth, the user is able to alternate the voice of each synth using a [pgmout] object. As each instrument is sent out via its own MIDI channel, its voice can be specified individually using the [pgmout] object by sending number messages via [message] objects to it. Whilst the user can specify which voice each instrument uses by operating a custom [umenu] object in the interface, the user can also operate the [textbutton] objects to randomise the voice as well as the pattern for each instrument.	
Button Toggle System	To provide the user with an easy to follow interface, [textbutton] objects were used greatly throughout the patch. However, these buttons work as individual toggles, and struggle to work together as a series. Looking at the button system in *Figure 7*, when each button is toggled, it is followed by [message] objects and a [button] object. In a specific example, if the user toggles the '4 Bars' button, the [textbutton] object will send a signal to the [set 0] object, which will turn all of the other buttons in the series off. The [textbutton] object will also send signal to the [set 1] object, which will turn the origin [textbutton] on, even after being toggled again, preventing the user from turning the button off. The signal from the [textbutton] is also sent to a [button] object, which carries the signal from the chosen button to the appropriate destination.	2, 10

Key Change	The user is able to change the key at random intervals or in an instance using the key change system. Displayed in *Figure 19*, using a [textbutton] system that is familiarly used throughout the patch, a bang is sent for each specified bar to the [p key] object, shown in *Figure 20*. This bang then generates a random number from the [random 127] object, sending it to the [slider] object. From there, the signal from the slider, valued from 0 to 127, is scaled using the [scale] object. Written as [scale 0 127 48 60], the scale object transforms the values 0 – 127 to 48 – 60. This system effectively creates a random number between 48 and 60, which is used as the master key for the patcher. The user may also specify a key using the kslider shown in *Figure 19*. The randomly generated number from the [p key] patcher is sent to the [kslider] object, allowing the [kslider] object to be the definitive value for the key. This avoids confusion when operating the key to be changed and random intervals by also providing a visual display of the current key for the user to observe. To implement the key with the randomly generated pitch of the notes, a plus object is used, shown in *Figure 16* to combine the number value of the selected/generated key with the number value of the generated pitch.	2, 3, 8, 16, 19
Scale Menu	The user is able to specify which scale the randomly generated notes will be played in. The user operates the custom Scale [umenu] object, listing a range of scales to choose from, which sends the list number value to a [p scale] object, which is the patcher object in which each instrument generates its pitch number value. Shown in *Figure 21*, the [umenu] objects list number value alternates the signal flow from the [metro] object to the appropriate collection of objects for each scale. Each collection has a variation of [message] objects which follow the values of their specified scale, meaning that the pitch generated by each collection will follow a specific scale, as specified by the user.	2, 3, 6, 13, 17, 19

Overall Presets Offered	The user is able to recall a range of presets, specified by the [preset] object on the right of the interface. The presets on the [preset] object are already saved, providing the user with some great starting points in operating the patch. The [dial] objects and [umenu] objects have been saved in to the [preset] object by specifying their values and shift-clicking an allocated slot. However, since many of the user interface objects are [textbutton] objects, which cannot be specified by the [preset] object, a range of [sel] objects capture the preset number value, shown in *Figure 22*. For each preset number value, a bang is sent to a selected group of [textbutton] objects to toggle them appropriately. If the user selects preset 1 on the [preset] object, the [preset] object will send a bang to the [sel] objects, but only the [sel 1] object will carry the signal to a [button] object, which will toggle a selected range of [textbutton] objects.	6
Create a suitable user interface	To help the user understand the patch and operate it appropriately, a suitable user interface was implemented in to the patch. This includes labelling each control the user has access to, gathering the controls in to appropriate groups and providing visual feedback for the user to interpret. Shown in *Figure 23*, a range of [panel] objects are used to help group the objects in to appropriate collections, such as drum kits, bar loops and kit instruments. [led] objects are used to provide visual feedback for the noteout activity for each instrument. The values from the [tempo] object are displayed in individual [integer] objects, labelled for the user to observe.	1, 6, 10
Implementation of a Max for Live patch	To further expand the output capabilities of this patch, the drum sequence generator was implemented in to a self-contained Max for Live patch, shown in *Figure 24*. The noteout objects were re-formatted in order for the signal to be implemented with Ableton Live. A range of Max for Live objects, such as [live.thisdevice] and [M4L.api.ObserveTransport] were implemented in to the patch, displayed in *Figure 25*. The user interface was re-formatted to suit the size of the MIDI plug-in window and the functionality of the drum sequence generator.	23

Programmed Constructions

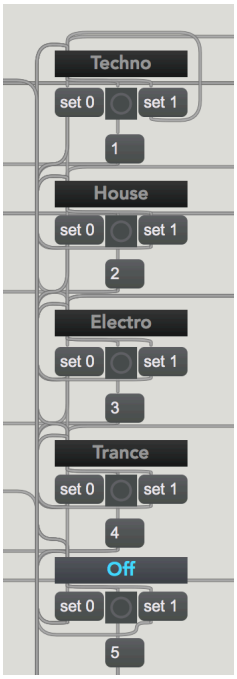

Figure 2 – Genre selection system utilising [textbutton] objects

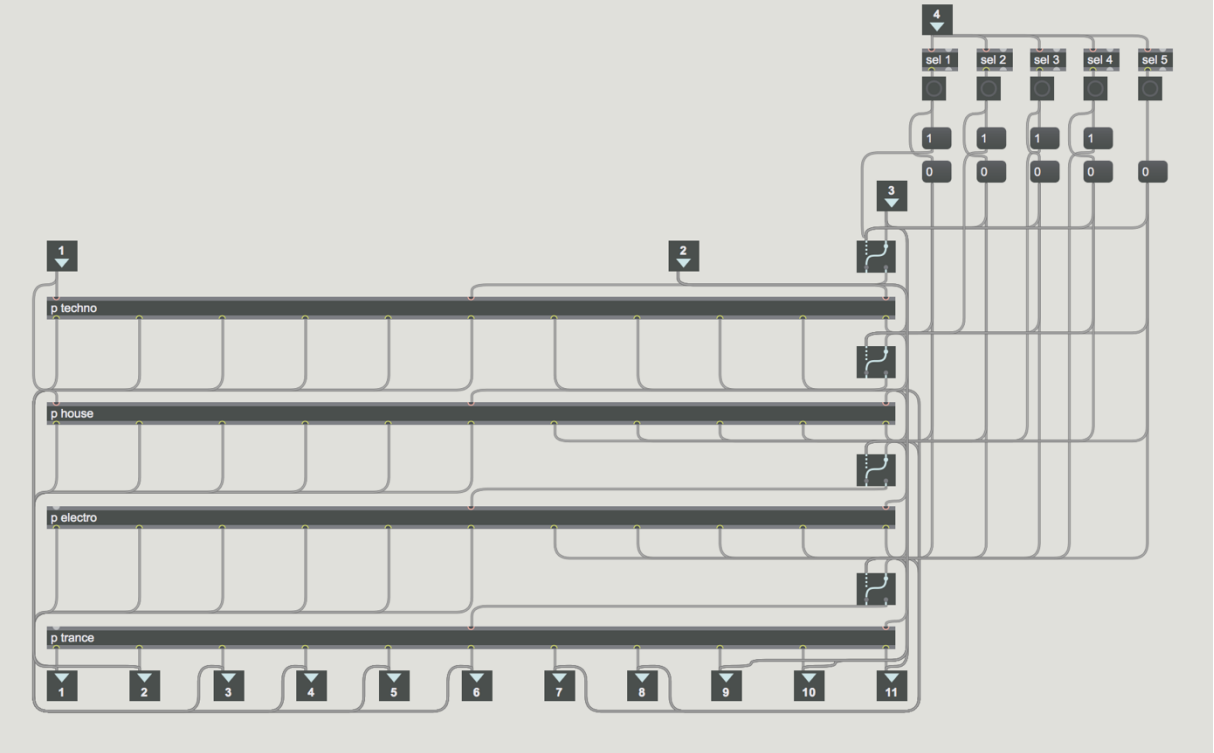

Figure 3 – Inside the genre selection system utilising [patcher] objects

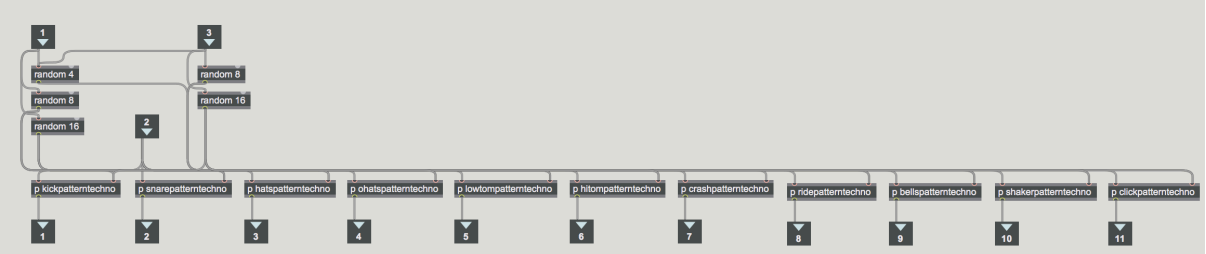

Figure 4 – [patcher] objects holding patterns for each instrument

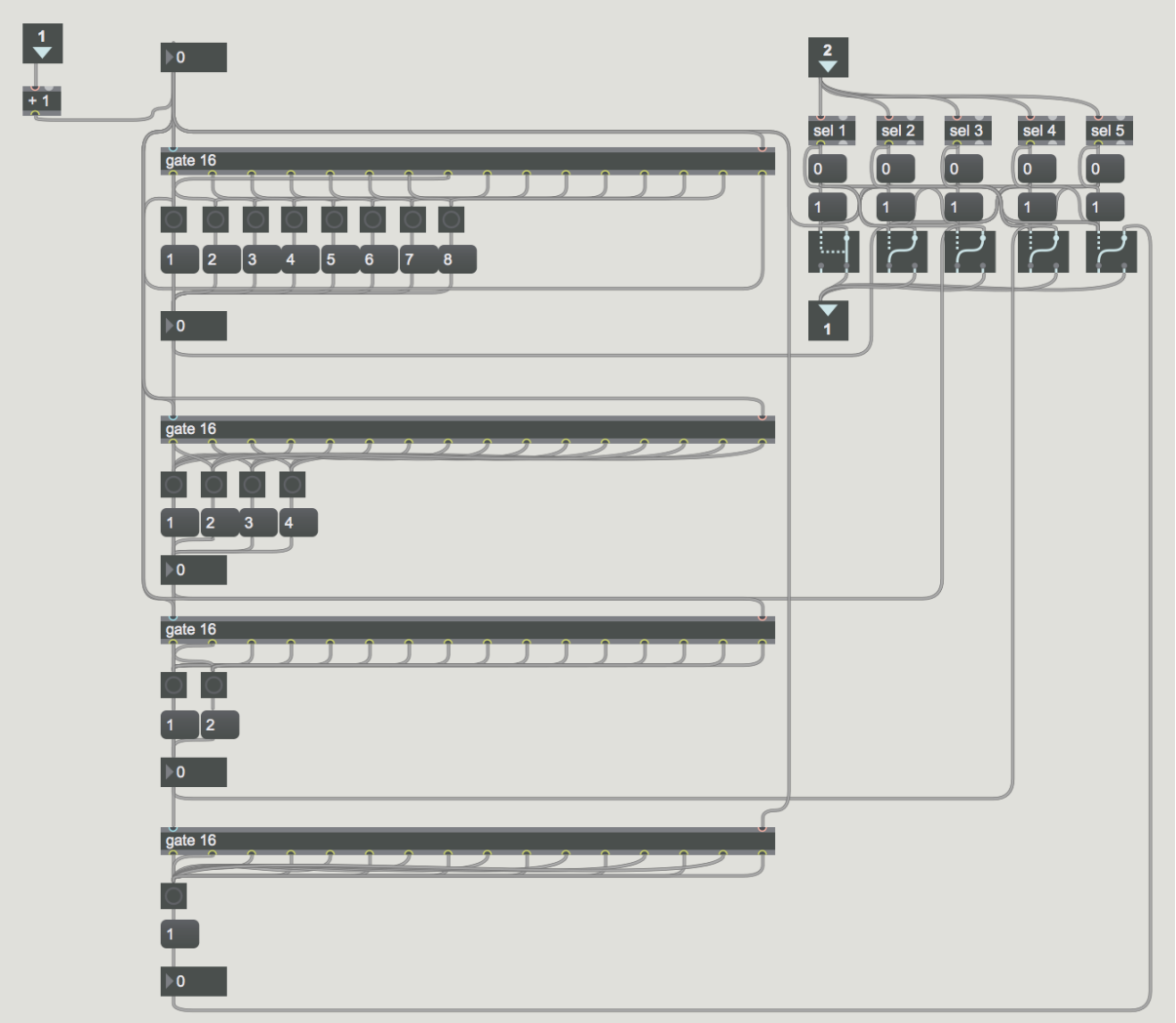

Figure 5 – Roll system utilising [gate] objects

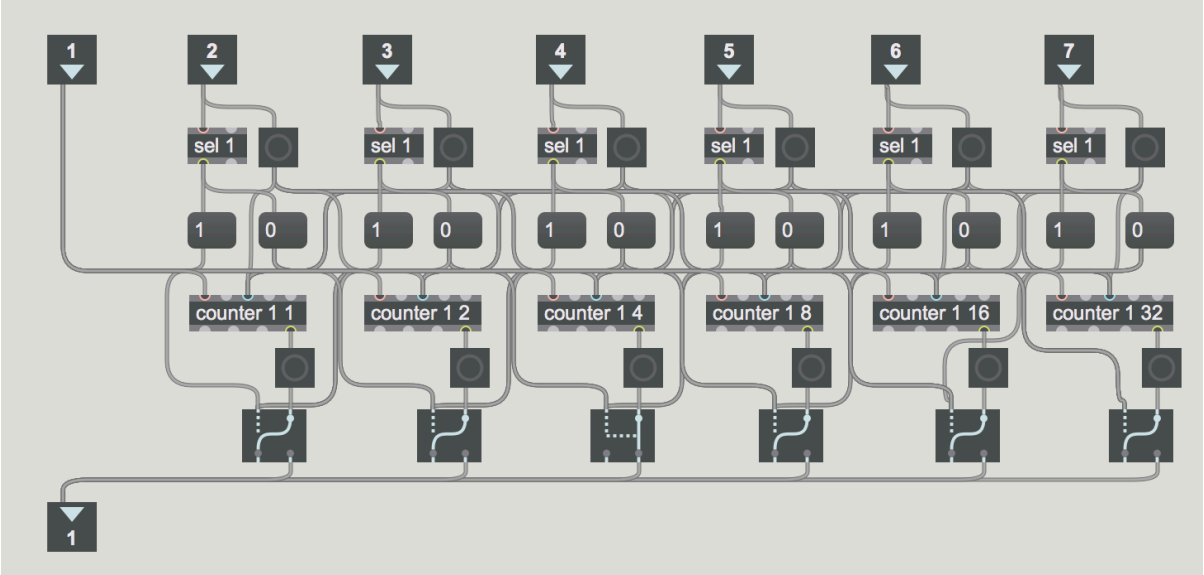

Figure 6 – Pattern and Mute randomiser

Figure 7 – Pattern and mute randomiser interface

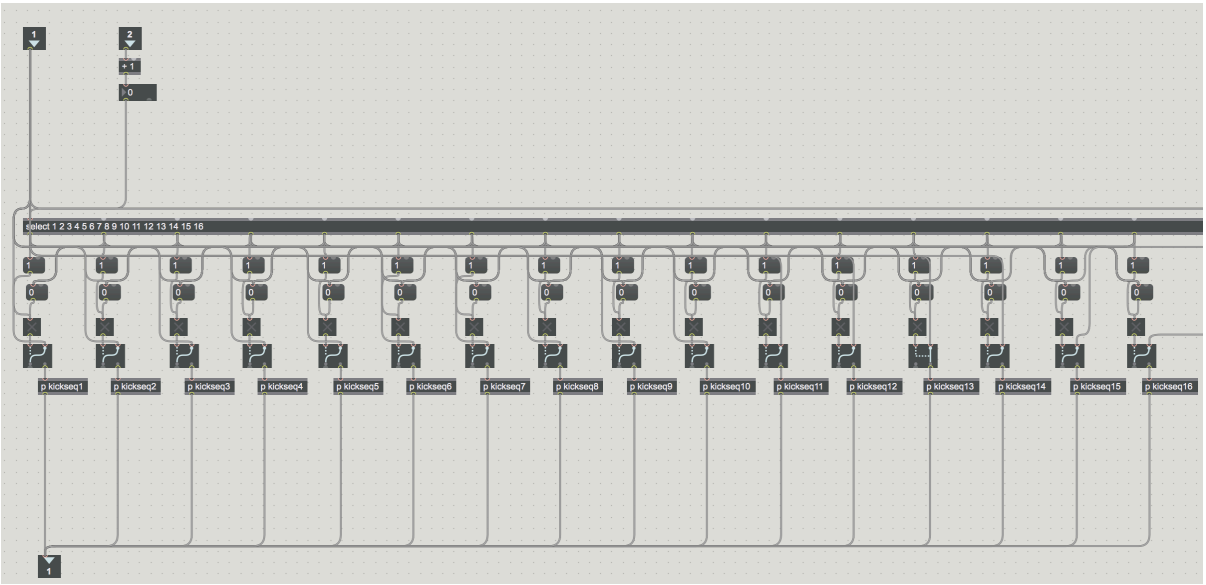

Figure 8 – Sequence selector for the kick drum

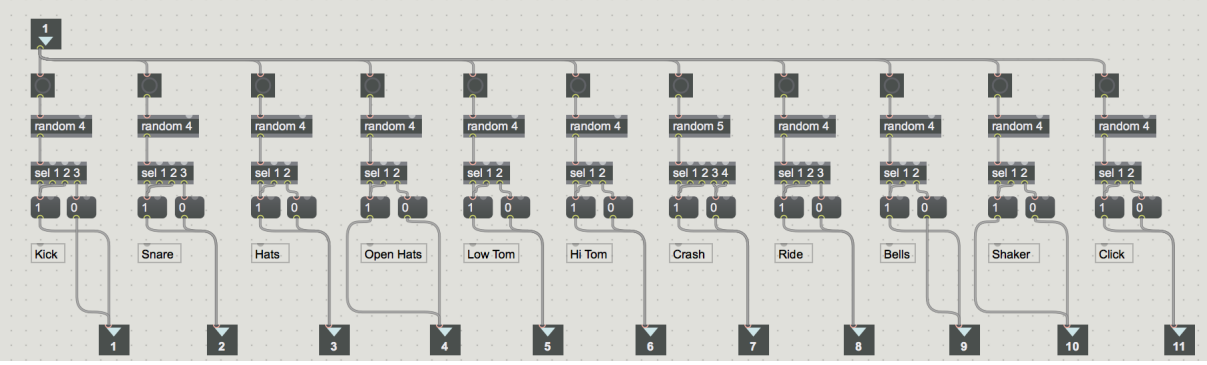

Figure 9 – Random mute system for the drum instruments

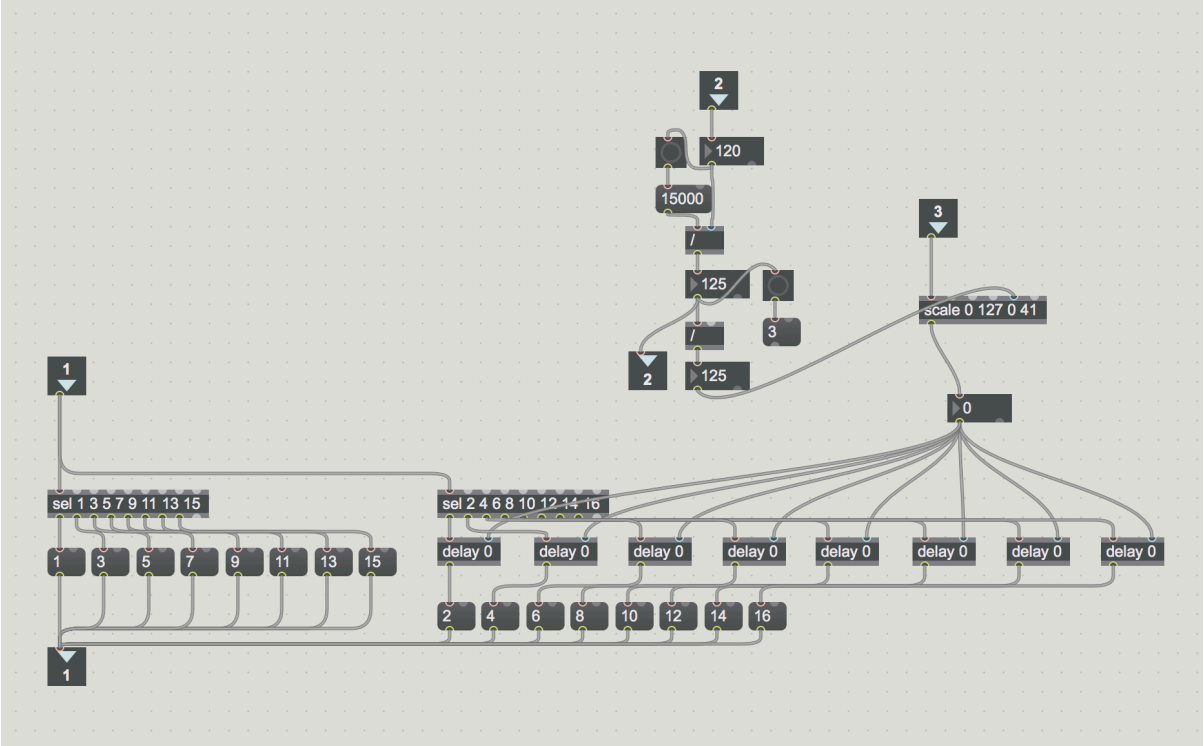

Figure 10 – Swing system, altering the even numbers for each even tick

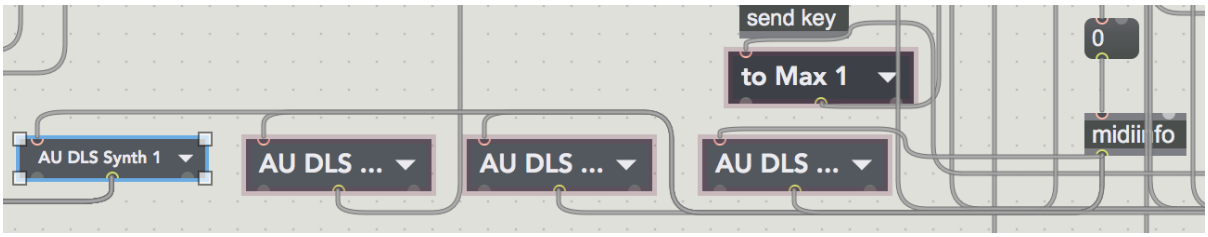

Figure 11 – Umenu objects displaying MIDI output paths

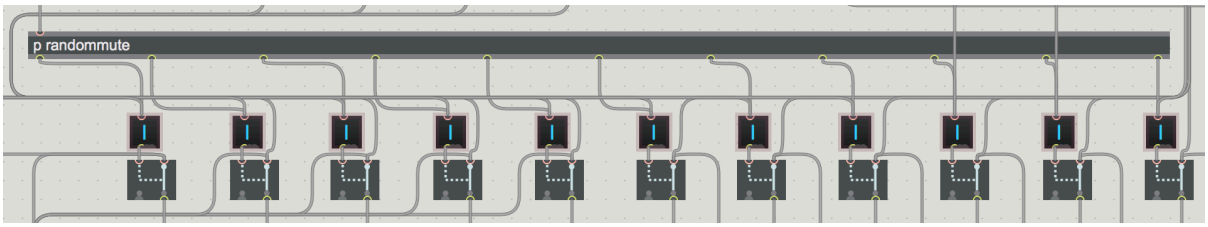

Figure 12 – [Textbutton] objects toggling the [gswitch2] objects for mute function

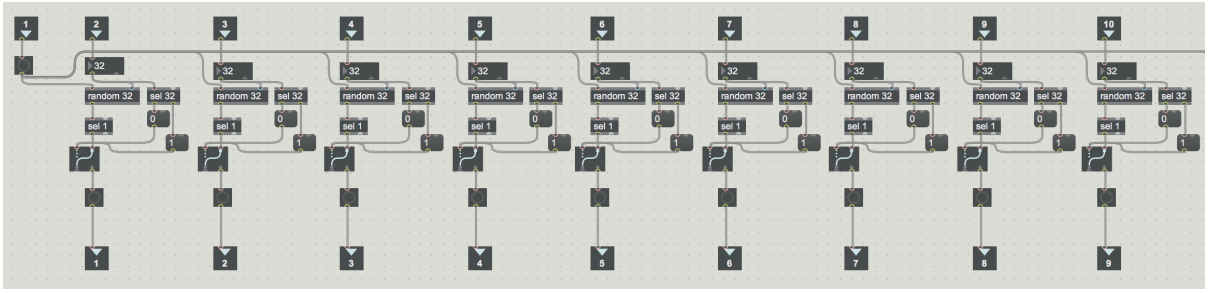

Figure 13 – Randomization scale objects

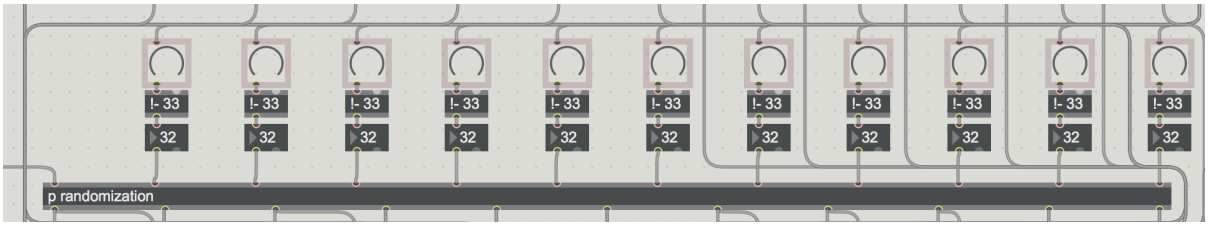

Figure 14 – Inverted randomization dials

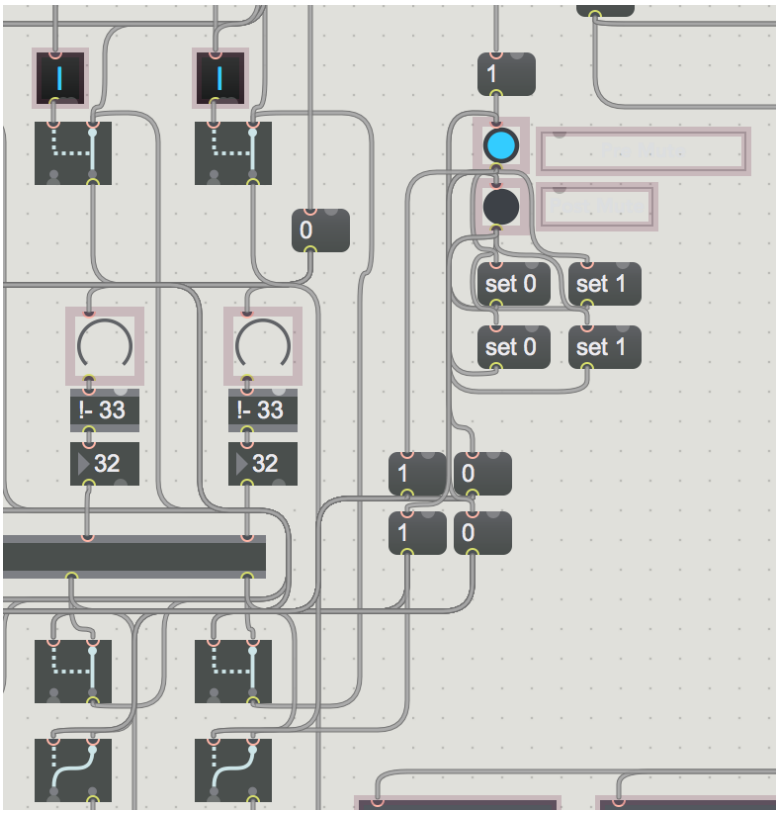

Figure 15 – Randomisation post/pre menu options

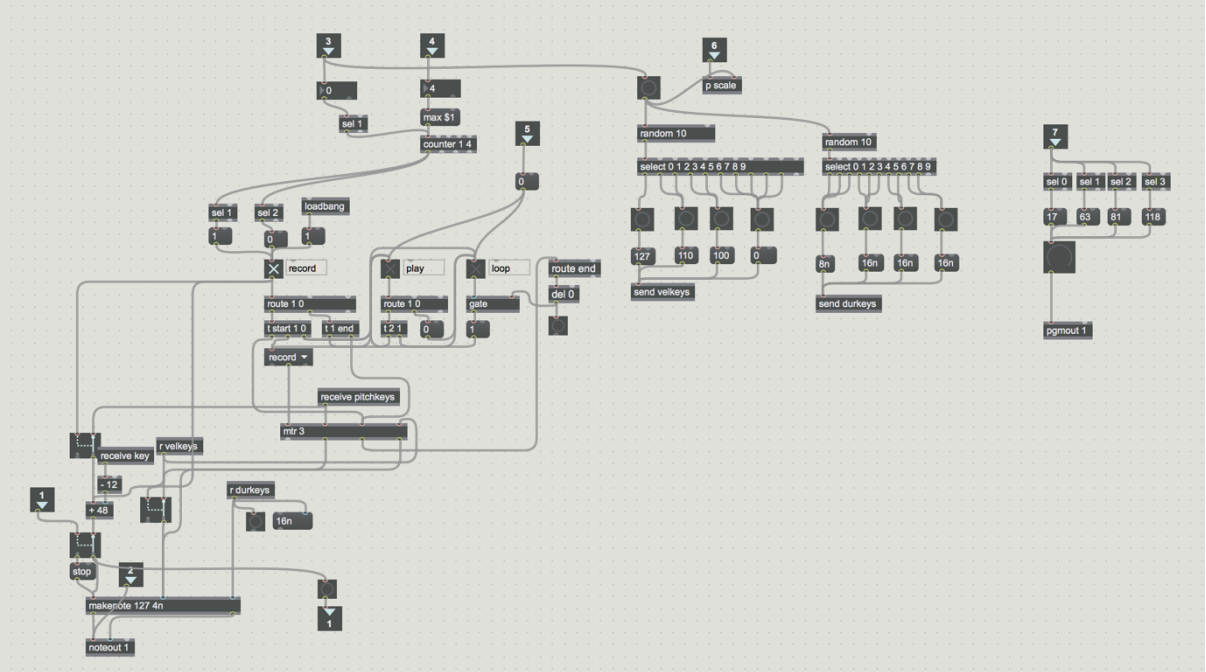

Figure 16 – [p keys] object

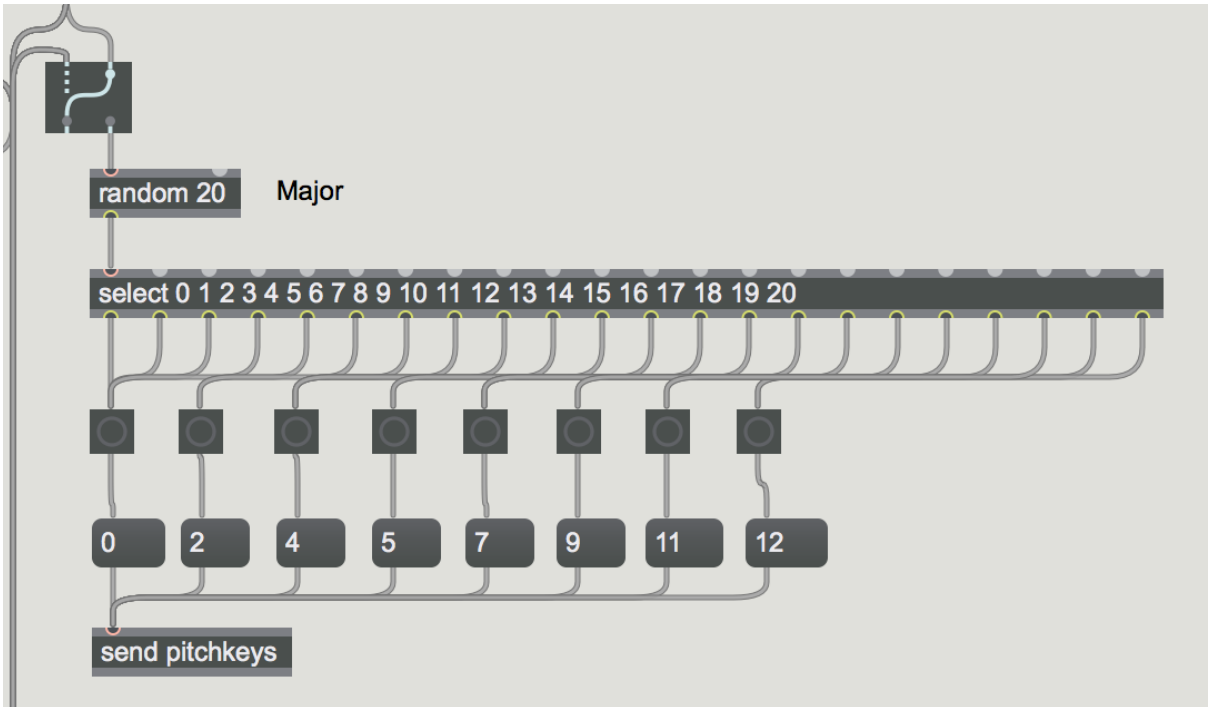

Figure 17 – Pitch randomizer

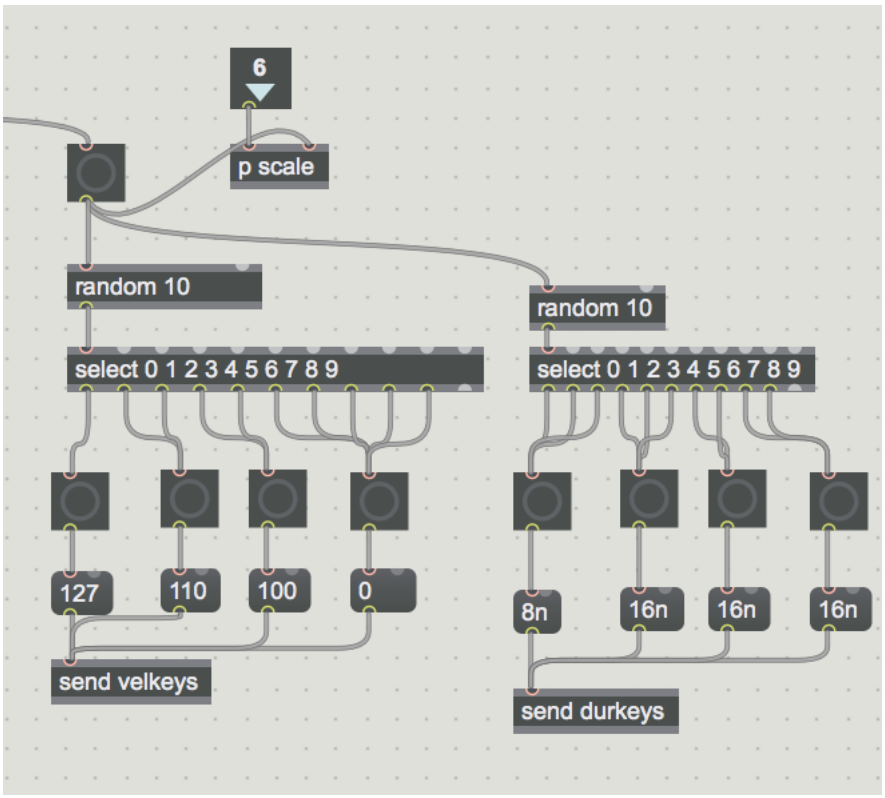

Figure 18 – Duration and velocity randomization

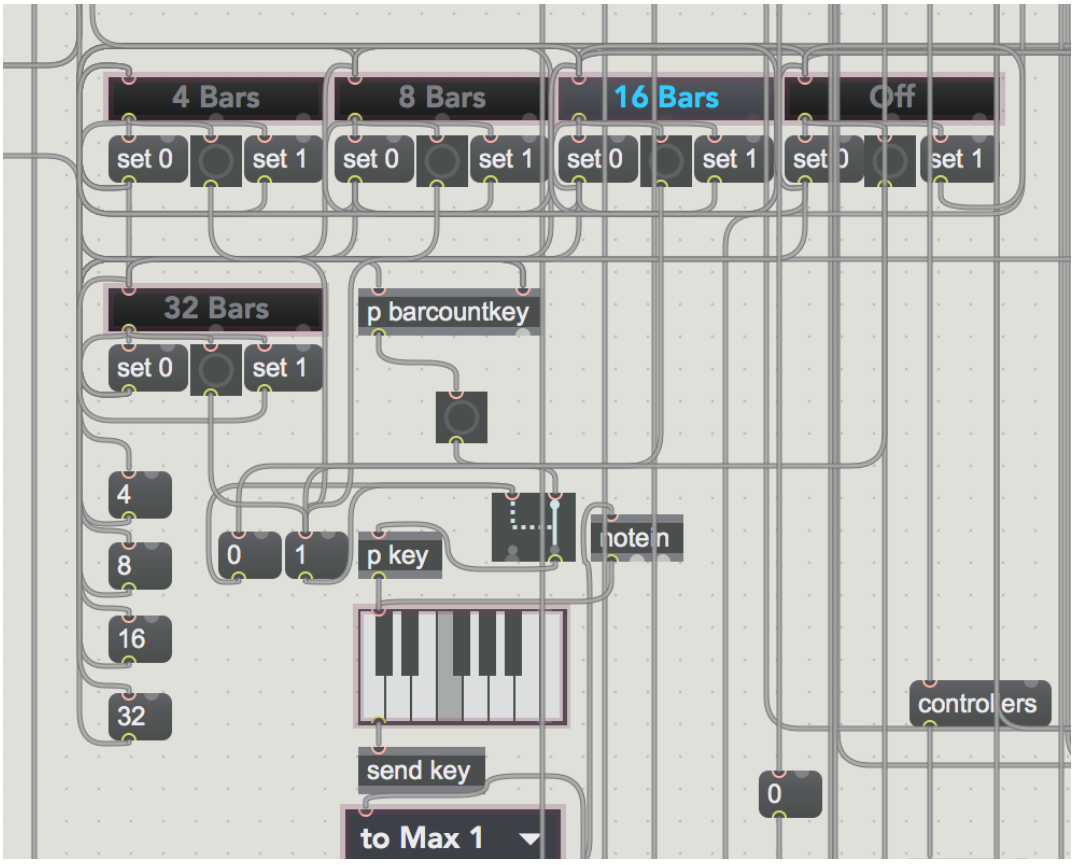

Figure 19 – Key randomization interface

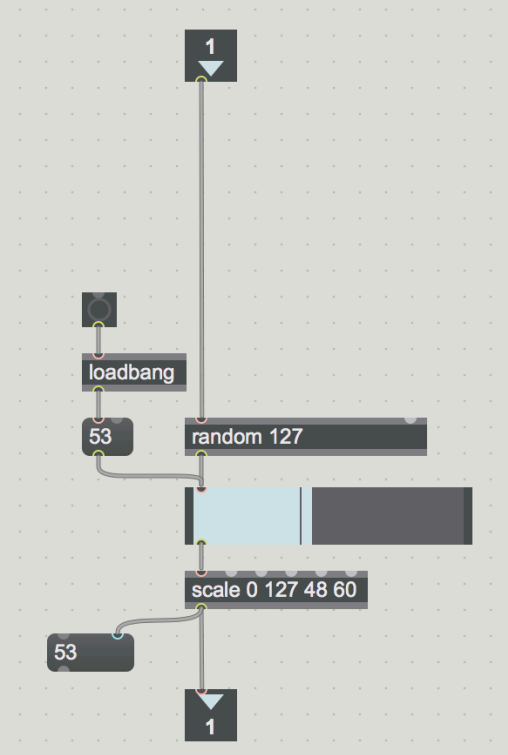

Figure 20 – Random key generator

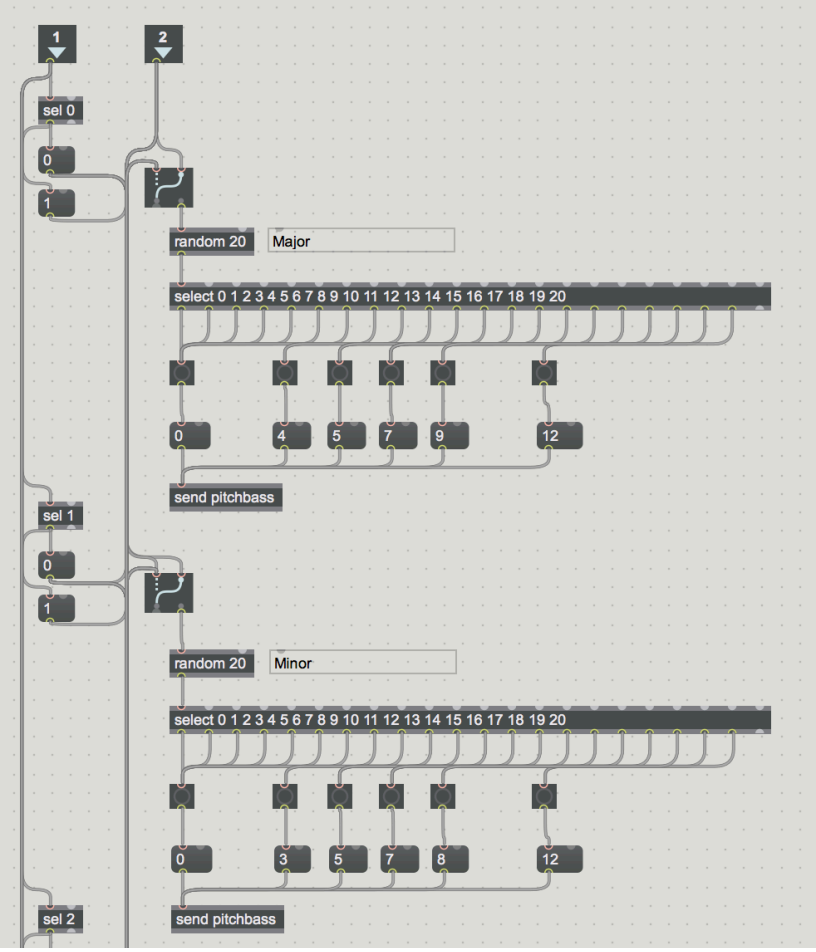

Figure 21 – Scale parameters, showing different scale values

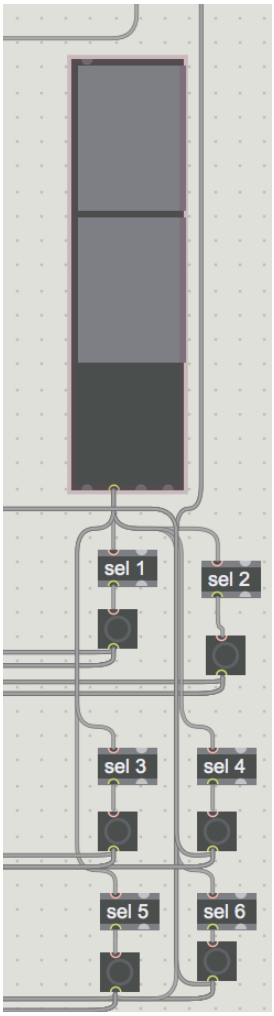

Figure 22 – Preset object and select objects

Figure 23 – User interface

Figure 24 – Ableton Max for Live patch

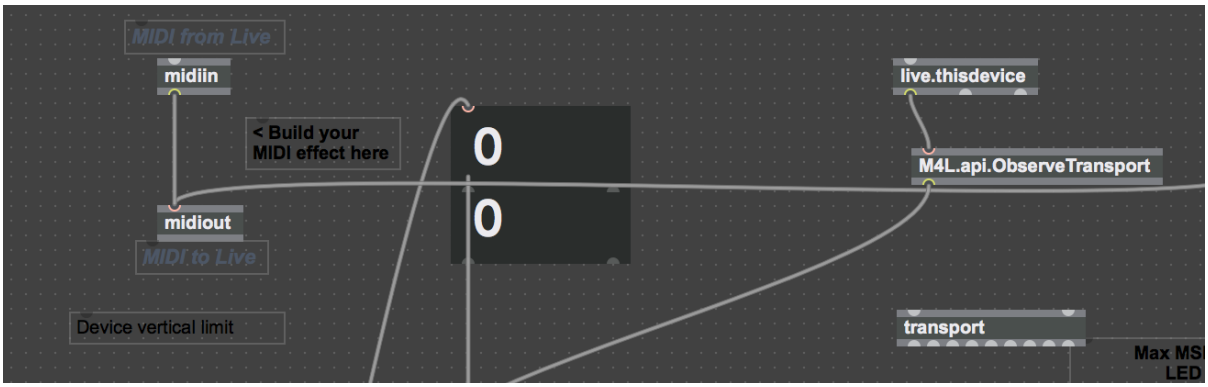

Figure 25 – Max for Live objects