

Alexis Pavone - 1913815 - Development Assignment - Element 2b – Instrument Essay

Introduction

This essay documents the research, development and creation of a digital instrument, aimed to be deployed as a guitar-based signal processing patcher. My intentions are to create a universally compatible VST style plug-in that uses a DI input from a guitar and produces an effects chain in the vein of Native Instruments' Guitar Rig. My motive is to discover if there's a way to create a guitar processor that offers decent standard audio quality and effects for free. This originates from my early experiences of struggling to find such a plugin; having explored and experimented with countless guitar-based plug-ins I am confident that I understand which effects are fundamentally necessary, which would be ideal, and which would be pleasant to incorporate in my model.

I have chosen to use Max/MSP to develop this instrument on the recommendations of experienced technicians and musicians within my professional network, some of whom will participate in user feedback. I believe this would be the most fulfilling learning experience, as I have no underpinning knowledge of Max/MSP, nor Pure Data or Reaktor, but some of FMOD. I am confident that I will be able to complete this task and I look forward to gaining further knowledge and understanding in order to continue growing and developing my skills beyond the completion of this stage.

Part one – Research

I began by studying audio processor patcher tutorials and cross referencing the theory of those models using books such as Microsound by Curtis Roads, Musimathics by Gareth Loy, and Sound Synthesis and Sampling by Martin Russ. I quickly identified the methodologies that are relevant to which operations, as the signal flow is an indicator. Once this was understood, I copied the patchers and experimented with the components by substituting key elements with a variety of alternatives. After discovering line-in, signal flow and MIDI objects, as well as how MIDI signals are turned into numerical values, I explored more complex objects and the means to control them.

Looking at a variety of message types I experimented with unpack, print and menus, learned about cycle~, metro triggered signals, and transposition/adding degrees. Many of these objects, while very basic, perform fundamental operations, and form the infrastructure of my instrument. I experimented with cycle~ and adsr~ values, frequency and modulation, and switches and gates. Using a notein object I compiled numerous patchers to understand how to generate and shape waves, explore additive and subtractive synthesis, and visualise the inner workings of an LFO. I discovered stealing patchers and began incorporating examples from Max help files into my own work; doing this I learned the principles behind them, but primarily how to combine different elements.

When examining the overview of my instrument, I questioned which elements to use and how to incorporate them, or whether to include them at all. Working with umenu and sel objects I encountered looping with buffer~ and reshaped the concept of my instrument. I redirected my studies to explore the possibilities of creating an instrument that runs parallel to the guitar processor. I mapped out a design for such an instrument and discussed its tangibility with musical colleagues. User feedback confirmed that although the concept behind it is achievable, it is very big for a novice and needs rescaling to a considerably limited device. On that basis I chose to simplify it and initiated researching and utilising the input methods retune~ and fzero~, and their use with effects typical of pedal boards.

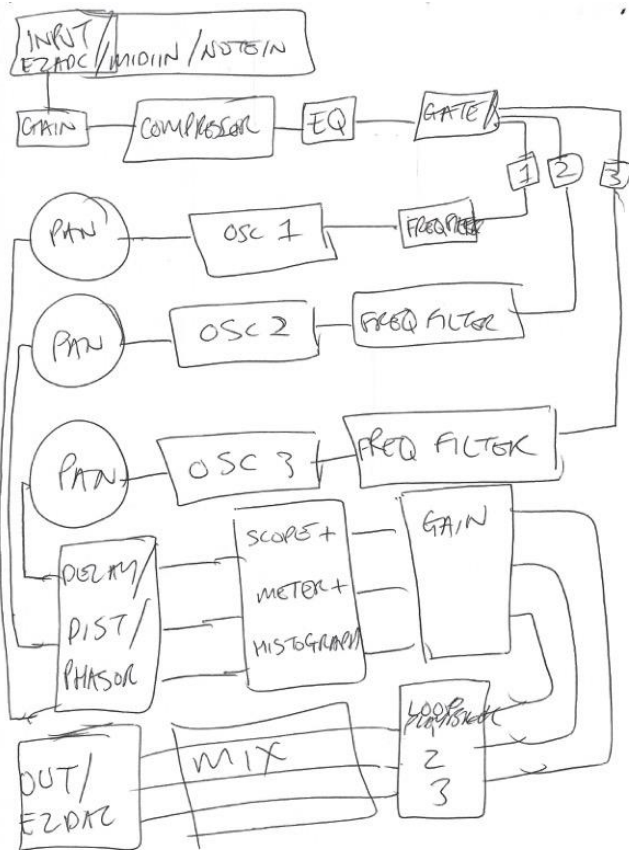


Fig. 1 First draft of instrument.

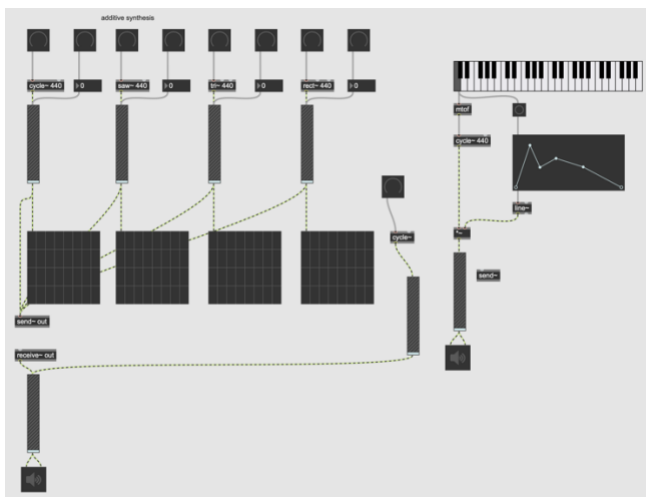


Fig. 2 Basic patchers made using Max help files.

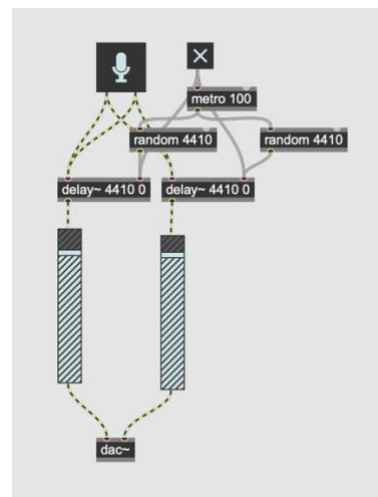


Fig. 3 Dual randomisation of input signal.

To demonstrate the effectiveness and efficiency of Max help files these patchers were constructed following the object and argument notes in the inspector. To aid workflow and housekeeping I initiated working from a constantly growing template containing objects and subpatchers. I discovered that as the work progressed this method led to cluttered files, which I will combat in future projects using outside suggestions such as compiling patchers into projects and encapsulating. The effect achieved with Fig. 3 is dubbed ‘fake stereo’. This imitation spread was intended as a makeshift production tool for simple music production patchers, but to the discerning ear it is clearly a double flanger effect. As with the other effects, for convenience I kept it in my collection of tools until the later discovery of the `cverb~` which performed immeasurably better.

I continued to replace elements with the developed better alternatives until I reached the feedforward stage, at which point they were peer reviewed - discussions were then held about functionality, efficiency, effectiveness and performance. I felt I needed more extensive research to innovate most effectively, so I then spent some time using Max/MSP, unlocking and studying existing patchers and help files. This caused some frustration until I acquired 'Electronic Music and Sound Design' by Alessandro Cipriani and Maurizio Gigi. This further research enabled me to incorporate theories from various resources, including: (Loy, 2006) 'The frequencies of tones that make up an interval can be compared by making the ratio of the frequencies' (p15), and 'intervals are added by multiplying the ratios and subtracted by dividing the ratios' (p93). In Max/MSP this translates to transforming one value into another by means of scaling, formula objects (*, /, and %), and writing logic. This means that using these computational tools the concepts of music theory can be applied in the form of numbers and calculations.

My background in music teaching influenced my early approach to this task, and while I didn't want to create an educational tool I felt the need to incorporate music theory or relationship/ratio-based material. In Chapter 1 Cipriani and Gigi (2019) document the fundamentals of logarithmic function, synthetic sound and frequencies, and the frequency/musical interval relationship, most of which formed the concept behind my instrument and elements I would begin to incorporate. Several variations of the instrument were made before I fully grasped the concept of signal flow. The biggest challenge was understanding which units and values each component was capable of sending and receiving.

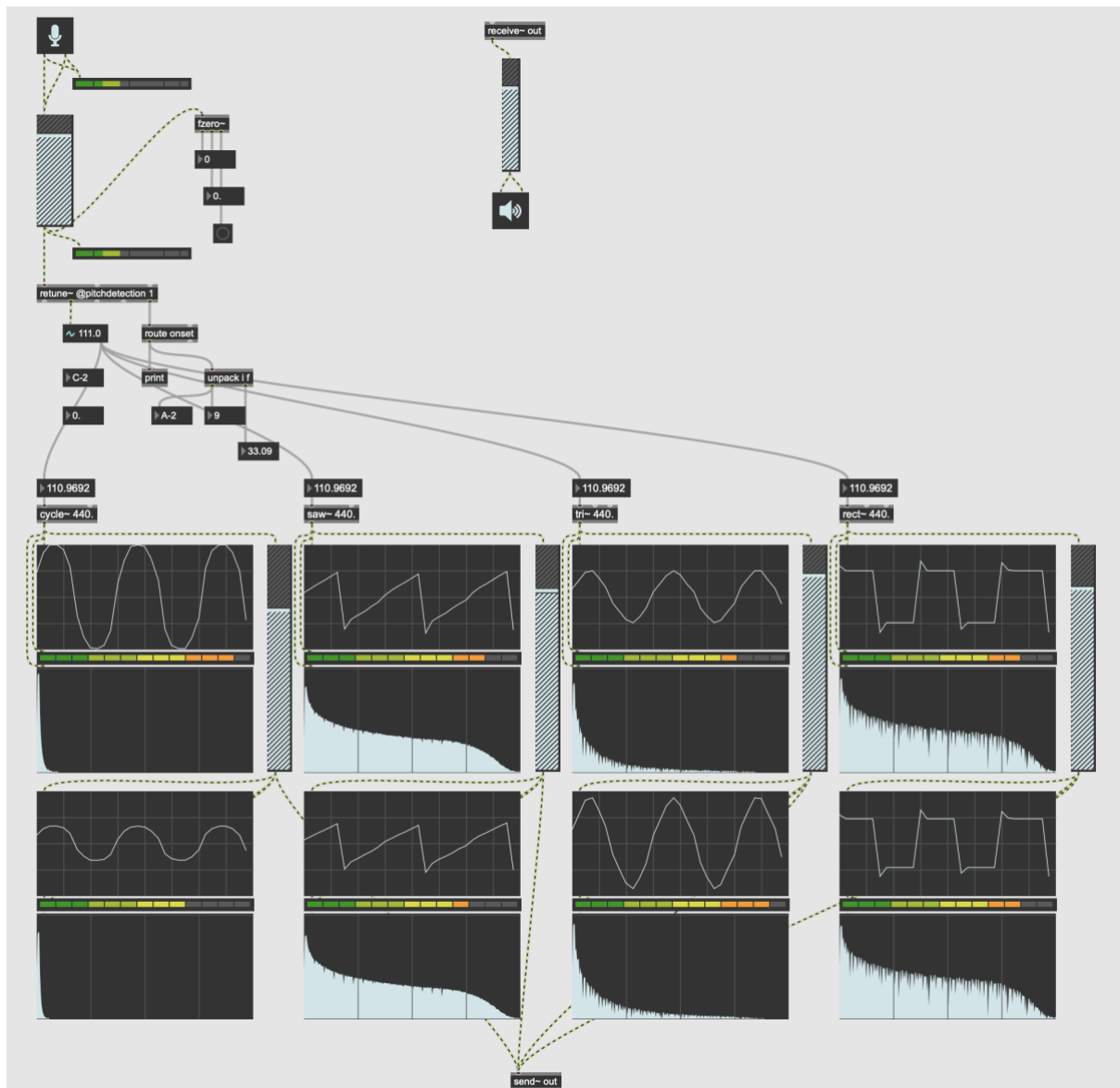


Fig. 4 Instrument version 1.

I used many different tutorials on several versions of Max/MSP to build a solid knowledge base, which gave me the foundations of principles and methodologies required to improve structuring and routing. I had reached the first stage of my task, a 4-oscillator instrument with an input splitting to the compressor. Thorough testing revealed issues with `retune~` and `fzero~`; a frequency number displayed and confirmed that the `retune~` and `fzero~` out signal was not steady but functioned. Although I attempted various reconfigurations, unfortunately this remained unsteady.

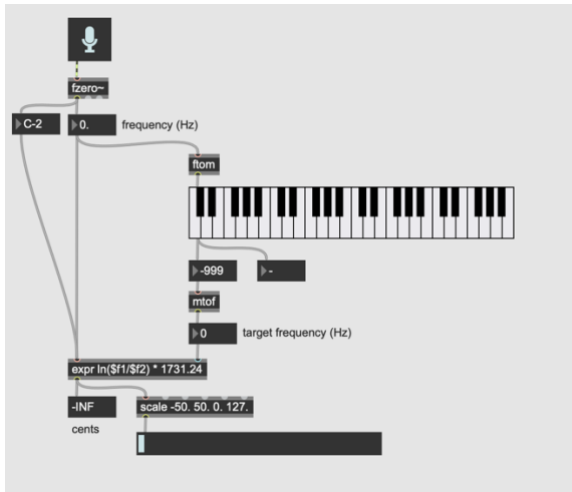


Fig. 5 Example of patcher using logic and scaling, one of many reconstructed in studies.

Part 2 – Making the Guitar Processor

I approached this task sequentially. Grosse (2015) suggests using a good quality interface and avoiding acoustic guitars with pickups, due to feedback and audio quality issues that can arise. A preamp also offers a steady boost and frees up the machine’s power for later computations and routing (Benson and Hart, 2020); my DI is the reliable Focusrite 2i2 audio interface, complete with all necessary inputs, headphone and speaker outputs, and built-in gain.

Shaping and obtaining the desired signal depends on the structure of the system and particularly how it is operated. Programmes like Guitar Rig contain hundreds of elements performing millions of computations per minute; its nature displays the years of specialist research and development to produce it, and the calculations required for obtaining just one effect requires months of coding and engineering. The solution is to create a limited but manageable unit - so using an old guitar pedal I mapped my guitar processor as an all-in-one tool.



Fig. 5 The BOSS ME-70 guitar effects pedal.

Input – Gain – Compressor – Distortion – EQ – Modulation – Delay – Reverb – Output

Fig. 6 The first draft of guitar processor.

The input simply ran an `ezadc~` into the envelope; many examples of patches demonstrated the ease of getting an audio signal in for a quick and easy build. I kept this approach for the duration of this part of the build and used little more than a `live.gain~` to effectively control the amplitude.

The envelope, while not a typical guitar process, is particularly effective with acoustic and percussive instruments that have short and dynamic attacks (Dodge and Jerse, 1997). This degree of control should be enough to eliminate the need for a noise gate to hide guitar hum; however, the compressor offers an attack and threshold parameter control that can tighten the signal further. This simple model performs effectively and delivers a decent signal, despite the guitar's condition. Fortunately, as the signal does not pass through any processor yet, it is controllable and steady. It doesn't appear to need limiting so I believe the preamp is producing a steady signal. My reason to create one is simply that I use it in my DAW as practice.

The compressor proved problematic. Routing its components and assigning correct values and properties was difficult and resulted in failure. Darwin Grosse (2008) discusses signal processing and his choice of compressor; he declared lacking full understanding regarding values and internal intricacies. His model is simple but practical, and suitable for my instrument build - to comply with the Guitar Rig criteria, I have adopted his compressor. I chose to do so because he 'kooped' (Grosse, 2008) it from a Max/MSP help file and after careful analysis and testing they both performed adequately. Calculating and plotting attenuation curves seemed overly complex for the desired outcome of this project, so I altered the values to imitate those of Logic Pro's Vintage Opto, demonstrating how to steal and modify a patcher. Katz (2015) recommends excluding gain reduction meters because they are slow to respond and can overestimate real reduction, just as peak-sensitive GR meters can produce 'a falsely high indication' (p84). The relevance of this relates to how the pitch detector in Max/MSP is coded; this concern encourages me to rely on hearing and perceived loudness during operation.

I opted to avoid multiband compression; it controls numerous individual bands but can also produce level peaking or disrupted harmonics and be unnecessarily complicated (Robjohns and White, 2002). Instead frequency bands are managed by the parametric EQ. Options of 1, 3, 5 or 10 bands are available simultaneously so while less is more, additive or subtractive equalisation is highly tailorable as demonstrated in my model. EQ band messages into `filtergraph~` inlets are a simple but limited option; no two guitars produce identical voltages or frequencies, so this is a comparative disadvantage. EQ inspired the idea of a parallel instrument whereby the line-in could split into a guitar processor and an oscillator with separate parameters and features, manipulating the mix of fundamentals and partials. At every stage EQ is highly controllable, and `in/dry/wet/out` meters enables easy and effective monitoring, further reducing the need for additional limiting on each process. While the dry signal is a path directly to the `ezdac~`, the wet signal path is recursive throughout. I encountered several dilemmas when deciding which modulation effects to use - distortion or overdrive, tremolo or vibrato, flanger or phaser? The final decisions were based predominantly on textbook based methodology and numerical values.

The difference between overdrive and distortion seems trivial; the smaller signal-separation of overdrive more closely mimics true-amp distortion, while distortion relies on aggressive clipping (Dawsons, 2020). The timbral effect of distortion on monophonic instruments brightens the sound, highlighting select frequencies, while on polyphonic instruments can cause discordant frequencies or noise to fill the mix (Russ, 2019). Combining overdrive with the up-to-10-band equaliser minimises the problem, however any synthesiser elements require fine tuning. Using the series

patching method by Grosse (2008) two overdrive~ objects, controlled by slider values, produce the range of tones bridging the gap between overdrive and distortion. This demonstrates an increasing metal effect can be achieved and controlled in one stage, thus removing the need for a second effect process. I encountered a problem with peak loudness - the signal had been boosted so much that the overdrive was screaming, causing painful listening. This was resolved by using monitoring output messages and using specific scale arguments.

Modulation introduced complications. Musical genres and musicians argue different tools for different tasks; personally, I have no preference regarding modular pedal board setups. After studying user builds and analysing their frequency modulation patterns, I wanted to incorporate a multi-wave LFO to expand its capabilities. Using a series of selectors and multipliers the wet line of the signal is manipulated into variable rates and depths of sine, saw, triangle and square waves. Set at an initial reference point of 50% power level for greater accuracy (Oscium, 2020), additional duty controls bridge the gaps between triangle-to-sawtooth and square-to-rectangle waves. The combination of the dial and message values make it easy to reach exact musical intervals but avoid certain partials; this was resolved by changing the step size in the inspector. This is a rudimentary function but principally similar to the BOSS ME-70, and a parameter I value.

Karplus-Strong synthesis principles also have post-production applications. In MIDI, attenuating a continuous loop parallel to the original signal results in a 'physical model of a plucked string' (Loy, 2011, p434) or similarly the delay effect. Using tapin~, tapout~ and a sub 1 multiplier to alter the decay constant I can adjust a note's tail and blend chords or multiple sound waves. When sent MIDI signals, it is a higher quality alternative to using a delay~, which I will use because its value can be matched to the sample rate for higher accuracy. The delay line introduced numeric complications, but it shares the I/O controls of the previous elements and is more sophisticated and reliable overall. First the D/W requires either an expression (expr 1. - \$f1) or subtraction object (!- 1.) to produce a numerical value with which to multiply (*~ 0.) the input to produce the dry signal. The input signal is then multiplied (*~ 0.) by the delay slider values multiplied (*) by the rate (currently 44100) and then multiplied (*~ 0.) by the D/W slider values. This model functions on the sample rate of 44100 per second but the depth slider now indicates a max output value of 396900, so changing the delay~ value to this number gives an accurate sample capture rate and delay playback (bfuhrman, 2002). This difference can't be heard, but when combined with the buffer's sample rate the recordings captured are superior and mathematically precise.

The reverb is basic in construction as it does not factor ER/LR but as per Newton's laws of motion, the signal travels straight at a constant speed until it is reflected by a surface (Loy, 2006). It also doesn't reflect much timbre nor shape 'the boundaries of the environment' (Russ, 2019, p441). Timbres contain data we can (typically) identify and that is the basis of reverb design; I did not manage to artificially reproduce these calculations but gave some thought as to how this might be achieved, perhaps an fft~ or zl.filter with a preset matrix to access the compiled data. In turn, I had planned to include the vst~ to enable plugin hosting specifically for the reverb because it is a controversial tool for the discerning musician - it would be interesting to patch in one's own favourite effects to personalise the musical experience to one's own preferences.

User feedback overwhelmingly stated that the reverb is not a tool that can be compromised with, it is generally agreed that some tools are essential, while others are a matter of personal preference. An option for future models could include variable reverb or tonality with preset functionality, making it closer in resemblance to the aforementioned Guitar Rig.

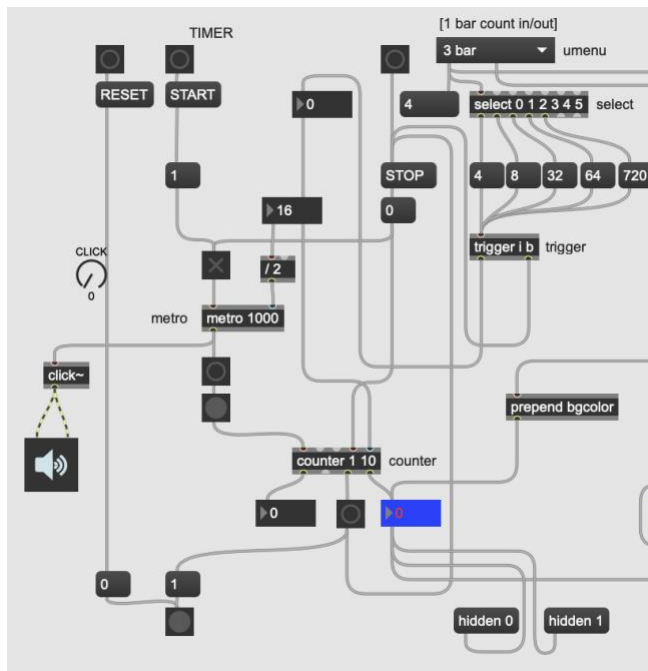


Fig. 7 My improvisation on ‘Timer’ by Reed (2018).

Reed (2018) demonstrates the simplicity of the metronome in a sophisticated patcher. Originally my modifications included additional time values synonymous with song structure, but this seemed excessive. I have created a timer that I linked to a buffer, and a metronome which simultaneously triggers the recording with the start/stop message and bang button. The buffer functions include write, open and replace, to enable filesharing and exporting wav files to other devices and DAWs. This device sits at the end of my instrument and captures the entire production process; a series of tests indicate full functionality and clean signal recording. After reaching the end of this process I revisited the closed envelope an attempt to solve the problem. Following further research, with extensive trial and errors, one solution was to use logic.



Fig. 8 The new envelope logic.

This logic uses an adjustable peak amplitude to trigger the envelope to open. However, this both solved and created a problem. With the envelope now opening, setting up can be longer and needs precise adjustments to avoid making a pulsating/clicking effects.

Part 3 – Making the Synthesiser

The task of controlling an oscillator with the guitar combines FM, AS and SS, which brought to light some important issues. Despite having reverse engineered and reconstructed several versions of the instrument, I concluded that simple is effective but I wanted to develop this further. Using the +~ to increase neighbouring oscillator values I discovered varying values within encapsulated

patchers; this streamlined the model I constructed and facilitated housekeeping. Like the LFO effect stage, this instrument is based on frequency modulation of the input, so the cycle objects became the modulators. The second oscillator introduces harmony and potentially phase cancellation, however probable, even with a range limited to 1024Hz (approximate pitch of snare). I chose this because it can produce decipherable oscillations at an abundance of low-mid frequencies; unfortunately, after some testing with the variety of effects on both sides of the instrument, I discovered that the attacks, sustain, tones and residual guitar hum resulted in some unpleasant noises that are uncharacteristic of actual musical tones. When tested through built-in and desktop speakers it caused the cones to buzz wildly under sudden and extended playing conditions. The solution was a lengthy reordering of the maximum output values of each component to a safe peak.

Adding the third oscillator introduces another element of music theory- if two notes can produce the equivalent of a bar chord, then a third produces triads. This lends itself to a wide variety of western music intonation, as major and minor chords can now be produced. The fourth and fifth oscillator changes things further - the synthesiser is now capable of producing, for example, minor sevenths and major ninths, octave with triads and doubling octaves. It is also worth noting here that with that much of the wavetable and frequency spectrum open, the potential is vast.

To fully appreciate John Chowning's FM model I had to develop input methods within Max/MSP. My initial plan was to convert a guitar signal in order to manipulate a synthesiser with float numbers. There are two techniques, both of which were tested thoroughly; `retune~` and `fzero~`.

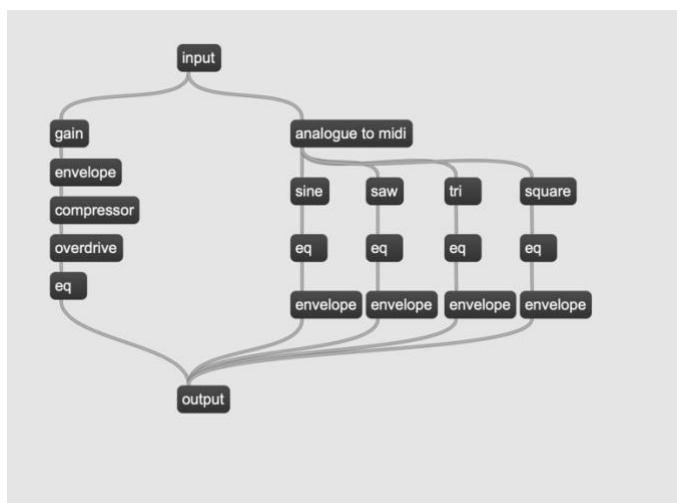


Fig. 8 A comprehensive layout of the desired patcher.

The `retune~` identifies and outputs detected frequency and closest note but not polyphonic audio. During trials the guitar did not hold notes and the top E string at the 12th fret deviated by approximately a semitone. Loy (2006) links this fault to recursion in 'the rule of 18'. The guitar was suitable so the alternative was to alter playing techniques and compensate for the flat pitch. Dodge and Jerse (1997) state 'at least one period of the waveform must elapse before the pitch detector can report a value (p414); the delay of typically 20-50ms was highly amplified by the tonal deviations, causing time and pitch discrepancies. The flickering pitches produced crackling between notes at a rate approximately between 30Hz-110Hz.

When passed through the instrument's effects chain it caused the envelope sustain to buzz, the compressor meter needle to vibrate, and the overdrive to feedback. Furthermore `retune~` could not detect the envelope or amplitude thresholds resulting in abrupt truncations. The similar 'estimate fundamental pitch' function of `fzero~` also failed; it produced vocoder-like effects and created amplitude peaks. I concluded that these methods were not improvements on the `ezadc~` and opted to continue using this instead.

Russ (2019) states ‘FM really only requires the following parameters to define a timbre: the carrier frequency, the modulator frequency, modulator level’ (p.263). Returning to the LFO model, I attempted combining modulation and parameters. By routing oscillators in series I experimented with the capabilities of additive synthesis, modulating FM with RM, RM with AM and AM with FM, and unipolar with bipolar frequencies. This worked in theory, and some combinations worked with amplitude limitations in place, but predictably more sensitive exponent, step and range boundaries were needed to increase control and performance.

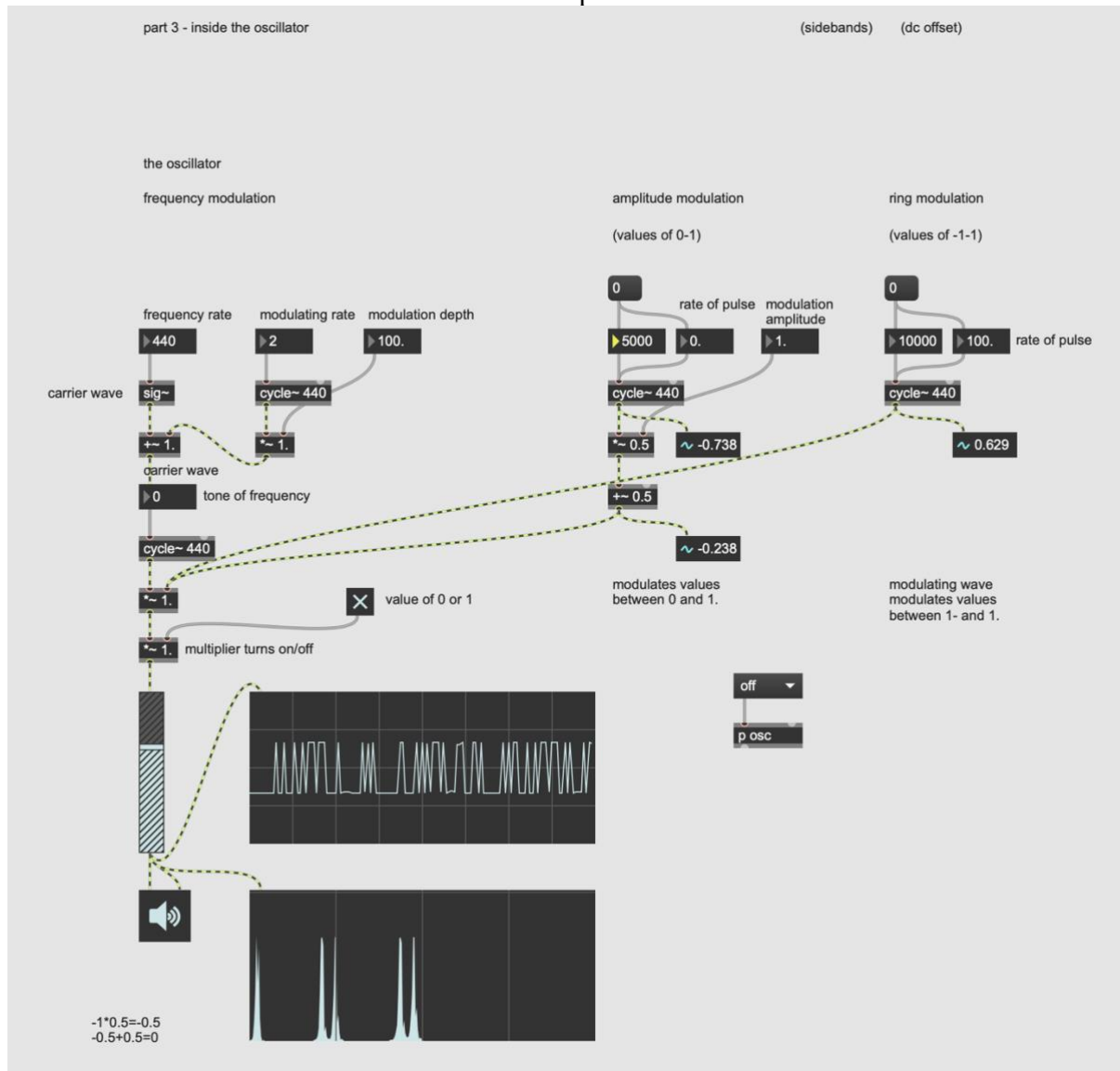


Fig. 9 The core of my additive synthesiser.

My idea behind mixing five sound waves came from an analysis of a dynamic spectrum produced by John Chowning’s FM design (Dodge and Jerse, 1997, p123), combining multiple waveforms to imitate acoustic instruments. Loy (2011) explains how dynamic elements such as ‘vibrato, tremolo, glissando, crescendo, and decrescendo’ (p33) make up an instrument’s spectrum. If we create an instrument that produces only fundamentals and fixed-value waves we limit musical capabilities; if it produces partials and waveforms with varying duty values, we enable the potential of microtonal and binaural audio and vocoding. Ultimately, I didn’t assign oscillators to bandwidths or wave forms, so just as octave doubling can be produced, so can timbre layering. Theoretically, the

instrument could be capable of phase cancellation and even distortion if the parametric conditions are met. In future models I might combine unipolar and bipolar FM and variable scaling to further reduce the possibility of phase cancellation.

These combined effects could be interesting when paired with panning, and perhaps even an `fft~` to command certain frequencies. An interesting idea is to identify each string of the guitar and fundamentally alter its frequency and characteristics; given enough time and resources, I feel this would be an avenue I would like to explore further. The EQ on each oscillator is highly effective; the 10-band-function offers even more pitch isolation and wavetable exploration. I'd like to expand this instrument to 8 oscillators, in order to manipulate fundamentals, harmonics, overtones and partials based on the western musical scales – however, building on '1xf=f' (Loy, 2006, p29) utilising a message or logic with this patcher could accommodate such a task and saving to a preset object.

Finally, the signals are summed and bussed to the `buffer~` via filters configured to the Nyquist frequency for optimum sample rate (Miranda, 2005). Upon final review, I realised I was still using the expression 'expr 1. - \$f1' (bfuhrman, 2020) and although this remained very effective and reliable I felt that this was a good opportunity to expand into more original concepts.

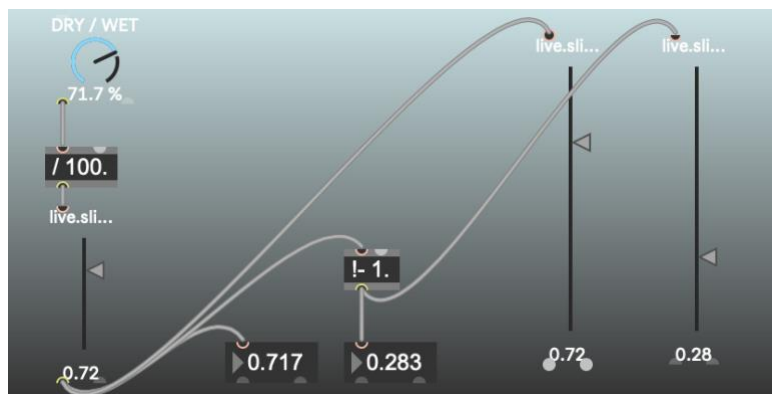


Fig.10 The new system for fading an alternating dry/wet signal written for the oscillator instrument using the 'divide input from a number' argument to reverse the orientation of control features.

Conclusion

The first stage of research included studying a wide variety of Max/MSP tutorials, reading through a selection of books dedicated to computer music, sound synthesis, midi programming, and digital sound design, and working through cycling'74 tutorials. I built a foundation of understanding of the capabilities of Max/MSP and therefore had a much clearer idea of how to execute this task. The process used in creating the final instrument involved a certain amount of trial and error; my initial ideas found room, within extensive research, to evolve and grow in unexpected directions. This enabled many different concepts and approaches, including using logic, audio capture and adopting music theory. Having met my objectives in this task, I look forward to developing these skills further, as the final instrument did not quite match my original concept. During the course of my research I encountered resources and material referencing the capacities of Max/MSP when integrated with visual effects generators, for example, and it excites me to think of the potential in the future.

Bibliography

- Loy, G., 2006. *Musimathics*. Cambridge, Massachusetts: MIT Press, pp.15-93, 29.
- Cipriani, A. and Giri, M., 2019. *Electronic Music And Sound Design*. 4th ed. Rome: Contemponet.
- Grosse, D., 2015. *Guitar Interfaces For Max*. [online] cycling74. Available at: <<https://cycling74.com/articles/guitar-interfaces-for-max>> [Accessed 20 June 2020].
- Benson, A. and Hart, L., 2020. *Expand Your Guitar, Vol. 1*. [online] cycling74. Available at: <<https://cycling74.com/articles/expand-your-guitar-vol-1>> [Accessed 20 June 2020].
- Dodge, C. and Jerse, T., 1997. *Computer Music: Synthesis, Composition And Performance*. 2nd ed. Australia: Schirmer/Thomson Learning, p83, 414, 123.
- Grosse, D., 2008. *Max 5 Guitar Processor, Part 1*. [online] Cycling '74. Available at: <<https://cycling74.com/tutorials/max-5-guitar-processor-part-1>> [Accessed 17 July 2020].
- Katz, B., 2015. *Mastering Audio: The Art And The Science*. 3rd ed. Oxon: Focal Press, p84.
- Robjohns, H. and White, P., 2002. *Multi-Band Compression: Practical Techniques*. [online] Sound On Sound. Available at: <<https://www.soundonsound.com/techniques/multi-band-compression-tips>> [Accessed 19 July 2020].
- Dawsons, 2020. *Overdrive And Distortion Effects – What's The Difference?* [online] Dawsons. Available at: <<https://www.dawsons.co.uk/blog/overdrive-and-distortion-effects-whats-the-difference#:~:text=of%20these%20effects%E2%80%A6,Overdrive,with%20a%20more%20dynamic%20response.>> [Accessed 20 July 2020].
- Russ, M., 2019. *Sound Synthesis And Sampling*. 3rd ed. London: Routledge.
- Oscium. 2020. *How To Measure Pulse Width & Duty Cycle*. [online] Available at: <<https://www.oscium.com/sites/default/files/wipry-combo/Final%2C%2520Tutorial%2520on%2520Pulse%2520Width%2520and%2520Duty%2520Cycle.pdf>> [Accessed 20 July 2020].
- Loy, G., 2011. *Musimathics*. Cambridge, Massachusetts: MIT Press, pp.434, 82-85, 33.
- bfuhrman, 2020. *7 And 8-1 Delay*. [video] Available at: <<https://www.youtube.com/watch?v=KHTP3FWFdnY>> [Accessed 17 July 2020].
- Russ, M., 2019. *Sound Synthesis And Sampling*. 3rd ed. Abingdon: Focal Press, p.441, 263.
- Reed, J., 2018. *Max 8 Tutorial #4: More Objects*. [video] Available at: <<https://www.youtube.com/watch?v=jd3xV8mtmTM&list=PLVIa8UkRzErsL95NoKH0QFaoLVMFqxbnA&index=5&t=0s&app=desktop>> [Accessed 21 July 2020].
- Miranda, E., 2005. *Computer Sound Design*. 2nd ed. Oxford: Focal Press.

Reed, J., 2019. *Max 8 Tutorial # 19: Recording And Playing From Buffers~*. [image] Available at: <<https://www.youtube.com/watch?v=n1H2lN4FSN4>> [Accessed 23 June 2020].

Mooney, J., 2009. *Introduction To MaxMSP: LFOs*. [video] Available at: <<https://www.youtube.com/watch?v=pmbYJkpwsio&t=288s>> [Accessed 15 July 2020].