Live Thematic Music Composer Written Report

By Nicholas Salter (NDS8)

1 Abstract

The Live Thematic Music Composer is a program written in Java (SE 14.0.1) using the built in MIDI system as to create music on the fly based on the thematic that is chosen by the user. It also allows for a transition from one theme to another using a dropdown menu and creates a nice transition between them. This allows for a dynamic and creative generation of music to use in live events where mood music and quick changes are needed such as playing board games, drama improvisation, live storytelling and so on. This system does so by choosing notes and chords based on the last set of notes, making a chain of notes to a random rhythm such that it imitates actual musicians improvising on the piano.

2 Introduction

Music is a tool we have had from the very beginning. From as early as cavemen carving flutes from mammoth tusks to the modern day where we have Artificial Intelligence writing music based on songs run through it, we've developed our musical palettes with ever increasing technology [1, 2].

However, with our increased technology in computational musical improvisation in different genres, we have not taken the chance to combine musical transitioning between two thematic scales seamlessly. This is what our project sets out to do, to not only creatively compose music but to have it be seamlessly changeable from one theme to another. It may not seem clear to some what the benefit to such a system would be, to this I suggest the use of improv for telling a story on the go without knowing the outcome prior to when the music is needed.

Examples range from improv drama performances in order to use music to strengthen moods of scenes seconds after they occur, using the program for roleplaying games such as dungeons and dragons to evoke an emotion for descriptions and actions taken by players and even live reading of stories and books to an audience. While niche, it is an excellent chance to test what we can do with technology in the modern day as well as providing an excellent mood fitting tool.

3 Background

Now having discussed the intentions, the journey to creating an unending melody along with chords lies in computed musical improvisation history which goes back to 1972 where Moore suggests the use of formal languages and grammar as a method for developing a melodic tune [3]. Moore suggests establishing a set of algorithmic rules for the grammar to produce a melody, likening the ability to speak as similar to the ability to compose music saying that the two both have large amounts of content derived from small grammars and syntaxes.

Following this, many others came such as Lidov and Gabura (1973) who took interest in the idea and a proposition of how such grammars could be constructed were proposed [4]. These rules were eventually developed into synthesis engines such as in Ulrich's model for computing jazz [5] and goes even further as to use inferred rules from classical music to generate music in the style of these composers of the past in Baroni and Jacoboni's system [6].

Easily designed, these systems looked perfect for the defined task but to make sure, research into alternatives in live music composition was necessary. With recent advances into artificial intelligence, as a species we have managed to artificially craft impressive pieces of music. Using a mixture of techniques such as neural networks, genetic algorithms, and other evolutionary techniques, computer's have created music so successfully that they've even managed to have concerts from all the music their software [1].

Further examples of this are seen in Mozar's neural network that took my interest for its psychophysically grounded constraints [7]. It was designed to investigate both local and global patterns within its training data, looking for similarity in pitches and match them to similar outputs for a stunning result.

Unfortunately, these methods are computationally expensive and require lots of training data that would be hard to come by without any kind of funding or at least an exceptional amount of time like Flow Machines had. Not only this but rule based systems were surprisingly quick for calculating melodies and are perfect for this project as due to using rules, this methodology was computationally inexpensive which is what would be required for the live improvisation of music in public settings. This, overall, was the primary reason for the methodology used.

4 Methodology and Design

To begin this, I required a system in which I could create musical notes using code. A variety of options are open for this ranging from cheap and extremely simple programs like Scratch all the way to Logic Pro which manages to not only be an extremely well put together music program but also allows for the programming of music. Unfortunately, with the constraint of having my program be able to be run quickly and simply with the least amount of extra bulk, I settled for something with a lot less extras built in: Java's MIDI system. Not only was I experienced in it, but it was efficient and clean. With this I got to work.

4.1 User Interface

Firstly, I needed a UI to allow people to play, pause and transition easily which lead me to get designing in paint. Having created the framework to open UI, I took to paint to create an idea of what it should look like as seen in figure 1.

Figure 1.

Though crude, it allowed the user to stop whenever and to play whenever and even allowed users to listen back to the song created by pressing the reset button. Not only this but it had a drop-down menu for selecting which type of song they want and upon selecting it, the music would change to it. A plan for the UI had been made and now it was time to investigate how to craft the melodies that continued.

4.2 Melody Creator

4.2.1 Note Generation

This is by far the hardest part of the composer to design since it requires the music to be fresh and develop overtime without sounding too repetitive. For this, inspiration was taken from how humans improvise on scales and the design was to play notes on a scale and base the next note on the last note, determining whether it sounded good or not based on said last note.

This would require each note to have a list of other notes it could change to based on rules to create a grammar for this improvised language of music. An example table is shown below in Figure 2, displaying what notes would have what percentage change of being played. In the figure, +12 refers to the same note one octave higher, -12 is one octave lower and same refers to the chance of the note being played once more to allow for some repetition.

Table for the C Note in Major Scale							
Notes Above C							
C (Same)	D	E	F	G	A	B	C (+12)
20%	15%	10%	2%	1%	1%	1%	10%
Notes Below C							
C (-12)	D	E	F	G	A	B	C (Same)
10%	1%	1%	1%	2%	10%	15%	20%

Figure 2.

A table like this would have to be created for each different note in the Major Scale as well as being flexible enough to apply to that note of any octave. This being crucial for efficiency as the MIDI piano has 128 notes, 75 or so notes being of the Major Scale. Since each note gave a certain remainder if divided by 12, it would be easy to create a calculation to determine if a note were of a certain melody. The formula is the following: N%12 where N is the note's value. This could be used to determine a note as well as what note would be above it or below it and setting the next note by adding or subtracting to the current note.

Once this machine was set up, it would still require testing and adjusting as certain notes may not go with others in the scale meaning adjustments to what notes could be played based on the last note would have to be made. Not only this but a set of tables would have to be made for each scale that is to be implemented and each one would require testing.

4.2.2 Rhythm Generation

While all this creates the notes that are to be played, a rhythm had to be created. With time constraints as they are, I have opted for a simple solution to making a relatively varied rhythm that changes dynamically. The music generated will be in a 4-4 pattern at a tempo of 220 beats per minute (bpm) for a relatively fast paced song. Every beat in the 4-4 pattern will contain a note with a chance for the note to be skipped and the next note is instead played one beat later. This simulates the general improvisation rhythm that comes with practicing piano and allows for something that works exceedingly well with live content. Not only that but with such a fast tempo, transitioning from one theme to another is quick and complements the fast-paced nature of improv and storytelling.

4.3 Chord Creator

The chords would have to be generated based on the notes that are playing to keep them in tune with the rest of the improvisation going on. This being a must, a simple solution is to take the note that has just been played and play a chord that works alongside it. This requires similar code to the note generation, identifying what note is going to be played and playing a chord based on that. This means finding the note and finding notes below it that fit it within the scale defined by the UI.

4.4 Scale Transitioning

This project differentiates itself on the ability to have the music not only be improvised but to be able to switch to another theme just as quickly. It was not plausible to have 3 sets of music being generated at once and just switch between them when called for and so music must be generated quick enough that the program can just generate music within the next 4 beats. This would be hard with a tempo of 220bpm so the code must be efficient. However, the advantage of such speed would mean that the program would be responsive to changes quickly and the transition would be short. This would mean the transition would have to be at the end of a 4-4 beat in, basing the next note on the last note played at the end of the 4-4 beat and as a result required a conversion of that scale to this new scale. This was especially efficient and would make the transitions almost unnoticeable.

4.5 Testing

To test this program, we would need to save the songs to show to others and as a result, the UI needed an extra button to write MIDI files giving us with the final UI as seen in Figure 3:

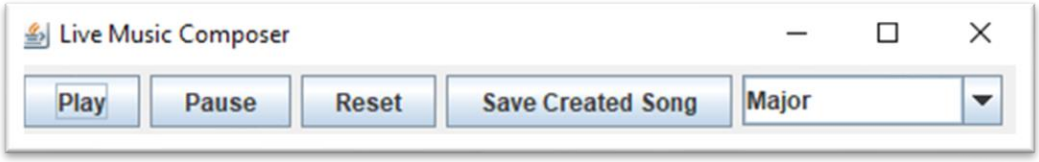

Figure 3.

Furthermore, testing would require multiple listening by multiple people to adjust the program and then a final test to show the success of this program, I would put an exert of myself playing piano in a folder with the program's MIDI output and ask people to distinguish which were human and which were not as a simple Turing Test to see the success of the program.

However, this would imply that there must be one or the other so as to stop people from guessing and to remove the Hawthorne effect from participants, they were told that there is a chance that both were human and a chance that both were robots as to have a fair trial. Not only this but half of the participants were given the samples in the opposite order as to remove Order Effects in testing as to have more accurate results.

5 Results

5.1 Testing Results

The following Figure 4 displays the results of 19 participants who were asked if each sample was composed by a human or a robot. These results from the testing show that despite Sample

1 being recognised as a robot more often than Sample 2, on the whole, Sample 1 was said to be human more than a robot which would imply that it is close to passing the Turing test. However, despite this fact, it did get distinguished as a robot more than a human and so it inevitably failed this test, even if it were close. It would need to be called a robot equal to or less than its human counterpart.

Participants (Sample 1 First)	Sample 1 (Program)	Sample 2 (Human)
1	Human	Robot
2	Human	Human
3	Robot	Human
4	Robot	Human
5	Robot	Human
6	Robot	Robot
7	Human	Human
8	Human	Human
9	Robot	Human
10	Human	Human
11	Human	Robot
Participants (Sample 2 First)	Sample 1 (Program)	Sample 2 (Human)
12	Human	Human
13	Robot	Robot
14	Robot	Human
15	Human	Human
16	Robot	Human
17	Human	Human
18	Human	Human
19	Human	Human

Figure 4.

5.2 Program Results

This program not only creates music on the fly of three different themes, it manages to transition between them and is often been called human in its processing. The program has successfully completed its two primary goals of creative computational of music as well as being able to transition between the music it makes. Not only this but it manages to create music that has likely never been heard before from how many combinations it can make. With more time perhaps, additions I would make are a volume bar, a tempo selector, and more themes/scales within the program. Along with this I would add a queue to that can be lined up for easier control of the output.

6 Evaluation

6.1 Errors Handling

The project will not work on all computers without the installation of Java SE and a sound bank containing MIDI notes and as such an error detector was created to state that the user did not have the either and to please install them. Not only this but the program has many methods to catch issues that may occur within start up. During the process of making a song, the code makes sure to have default values in case a note or chord is not found, allowing for the work

to keep going regardless of missing notes allowing for a clean and continuous system without erroneous behaviour. An issue that may occur is that every so often odd notes are played, while not an error in the code, the rules created to design the melody are not completely perfect and with more time and refining, those odd notes could be removed.

6.2 Theories of Creativity

To analyse the creativity of the program, I have elected to use the Creative Tripod [8] by Colton for its creative direction toward skill and imagination which are two contributing factors to human musicians improvising music as well as the FACE [9] by Colton, Charnley and Pease for its focus on aesthetics and concept driven understanding of creativity as these are important factors in themes and the ability to craft them such that this program does.

6.2.1 Creative Tripod:

Skill – A system defined as having skill would have to have to consciously understand or display a talent for this particular area. With this music generating music as fast as it does on the go and still managing to sound nice to listen to, it takes great talent from a human musician to replicate such skill to not only play notes as fast as the program does but to also make it sound good. Not only this but at this speed a transition between two scales would take a lot of mental processing power which is a skill most competent musicians who improvise often don't have as they often do not need to transition between themes. While clearly skilful, it is constrained by rules and would not be considered to understand the area so much as just following rules set by a human. Despite this, the program has some clear talent for what it can do compared to the average musician and as such would score highly in this section.

Appreciation – A system defined as being appreciative of what it is creating is defined by understanding which areas of what it has created are good or bad. This system has rules to determine which notes to use based on previous notes which allows it to produce a piece of value however it is not evaluating the piece as a whole and may produce songs that are similar without realising. This would overall score a medium value as it does produce a nice melody for each scale but only evaluates it on a note by note basis and not the entire song.

Imagination – A system defined as being imaginative is defined as producing works not created by humans before. For this, due to each tune being flexible and relying only on the previous note, each melody created has a large amount of imagination, using all the notes available on a scale. To add to this, the rhythm being randomly generated adds to the imagination created by this program. It continuously looks outside of the box with the only way to improve being the chance that it might go outside of its scale to add notes not previously being utilised. As a result of how many songs can be generated with each run of this program, the imagination would score highly, more so than in skill and appreciation.

6.2.2 FACE:

Framing Information – Framing information is defined as generating content within a context, framing it, and giving it value. This program does such by changing scale based on the theme selected, putting each song in the style of the selected theme. This however is not generated by the program itself and as such would score poorly in this section.

Aesthetics – This section looks at how the program looks at parts it creates and makes decisions based on those generations to match aesthetics. With how the program works, deciding which

chords to add based on which notes are going to be played and remaining in the constraints of a theme allow the program to judge what would and would not fit the aesthetic of the program. As a result, the program would score highly in the aesthetics category.

Concept – The concept section identifies whether a program or executable can take inputs and produce an output as a result. In the case of this program, the input is the theme as well as the time at which the theme is changed to another. The program takes both and manages to craft an excellent output based on such few inputs, ultimately scoring quite highly in this section.

Expression of Concept – This final section looks at if the concept has not only been instantiated but has instantiated a new creation rather than stagnating creatively. This program with its range of 75 notes per scale as well as a total of 24 different 4-4 rhythms selections, this program has over 750,000,000 possibilities for each bar within each 4-4 bar giving it an exceedingly high score in expression of concept.

Overall, judging this program from an understanding point of view, the program does not have too much nuance in telling what song is good and what song is not but does have a good idea of what notes go together. However, looking at this program's ability to compute brand new tunes and songs on the fly, it scores immaculately, displaying a high grade for creativity.

6.3 Further Research and Expansion

With the completion of the primary goal of having a transition between different music themes, a follow up goal should be to improve this to the point of which a large structure is put in place, where the composition is split into verses and choruses and manage to change such that the music is at a standard where one would consider it a self-writing musical. While extremely versatile in its current form with change between themes being near instant, this 4-4 beat change can definitely be implemented mid chorus in order to create a very enticing change between tone. A good example of this would be in the song "Confrontation" in the musical "Jekyll and Hyde" where the song changes from a lament to a villain's song and then a duet. It would require a clever idea of where the song currently is and the pacing that the program has created up until this point so that it may transition without forcefully removing the audience's immersion and pulling the spotlight away from the performance this music may be accompanying.

Not only this but another idea of further research would be to add transitioning between different tempos of music as well as transitioning between two songs. This could be done with queueing up verses and choruses of different themes but would require a significantly sophisticated transition system to not sound awkward when transitioning.

Finally, another area research could take these concepts is to add lyric generation to these songs. This would allow the use of this software to create songs on the go for concerts, opening the door to what may be the future of music generation. This could even take similar concepts from grammar and syntax rules as early language generation did as to do this. However, another way of generating these lyrics may be to use a neural network to look through songs of a band to generate lyrics similar to their own but with maybe a few extra song lyrics to add variation.

Of course, these are speculative concepts but with this door opened, the possibilities of live music content generation are endless.

Suggested further reading would be Mozer's work into having atmospheric music composition using neural networks and human defined constraints as the technology would be able to create chords and other instruments to fit to this project's method of creating music [7]. This can be further seen in work done by Flow Machines which is an exceedingly well made music generator [1].

7 Conclusion

To conclude, this program successfully completed its primary goals of seamless transition between themes of music and the generation of multiple themes of music on the fly. Of course, with more time, I would implement a lot more themes as well as different tempos to test this concept fully. It would also be nice to see how this program worked with the combination of different instruments playing at once in a bad but overall I had learnt a lot about music in order to complete this project as well as the depths of which the MIDI system built into Java can be taken. To finalise, this project demonstrated the ability to have live music transition from one theme to another as well as the creation of music within these themes successfully.

8 References

[1] "Flow Machines," Sony, [Online]. Available: http://www.flow-machines.com/. [Accessed 02 05 2020].

[2] "Earliest music instruments found," BBC, [Online]. Available: https://www.bbc.co.uk/news/science-environment-18196349. [Accessed 02 05 2020].

[3] J. A. Moorer, "Music and computer composition," *Communications of the ACM,* vol. 15, no. 2, pp. 104-113, 1972.

[4] D. Lidov and J. Gabura, "A melody writing algorithm using a formal language model.," *Computer Studies in the Humanities and Verbal Behavior,* vol. 4, no. 3-4, pp. 138-148, 1973.

[5] J. W. Ulrich, "The analysis and synthesis of jazz by computer," *Proceedings of the 5th international joint conference on Artificial intelligence,* vol. 2, p. 865–872, 1977.

[6] M. Baroni and C. Jacoboni, Proposal for a grammar of melody : The Bach chorales, vol. 4, Montréal: Les Presses de l'Université de Montréal, 1978.

[7] M. C. Mozer, "Connectionist music composition based on melodic, stylistic and psychophysical constraints," *Music and connectionism,* pp. 195-211, 1991.

[8] S. Colton, "Creativity Versus the Perception of Creativity in Computational Systems.," in *AAAI spring symposium: creative intelligent systems*, London, 2008.

[9] S. Colton, J. W. Charnley and A. Pease, "Computational Creativity Theory: The FACE and IDEA Descriptive Models.," in *ICCC,* 2011, pp. 90-95.