During this module I have expanded my knowledge of algorithmic composition, its history, practices and where it may be headed. As well as this I have learnt about the theory of group work and presentation and found ways to build my resilience and motivation as an artist.

When I started generating ideas for the individual composition, I decided I wanted to explore biodata sonification, this is something I had already been working on in preparation for my final project, so it seemed suitable to use it in my composition. However, I was aware that, due to the time restraints, it would be wise not to make it integral in case I couldn't get the circuit to work on time. Considering this I decided to use the plant as a “random” trigger, effectively generating bangs that could then be interpreted by the Max patch and used in some way. This setup allows me to simply replace the plant with a random number generator if the Arduino stopped working for example.

Once I had decided on this I looked to the next issue, how to create a generative composition with a clear and pre-defined structure. We covered the use of tendency masks in class and this seemed like the best way of creating a shape that the piece could follow. Using two [function] objects I could simply draw two sections, one with rising energy and one with falling. I initially used these graphs to control the speed of a metronome connected to a basic random note generator, the effect being the tempo rose and then fell with the shape of the graphs. With the mechanism for shape decided, I started to control more parameters, controlling velocity and octave range added a lot of expression as the piece played out. Both these parameters are controlled by the graphs, this was fairly simple to implement, the chance that the system increases or decreases the octave of a generated note follows the curve and the velocity range does the same. All this lead to code that generated interesting and clearly structured pieces for a single instrument within Ableton live.

The next decision I made was how the notes were generated. I wanted to use more than just the random generation so I decided that, during the first section I would feed a Markov table the note outputs of the random object and then use the generated Markov Chain during the second half. I also decided to re-seed the random object every four notes, so it created four note melodies. This also gave me a use for the plant trigger, changing the seed was something relatively easy but still gave the plant a fairly significant roll, at least in the first section. This works well as I found I could “perform” using the plant, manipulating it to change the melody or simply waiting for its activity to spike naturally. This setup led to an interesting structure and sound, the first section with rapidly increasing energy and many repeated phrases suddenly breaking out into the Markov section where the notes conform to those played in the first half but played in more sporadic orders while the energy dies away. I later added a short ending section where the piece returns to seeded randomness for a short time to create a convincing ending.

The next obvious step was to add more instrumentation. I originally had issues with this as Max4live devices can only reliably control one instrument on one track at a time however I found that by using the send and receive objects I could simply send data across multiple patches. With this I was able to start adding different instruments, starting with the bass. This is a simple patch that plays the first two notes of every four-note phrase. This was done using the [count] object connected to the main [metro] of the patch. On the four and the one

the counter sends a bang along with the current midi note to the bass patch where it is pitched down two octaves and played. I feel this works well and is an easy way to add more instruments while keeping the cohesiveness of the piece and adding more structure. With the bassline added I continued to develop the instrumentation, adding a harp which plays an arpeggio based on a Markov Chain every eight counts, a crash that triggers when plant activity is detected as well as during the end of sections and strings during the final, seeded randomness section to bring the piece to a close.

The piece was now complete and with the implementation of some serial read objects I was able to add the Arduino and plant. The method for detecting plant activity is fairly simple and not something I designed myself. The circuit uses a “555” timer set as essentially an oscillator sending out pulse waves. The length of these pulses is dependent on the resistance across two conductive pads attached to the plant's leaves. The theory here is that plant activity will change the resistance between the two chosen points and so alter the length of the pulse. The Arduino then detects if there is a change in pulse width and, if there is, sends out a “1” via the serial port, this is then converted to a bang in Max which can be used in any way I like, in this case setting a random seed and triggering a sound. As a first exploration into biodata this works well, particularly considering it wasn't the main objective of the project. In future I will be looking at how the data from the plant can be more direct in its control of the music, rather than acting as a source of randomness. I will also create better and more reliable systems for detection and interpretation.

Overall, I am happy with both the outcome of my solo composition and the techniques I learned during its creation as well as generally throughout the module. I hope to utilise these skills in later pieces as well as further explore biodata sonification as part of my final project. There are several things I could improve about the project for example, the Max devices are fairly messy as I couldn’t encapsulate systems without them breaking and generally struggle with keeping my patches neat, this is something I will work on in future projects as it will make debugging and explaining my code to others far easier.

During discussions around the group project, it became obvious that we wanted to use OSC to create a composition across multiple computers, each running a different Max patch. Initially this caused problems as the university network did not allow for OSC communication between machines, this was solved by using a personal network with the machines being connected to a router via ethernet. With communication between devices sorted we started working on patches and code to generate music.

One of the developments came when Sam created a small web server where participants could enter a word, and that word would be sent out via OSC. I decided to take this and develop a Max patch which takes in the word, converts it to ASKII and then plays this as midi through Ableton live. This seemed like a simple way of converting letters and words into notes and melodies, and I feel it works well.

Throughout another of our brainstorming sessions we talked a lot about hands, both how gesture and movement could be used to generate music such as in Matias' use of "Hand-OSC" (faaip, 2020) as well as how the structure of the hand can be used to structure a piece of music. This latter idea interested me so I began researching ways that the hand has been used in this way, quickly discovering the Guidonian Hand a "Mediaeval mnemonic device to help singers - mostly monks and nuns - in their learning and memorization of hymns and masses"(Voss, 2021)I wanted to use this image to create some way of generating music so I simply used an image of the hand on a [pictslider] object that would allow me to move my mouse around the hand to play the different notes. I used WEKINATOR to assign each mouse position to a note (this avoided having to use a lot of if statements and seemed to work reliably). While my patch isn't a perfect recreation of the hand (I missed out some notes as they were too close and I wanted to prioritise playability) I like the aesthetics of using such an old idea to drive new music.

With the hand now working I thought of ways it could be controlled, originally using a colour tracker as a simple motion tracker with the idea of a performer moving their arms to play the notes on the hand I decided instead to collaborate with another of Sam's creations, a letter drawing and detection device where the user draws a letter from a grid of pixels and a WEKINATOR device is used to try and guess which letter is being drawn. The nature of this device lends itself to the hand as the performer will be making repetitive motions with the mouse, these motions can be turned into melodies using the hand so with Sam sending mouse data via OSC to my hand Max patch there was a new interactive way of generating notes.

A late addition to the performance was a "robot voice". This is a simple "espeak" program which runs on my laptop and takes the words from the text prompt and says them out loud, this is done in Max using the [shell] extension. This addition led to an interesting way of ending the piece with the voice repeating the word "hand" as our various sounds faded out.

Overall, I feel that the group project was successful, we worked together, and the blending of ideas and skills led to interesting compositions and code especially considering the time frame. In general, I think my code could be neater and I would have liked to explore the Guidonian Hand a bit more however that would likely involve a lot of research as I have very little understanding of mediaeval musical structures.

During both Projects I have expanded my knowledge not just of generative composition and algorithmic techniques but also biodata sonification and A.I. While learning about Markov Chains for example I experimented with the use of "Markovify" (Singer-Vine, 2020) (a plug-in for Python that allows you to generate strings of text based on Markov Chains) to generate poetry and further my understanding of Markov Chains. I have also, recently, experimented and researched Lehmer's Linear Congruence Formula a little more, I found the graph shown

in figure one of “Musical pattern generation with variable-coupled iterated map networks”((BATTEY, 2004) useful for visualising the way that the formula can create repeating or self-similar patterns so decided to create a Python program to generate graphs based on its output(see fig.1 and fig.2). This helps me understand and visualise how different inputs can drastically affect its output and to discover settings with interesting shapes that might be useful in compositions.

When it comes to biodata sonification I have learnt a lot of the basics through the solo composition. I haven’t gone into too much depth on the code or circuit I used for this project but much of my research uses “Electricity for Progress’”(Cusumano, 2020) website and GitHub page for the circuit and as a base for my code as well as various other tutorials and articles which can be found in my bibliography.

Throughout this module I have built upon my existing skills, both practical (in the form of programming techniques, use of tools such as WEKINATOR) and soft (group work, resilience etc) I have also discovered useful creative techniques through the course of this module, one being the use of morning pages. I often found it hard to keep this as a habit and still struggle but can see the benefits when I do remember. As well as this the ideas surrounding group work and thinking were useful and interesting, particularly Jo Butterworth’s “Didactic-democratic framework model” and ideas around community art projects. When it comes to my portfolio, I am largely happy with the results I achieved though there are areas I would have liked to explore more such as using iterative maps in my work. I would also have liked to do more research into AI and its use in art, particularly the ethical questions it poses.

Word Count: 2091

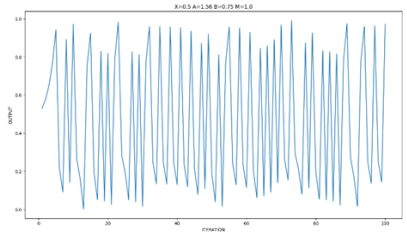

Fig.1 a graph generated by Lehmer's linear congruence formula (drawn in Python using matplotlib)

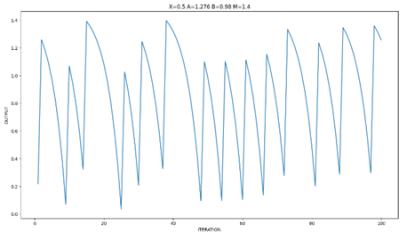

Fig.2 another graph generated by Lehmer's linear congruence formula (drawn in Python using matplotlib)

Reference list

BATTEY, B. (2004). Musical pattern generation with variable-coupled iterated map networks. *Organised Sound*, 9(2), pp.137–150. doi:https://doi.org/10.1017/s1355771804000226.

Cusumano, S. (2020). *Biodata Sonification*. [online] Electricity for Progress. Available at: https://electricityforprogress.com/biodata-sonification/.

Cycling74.com. (2024a). *split floating point number at decimal - MaxMSP Forum | Cycling '74*. [online] Available at: https://cycling74.com/forums/split-floating-point-number-at-decimal [Accessed 11 Nov. 2024].

Cycling74.com. (2024b). *Tool: shell | Cycling '74*. [online] Available at: https://cycling74.com/tools/bernstein-shell [Accessed 14 Nov. 2024].

faaip (2020). *GitHub - faaip/HandPose-OSC: Handtracking using MediaPipe HandPose. Runs as an Electron app and outputs OSC*. [online] GitHub. Available at: https://github.com/faaip/HandPose-OSC [Accessed 10 Nov. 2024].

Programming for People (2015a). *MAX msp 7: colour tracker - part 2*. *YouTube*. Available at: https://www.youtube.com/watch?v=520LIom0U4k.

Programming for People (2015b). *MAX msp 7: colour tracking - Part 1*. *YouTube*. Available at: https://www.youtube.com/watch?v=t0OncCG4hMw [Accessed 4 Mar. 2021].

Programming for People (2016). *MAX msp 7: Sending data from arduino into Max*. [online] YouTube. Available at: https://www.youtube.com/watch?v=6bT3G4Mep7E [Accessed 16 Oct. 2024].

ProgressFollow, electricityforprogressElectricity for (n.d.). *Biodata Sonification*. [online] Instructables. Available at: https://www.instructables.com/Biodata-Sonification/ electricity for progress Instructable for biodata sonification using Arduino.

Singer-Vine, J. (2020). *jsvine/markovify*. [online] GitHub. Available at: https://github.com/jsvine/markovify.

Voss, C. (2021). *Chris's Curiosities: A Medieval Helping Hand*. [online] CRB. Available at: https://www.classicalwcrb.org/blog/2021-09-29/chriss-curiosities-a-medieval-helping-hand.

Wikipedia Contributors (n.d.). *File:Guidonischehand-en.gif*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/File:Guidonischehand-en.gif.