Drum Machine and Pattern Generator

Irfan Adil Navaz (Student ID: 301074887 Exam no: Y3928473)

CONTENTS

I. INTRODUCTION AND MOTIVATION

The acronym Max/MSP, which stands for Max/Music and Max/Signal Processing, refers to a flexible visual programming language and environment that is mainly utilised in interactive art installations, multimedia, and music production. Max/MSP, created by Cycling '74, offers a robust platform for real-time creation, manipulation, and control of audio, video, and other media. Max/MSP's intuitive graphical user interface allows multimedia artists and musicians to express themselves freely and experiment with new ideas.

The fundamental idea behind Max/MSP is "patches," which are graphical depictions of algorithms and procedures. Within these patches, users can connect virtual objects known as "objects" to build intricate chains involving data analysis, MIDI manipulation, signal processing, and more. This visual approach to programming makes Max/MSP accessible to users with varying levels of technical expertise, from beginners to seasoned professionals.

Flexibility is one of Max/MSP's main advantages. Whether they're creating interactive installations, live performance tools, or custom synthesisers, users can create patches that are specifically tailored to their needs. Max/MSP's modular design makes it possible to seamlessly integrate external third parties, further enhancing its capabilities. Furthermore, Max/MSP facilitates communication with external devices like MIDI controllers, sensors, and cameras by supporting a wide range of hardware interfaces and protocols.

My patch is named 'Drum Pattern Generator', and I've decided on this particular instrument patch because of the unique and groovy patterns that can be generated in a patch like that. Especially since the sound sources are developed from a noise or a sinusoidal wave, there is opportunity to expand and even change the sound sources to whatever we need or feel.

II. MARKET RESEARCH

In the process of producing music, rhythm and groove can be created using drum machines and drum pattern generators[1]. The majority of musical compositions are essentially composed of drums, which act as the piece's structural core and source of rhythm. Drum machines provide musicians with a wide variety of pre-programmed patterns, programmable sequences, and real-time performance options that allow them to explore a variety of rhythmic textures and styles, from conventional beats to innovative electronic rhythms.

It can produce a level of consistency and accuracy that is frequently impossible to accomplish with just live drumming. They enable producers to create precisely timed rhythms that mesh flawlessly with vocals, melodies, and basslines in other parts of the song. This accuracy is particularly important in genres where rhythm drives the energy and momentum of the music, such as pop, hip-hop, and electronic dance music (EDM).

III. DESIGN AND IMPLEMENTATION

The design of the patch can be broken down into three sections, Sound Source, matrix Control, Pattern Generation.

For this particular patch, we have three specific sound sources that we use to generate drum tracks. It's a combination of a kick, snare drum and hihats. For each sound source, we use an ADSR (Attack Decay Sustain Release) amplitude envelope. We construct that envelope to make the sounds generated as transient and percussive in nature. For the kick, we generate a sinusoidal wave of 100 Hz and use the envelope to develop a nice sounding kick. For the snare and hihats, we use a filtergraph to control the EQ and modify the sound to mimic a snare and hihats.

Once we have constructed our three sounds, now it's time to play according to a pattern. For generating a way to store patterns, we use the Matrixctrl object[2]. Here, we keep the matrix to 3x8 size, as we are currently using three separate sound sources over an 8 step bar. Using the matrixctrl, in combination with getcolumn and unpack, we can receive unpacked list values and use these values to trigger our sounds as we go 1 step at a time. To keep it in 8 steps, we use a counter object to go from 0 to 7, and with each step, the column is read and the sounds are triggered accordingly.

Now that we can generate patterns and play sounds, we now move forward to the next step of creating random pattern generation. In this particular step, e use the pack object to obtained list values from the matrix to feed into an itable and generate a probable value. So for each instrument, a value between 0-1 is determined and the probability of choosing any of the specified values is controlled between a range of 0-10. We set it up in such a way that we use the Uzi object to generate a stream of 8 bangs, we is then

selected from 0 to 7, and the pattern for each instrument is determined and is read into the matrix, from there, we can hear out randomly generated patterns and we can start exploring new patterns.

We can also use a sel object, to hook it to the counter so that at the end of every 8 step bar, the pattern is refreshed and randomised.

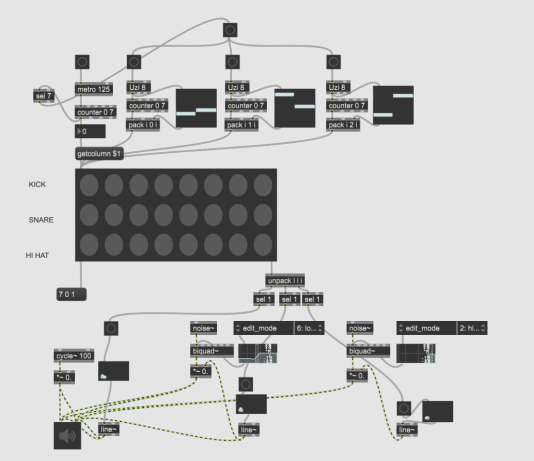

Fig. 1. Pacth Design

IV. Application of HCI principles

1. Friendly Interface: Imagine opening up the drum generating machine and feeling like you're instantly familiar with it. Everything is where you expect it to be, labelled clearly, and easy to understand. It's like meeting an old friend who knows exactly what you need.

2. Encouraging Feedback: Every time you hit a drum or tweak a setting, the machine responds with a reassuring nod – a visual cue or a satisfying sound that lets you know you're on the right track. It's like having a supportive companion cheering you on as you explore your musical ideas.

3. Tailored to You: Just like your favourite pair of jeans, this machine fits you perfectly. You can adjust the tempo, mix up the rhythm, and choose the exact sound you're looking for. It's all about making something that's uniquely yours, reflecting your musical taste and style.

4. Oops-proof Design: We've all been there – accidentally deleting something important or making a change we didn't mean to. But with this drum machine, it's like having a safety net. It gently reminds you to save your work or double-check before making big changes, preventing those frustrating "uh-oh" moments.

5. Effortless Flow: Time is precious, especially when inspiration strikes. That's why this drum machine is designed to keep you in the groove without getting in your way. With shortcuts, drag-and-drop features, and seamless transitions between tasks, it feels like the machine is working with you, not against you.

6. Easy Learning Curve: Learning something new can be daunting, but not with this drum machine. It's like having a patient teacher who guides you through each step, offering helpful tips and encouragement along the way. Before you know it, you're creating drum patterns like a pro.

7. Accessible to All: Music is for everyone, and so is this drum machine. Whether you're a seasoned musician or a complete beginner, it welcomes you with open arms. With options for different needs and preferences, it ensures that everyone can join in the fun, regardless of ability.

8. Consistently Comforting: Like your favourite song that never gets old, this drum machine is reliably consistent. You know exactly what to expect each time you use it – from the layout to the terminology to the way it responds to your commands. It's like having a dependable friend who's always there for you.

By infusing these human touches into the design of the drum generating machine, it becomes more than just a tool – it becomes a companion on your musical journey, supporting you every step of the way.[3]

V. Critical Evaluation

After creating and testing the patch, we do find some challenges involving creating a more substantial drum machine with recording capabilities and maybe possibly depending on a sample bank. With the use of noise and wave generators, we can still create almost any sound we want really, but it might just be easier and more efficient to create and own a sample bank that you might like, that can just be loaded into the instrument.

We could possibly extend the steps involved in each bar, but that probably depends on the particular musical project an artist would be working on. There are other areas that can be critiqued upon. The sound generation from waves and noise is a little inefficient, considering the amount of noise that can be introduced in the final drum track unless filtered properly by the user themselves.

Through the utilisation of Max/MSP, users can create a customised drum machine tailored to their individual creative requirements and preferred workflow. With its unmatched flexibility, this method lets users play around with different sound combinations, interactive elements, and creative sequencing strategies.

VI. Conclusion

The assessment of drum machines critically and the possibility of building a custom Max patch for a drum machine highlight how dynamic music technology is and how much it influences artistic expression. Through a commitment to innovation, teamwork, and user-centered design principles, we can keep pushing the envelope of creativity and discovering new avenues for music production and performance.

Within the Max/MSP ecosystem, creating a Max patch for a drum machine offers chances for community involvement, investigation, and cooperation. By exchanging ideas, sharing patches, and adding to the body of collective knowledge, users can promote innovation and ongoing development.

References

[1] R. Vogl and P. Knees, "An intelligent drum machine for electronic dance music production and performance," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2017.

[2] M. Blackburn, “Working with brevity: Short soundfiles in electroacoustic composition,” in *Proceedings of the 2019 International Computer Music Conference, ICMC-NYCEMF 2019 - International Computer Music Conference New York City Electroacoustic Music Festival*, 2019.
[3] I. Bergström and A. F. Blackwell, “The practices of programming,” in *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC*, vol. 2016-November, 2016.