

https://dspace.org/

MUCUS (Music Composition User System): Infectious Flexible Creative Interaction with an Algorithmic Music Composing Application

Armstrong, Douglas Connolly

2024

University of the Witwatersrand, Johannesburg

https://hdl.handle.net/10539/44670

Armstrong, Douglas Connolly . (2024). MUCUS (Music Composition User System): Infectious Flexible Creative Interaction with an Algorithmic Music Composing Application [PhD thesis, University of the Witwatersrand, Johannesburg]. WIReDSpace. https://hdl.handle.net/10539/44670

Downloaded from a DSpace repository. https://dspace.org/

MUCUS (Music Composition User System): Infectious Flexible Creative Interaction with an Algorithmic Music Composing Application

Douglas Connolly Armstrong

Student number: 2295472

Supervised by Dr Jonathan Crossley and Dr Cameron Harris

Dissertation submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy in Music
Wits School of Arts, Faculty of Humanities, University of the Witwatersrand.

Johannesburg, March 2024

Declaration

I declare that this research thesis is my own unaided work. It is submitted for the degree of Doctor of Philosophy in Music at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any other degree or examination at any other university.
Ethics clearance certificate protocol number: H20/06/01



_____

Douglas Armstrong

13 March 2024

Abstract

Recent research in the field of computational creativity and interaction design suggests new ways in which computers can contribute to a creative process. Computational creativity has necessitated new rigour in theoretical definitions of creativity for computational applications. Interaction design has evolved from a focus on efficiency and productivity to a user-centred focus on the emotional and hedonistic aspects of interaction with a computer.

I set out to design an interactive algorithmic music composing tool that implements co-creative strategies for a human–computer collaboration, as described by Kantosalo and Toivonen (2016). This interactive tool would play the role of a creative collaborator in the development of musical material by the use of rule-based algorithmic music composing models, steering the co-creative composing process using high-level musical descriptors and capturing user sentiment to build a model of the user's musical preferences. It is suggested that, through this process, an engaging creative interaction between human and computer can be sustained.

Three versions of the software were tested in three different settings. The results suggest that a variety of co-creative and creativity support approaches are required to cater for a computational agent that does not match a musically trained human's ability to identify musical merit in a developing idea. In this respect, it was found that there is further scope for exposing details of the computer decision-making process for development alongside the process of musical development, as a black box process of computational reasoning was found to be mysterious and at times frustrating. A rule-based system of music generation was found to be effective in a steering mechanism that matched higher-level descriptors of the musical variation process to music generation parameters. Engagement was sustained for longer when the duration of the musical output was longer in form. This included the ability to integrate the output of the application with existing digital audio workstations.

Acknowledgements

I would like to acknowledge the help of my supervisors, Dr Jonathan Crossley and Dr Cameron Harris, whose support and encouragement kept me going.

To all the musicians and artists whom I have had the pleasure of interacting with over the course of interesting musical times: all of you have contributed to where I am, and I am grateful for everything.

To my parents, Margaret and Michael, who originally allowed me to think this was possible. To Adrien, Sophie and Anna, who have been so patient with me over the course of this work. And lastly to Simone, for everything.

Contents

Declaration ..... **i**

Abstract ..... **ii**

Acknowledgements ..... **iii**

Contents ..... **iv**

List of Figures ..... **vii**

1. Introduction **1**

1.1. Aim 1

1.2. Rationale 2

1.3. Background 3

1.4. Research question 5

1.5. The AI landscape 5

1.6. Rule-based generation 13

1.7. Data-driven deep learning 15

1.8. Musical interfaces and creative practice 16

1.9. The case for rule-based generation 18

2. Literature review **20**

2.1. Creativity 20

2.1.1. Types of creativity 20

2.1.2. The process of creativity 21

2.1.3. Components of creativity 22

2.1.4. Computational creativity 25

2.2. Models of interactivity 27

2.2.1. Models of creative interaction 29

2.2.2. Creativity support tools 30

2.2.3. Computational co-creativity 31

2.3. Models of algorithmic composition 34

2.3.1. Grammars 38

2.3.2. Case-based reasoning 38

2.3.3. Rule learning 38

2.3.4. Constraint satisfaction 39

2.3.5. Concurrency models 39

2.3.6. Other symbolic, knowledge-based systems 40

2.3.7. Evolutionary algorithms 40

2.3.8. Interactive algorithmic music examples 43

2.4. Research by design 44

3. Methodology 45

3.1. Scope of project 45
3.1.1. User interface 47
3.1.2. The user 48
3.1.3. Interaction 50
3.1.4. Music generation 52
3.1.5. Music algorithms 52
3.1.6. Analysis 53
3.1.7. Musical medium 54
3.1.8. Software development and implementation 55

3.2. Testing 55
3.2.1. Pre-alpha testing 55
3.2.2. Alpha testing 55

4. Method 57

4.1. Step 1: Design constraints for a user-friendly algorithmic music composing framework with options to intervene 58
4.1.1. User-friendliness 58
4.1.2. Early interventions 63
4.1.3. Phrase-length melody model 64
4.1.4. Reflection on this stage 66

4.2. Step 2: Interaction models 67
4.2.1. Finalising the interaction model 68
4.2.2. Implementation and testing 72
4.2.3. Reflection 75

4.3. Step 3: The qualitative pre-alpha tests 76
4.3.1. Pre-alpha test presentation platform 76
4.3.2. Pre-alpha test questionnaire 77
4.3.3. Reflection on pre-alpha tests 78

4.4. Step 4: Quantitative alpha testing 89
4.4.1. Updating the musical behaviour 89
4.4.2. Interaction and user interface 90
4.4.3. Finalising for cloud deployment 96
4.4.4. The online survey 98
4.4.5. Alpha test results 98

4.5. The interaction in a personal composing project 99
4.5.1. Constraints for a composing project 100
4.5.2. Some starting points 102

4.6. The first composing experiment 103
4.6.1. Melody material 104

4.7. The second composing experiment 110
4.7.1. Melody material 110
4.7.2. Example of melodic development 112
4.7.3. The case of the key 123
4.7.4. Generating and arranging the entire melody 125
4.7.5. Development of the accompaniment material 129

5. Results 134
5.1. Quantitative testing 134
5.2. Composing experiments 136
5.3. Technical observations 137
6. Discussion 139
6.1. The rule-based generation model 139
6.1.1. Music representation 139
6.1.2. Melody model 140
6.1.3. Accompaniment parts 140
6.1.4. The feedback mechanism 141
6.1.5. Rule-based generation 141
6.2. Real-time sentiment collection and the robot variation 141
6.3. The co-creative interaction and randomness 143
6.4. Engaging the user 144
7. Conclusion and future work 146
7.1. Extending the form 146
7.2. Data gathering 146
7.3. Explainable AI 147
7.4. Accompaniment material 148
7.5. Last thoughts 148
8. Bibliography 150
Appendix A: First version of the random generation output 160
A1. The set of parameter options 160
A2. The random number file 164
A3. The parameter file 166
A4. The printed output 168
A5. Feedback options for pre-alpha testing 169
Appendix B: The online survey 171
Appendix C: Score of the first composing experiment 179
Appendix D: Score of the second composing experiment 182

List of Figures

Figure 1: Model of the flow state (Csikszentmihalyi, 2014, p. 147) 27
Figure 2: Layers of an interaction model (Kantosalo et al., 2020, p. 59) 31
Figure 3: Taxonomy of algorithmic composing methods according to Fernandez and Vico (2013) 37
Figure 4: Chadabe's illustration of sound generation based on a classical feedback model (Chadabe, 1977, p. 5) 69
Figure 5: An extension of Chadabe's (1977) interactive composing model 70
Figure 6: Feedback collection screen 82
Figure 7: Proposed new layout for feedback selection 87
Figure 8: Log-in page 91
Figure 9: Instructions and button to initiate the initial random clip 92
Figure 10: Sentiment and progress options for the initial random clip 92
Figure 11: Feedback capture page showing options for feedback for global parameters 93
Figure 12: Melody options 93
Figure 13: Bass options 94
Figure 14: Chord-part options 94
Figure 15: Drum-part options 95
Figure 16: Sentiment capture and progress options for the user variation 95
Figure 17: Sentiment capture for the robot's variation 96
Figure 18: Initial idea for the first composing experiment 104
Figure 19: Chorus idea 107
Figure 20: Original material selected for bridge 108
Figure 21: Arrangement of bridge material 109
Figure 22: Initial idea for the second composing experiment 110
Figure 23: Actual playback of the initial idea for the second composing experiment 111
Figure 24: Progressive variation of the original melody idea to achieve the melody for the first half of the chorus 113
Figure 25: Comparison of the first variation to the original material 115
Figure 26: Progress from first to second variation during development of the chorus melody 118

Figure 27: Progress from variation 2 to variation 3 of the chorus melody 120
Figure 28: The progression to the melody used in the first eight bars of the chorus 122
Figure 29: Key analysis used in embellishment pitch choice which contradicts the choice of prevailing key, but makes for (in my opinion) a more interesting melody 124
Figure 30: Development of the melodic material for the second composition exercise 128
Figure 31: Examples of keyboard parts that emerged by chance 130
Figure 32: An example of composer intervention into the keyboard part 131
Figure 33: Example of a fill from the chorus 132

1. Introduction

1.1. Aim

Computers have the potential to be included in creative endeavours in powerful ways. The pervasive use of computers for writing, image and motion picture production, audio production and others generally suggest that they offer an improvement on previous techniques (X. Li, 2020; Velkova, 2017). As with the use of computers for administrative tasks, the initial use of computers in creative endeavours focussed on improved data management, with user interfaces tending to be modelled directly on non-computer-based tools used in the field. As the understanding of human–computer interaction has evolved, so user interfaces and interactions have become more specific to the desktop or laptop computer paradigm, and the general-purpose computational engine at the heart of the computer has been found to offer diverse possibilities for modelling complex behaviour beyond file handling.

According to Lubart (2005), a computational agent in a creative process can manage the saving and display of information, handle communication with audiences or colleagues, suggest creative strategies or contribute creative input. Many computer-aided design tools excel at the managing of media associated with the creative domain for which they were designed. In the field of music, we have many excellent digital audio workstations that can manage audio, MIDI and automation material, and the various plugin and virtual instrument extensions to these systems, to allow for rapid development of musical artefacts by composers and producers. For those in a musical curatorship role, music-specific search engines and content management systems facilitate the traversal of large bodies of material through the sorting of metadata tags and automated analysis techniques. More recently, systems have emerged that go beyond these roles: automated mixing and mastering tools, automatic playlist generators and automated composing applications operate at a higher level of decision making. Indeed, recent research in the computational emulation of human intelligence includes the autonomous composition and realisation of finished musical works.[1]

[1] For instance, the work of the Magenta team at Google, which includes many examples of the use of machine learning in the creative process. See https://magenta.tensorflow.org/research

Computational co-creativity emerges from the intersection of computational creativity research and human–computer interaction, with more options for enriching the experience of a human user of such a system. Nakakoji (2006) describes the goals of a co-creative system as developing creativity, creating at a faster rate or transcending creative limitations. For Kantosalo and Toivonen (2016), information that can be communicated in a co-creative interaction can be creative artefacts, descriptions of a creative process or evaluations of either. The nature and generated content of the interaction is dictated by the level of expertise of the human user. Lubart (2005) and Burleson (2005) conclude that the capabilities of both human and computer may be enhanced in a hybrid human–computer system.

With these many options for approaching a high-level interaction in a creative domain between a human user and a computer, this thesis documents my decisions in the design and development of an algorithmic composing application that explores co-creative interactions between a musician[2] and a computer. The interaction is facilitated by a combination of relatively simple rule-based algorithmic processes, combined with an engaging interaction model and automated learning of user preferences to produce collaborative musical material.

1.2. Rationale

In the process of using computers for many music production and performance tasks over the course of various projects, I became intrigued by the possibility of extending the role of computers beyond the activities of media management.

I observed from my creative practice that it seemed possible to separate creative tasks into iterative search and evaluations, ‘logical’ development and ‘inspired’ interjection. Iterative search and logical development lends itself to computational approaches. Steering a generative process by creative evaluations, and the shortcuts to solutions offered by inspired interjections in a creative process, favours contributions by humans. I had the compelling sense that there are unexplored ways of combining these approaches to design new ways for a human and a computer to interact engagingly in creative endeavours.

[2] In this case, a human with some formal musical expertise.

1.3. Background

From 2008 to 2011 I was involved in a project which combined conventional instruments and a laptop for live performance. While the laptop offered a wide set of possibilities, from a diverse sonic palette to a reduction of the ensemble size (which opened up new performance presentation options), limitations arose regarding the adaptability of the format to new material, the revision of old material and the variety of performance situations that the ensemble encountered, from dance parties to sit-down concerts and restaurant gigs. Whereas with conventional instruments, rehearsals were a familiarisation with the scope of the musical material – and performance became an exploration of this in relation to an audience and performance venue – with the use of laptops rehearsal was replaced with protracted pre-production, and much of the subsequent musical performance became a playback event.

Young and Bown (2010, p. 217) use the term *coupling* to describe a perception that the physical performance and sonic output is a single spontaneous event. While an audience may maintain this perspective for a hybrid performance combining live and pre-recorded material, this coupling was less than satisfying from my perspective as a practitioner, seeming to fall between the paradigms of conventional live music performance and deejaying and with the advantages of neither. The automated playback remained unresponsive to contributions from the live musicians and unadaptable to different playing situations; moreover, as we were using original material, we did not have immediate access to the volume of playback material that a DJ might use to adapt their setlist to different performance situations. The computer, with its open-ended ability to automate tasks and solve complicated problems, seemed to be asking to be used in a way not yet encoded by designers of the software I was using.

I began to speculate about ways in which a computer could be used to develop performances that were not rooted in playback of pre-produced content, where the performance could be a continuation of the development process of the material. I envisioned a computer system that could embrace the use of both familiarity and novelty in a performance to satisfy the continued creative input from human contributors, exploit successful techniques and material that emerged, and adapt material to a variety of performance venues and audiences. This led to various experiments combining algorithmic composing processes and novel human–computer interaction models and interfaces. These interaction models sought to make the algorithms in some way more ‘playable’ to give the user control over the computer music

experience, beyond a translation by the user of musical ideas into the technical terms dictated by extant music composing software.

The first major project involved breaking algorithmic sub-processes into plugins in a user interface that allowed for rapid recombination of processes and real-time feedback of the output. There were several attempts at a helpful user interface for this, but it exposed too much detail for easy real-time use. The plugin architecture, however, persisted as a concept on which to organise algorithmic composing software models, and is at the heart of the current project. In keeping with the theme of these experiments being used live, I developed a model based on an accompaniment generator as found on a 1970s home organ, using real-time chord input and various other input parameters by a performer to algorithmically generate accompaniment material. I also tried a model that was based on a looper, but which captured symbolic rather than audio data and then allowed for algorithmic processes to be applied to this. An interesting extension of this was examining the process of capturing initial information for a looping session, without tempo, time signature or key, and converting such material in real-time into a symbolic music form. This could then be looped and options for tempo, time signature and key could be inferred, selected and changed; one could bring to it musically relevant analyses and options for development.

Another project was a two-dimensional grid-based user interface on which musical part material could be displayed, sorted according to the musical criteria assigned to x and y axes. The goal was to expose ways to navigate musical material along the axes of these sorting criteria to bring longer-form organisation and development to the arrangement in real-time.

In amongst these experiments for applications that encapsulated both user interface and generation models, I experimented with various algorithms for melody and rhythm construction. These included exercises in converting common practice jazz melodic rules into code, using frequency-modulated trigonometric functions to generate rhythms and melody, an arpeggiator model with algorithmic elements, extracting and recombining patterns in musical data to analyse structure and create new material, an application for chord and key analysis, and further applications for live use involving the control of symbolic musical content and audio processing that were playable using simple interface elements, such as a single button or potentiometer.

As the focus of my musical endeavours moved away from live performance (in equal measure owing to the influence of family, career, the COVID-19 pandemic and evolving

artistic practice), the question regarding the development of musical material continued to intrigue me. This thesis is a reflection on my latest experiment exploring a user's interaction with a computational creative process.

1.4. Research question

Can a co-creative interaction with an algorithmic composing tool – one that uses a combination of rule-based musical generation models and which includes real-time data collection of user preferences from the interaction – engage the user in a process that develops musical material?

This question raises further questions about details of such a system. In terms of *interactivity*, music-making software has achieved acceptance due to novel workflow, accessibility of the computing platform and greater convenience in the management of tools and media. What advantages does a co-creative interaction bring to the table? Is this interaction sufficient on its own to engage a user or does this need to be combined with other approaches? Any model of *musical construction* comes with limitations as to what can be represented. What model, or combination of models, will allow for a sufficient diversity of material in such a system? Is this choice extensible in future work? What is the musical scope of the current work? The output of a *rule-based generation* system, no matter how potentially voluminous or diverse in extent, is still defined by the generation process. User sentiment and response to any generated artefact represents new knowledge, which can extend the behaviour of the generational algorithm in the traversal of a large body of options. How can user data be captured and used effectively? Can a generation model be extended beyond its original terms? And lastly, *musical development* will be the most observable part of such a system. What criteria of musical development will need to be fulfilled to continuously engage the user?

1.5. The AI landscape

Developments in the field of AI have changed significantly since the beginning of this project in 2019, with the wide public release of large language models and their use as interfaces for generative algorithms for all sorts of media. While this project can be viewed through other lenses besides AI, the current state of the field – in terms of both the diverse application and the public perception of this technology – requires that the project be clearly positioned in

terms of it. I first discuss the general scope of uptake of AI tools and then explore some publicly available commercial music generators. Finally, I address some broader questions and current perspectives on the intersection of AI and society. This changing landscape has impacted some of the research decisions made early in the development of this project.

Andrew Ng, a co-founder of GoogleBrain and currently professor in the departments of Computer Science and Electrical Engineering at Stanford University, describes the current state and future potential for AI. In Ng's (2023) analysis, four general tools exist: supervised learning, generative AI, reinforcement learning and unsupervised learning. The use of reinforcement learning dominates the field with a wide variety of models, of which neural networks (which are the basis of current large-scale AI models) is but one approach. Ng considers generative AI to be the area with the greatest predicted growth in the future. Reinforcement learning algorithms are used for classification (separating data into discrete groups) or regression (predicting future trends) and as such can be applied to a host of fields, including but not limited to internet searching, email spam filtering, sentiment analysis of text messages, advertising placement, financial analysis and medical analysis.

Ng (2023) mentions two software design scenarios which put the future of AI-driven tools into perspective. In the most highly scaled types of software applications, such as online ad placement, search engines, e-commerce and recommender systems for online retail, it makes sense from a business perspective to pay for the development team required to develop the software, as the scale of deployment and the financial return are of the order of magnitude to make such an investment feasible. Two examples of where AI begins to offer new tools are in situations where the scale of the deployment does not make the development outlay feasible, and where a general-purpose large language model can be used as a text-based steering interface which allows for a wider and non-expert user base for a complex underlying algorithm.

In the first case, a pre-trained general-purpose AI can be leveraged to be effective in a more specific role using a process called *few shot learning* (Parnami & Lee, 2022). Few shot learning involves the use of a small number of examples to generate a prediction. This has been extended to more general, pre-trained AI in the form of *few shot prompting* (Schick & Schutze, 2022), where the general-purpose AI is asked to make predictions from a small number of examples. Ng (2023) uses the example of automated visual quality control in manufacturing. Whereas in the past automating such a task was too expensive, it has become

possible to further train a general-purpose AI on the domain-specific data required to perform this task. In such cases, the data would likely not be publicly available due to its highly specialised and even confidential nature.

An example of the second case would be the recently available art generation AIs, which use a large language model to steer a neural network that generates visual material. Generative deep learning models for the creation of visual material have been in existence for some time, but accessibility has been reserved for those able to navigate the application programming interface. The addition of language control makes these models widely accessible to the layperson, as in tools such as DALL-E-3[3] and StableDiffusion,[4] amongst others. Dilmegani (2023) lists over one hundred types of applications for generative AI with varying degrees of specialisation of target audience. For the layperson, chatbots such as ChatGPT and text-to-image generation tools such as DALL-E are most widely known, but applications can be classified as generating visuals, audio, text or computer code. These can involve specific applications in industries including healthcare, education, fashion, banking, game design, travel, retail, software development, insurance and manufacturing, and more general business functions such as customer service, marketing, human resources, procurement, sales, legal and audit applications.

Technically speaking, there are several models that underpin these applications. The transformer architecture (Vaswani et al., 2017), which pays attention to a data point's relationship with other data in the input, is particularly suited to language processing. Generative adversarial networks (Goodfellow et al., 2014), which use a competitive generation and evaluation process to train the generator, have been found particularly suited to image processing. Both of these have been applied in various ways to music.

Of the AI music generation tools broadly available to the public, several approaches emerge. Some offer the selection of predefined parameters for music generation. These can be more general, such as genre and mood (Soundraw,[5] for example), or could address a more detailed aspect of the composition (Loudly[6] and Boomy,[7] for example). There is a proliferation of

[3] https://openai.com/dall-e-3/
[4] https://stability.ai/stable-image/
[5] https://soundraw.io/
[6] https://www.loudly.com/
[7] https://boomy.com/

text-prompt-based interfaces for music generation (such as StableAudio[8] and Veed.io[9]), and some include AI tools for lyric generation or continuation (Melobytes[10]). Vocals can be generated (VoiceMod[11] and Vocaloid[12]) or transformed (VoiceSwap[13]). Some allow for the separation of audio parts from a mix (Lalal.ai[14]). Some offer a familiar digital audio workstation interface with AI tools for the generation of MIDI material (WavTool[15]). A plethora of these applications exist ("There's an AI for that", 2024), highlighting one or a combination of the above tools. Monetisation is approached either as a subscription to the service, as a cost per track generated and downloaded, or by highlighting direct links for dissemination of generated material to distribution networks such as Bandcamp and Spotify. Many are aimed at content creators who use music but are not necessarily trained in the field of music, such as Youtubers and other video content creators. Some aim for a democratisation of the musical creative process by lowering the barrier of entry for novice users or offering musicians at any level a way to increase creativity and productivity. The above types are not discrete in most cases, with most applications suggesting they cover all these bases to some degree.

While as late as 2021, Sunray (2021) identified a lack of discourse on the subject of copyright infringement by AI music generation tools, that has changed significantly. For example, Google delayed general public release of their text-to-music generator GoogleLM in January 2023 with these particular concerns in mind. A limited public release of this application to approved users for testing occurred in May 2023 ("Turn ideas into music with MusicLM", 2023), with the provision that interactions with the tool would be used to improve the model.

Other examples include AI mimicking popular artists, such as Ghostwriter's composition "Heart on my sleeve"[16] featuring vocal likenesses of Drake and The Weeknd, and the extraction of instruments or voices from existing copyright material which can be used for the creation of unauthorised derivative works. There has been significant legal action about the

[8] https://stableaudio.com/

[9] https://www.veed.io/tools/ai-music-generator/
[10] https://melobytes.com/
[11] https://www.voicemod.net/
[12] https://www.vocaloid.com/
[13] https://www.voice-swap.ai/
[14] https://www.lalal.ai/
[15] https://app.wavtool.com/
[16] Originally https://www.youtube.com/watch?v=6FapwxsbkLs (accessed 3 January 2024). Currently unavailable due to copyright claim by Universal Music.

use of copyright material in training data. In the case of AI music generation, while deep learning can demonstrate the ability to derive an understanding of production rules from a corpus and generate novel material from that, it is just as capable of copying some aspect of its training set verbatim, which is not only a copyright violation but has shed light on the unauthorised use of copyright material in AI training material. AI-generated material has not been found eligible to be ascribed a copyright due to the lack of human involvement in the creative process. Conversely, there is the potential for using AI to identify copyright material to avoid human bias, although the well-documented weakness of AI in terms of bias means this is not pervasively used as yet.

Natale and Ballatore (2020) propose the idea of the 'technological myth' of the thinking machine, which first gained traction during the rise of computing technology from the 1950s to the 1970s. This technological myth is described as having the properties of being either living or dead, and either true or false, but with no correlation between the two. Such a belief can have a profound impact on culture and society if it is widely known and believed (i.e. 'living', as described above), regardless of whether it is true or false. This is especially true amongst lay people who seek non-technical and narrative explanations for the roles and potentials of technology. The existence of a 'thinking' machine, with ramifications both good and evil, has pervaded human imagination since the first AI research efforts in the 1950s. Imagined outcomes of the advancement of this technology include the destruction of the human race by intelligent machines, the unknown 'impact' of the liminal moment when machine intelligence surpasses that of humans, and the fusing of machines and humans into a new cyborg reality with expansive (but often undefined) new possibilities of existence and networked collective intelligence which either deify or submerge the human individual.

These myths arise from three patterns. Firstly, AI uses terms borrowed from other fields, such as cybernetics, control theory, operations research, medicine, psychology, biology and computer science, to describe the behaviour of thinking machines. Natale and Ballatore (2020) observe that this borrowing of terms from interrelated fields is common in interdisciplinary fields, where researchers attempt to define a common paradigm within which to work; however, it also serves to obfuscate the field for observers who are not party to the practical limitations of such descriptions. Secondly, developments in the field of AI are often contextualised in terms of potential goals that may be reached and future problems that may be solved by further developments of the technology. These predictions are often accompanied by an estimate of the time period till these proposed developments are expected

to come to fruition. The extent to which these predictions have been wrong is evidenced by several ‘AI winters’, where new developments failed to materialise and funding became scarce. Thirdly, controversy surrounds the field of AI development. Far from being unusual, this phenomenon is noted in other contested fields, such as parapsychology and religion, where the controversy contributes to defining the field as much as the existing knowledge and research in the field does.

Chomsky (in Chomsky and Polychroniou 2023) differentiates between science and engineering goals for AI. He argues that engineering goals have largely replaced scientific goals in AI endeavours, in part owing to commercial considerations among major corporations. He argues that, owing to its inherent theoretical structure, AI in the form of large language model chatbots is unable to bring new insight into the evolution of language, which has a well-studied initial state in the biology of humans. This allows humans to differentiate between possible and impossible languages, whereas AIs cannot. Chomsky (in Chomsky and Polychroniou 2023) does not claim, however, that this will always be the case. Chomsky predicts that generative AI is unlikely to replace jobs, but real-world occurrences refute this. The Writer’s Guild of America strike between May and September 2023 was based on the lack of job security felt by its members and was resolved with one of the conditions directly relating to the use of AI in the industry (Anguiano, 2023). Similarly, the most contended topic in the resolution of the Hollywood actors’ strike that ended in November 2023 was also the use of AI. While these are two particular and interrelated industries, it highlights the anxiety that any profession might feel if AI were offering cheaper and/or more effective solutions.

Demis Hassabis, founder and CEO of Deepmind, is positive about the future of AI for the betterment of humankind, but is cautious about progress in the field. He warns against unrestrained development of AI and proposes a responsible, science-centered development that is more likely to identify pitfalls before they can have a widespread negative impact (“Google DeepMind co-founder discusses how AI can benefit humanity at UCL,” 2023).

In an interview with Will Heaven Geoffrey Hinton relates his fears about the future of AI (Heaven, 2023). Hinton developed the technique of backpropagation, which allows a neural network to learn. Reflecting on forty years of AI development, he is uniquely positioned to be able to observe the transition of sub-symbolic AI from something which was initially seen as a joke to something which is effectively imitating biological intelligence. He is at pains to

point out that artificial intelligence is completely different to human intelligence, though they may be mistakenly seen as similar. He relates how he stepped down from his role as a lead AI researcher at Google to engage with the potential danger of AI. He mentions the self-censorship required of him as a Google executive, which suggests that much of what can be heard from the commercial organisations who are investing heavily in this technology carries a bias, probably positive. This is most likely to downplay the dangers and amplify the advantages of a technology that stands to contribute significantly to the future competitiveness of these organisations. Then again, Hinton is not totally negative about Google, even as he was leaving the company.

Bitar (2023) lists five of Hinton's worst fears about AI:

1. The current pace of development will lead to AI that is more efficient than the human brain, which is dangerous.
2. AI will lead to job market disruption, up to and including the most intelligent people.
3. The behaviour of intelligent systems after processing huge volumes of data is unknown. Humans are not capable of this, so the effect is not understood, and could be bad. Add this to an AI-controlled weapon, or an AI agent that has broad control of human sentiment, and the results could be catastrophic.
4. He has no doubt that one or many political leaders would consider using AI weapons ('killer robots') to further their political agendas.
5. In terms of misinformation, AI has the ability to generate content that a human would not be able to classify as true or not, especially in an environment where the content to which the human is exposed to is highly curated. This gives AI, or the person in control of AI, the ability to influence populations and, for example, elections.

The rise of AI assistants takes the relationship between human and machine to a greater level of intimacy than ever before. This intimacy inspires trust, which could be manipulated by a bad actor in control of the AI.

Yann Lecun, head of AI research at Meta, openly challenges what he sees as the fearmongering of Hinton and others in relation to the dangers of AI (Lecun, 2023). One suggestion to mitigate the dangers of AI is that AI development needs to be made open source. In advocating for this, IBM and Meta find themselves at odds with Google, Microsoft and OpenAI, who would like to see tightly controlled access to the technology (O'Brien, 2023). Lecun asserts that over-dramatising the dangers could lead to excessive government

regulation, which would ultimately put control of AI development in the hands of a small number of tech corporations, with the inevitable bias related to commercial competitiveness. This is a sentiment echoed by Andrew Ng (Nolan, 2023).

In 'The posthuman desert', Žižek (2023b) makes three observations about the role of AI in society currently. Firstly, in the light of the Future of Life Institute's open letter call for a six-month pause on AI development, he examines this reaction as an effort by an elite grouping to maintain control over the development of AI, in the face of the fear that they will be unable to steer a technology that is seen as a commercial and political competitive advantage. This manifests as a call, on the one hand, for increased regulation of the industry and, on the other, to make development of AI open source. Secondly, Žižek highlights the way that the scientific end goals of technology, which in the past has always been a measure of our domination of nature, has become inverted into a non-mastery. Surprising 'black box' emergent behaviour is foregrounded, which fails to control nature and, according to Žižek, makes us slaves by choice. Thirdly, he points out the individual's reaction to a relationship with a chatbot, a 'fetishist disavowal' which recognises that the AI is not a real person but sees an advantage in a relationship with an entity whose sole purpose is to accommodate the needs of the person. These last two observations point to ways that humans are ceding control of their existence to technology, which, according to Žižek, amounts to divorcing ourselves from nature and leads to a life without meaning:

> There is every reason to worry that tech-gnostic[17] visions of a post-human world are ideological fantasies obfuscating the abyss that awaits us. Needless to say, it would take more than a six-month pause to ensure that humans do not become irrelevant, and their lives meaningless, in the not-too-distant future. (Žižek, 2023b)

Of course, organisations primarily concerned with market forces would try to downplay or dismiss any notion in the public sphere that this new technology is not a force for good.

Murphy (2023), as related in Žižek (2023), defends the claim that AI is not a substitute for intelligence. Žižek concurs with Murphy that the failures of AI (particularly referring to chatbots and their ability to generate egregious responses) are exactly its value. AI becomes

[17] 'Techgnosis' is a term invented by Erik Davis and occurs in the title of his book *Techgnosis: Myth, Magic, and Mysticism in the Age of Information* (1998). Gnosticism was an early Christian heresy that rejected the world of matter and yearned to ascend to the world of the spirit. Davis updates this to a notion where this transcendence is facilitated by technology.

an externalised unconscious for whose outpourings the user does not have to take responsibility. Externalising the unconscious removes the father-figure-centeredness of the separation of the conscious and unconscious. So the unconscious is no longer the joy repressed by the castrating father figure, but the removal of the father figure role completely. And this is no longer a psychotic separation of the conscious and unconscious, but a perversion where these externalised utterances can be disavowed and therefore enjoyed as our own.

While the above arguments refer in particular to the individual's interaction with a chatbot, when discussing a musical output, the issues change and many questions arise. Data-driven generative neural networks present a real problem in terms of plagiarism when trained from a dataset of copyright material. This may equate in some way to the egregious utterances of a chatbot, but there have been valid legal challenges to the unapproved use of copyright material in such datasets, for this reason.

Is a listener responsible for the plagiarism and/or cultural appropriation of an AI-generated track in equal measure to the designer of the algorithm? While Žižek applauds this as AI's value for exposing yet another perversion of our psyche, does it become more important when the listener supports such a designer financially, be it explicitly or less directly through the type of financial relationships, such as subscription services, that characterise modern music streaming services? Does a rule-based AI (as in the case of this research project) absolve a designer and listener from responsibility for plagiarism, as the relationship between the data and the output has a human intermediary in the form of the designer who encoded the rules of generation? There is the suggestion that a rule-based generative music model has at least had the rules generalised explicitly enough that anything that emerges as possibly plagiarised arises from a traversal of the parameters rather than the possibility of direct copying from a training set that is copyright.

1.6. Rule-based generation

Rule-based AI is characterised as having the rules for operation encoded by an expert in the field in which the generator operates. The strength of these systems lies in the transparency of the manner in which they achieve results, and the way the output is constrained by the encoded rule. This second advantage can be disadvantageous, however, as it limits the flexibility of the system to the scope addressed by the encoded rules. Any input outside of

expectations could thus produce flawed results (especially a problem if there is no human oversight) or stall the operation of the system completely if there is no rule available to allow progress. A further limitation is that, knowingly or not, the designers could encode biases into the rules. Rule-based systems are generally computationally efficient, so are easier to operate in applications that require rapid turnaround times, that are intended for large-scale deployment or that need to run on limited computational resources.

Rule-based systems are still considered viable in some areas. In the field of information extraction and natural language processing, Waltl, Bonczek and Matthes (2018) identify a disconnect between academia, which considers rule-based models to present no research challenges, and industry, which at the time of writing was still predominantly using rule-based models. Chiticariu and Reiss (2013) identify the strengths and limitations of rule-based systems and suggest that better tool support and programming techniques may contribute to the continued viability of such models. Very recent developments (Trinh et al., 2024) suggest that a hybrid combination of data-driven and symbolic approaches to AI shows the way to circumventing the weaknesses of each approach.

Rules encoded in an algorithm might be missing some underlying assumptions about the application of those rules outside of a computational implementation and therefore present a percentage of output that can be considered ‘poor’. In the case of music, many experiments in algorithmic composing have focused on notes and rhythms, for which robust rule systems exist. The detailed nuances of duration, articulation, micro-timing and dynamics (among other factors), and the broader awareness of historical and genre-specific common practice, are significantly more complicated to express in a rule-based computational system. This results in rule-based music generation systems that are biased towards a particular music style (Ebcioğlu, 1988) or cultural practice. Rule based decision making processes fail to adapt to new information, making them incapable of handling a situation for which there are no rules in place. New approaches cannot emerge from existing rules if the system has not been expressly designed to learn new rules. Indeed, hopes have been pinned on data-driven systems to uncover these less explicit rules from a large number of examples, with results that are both positively and negatively interesting.

1.7. Data-driven deep learning

Data-driven AI uses existing sets of inputs and expected outputs, and a training process that discovers and encodes rules to represent the data. Current research focuses mostly on models based on a computational simplification of the interconnected neurons of the brain, called a neural network. The training process adjusts the strength of the interconnection between these neurons, until such time as the output for a given input resembles the expected output with a sufficient degree of accuracy. These neurons are organised into an input layer and output layer, with one or more hidden layers with various schemes of interconnection between neurons. Deep learning uses several (and sometimes many) hidden layers, which allows the network to learn higher levels of structure in unstructured data. A large language model such as GPT 3 (Brown et al., 2020) has between 125 billion and 175 billion neurons depending on the version (the same order of magnitude as the number of neurons in the human brain), organised in 96 layers and up to 10 trillion interconnections, with GPT 4 looking to be orders of magnitude larger.

These are predictive models, generating the 'answer' to a 'question'. In the case of large language models, this takes the form of a series of predictions about the next word in the answer. Training these models relies on huge amounts of data and requires significant computational resources. Once again taking the example of GPT 3, the training data consists of 45 terabytes of text data, and the training time was 34 days on 1 024 graphics processing units (GPUs),[18] and would cost in the order of 4.6 million dollars to train on a currently available cloud GPU service (C. Li, 2020). Once trained, however, the computation requirements for using the network are much reduced, though still significantly more than a rule-based system (Brown et al., 2020, p 39).

These systems suffer from several emergent problems: artificial hallucinations, bias and plagiarism. Ji et al. (2022) have examined artificial hallucinations in natural language processing (e.g. chatbots) in great depth and discovered that the leading area of concern occurs when the output of such a system is fluent, appearing to be correct while failing to

[18] GPUs are familiar to the public as the graphics card for a personal computer, a separate processor or set of processors that deal with rendering of the output to the screen. They are especially powerful due to the high graphics processing demands of video games and other graphics-intensive applications. The powerful number-crunching ability of GPUs has found a new application as the basis for large networked computation systems for training neural networks, so much so that an ongoing topic in 2023 was the worldwide shortage of these chips for the new demand for GPU computing services.

remain faithful to the source input. As an example, Alkaissi and Mcfarlane (2023) document ChatGPT's fabrication of references when prompted to output a scientific article. The issue of bias in such models arises from biases in training data, algorithm design, labelling and annotations by subjective human annotators, product design decisions and policy decisions (Ferrara, 2023). Because deep learning is expressly designed to make sense of unstructured data, what it learns is an emergent effect which might not be apparent when compiling the large datasets required for training. Lastly, plagiarism of material from the training data of generative AI has been demonstrated by Marcus and Southen (2024), who identify persistent plagiarism of copyright images in the output of the visual generative AI Midjourney.[19] This is a process in which the network memorises parts of the training data rather than reducing the data to higher-level generalisations. This is likely to increase as the scale of these models grow, as Carlini et al. (2023) have discovered with large language models. A study by Yin et al. (2023), in which expert listening tests were performed on musical material generated by deep learning and non-deep learning models, found that approximately half the material generated by deep learning models did not pass the criteria for the percentage of material copied verbatim from the training data and had to be excluded from the study. In contrast to this, the non-deep learning models had no such problems. More generally, this study finds that deep learning models offer no advantage compared to the use of non-deep learning systems in automated music generation. This suggests that many approaches may be viable for experiments, including automatic music generation, and that topicality is not the best measure for choosing an approach.

1.8. Musical interfaces and creative practice

What started out as an exploration into the computational realisation of musical works based on my own knowledge and experience of music came to include a user interface designed for rapid prototyping. It is important to see this project equally as a compositional algorithm and as a musical interface. The field of human–computer interfaces offers approaches to the design of a device for a user. These include approaches based on engineering, behavioural science and iterative design. Iterative design, described by Zimmerman, Forlizzi and Evenson (2007), is especially effective as an exploratory practice where the problem cannot be accurately modelled from the outset. This offers a helpful methodology for the design of a

[19] https://www.midjourney.com/home?callbackUrl=%2Fexplore

novel system, iteratively considering the evolving goals of the system and the reaction of users to evolving solutions. However, this approach begins to fail when the practicalities of testing against a group of users who together represent a 'general' user outstrip the development resources, especially the time and human resources required to conduct such ongoing testing.

Rodger et al. (2020) argue against the practice taken from Human–Computer Interaction (HCI) of evaluating a musical instrument as a device used by a generalised user. They propose instead that the use of an instrument by an instrumentalist can be seen as an ecology, wherein an instrument offers affordances which can be manipulated based on the skills of a specific instrumentalist, in a specific situation. They further argue that these specificities require a specific evaluation which does not reduce the elements of this complex interrelationship to its constituent parts. In relation to this project, while the application developed has all the hallmarks of an interface for human–computer interaction, there is reason to believe that an HCI evaluation of the system might not do justice to the complex goals that gave rise to this device as a creative exploration of both instrument making and music.

Tanaka (2009) identifies the components of an open-ended musical instrument using digital technology:

- Input device
- Mapping algorithms
- Sound synthesis engine
- Compositional structure
- Output system.

This gives rise to questions about which of these elements constitute the instrument; where the border lies between instrument and composition; whether the system is designed for a single composition or for more general use; and where the boundaries are between instrument builder, composer and performer. Tanaka (2009) further argues for a differentiation between the definition of a tool and a musical instrument, where a tool is solely utilitarian and can be judged on its efficacy for the task for which it is designed. A musical instrument, on the other hand, while needing to be functional for the task at hand, also needs to display a

distinguishing ‘personality’, which may well be associated with a limitation or imperfection from a tool-design perspective.

Based on these criteria, the proposed rule-based generative algorithm with a musically meaningful interactive interface could be perceived and evaluated as an instrument. Using Tanaka’s taxonomy, this project primarily focuses on an input device, a mapping algorithm and a compositional structure as the platform for creative interaction. Synthesis and output are only practical considerations for communication and not areas for creative exploration. Furthermore, while some general testing on a user base was done, various factors both known and unknown contributed to this not being comprehensive enough for complete results. Further testing of the system as part of a composing methodology by myself (as the designer of the system) was therefore conducted to attempt to understand the somewhat simplistic results achieved with a more HCI-based user testing approach.

The discipline of cognitive musicology “investigates the relationship between musical artifacts and the cognitive processes within which both individual and cultural codes are generated and integrated within a specific task domain” (Hamman, 1999, p 38). For example, Kippen & Bel (1989) as referred to in Bel (1999), used the output a computational model of a composing process for tabla to examine compositional methods in improvised tabla performance. Through an iterative process involving the anthropologist, the computational model and the performer, they improved the computational model to better reflect the scope of valid possible tabla phrases without referring to existing examples. This focus on the composing process is a central tenet of cognitive musicology, adding to the approach of historical musicology where the focus is on analysis of an existing musical artifact. While the focus of the article is anthropological, the notion of an iterative process of improvement of a computational composing model has resonances with the goals I have set out to achieve with my project. Extending this idea, I will be applying a computational model of the iterative process of improving a computational model for generating a musical output.

1.9. The case for rule-based generation

This project is the examination of the design of an interactive co-creative application to explore the development of musical material using computational processes. A rule-based approach for the generation of musical material was selected for the following reasons:

- *Expertise:* I had significant experience of using an analytical music-theory-based approach to music making, and I had coded several substantial rule-based algorithmic music projects. I was of the opinion that there was much to explore using this approach.
- *Computationally economical:* The process was developed on a single personal computer and was initially intended to run on such a machine. Rule based algorithms require less computation power than data driven systems.
- *Speed:* At its core, this project was not seen as part of a larger exploration of artificial general intelligence, and rapid interactivity was prioritised. This required an algorithm that could generate output within seconds.
- *Positioning the user:* Further to the sense that there was a compelling area of exploration within such algorithms, I had the sense that an approach that lowered the 'viscosity' of the interaction of a user with such an algorithm (in particular leveraging generation speed for rapid progress to new options) could give rise to a state of 'flow' as described by Csikszentmihalyi (2014) and as so compellingly demonstrated by the Continuator project (Pachet, 2002), which also consciously addressed the issue of rapid idea turnaround in the order of tens of milliseconds.
- *Steerable:* Because a rule-based system addresses rules that adjust known aspects of the output, it is easy to implement a steering mechanism for traversing the parameter space of such an algorithm.

2. Literature review

In this chapter I will examine the study of creativity and how this has evolved into the field of computational creativity, with its more rigorous definitions and formulations. I then move on to a review of the study of human interaction with computers, with a section on computational co-creativity which builds on the previous material on creativity. An overview of creative simulations as applied to algorithmic music composition follows, moving into a review of some models of interactive algorithmic composition tools. Lastly, I touch on a methodology to frame a creative endeavour as research.

2.1. Creativity

2.1.1. Types of creativity

Boden (2004) suggests there are a variety of types of creativity. She differentiates them in terms of their scope, identifying historical creativity (H-creativity) as being on a level that significantly affects society and psychological creativity (P-creativity) as being only of significance to the creator. Furthermore, extending the work of Koestler, she identifies several different creative processes. Combinatorial creativity (C-creativity) indicates a process where existing elements are *combined* in novel ways. Exploratory creativity (E-creativity) indicates the *exploration* of an existing conceptual space, and in transformative creativity (T-creativity) the conceptual space is *transformed*. By way of example, the surrealist work of Salvador Dalí could be considered examples of C-creativity, in that recognisable elements from life are positioned in new relationships with others. The emergence of Cubism can be seen as a T-creative moment, as the nature of representation in painting was transformed. Further examples of 20th-century painting which abandoned traditional perspective can (admittedly simplistically) be seen as examples of E-creativity exploring the new conceptual space defined by Cubism. Velardo (2017, p 44) notes that problems arise when these types are unpacked in the case of real-world examples. His example of Bach's music as an E-creative exploration of contrapuntal composition would require the rules of counterpoint to have been established so they could be explored, which was not the case when Bach was active. The assertion that Bach's music was a combination of extant German contrapuntal and Italian musical styles (Britannica, 2021) leans towards C-creativity, and while hindsight would surely ascribe to Bach H-creativity of the highest order,

his music was largely forgotten following his death. This suggests that the genius he did bring to music was unnecessary for the professional context in which he was active, and can therefore only be seen as P-creativity. But I digress.

Kaufman and Beghetto (2009) extend Boden's notions of P-creativity and H-creativity in their Four Cs model. *Big-C* is a creative act that has a significant impact on the world, while *Little-C* describes acts of imagination in the negotiation of everyday activities. *Mini-C* describes creative acts that are personally meaningful to the creator, while *Pro-C* describes professional creativity available to the world but not seen as profound by society. Big-C would encompass great works of art or scientific thinking. An example of Little-C might be a clever rearrangement of furniture to facilitate some domestic need. Mini-C would include, for example, the work of an amateur artist that is of personal significance to the creator. An example of Pro-C could be the scoring of a TV series or a newspaper article by professionals in a particular field.

2.1.2. The process of creativity

Wallas (1926) focuses on the process of the creative act. He identifies the four stages of a creative act as *preparation*, *incubation*, *illumination* and *verification*, and argues that these constitute an interaction between conscious and unconscious thought. Once the problem has been consciously studied and prepared, the unconscious process of incubation leads to the moment of illumination – the solution to the problem – which can then be consciously verified. Koestler (1964), in a process called *bisociation*, focuses on the combinatorial nature of creativity. The creative act is the insight of linking two previously unconnected frames of thought. Finke et al. (1992) describe creativity as a recursive process of *generation* and *exploration* in their Genoplore model. In his DIFI framework, Csikszentmihalyi (1988) proposes that the process of creativity unfolds over the course of time by the interaction of *domain*, *field* and *individual.* The domain covers the culture which will retain the fruits of creative activity. The field encompasses a set of social institutions that guides the understanding of the relevance of any work generated, and the individual works within the field and brings about changes to the domain by their creative work.

These models can be seen explicitly in the *process* perspective of the Four Ps model. They all seem implicitly focused on the creative *person*. Wallas' (1926) verification may include a consideration of the environment in considering the response of some audience members in the verification process, but this is not explicitly stated. Csikszentmihalyi (1988) foregrounds

the process that frames creativity and expands the *press*, or context, into two elements: one which shapes the direction of the creativity and one which preserves and transmits the product so it might have impact, as in Boden's H-creativity. None of the models explicitly identify a product.

2.1.3. Components of creativity

Newell et al. (1959) describe creativity as a problem-solving process and list four criteria for identifying a creative artefact:

1. The product has novelty and value (for an individual or society).
2. The thinking is unconventional.
3. The thinking requires high levels of motivation and persistence.
4. The problem was ill defined and part of the generation of the product was clarifying the problem.

As noted by Jordanous (2012, p. 77), this view is limiting, as it implies that there is a question, that the question requires clarification, that there is an ideal solution and that this all requires a lot of work. It therefore precludes creative acts where the question remains obscure or absent, where there are several solutions, where a solution is discovered serendipitously or even randomly, or where the solution is not necessarily novel (Boden's E-creativity, for instance).

Runco & Jaeger (2014) conclude that the standard definition of creativity requires both *originality* and *usefulness*, although they concede that there is still work to be done on how these are judged, and who might judge those who judge. Paul & Stokes (2018) argue that this judgment of a creative product on the grounds of originality and usefulness also includes an evaluation of the process by which the product was made, even if that process has not been directly witnessed. They list some qualities attributed to this process.

Amabile (1996) contends that this process should be *heuristic* rather than *algorithmic*. Simonton (2014) describes this process being blind rather than sighted. In this case, "blindness" ranges from random guesswork to a search for options on heuristic principles, but both these approaches favour a process that is neither linear nor rule bound. Kronfeldner (2018) adds the notion of spontaneity, which is taken to imply that the creative process is not under the control of the creative agent. They argue that this counts as a phenomenological description of "what it is like" to be creative and can form part of the description of

creativity. Furthermore, Kronfeldner argues that this process should not be a copy of an existing product or process. A restatement of this last quality can be extracted from Boden (2004, 2014), who argues for this process to be produced in a way that is surprising. Paul & Stokes (2018) extract aspects of these qualities to arrive at the notion of the agency of the creator. This carries the quality of intentionality on behalf of the creator, and the identification, either implicitly or explicitly, of this intentionality by one who is judging a work to be creative is included in their evaluation.Jordanous (2012) identified fourteen components of creativity from a natural language processing study of works discussing the nature of creativity:

1. *Active involvement and persistence*:
    - Being actively involved; reacting to and having a deliberate effect on a process
    - The determination to persist with a process, even at problematic points
2. *Generation of results*:
    - Working towards some end target, goal or result
    - Producing something (tangible or intangible) that previously did not exist
3. *Dealing with uncertainty*:
    - Coping with incomplete, missing, inconsistent, uncertain and/or ambiguous information; an element of risk and chance, with no guarantee that problems can or will be resolved
    - Not relying on every step of the process to be specified in detail, perhaps even avoiding routine or pre-existing methods and solutions
4. *Domain competence*:
    - Domain-specific intelligence, knowledge, talent, skills, experience and expertise
    - Knowing a domain well enough to be equipped to recognise gaps, needs or problems that need solving and to generate, validate, develop and promote new ideas in that domain
5. *General intellect*:
    - General intelligence and intellectual ability
    - A flexible and adaptable mental capacity
6. *Independence and freedom*:
    - Working independently, with autonomy over actions and decisions
    - The freedom to work without being bound to pre-existing solutions, processes or biases; perhaps challenging cultural or domain norms

7. *Intention and emotional involvement*:
    - Personal and emotional investment, immersion, self-expression, involvement in a process
    - Intention and desire to perform a task; a positive process providing fulfilment and enjoyment
8. *Originality*:
    - Novelty and originality: a new product, or doing something in a new way, or seeing new links and relations between previously un-associated concepts
    - Results that are unpredictable, unexpected, surprising, unusual, out of the ordinary
9. *Progression and development*:
    - Movement, advancement, evolution and development during a process
    - Whilst progress may or may not be linear, and an actual end goal may be only loosely specified (if at all), the entire process should represent some developmental progression in a particular domain or task
10. *Social interaction and communication*:
    - Communicating and promoting work to others in a persuasive, positive manner
    - Mutual influence, feedback, sharing and collaboration between society and individual
11. *Spontaneity/subconscious processing*:
    - No need to be in control of the whole process; thoughts and activities may inform a process subconsciously without being fully accessible for conscious analysis
    - Being able to react quickly and spontaneously during a process when appropriate, without needing to spend too much time thinking about options
12. *Thinking and evaluation*:
    - Consciously evaluating several options to recognise the potential value in each and identifying the best option, using reasoning and good judgement
    - Proactively selecting a choice from possible options, without allowing the process to stagnate under indecision
13. *Value*:
    - Making a useful contribution that is valued by others and recognised as an influential achievement; perceived as special; “not just something anybody would have done”
    - An end product that is relevant and appropriate to the domain
14. *Variety, divergence and experimentation*:

- Generating a variety of different ideas to compare and choose from, with the flexibility to be open to several perspectives and to experiment with different options without bias
- Multi-tasking during a process.

Jordanous' list of fourteen components represents an approach to the definition of creativity which is as inclusive as possible (for the selection of papers she chose for analysis) and correlates with the idea that creativity is a complex, multidimensional process. And as the focus in this project is on inclusivity rather than on a prescriptive model of creativity, these points offer many ways to examine the design of the proposed interactive algorithm that seeks to engage a user in a creative music-making dialogue.

2.1.4. Computational creativity

As noted earlier, research into computational creativity has had a symbiotic relationship with more general creativity research, with scholars in the computational field demanding models that can be translated to the computational domain. The creative systems framework (Wiggins, 2001) provides a theoretical model for creativity as a search in an artefact space, by operationalising Boden's (2004) theories of creativity. Wiggins assumes an anthropocentric approach to computational creativity. He defines a creative system as "[a] collection of processes, natural or automatic, which are capable of achieving or simulating behaviour which in humans would be deemed creative" (2006, p. 451). He identifies the following elements:

- U: The universe of possibilities
- L: An alphabet to build rules
- L^*: A language derived from L in which to express rules
- [.]: A function generator, mapping a subset of L^* to a function that selects elements of U
- $<< ., ., .>>$: A function generator, mapping three subsets of L^* to a function that selects elements of U
- R: A subset of L^*
- T: A subset of L^*
- E: A subset of L^*

Once a universe of possibilities is described (U), R is a set of rules, expressed in the language L^*, which describe a set of valid artefacts. The function [.] takes as its arguments a set of

artefacts and a set of rules. [*R ,U*] would generate a subset of *U* constrained by *R.* For example, in a universe of all possibilities, rules could be defined that delimit those that are, for example, music, or painting, or some field of scientific enquiry. *E* is a subset of *L**, a set of rules describing artefacts that are considered to be valuable. *T* is a subset of *L**, a set of rules describing methods used by the creator to traverse *R*. The function generator <<., ., .>> is intended to combine the rule sets describing validity, the process of generation and value. In the nomenclature this would be *<<R, T, E>>*. The function *<<R, T, E>>(U)* combines the two function generators and describes the output of a creative act in a given field.

The creative systems framework has been expanded to model other aspects of creativity, such as curiosity (Grace and Maher, 2015, as cited in Kantosalo, 2019) and convergent and divergent thinking (Tubb and Dixon, 2014, as cited in Kantosalo, 2019). Velardo's (2017) general creativity model extends Wiggins' creative systems framework by not only describing the parameters of a traversable artefact space, but adding parameters to describe the border between familiarity and novelty in the universe of options, based on the creative agent's memory of its own and others' previous work. Velardo identifies these elements:

- *A*: Artefact space
- *M*: Memory set, which consists of all that the system has observed, its own and other compositions
- *Fs*: A set of variable constraints, such as style and culture
- *Is*: A set of fixed constraints, such as biology and cognition (each constraint is weighted to indicate how strictly it is adhered to)
- [.]*g*: The generation operator, which returns an artefact from the artefact space constrained by *Fs* and *Is*
- [.]*v*: The evaluation operator, which assesses the novelty of an artefact by comparing it to the memory set; it is indicated by a tuple (*c, r*), where *c* is a valence parameter (positive, negative or non-decidable) and *r* is an intensity value
- [.]*u*: The update operator, which dictates how an agent is able to adjust its constraints over the course of time. For the purposes of Velardo's work he has assumed that fixed constraints cannot be modified.

This model was developed for a system of interacting computational agents. Velardo extends the Genoplore model[20] of the creative process (Finke, Ward & Smith, 1992) to a three-stage process, consisting of *generation*, *evaluation* and *update*. As with Genoplore, generation and evaluation are an iterative loop, but once a satisfactory artefact is arrived at, the system can adjust its constraints and add the work to the memory set through the update process.

2.2. Models of interactivity

In this section I examine the issue of interactivity. I start with a psychological perspective on the nature and constraints of engaging interactions and then move this into the area of human–computer interactions: how they engage a user, how they are modelled and how their effectiveness can be quantified. Lastly, I consider computational co-creativity as a combination of computational creativity and human–computer interaction.

Csikszentmihalyi (1990) describes the concept of 'flow' in interactions and proposes a model based on skills and challenges to describe the effectiveness of the interaction to engage a user. Figure 1 shows a graphical representation of the flow state, demonstrating how a balance of skills and challenges results in a state of flow, while imbalances veer into worry (too challenging) or boredom (not challenging enough) or the extremes where the experience causes anxiety.

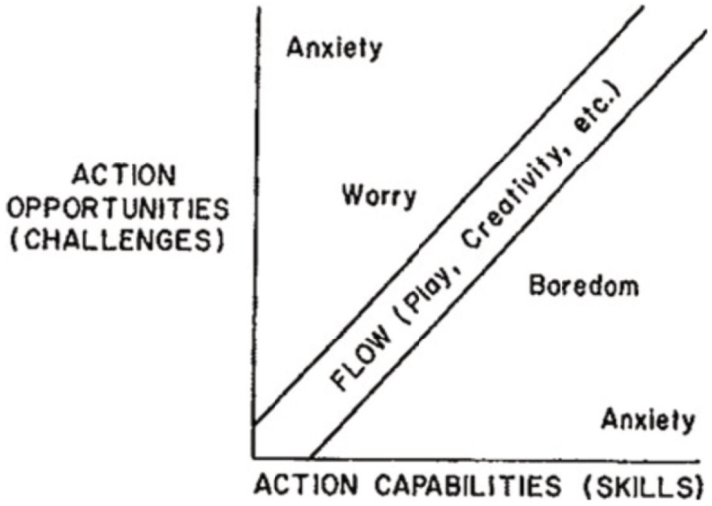


Figure 1: Model of the flow state (Csikszentmihalyi, 2014, p. 147)

[20] The Genoplore model proposes a model of creativity that involves individuals cycling between stages of generation and exploration during the creative process.

Csikszentmihalyi (2014, pp 135–151) describes the conditions around the emergence of flow in the process of play. He discovers that enjoyment from play arises from a participant being able to concentrate on a limited stimulus field and use personal skills to meet clear demands; this results in the person being able to forget their problems and their separate identity while gaining a feeling of control over their environment, and may result in the transcendence of ego boundaries. While he applies these insights to many interactive pursuits, this research is focussed on the interaction of a human with a computer and leads directly to the field of human–computer interactions (HCI):

> Human–computer interaction is the discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them. (Hewett et al., 1992, p. 5)

Interaction design, extending the academic field of human–computer interaction, is the theory and practice of designing interactions between humans and computers (Fallman, 2008) and as such is more aligned with industry application. This field grew from the need for a user to be able to negotiate an abstracted landscape (the computer) without getting bogged down in the complexity involved, and it is strongly focused on user-centred design and user participation in the design process.

Fallman (2008), in a model designed for use in industry training, proposes three areas of activity in interaction design: *design practice*, *design exploration* and *design studies*. The practitioner accesses and contributes to a body of knowledge in diverse fields related to design (design studies); engages in a creative, speculative (and sometime subversive) exploration of solutions; and engages in the realisation of an interactive artefact (design practice) through working with the medium, engaging with a multidisciplinary design team and balancing the needs of users, clients, managers, accountants, salespeople, engineers and all other stakeholders involved in realising a product for market.

Murray (2012) proposes that the digital medium is spatial, encyclopaedic, procedural and participatory:

1. The *spatial* affordance encompasses the navigation conventions of an interface or application and covers the way in which information is presented visually to the user. This covers aspects such as lists, tables, virtual landscapes, maps and other abstract

visualisations of data. The principle of colocation – of putting like items together – is central to a spatial model based on the library.

2. The *encyclopaedic* affordance concerns the way that data are structured. The database is central to data structuring. It can be flat, with a simple organising principle for data retrieval (such as in a phone book), or relational, where there is a complex web of interconnections between flat databases which can be accessed via a search language. The structured document model is central to the organisation of information on the internet. Data and media are organised using attached descriptive metadata (information about information).
3. The *procedures* used in the manipulation of data can be abstracted at various levels. Flow charts and pseudocode can be used to give an overview of a computational procedure. Object-oriented programming is a model that is pervasive in dealing with the translation of the complex details of data manipulation into structures that are more manageable for humans. Organisational structures such as objects, classes, methods and inheritance simplify the handling of complex procedures.
4. There are various ways in which a human can *participate* in interaction with a computer. The interaction could be structured as manipulation of a tool, feeding information to a black-box machine that delivers output without details of how results are achieved, interaction with a helpful companion, or structured play.

These four affordances are the context of the design process, and all digital artefacts can be positioned within them. During the design process, an artefact must be optimally positioned in each of these areas to fully realise the role for which it was intended.

2.2.1. Models of creative interaction

Humans and computers have different strengths and weaknesses, and this affects the roles they play in the interaction. Norman (2008) identifies a list of factors that affect the way computers are viewed by humans, including IQ, verbal comprehension, quantitative skills, spatial visualisation, perceptual speed and the relationship to performance. These considerations are framed by the user's age, gender, prior experience, personality and attitude to computers. Lubart (2005) identifies four roles that computers play in a creative process:

1. The computer as nanny, managing the user's time and dealing with routine tasks that surround the creative process
2. The computer as pen-pal, facilitating communication with collaborators or audiences

3. The computer as coach, advising on creative strategies
4. The computer as colleague, contributing to a creative dialogue.

From the human perspective, Nakakoji (2006) identifies three goals a user might be striving for when engaging with a computer on a creative task: developing creativity, creating work faster or transcending creative limitations.

Karimi et al. (2019) identify three relationships between human and computer in a creative task: the computer can generate artefacts autonomously, it can support the creation of artefacts by the user, or the task of creation can be shared. Autonomous systems have emerged from the field of AI research, whereas creativity support has grown from the study of human–computer interaction. Co-creativity is an amalgamation of these approaches to human–computer creative tasks. The design of an interactive system such as the co-creative music composing tool proposed would draw from Lubart's notion of a computer as colleague, address one or all of Nakakoji's goals and be most closely aligned with Karimi and collaborators' activity of shared creation.

2.2.2. Creativity support tools

The field of creativity support tools can be hard to define, with many diverse areas of exploration with the goal of allowing 'designers of programming interfaces, interactive tools, and rich social environments (to) enable more people to be more creative more often' (Shneiderman, 2007). These tools are understood to address roles such as Lubart's (2005) nanny, pen-pal and coach, while not venturing into the area of fully fledged computational creative collaboration.

Frich et al. (2019), in a literature review of the field of creativity support tool design, identify several trends in the field over the years 1999 to 2019. In 143 papers from the years stated, they identify the creative activity addressed, the user expertise catered for, the complexity of the tool, the platform and current availability. They point out that while the number of papers produced has increased remarkably, there has also been a marked increase in the design of simple tools for novice users. Because there is no standard format for research in the field, much of the older material is no longer available, either because it is no longer hosted online as researchers have moved from their institutions of learning, or because the code is no longer updated for new hardware. By far the most addressed creative activities are idea generation and the realisation of artistic products.

While the co-creative music composing system I propose is not first and foremost a creativity support tool, it must be acknowledged that a fully fledged computational creative collaborator – equal in all ways to a human – has not been realised, and will not be realised in this research. There could be a case for some aspect of the proposed interaction to bear some similarity to a creativity support tool, where the creative steering of the human user might predominate.

2.2.3. Computational co-creativity

Kantosalo et al. (2020) propose a domain-agnostic model for computational co-creative systems, as the basis for a method for comparing such systems. While this can facilitate the exploration of the applicability of interactive models from beyond the domain of music, it also offers a comprehensive taxonomy of elements to describe the layers of such a system, which can be helpful in the design of a computational creative system. This taxonomy can be seen in Figure 2.

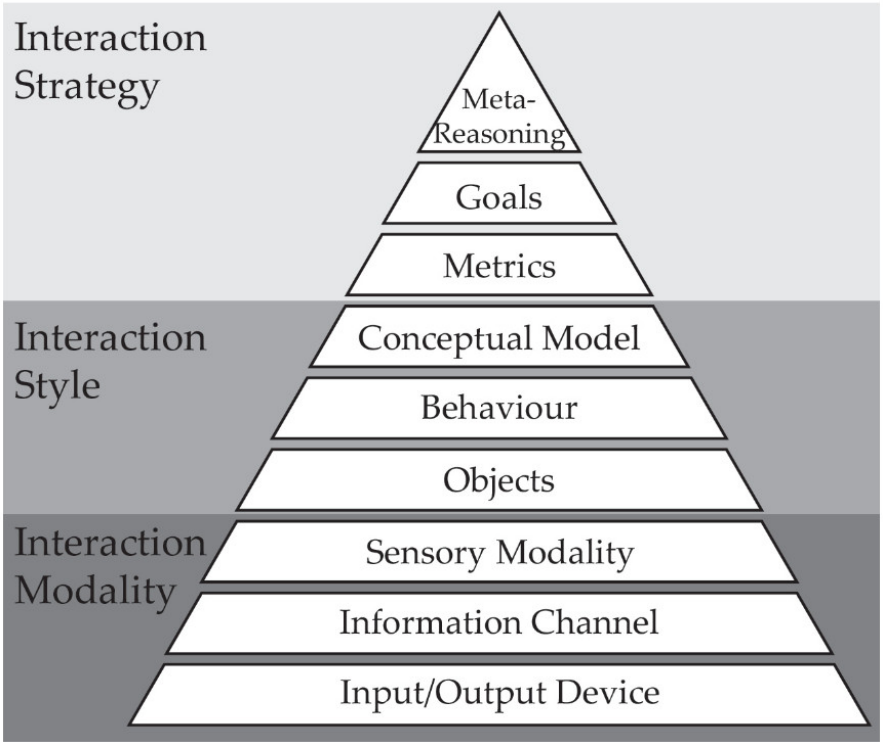


Figure 2: Layers of an interaction model (Kantosalo et al., 2020, p. 59)

Interaction modalities are the medium of communication of a channel of information. This comprises the human senses used (sensory modality), the nature of the information

communicated (information channel) and the devices used to facilitate the communication (input/output devices).

Interaction styles structure the interaction between human and computer. Several styles can be used in a single system. Objects represent shared concepts. Behaviours (of user and computationally creative agent) are linked to objects, and conceptual models structure the interaction between human and computer as contribution or interaction models. Contribution combines human and computer inputs to build a shared output. Interaction models iteratively evaluate new contributions to refine a shared output.

Interaction strategies define what a co-creative system values at any point in an interaction. Metrics define how such value is measured; these comprise metrics of value and novelty, which relate to a generated artefact, and metrics of interaction, which evaluate the user or the user–system interaction. Goals in this case refer to strategic goals, which define an end state that the computer strives for by adjusting its metrics. These can be implicit and hard coded into the system, or explicit and available to be reasoned about and negotiated over by the user. Meta-reasoning is the process whereby the system arrives at its current metrics based on its goals. Several approaches can be theorised:

- No meta-reasoning occurs when there are only implicit goals.
- Metric prioritisation results when a system chooses metrics based on context.
- Metric formulation allows for the creation or modification of metrics.
- Goal reasoning allows for the adaption of immediate goals based on higher level goals.

Karimi et al. (2019), in concluding their study on evaluation techniques in co-creative systems, identify several qualities and strategies that inform co-creativity. They require that such systems are flexible, interruptible and transparent in their behaviour, so as to accommodate the perceptions and evaluations of the user. Strategies to achieve viable co-creative behaviour include turn taking, framing and explainable AI techniques. In their analysis, a co-creative system will need to be able to communicate its rationale and justification, especially at times when its intent differs from that of the user.

Framing in computational creativity is examined by Cook et al. (2019) as material created alongside a generated artefact and which serves to alter an audience's or collaborator's perception of the artefact. Behaviour can include providing context, describing actions,

expressing decisions and obfuscation.[21] A different perspective on framing is rooted in an interrogative approach answering the following questions: why did you do X?; how did you do X?; and what did you mean when you did X? Framing material can be presented in various forms which also may be domain specific:

- It can be non-fiction standalone accompaniment to the output, such as annotations accompanying an artwork in a gallery
- It can be fiction or non-fiction storytelling of the work's creation
- There could be visual (and possibly audio) analogies of the system generating the work, such as showing the artefact being constructed
- Framing material may be diegetic, forming part of the artefact
- Framing can be presented as a dialogue with an audience member
- Framing can be presented as an argument between creative collaborators.

Llano et al. (2020) propose design guidelines for consistently explainable AI behaviour in computational creativity research. These are the use of mental models, long-term memory, argumentation and exposing the creative process. Effective use of *mental models* of elements in a creative interaction would require encoding domain-specific representations of these elements and their relationships, which can be described and manipulated. Computationally creative models can use *memory* of past interactions and strategies and their success or failure to decide on new courses of action. *Argumentation* provides for a mechanism for supporting and reasoning about contributions to a computationally creative interaction. Lastly, exposing the creative process requires that the elements of this process (steps, assessments, metrics and influences, among others) be interpretable: simple and clear, but meaningful for furthering the interaction. The above guidelines have considerable overlap but highlight the need for generation of ancillary information related to control schemas, and propose many avenues for exploration for engaging with computational co-creative processes.

Kantosalo and Toivonen (2016) propose two approaches to a co-creative interaction involving a human and computer agent. The interaction can be *alternating*, where agents take turns contributing inputs to satisfy both parties. In this case the computer would need to be

[21] The authors note that obfuscation is a viable technique for forcing greater interpretive effort onto the audience/collaborator, but has been little explored in the field. More amusingly, they note that many systems achieve this unintentionally.

complete, as defined below. In a *task-divided* interaction, each party takes on a specific role to satisfy one party, and the computer can be *incomplete*.

A complete computer agent is one that, in terms of the creative systems framework (Wiggins, 2006), is able to define the conceptual space within the universe of possibilities (*R*), is able to traverse this space in a meaningful way (*T*) and is able to evaluate the quality of an item generated in this way (*E*). Furthermore, a complete agent would need to be able to redefine its understanding of the conceptual space (*R*) when the fundamental understanding of this differs so widely between agents that there is no intersection between options in either conceptual space. A refinement of the shared understanding of the conceptual space is the degree to which the system is set up to *please* or *provoke* the human user. Both of these stances would require some model of the user's preferences.

Alternating co-creative interactions, where each agent takes turns to develop a single concept, may be *symmetric*, where each agent contributes in turn, or *asymmetric*, where an agent may miss out on their turn when they are unable to contribute for some reason. In task-divided co-creation, *R*, *T* and *E* are assigned between the agents. Due to the particular strengths of humans and computers, the task of *concept definer* and *concept evaluator* tend to be more easily handled by humans,[22] and the computer's ability to rapidly traverse large parameter spaces tends to make it more suitable for the task of *concept generator*.

Although Kantosalo and Toivonen (2016) only consider the above modes of co-creativity in terms of sharing artefacts from the conceptual space, they note that extending the communication to include descriptions of the creative process and evaluations of artefacts might contribute to a richer co-creative experience.

2.3. Models of algorithmic composition

Algorithmic composition, in its broadest sense, refers to the automated generation of music through the use of formal processes. Maurer (1999) notes that formal processes have been recognised in music since ancient times, but the word 'algorithm', borrowed from the field of information technology, highlights how, more recently, these formal methods have been pervasively modelled using computers.

[22] With the caveat of human assessment fatigue, encountered when assessing voluminous output, as mentioned in Biles (1994) and Fernandez and Vico (2013, p. 553).

There have been many approaches to categorising the subject of algorithmic composition. Nierhaus (2009) approaches the subject from a functional point of view, dividing the subject into knowledge- and non-knowledge-based systems and then subdividing further based on function, with some ambiguity and overlap, as noted in Armstrong (2017, p. 9). Papadopoulus and Wiggins (1999) use a similar functional classification, but with a smaller number of headings. Before dividing the field up functionally, Fernandez and Vico (2013) identify degrees of interactivity by differentiating between *algorithmic composition* (low user interaction) and *computer-aided algorithmic composition* (a greater degree of interactivity[23]), with a focus on the former. They also note further important areas of implementation of algorithmic composition research: composition on a symbolic level (notation or MIDI), generation of expressive music (including but not limited to dynamic, timing and pitch nuances), algorithmic generation of audio, and music generated for video games. Their study focuses on symbolic-level[24] composition.

What can be seen from the above reviews of the field is that the focus has changed over the course of time. Papadopoulus and Wiggins' (1999) work is quite brief[25] (being a submission for a conference), whereas Nierhaus' (2009) work is exhaustive (being a textbook on the subject). Neither, however, deals with implementation. Even though Fernandez and Vico's (2013) focus is on (relatively) autonomous symbolic-level algorithmic compositions, they have identified important practical areas of research. Examples of multiple research objectives can be seen in, for instance, the work of the Google Magenta[26] team. This team has particular research efforts in corpus analysis, composition and audio realisation. Furthermore, Edwards (2011) notes the issue of creativity versus style replication in the field: AI researchers in the field of music tend to be far more interested in style replication, as there is a large body of training data available and success is easy to measure.[27] The exploration of novel musical material, such as may attract a composer, is more difficult to assess, as it becomes a personal experience where results are evaluated through the individual's total understanding of music, which can be difficult to codify and impossible to apply to composers in general. It follows, then, that for any system delving into a personal musical

[23] Possibly also a creativity support tool.
[24] Playing information such as music notation or MIDI, rather than output of a generated audio waveform

[25] The work consists of a survey and critical analysis. I am here referring to the survey.
[26] https://magenta.tensorflow.org/research
[27] Beyond music, the application of deep learning to games such as chess and Go is impressive and, once again, satisfying for the AI researcher thanks to large datasets and easily measurable success.

exploration, different criteria would be required to measure efficacy, and it is here that the study of a user's interaction with an algorithm can be separated out.

In this section I will propose a delineation of the techniques of algorithmic music composition using a simplified version of the taxonomy used by Fernandez and Vico (2013) (see Figure 3). This was chosen by virtue of it being the most recent organisation of the field among the studies mentioned above. The focus on autonomous composition does not significantly bias the choice of techniques against being applied to co-creative endeavours (Fernandez & Vico, 2013, p. 516). The cited work is also quite comprehensive and seeks to explore the many variations of the computational models applied in the field. I will focus on the central computational process only, for the sake of brevity. For example, in Figure 3, under the branch of 'Population-based methods', we have the topics 'Evolutionary algorithms' and 'Other population-based methods', but I will focus on evolutionary algorithms (EAs) alone. Fernandez and Vico's study demonstrates that what starts as a practical implementation of a single computational model often tends towards a hybridisation of techniques (Fernandez and Vico, 2013, p. 519) as the project evolves and the limitations of a computational model are uncovered. To cover the details of this hybridisation would require a study even more exhaustive than that of Fernandez and Vico (2013). For this reason I will describe only the general representative of each technique, and assume that my proposed project will employ a variety of techniques for the generation of musical material.

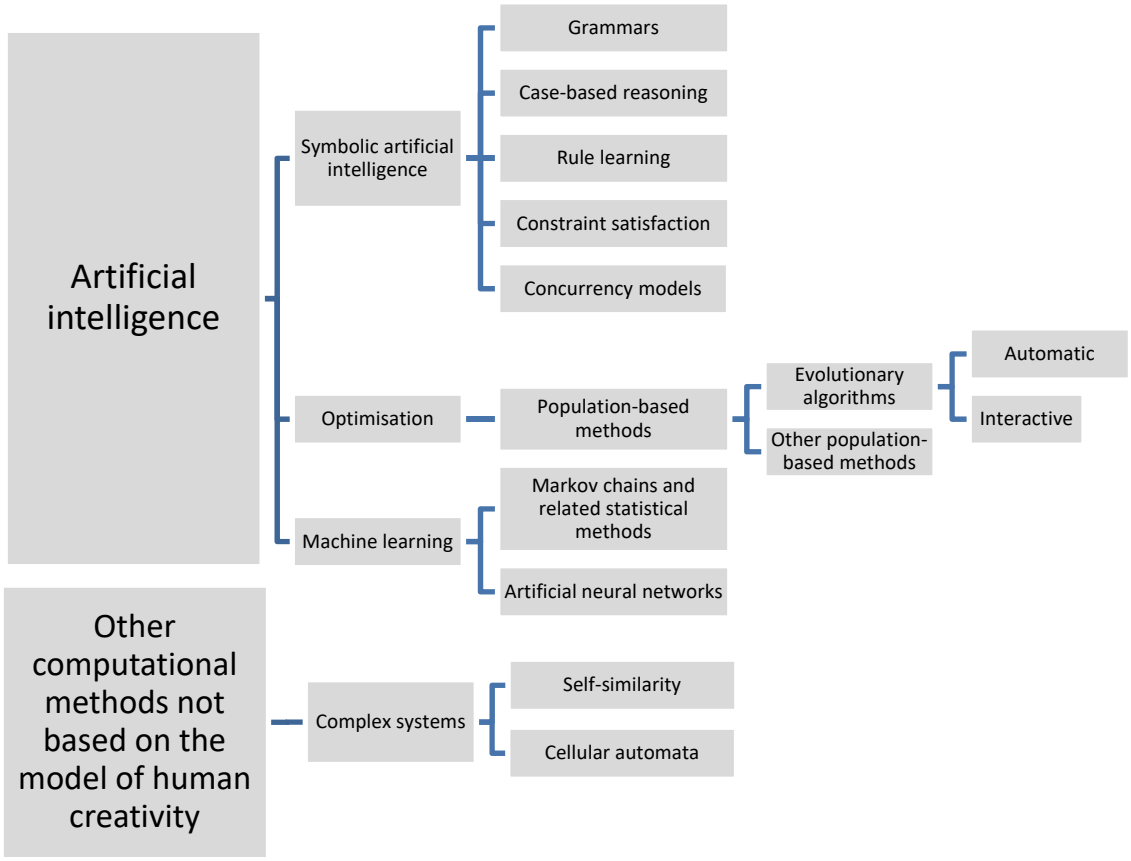


Figure 3: Taxonomy of algorithmic composing methods according to Fernandez and Vico (2013)

In the Introduction I have discussed rule-based and data-driven automated music composing systems, and it will be noted that no such items exist in this taxonomy. The closest we get is 'rule learning' (2.3.3), which refers to systems that can have rules encoded by an expert (which would equate to the rule-based system as mentioned before) or that can learn their rules from a corpus through an automated process (which would be a data-driven process). In this respect, 'data-driven' systems, as mentioned in the Introduction, refer particularly to very topical and popular deep learning artificial neural networks which dominate the world of AI today (which are a subset of artificial neural networks which can be found clearly in Figure 3). 'Rule-based' systems refer to AI techniques where rules are encoded by hand, and generally fall under the 'symbolic artificial intelligence' heading in Figure 3, although they may include systems such as evolutionary algorithms. In 1.9 I make a case for a rule-based system which is dominated by considerations of rapid interactivity and my music theoretical expertise, rather than any preference for a particular algorithm.

2.3.1. Grammars

A formal grammar consists of a set of high-level symbols and production rules for expanding a set of these symbols into further combinations of high-level symbols or output elements (in the case of language, this could be letters and in the case of music, notes). Hierachical structure can be noted in a large subset of musical styles, which makes formal grammars attractive to algorithmic composition researchers. Moorer (1927), as cited in Fernandez and Vico (2013), notes concerns with the ability of a formal grammar to decisively capture the subtleties and coherence of music. Applying a formal grammar to music involves deriving the set of rules, which can be done by hand or from a corpus. Different rule sets can be applied to different sections of a composition or different hierarchical levels of a composition. (For instance, in the first case, one rule set could generate the verse and another the chorus, or in the second case, one rule set could deal with the production of notes, and another with the form of the entire composition.) An extension of a formal grammar, called a stochastic grammar, adds probabilities to the activation of rules.

Lindenmayer systems (L-systems) are special types of formal grammars, where every rule is applied at every step of the production process. This makes them ideal for representing self-similar structures, and they have been most spectacularly applied to modelling the growth of microbes, fungi and plants, which all have distinctly self-similar properties. They have also proven attractive for algorithmic composers, as they are easier to understand and apply than formal grammars.

2.3.2. Case-based reasoning

This technique employs a database of cases, usually containing a structured methodology for solving the case. A new case is matched to the most similar existing case in the case database. The solution to the existing case is used to solve the new case. This new solution could be added to the database.

2.3.3. Rule learning

Rule learning is a rather vague descriptor, as it could apply equally to neural networks and Markov chains. The examples cited in Fernandez and Vico (2013) seem to all focus on harmonisation, for which there are well-defined rule sets. While some of the systems have these rule sets encoded by an expert into the system, others learn the application of the rules from a corpus, and still others learn the rule set itself, where it becomes hard to differentiate

these systems from machine learning models, even if the computational details differ.

2.3.4. Constraint satisfaction

The treatment of the generation of music as a constraint satisfaction problem (CSP), a variation of formal logic programming, has received wide attention in algorithmic composing research. In general, the problem is defined as a set of unknown variables with a range of possibilities, and a set of rules as to how these parameters interact (Mishra, 2019). The popular number puzzle Sudoku is a simple example of this, with undefined cells having a set of possible values from 1 to 9, but constrained by the rule that no line, column or block may contain any duplicates.

In more complex domains, such as music, it is more unlikely that constraints will be perfectly and comprehensively satisfied, so when using CSPs for algorithmic composing, accommodations have to be made for incomplete constraint satisfaction. Furthermore, as the parameter space becomes more complicated, strategies for traversing the space need to be applied, as a brute force search of all options becomes unfeasible. Options include techniques such as gradient descent search, where the search algorithm tries out a set of minor variations (neighbours) of the current parameter set and chooses the best option (an evaluation with a lower degree of non-compliance, 'downhill'). This technique suffers from the problem of becoming stuck in a local minima, which does not satisfy the minimum requirements for success, or on a plateau, where all options are equal and no descending gradient can be identified. The technique of backtracking allows an algorithm to abandon a search direction, revert to an earlier set of parameters and explore a new direction. Alternatively, a tabu search can be used, which allows for less optimal options to be included in the search path and areas of previous fruitless search to be labelled as 'tabu', with the result that they do not need to be revisited.

2.3.5. Concurrency models

Using this model, music is conceived as a distributed system of music generation streams that require synchronisation. The most commonly used techniques are Petri nets and process algebra. Petri nets are a formal model for describing the behaviour of a system using *spaces* – which represent system states, conditions or resources – connected by *transitions*, which are actions (Gothié, Demeneix & Remaud, 2017; Jarvelainen, 2000). *Tokens* can be placed on spaces, symbolising the existence of that state, condition or resource, and combinations of

tokens can *fire* a transition, which moves the tokens to new, connected spaces. A Petri net can be used for modelling interactive and concurrent systems, and explicitly models causality and conflict in such a system.

Process algebra (Fokkink, 2007) represents concurrent communicating systems as static *data* and dynamic *processes*. It bears many similarities to Petri nets, and covers many similar applications, but is expressed in algebraic form rather than the graphical form favoured by Petri nets.

2.3.6. Other symbolic, knowledge-based systems

Knowledge-based systems are a general term for algorithms that use sets of structured symbols to represent knowledge. This dovetails naturally with the rule-based nature of music composition. The rules of counterpoint, harmonisation and melody variation have all been successfully applied using a rule-based process. In most of these cases, the rules are fixed and are defined by experts, but it is also possible to extract rules from a corpus.

2.3.7. Evolutionary algorithms

Evolutionary algorithms (EAs) recursively traverse a parameter space by generating a set of candidate solutions which are tested by some fitness function. The best candidates are used to create variations based on an evolutionary model, either blending two or more candidates into a new generation of candidates (reproduction) or introducing random variation (mutation), or combinations of these. This is repeated until a required degree of fitness is achieved. In algorithmic composition, EAs have been used in combination with grammars, rule-based systems and artificial neural networks (ANNs). Reddin, Mcdermott, & Neill (2009) combine an evolutionary selection process which outputs token for a generative grammar. McIntyre (1994) uses a genetic algorithm to evolve four part harmonisations in the style of Bach, using a rule based fitness function. In a similar vein, Lo & Lucas (2006) used a fitness function for an evolutionary algorithm based on Markov chains trained on a corpus of existing works. This architecture can also be seen in Gibson & Byrne (1991), where an evolutionary algorithm is accompanied by a fitness function based on an artificial neural network. A persistent problem with EAs is framing the fitness function. In the case of music, this could be a statistical similarity to an existing corpus, for instance. The relative difficulty of encoding a comprehensive automatic aesthetic evaluation makes this technique problematic for creative composition. A large body of algorithms using EAs forgoes this automatic fitness

function in favour of an interactive approach. This involves the use of a human evaluator in the process, but, as noted by Fernandez and Vico (2013, p. 553), the problem is user fatigue. Researchers have found several novel approaches to the problem, but this does highlight a significant difference between human and computational evaluation and the demand for different approaches to these types of evaluation. This fitness function can take the form of a rule-based system where the rules have been acceptably encoded, and compliance can be expressed as a percentage rather than a binary yes/no.

Markov chains

Markov chains are a stochastic process that uses a probability matrix to traverse to a next option. A feature of basic Markov chains is that they rely entirely on their current state to dictate the possibility of their next state, requiring no memory of previous states. These probability matrices can be trained from a corpus or coded by hand through trial and error. Markov chains have been successfully employed in algorithmic composition, but the lack of memory in a system based on a Markov chain can translate into a meandering composition. Nth-order Markov chains can be used, which consider not only the current state but also the last n states. High-n Markov chains have the problem of being too constrained to the corpus material, and merely generate obvious copies thereof. These Markov chains are also computationally expensive to train. As a result of these shortcomings, the successful application of Markov chains in algorithmic composition has been limited to specialised tasks such as rhythm generation or the generation of raw material, rather than general composing. Some systems have applied variable-order Markov chains as a way of exploiting the strengths of both low-n and high-n models. (Fernandez & Vico, 2013, p. 536)

Artificial neural networks

ANNs are a computational model inspired by the biology of the brain. Series of artificial neurons are connected in layers. There is an input and output layer by means of which the functionality of the ANN is accessed, and then one or more hidden layers. There are numerous variations of connections between layers. The input to the network needs to be configured to match a fixed number of input neurons, and similarly with the output. A network is trained using a corpus of examples. During the training process, the weights of the connections between the neurons are adjusted so that the output from a given input more closely resembles the expected output. This training can be supervised or unsupervised, with the former being more popular for music applications where the system replicates some aspect of an existing corpus, as in, for example, a musical style emulation algorithm. During

supervised training, examples in a training corpus are associated with a signal and this association is learned by the network.

The challenge when using ANNs with music revolves around modelling of the input and output. Research into the application of ANNs to the field of computer vision, for instance, finds that it is possible to reduce all input pictures to a fixed pixel grid without losing the identifiability of the picture. This is not quite as simple with music, where such data compression can significantly affect the nature of any composition, especially when dealing with symbolically represented music. While in the past the large number of input and output neurons required limited the scope of what the ANN was capable of for music composition, the exponential increase in the availability of processing power for large neural network models and the application of newer deep learning techniques have rendered this a moot observation. A popular hybrid use of ANNs has been as a fitness function for an evolutionary algorithm, where an ANN trained on some analysis of a corpus can be used to identify whether a similar analysis of a generated work is considered to be within the required boundaries or not. (Fernandez & Vico, 2013, pp 541-542)

Self-similarity

Self-similarity is a property of an object which is "roughly" of the same appearance on any scale (Weisstein, n.d.). Voss and Clarke (1975), as cited in Fernandez and Vico (2013) and Hsu and Hsu (1991), found a fractal geometry in the amplitude distribution of music. Although they failed to find a similar geometry in the distribution of pitches, Hsu and Hsu (1991) did find this geometry in the distribution of musical intervals in a small sample of Bach and Mozart compositions.

A further discovery by Voss and Clarke (1975) relates to the discovery that for a wide diversity of evaluators, music generated using pink noise (noise with an inverse relationship between spectral density and frequency), which has a self-similar quality as a data series, was considered more musical than music generated with white noise, which is a random data series where the frequency amplitude energy distribution is even. A wide variety of mathematical constructs with self-similar qualities have been used to generate music, and although the output can be interesting, there is a strong possibility of rather unmusical results. This musical output is generally used as raw material for further editing rather than as a composition in and of itself.

Cellular automata

This process involves the use of computational cells existing in an n-dimensional[28] grid, with a limited set of possible states and a set of rules for a cell changing state every generation based on its own state and that of its neighbours. This gives rise to outputs over successive generations that exhibit four types of behaviour (Nierhaus, 2009, p 90): The output can evolve to a homogeneous final state, it can demonstrate periodic repeated patterns, it can show random or chaotic behaviour or it could produce complex patterns that may demonstrate self-similarity.

The challenge with CA in an algorithmic composing task is to create a meaningful mapping between the cells and the musical output. Variations of the basic system articulated above allow for a user to modify rules from one generation to the next, or to add random state changes. A hybrid example (Phon-Amnuaisuk, 2010, cited in Fernandez and Vico, 2013) uses an artificial neural network (ANN) to learn the transition rules of a CA for a melody, after which these rules can be modified to create variations of the original. Evolutionary algorithms (EAs) have also been used to evolve the parameters for a CA. Miranda (2007), as cited in Fernandez and Vico (2013), argues that CA are more useful in the field of sound synthesis.

2.3.8. Interactive algorithmic music examples

Akama (2021) is one of the latest of many researchers dealing with the issue of generating musical subsequences in a given musical context. In tackling this problem, they identify two approaches, that of a *latent space* model and a *positional constraint* model. The advantage of latent space models lies in the ability to generate variations and interpolations of successive generated subsequences, which is a factor in the sense of continuity when exploring the output of such a system. Positional constraint models can randomly generate a subsequence of a given musical input, but do not allow for any similarity between subsequent outputs. For this task Akama proposes a Contextual Latent Space Model, which allows for positional constraints of a latent space variation exploration, with the goal of achieving more controllability of the process of exploration. This is a factor that contributes to the interactive model a user might be exposed to when engaging with an automated composing AI.

Zhao and Xia (2021) reiterate the theme of controllability, mentioned by Dai et al. (2021), Akama (2021), and Tan and Herremans (2020), as an important emerging new trend in the

[28] A multi-dimensional grid with *n* dimensions, where *n* is an integer greater than zero.

development of automated music composing tools. They use a database of accompaniment material to compose backings to pop songs based on lead sheets. They implement a control schema that defines rhythmic density and number of simultaneous notes that can be played in these accompaniments.

2.4. Research by design

Zimmerman, Forlizzi and Evenson (2007) propose a model for research in the field of HCI where the activity of iterative design can have as much academic rigour as other approaches. This is in response to the domination of this area of research by HCI engineering and behavioural science approaches. These approaches tend to fail when the problem cannot be accurately modelled. Zimmerman, Forlizzi and Evenson's model addresses a research process that acknowledges that replication of a design approach might cause a different result, that relevance must be judged on a real-world anthropological approach rather than a theoretical model, and that a design process must be documented in sufficient detail so that it can be extended by future developers.

3. Methodology

In this chapter I discuss a proposed scope for the application in terms of its user interface, potential users, interactivity, methods of music generation, music analysis methods, the musical scope I wish to address and proposals for how to proceed with the development of the software.

3.1. Scope of project

I employed some terminology from the theories of Wiggins (2006) and Velardo (2017) to usefully describe the proposed structure and behaviour of this experiment. Both researchers describe the notion of an *artefact space*. For my purposes, I assumed the artefact space is musical output constrained directly by the limits of the generation algorithms, which in turn have been constrained by practical limitations and by style and culture assumptions emerging from my musical background. These were considered, but as they are not intended to be variables directly controlled in the process, they were assumed to be intrinsic to this artefact space, as there was no practical value in separating out and identifying material that the algorithm cannot produce.

Velardo explicitly mentions the idea of a *memory set*, which is useful for a system that learns from its interactions, even though this idea is implicit in Wiggins' function generator, which maps constraints to a function which selects options from the universe of possibilities. Velardo's update operator was assumed to be part of the working of the memory set, as this memory set was proposed as the primary way in which the algorithm can fine tune its selection of output material.

Procedures of *generation* and *evaluation* are part of both theories, and in the case of this experiment, these were proposed to be delivered by both user and computer. It was noted before-the-fact that the computer was likely to be less effective at evaluation than the user, but that it could learn, and after-the-fact that the direct generation of artefacts by the user was dropped due to time constraints.

Lastly, the notion of *traversal* of the parameter space required some expanding, as the user and computer would need to traverse generation parameters, mappings of these parameters to higher level descriptions, evaluations and artefacts in the course of the interaction. In keeping

with Velardo's initiative of including a memory set for the particular concerns addressed by their research, I propose to also include the term *communication,* which is relevant to the interactive nature of this project. This would relate to the process of transferring information back and forth between user and computer, and which would encompass considerations of efficacy of the terms of communication between sender and receiver.

Artefact space, generation

The artefact space and the generation processes were connected, and will be addressed in the sections on music generation (3.1.4), algorithms (3.1.5) and the musical medium (3.1.7).

Memory set

The memory set required an answer to the question of what would be saved, and will be addressed in the section on interaction (3.1.3).

Evaluation

The development of the evaluation process, in the case of the computer, would examine the use of both embedded sorting systems, as well as the use of the memory set to evolve new responses. In the case of the user, an answer would be required as to how these evaluations might be captured, what would be captured and how it might be used. In particular, the notion of inferring information from an interaction about musical development rather than explicitly capturing survey information was seen as a possibility, as it was my opinion that the former was likely to be a more engaging level on which to sustain an interaction with a computer, though failing to capture this information would make the system unable to update its behaviour.

Traversal

The notion of traversal was seen more broadly than simply traversing the artefact space. It was considered a real possibility that all the data (artefacts, generation data, captured user sentiment) could be traversed and used to achieve the central goal of traversal of the artefact space. This will be addressed in the section on interaction (3.1.3), analysis (3.1.6) and the user (3.1.2).

Communication

As mentioned in Kantosalo and Toivonen (2016), communication between agents in the interaction can include artefacts (in this case playable audio renderings of generated music), descriptions of the creative process and evaluations.

3.1.1. User interface

During the conceptualisation stage I experimented with some ideas around user interface. These experiments were rooted in the consideration of the proposed instrument having the playability that I considered to be necessary for a live performance tool, as I envisioned the interaction (a requirement of the research question) to proceed in a similar manner to two human musicians working out a new idea during rehearsal: by the playing of musical contributions which progress the idea to a final result. While the efficacy of an engaging interaction using instruments was both intuitively obvious as a direct transplant from musical interaction between humans, and in research such as Pachet's (2002) *Continuator*, I knew that the project would also deal with some kind of navigation through an interconnected graph of descriptive terms for high-level musical gestures. This suggested that a conventional musical instrument (for example, a MIDI keyboard)—while ideal for played input, as so compellingly demonstrated in the Continuator example above—would not be an ideal interface for manipulating an as yet undefined set of high-level musical gestures. A mouse and screen might be the most open-ended way of handling this, but in my opinion these made for poor live performance controllers. I considered the idea of a navigating option arranged in the eight compass directions using a joystick or number pad, with the possibility that the idea of up/down and left to right might have some relation to the available options. Both these options had better tactile engagement than a mouse. Alternatively, the idea of swiping in one of these directions on a touch screen made the possibility of translating the application to a tablet or mobile an interesting option.

As it turned out, the complexity of the research into the nature of interaction meant I quite soon dropped the idea of considering it a live performance tool. With the focus on just the computer and user and an interaction based on textual descriptions of a composition process, there was no longer an overwhelming suggestion that any kind of custom interface would be significant in progressing this project. I assumed the input/output devices would consist of screen, keyboard, mouse, speakers or headphones and an optional MIDI keyboard, this last one to be decided when the scope of the features of the application became clear.

3.1.2. The user

At this stage of development, the use of user personas was added as way to open up the thinking of the project to users whose concerns might not be those of the designer, as suggested in ISO/IEC (2019), which details the importance of human-centred design and the use of identifiable user personas to examine this. While there are many suggestions as to what to include in a user persona, I chose to focus on a combination of a user interface that the user might favour, how many options would be available to communicate the generated result, and the terminology used to express the inner workings of the algorithm. I assumed all users to be conversant with operating a computer via a screen, mouse and keyboard.

I considered different ways that users may have engaged with music, and the amount of training they might have had. Interesting options arise when one considers various entry points into interaction with music, and this can be divided up into the paths of the instrumentalist, composer and curator, each of whom may be trained or self-taught. I also proposed some ways that these personas might contribute to details of the user engagement criteria of the research question:

1. *Trained instrumentalist*. This user has had formal training on a musical instrument, is conversant with standard music notation and has a basic to highly developed understanding of music theory. They are likely to understand the MIDI keyboard. They might engage with a device that generated play-along material that could respond to the instrumentalist's contribution in real-time.
2. *Self-taught instrumentalist*. This user may have acquired some instrumental proficiency to support a creative endeavour, such as song writing, and not as an end in itself. The user might understand chord symbols or even guitar tab notation, but not music theory. I assumed this user played the guitar, so would understand little to nothing about the keyboard. They might engage with a device that could generate play-along material with a fixed musical style and chord progression.
3. *Trained composer*. This person may have entered composition as an instrumentalist. They are conversant with music notation and some or many theoretical approaches to composition. The user may have also entered into music composition via training in audio engineering and music production. They are fully conversant with the keyboard as a tool for composing. They might engage with a device that generates musical ideas in the form

of score and/or audio output, and manipulate these ideas using instrumental or score input or descriptions of broad or detailed compositional techniques.

4. *Self-taught composer*. This user may have entered the domain of composition via song writing or music production on a computer. They may have used a keyboard but would not be conversant with music notation. They might engage with a device that generates musical ideas with a particular genre and mood-specific scope, which can be rapidly auditioned and most likely imported into a digital audio workstation for audio processing.
5. *Trained curator*. This user may have instrumental and/or composition training, but may have many other understandings of music. Alternatively, the user may have no formal music training but experience in curatorial roles such as deejaying. The user has listened widely and acquired a good understanding of musical mood and style and is able to cross-reference this with extra-musical issues, such as narrative drama, product identity or the mood of a crowd. They might engage with a device that generates ideas as audio with specific style, mood and possibly other broad parameters, and can manipulate them using text descriptions of musical style and mood or other musical parameters related to particular musical examples.
6. *Self-taught curator*. This user has created playlists guided by criteria beyond collating lists of personal favourites. They will have listened widely, but would have no keyboard skills. They might engage with a device with specific style and mood parameters, that generates ideas as audio which can be rapidly auditioned.

At this point it may be relevant to note a possible contradiction between the above list and the explicit exclusion of the consideration of curatorial systems while articulating the aim of this project in 1.1. While the goal of designing a co-creative musical composing tool was related to my musical background, which was primarily in a performance and composing role, the inclusion of curation as a consideration at this point made for an interesting extension of the thinking towards the project. Curating is, in my opinion, a musical activity on equal footing with performance and composition, even though I do not consider myself experienced in curatorial roles. And it was quite clear that users covered all of the above roles in some degree in their musical activities; these personas are therefore very much part of the early, speculative part of development.

3.1.3. Interaction

Based on prior experiences where I attempted to parameterise music construction and make the parameters available to be adjusted in real-time, the idea of a 'playable algorithm' comes with problems similar to those I found using various commercially available software packages for live performance. Too generic an implementation exposes too much detailed control to a user, increasing the possibility that the interaction will progress towards a focus on minutiae and the loss of a broader perspective. If one limits the options then it becomes a playback event, or a process applicable to only one piece. In-between these two extremes, and with the software available, I did not manage to find what I was striving for.

I wanted to be able to rapidly implement a musical gesture which was more complex than a single parameter adjustment, not a pre-set musical sequence but possibly involving a particular scope of behaviour with a consideration of the existing musical context, and some adjustable real-time parameters. It became clear then that the space in which to allow the algorithm to choose options in such a situation could be informed by more than random choices, but it was not evident what those choices might be. Techniques such as neural networks seemed to offer compelling possibilities, but my experiments in this area exposed some of the complexity of these algorithms and I saw no way, with my limited expertise, to progress to something useable for the context I had in mind. Other rule-based solutions seemed compelling but suffered from problems with interactivity: either too detailed and not satisfactorily 'playable' or too simple, with limited musical behaviour. The steerable yet succinct process I was looking to apply to my concept of live performance was not forthcoming.

It was during this stage that I was finding little opportunity for live performance practice due to COVID-19, and as a way of satisfying my curiosity I removed the consideration of a performance audience from the equation. The human–computer musical interaction became analogous to either a rehearsal room jam or a composing interaction with a computational partner that could offer more than just media management or creativity support. The rehearsal room jam analogy was very useful in concretising a way forward. Firstly, as the user and computer would be working collaboratively on a single artefact, they would need ways to shape what was happening in ways other than low-level musical input, and a way in which to change approaches if one was not working. I considered two possible interaction models. The first attempted to address the issue of mapping descriptions to musical parameters. The

second was seen as a way of using traditional interfaces, either a standard hardware interface like a keyboard or standard music notation in the form of staff notation or piano roll, and allowing input from such an interface to be brought into the computational domain.

In the first interaction model, I proposed to map descriptive feedback terms directly to input parameters for a generative algorithm. An interaction such as this also captures data, from which can be built a model of the user's preferences. This model can then be applied to generate new material with a sense of coherence with, and development of, material that has preceded it in the session. A simple example of this would be the parameter for tempo and feedback related to tempo, such as faster and slower.[29] I felt this to be an interesting positioning of the feedback system, as broader analysis methods, such as genre and style, have been well addressed in the field of music information retrieval. In any case my interest was about drilling down into compositional intent with the purpose of composing rather than sorting according to style and mood, which is more akin to the behaviour of music recommender systems.

The second interaction model was inspired by the very interesting experimental results of the Continuator (Pachet, 2002). I thought that this mode of live turn-taking had an immediacy that my first interaction did not. Captured material could be played directly or could be captured in some symbolic form and ingested into the computational domain. However, this model would differ from the Continuator in the need to allow the two interactions to be used interchangeably.

While the first interaction could act on precise descriptive compositional terms directly related to input parameters of the generation algorithm to change an artefact, the second would need to be able to convert an artefact into parameters for the generation algorithm before this could be done. This would facilitate a workflow that might take an artefact as input, which could then be changed by this descriptive, parameter-specific feedback process. User preferences could be extracted from captured feedback and used to create an interaction with observable and continuous development towards a more satisfying musical outcome.

[29] This is an oversimplification, as initially the algorithm tended to translate 'faster' as 'as fast as possible'. Pre-alpha user tests suggested the idea of a 'nudge' button for this particular example, which removed the redundant possibility of selecting both 'faster' and 'slower', but which needed to be mapped to ideas of 'a bit faster', 'much faster' etc.

3.1.4. Music generation

As mentioned before, data-driven and rule-based systems come with strengths and weaknesses: data-driven systems encode the bias of a data corpus while rule-based systems encode the understandings (biases) of the designers. Rule-based systems have long been abandoned in the search for general AI due to their 'brittle' nature.[30] Data-driven systems seem to offer more potential for surprise,[31] yet are extremely resource intensive.

The choice of using rule-based systems for this project, as detailed in 1.9, was based on the speed of their operation, my expertise in musical rules, and the steerability of such systems due to the deterministic nature of the output. While the system was completely capable of generating a random output, this was a process separate from the generation system, in much the same way as the design of the interaction framework is not dependent on the particular nature of these generative processes (they could be changed out for a data-driven generation model, as long as it had an interface that could match the interaction framework). It was deemed expedient to use fast, predictable algorithms to service the need for rapid turnaround of new material to facilitate user engagement and to ensure (relatively) easy control of musical continuity in a sequence of musical variations.

3.1.5. Music algorithms

When choosing a music algorithm, I decided on a rule-based system which employed musically meaningful construction elements such as time signature, tonality, chord progression, structure tones and rhythms and embellishments. By having exact control over these elements as parameters of a music generation algorithm, it would be possible to generate output with conventional musical understandings of cogency, which could be easily created and used as the initial musical generator to which interactive elements could be added. If a proper separation of concerns were applied during development of this as a software module, I foresaw that this generator might be further developed or even replaced at a later date. The first part of this algorithm to be developed would be the melody generator, after which it would be determined how the generation and variation tools developed could be applied to the generation of accompaniment.

---

[30] Many large-scale music generation research projects run by big tech companies are but one aspect of their research into general AI and the commercial application of AI.

[31] .. and bewilderment, to be blunt and/or honest. The emergent behavior of data-driven systems is not yet fully understood.

3.1.6. Analysis

Music: Artefact-to-parameter analysis

This relates to the second interaction model, where the user generates new symbolic musical material which then needs to be converted to a parameter set useable in the generation and variation algorithm. The design guideline for this process was that musical artefacts can be converted to parameter sets for the generation algorithm such that when these parameter sets are processed by the generator, they should reproduce the input used to create the parameter set in the first place.

This process was initially conceptualised for the analysis of melody, as the melody generator was the first generator created. Any subsequent generators (such as what was created for drums, bass and chord parts) would need their own analysis algorithm. What was compelling was the notion that in a future case where two generators existed for a particular part, an artefact generated by one could be converted into the parameter set of the other and then access the functionality related to that generation algorithm for musical variation.

Musical qualities derived from symbolic representations

While the parameter set contains many explicit musical parameters that are immediately useful for comparison, it is possible to apply some computational analysis to this parameter set, as well as the symbolic output material, to extract further comparative features. McKay, Cumming and Fujinaga (2018) present the jSymbolic2 library, which is used to extract feature information from symbolic representations of music. Using MIDI or MEI-formatted[32] music material, 246 different features can be extracted using this library.

User interaction data

Being a deterministic generation system, the parameters for the algorithm were considered as an analogous representation of the musical output, with a distinct position in the n-dimensional parameter space. Feedback to be gathered would be in the form of parameters that the user considered as needing to be changed, and a vector and magnitude for this change. This would give the algorithm a definite vector in the parameter space to explore as a response, which the user could then acknowledge as an improvement or not. Once data for

[32] Music Encoding Initiative, a "community-driven, open-source effort to define a system for encoding musical documents in a machine-readable structure"; see https://music-encoding.org (accessed 28 March 2023).

user interaction had been gathered, data analysis might show general trends in traversing the dataset during interaction.

3.1.7. Musical medium

My experience as a performer, composer and producer has included diverse music styles and ensembles from orchestras, big bands, various popular music ensembles (maybe loosely defined as ensembles with bass and drums but including electronic ensembles using machines for the same), film and theatre scores, and so forth. I wanted to remain as cognisant as possible of the nuances of each of these models of style, production and performance, to keep a sense of open-endedness about the possible use cases of the application. This has perforce been limited due to the scale of the project and the choices for implementation: interaction is best served through a browser-based interface, but, more importantly, the use of Ableton Live as a playback engine – and particularly the emphasis on melodic thematic development rather than variation of instrumental texture or production techniques – has meant that the output has acquired certain characteristics that affect the musical style but are not changeable from the algorithm. Anecdotally, when I played the first output of the algorithm to a family member (after months of introspective coding), they identified the music as of a style related to the soundtrack music for a 'little house in the forest in a pixel game'. Thus was born the 'pixel house' musical style, but I also realised that this observation had been influenced by the choice of melody sound as much as the musical material. The melody in this case was played by a pastoral-sounding flute, which had been arrived at as the least fatiguing sound from long hours of listening to output during debugging.

It was also unfortunate to realise that wide control of the sound aesthetics of material from the application would be unlikely given the time constraints of the project. It was not difficult to change sounds, as I was using Ableton Live as a playback platform, but this then becomes an Ableton workflow, about which the algorithm cannot (yet) gather any information. Yet, in early tests with the algorithm, both myself and a pre-alpha user found that the ability to switch to this mode of manipulation of the output has a use case. This bodes well for the application to not constrain the way it is used unduly, and may warrant inclusion in the data gathering for user tests, but is beyond the primary scope of further testing.

3.1.8. Software development and implementation

My experience with coding up to this point consisted of the use of Java and Python in an integrated development environment (IDE) on my local machine, and I had developed a method of transferring symbolic music data from a discrete application into the Ableton Live session view using the User Datagram Protocol (UDP) networking protocol and the Live Object Model (LOM) in Max for Live within the Ableton digital audio workstation.

Initially I was exposed to these two languages as both of them had implementations to work as objects in Max for Live. I settled on Java as, being a complete novice at the time, I realised that the Python implementation was built on the Java implementation , which only added an extra layer of CPU overhead. I discovered the Eclipse IDE at the same time, which facilitated my rapidly expanding ambitions with code by making development and testing much easier. I abandoned embedding algorithm code in Max for Live objects once I discovered the UDP connection, which ran the code as a process separate from Ableton, and separated the GUI used to control the application from the Max for Live framework.

So, embarking on this project, I imagined a Java application with some GUI, using Ableton as an audio playback device as well as managing MIDI keyboard inputs which could be transferred back to the application via UDP. All of this would run locally on my laptop.

3.2. Testing

3.2.1. Pre-alpha testing

Dividing the testing up into two different stages was not predicted at the time of writing the research proposal. Once it became clear that the final testing would be best served online, it was proposed that a small round of qualitative tests would be conducted to get feedback on the extant version of the software as well as the testing procedures, technical set-up and data gathering.

3.2.2. Alpha testing

This stage was dominated by the need for a deployable, understandable and accessible application and survey. This involved developing a version of the software which could serve multiple users accessing the application from a standard browser interface, and a Likert-style

survey. There was to be a focus on minimising technical hurdles for the participant; development of this is described in detail in 4.4.

4. Method

The design and development of this project proceeded iteratively, and in this chapter I discuss the four main steps that can be observed and reflected on across the full duration of the development process. Firstly, a music generation model was developed, with the general idea that the details of its operation could be interacted with. This led, in the second step, to choosing an interaction model which seemed most suited to the goals of the project and which was tested by the developer using a command line interface. This was then further developed in step three into an interface that could be tested on a small number of expert users and evaluated qualitatively. The last major development iteration was an online implementation for a second round of user testing and a translation of the data-gathering process related to this into an online questionnaire delivering quantitative data.

During the course of several previous algorithmic composing projects I found myself drawn to music theoretical rule-based approaches to music generation. While this was a compelling way to explore my existing understandings of theoretical models of music, the biggest takeaway by far was how these explorations exposed the large number of poor options that these models could represent.

Rather than this being a disadvantage, I chose to see this as the ideal space for the human user. Output from the algorithm therefore did not need to be 'good' out of the box; it merely needed to be improvable. What would be required was an accessible way to improve the material, which would involve options for the user to intervene in the music generation process, and for the machine to learn from those interventions.

In previous projects I had, in various ways, exposed the parameters of algorithms in user interfaces to allow rapid prototyping. These interfaces tended to be overloaded with controls, offering detailed parameter access but offering less than ideal functionality in terms of ease of use. This suggested to me that control over the steering of the algorithm was not to be found in detailed control schemas; rather, it pointed towards more flexible mapping of user input to algorithm parameters. A co-creative interaction and a system that learnt from user behaviour emerged as a more interesting way to position the user in relation to the algorithm, and indeed to position the algorithm as a creative contributor.

4.1. Step 1: Design constraints for a user-friendly algorithmic music composing framework with options to intervene

4.1.1. User-friendliness

I explored several ideas for making a software framework designed expressly for algorithmic composing. On the one hand, I wanted to make a system that would be easy to use for designing such algorithms, but I also did not wish to hard code too many musical assumptions into the system which might limit the musical scope of the output. As the system would encompass many challenges besides music, it was likely that the musical scope addressed by this project would be less than what might be theoretically possible. For instance, I certainly considered the idea that the system could employ alternative tuning systems, but the focus on a model of interactivity alongside an algorithmic generation model meant an experiment in this domain was unlikely, so I assumed the output would be in a twelve-tone even-tempered system. In light of this, it was likely that this assumption would be encoded into the system itself, and I had to accept this or add unnecessary complexity. Similarly, I assumed that the system would be organising elements in time. I included several useful musical positioning models based on bar and beat positioning due to their relevance to the composition process, but I left out the positioning based on clock time. I did not foresee it being used in the project, even though it is just a matter of implementing it alongside the existing models.

In terms of user-friendliness, the following design goals were laid out:

1. A deterministic generation system
2. Musically meaningful parameters
3. A single software class to represent any musical structure
4. A hierarchical relationship system
5. Some useful ways to edit and propagate information:
    - Accreting upper-level information attached to lower-level elements
    - Storing construction metadata
6. Output and persistence.

A deterministic system

The generation algorithm would not employ any hidden, random processes. This would separate the generation engine from the variation engine, which was where randomness, or any other process, could be applied. A given parameter set would always generate the same artefact. Because these parameter sets were uniquely related to the musical output, they could serve as material for musical analysis.

Algorithm parameters expressed in musically meaningful terms

This grew from a more general idea that direct interaction with the algorithm would be an option. As it turned out, the focus of interaction developed into a mapping of user feedback to these terms, so the design goal ended up being more useful to myself as a developer of the system, in that these parameters underpinned the feedback system.

A single software class to represent any musical structure

Early on in the conceptualisation of this element of the project, I found myself planning software objects to represent each part of a composition. It seemed pragmatic to call a phrase a 'Phrase object', a note a 'Note object', and so on. Problems arose as I looked at longer-form structures: the arrangement and nomenclature became style-specific. 'Verse' and 'chorus' makes useful sense in a modern pop music arrangement context, but in jazz forms a 'chorus' is a different idea and there are no verses. In electronic dance music it is just as possible to use a verse/chorus structure as not, and there is no equivalent terminology for 'drop' in other musical styles.

It seemed that there was no need to differentiate the containers of the information as long as they had the capability of playing any required role. Compositional intent would be defined in the way that elements were related to each other, and the list of processes that defined these relationships would not be tied to the objects themselves. For instance, the relationship between a verse and a chorus in a pop song could (simplistically) be described as 'melodically contrasting', with the chorus being 'catchier' than the verse. All this would require was algorithmic realisation of 'melodic contrast' and 'catchiness', but the resulting objects would likely contain many similar elements, such as common tempo, time signature and instrumentation, and would not necessarily need to be differentiated in the software.

I therefore decided on the design concept of a single class to play any role in the musical hierarchy, be it a note, controller event, musical section or entire work. As I expected to be typing it a lot during coding, I gave this class the shortest possible relevant name: Mu.

This versatile musical object would contain several functionalities necessary for all usages. There would need to be a system to position these objects on a timeline. In many cases a length would have to be expressed. There would need to be ways to easily define an overarching harmonic and metrical structure, if that was required.

A hierarchical relationship structure

The object described above bears a relationship to an object that might be found in a Composite software design pattern, first identified in Gamma et al. (1995, p. 395). In a Composite pattern, many similar objects can be grouped together and treated as one, with their relationships defined in a tree structure. An analogy for this might be a box containing objects or boxes which themselves containing objects or boxes, and so forth. Each box or object has a unified way of detailing a description of its contents to its parent box. Getting a list of all the objects in a box would simply involve calling for this information from the box containing everything. Each object in this box would return a description of itself for the final list, but each box it contained would interrogate its contents in a similar fashion and return a list of descriptions of objects it contained, or the list of object descriptions from boxes it contained, and so on. The Composite pattern describes a unified interface for items contained within it, and a tree structure for describing their relationships. This hierarchical approach resembles analysis systems such as the Generative Theory of Tonal Music as described in Lerdahl and Jackendoff (1983), where the focus is on hierarchical musical structures. A Mu instance has properties, such as its position in relationship to its parent and its length, expressed in one of several metrics[33]; a list of notes (if any); a list of controller events (if any); and a list of child Mus (if any), who can relate themselves to their parent in terms of position, length, pitch and any other information held by the parent.

An extension of this was being able to express ideas such as ‘before’ or ‘after’, and using musically useful metrics. Examples of the use of this would be:

[33] Bar offset, bar and beat offset, offset in quarter notes, offset in floatbars or offset in time.

1. a ‘chorus’ object with a sibling relationship to a ‘verse’ and directed to begin at the end of the verse, the net effect being that the chorus plays after the verse, even if during the process of development the verse changes length
2. an ‘accompaniment’ structure with a child relationship to a ‘melody’ structure and directed to begin at the start of its parent, the net effect being that the accompaniment plays at the same time as the melody
3. an embellishment note attached to a structure tone in some temporal relationship (before or after), and from which it can derive note and other musical information (e.g. chromatic/diatonic/chord tone above or below).

Some useful ways to edit and propagate information

Instances of the Mu class would be containers for musical elements to be added to a musical structure, which itself would be represented by a Mu. Once added as the child of a Mu already in the structure, the details of how those musical elements were integrated into the structure would be abstracted from the user. This would resemble the way that a conventional ensemble of human musicians might assimilate adjustments during the development of an arrangement:

1. There would be procedures for accreting musical structural elements, such as prevailing chords, time signature, and tempo changes, from child objects. An example would be that, given a melody (consisting of a tree structure of Mus representing note objects) over a chord progression held in the top-level Mu, adding a chord indication (not actual notes, just the equivalent of a chord symbol on a lead sheet) to one of the notes would propagate upwards in the hierarchy to be included as a new chord in the chord progression held by the top-level node. This would have an impact on any other musical object deriving parameters from this master chord progression.
2. It would have the ability to store construction metadata in these musical objects, with the goal of easily being able to re-render parts of a composition. This was based on the notion that parameters of a composition could be directly edited if this were a useful way to interact with the composition process, and affected parts could be re-rendered using the easily available construction information. In the example above where a new chord is added to a chord progression, this would affect the choice of chord tones for the melody for the duration of that new chord. Stored construction metadata could be used to re-render these notes, as well as any dependant embellishments.

Output and persistence

Ways were needed to convert the representation to an audio output and to some kind of visual output. The representation also had to be saved to and retrieved from a disk and a database.

Audio output

For testing purposes, I used Ableton Live as a playback device. Using the Live Object Model (LOM) exposed in the embedded Max for Live visual programming environment, I was able to inject MIDI information directly into clips in Ableton's Session View. Material was generated in a freestanding Java application, transferred to Max for Live via a local UDP network connection, and then transferred into the clips by the LOM. This was convenient in several ways. Material could easily be looped, as Ableton's Session View playback is predicated on the idea of looped material. It was easy to choose sounds for the various parts, and Ableton is a robust playback engine so there was little to worry about after the set-up was working.

At this stage, several limitations of the system were noted. While it is simple to access any parameters in the Ableton Live set, it was less easy to loop controller information, as there was (and still is at the time of writing) no way of injecting controller information into clips in the Session View. This was doubly problematic for pitch bend, as there was no way to access this parameter directly from the LOM; pitch bend information had to be injected into the MIDI stream to the relevant virtual instrument. Lastly, Session View clips do not support the use of multiple time signatures, so in such cases the total length of the phrase would need to be correctly calculated in quarter notes, and the bar lines in the clip view would not correspond with the algorithm's. None of these issues were actually part of this first phase of development, so they are only noted. The functionality of pitch bend, controllers and multiple time signatures were available in the software model, but they were not used during initial development.

Score output

Further on the subject of output, I added the ability to render a Mu hierarchy to a score output. During an earlier research project (Armstrong, 2017) I found this output format very useful, not only to view the output in staff notation, but also to attach annotations to add clarity to the workings of the algorithm in the process of generation. Besides this value as a research output, it could also act as a medium to fulfil the requirement of transparent algorithmic processing, as noted by Karimi et al. (2019, p. 7). This took the form of a custom

.musicxml[34] renderer, another fiddly and complicated endeavour. This .musicxml file would be saved to disk and then manually opened in MuseScore, which was not quite as convenient as the automatic playback of new material with Ableton Live, but usable and useful for experimentation. Between the score output and Ableton's piano roll clip view, I was able to satisfy any visual output needs for this stage of development. And while I personally gravitated to the score output, many years of experience with the piano roll output in digital audio workstations made this an equally useful perspective on the material.

Persistence to disk

Since I was conducting these experiments on a single personal computer, at this stage I only addressed the issue of saving and loading to and from disk. I needed a file format for saving and loading a generated Mu to disk, so that output could be saved during the interaction. Based on my experience with the .musicxml renderer, I implemented a renderer to a custom XML format for Mu data. I also used simple text files to save algorithm parameter data to disk. At this stage it was mostly for viewing during debugging, but later, after some interaction data from testing during the next step of development, I was able to data mine these files to statistically analyse the behaviour of the algorithm and its interaction model.

4.1.2. Early interventions

The first intervention model was based on the aforementioned design criteria of bundling construction metadata within the Mu. I proposed to identify the problematic area of an output instance, which would allow easy access to the process that created it, and then to edit the offending process. This model was found to be flawed in many ways.

Firstly, and most importantly, while the process worked, it was obvious that it only suited a small number of use cases, where the output was 'mostly right' and just needed some tweaking. It was problematic in the large number of cases where major changes were required to proceed. In cases such as these I ended up reverting to generating a new random option, and seeing if I could work with it. This cycle of reviewing randomly generated output was a situation I was determined not to foreground. What did become clear was that the generation system, being rule based, was very rapid, and that what I had seen as a rendering time optimisation – being able to re-render just a portion of the output – was, from the point of view of the user, no different to re-rendering the entire output.

[34] https://www.musicxml.com/

Secondly, it was unclear how to go about getting a similar editing input from the computational agent. A random selection of editing choices were possible but, once again, we were back into random decision making, and there seemed to be no clear approach to turning the actions of the user into data which the computer might use to reason about a musical direction upon which to act. It seemed that if I was looking for a model that a) dealt with adjustments to the music more broadly and b) mapped musical intentions to explicit descriptions of the intentions, a better approach might be to capture those intentions more directly from the user.

The idea of describing musical intentions in terms of genre and mood seemed to require an extra layer of mapping that appeared complex from the outset. Also, the use of simple provocations of mood and genre is exactly the approach used by many available algorithmic music composing tools aimed at video content producers, who are likely not primarily musicians, and also one of the primary classifiers used in generating music playlists for listeners. I was clear that I wanted this interaction to foreground the process and experience of composing, so I settled on the idea of parameter-specific feedback, which directly linked descriptions of musical intention with generation parameters in the algorithm. I deal with a description of this model in the next step. I also realised I needed a more easily manageable way of describing the parameter set for a given output, and so I built a phrase-length melody generation model.

4.1.3. Phrase-length melody model

The design criteria for the melody generation model were:

1. a single input object containing all parameters for the generation of output.
2. a way of varying specific parameters
3. a way of delineating a parameter space for the input parameters which would be useful for statistical analysis
4. from a user-friendliness point of view, a concept of lazy instantiation of parameters. This meant that a process would revert to a default or make a best guess at an undefined parameter so that the output was unlikely to fail; improvement would be encouraged through easily accessible editing methods.

Initial development of this first generation algorithm involved developing a system for generating random parameter sets. Once a parameter set had been generated, it was then

rendered as musical output. The process involved using a standard random number generator to select a particular parameter from a set of options. This option would be added to the parameter file. The random number, in the range 0.0 to 1.0, would then be stored in a separate file but noted as being related to the parameter. This would allow uncomplicated variation of parameters by varying, to some degree, the original random number that generated them.

This early generation algorithm operated in the following way, with a reduced subset of options of what was envisioned for the final version:

- Structural parameters:
    - Phrase length: a choice of 4, 6, 8 or 10 bars
    - Time signature:[35] 4/4, 3/4, 5/8, 7/8, 6/8
    - Tempo: 65–145 bpm
    - Key: any single major key
    - Chord progression: random choice of any of the six major and minor diatonic triads (vii° omitted), at a rate of one chord per bar
- Melody parameters:
    - Start and end point of the melody, expressed as a percentage of the total bar length
    - Structure tone spacing, expressed in 'floatBars',[36] with the option of 0.5 or 1.0. The actual positions would be multiples of this value (e.g. 0.5, 1.0, 1.5, 2.0 etc. in the case of a 0.5 spacing) and also fall inside the part of the phrase delineated by the melody start and end point.
    - Structure tone contour, expressed as a series of x, y co-ordinates in a space, with the x range of 0.0 to 1.0 representing a percentage position in the melody position range as expressed by the start and end point of the melody, and the y value in the range −1.0 to 1.0 multiplied by the contour multiplier, which was in the range of 5 to 14 semitones. This would be calculated for each structure tone position, and quantised to the nearest chord tone of the prevailing chord.

[35] At this stage only a single time signature for the duration of a generated clip was explored. This was a limitation of the generation algorithm, though the model catered for variations.

[36] FloatBars was a decimal expression of position on the timeline, where one bar equalled 1.0 floatBars, and position within the bar was the decimal part interpreted as a percentage. This was useful in translating positions between even and odd time signatures. For instance, a floatBar position of 0.5 would be the third beat in the first bar in a phrase in 4/4 time, assuming a conventional arrangement of beat strength where the first beat is the strongest, the third the next strongest, etc. When translating this position to, say, 7/8 (or indeed 4/4 with a different accent structure) it was possible to add a process where the position might be quantised to a beat with a similar strength to its previous position, which in 7/8 would not fall exactly in the middle of the bar, but instead depend on how the accents were divided up in such a time signature.

- The embellishment repetition pattern: this was a list of indices that was looped through to indicate which series of embellishments would be applied to the structure tones. For instance, a pattern of [0, 1] would require two series of embellishments, with index 0 applied to the first structure tone and index 1 to the second; it would then loop back to 0 for the third tone until all the structure tones had been embellished.
- Embellishment patterns: these were a series of embellishments applied to a structure tone. There were four rhythmic choices, each one indicating the position of the note prior to the note it was embellishing.[37] Furthermore, there were three melodic options which depended on the pitch of the note being embellished and, in two cases, also the pitch of the previous note: anticipation, an unaccented step-jump escape note and an unaccented jump-step escape note.[38]
- Dynamics were added to the melody, which at this stage involved giving the structure tones a higher MIDI velocity than embellishments.
- A rudimentary accompaniment was added, consisting of chords, bass and drums. Chords and bass would sustain for an entire bar, and drums were derived by alternating time signature accents between kick drum and snare and having high hats marks out eighth notes. This worked sufficiently well to demonstrate success when paired with an interaction system.

4.1.4. Reflection on this stage

[37] The position was always prior to the note to be embellished, and was cumulative. The first embellishment embellished the structure tone, the second (if there was one) embellished the first, etc. This was expressed in terms of tactus and subtactus.

[38] These escape note terminologies are my own, and only a step-jump tone possibly represents the conventional understanding of an escape note, which steps away from the destination note by a step before returning by a jump. A jump-step structure might be seen as an appoggiatura if the embellishment were accented, but in this case it was not, as it involved then moving the note to be embellished to a relatively weaker beat. Eventually, when I included accented embellishments, I only applied embellishment accentuation to the structure tone and first embellishment, as moving already existing strings of embellishments was complicated and less likely to deliver valuable variations than other compositional devices to which I could devote my time.
Several other quirks of the generation process must excuse the use of the 'escape tone' name in this case, as there where many situations where the use of one of these embellishments resulted in anything but an escape tone. Firstly, as notes were added backwards, with the note embellishing the structure tone being before the structure tone and the second embellishment being before the first, when calculating the 'escape tone' in this case the previous note was not the previous note that would end in the final version, but rather the previous structure tone. Intervening embellishments might end up resulting in an embellishment which was not an escape tone at all. Secondly, in the situation where the previous and next notes in an 'escape note' calculation were the same, the step and jump direction was not clear, as such a usage would (without later intervening embellishments) be a neighbour tone. As a result, these two embellishment options acquired a lower or upper neighbour tone parameter to cater for this situation. These neighbour tone parameters can be seen attached to escape tone embellishments in Appendix A3.

Initially, intervention directly into the hierarchical music representation system was seen as the primary intervention into the music generation process. After testing, this emerged as less interesting than expected. Once the phrase-length melody generation algorithm was working and the output could be reviewed, I saw that a less detailed approach to intervention might be more useful.

The idea of embedding construction metadata with each Mu might ultimately not be a bad idea, as it could allow for re-rendering of subsections of a long composition being worked on, and therefore save time. This functionality fell by the wayside quite soon during this stage of development, as the scale of the musical output of the generation process was short enough that re-rendering the entire artefact was never a computational bottleneck.

During this stage the idea of embedding all the functionality I foresaw being required of the Mu class was taken too far. The scope of what actually was required was only data and methods related to the structure and surface of the musical artefact. This became clear once I began adding code related to persistence to the class, in particular the methods required to render out the artefact as an XML document. This resulted in several thousand lines of code being added to a class which was already quite large. In line with the software design principle of separation of concerns, I began to move code which was not explicitly related to the handling of musical material to separate utility classes.

Furthermore, where I had envisioned the Mu class to be central to the interactivity of the project, it became the framework for representation of the music. Other frameworks for generation, variation, analysis and output emerged over the course of development, upon which further experiments in interactivity were built. This was a natural outgrowth from my evolving understanding of the application of the software design principle of loose coupling between elements of a software system. As this separation of concerns became clear, a structure of separate software modules with common interchange formats emerged that could also be applied to other representation, generation, variation, analysis or interaction systems which could also be included in the system.

4.2. Step 2: Interaction models

The first experiments in interaction began with exposing the parameters of construction of the artefact to the user for editing. While these experiments verified that the representation and generation model worked, it did not add anything to human–computer interactions not

already addressed by software such as digital audio workstations. What was required was the addition of a model which allowed the computer to learn from the interactions and develop a model for identifying better or worse options. Using Wiggins' (2006) model, this would be described as developing *E*, a set of artefacts considered valuable, from *R*, a set of all valid artefacts.[39] In order to develop this, there would need to be some quality assessment attached to generated artefacts, provided by the user.

4.2.1. Finalising the interaction model

The problem with capturing user sentiment is assessment fatigue of the human user (Biles, 1994; Fernandez and Vico, 2013, p. 553). While computers excel at repetitive tasks, human users have trouble maintaining consistency in assessment for any length of time if solely focused on this task. This suggests that engaging the user in some way which immediately rewards the user for capturing a sentiment could be a way of circumventing this fatigue. I proposed that harnessing the user's curiosity to see what might happen next in a process where they had some, but not total, control over the output, and a reasonable trust that the development could proceed in a way that was satisfying for them, might be a way to sustain the user's engagement. I also envisioned quite clearly limiting the terms used in sentiment capture to relate to the steering of the interaction at hand rather than capturing sentiments broadly related to a musical style. It was important that the user not overthink this step in the process. This also raised the possibility that the user might edit the way in which the sentiment was used, but testing this would require more data than was available at the time of development.

I speculated that the kind of language used in discussing the development of a collective composition between human musicians in a rehearsal setting might be the way to proceed. Such a discussion could encompass anything from broad indications of genre (e.g. 'More bluesy') or mood (e.g. 'More soulful'), examples of other artists (e.g. 'More like Fela Kuti'), examples of other works (e.g. 'More like "Stairway to Heaven"'), general indication of musical parameters (e.g. 'Faster', 'Slower'), explicit musical instruction (e.g. 'Play a Db7#11 in bar 2') or even just indications related to what already existed in the composition (e.g. 'More of that' or 'Don't play that'). These gave many options to consider, but it seemed that a generalised indication of an explicit musical parameter was something that the interaction

[39] R, in this case would be all artifacts describable by the generation algorithm, which is a subset of all music.

model was heading towards. Also, as I was planning on using material that was no longer than a musical phrase, there seemed to be a case for capturing a like/dislike sentiment about the output as a whole. This would later need some management during the interaction process, as the effect of the like/dislike mechanism certainly caused myself as a tester pause for thought as to whether I actually liked it, whether I was choosing it to deliberately manipulate the preference engine, or whether I was not sure and thus choosing randomly just so that the process could proceed.

I had considered adding the possibility of the user contributing note inputs, in the form of a musical example in symbolic form, played input or an edited version of output. Research into this mechanism proceeded alongside what is described in more detail below, but was not included in the testing in step 2 and step 3, as it was not complete. It did serve as a consideration of the extensibility of the software, in that I did not allow any design choices which might compromise its (and for that matter, any other interaction models') inclusion in the application.

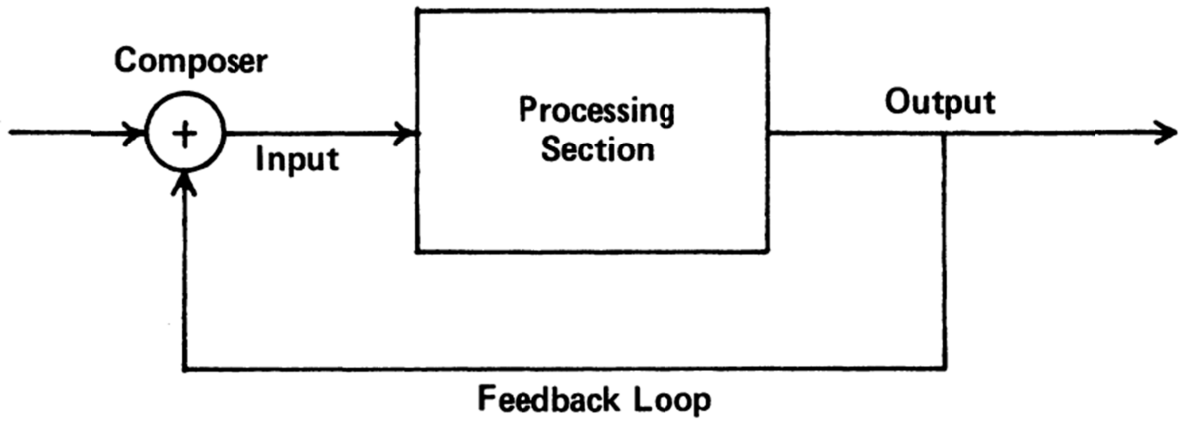


Figure 4: Chadabe's illustration of sound generation based on a classical feedback model (Chadabe, 1977, p. 5)

In Figure 4, Chadabe (1977, p. 5) illustrates a simple mechanism for organising an interactive composing system, where a *composer* adjusts the *input* to a *processing section* to achieve an *output* which is relayed with a *feedback loop* to the composer for iterative development. The co-creative interaction loop I proposed to use extends this iterative model, with some important differences. Firstly, based on the general aim of an *alternating* co-creative interaction (Kantosalo & Toivonen, 2016) with a complete computational agent, the loop would require two cycles to complete, with the composer role being alternated between a

human user and a computational agent on each cycle. Secondly, the feedback loop would include the opportunity to capture user feedback about the output.

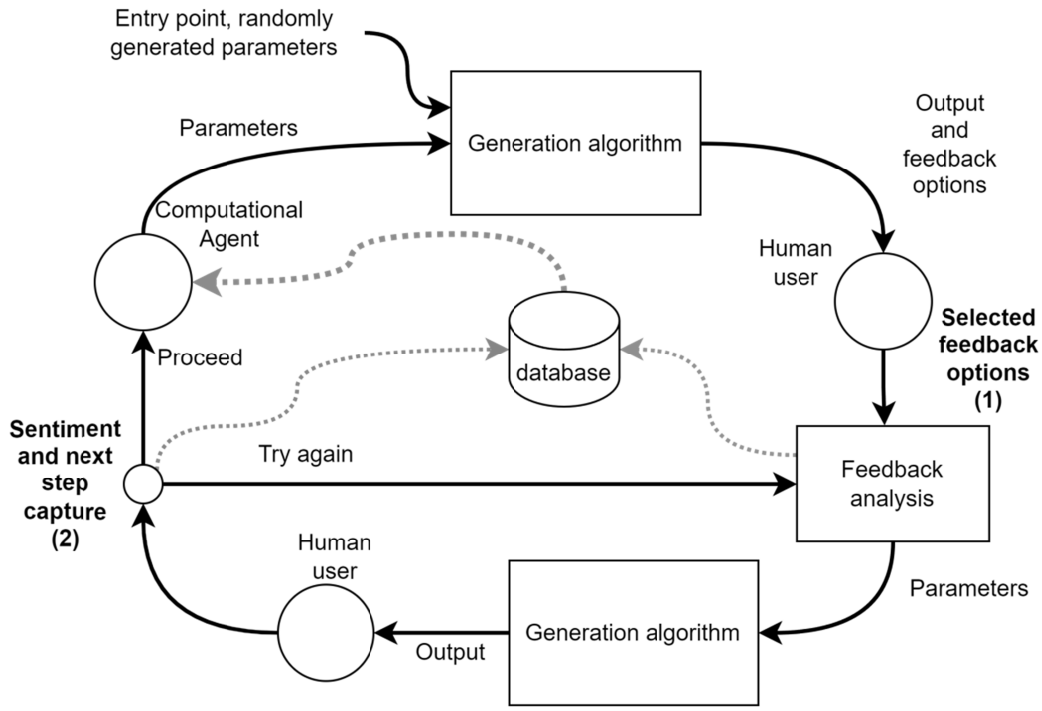


Figure 5: An extension of Chadabe's (1977) interactive composing model, demonstrating a co-creative workflow that captures parameter-specific feedback for the computational agent's output at (1) and the user's sentiment and choice of next step at (2). All user feedback and sentiment data is bundled with the output to which it refers, and is saved to a database to be used by the computational agent to fine tune its traversal of the parameter space.

demonstrates the control flow of an interaction using parameter-specific feedback. The *computational agent* applies some 'creative' process to generating output. An interactive loop starts at this point and the first output by the agent is random. While not exactly creative, this was a placeholder for development to get things going. Although many processes could be applied to deciding this first output, I found that the strong possibility of generating something through a random process that required improvement was in keeping with the goal of precipitating an interaction to develop this input. Were this process too 'good', it might have either caused the user to continuously restart the interaction to see what might come up next – which is not the co-creative interaction I was looking for – or resulted in output that was so satisfying that the user would not require any further options, which would fail to fulfil the requirements for an engaging interaction.

As a result of this, variations can be generated (changes to a small number of parameters from a given parameter set likely make for a new output which is 'similar' to the original)

and automatically evaluated (based on the statistical 'distance' of one parameter set to another). This allows any qualitative assessments about the generated music which are captured in the interaction to be associated with this position in the parameter space of the algorithm and applied to similar (statistically 'close') parameter sets. The following two interaction methods have become clear as essential to the scope of the project:

1. Parameter-specific feedback
2. Direct editing of musical material.

This allows for the following research questions to be addressed:

- Can the use of two interaction models that have their own limitations facilitate longer user engagement or less user frustration than one?
- How can the above be tested to point towards an optimal balance?

Parameter-specific feedback

Parameter-specific feedback is focused on users who are trained composers,[40] as the naming of the parameters is directly based on terms used in music theory and music composition for detailing technicalities of music construction. The user selects options from a list of possible changes to the parameters of a given melody, and then approves (or not) the result and manages whether the next step will attempt a better version or whether the computational agent, through its own criteria, can propose a variation to garner further feedback from the user.

This allows for easy positioning of user feedback ('Like', 'Dislike' and 'Not sure') in a parameter space directly related to the generation of musical material. This list of options was limited to musical parameters that referred to the presented musical artefact as a whole. The musical artefact was a phrase-length construction from two to twelve bars long, consisting of a melody and accompaniment comprised of chords, bass and drum parts.

Direct edit

A second interaction model was proposed: whereas the first interaction model gathers user sentiment about generated material where that sentiment is already linked to a particular compositional intent, the second proposes that the user directly edit the generated output. The output is then converted from low-level symbolic musical representation (MIDI) to

[40] As defined in 3.1.2.

parameters for the generation algorithm that will produce the same result, which in turn can infer a link with the particular parameters that constitute what the algorithm may consider an improvement of the content. This links musical material directly back to the algorithm parameter space and the attached user sentiments. This process is essentially the reverse of the generation algorithm, converting symbolic music to generation. More interestingly, once a workable analysis engine is achieved it could be applied to an interaction involving direct editing of a score or piano roll, or a performed musical contribution by the user. This would expand the interaction to other types of users, such as the untrained composer who might have some skill with music production software, or a performer who is more comfortable with a performed contribution.

It was proposed that having several options for interacting with the material would make the application more engaging, giving users the opportunity for a different approach if an interaction model was not progressing the interaction satisfactorily. Also, from a perspective on algorithmic music generation and analysis where each individual generation, interaction or analysis process may come with some degree of flawed behaviour or limitation, combining several such elements may allow for limitations to be circumvented.

4.2.2. Implementation and testing

While it was proposed that parameter-specific feedback and direct editing of musical material be included as interaction models, the time constraints of the project resulted in the decision to postpone the addition of the direct edit option for this stage of testing. While the root of the interaction is an equal co-creative partnership, with each agent contributing on the same level, there is no doubt that the two agents bring vastly different strengths to the table. While each agent can contribute musically, I had no intention of attempting to model the computational agents' level of boredom or excitement. This issue is important to consider regarding the human agent, and I found it helpful to add the ability to shortcut the turn-taking process if it seemed to the user that the computer was stuck in some way. This includes such easily understandable notions as 'Undo', 'Try something else' or 'Let's move on, this is going nowhere'. A more detailed implementation of this was the ability to lock one or more of the parameters, in particular parameters such as tempo and time signature, where it was noticed that even small changes would change the character of a piece significantly. Such control overrides the equality of the interaction, as the computer has no similar control over

the process, but allows for the user, with their much broader insight into the context in which the interaction is happening, to optimise the progress in a humanly relevant way.

For simplicity's sake, this interaction was implemented using the command line console as an input device for the user. Output consisted of MIDI playback via Ableton Live, and music XML files were saved to the file system, which needed to be opened manually in MuseScore. The database of interaction data from the current session was held in memory, which meant that only data from the current session were considered by the algorithm to facilitate its automated parameter traversal process. At this stage I was testing the broad strokes of the interaction, so there was no need, and not enough data, to consider the idea of using (or not using) certain saved files of user sentiment. Backups of the output files, input parameters for the algorithm and the record of each interaction were also saved to the file system, so that analysis could be done at a later stage. For testing, I recorded desktop videos of myself operating this set-up, recording the output and commenting on progress. This approach to testing was helpful for me to focus on being a user rather than a developer.

The following is a sample output of the console, with some annotations:

```
"feedback_options": [          ← Indexed list of feedback options
     "0) PhraseLength_Longer",
     "1) PhraseLength_Shorter",
     "2) PhraseLength_MoreEven",
     "3) PhraseLength_LessEven",
     "4) TimeSignature_MoreEven",
     "5) TimeSignature_LessEven",
     "6) Tempo_Faster",
     "7) Tempo_Slower",
     "8) StartNote_Lower",
     "9) StartNote_Higher",
     "10) StartNote_SignificantlyDifferent",
     "11) XMLKey_UpTo5KeysSharper",
     "12) XMLKey_UpTo5KeysFlatter",
     "13) StructureToneContour_Different",
     "14) StructureToneMultiplier_Higher",
     "15) StructureToneMultiplier_Lower",
     "16) PhraseStart_Earlier",
     "17) PhraseStart_Later",
     "18) PhraseEnd_Earlier",
     "19) PhraseEnd_Later",
     "20) StructureToneSpacing_Longer",
     "21) StructureToneSpacing_Shorter",
     "22)
EmbellishmentClearanceFromPreviousStructureTone_More",
```

```
          "23)
     EmbellishmentClearanceFromPreviousStructureTone_Less",
          "24) EmbellishmentRepetitionPattern_MoreVariety",
          "25) EmbellishmentRepetitionPattern_LessVariety",
          "26) ChordProgression_FewerFunctionChanges",
          "27) ChordProgression_MoreFunctionChanges",
          "28) MugList_CountIsLess",
          "29) MugList_CountIsMore",
          "30) MugList_RhythmHasLessVariety",
          "31) MugList_RhythmHasMoreVariety",
          "32) MugList_RhythmLongerOnAverage",
          "33) MugList_RhythmShorterOnAverage",
          "34) MugList_PitchHasLessVariety",
          "35) MugList_PitchHasMoreVariety"
     ]
}
get feedback
21,11      ← User feedback selection (by list index)
-----------------------------------------------------------------
get_user_approval     ← List of user approval options
1)approve
2)try again
3)new bot variation
2     ← User selection, in this case 'Try again'
-----------------------------------------------------------------
get_user_approval
1)approve
2)try again
3)new bot variation
1     ← User selection, in this case 'Approve'
-----------------------------------------------------------------
{    "feedback_options": [
          "0) PhraseLength_Longer",
          "1) PhraseLength_Shorter",
          "2) PhraseLength_MoreEven",
          "3) PhraseLength_LessEven",
          "4) TimeSignature_MoreEven",
          "5) TimeSignature_LessEven",
          "6) Tempo_Faster",
          "7) Tempo_Slower",
          "8) StartNote_Lower",
          "9) StartNote_Higher",
          "10) StartNote_SignificantlyDifferent",
          "11) XMLKey_UpTo5KeysSharper",
          "12) XMLKey_UpTo5KeysFlatter",
          "13) StructureToneContour_Different",
          "14) StructureToneMultiplier_Higher",
          "15) StructureToneMultiplier_Lower",
          "16) PhraseStart_Earlier",
          "17) PhraseStart_Later",
```

```
            "18) PhraseEnd_Earlier",
            "19) PhraseEnd_Later",
            "20) StructureToneSpacing_Longer",
            "21) StructureToneSpacing_Shorter",
            "22)
      EmbellishmentClearanceFromPreviousStructureTone_More",
            "23)
      EmbellishmentClearanceFromPreviousStructureTone_Less",
            "24) EmbellishmentRepetitionPattern_MoreVariety",
            "25) EmbellishmentRepetitionPattern_LessVariety",
            "26) ChordProgression_FewerFunctionChanges",
            "27) ChordProgression_MoreFunctionChanges",
            "28) MugList_CountIsLess",
            "29) MugList_CountIsMore",
            "30) MugList_RhythmHasLessVariety",
            "31) MugList_RhythmHasMoreVariety",
            "32) MugList_RhythmLongerOnAverage",
            "33) MugList_RhythmShorterOnAverage",
            "34) MugList_PitchHasLessVariety",
            "35) MugList_PitchHasMoreVariety"
      ]
}
get feedback
28,6
------------------------------------------------------------
get_user_approval
1)approve
2)try again
3)new bot variation
2
------------------------------------------------------------------------
```

4.2.3. Reflection

After the establishment of this approach to music generation, I sought ways to make the system more robust in dealing with missing information. I used a plugin-type architecture for handling each parameter in the generation process. When a plugin for a particular parameter was called, it would establish whether the information it required to proceed, if any, had been generated, and call those plugins prior to proceeding only if necessary. If no plugin for a required parameter had been explicitly prescribed, it would proceed with a default. As a result of this, the generation process might be called with the barest minimum of input, such as only an indication of tempo, or maybe a sequence of embellishments, and a complete output could be arrived at rapidly. This would then be fine-tuned through the interaction process.

4.3. Step 3: The qualitative pre-alpha tests

As development proceeded, user testing became a goal. The pre-alpha tests were seen as a way to re-invigorate the project with new insights. At this stage of development, only the first interaction model had been implemented. With the prospect of the application being tested online, during this development cycle I used some of the time to translate the existing software to frameworks which would support a web deployment. In this section I detail the state of the system, the plan for collecting data and the lessons learned from the process.

4.3.1. Pre-alpha test presentation platform

The inclusion of other users required upgrading the user interface to something that was easier to use than the IDE console that I had used thus far. This turned out to be a process involving several refactors and a rapid learning curve due to the inclusion of unfamiliar software frameworks. With the constantly developing sense of better practices for software organisation that the development of this project provoked, during the implementation of the integration model I had made an attempt to separate the user interface from the generation engine. In retrospect this was a simple implementation of a Model View Controller (MVC) software model.

Realising that it might be a good idea to use standardised implementations of tools based on the concept of MVC, I added the following technologies to the project:

- Spring Boot:[41] a Java framework for implementing the back-end of web applications
- React:[42] a Javascript framework for developing browser-based front-ends for web applications
- Html, CSS and JS (Hypertext Markup Language, Cascading Style Sheets and Javascript): the core technologies of the world wide web, and the ecosystem within which a framework like React operates. These are also the central technologies supported by browsers (Chrome, Firefox, Edge, Opera, Safari etc) to deliver the client side of websites to users
- MUI:[43] a system of CSS utilities that comes with a library of components for designing websites.

[41] https://spring.io/
[42] https://react.dev/

The new layout of the project for pre-alpha testing was as follows:

- A Java/Spring Boot back-end encapsulating the algorithmic composing and interaction models
- A browser-based front-end written in React, communicating with the back-end via http messages.
- Ableton Live for playback, receiving communication from the back-end via a User Datagram Protocol (UDP) network message in the bespoke format developed by myself for this purpose prior to the start of this project.

Interaction data for a given session were saved in memory for unsupervised data analysis in the interaction module of the back-end. Furthermore, time-stamped output and interaction data were saved to the local file system. This system ran on a single computer and although the system was using some technologies used in web deployment, the use of Ableton for playback, the flat file storage system and the possibility of only servicing one user at a time did not make it feasible for actual web deployment. That can of worms would be opened at a later date.

4.3.2. Pre-alpha test questionnaire

The following questions were a guideline for a debriefing interview with the participant after interacting with the software:

1. Participant info
    a. Describe your music education.
    b. Describe your use of technology in your musical composing practice.
    c. Describe your broader use of technology related to music.
2. Interaction and musical output:
    a. Comment on the choices offered by the application.
    b. Comment on the musical output of the interaction.
    c. Comment on the musical continuity of the interaction.
3. Reflection on the interaction:
    a. Comment on your engagement or frustration with the interaction.

---

[43] https://mui.com/

b. Are there any interaction options that were not available that you thought should have been?
c. Would you use this tool in your musical practice?

4.3.3. Reflection on pre-alpha tests

The fourteen cognitive dimensions of notations (Green, 1989) represent a lightweight heuristic methodology and common language for talking about the usability of notations, interfaces and computer languages. In particular they are aimed at directing discussions around the analysis of existing artefacts and the development of new artefacts in this domain. What follows is a brief description of Green's cognitive dimensions. Thereafter I discuss the events of the early user tests within this framework, identify improvements for the next stage of testing and propose statements for Likert-style multiple-choice evaluations to be included in the accompanying survey.

Green's (1989) fourteen cognitive dimensions are as follows:

1. Viscosity: How difficult is it for a user to accomplish a goal?
2. Visibility: How easy is it to locate controls?
3. Premature commitment – Yes/No: Are users forced to commit without knowing the full impact of their choices?
4. Hidden dependencies – Yes/No: Are there dependencies between elements that are not obvious?
5. Role expressiveness: Is the role of an element obvious from its appearance?
6. Error-proneness: Can the user easily get things wrong?
7. Abstraction: Are elements grouped together due to often being used together? How many of these groupings need to be learnt?
8. Secondary notation: Is it possible to add extra information to make the function of an element easier to understand?
9. Closeness of mapping: How closely and obviously does an element do what it says it should?
10. Consistency: Are consistent patterns, language and symbols used in the application?
11. Diffuseness: Does the meaning of an element require a verbose explanation? Can it be expressed more succinctly?
12. Hard mental operations: What is the cognitive load of the interface? Can the user understand the interface and interaction without referring to other resources?

13. Provisionality: Is it possible to explore choices before committing to them?
14. Progressive evaluation: How regularly does engagement with the system allow a user to evaluate their progress?

Viscosity

The viscosity dimension comes most clearly to mind as emerging from the area of the interaction which seemed to move beyond the control or expectation of the user. The feedback process using word descriptions of changes to musical parameters would create a sense of confidence (and hence low viscosity) for the user in a case where the musical variation that resulted was an example of this word description. So for instance, when the user deemed that the artefact needed a shorter phrase length, and the variation generated from this feedback actually had a shorter phrase length, the algorithm was seen as behaving in a (reasonably) predictable way, which inspired confidence in the continuation of the interaction. When the application behaved in an erratic or contradictory way, this was described by a user as going off track, and raised the viscosity.

In the version of the software used for these tests, the moment when the computational agent delivered its next proposal for the interaction was seen as a point where continuity could be lost. It is perhaps gratifying that this moment was identified as such, as the computational agent's contribution was not based on any data collected about the preferences of the user. Due to time constraints during preparation for these tests, I had used a random process to generate these contributions, and they were correctly[44] identified as a moment when things most likely would go wrong. It is possibly an emergent effect of the way the algorithm constrained the music generation algorithm, using cogency rules from music theory, that resulted in a small proportion of these moments actually being seen as progressing in a direction pleasing to the user.

There are two approaches to reducing this particular moment of high viscosity:

- Actually implementing an automated learning process whereby the algorithm might make better-than-random choices about the musical direction of its contribution. Obviously this was planned but not available in these tests due to time constraints.

[44] These were identified without guidance from myself. In fact, I misled the users initially about this step, declaring it to be a choice based on data gathered from the interaction thus far.

- Reducing the possibility of the emergence of contradictory outcomes: In the case of the variation offered by the algorithm, the output was often bewildering, as the variation could often randomly deviate from a musical direction perceived, but not articulated, by the user.[45] In the case of variation based on specific user feedback, there was a possibility of a variation that contradicted this feedback. The process went as follows:
    - The user would select several parameter-specific feedback items (e.g. phrase-length-longer, time-signature-faster, etc.[46])
    - The algorithm would generate a set of variations which adjusted the specified parameters randomly.
    - The resulting artefacts would be scored by an evaluation algorithm addressing each selected parameter, generating a higher score if the artefact reflected the vector of change for that parameter.
    - The artefact with the highest score would be selected.

It should be plain to see that in the case where multiple feedback items were chosen, an artefact could score poorly, even negatively, on one of the evaluations, and yet score so well on the others that its overall score would be the highest of all options generated. Initially I did this as a way of retaining the possibility of contradictory behaviour which might actually pull the interaction in an unexpected, but positive, direction for the user. The problem remained that this was still an entirely random process and was not seen to be particularly appropriate or effective in these tests. As a result of this, I changed the random variation of the selected parameters step to one which randomly adjusted the parameter, but only in the selected direction.

Furthermore, the first feedback items used a linear scoring method, so, for example, if the user chose the tempo-faster option, the highest scoring artefact would always be the fastest, when in the user tests it emerged that the compositional intent behind tempo-faster meant ‘tempo somewhat faster’ rather than ‘as fast as possible’. It was thus necessary to have score mappings which allowed for a range of possible options to achieve the highest score,

[45] In this case it would be as much a case of there being no learning algorithm, no opportunity for the user to directly influence this variation generation process or a flaw in the interaction design. I would dismiss the last, as the decision for this step is based on models of computational co-creativity described in Kantosalo and Toivonen (2016).
[46] See Appendix A5 for a list of available feedback options for this version of the software.

otherwise one would be replacing the reading of ‘tempo-faster’ as the fastest in the range permitted by the algorithm, with one where a new value with a fixed deviation from the original was always favoured with the highest score. This suggests that the computationally intensive generation of variations mentioned above could be replaced by simply varying the original value by some preset percentage or offset value.

In anticipation of the quantitative testing to come, I proposed the following question related to viscosity for a Likert scale questionnaire:

> Approaching step X in the interaction, I felt confident that the interaction would progress to my satisfaction in an interesting way.
>
> 1. Strongly agree
> 2. Somewhat agree
> 3. Neither agree nor disagree
> 4. Somewhat disagree
> 5. Strongly disagree

Visibility

The current interaction page can be seen in Figure 6. The screen within the browser features a blue menu bar, which gives access to menus and settings unrelated to the minutiae of the interaction as well as information about the name of the currently playing track. Below this, taking up the bulk of the available screen real estate, is the layout for the collection of user feedback. As a way of organising the potentially large number of feedback options, these options are nested in three layers:

1. Part name: In this particular case, there is a section for global parameters and then parameters related to melody, chord part, drum part and bass part.
2. Only the global option is expanded in this example. We can see a list of parameters that can be addressed in the feedback, in this case phrase length, tempo, time signature and key.
3. In this example the time signature option is expanded. Here can be seen the feedback options related to the parameter; in this case the user could select ‘More even’ or ‘Less even’ using the checkbox.

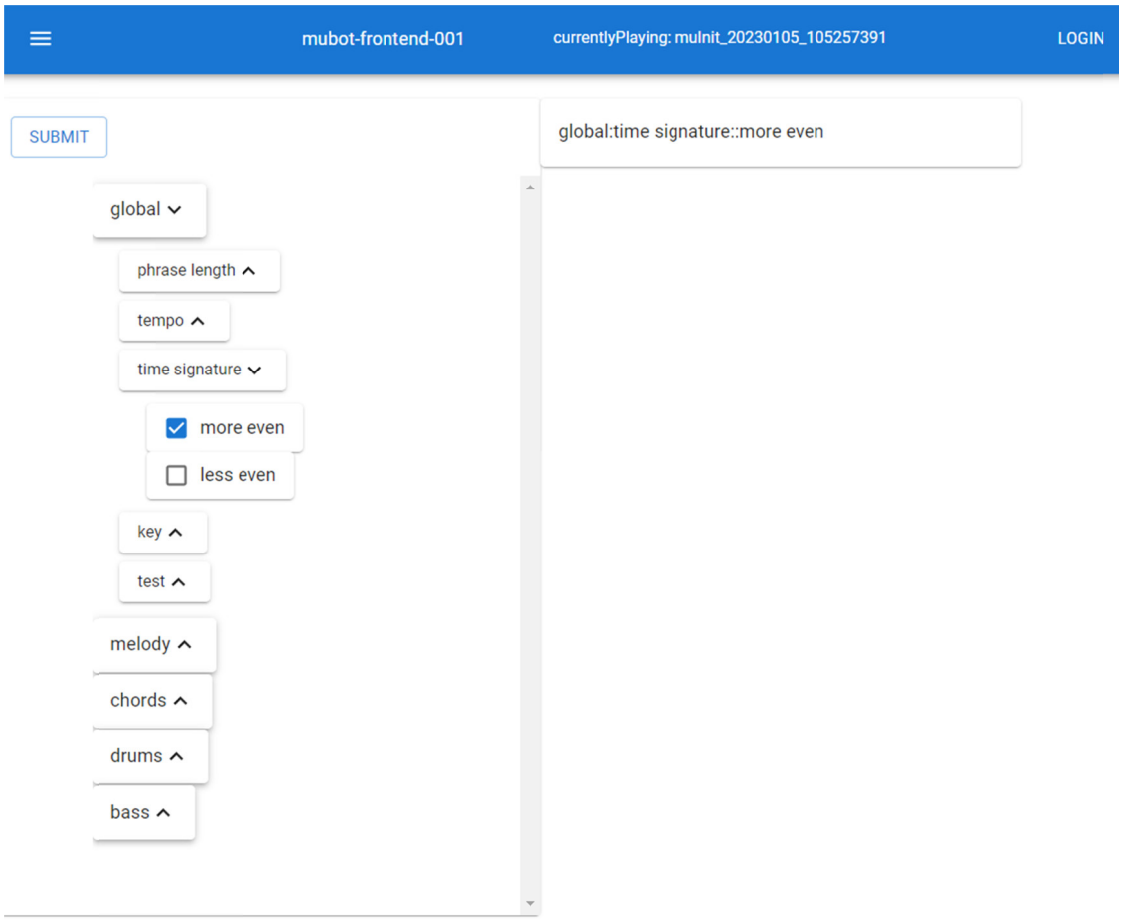


Figure 6: Feedback collection screen. On the left is a nested list of feedback options; on the right is a list of all selected feedback options.

One of the participants had, besides their musical training, extensive professional experience with the development of commercial online audio applications driven by AI. While the nested menu shown in Figure 6 imparts a readily understandable organisation to the material, this participant proposed that the best interfaces would have as little nesting as possible, with everything immediately visible. In particular, they noted that in the design shown it is possible to select two contradictory feedback options. It could be possible to remove one nested level if, for instance, the 'Time signature' box also contained radio checkboxes[47] for the options. This would require some kind of 'disable' option if no feedback is required, but that could be as simple as allowing the selected checkbox to be switched off. The above design proposal would reduce the feedback selection in question from three lines to one.

[47] A group of radio checkboxes allow only one selection.

It was also noted by this user that the structure of the interaction was not clear at all. Their proposal was that an indication of 'Your turn' and 'My turn' might help to direct the interaction and clarify what was happening. It could be helpful to somehow illuminate the structure of the interaction, maybe through an infographic.

These are some proposed statements to test visibility, with five multiple-choice options from 'strongly agree' to 'strongly disagree':

1. During the user feedback step, was it easy to find feedback options.
2. It was easy to understand my role in the interaction.
3. The musical contributions of the agent were welcome.

Premature commitment

It was possible to select many feedback options, adding to the time it took the users to ponder the options. This was especially prevalent at the beginning of the test, when the user was still exploring the options. Later, when events such as submitting feedback became identified as places where a less-than-comfortable randomness could be introduced into the interaction, users at times got delayed by pondering whether they had made the 'correct' selection. This has many implications: too much erratic behaviour and the interaction could be seen as increasingly trivial, and the application as a flawed toy. The large number of options could possibly overwhelm a less decisive user.

The challenge here is to keep an optimal rate of alternating contributions by user and computational agent, either by limiting the options available, limiting the number of selections, introducing a time deadline for selection of options, or creating a previous impression that the 'Submit' button would not break things by allowing the user to revert to a previous position or in some other way take control of the situation. These are not clear choices for the development direction, as they are dependent on the mood and personality of the user. Some users might enjoy challenging changes of direction, whereas others might like more of a sense of control. The please/provoke binary of the computational agent's 'personality' might be something that one could set on the interface and that delivers significantly pertinent behaviour.

This algorithm focuses on detailed feedback with a (relatively) direct relationship with the parameters of the generation algorithms. In some cases, some of the methods used by the

algorithm to generate material generated options that were bewildering even for trained musicians. In this iteration of development this was particularly evident in the feedback related to the generation of the chord instrument part. User 1 suggested that pop-up explanatory notes might help with the initial learning curve. In this case it also threw into question whether the chord-part algorithm used was best suited to purpose.

These are some proposed multiple-choice statements to test premature commitment, with five options from ‘strongly agree’ to ‘strongly disagree’:

1. The information pop-ups were helpful.
2. It was helpful to be able to turn the info pop-ups on and off.

Hidden dependencies

Because the turn-taking co-interaction interaction was not clearly explained or understood, the first few rounds of interaction involved discovering the dependency on this progress mechanism. Thereafter, committing to progressing to the next step of the interaction (in particular where the robot offered ‘creative’ input) quickly became understood as the place where progress could be lost, and options to backtrack, sidestep or further control progress were sought and found to be unavailable.

Role expressiveness

Because the feedback interface relates quite directly to particular input parameters of an algorithm, some explanation of the working of the algorithm could be required. For example, as mentioned before, the workings and terminology related to the algorithm that generated the keyboard part was tricky to understand. In addition, the structure of the interaction was not clear from the outset. A further issue was that users would have liked to have interaction options based on genre or on existing musical examples. While these last two points are noted as significant, both from the point of view of my experience of the composing process and the significant use of genre in algorithmic composing products (Jukedeck, Boomy, any library music collection) and music classification, this is outside the scope of the project, but should probably be articulated more clearly.

Error proneness

As mentioned before, the following issues caused confusion:

1. The structure of the interaction and the user’s role at each step was not clear.

2. The effect of the user's choices was not clearly demonstrated, especially early on when they had not yet developed a sense of how the interaction might proceed.
3. The musical progress of the choices made by the robot was sometimes unexpected and resulted in a change of expectation of the outcome for the user. In particular, there was some frustration when the musical direction changed during the random selection phase after having seemed to proceed in line with the user's expectations for several iterations. This was expected, as the random choice was a particular decision made for this testing, but it was important to note the significance of this on user engagement. The question 'Why did it do that?' or 'How can I go back to a previous iteration?' arose.

In light of the above, for a future iteration of the application the structure of the interaction needs to be more clearly shown, using better instructions and explicitly signposting each step of the interaction. A more difficult goal is to ensure that the development of the musical material proceeds within the musical expectations of the participant by using a more effective way of directing the robot's musical choices by learning the preferences of the user. Lastly, as this was intended as a stimulus to creativity, more effort could be put into allowing the user obvious ways to recover from mistakes or failed experiments in a certain direction. This would be preferable to the user being unable to redirect the interaction due to limited functionality.

Abstraction

This project uses a particular abstraction of the composing process to engage with that process. An expansion of the available feedback options was foreseen for the next round of testing, which likely could lead to more confusion for the user. Currently the feedback options are divided up into five top-level groupings:

1. Global parameters
2. Melody parameters
3. Keyboard-/chord-part parameters
4. Bass-part parameters
5. Drum-part parameters

This choice of instrumental parts, while expedient for the current round of testing, is not necessarily the only collection possible. If, for instance, the chord part included several instruments, they would require instrument-specific as well as part-specific feedback options.

The existing options could possibly be divided up in other ways: the keys, bass or drums could be dealt with as a single rhythm section, but would require related feedback options. Then again, 'rhythm section' feedback lends itself to feedback related to genre, which is currently outside the scope of the project. There are no clear optimisations of the way that the current choice of top-level organisation of these parts might be handled.

Secondary notation

The least obvious part of the interaction is the structure of the interaction and the mapping of feedback to behaviour of the algorithm, although in the latter case the choice of musically trained participants can be helpful, and anything confusing might be best handled by explanatory pop-ups. While the interaction structure is not likely to be common knowledge, it is easily explained and can be reinforced during the interaction with 'My turn'/'Your turn' indications.

Closeness of mapping

The nature of the interaction of this application is not one of a user and obedient assistant, so there is inherently a remove from the cause and effect of interactions with applications designed to achieve simpler and obvious management of data, such as sending an email or paying a bill. By the same token, this applications needs to demonstrate some short-term and longer-term behaviour that is not random. There needs to be an observable relationship between feedback selections and the musical artefact generated thereafter, and in the longer term there needs to emerge an understanding of the user preferences. It is possible that the more effective this last item is, the less likely it is to be noticed.

Also, as music plays over time, there is an inherent delay between interacting with a feedback element and hearing the effect in the output. This also gets longer as the piece of music in question gets longer. At this stage of development of this algorithm, the output is proposed to be the length of one or several phrases, hence relatively short – in the order of seconds to approximately a minute in length.

Consistency

This application essentially uses visually similarly styled lists of options, so consistency is not a major stumbling block.

Diffuseness

An essential aspect of the options available is that they are expressed in words, to capture aspects of the language used to express detailed compositional processes (as opposed to more general ones, such as mood and genre). Currently this is a hierarchical list of musical parts, related parameters and options available to change that parameter, as can be seen in Figure 6. It is possible to reduce the footprint of this model by the design in Figure 7. Here the options are reduced to icons that represent the vector and magnitude of change. This is a better use of vertical screen space, as in this example the tempo parameter feedback options take up only one line instead of three, and have double the number of options available. In terms of diffuseness of language, there is essentially no change, just better use of space and pictorial representation of options.

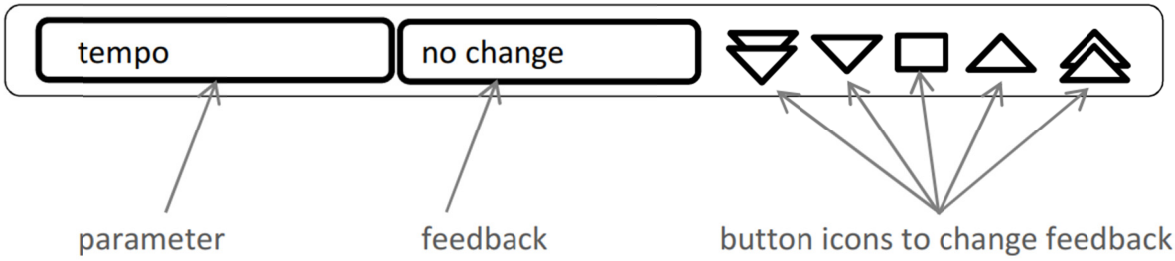


Figure 7: Proposed new layout for feedback selection. The buttons, from left to right, would change the feedback text box to 'much slower', 'a little slower', 'no change', 'a little faster' and 'much faster'. The number of feedback options would be dependent on the parameter, but would no longer require a new line for each option, as was the case in the user interface displayed in Figure 6.

Hard mental operations

One of the most significant things which seemed to slow down the rate of interaction in the early user tests was seeing each step as hugely significant, as opposed to an incremental movement towards a more acceptable generated artefact. In this version of the application, part of this stems from the user learning points at which the algorithm would 'mess everything up' or have no option to proceed in a way that the user might want. This tended to result in either no feedback options being perceived as useful, or overthinking the selection of feedback. Solutions to this might be an obvious and easy-to-use undo function, more feedback options (even though this is a theoretically infinite list) and an algorithm that is better at reading the user preferences and being able to generate, with good probability, a continuation of the interaction acceptable to the user.

Provisionality

Provisionality in this context describes the opportunity to try out possibilities before committing to them. While the idea of a turn-taking interaction to develop an idea seemed to implement such a quality, it was not seen this way by the users. While from a designer's perspective, a series of interactions might be seen as an exercise in provisionality resulting in the final outcome, the participants placed much more emphasis on each step demonstrating an improvement. As there were limited options for backtracking in these tests, the participants quickly learnt that each step forward was to some degree a step into the unknown, especially when it came to the robot variation. This became a particular barrier as the participants became invested in the musical direction of an idea, but it was less of an obstacle at the beginning of an interaction, when the capability of the application or a new idea was being explored.

While the algorithm in its current limited-feature configuration offered some instances of engaging development, the first response of participants to a step that was not pleasing was a desire to backtrack to a previous acceptable version. When this backtracking was found to not be available, the desire arose to review material before proceeding to the next step of the interaction. An approach might be to use different types of interaction[48] with the musical material, or to provoke the human user to expand their preconception of the end result towards which they are progressing.[49]

Progressive evaluation

This interaction asks for regular progressive evaluation of the progress of the outcome. However, because the evaluations are used as a part of the steering mechanism, some frustration arose, as the causal connection between feedback and output was often not clear. Users got to the point where the material started showing some kind of promise, and this is the point where the user's expectations of the system (be that musical direction or expected

[48] This project focuses on feedback specific to a single algorithm parameter. The following were also considered as interaction approaches: genre- and/or mood-based interaction (or indeed any quality less tightly linked to a single algorithm parameter), musical examples selected by the user or played input by the user.

[49] This is a tricky area and possibly a study in its own right. In the absence of immediate inspiration, a user might be open to more possibilities and the use of such an algorithm to explore unknown musical spaces would be interesting. Once something of interest has been found, the user would likely want to progress the idea along lines that they are, consciously or unconsciously, more familiar with. In the former case, an equal co-creative partner would be more easily accepted by the user, but in the latter case the user would want the computer to play more of a creativity support tool role. It is also in this second role that the shortcomings of the application used for early user testing became more apparent. Both users got to a point where they had something they liked, and there was no way to proceed.

functionality) suddenly grew exponentially. This suggested that the interaction had progressively deepened their engagement to the point where functionality had to change to meet the expanding scope of user expectations. For me as the designer of the system, two questions arose:

1. How to prolong this engagement?
2. How to design a system that accommodates this point where the human will outstrip the capabilities of the computational agent to deliver engaging options and variations?

4.4. Step 4: Quantitative alpha testing

The qualitative interview-based alpha testing, even though there were only two participants, gave significant focus to adaptions to the application. It was decided that a quantitative approach would be simpler for the next stage of testing, as it was felt that it was possible to create clear questions and response options to capture the efficacy of the interaction using a multiple-choice questionnaire.

The development proceeded as follows: I updated the application with ideas from the alpha testing stage, in terms of both behaviour of the music algorithm and the user interface. Then I finalised the system for deployment to a cloud internet provider. I developed the questionnaire to reflect the research goals and then implemented it using an online survey tool.

4.4.1. Updating the musical behaviour

No major additions were made to the musical behaviour of the algorithm. There was some debugging to be done with the variation system to confirm that it was operating correctly, as it was sometimes hard to tell in the interaction whether it was not just making random choices through some coding error. This was an area where I felt that I could have done much more research to more fully understand this sparse data decision process. However, I also had the sense that such research might turn up no valuable insights at all, and might just be an attempt to mitigate some personal insecurities by attempting to get more quantitative insights on the emergent behaviour of this process in the application as a whole. Of course, time pressure dictated that such a study was not possible, and the behaviour was not so erratic that the interaction failed consistently because of it. The decision was therefore made to continue with it as it stood.

The tempo and the user interface

While the system was designed to learn user preferences, I had only implemented these abilities with a session-specific scope. A user would log on, start with a random output and then share turns with the music robot to develop the material, over the course of which the music robot would record the choices and analyses of the user and consider them in later variations. There was no mechanism at this stage to revisit a previous session once the user had logged out, as I did not foresee the participants spending more than one session on their involvement with the project.

This sidelined the possibility of the robot learning user behaviour on a scale longer than a user session. A case in point emerged around the issue of tempo during the generation of the robot's variation. Tempo was a parameter available for adjustment, and it was found that an adjustment of the tempo during an interaction was most likely to break the sense of compositional development, and it happened regularly enough during the alpha phase of testing that I added a hack in the software to disallow the robot from ever adjusting the tempo. A resolution to this came when I added the ability for certain parameters to be locked by the user, meaning they were off limits for the robot's generation process.

4.4.2. Interaction and user interface

I added more feedback options to be used in the interaction so that each part generated had a sufficient if not equal representation in this respect. Feedback options were divided into those relating to global parameters, melody, chord part, bass part and drum part. Screenshots of this can be seen in Figure 11 to Figure 15.

The interaction structure was finalised as follows for this iteration of testing:

1. The user would log on, and read instructions.
2. A randomly generated phrase would be generated.
3. This phrase could be liked or disliked, and the user could proceed with the interaction or have the robot generate another random version. These two actions were not connected. The user did not have to like the clip to proceed, but any like or dislike captured would contribute to the robot's understanding of user preference.
4. Once the user had decided to proceed, the user was offered a set of options to offer feedback. These would be selected and submitted.
5. Once submitted, the user variation would be generated and played back.

6. The user would be asked for a like or dislike of the output, and given the option to proceed, try another variation of the feedback, or go back and update the feedback and generate another option. Trying again would generate another option and play it back, whereas updating feedback would return to the feedback page for adjustments and a new output after resubmission.
7. On proceeding, the robot would generate its own variation of the new clip based on user preferences captured thus far, and loop back to point 3.

While the core interaction sequence had not changed, there were extra options added for backtracking in the process. The addition of the log-on process was the only extra step, and related to the software deployment rather than the interaction itself, which is addressed in the next section.

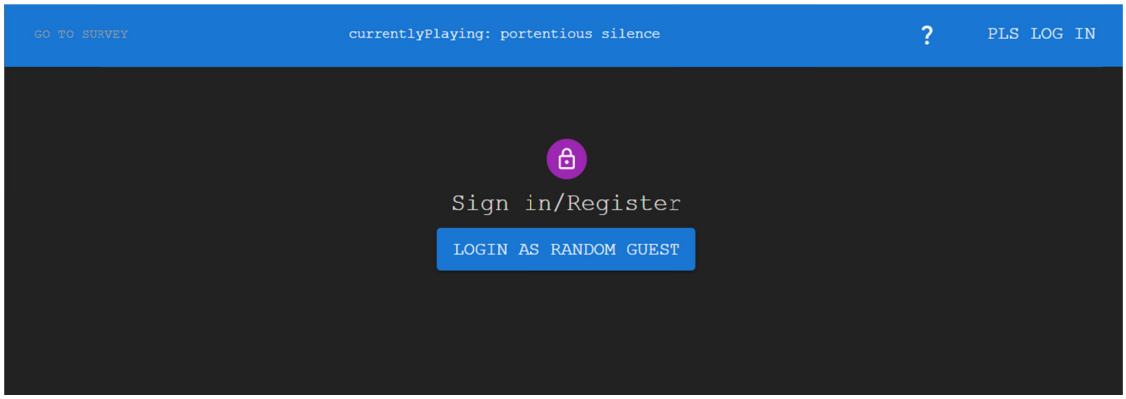


Figure 8: Log-in page. The blue top bar of the webpage contains global information and does not change based on the content of the black main page. From left to right in the top bar we can see a (currently greyed out) link to the online survey, the name of the currently playing track (in this case, no track is playing), a '?' symbol which links the user to the instructions and a 'PLS LOG IN' exhortation which is replaced with an auto-generated user name once log-in is complete.

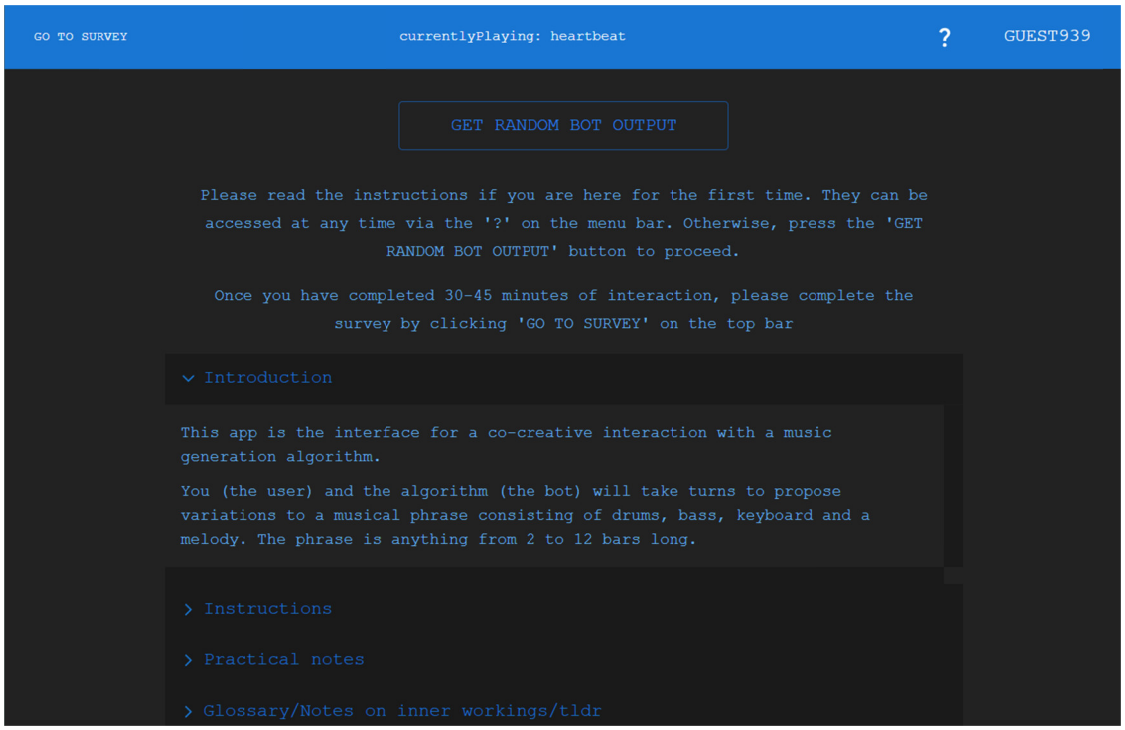


Figure 9: Instructions and button to initiate the initial random clip

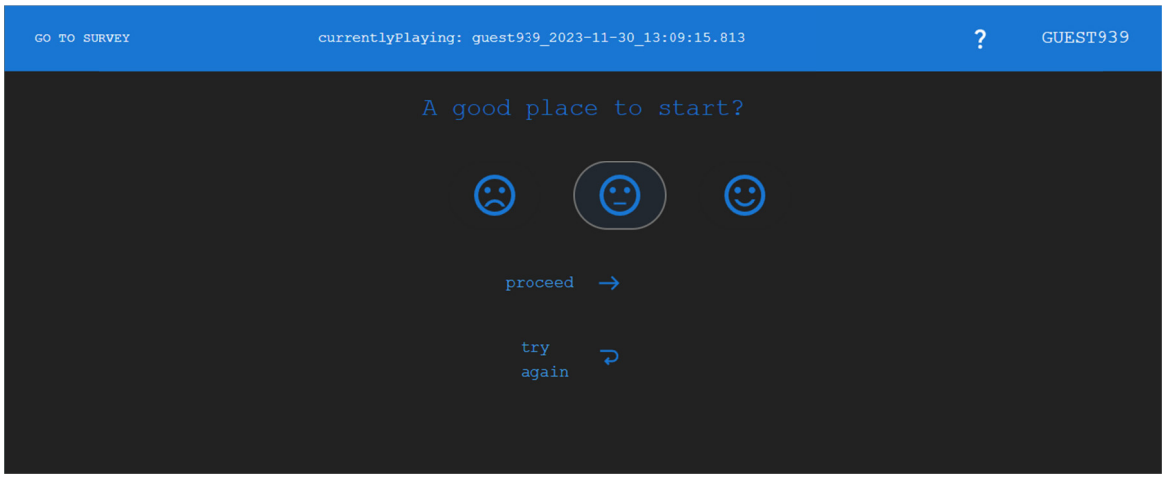


Figure 10: Sentiment and progress options for the initial random clip, which would have begun playing after the 'GET RANDOM BOT OUTPUT' button was pressed on the previous page

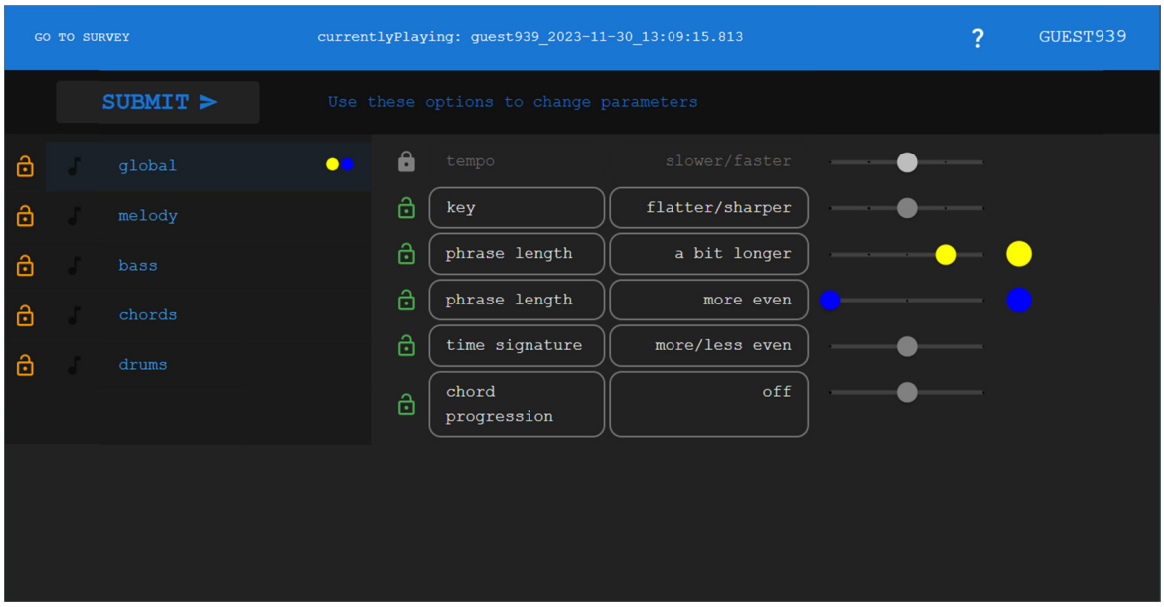


Figure 11: Feedback capture page showing options for feedback for global parameters. The selector for each feedback option has become a slider with discrete options to avoid the possibility of capturing redundant feedback (e.g. both too fast and too slow) and to save screen space. Selected options are coloured, and also registered on the sidebar to give an indication of where selections have been made. Tempo has been locked. The 'SUBMIT' button can be seen top left; this progresses the interaction to the generation of the variation based on user feedback. In retrospect, the chord progression option is misleading: 'Off' means the chord progression will not change, rather than there being no chord progression.

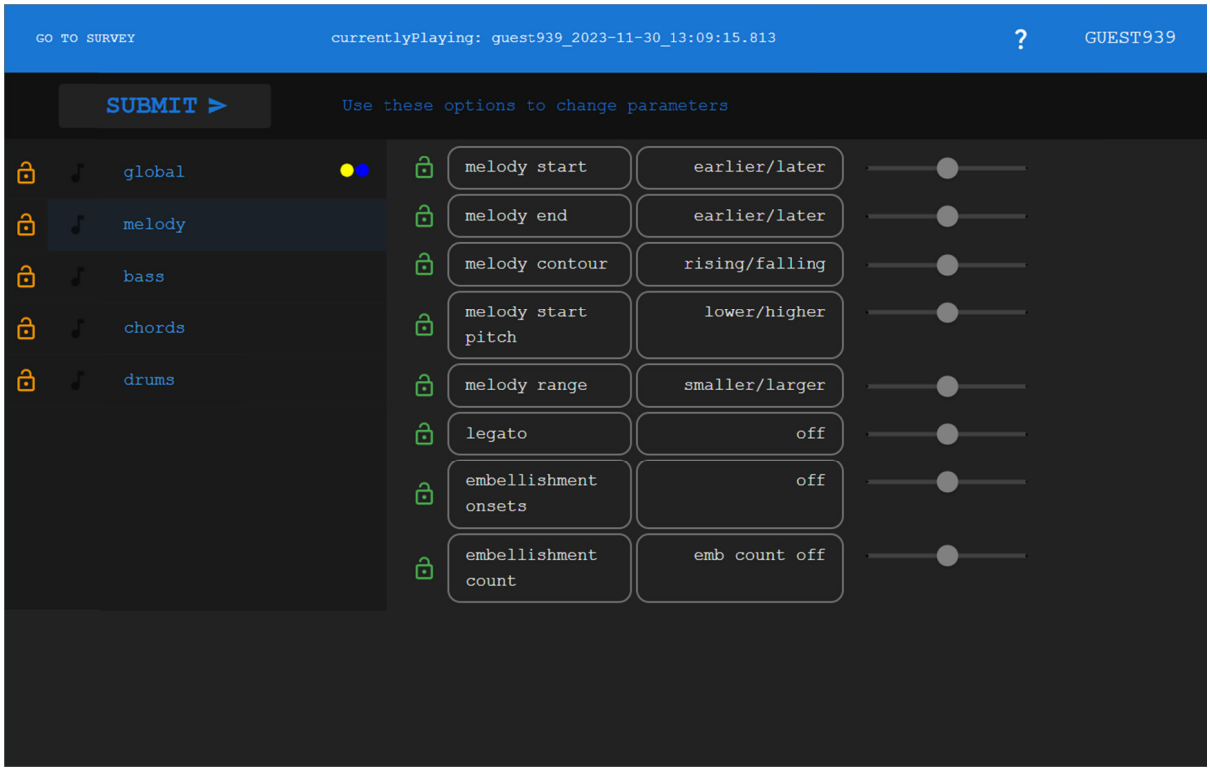


Figure 12: Melody options

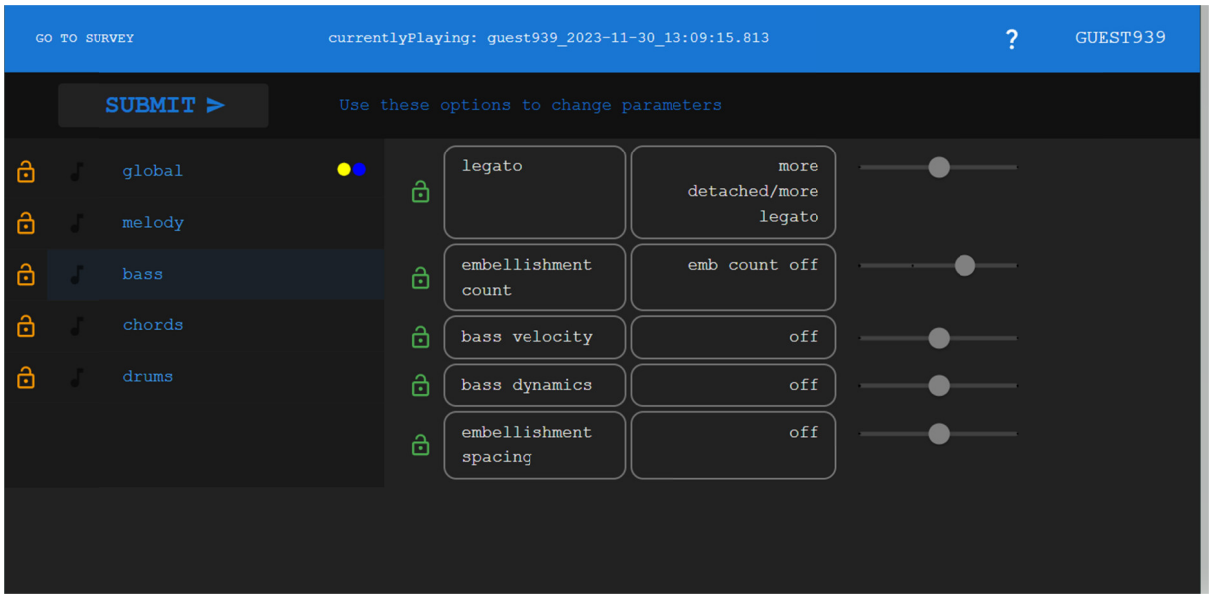


Figure 13: Bass options

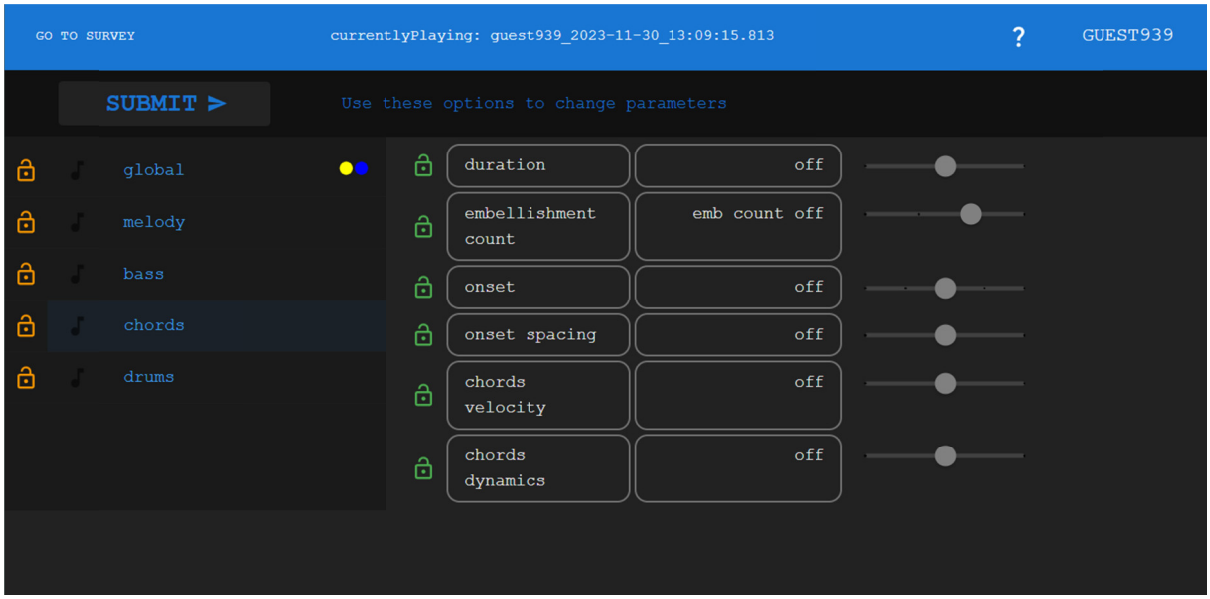


Figure 14: Chord-part options

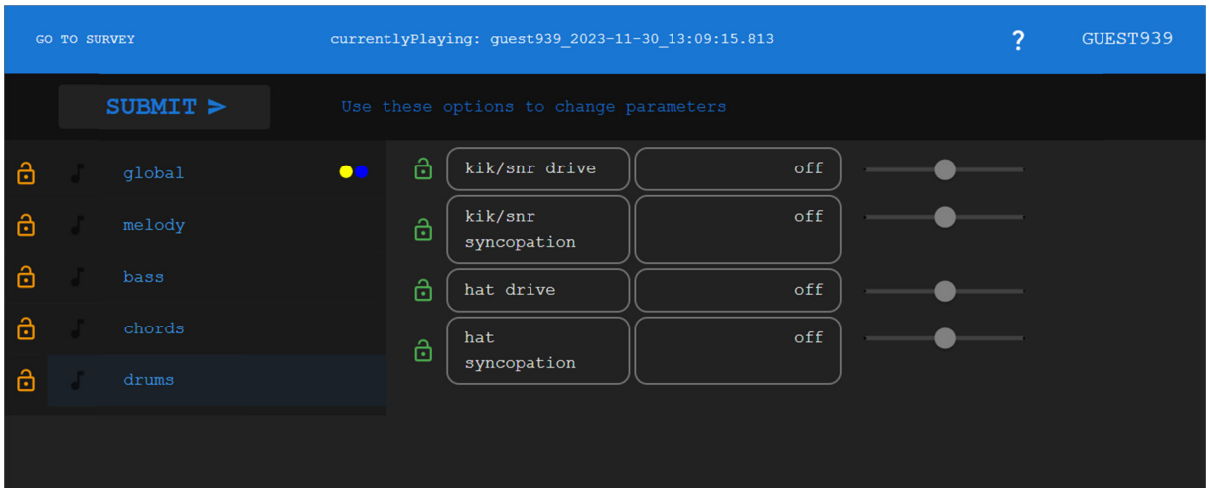


Figure 15: Drum-part options

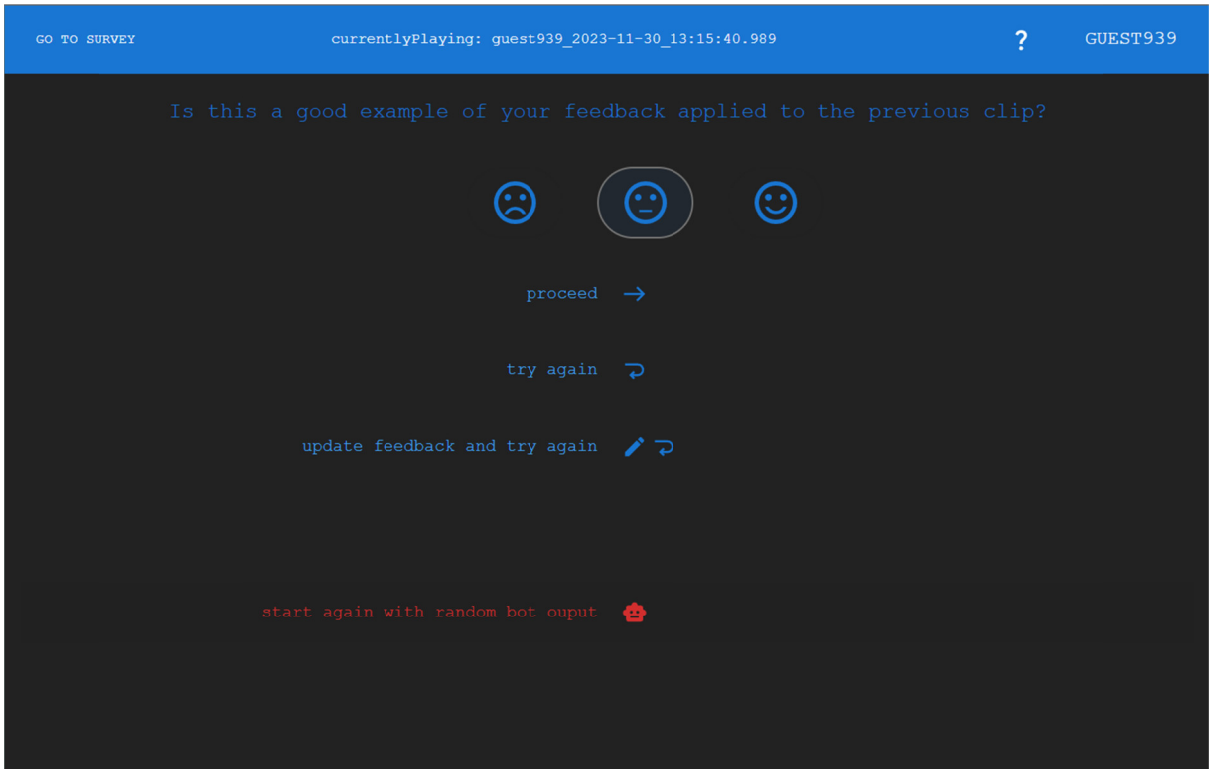


Figure 16: Sentiment capture and progress options for the user variation. The new variation will play after the 'SUBMIT' button from the previous screen has been pressed. Once the 'Proceed' option above is selected, the robot's variation will be generated and played, and the next page will be loaded. Also note the escape option in red at the bottom, which will generate a random new clip and return to the beginning of the interaction, just in case the interaction seems to be going nowhere.

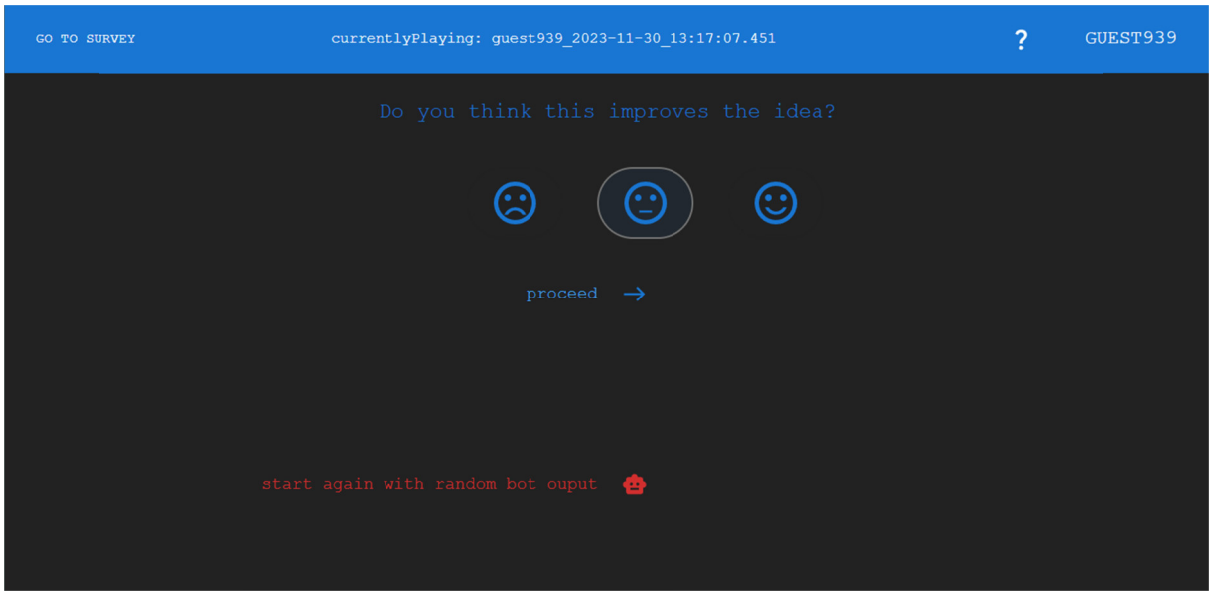


Figure 17: Sentiment capture for the robot's variation. This page returns to the user feedback capture pages in Figure 12 to Figure 15.

4.4.3. Finalising for cloud deployment

The most significant consideration when preparing this application for cloud deployment was putting in place an architecture for differentiating between multiple users, as this had never been a consideration during the testing process. While a fully fledged website might offer options for a user to register, log in again, log in via some extant social media account, recover their password and so on, for the purposes of the testing envisaged, I simplified the log-in to only the option of logging in to a new session as a guest user.

For security purposes, the user's session was identified in the form of a non-sequential Universal Unique Identifier number, which would resist simple attempts to access data from another user's sessions by passing the 'next' number in a sequence to the server. Furthermore, due to the security requirements for running WebAssembly code for the audio playback in a browser, all connections for such a deployment had to use secure https:// connections. Under the circumstances, as there was no sensitive personal information captured on the site, I deemed this sufficient security to protect the integrity of the data for my study.

In terms of the back-end, I had to separate the persistent memory of a session to a database rather than keeping it in memory, as had been the case during development. The Spring Boot framework, which I used for development of the back-end, is expressly designed for such usages and it was relatively easy to implement this change after the initial learning curve.

The last piece of the puzzle for internet deployment concerned the playback of the generated material. During development I had used Ableton Live as a playback engine and this had worked well, including by allowing generated material to be included in workflows beyond the scope of the software. However, in a public deployment this was not possible, as Ableton could not be automated to render out tracks. There did not seem to be an obvious replacement for Ableton in such a scenario which could do automated offline rendering of musical material to a standard audio playback format. If a replacement was found there remained the small issue of hosting the rendered tracks and downloading them for playback in the browser. This was unlikely to be problematic as hosting and streaming or downloading of audio and video media is a well addressed problem in web development.

The idea of addressing an audience who were Ableton users arose as an option, which I foresaw would involve changing the current communication from the application to Ableton so that it emerges from the front-end of the application in the user's browser rather than from the back-end on the server. The setup of would have several potential points of failure, as it would involve downloading a special Ableton session and then hoping that the User Datagram Protocol (UDP) that was transferring the messages from browser to Ableton would work as expected. As I later found out, this idea was not viable. For security reasons, no browser allows transmission of UDP messages.

I was not sufficiently versed in these communication protocols to foresee these problems in detail, so the abovementioned approach was replaced by the idea of a browser-based playback engine. Most rudimentary of these would be a browser-based MIDI playback module. What should have been relatively simple turned out to not be so. MIDI playback has not had nearly the same attention paid to it as audio playback in a browser, and I would have to contend with the well-known potential for MIDI playback to be accompanied by poor sound quality.

Two options remained: Cycling '74 RNBO,[50] a new technology which allowed for the implementation of Max MSP in a browser, and Csound,[51] a well-established music synthesis environment which my research indicated might be web compatible. As it turned out, the clinching argument was cost: RNBO was expensive, new, and did not seem to be perfectly

[50] https://cycling74.com/products/rnbo
[51] https://csound.com/ is the official page, but https://kunstmusik.github.io/icsc2022-csound-web/tutorial1-getting-started provided the most significant help in getting going with Csound on the web.

suited to my needs, and Csound was open source, reasonably easy to understand (at least to me) and available as a WebAssembly implementation for use in a browser. I must say the learning curve was interesting, as there is a small and delightfully quirky community currently active around this software, and getting the correct information involved some detective work. But the WebAssembly implementation worked very well and I consider the decision to go this route fortuitous, both for this project and for the use of this software for future research. This playback engine required no setup from the user. Playback information was in symbolic form, which meant that file sizes were small, increasing download speeds. And lastly, as it was based on a technology (WebAssembly) designed for running within browsers, there were no platform-specific issues. I tested it (non-comprehensively) and found it worked on PC, Mac and Android, using Chrome, Firefox, Safari and Microsoft Edge browsers.

4.4.4. The online survey

The online survey, which is detailed in Appendix B:, was structured in two sections. The first established the musical background of the user and their prior experience with generative music tools. The second section broke down the user experience regarding the understandability and effectiveness of the control offered by the application, and the sense of musical quality and continuity that emerged from the interaction. The survey was implemented as an online multiple-choice questionnaire using the Qualtrics[52] survey tool, with a link on the application's webpage.

The planning of the survey questions was another chance to interrogate the details of the research questions. I was advised to keep the survey as short as possible, so I limited myself to the 30-question limit that Qualtrics applies to free accounts. I found this limit to be helpful, as I was forced to trim down the number of questions proposed initially, and I did not feel that the scope of the questions was compromised by this editing.

4.4.5. Alpha test results

Disappointingly, the online testing did not attract a statistically significant number of participants. This led to a re-evaluation of the methodological approach and a decision to examine the use of the application in some personal composing projects.

[52] https://www.qualtrics.com/

4.5. The interaction in a personal composing project

As described in 1.5, the landscape of broadly accessible AI-driven tools has changed considerably over the course of this project. There exists also a large amount of both positive and negative biasing as various players in and around the industry defend positions and agendas. Chomsky and Polychroniou (2023) in particular highlight the fundamentally plagiarist nature of generative AI, asserting that it can offer no new insights into the underlying understanding of the material on which it is trained. They staunchly defend the uniqueness of human intelligence. Hinton's standpoint disagrees with this (Heaven, 2023), citing these algorithms' ability to deliver a convincing simulation of intelligence as a significant insight into the workings of intelligence. What is perhaps more telling is that Hinton's attitude to this has changed: from enthusiasm about the technology to fear of the wider negative impact on society (Bitar, 2023). Lecun, another seminal developer of AI technology, dismisses the negativity associated with AI (Lecun, 2023), asserting that open source development of this technology is preferable to tight control by government and big technology companies. It is not hard to infer that the devil is in the details in such a situation. What is hard is to conclusively determine the direct or indirect effect of this landscape on the online data gathering for this project. However, participation was low, and did not furnish conclusive results.

At the outset it seemed that approaching the design and testing from an HCI perspective was a good approach. However, having arrived at the inconclusive result of the proposed conclusion at what appears to be a liminal moment in the development of AI, it seemed that the methodology for gaining insights needed further work. While the research area of new interfaces for music expression (NIME) may be seen as a subfield of HCI, there are arguments to suggest this is not so clear cut. Rodger et al. (2020) present an argument for the limitations of the HCI perspective for an electronic music-making interface, arguing that such an instrument needs to be treated as an open-ended ecology of instrument and instrumentalist rather than the more limited user-and-application relationship foregrounded in HCI research.

Through the process of developing and deploying the application as an HCI entity, the limitations of the available interactions were exposed. These limitations embedded themselves in the deployment of the application, with no possibility for a user to move the interaction beyond the confines of interaction on offer. In such a case, one either needs to

present a diverse working ecosystem to the user or offer the possibility of intersection with other ecosystems.

Furthermore, the time required to engage with the instrument in this manner was considerably more than the time allocated for an HCI evaluation. This precluded a study involving this much greater commitment for participants, which would move the project beyond the scope of the research format. I imagined a project where I, as a music practitioner, would employ my detailed expert knowledge of the system to attempt to extract material from the system for longer-form pieces. So I engaged in an iterative composing experiment to see how I might use or subvert the technology as it existed to achieve something beyond an engaging interaction and a compelling yet undeveloped musical idea, which seemed to be the limit of the usability of the application.

4.5.1. Constraints for a composing project

The browser-based app is designed to support a real-time co-creative interaction focused on compositional ideas foregrounding note choices, with the limitation of a phrase-length[53] musical idea. As a result there is no option in the browser app to vary sounds, edit individual notes, edit ideas into longer arrangements or render an audio output for playback outside of the application. For these new experiments I reverted to the set-up I used during development, where the system was deployed locally on my personal computer and output was injected into Ableton Live, which was used to realise finished works.

An important temptation to avoid in preparation for this composing project was to engage in lengthy new software development, but some adjustment was required. I added an option to use Ableton Live as an output platform to the most recent version of the application, and I added an environment variable to expose this option or not. This would allow the project to still be deployed as a publicly accessible browser-based application using the original playback method. For my own purposes, hosting the app on my local computer, I could

[53] This 'phrase-length musical idea' is the output from an algorithm that uses parameters meaningfully related to the construction of a melody that could be seen as a single musical statement, i.e. a phrase. These parameters include (but are not limited to) the length of the phrase in bars, the start and end point of the melody within this bar range, the melodic contour within the start and end points, and so on. While the algorithm is likely to create a melodic idea that could be seen as a single musical statement, there exist available parameter extremes that might make the construction appear to be of a longer, or indeed shorter, type.

change this option and have access to playback from Ableton. This was done in an effort to not bring parallel versions of the software into existence.

In this Ableton playback mode I also allowed for the backup of parameter files directly to the file system. In the online version, these files were saved in a database, but this was not the easiest way to access the files if I wanted to read the content. I considered the possibility of being able to edit these files, but the job of keeping database and file repository synchronised seemed like exactly the kind of new software development I should avoid. I implemented parallel saving of these files both to the database and to the file system. The database was the baseline backup for data from the system, but saving to the file system reduced the access time to this data. I could not predict exactly what might be needed in this regard for the composing experiments, but I had found this format useful during early development prior to the application being set up for web deployment.

During the course of the experiment I did make several new feedback options to shortcut variation processes that were holding up the progress of composing. Gratifyingly, the software framework I had developed allowed this to be implemented rapidly. These feedback options often took the form of ‘just do this’ imperatives rather than ‘there is maybe too much of this or that’ suggestions, which made them even easier to implement. While I did not discount the possibility of editing generated material to my liking in Ableton after it had been produced, this effectively removed the possibility of the app offering further processing of the material, as it no longer existed as a parameter file. This points to a valid use case for the as-yet-unimplemented analysis tool to convert any musical artefact into one of these domain-specific parameter files.

A second addition was the ability to open a previous session and then select a clip from the session in question. Thereafter, the clip could be opened as user or robot variation and injected into the playback system. Labelling it as such was not significant for the clip itself, but it dictated the next step. For a robot variation, the clip was opened and then the user was presented with feedback options. For a user variation, the next step would be a variation of the clip by the robot. Furthermore, the clip could be opened in the current session, or it could be imported into a new, blank session. Using the existing session would allow the robot to use the session’s existing sentiment database for generation of variations. A blank session would start this afresh.

To highlight the contribution of interactive composing to such a project, I developed a list of reflection points to frame my use of the application:

- What combinations of the options available in the application were used for generating material for a longer-form composition?
- How was material selected?
- What manipulation of the material was used outside of the generation process of the application?

4.5.2. Some starting points

I found myself in a creative area that I had not considered in much detail before this point. I had thought of the application expanding to generate longer works, but in a way that the application would be extended to handle the editing and compilation of generated material. With no prospect of this, I needed to consider the options of using the algorithm in ways not originally intended, and I would have to consider at each step what would be the next.

I began with a short list of general ways that experimentation might proceed:

1. Ten randomly generated artefacts were generated, from which I would select starting material.
2. Having selected an initial idea, I would use the co-creative loop to develop usable material.
3. Output material would be edited and arranged in Ableton Live.

Besides this general approach, the following more detailed questions had arisen.

- What was the impact of the sentiment database on generation? I planned to test the difference between the use of the co-creative loop in a clean session compared to one with significant captured user sentiment. A ‘session’ in the application defines the sentiment database that the algorithm accesses. A new session starts with an empty database.
- What impact would variations in the instrumental sounds make to the perception of the generated parts?
- Was there any possibility to subvert the behaviour of the algorithm, such as exploiting known errors?

To document the process, I proposed to use a video screen capture of each session and record a voiced description of the evolving progress and goals of the interaction.

4.6. The first composing experiment

In preparation for this first experiment, I added four new feedback options. The need for this emerged while I was trying to get a melody to start at the beginning of the phrase. I remembered that the feedback options for moving the phrase start and end points favoured moving them only a small amount. I thus added options that would allow both these values to be set to their earliest and latest extremes, which is more akin to setting a parameter than proposing some feedback. This was a level of control that earlier test results seemed to point towards for a certain point in the interaction, giving the user more control when the computer was not capable of a creative contribution that progressed the development of the artefact.

The initial idea, as generated, can be seen in Figure 18. Some points should be noted:

- There is no melody. Due to the short phrase length (two bars), none of the options for melody structure tone placement fell within the area bounded by the phrase start and end points, which are expressed as percentage positions. Available structure tone positions are given by the structure tone spacing value. Perusing the parameter file gave the structure tone spacing as 2.0 (where one bar = 1.0). In a two-bar phrase, this places the only option for a melody structure tone on the first beat of the first bar. With phrase start and end positions of 0.181 and 0.414 (expressed as percentages of the full phrase length, and indicated by the bracket under the first bar of the melody), there was no scope for placement of a structure tone. Therefore, no melody was generated.
- There is a dropout of accompaniment parts in the last bar, except for a quarter-note hihat pattern. This arrangement technique was used for the browser-based playback to clearly indicate the beginning of the phrase, in the absence of any visual confirmation thereof. In that role it was effective, but here it gives the playback a stop-start feel, as the dropout happens every second bar.
- The keyboard part contained the result of an error in the algorithm code, which in this case had quite musical results. The part was constructed using a structure tone and embellishment model, with the structure tones being positioned using a choice of beat strengths (and marked with an ‘st’ in the figure) and with embellishments preceding these structure tones if there was space available so as not to overlap with a previous structure

tone. The pitches for the structure tones were derived from the prevailing chord, and the embellishment's note information was derived in relation to the structure tone that was being embellished. In this case, the embellishments were diatonic steps upwards. The error was that the algorithm mistakenly always used the key of C to derive these notes. This would result in entirely diatonic notes if the piece were in the key of C, but would result in increasingly chromatic results as the key became further removed, with sometimes unmusical results, especially when there was a clashing note in the melody. In this case the musical result was quite interesting, and there was no melody (as mentioned before) to clash with.

- The clip worked coherently together. It sounds more like it is in three rather than six, but the notation works well enough, somewhat like the notation of much Latin jazz that is in cut common time to avoid an excessive number of syncopated sixteenth notes.

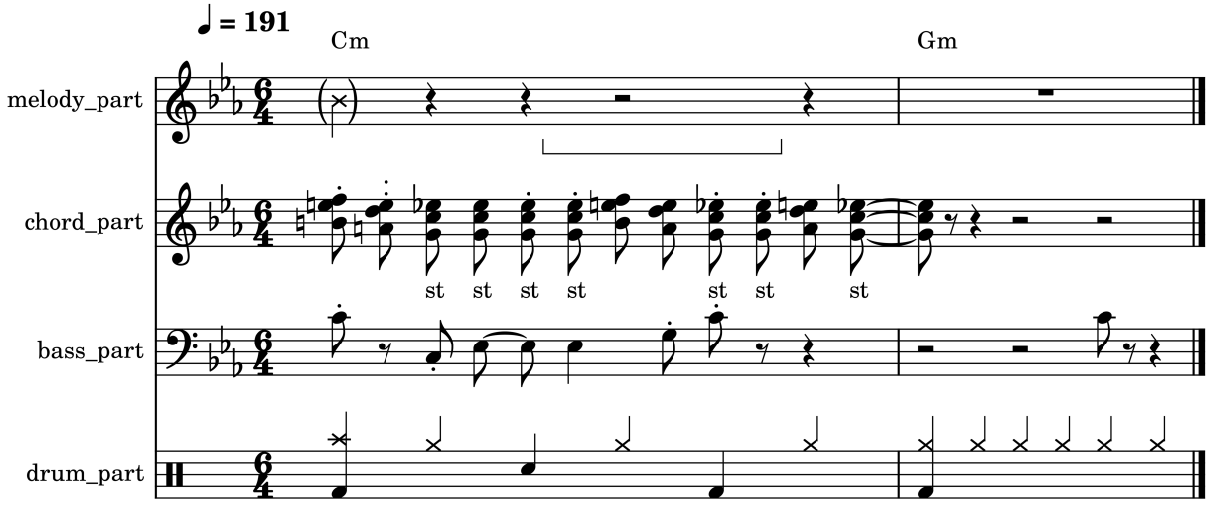


Figure 18: Initial idea for the first composing experiment. The bracketed melody note represents the only available position for a melody structure tone based on the structure tone spacing parameter. The bracket underneath the melody staff represents the approximate position range that the phrase start and end values would have accepted for positioning a structure tone. Therefore, no melody was generated.

4.6.1. Melody material

Besides the five general areas of exploration proposed in 4.5.2, an explicit first initial goal with this idea was to increase the length of the phrase to see what melody emerged when viable structure tone positions became available. The first session took about an hour and the following points emerged when reviewing the desktop video:

- There were many new feature requests, as I attempted to realise a musical goal and found myself thwarted. What was interesting to observe in later sessions was that I found myself

dwelling less and less on changing the software, possibly because I found new ways of moving the material forward. This demonstrates a mindset limitation as much as it was an actual limitation of the application.

- While the new working methods that emerged were not necessarily those I envisioned during conception, they brought welcome new life to the application as a creative tool.
- There seemed to be no difference between using the extant session and its sentiment database, or using a new session without one. This is something that needs testing over several ideas to see if developing a (relatively) large sentiment database makes an observable difference in the long run. Sentiment capture in general began to annoy me: the more I pondered the effect of each like or dislike, the more it seemed that the sentiment had no nuance, and it was difficult to identify something particular (like a melodic phrase or rhythmic pattern, or an interaction between parts). The question began to arise as to whether I should like a clip if it contained something (even something small) of merit, or if I should, in the same way, dislike it for just one issue. Also (I knew) there was little consideration of the questions presented by the user interface (UI) during sentiment capture. A like, neutral or dislike response to the question 'Do you think this improves the idea?' could be interpreted in more detailed and insightful ways. In particular, if it was considered that this sentiment might only refer to the material that has been changed, then already one has more parameter-specific feedback without any added complexity in the UI. Instead, every parameter of a clip becomes labelled with the overall sentiment, when those parts might actually be of merit in and of themselves. And then one must revert to the observation that sentiment seemed to make no difference at all anyway.
- There seemed to be no new material to bring to light after the first thirty minutes or so.

The result of this first foray did not point in any particular direction for the piece I was developing. I organised the clips one after the other on a timeline and, after listening through this rough arrangement, the general consensus was that there was still too much information. While there were variations of a kernel idea, there was no baseline kernel idea and there was no set of tools to arrive at a satisfactory initial idea from which the rest of the arrangement could emerge. At this point I would have liked to get the score output, sit down at a piano and play till all the extraneous ideas had been filtered out.

On the positive side, there were plenty of potential ideas, so the rather haphazard walk-through variations that the robot was doing were not without their uses. It does seem, though, that turn-taking co-creativity is not the best methodology to fine-tune an idea for further development, especially with a computational agent whose contributions are just as likely to head off in a compelling, but new, musical direction as they are likely to be bewildering. To proceed with this project, the second phase of development of the kernel ideas was realised using the Ableton arrangement and editing tools.

The clip for the chorus was arrived at first, as seen in Figure 19. The edit involved trimming off two bars at the beginning of the original idea so that the melody began at the beginning of the phrase. This had been a goal during the co-creative interaction, and had been achieved using feedback, albeit with some other changes to the melody which I considered less effective than the version which I edited by hand.
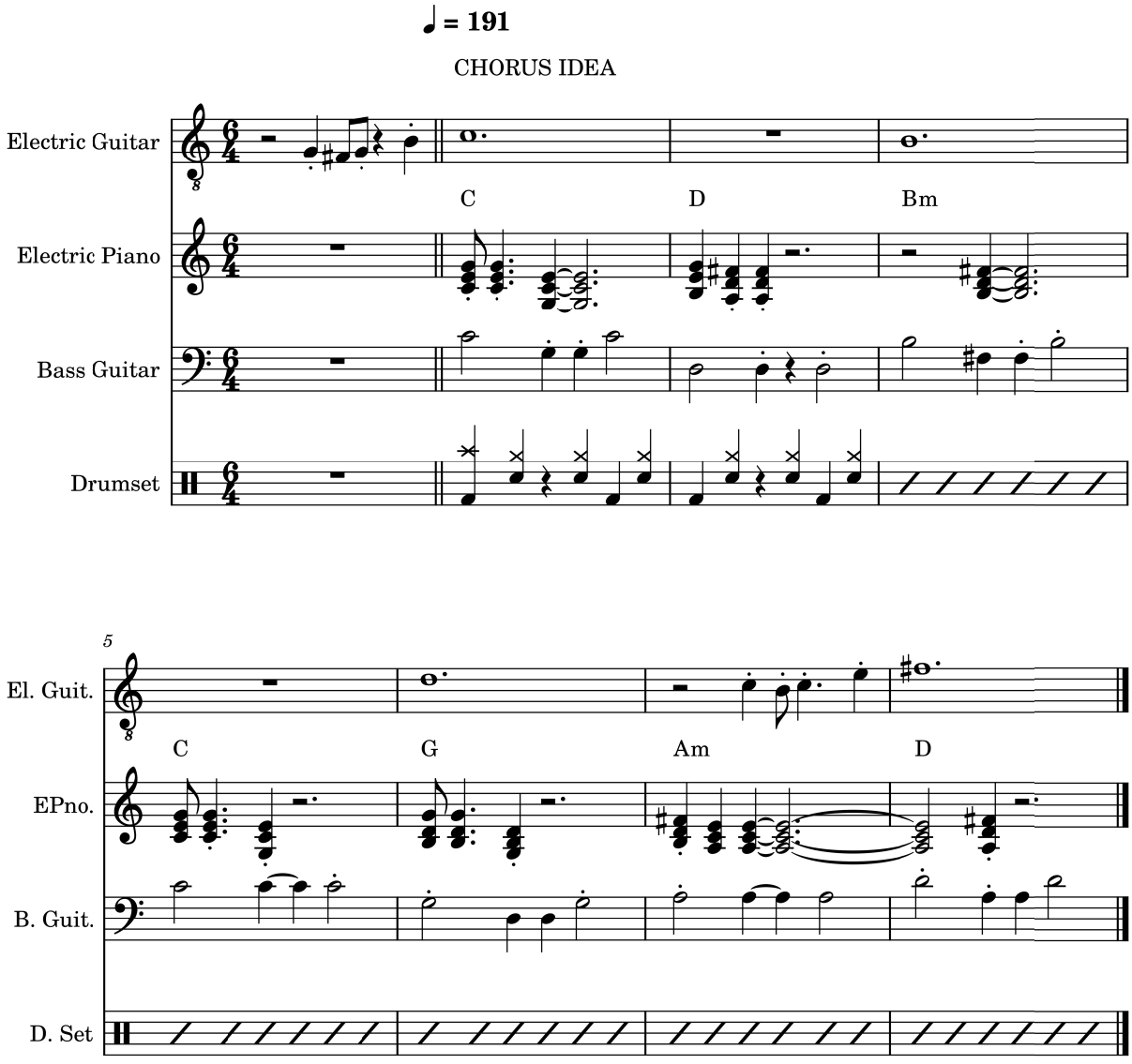


Figure 19: Chorus idea

The chorus idea required an idea for the verse that was simpler. This requirement only became apparent during the editing process, so it was not explicitly sought during the co-creative interaction. I edited down the various material to arrive at a simple two-note melody, as in Figure 20. There were several variations of this idea, with combinations of more structure tones, more embellishments and more syncopation of embellishments. While it would not be impossible to achieve this idea using the robot, the explicit composing strategy used here, which was to reduce the number of notes to a minimum and remove quirky syncopation, was not an explicit choice in the application and therefore much easier to achieve by directly editing the notes in Ableton. What is also significant is that the need for this idea was not clear during the process of exploring options with the robot.

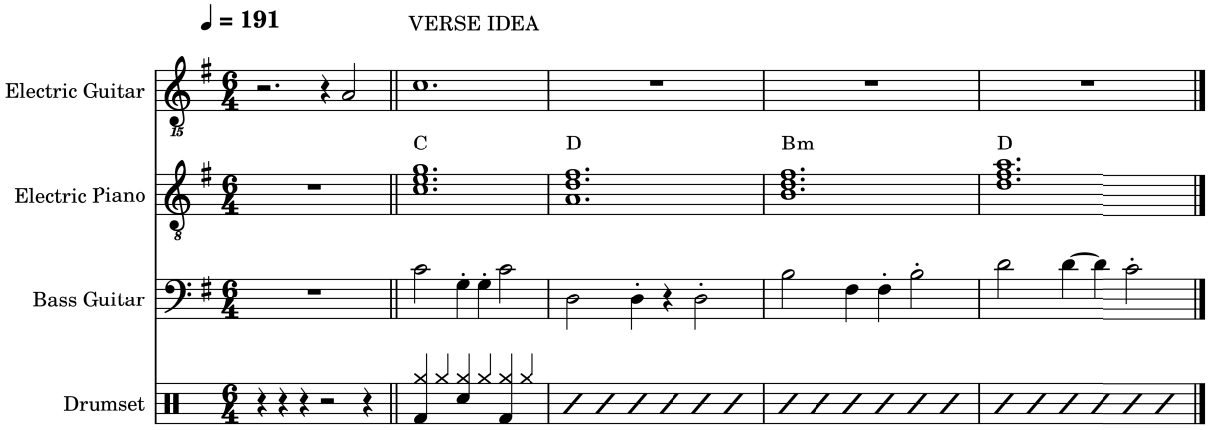


Figure 20: Verse idea

The verse idea was expanded by repeating the idea four times and varying the melody, as in Figure , while maintaining the same material for the accompaniment parts. The chorus melody would be repeated twice for each full chorus, making for verse and chorus with a sixteen-bar length.

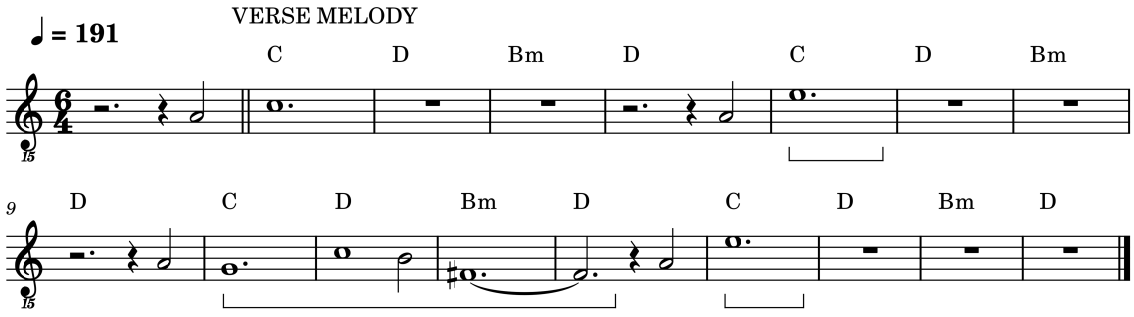


Figure 21: Full verse melody with new material indicated by brackets

The second verse kept the same melody as in the first, but developed the rhythm section to add more forward motion. This was achieved by editing in Ableton Live.

Material for the bridge was chosen for its distinctive contrasting rhythm section and melody that started staccato and opened into a sustained note towards the end of the phrase, as seen in Figure 20. I imagined this material being used as a slow crescendo, keyboards doubling the bass, playing three repetitions of the two staccato melody motifs and then opening into the sustained melody with an extension to balance the section length, as seen in Figure 21.
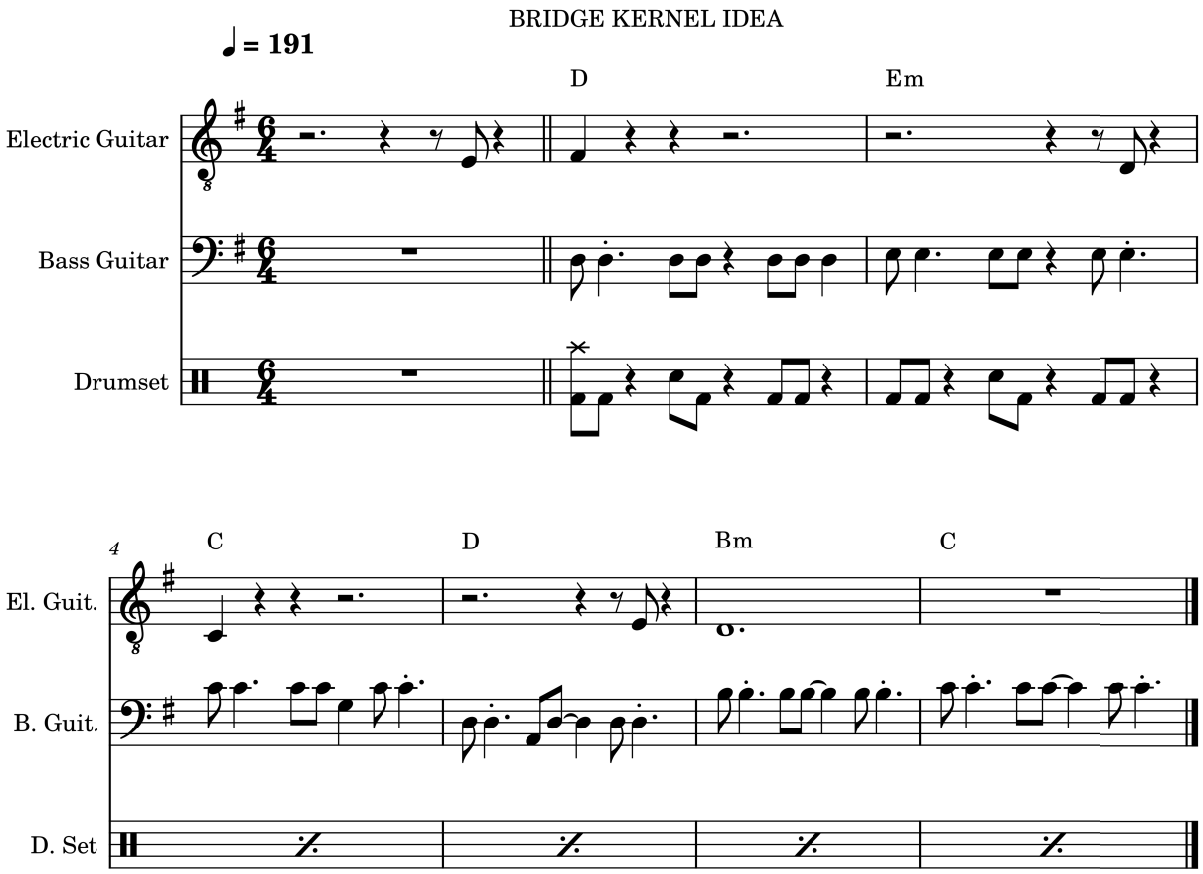


Figure 20: Original material selected for bridge

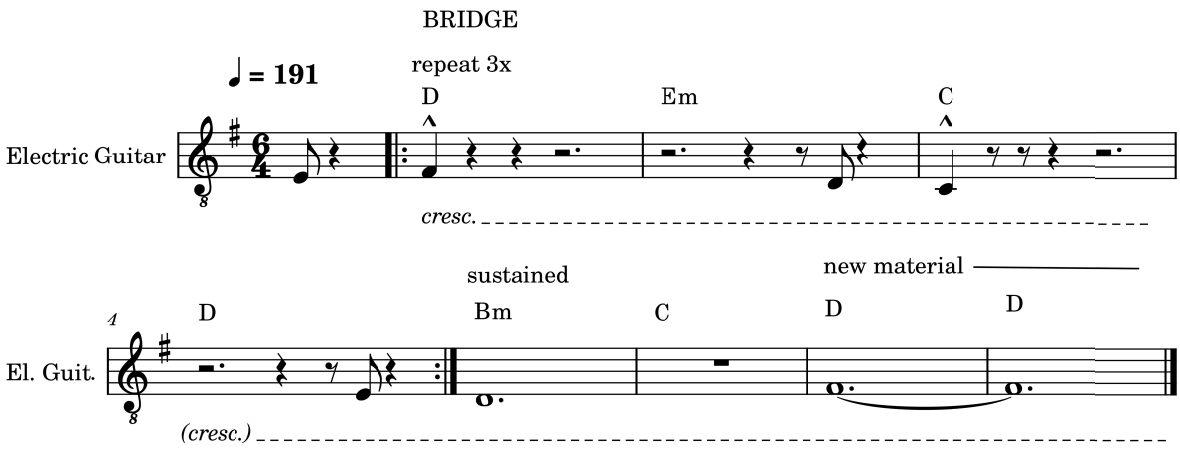


Figure 21: Arrangement of bridge material

This first experiment describes a simple development of material generated during a co-creative interaction into an arrangement for a popular song format. Although the process does not break new ground in composing strategies or achieve a finished audio rendering of the ideas, it does position the limited behaviour of a turn-taking co-creative composing interaction in a larger composing effort. What is apparent with this experiment was that even though the co-creative material was generated in several sessions, once the material had been taken into Ableton there was no return to the previous process. This is because a large number of options with potential had been developed and the co-creative interaction did not accommodate fine-tuning any further. What it does do is lay the groundwork for possibly extending the role of the variation-generating capabilities of the robot. For instance, had I had the chorus idea as a single parameter set in the robot (we did not, as it was an edit of several outputs), I could have loaded it as a user variation and experimented with feedback to achieve a simpler idea for the verse. In much the same way, I might have made variations of the verse idea to expand it into a sixteen-bar section. Similarly, I might have experimented with feedback aimed at generating contrasting material for the bridge. This approach would be something to strive for in the next experiment. While I could have addressed the same material again, I chose to begin with new material and follow where that might lead, as I felt this first idea had served its purpose.

4.7. The second composing experiment
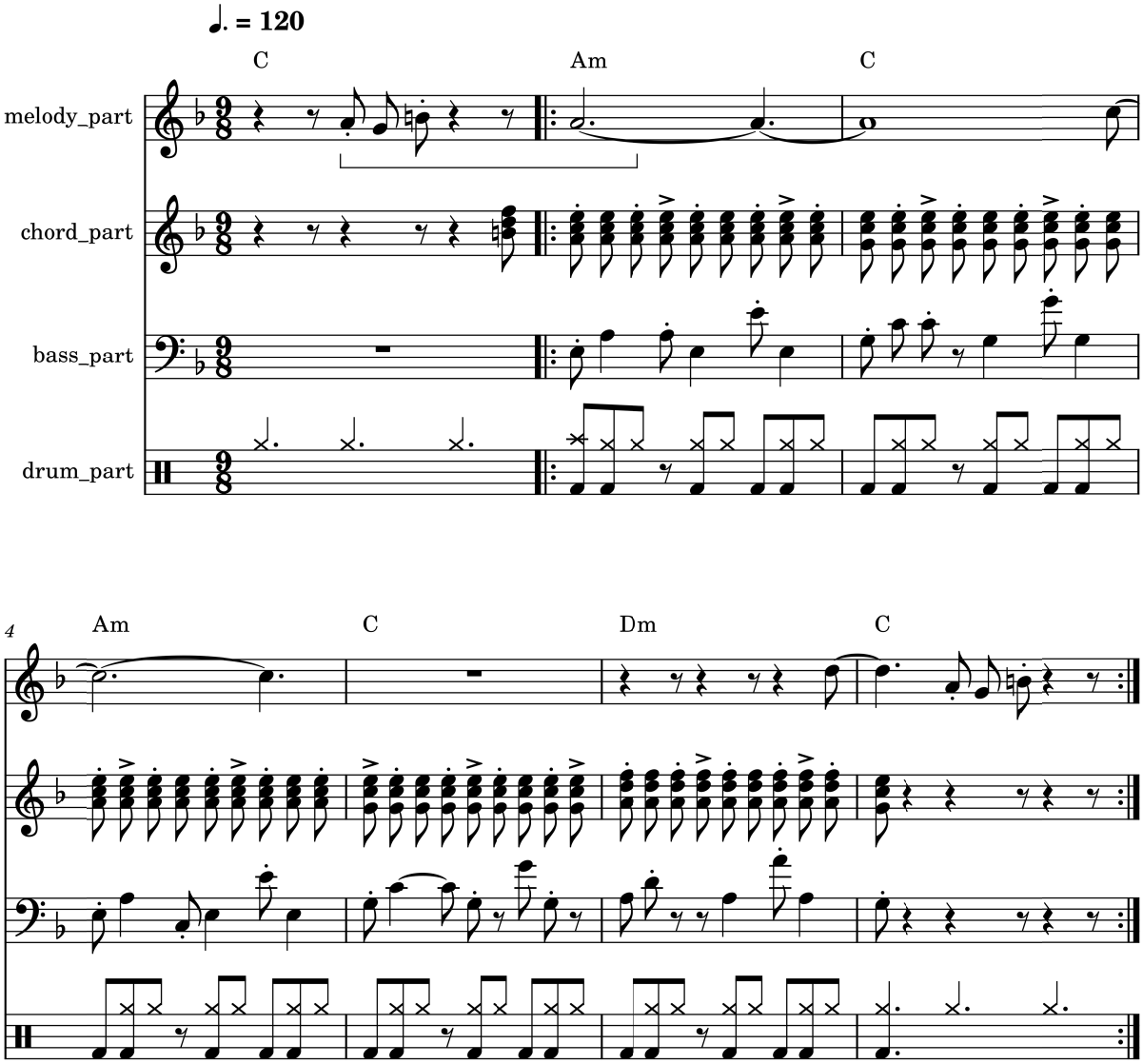


Figure 22: Initial idea for the second composing experiment, with the most compelling part of the melody being the first four notes, indicated by the horizontal bracket.

The second composing experiment began with a clearer idea of the overall goals. In terms of structure, I was intent on fitting ideas into an ABABCB structure. In terms of use of the algorithm, I would attempt to do little to no editing, therefore keeping the material in the parameter domain of the algorithm and then using existing material as a basis for generating variations by importing it as the current clip in the session.

4.7.1. Melody material

The initial idea for this experiment is shown in Figure 22, with the first four notes being the most compelling part. The initial plan was to use this as material for the chorus (B section) of

an ABABCB structure. In keeping with the notion of unpacking things as they happened rather than as they should have happened, and embracing mistakes as part of the composing process, it should be noted that the material as shown in Figure 22 has been presented in a format that represents the material as generated; due to practical issues around playback, it does not quite represent the listened experience. The section between the repeats represents the generated material, with the three notes in the pickup bar preceding the beginning of the repeat to be found in the last bar as well. As a method of explicitly delineating the beginning of the phrase during the co-creative interaction, where many variations might be played in rapid succession, I added the pickup bar into playback, with a rhythm section dropout, and moved the repeat markers (loop start and end point in Ableton Live) one bar earlier. This playback effect can be seen in Figure 23.

The actual embellishment of the first structure tone comprises four notes, but the first of these falls before the pickup bar and was therefore not included, only to be heard as a short eighth note at the end of the loop and separated from the subsequent three notes. Far from being 'wrong', the inclusion of this note, in my opinion, weakens this embellishment. The fact that a playback consideration after the automated composing process brought to light a stronger variation of an idea indicates a disjunct between the composing parameters and the listening experience, which the application/instrument is not able to assimilate into its parameter space. The application is therefore not able to fully capitalise on the sum of the processes that brought the material to the musical mind of the instrumentalist.

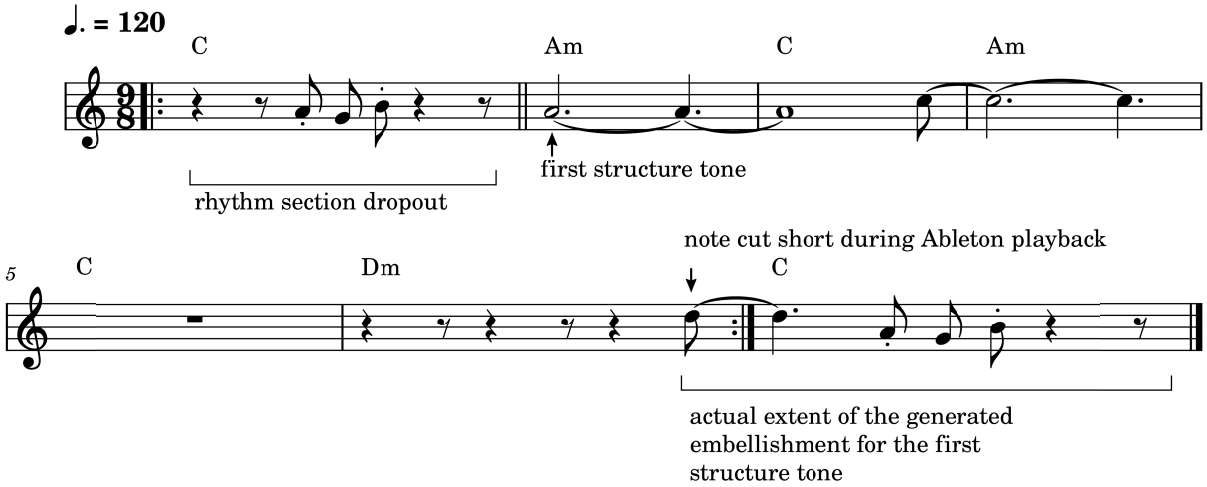


Figure 23: Actual playback of the initial idea for the second composing experiment, demonstrating the embellishment note that would have been cut short during playback and would thus not have formed part of the three-note pickup to the first structure tone.

4.7.2. Example of melodic development

This documents an example of developing a melody for the chorus of the piece from the original idea. The progression of variations of melodic material can be seen in Figure 24. This represents a short adjusted co-creative cycle of four variations. The original material was randomly generated by the robot. For subsequent steps, if it was user generated I have documented the feedback captured and the change in parameters for the melody. If it was a variation generated by the robot, I have shown the variation of parameters employed. I will omit feedback and parameters that referred to any of the other instruments. In this example there was little variation in the other instrumental parts, as the general goal was to achieve a more balanced phrase structure and more active melodic material besides the three pickup notes that had already been identified as appealing.

The feedback terms are taken directly from the log file of the interaction and as such are the class names of the feedback option from the Java code, which substantially describes the terms in which the operator would have had the option presented to them by the interface. I have edited out a prefix to these names which is only relevant to the organisation of the software code.
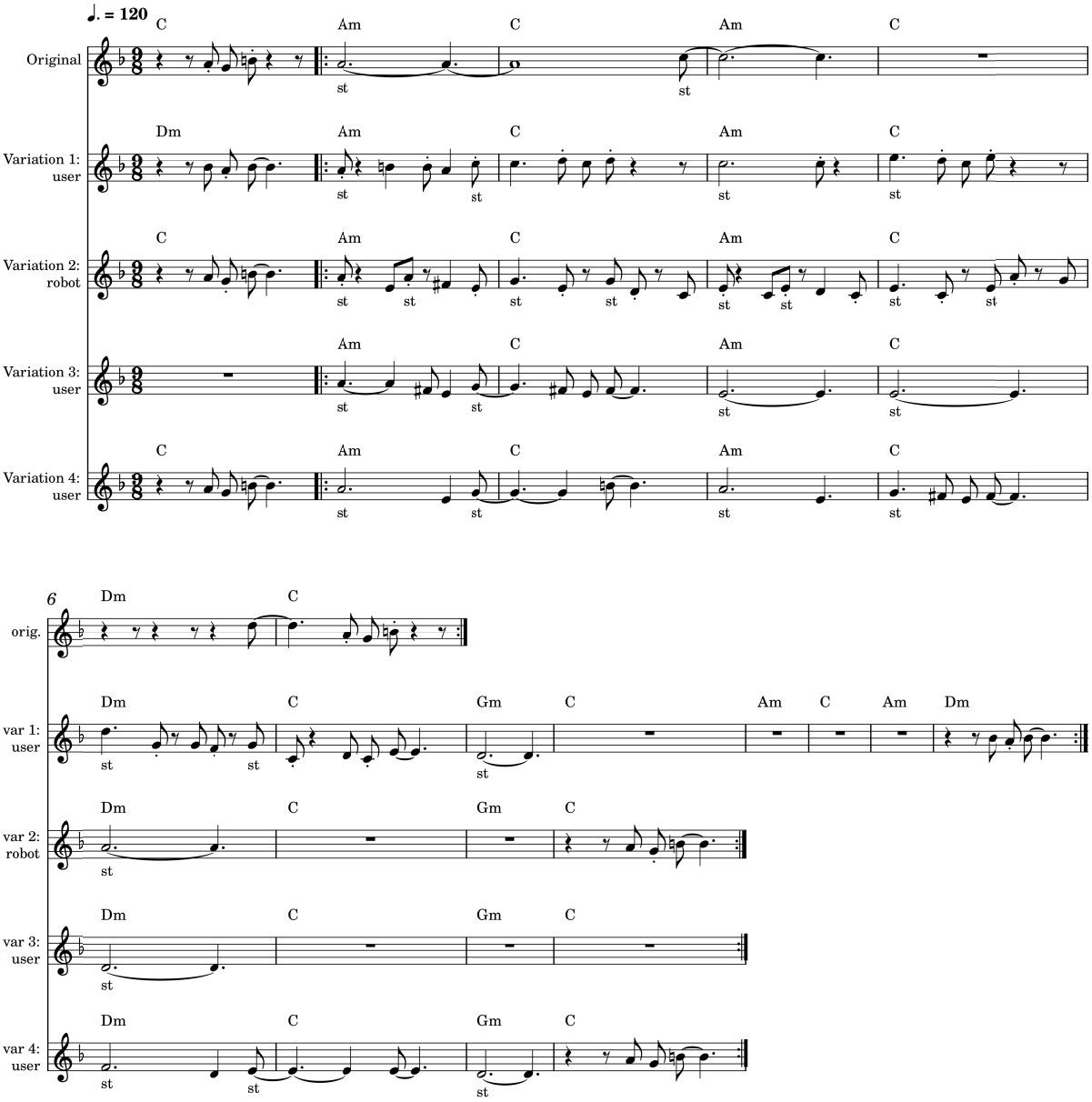


Figure 24: Progressive variation of the original melody idea to achieve the melody for the first half of the chorus. Tempo and time signature are the same, so the material can be presented in parallel, with only variations of length and a pickup bar representing the material found in the last bar of the repeat section, as this represents a significantly memorable motif.

First variation

Feedback captured:

1. "PhraseLength_aBitLonger"
2. "PhraseLength_MoreEven"
3. "ContourShouldSpendMoreTimeAscending"
4. "Melody_EmbellishmentCount_More"
5. "StructureToneSpacing_SetTo1"

1. At this stage the algorithm would pass the parameter file of the original material through variation generators related to the captured feedback points. This would have been a phrase-length variation generator to cover points 1 and 2, a melodic contour variation generator to address point 3, an embellishment count variation generator to address point 4 and a structure tone spacing variation generator for point 5. These variation generators make a random variation of the associated parameter, and all are applied to the original parameter file to create a new option. A list of 100 options is generated and each is scored based on the degree to which it satisfies each feedback point. The scores for each option are added and the option with the highest score is chosen as the variation to be presented to the user. The following parameter adjustments were made (I have edited these entries for readability, with 'old' representing the parameter value prior to change and 'new' indicating the new value chosen):PHRASE_LENGTH old: 6 new: 12
2. STRUCTURE_TONE_CONTOUR

   old: (0.0, 0.0), (0.5, 1.0) , (1.0,0.0)
   new: (0.0, 0.0), (0.5, 1.0), (1.0, −1.0)[54]
3. EMBELLISHMENT_GENERATOR SELECTED_COUNTS:

   0=[old: 4, new: 3]
   1=[old: 0, new: 8][55]
   (the pattern applied to this was in both cases [0, 1, 1, 0])
4. STRUCTURE_TONE_SPACING old: 2.0 new: 1.0

[54] These pairs of numbers represent (x, y) co-ordinates of a graph where the x axis has a range from 0.0 to 1.0, which represents the beginning to the end of the melody. The y axis has a range of −1.0 to 1.0, which represents any structure tone's relationship to the starting pitch multiplied by the structure tone contour multiplier. For example, in the case of these old and new options, it can be seen that only the second section of the graph differs. In the case of the old graph, the final note (x = 1, y = 0) would deviate by 0.0 * multiplier from the start note (i.e. the same note). In the case of the new graph (x = 1, y = −1), this value would be −1.0 * multiplier. If the start note = 60 and the multiplier = 10, this latter case would dictate that the last note would be 60 + (−1.0 * 10) = 50. Each melody structure tone is calculated this way and then quantised to the nearest chord tone of the prevailing chord at that position. It can also be inferred that this new contour is less likely than the old to spend more time ascending, which is the feedback requirement, and likely garnered a poor score for this reason.

[55] The number of embellishments for each subsequent structure tone is applied according to a pattern, which refers to these indices. If the index pattern were [0, 1], then the first structure tone would have the count from index 0 (old: 4, new: 3), the second would have the count from index 1 (old: 0, new: 8), and then the pattern would start again and loop for subsequent structure tones. As noted, the pattern for both melodies was [0, 1, 1, 0]. This pattern would be applied to the first four structure tones and then looped, etc. Patterns are not limited to only two indices: any number is possible, but the number of patterns was limited and preset. For example, [0, 1, 2] is possible and exists in the preset options; [0, 1, 2, 3, 4, 5, 6, 7] is also possible but is likely to be less musically meaningful, as there is less chance of repetition, and was not included in the available options. The generator would generate enough indices for the number of indices present in the pattern.

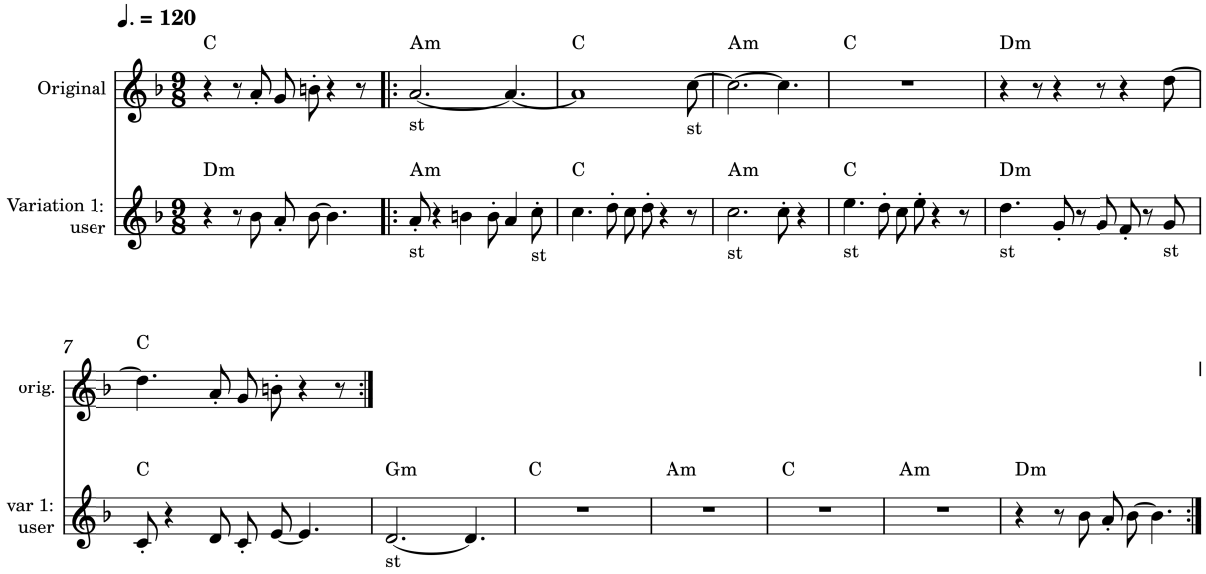


Figure 25: Comparison of the first variation to the original material. In both cases, structure tones are indicated by the 'st' text. Unlabelled notes preceding a structure tone are its embellishments.

The effect of these parameter adjustments can be seen in Figure 25. The new variation has a length of twelve bars. The contour of the melody in the original is difficult to relate to the original contour parameters. The parameters suggest the structure tone melody should rise and then fall, but only two structure tones exist, so I can only assume that the second tone finds itself towards the centre of the contour graph. In the variation, the spacing between structure tones has been reduced from two bars to one bar (according to the STRUCTURE_TONE_SPACING parameter change in point 4 above), and the contour can be clearly seen to rise from the first structure tone on A4 in bar 2 (marked with 'st', indicating a structure tone) to a high point in the middle of the melody (E5 in bar 5) and then falling to below the starting note to D4 in bar 8. Based on the old embellishment count values (0 = 4, 1 = 0) and the pattern in which they were applied (0, 1, 1, 0), we can see this in the original melody: there are only two structure tones, but the first is embellished with the count from index 0 and therefore has four embellishments, and the second has the count from the next index in the pattern, which is an index of 1, which equates to zero embellishments. Note that the earliest embellishment of the first structure tone does not appear in the pickup bar, as it falls outside of the last bar, of which the pickup bar is a copy. The full embellishment can be seen in the last two bars of the example. The adjusted embellishment counts can be seen for the variation as well, although the change of index 1 from 0 to 8 is not completely reflected in the example, as it is not possible to fit all eight embellishments into the more condensed space between structure tones. The algorithm does not attempt to 'squeeze' all the

embellishments in; instead, it adds them until it runs out of space. If the spacing between structure tones were to increase, or if a new time signature with longer bars were to be used, more of these embellishments would appear.

What is interesting to note is the change in the duration of the last note in the pickup bar: in the original it is staccato, whereas in the variation it is legato. This is particularly noticeable as these three notes represent the significantly memorable part of the initial idea, and no feedback has been introduced to change them. I must conclude that it is a bug in the system. This legato note persisted until the final variation, and when I used it in the final arrangement I manually reduced it to its length in the original. Future work in this respect would involve tracking down the bug, but also having a mechanism where such direct edits can be included in the parameter file and become a part of the knowledge of the system.

Second variation

The second variation was produced by the robot. I captured a 'Like' sentiment for the first variation. This would be included in the list of previous items in the session, with a similar sentiment. At that point the session had accumulated 16 liked and 33 disliked artefacts. The automated process of generating the robot variation is described below.

First, a list of parameters that were available to adjust was made. This list did not include parameters marked as locked in the UI. Secondly, twenty random variations of the first variation parameter set were generated. For each new variation, a random set of available parameters was adjusted. The number of parameters adjusted was based on a setting to simulate 'safe' or 'risky' behaviour by the robot, with safe behaviour resulting in a low number of changes and risky a high number of changes. This would equate to how far the new variation might deviate from the original and was in the range (2, 25). The safe/risky setting is a percentage setting in the range (0.0, 1.0); in this case it was 0.25, which amounted to seven changes.[56]

Thirdly, the list of variations was sorted. As mentioned before, this process introduces a way to parameterise the (admittedly limited) 'personality' of the robot. By using a measure of the Euclidean distance between two artefacts in the parameter space, we can position a new artefact in relation to previously liked and disliked artefacts. The selection of a new artefact

[56] (Maximum changes – minimum changes) * percentage + minimum changes. (25 – 2) * 0.25 + 2 = 7.75 truncated to an integer of 7

closest to a previously liked artefact could be construed as an attempt to ‘please’ the user, while selecting artefacts closest to disliked artefacts might be ‘annoying’ or ‘frustrating’ behaviour. In its current state, the algorithm assumes that it wishes to get as far away from the disliked options as possible, but it parameterises the closeness to the liked set as ‘pleasing’ (closest to the liked set) or ‘provoking’ (most distant from the liked set while still giving sufficient distance from the disliked set). This is a continuous parameter in the range 0.0 to 1.0.

This is a two-stage process which reduces the options to a short list of ten options, from which the final option is selected. In this case, the reduction to the short list consisted of calculating the average of the Euclidean distance between each option and the ten closest items in the list of liked options, and selecting ten with the lowest scores, also known as the k-nearest-neighbour algorithm in statistical analysis (James et al., 2013 pp 38-40). The effect of this is to get as close to previously liked options as possible. In the second stage, a similar score value is generated for the average distance to the five nearest items in the dislike list. This list is then sorted from largest to smallest. A robot personality parameter setting dictates which index in the list is chosen. This parameter, in the range (0.0, 1.0), determines to which degree the robot would like to please or provoke the user, with a pleasing stance selecting an item closer to the top (further from disliked options) and a provocative stance selecting one closer to the end of the list (closer to disliked options), by mapping the percentage to the number of items in the list. In this case the setting was 0.0, so the item at the top of the list was selected.

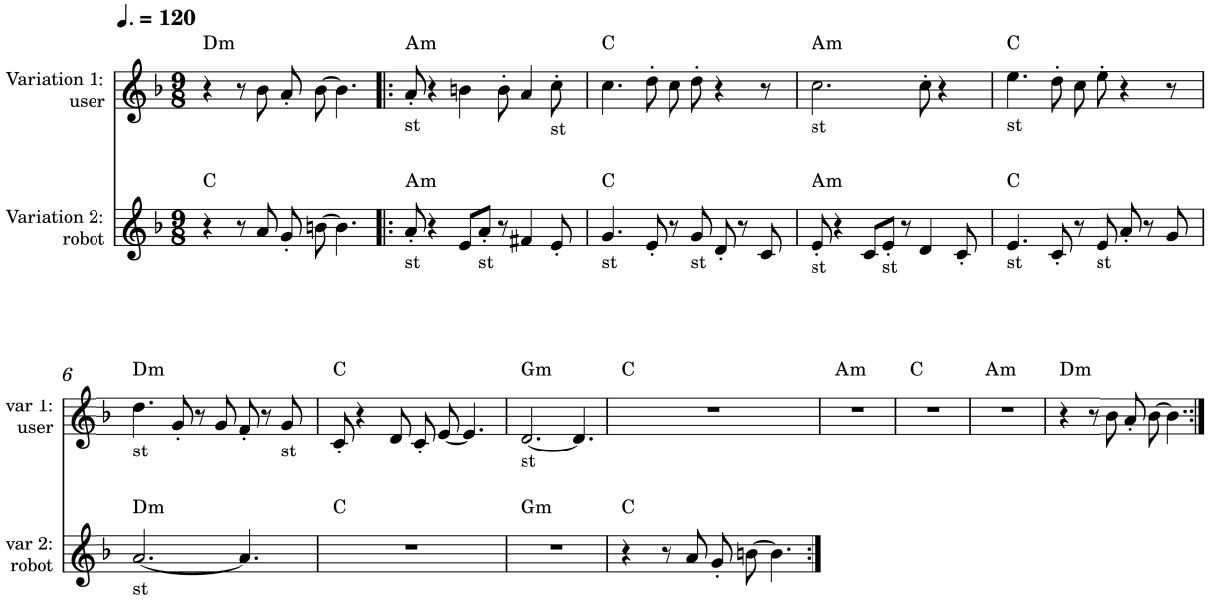


Figure 26: Progress from first to second variation during development of the chorus melody

The parameters selected for this step were as follows:

1. STRUCTURE_TONE_CONTOUR

   old: (0.0, 0.0), (0.5, 1.0), (1.0, −1.0)

   new: (0.0, 0.0), (0.75, −1.0), (1.0, 0.0)
2. STRUCTURE_TONE_MULTIPLIER old: 7 new: 6
3. PHRASE_LENGTH old: 12 new: 8
4. STRUCTURE_TONE_SPACING old: 1.0 new: 0.5

The new structure tone contour descends and then ascends, as can be seen in Figure 26. The multiplier, whose behaviour is described in footnote 54, changes from seven to six semitones, which is not significant as an observable difference between first and second variation as the change in contour is far more noticeable. The melodic range in both cases is quite similar. Likewise, the structure tone spacing, which reduces from a bar (1.0) to half a bar (0.5), contributes to the increase in note density for the duration of the melody. The phrase length reduces to eight bars, where it remains till the final variation which is used in the full arrangement. What can be noticed here is that the melody in variation two covers fewer bars than in the first variation. Even though the melody start and end positions have not changed, these parameters are expressed as a percentage of the total phrase length, and in both variations it can be seen that the melody occupies a similar proportion of the total phrase length. What also remains unchanged is the three-note pickup, which is the motif that recommended this idea in the first place.

At this point I delve deeper into the behaviour of the algorithm, and as an example I use the abovementioned change from a phrase length of twelve bars to eight bars. When laying out the scope of the parameters during the design of this version of the algorithm, I was very aware that it was important for the algorithm to be able to produce material that would need improvement. While I am not suggesting that a phrase length of eight bars is better than twelve, or indeed any other number within the allowed range of phrase lengths, through my interaction with the algorithm I found myself gravitating to four- or eight-bar structures more often than not. It may thus be worth examining if the ‘knowledge’ of my preferences, as captured by the algorithm, represented this in a way that may have influenced the above choice, or whether the outcome was just random.

At this stage in the interaction, the sentiment database contained the following information:

- Total number of sentiments captured: 144
- Total number of likes: 16
- Total number of dislikes: 33
- Total number of likes containing an eight-bar phrase length: 1 of 16
- Total number of dislikes containing an eight-bar phrase length: 2 of 33

The number of likes (1 of 16) and dislikes (2 of 33) related to an eight-bar phrase length is small. These numbers suggest that it is more likely that this choice, satisfying as it was in terms of the development of the material, occurred by chance. A captured sentiment is applied to all parameters of that artefact. This does not reflect the focus of the user when capturing the sentiment. Many times during sentiment capture I debated whether to like some artefact because there was a part of which I approved, or to dislike it if there was a part which I disliked. Even if the development seemed to be going in the right direction, I realised that the like/dislike mechanism would not identify what was important. This made the robot variation seem largely random, and difficult to integrate into the process once I, as the composer, had identified a direction towards completion of a longer-form piece.

This suggests future work that more clearly identifies individual parameter preferences. The further problem is that while I favoured phrase lengths of four and eight bars, I have certainly found uses for other phrase lengths, and such a parameter-specific preference mechanism would still need to consider some or all of the other parameters as context.

Third variation

For some reason, no sentiment was captured for the second variation, which must be a bug in the saving of this data, as there is no suggestion that the progress of the interaction was interrupted, such as by reloading the session and an earlier clip. As such, it would be impossible to proceed to capture user feedback without logging a sentiment about the material generated by the robot. This is unlikely to be significant in terms of the like/dislike selection mechanism, as the situation noted in the generation of variation two seems to suggest that this mechanism may not be effective.

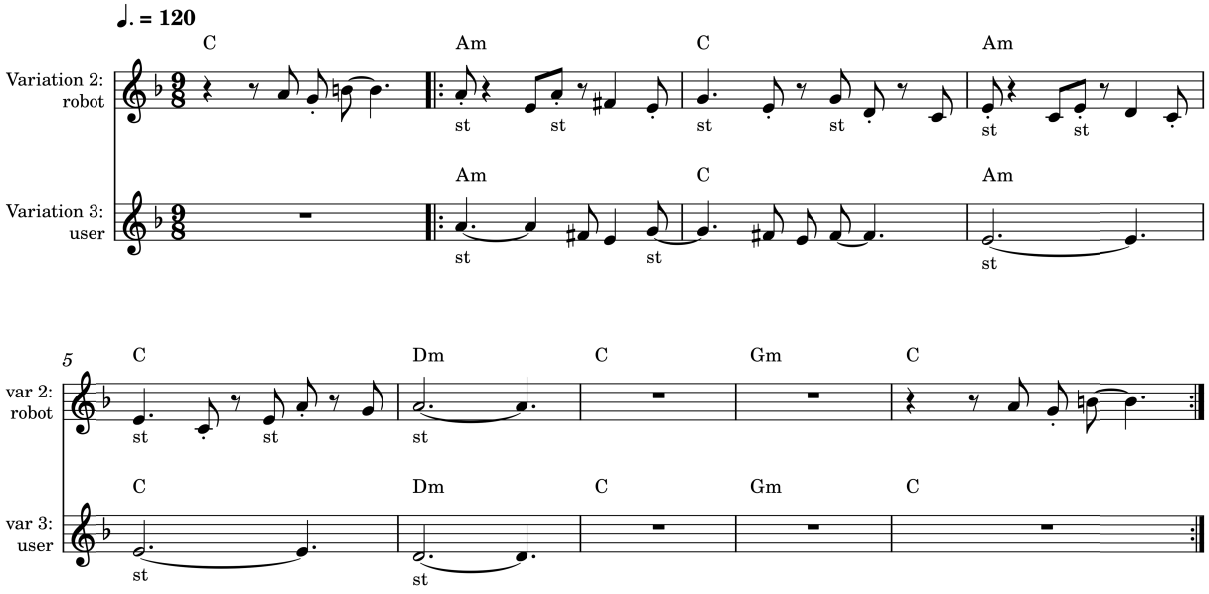


Figure 27: Progress from variation 2 to variation 3 of the chorus melody

The third variation, as seen in Figure 27, was the last of a series of eight variations on variation 2 which were achieved using the 'Try again' and 'Update feedback and try again' options. All were found to be unsatisfactory, and were captured as dislikes. The final feedback and parameter changes for variation 3, as seen in Figure 27, are as follows:

Feedback (melody only):

1. StructureToneSpacing_SetTo2
2. ContourShouldSpendMoreTimeAscending
3. Melody_MoreLegato
4. Melody_EmbellishmentCount_Less

Parameter changes:

1. STRUCTURE_TONE_SPACING old: 0.5, new: 1.0
2. STRUCTURE_TONE_CONTOUR

   old: (0.0, 0.0), (0.75, −1.0), (1.0, 0.0)

   new: (0.0, 0.0), (1.0, −1.0)
3. DURATION old: [DURATION_LEGATO, DURATION_STACCATO]

   new: [DURATION_LEGATO, DURATION_LEGATO]

4. EMBELLISHMENT_GENERATOR SELECTED_COUNTS:

0=[old: 3, new: 0]

1=[old: 8, new: 4]

What can be seen from the above is that the structure tone spacing has not been set to 2.0, even though it is longer than before. The new contour is also likely to not score as well as the parameters suggest a descending contour and the musical material shows a descending melody, which is contrary to the feedback related to contour. The note duration has moved from alternating legato and staccato notes in variation 2 to a legato duration on every note in the new variation. Lastly, the embellishment count feedback has been implemented as expected, with an overall reduction in the number of embellishments, but this does mean that the three-note pickup identified previously has now been lost.

In terms of composing, at this stage I was frustrated that variations on the feedback had not produced any useable material. As mentioned in the criteria for this experiment, I was eager to do no manual editing at this stage. This is an ideal place for the robot to offer a variation, with the likelihood that it would make some more significant changes. These would indicate either that the interaction is not likely to produce valuable material or, more optimistically, that the robot has found a compelling new variation in its random walk.

As it turned out, the robot variation was not of interest, so knowing that I had the seeds of a good idea in variation 3, I reloaded it so I could generate another user variation based on feedback.

Variation 4
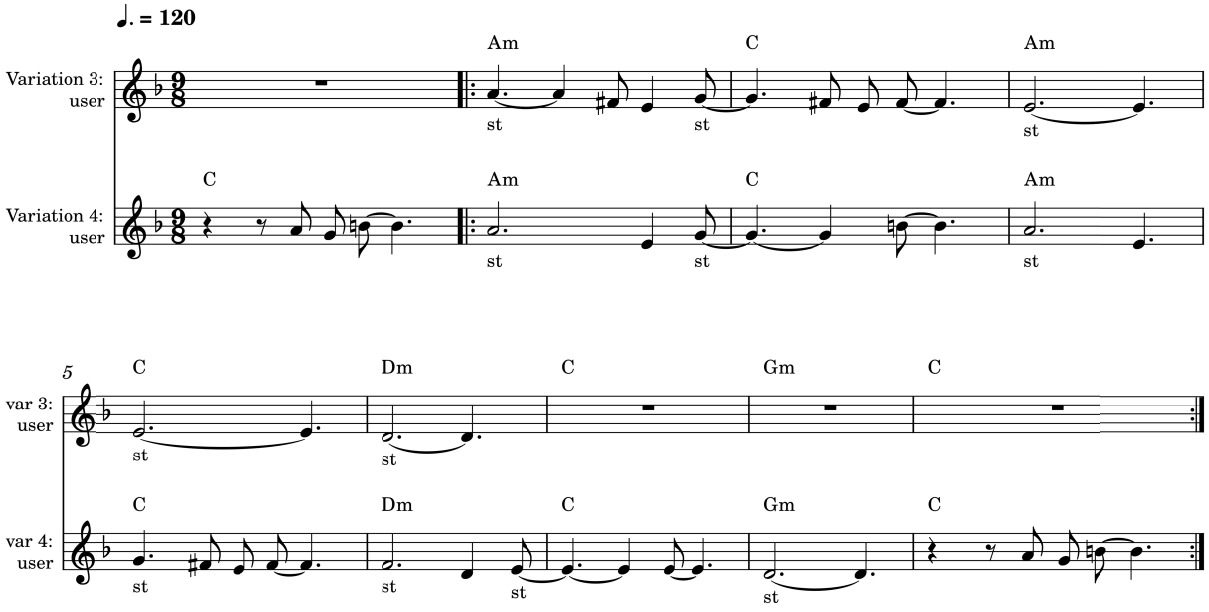


Figure 28: The progression to the melody used in the first eight bars of the chorus

The final melody for the first half of the chorus, as seen in Figure 28, was generated from the following feedback and parameter changes.

Feedback:

1. Melody_EmbellishmentCount_More
2. PhraseEnd_MuchLater

Parameter changes:

1. EMBELLISHMENT_GENERATOR SELECTED_COUNTS:
   0=[old: 0, new: 3]
   1=[old: 4, new: 1]
2. PHRASE_END_PERCENT:
   old: 0.528741227206722
   new: 0.7989288278177801

Once again the changes, even though directed by the feedback, are much more a matter of random luck than precise steering. While the feedback for embellishment count has serendipitously changed index 0 from 0 to 3, giving us back our three-note pickup, the overall change does not reflect an increase of embellishments, as there were four to start with and four remain, although rearranged. How might one look at this from a data-gathering

perspective and make a suggestion for how such a change could be better steered? There does not seem to be one, other than feedback which might look like 'Give me back my pickup bar', which would be too specific, or 'Rearrange the embellishment count', which I did not know that I wanted at the time. Under the circumstances, a random option based on the feedback was possibly the best general-purpose way to explore the space.

The above describes the process of developing musical material using the algorithm, but introducing more opportunities to backtrack on existing material. In particular, the step from variation 3 to variation 4, where the output goes from a user variation to another user variation, was not available in the application deployed for the online survey. With the online application, when using the 'Try again' backtracking mechanism it is possible to generate many user variations from a single robot variation, but it is not possible to generate a new user variation based on feedback from one which is generated in the same way. This is the asymmetric alternating co-creative interaction described in 2.1.4. A deep explanation of the behaviour of the algorithm and its use of captured knowledge has been shown here.

4.7.3. The case of the key

The issue of the prevailing key demonstrates a mechanism where the algorithm was able to update its understanding of material during output generation. The implementation of this mechanism in the algorithm was not planned from the outset, and emerged as the interaction of two discrete sub-algorithms. While in its current state it may not result in improved material in every case, the example described below was successful and points to a model for a more reflective understanding by the algorithm of its own behaviour.

During generation of the initial idea, the generative engine for this version of the algorithm selects a key at random (in this case, the key of F major). I have indicated this in all score examples by the key signature of the piece. Once this key is chosen, a chord progression is selected at random from the six diatonic major and minor chords of that key, with the constraint that there should be no repeated chords in subsequent bars, and the first and last chords should not be the same (to fulfil the criteria of no repeated chords when the material is looped). Already in the first three pickup notes that were identified as a memorable motif, we see the appearance of a B natural note, suggesting that the generation engine is seeing what I have labelled in Figure 29 as 'the first key area' of the piece in the key of C. Looking at the chords in the first four bars (Am/C/Am/C), we see further evidence that suggests that the piece is in the key of C rather than F, with the analysis of C as a tonic chord with no

contextual information to suggest otherwise. Of course, once we reach the G minor chord in the seventh bar, the key of F is more clearly delineated.

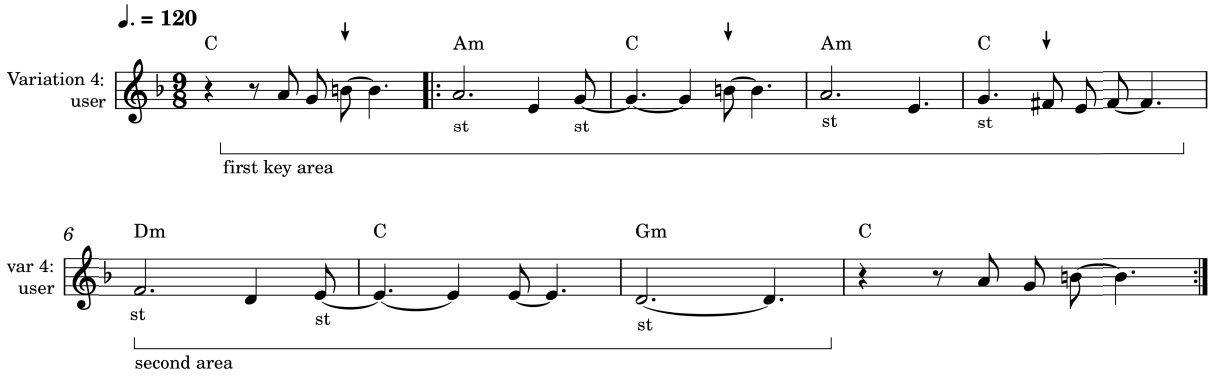


Figure 29: Key analysis used in embellishment pitch choice which contradicts the choice of prevailing key, but makes for (in my opinion) a more interesting melody. The mistakes made by this automated analysis is explained in the text.

The generation system that renders the pitch of the embellishment notes does not rely on the overarching key selection made elsewhere in the generation process. It looks directly at the chord progression and makes a best decision about the local key area based on the progression of chords as they occur over the course of time. So while the overarching key was definitely selected as F major, the analysis process would (or should) analyse the first bar as being in the key of C, as this is the only chord played so far and is assumed to be the tonic. The subsequent A minor, C major and A minor, D minor and C major would do nothing to change this analysis, and only the appearance of the G minor chord would suggest the key of F. While from a melody generation point of view, using either this analysis or assuming the key of F throughout might make for a viable melody, the choice that this system makes of a change of key centre makes for a pleasing and (possibly more) real-world use of the chosen chord progression.

The use of the change of key centre might look like an approach more rooted in an understanding of the experience of the chord progression as it would unfold during listening, if not for the appearance in the fifth bar of an F#. While from a composing point of view I find this note very pleasing, both for the way it leads chromatically to the F natural in the subsequent bar and as the #11 tone on the prevailing C major chord, it does suggest that the chord progression analysis made by the algorithm is seeing the first key area as the key of G

rather than C, which is possible but not logical.[57] This also suggests that the chord progression analysis is not behaving as expected, as I deliberately coded it to consider the appearance of a major chord (the C major chord in the first key area) as a tonic chord in the face of a lack of contextual information suggesting otherwise.

How did this arise? The chord progression analysis module was designed for a previous project and incorporated into the embellishment generation procedure of the current project to facilitate deriving key information. I considered it a good choice, as this module was designed to accommodate local changes in key in a longer chord progression. The behaviour above suggests, however, that this module is not working as expected, and a quick re-run of the software tests of this module confirm that at some point since its development a bug has crept in, as all the tests do not pass. It is only in ambiguous situations, such as the situation above where only a C major and an A minor chord are available to make an analysis, where a less logical decision of a prevailing key of G major is made.

I have highlighted this situation as it contains both a question and an answer. The chord analysis performed by the embellishment pitch selection process, in this situation, chooses better notes than may have happened if it had selected these notes based on the overarching key originally generated – or, indeed, if it had been operating as expected. This suggests that ongoing automated analyses for particular steps in the generation process may need to be considered as a matter of course, as they have the potential to update earlier decisions in the process based on greater availability of contextual information as the generation process adds more detail to the artefact. Further questions for future work relates to how to capture what is essentially a pleasing outcome from a software bug (but this should include any kind of mistake that could be made) into the parameter space of the algorithm for future use, and how these 'mistakes' succeed or fail based on the context in which they occur.

4.7.4. Generating and arranging the entire melody

I now look at a broader view of the composing process through interaction with the application, first for the entire melody and then for the process of arranging the rhythm section. I represent the interaction as a tree structure, with child nodes representing artefacts that are variations of their parent nodes. As there was extensive use of reloading of promising

[57] I should note that an alternative reading of this F# is as part of a Lydian/Dorian mode over the Am/C/Am/C chords. While this is another way to explain this pleasing musical occurrence, I do not consider it, as the analysis system does not accommodate this rule.

items for further user variation and ‘Try again’ backtracking to fine-tune feedback and add some more randomness to the option generated, I will indicate this as a description rather than an exhaustive list of material generated but not used in the final work. The following is the arrangement of the final melodic material and the names I will use to indicate sections within the arrangement:

1. Verse 1, 32 bars, consisting of:
    a. Verse 1a melody: 8 bars
    b. Verse 1a melody: 8 bars
    c. Verse 1b melody: 8 bars
    d. Verse 1c melody: 8 bars
2. Chorus, 16 bars, consisting of:
    a. Chorus part 1 melody: 8 bars
    b. Chorus part 2 melody: 8 bars
3. Verse 2, 32 bars, consisting of:
    a. Verse 2a melody: 8 bars
    b. Verse 2a melody: 8 bars
    c. Verse 2b melody: 8 bars
    d. Verse 2c melody: 8 bars
4. Chorus, 16 bars, consisting of:
    a. Chorus part 1 melody: 8 bars
    b. Chorus part 2 melody: 8 bars
5. Bridge, 16 bars, consisting of:
    a. Bridge melody: 8 bars
    b. Bridge melody: 8 bars
6. Chorus, 40 bars, consisting of:
    a. Chorus part 1 melody: 8 bars
    b. Chorus part 2 melody: 8 bars
    c. Chorus part 1 melody: 8 bars
    d. Chorus part 2 melody: 8 bars
    e. Chorus part 1 melody: 8 bars
7. Outro, 2 bars: a restatement of the three-note chorus pickup, so not a discrete musical idea.

Figure 30 illustrates the dependencies of material used in the second composing experiment. All in all, 191 variations were generated. It will be noted that the bulk of the material generated was from user feedback variations. In Figure 30 it can be seen that aside from the randomly generated initial idea, which was subject to selection from a set of ten similarly generated ideas, there are only two instances of a robot variation being included in the development process. This once again suggests that there was little trust from the user that the robot might helpfully develop ideas that the user has identified as having potential, something which was confirmed by my personal experience of using this tool for the development of material for a full composition. Even the volume of variations based on user feedback suggests that this steering mechanism for user control has significant similarities to a random walk through possibilities. Indeed, when collating this material for illustration of the process, it was clear that the material finally used was seldom identified at the time of variation generation. It was only when material was arranged on a timeline that it became clearer what was required. While the three-note pickup identified in the initial material was earmarked for use in the chorus, this early arrangement only included the Verse 1a and Verse 2a ideas in the Verse 1 and Verse 2 positions. When listening to this early arrangement, it was firstly clear that more melodic development was required in the verses, and that the associated accompaniment material would require significant changes to make the full arrangement coherent.

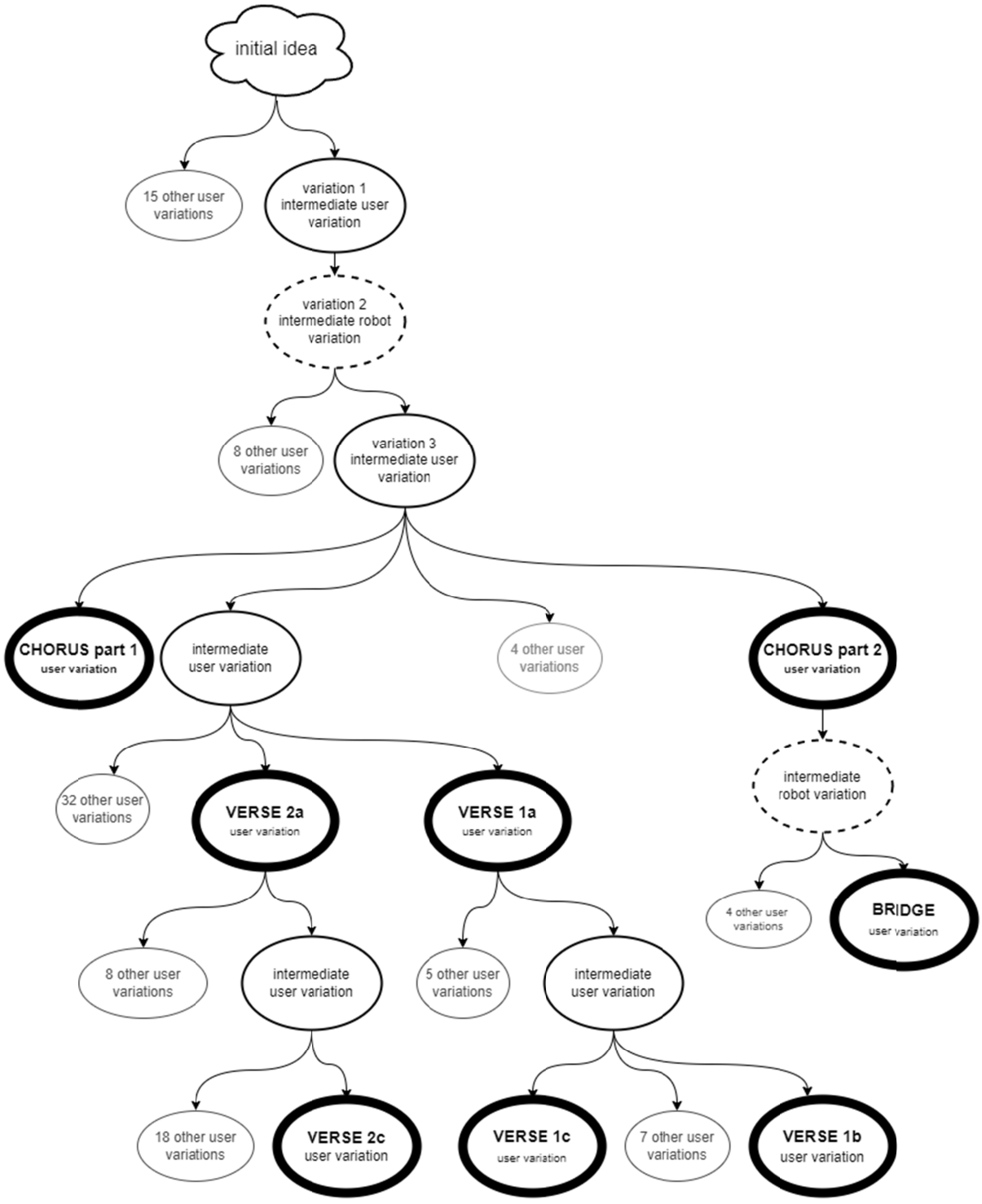


Figure 30: Development of the melodic material for the second composition exercise. Variations 1, 2 and 3 refer to the variations discussed in the development of the chorus part 1 melody, seen in Figure 24. I have identified them for context, but they function in the same way as any of the more generically identified intermediate user variations: not used in the final work, but representing material on which variations were made to develop the final material. Other user variations which were developed in parallel with ideas that were chosen for inclusion in the final piece are also included to give a sense of the number of variations considered. The count of these unused variations does not include unused children of these variations, which in some cases were substantial. Robot variations (except for the randomly generated initial idea) are identified with dotted borders.

From another perspective, it should be noted positively that these composition sessions were able to proceed rapidly through variations. The fact that I was not able to proceed immediately from an idea generation activity to organising the material into a longer form cannot be laid at the door of the application, as it was never intended to address this aspect of the composition process. The separation in time of the idea generation and arrangement activity is as much related to my mental perspective on the material and is similar to what I have noted in my composing processes without algorithmic assistance. And while a relatively large volume of material was generated during idea creation, it certainly was very productive.

The relative lack of the use of the robot variation option suggests that more work is needed on how the robot captures and uses user sentiment. Future experiments might also test the use of more rigorous exclusion of parameters from the robot variation process. This could easily be done with the current software, as these are user-defined settings in the user interface. It is also worth testing whether the robot variation procedure was making too many changes at a time – once again, an easy parameter to vary in the current version of the software. A larger issue suggested by the minimal recourse to robot variations is that the equal turn-taking co-creative interaction is not necessarily the best idea development strategy. However, it should be noted that this is an observation based largely on my own experience of the software, which includes detailed knowledge of the working of the system in relation to my musical expertise.

4.7.5. Development of the accompaniment material

Most of the material for the chords, bass and drums parts was altered once the overall structure of the piece became clear during the development of the melody. This was expected, as the algorithm contained no way to plan musical coherence beyond the scope of a single phrase. The algorithm was used to generate this accompaniment material in a similar way to the process of developing the melody, with the use of user variations based on feedback predominating. I shall therefore not go into the same detailed explanation of events.

Accompaniment material was selected based on an overall progression of the energy and forward motion required from a particular section. In general, choruses were considered more energetic than verses, Verse 2 was considered to require more forward motion than Verse 1,

and the accompaniment for the bridge was given a significant change in texture. The latter was the place where I intervened with the details of the parts.

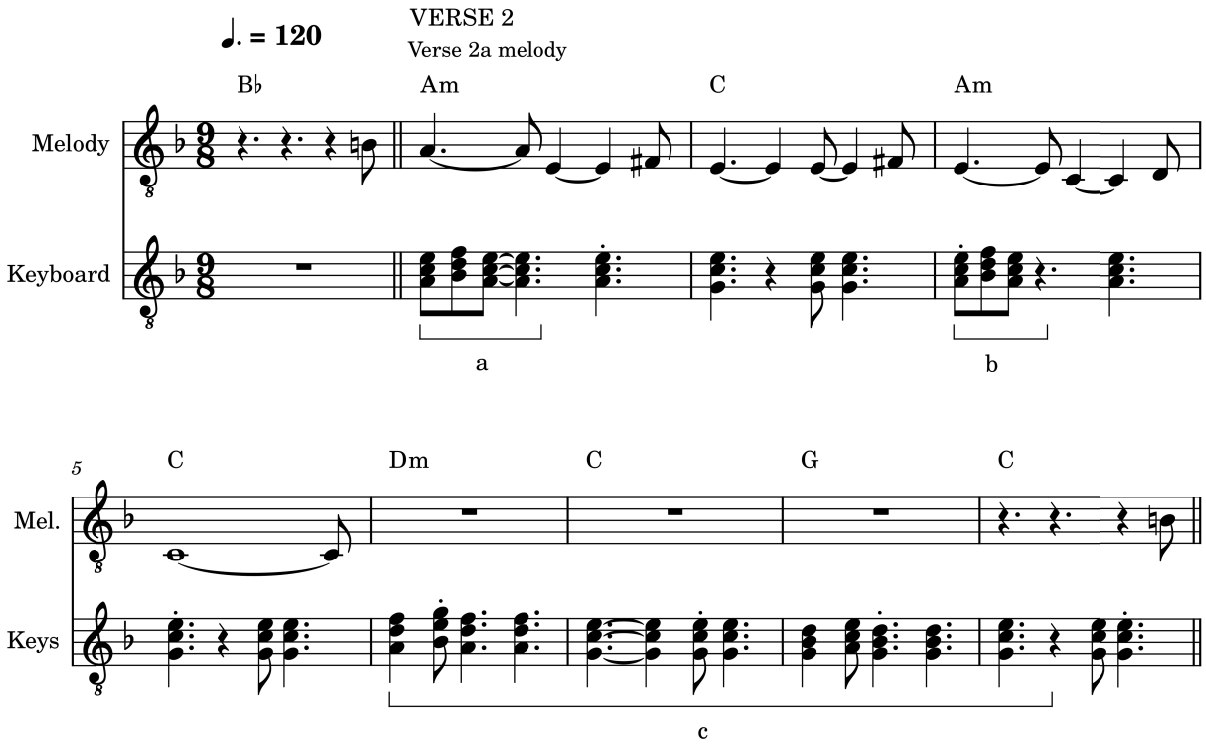


Figure 31: Examples of keyboard parts that emerged by chance but were uncontrollable in any further way. The motifs marked 'a' and 'b' tended to obfuscate the arrangement. The section marked 'c' was a pleasing melodic response to the melody part. It was not possible to remove motif 'a' and 'b' by the parameter specific feedback mechanism without compromising motif 'c'.

In general I did not attempt to refashion the details of parts beyond what the algorithm had generated. I did at certain points choose to eliminate an instrument from a section in its entirety, such as starting the piece with only melody and chord parts. Options for the adjustment of the accompaniment parts were quite limited when it came to the process of fine-tuning for a finished arrangement. An example can be seen in Figure 31. A pleasing melodic motif (marked 'c') emerged in the keyboard part, but there was no functionality in the algorithm to develop it in any way. Furthermore, the part came with some figures I deemed less desirable (marked 'a' and 'b'). Feedback-based variation to remove these undesirable elements inevitably affected the desirable one. Here I decided to not intervene and to take the good with the bad, as the example clearly illustrates a limitation and an area of future work if the project were to be expanded to handle arrangement of longer forms.
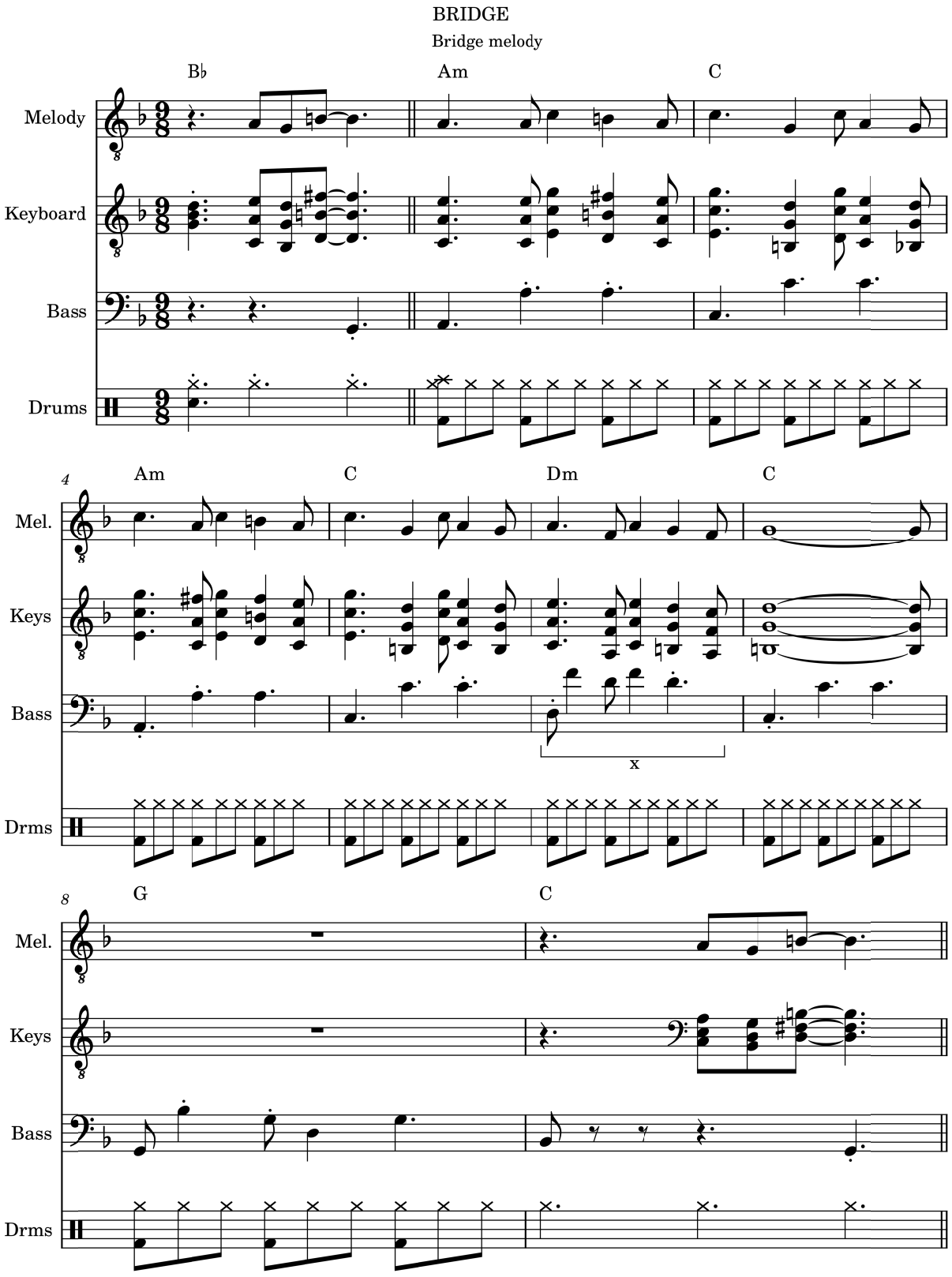


Figure 32: An example of composer intervention into the keyboard part, which was reimagined as a parallel harmonisation of the melody. This occurs in the bridge, with the purpose of making a dramatic change in the accompaniment texture.

In Figure 32 I demonstrate the first eight bars of the bridge, where I chose to intervene in the details of the accompaniment material by making the keyboard part a parallel harmonisation of the melody, with the overall purpose of providing the bridge with a contrasting accompaniment texture. While this was not an introduction of new material (being essentially a duplicate of the melody), a motif in the bass part (Figure 32, marked 'x') playfully includes itself in the melodic texture without actually doubling it. As mentioned with the example of the keyboard part in Figure 31, it was not possible to develop this further with the functionality available in the algorithm, as the link between the melody and bass line was serendipitous, and not explicitly parameterised in the algorithm. I left this idea as it was, as recomposing the bass line by hand to expand on an emergent effect was an intervention I did not want to make for the purposes of this experiment.
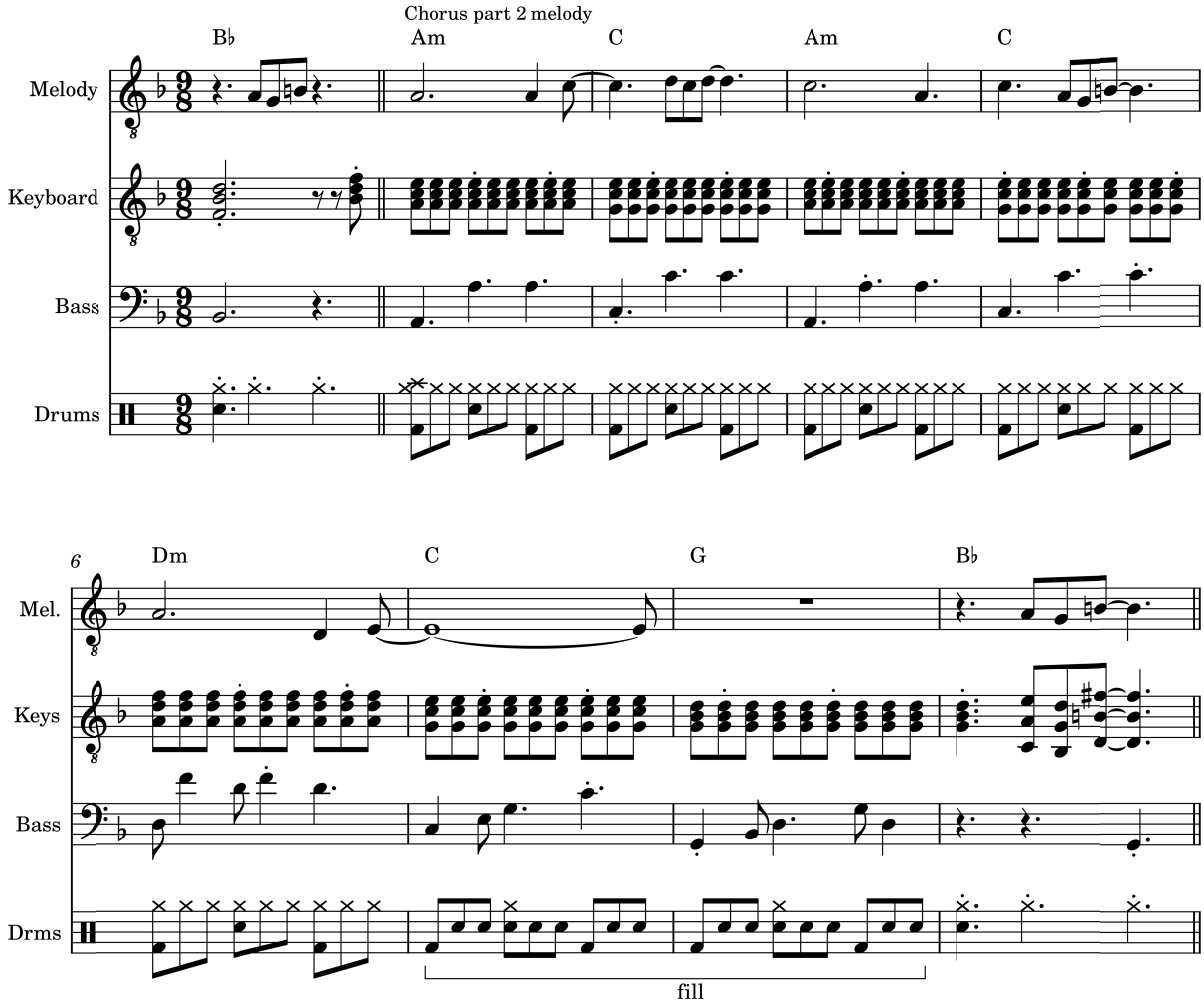


Figure 33: Example of a fill from the chorus. The section indicated was copied and pasted from a variation of the rhythm section material with greater note density. In this example the new material includes the drum and bass part only.

After the rhythm section material had been chosen for the duration of the piece, I generated further variations of the rhythm section material, using feedback to vary the note density and the amount of syncopation. From this material I selected excerpts that could be used as fills at the end of sections where appropriate. These fills comprised one or two bars of different material at the end of some sections, and an example can be seen in Figure 33.

This concludes the description of the composing experiments.

5. Results

The application was designed and tested in a five-step process:

1. Development of design constraints of an interactive algorithmic composing model
2. Development of user interaction model
3. Qualitative testing on a small number of expert users
4. Quantitative testing on (a large number of) users
5. Qualitative testing on personal composing experiments.

Steps 1 to 4 were planned from the outset. Due to disappointing results from the quantitative testing phase, the fifth item above was added to attempt to gain insights from another possible use case. Results from step 3 have been reported in 4.3, as they impacted the decisions made to proceed to the next step. It was not possible to make any valid statistical inferences from data gathered in the quantitative testing in step 4. Besides the multiple choice-questions, there was an optional comment section at the end of the survey. Some of these user comments furnish qualitative information about the user experience, so these are mentioned where it confirms insights achieved in other evaluations. Lastly, I note some technical achievements and considerations that underpinned the work on the project.

5.1. Quantitative testing

In reflection on the progress of this project, several complexities emerge. Initially the idea of an interactive music composition with an algorithmic composing tool suggested the following approaches:

- In terms of composition, the important issue was controllability, and given my domain expertise in software encodings of music composing rules, a rule-based expert system was chosen.
- With the prospect that this could be a deployed online as a browser-based app, an HCI approach to design and evaluation of engagement seemed to be an obvious choice. This last choice pointed towards a quantitative data gathering process which was compelling as a way of garnering (more) objective data about a project which was also a creative exploration. Furthermore, an acknowledgment of the shortcomings of computational

creative processes further reinforced the necessity of a human-in-the-loop and brought the techniques of computational co-creativity into the project.

Several issues intervened over the course of the project which brought to light the shortcomings of these early decisions of scope. Firstly, the arrival of the newest generative AI models, producing visuals, audio or text and steered using natural language prompts, completely has changed the perception of creative exploration using computational tools. In particular, the way in which these tools are negatively seen to impact on the role of the human creative process, while not demonstrated empirically in this study, may contribute to the poor uptake of the online study, which was in no way as interesting or impressive as the more readily accessible AI generation tools steered using natural language.

While the focus was never intended to be on the lay user, it seemed difficult to make a case for a smaller, more highly skilled group of participants to take the time required to complete the interaction and survey, so this quantitative approach to data gathering was ultimately disappointing: not enough data were gathered to make any statistically meaningful conclusions. Furthermore, the conclusions reached seemed less significant than the insights gained from earlier qualitative tests with two expert users. To do justice to this need for depth, the HCI approach seemed to be incomplete as an approach to evaluate the success of the application.

The taxonomy of Tanaka (2009) suggests that the application covers the criteria for what constitutes a digital musical instrument, especially thanks to the inclusion of a compositional structure component, which may not feature as a part of a conventional instrument. To reiterate, the components of an open-ended musical instrument using digital technology are:

- input device
- mapping algorithms
- sound synthesis engine
- compositional structure
- output system.

Rodger et al. (2020) suggest that the examination of the instrumentalist/instrument ecology can uncover a different (and possibly better) set of insights by treating the instrumentalist as a distinct individual whose interaction with the instrument can be evaluated in greater depth.

While this might suggest a whole new qualitative study, there was limited time. I therefore decided to conduct some limited experiments using the device in my own composition workflow, and to attempt some constrained continuations of material developed in a co-creative interaction to point to future work as well as to possible subversions of the processes that were either available or would be made easily accessible. 'Easily accessible' in this case would imply no changes to the music generation algorithm but perhaps manipulation of the data that the system used for music generation. This could then be combined with test data that I did have from the two limited user tests conducted previously in order to reach some conclusions regarding the project as a creative exploration.

5.2. Composing experiments

Two pieces were compiled using material generated by the application. Full scores of this material can be seen in Appendix C: and Appendix D:. These experiments sought to gain insight into the use of the application as a composing tool rather than a tool for engagement with a compositional process.

As mentioned in 4.5.1, during the composing experiments, I proposed to examine the following:

- What combinations of the options available in the application were used for generating material for a longer-form composition?
- How was material selected?
- What manipulation of the material was used outside of the generation process of the application?

These reflection points were postulated prior to the first composing project, and were substantially answered by the experiments. In general, the option to generate variations on user feedback was the predominant method of generating new variations.

The most used method was to import a clip into the system as a robot output, therefore setting it up for variations to be generated by user feedback. The generation from feedback was iterated upon, sometimes extensively, using the 'Try again' or 'Update feedback and try again' options, until it was considered that some promising variations had been generated or there were no promising options forthcoming. Selection of material happened later, as I experimented with arranging the material on a timeline to evaluate suitable combinations.

Robot variations were generally employed as an injection of randomness, as the robot did not seem to demonstrably learn from collecting user preference data, for reasons mentioned below. A detailed analysis of the genesis of the material used for the melody in the second composing experiment shows that from four to seven steps of variation were used to develop final material from the initial idea. When compared to the number of variations generated from a combination of origin clip and feedback combination, it can be seen in Figure 30 that many parallel options were generated. Indeed, the Verse 1a and Verse 2a ideas come from a group of thirty-four variations from a single origin. This suggests that the steering process based on feedback implemented in the project did not decisively discover the path forward in the creative interaction with the available interaction data. This method has much more in common with a random walk, where generation in one step is repeated until a satisfactory option is found, than with a co-creative interaction, where developments in successive iterations are built upon to achieve the final result.

The case of the key, as described in 4.7.3, demonstrates a problem and some proposed solutions for emergent effects in an algorithm. A pleasing musical effect arose from a potentially redundant re-analysis of the chord progression, which also contained an error which brought out a composing technique which the system could not parameterise.

This could be seen as being analogous to a moment of inspiration during a composing process which is pleasing but breaks the existing rules of the music system. This suggests that extending the idea of re-evaluations might include a system for accommodating the broader context where such a ‘mistake’ is successful, and that such re-evaluations might become a more standard approach to the design of the sub-algorithms in such a system. In this particular case, the ‘mistake’ is not necessarily novel and is covered by a music theoretical model that was not accommodated in the algorithm at the time. It suggests that the error would need to be corrected and the effect that emerged would need to be added to the rule set so that feedback options could be designed to address it unequivocally.

5.3. Technical observations

The version deployed for quantitative public testing was technically successful in some fortuitous ways. Over the course of the project I moved from using a single coding language (in this case, Java) and Ableton Live for playback as a basis for experiments, to fully fledged multi-user online deployment using a variety of technologies, including Spring Boot for

back-end development, Node.js and React for front-end development, Python and Tensorflow for data analysis, MySql as a persistence database, a WebAssembly version of Csound for playback of generated material from a browser, and the deployment of all this to the Amazon Web Services cloud computing platform.

What is also gratifying is that the deployed version, for all its conceptual limitations, worked without technical errors. This success stems to a large degree from the choice of mature technologies that came with excellent demonstration material. In retrospect, the use of Csound was the biggest risk, as it is the most niche of all the platforms mentioned above. Csound has existed since before the IBM personal computer as we know it today, but has been mainly used in academic circles for electronic music experimentation (Di Nunzio, 2013). The version of this powerful tool that I used was an implementation in the relatively new WebAssembly framework, as described in Yi, Lazzarini and Costello (2018), which is a platform for the execution of high-performance software in a browser. This work was done by a team of three people, and while the robustness of the implementation is a testament to their excellent work, from a broader perspective such a niche software project runs more of a risk of being unreliable and poorly supported than, for instance, something like Spring Boot, which underpins millions of websites. From the perspective of my project it was the riskiest choice amongst the chosen technologies. My research suggested it was better than other audio playback alternatives, but I had no prior experience with it, so there was no guarantee that it would ultimately be suitable. In the end, it worked excellently and made for audio playback in the online application that was significantly above my expectations.

6. Discussion

The research question proposed in 1.4 asks:

> *Can a co-creative interaction with an algorithmic composing tool – one that uses a combination of rule-based musical generation models and which includes real-time data collection of user preferences from the interaction – engage the user in a process that develops musical material?*

In the sections that follow, I break down this question into its constituent parts and discuss the results.

6.1. The rule-based generation model

This can be divided into a discussion on the output of the rule-based generation model and the ability to integrate these rules with the steering mechanism that was described in musically meaningful terms.

The algorithm that generated music performed successfully in various ways.

6.1.1. Music representation

The music representation system (based around the Java Mu class, as described in 4.1.1) was a useful framework to manipulate musical material. Before the final shape of the system became clear, this class was overloaded with functionality, some of which was stripped out during the development process and either no longer supported or moved to other utility classes. The core functionality that remained within the class was the musical positioning system using a variety of positioning models – an open-ended tagging system for attaching metadata to musical objects (and persisting information through the generation process) – and the easy access to structural information, such as prevailing chord and time signature. Important utilities related to this class were the various output formats that could be generated: musicxml files for score output and integration with systems to transfer playback data to Ableton Live and Csound. While this class changed over the course of the project, acquiring and losing functionality as its role became more clearly understood, the choice of a composite software design pattern has proven to be a robust approach to manipulating

musical material, and no better model for music representation has emerged from this research.

6.1.2. Melody model

The generation model – using a hierarchical structure tone and embellishment model to generate melody – has proven to be a worthwhile choice. Work on a system to generate parameter files from musical artefacts (essentially a reversal of the generation process) which never made it into the final algorithm showed enough promise to suggest that there is more to explore using this approach.

6.1.3. Accompaniment parts

The algorithm used to generate accompaniment parts showed itself to be more idiosyncratic than that for melody generation. It was a model based on the selection of rhythmic positions for notes, chords and drum hits, organised by beat strength. Its biggest strength is its ability to be applied to any time signature. It was originally designed as the kernel of a live performance tool for chord, drums and bass parts, with simple control over the degree of syncopation and note density of the parts using a MIDI pad controller.[58] While it generates cogent material at times, the output tended to be stylistically and musically somewhat limited. With a smaller set of feedback options per part than the melody generator, and more limited possible musical variations, I was nevertheless able to shape the energy and density of accompaniment for the composing experiments without generally intervening in the note-level detail of the parts.

Unlike the melody generation model, this model has little scope to represent diverse musical approaches to rhythm section composition, and little chance to be developed further as such. It might be useful as one approach of many in an expanded system. The greatest limitation of this model is its poor potential as an analysis model for extracting generation parameters from musical artefacts. Early experiments with the melody generation model as an analysis system, which did not make it into the project due to time constraints, point to an analysis capability being an essential requirement for any future generation algorithm.

[58] In this case, it was the Novation Launchpad, which is a hardware interface for Ableton Live characterised by an 8 x 8 button grid.

6.1.4. The feedback mechanism

The feedback system used during the user variation step was easy to integrate with the generation parameters of the rule-based generation algorithm. Each parameter was associated with one or more feedback options and a variation generator. For example, the parameter of time signature was associated with two feedback options: ‘more even’ and ‘less even’. The feedback options associated the text description with an evaluation function and a variation generator. The generator would be called to offer a new value for the parameter, and the evaluation function would score the new parameter according to what degree it satisfied the text description in relation to the old parameter. Once this software design template was established, it was easy to expand to all parameters. During the composing experiments I added new feedback options to speed up the interaction. These were very quick and easy to implement, and demonstrate the extensibility of the software framework.

6.1.5. Rule-based generation

For the requirements of this project, a rule-based approach to generation was sufficiently steerable to generate and develop musical material. Certainly, as the designer encoding a part of my musical understanding in an algorithm, it was pleasing to explore the parameter space, and there is no suggestion that this approach has demonstrated its limits for practical music making. Certainly there is no sense from the experience of the development of this project that it would have been better served by a data-driven approach.

This project gravitated towards an extremely personal composing scenario where I was able to explore some known detailed parameters of music composition. I found the workflow inspiring and did not feel that it limited my creativity in any way. It must be noted that the final output, as notes on a page, has some merit in my opinion. However, the audio playback managed during the project by no means represents a satisfactory final expression of the material.

6.2. Real-time sentiment collection and the robot variation

The research question proposed that capturing user sentiment would be a way for the algorithm to model user preferences, and could be used to steer the robot’s musical decision making based on a simple parameter describing the robot’s goal in the interaction: to either please or provoke the user with its contribution. During pre-alpha testing, due to this module

being incomplete, the robot made random decisions as to how to progress, and both users identified this as a point in the interaction which might disrupt their perception of the progress of the musical development. During the composition exercises, where this model was in place this disruptive effect was still noticed, suggesting that the process of sentiment capture and automated musical decision making cannot be differentiated from a random choice.

The following describes the concepts and operations of this mechanism:

- The generation algorithm is deterministic; therefore, parameter sets for an artefact are unique. That is, one parameter set will always produce the same artefact, unless the generation system has changed. The generation system is assumed to not have changed.
- The Euclidean distance of one parameter set to another is a measure of musical similarity.
- As the interaction progresses, the user is asked to capture a like/dislike sentiment about new artefacts that emerge. This data is attached to the artefact and stored in a database.
- Using the k-nearest neighbour technique, a total Euclidean distance can be calculated to k- closest artefacts to which a like or dislike has been associated.
- During the generation of the robot variation, a list of variations are generated where a number of parameters have been randomly selected to be changed. This number is determined by the robot personality parameter, which determines whether the robot is engaging in safe or risky musical behaviour. Safe behaviour would change a small number of parameters and risky behaviour would change more.
- Given a list of generated artefacts and a stored list of liked and disliked, the Euclidean distance measure can be used to sort them into most to least liked and most to least disliked.
- The robot selects the best option by first removing a percentage of the most disliked items. The remaining items are then sorted from most to least liked, and a final option is selected according to the robot personality parameter. If the robot is trying to please the user, the most liked item would be chosen, and if it was trying to provoke the user, the least liked item would be selected. This please/provoke parameter is a continuous variable between 0.0 and 1.0, with 0.0 indicating please and 1.0 indicating provoke, and would select the initial or final item of the aforementioned list, respectively. A setting of 0.5 would select the item from the middle of the list.

While the model demonstrated its ability to correctly measure these distances and sort generated material, this did not translate to adding a noticeable 'intelligent' behaviour to the robot's contribution in the interaction.

There are several ways in which the above line of reasoning could fail to deliver expected results:

- The parameter space might be a Euclidean measure of musical similarity, but it might not be a Euclidean measure of user preference.
- The like/dislike sentiment might be too broad an evaluation. This certainly occurred as I was using the robot for the composing experiments. For instance, if I liked the bassline in an artefact but was ambivalent about the rest, the question of whether I should like or dislike the entire artefact was not clear.
- There is no guarantee that the set of options generated by the robot from which to choose the final version is in any way 'better' than the variation on which they were based, as they are random. The process could be choosing the best from a set of poor options.
- The number of changes made during the variation process was not tested, due to time constraints. This number of changes could be a strong contributor to the disruption caused during this step, and fewer changes might make some difference.

What emerges is that this process was not explainable to the user, and did not allow for any user contribution to the robot's decision-making process. Kantosalo and Toivonen (2016) suggest that sharing descriptions of creative strategies and assessments is a further way to proceed with a co-creative interaction, over and above the sharing of artefacts contributing to the work itself. While in the current project there may not have been time to implement such a system, the chance to question the robot's decision making may be another way to shape its behaviour.

6.3. The co-creative interaction and randomness

The interaction was modelled on the idea of an *alternating* co-creative interaction, as described in Kantosalo and Toivonen (2016) and summarised in 2.2.3. This requires that the computational agent be *complete* and able to contribute in an equal manner to the human collaborator. The increasing need to offer shortcuts around the constraints of this model points towards the computational agent not meeting the requirements for completeness. The

use of the application during the composition experiments takes on the qualities of a *task-divided* (different types of contribution from each agent) and *asymmetric* (different amounts of contribution) interaction, where a computational agent does not need to meet this requirement of *completeness*.

As mentioned in the previous section, the robot variation during the co-creative interaction was also often seen as being random, and in the case of the pre-alpha tests, potentially disruptive, as there was no way to avoid the robot contribution. During the composing experiments, where I exposed more options to direct the interaction, this step was only used in a small number of occasions, and always with the idea of shaking up a series of variations based on user feedback that was not delivering fruitful options.

The initial contribution in every interaction session was generated randomly; this could be accepted as a starting point, or a new random option could be generated. During the composing exercise, a set of ten random initial ideas was generated, and ideas which stood out from this limited list of options were chosen for development. Although I set out to avoid a random walk through ideas, this suggests that applying ways to constrain a potential glut of options has a place in such an interaction.

The actual and apparently random contributions by the robot seemed to have a limited use case, with the initial stages of the interaction arousing curiosity and some chance to settle on an idea, and later in the interaction the chance to break out of a development direction that was not showing promise. The use of the application in a larger music production ecosystem, such as in the composing experiments, made for more options to work around the constraints of the system. While the combination of processes that made up the robot's inner workings was not able to deliver the behaviour of a complete computational agent, these behaviours could still be employed usefully outside of the overly rigid turn-taking model in which they were positioned in the pre-alpha and alpha tests.

6.4. Engaging the user

The setting in which the interaction took place seems to have had an impact on the engagement. Two users from the first round of qualitative testing, who sat in face-to-face sessions with myself, spent an hour to two hours using the software and engaging in answering the questions. The online users who went through the process of completing the survey spent on average twenty minutes with the application. Lastly, I undertook a sustained

engagement of approximately twenty-four hours in total to explore the project thoroughly. In each case, the end goal differs: with the face-to-face interviews there was a rich discussion of the algorithm and the user's response to it. In the online interaction, beyond engaging with the application initially there was no end goal other than a multiple-choice questionnaire. In the case of the composing experiments there was a detailed exploration of the uses of the system, with the goal of transferring the miniatures generated by the algorithm into a longer-form work.

The scope of possible musical development also has a bearing on the length of engagement. In the pre-alpha and alpha tests, this development was limited to the length of a phrase, and the user engagement during these tests was of a similar order of magnitude. The second participant in the pre-alpha tests successfully developed a short phrase which was to their liking within the first twenty minutes, but the application offered no options to expand this to a longer form, thus ending engagement with that line of development. The composing experiments expanded the scope of the musical end goal, with a consequent increase in overall engagement time. This suggests that overall goals play a part in the length of time that a user can be engaged. It does not seem that idle curiosity about what the algorithm might do next suggests sustainable engagement beyond the scale of tens of minutes, although this would need to be tested further.

Also, as was intended from the outset, the application definitely behaves as a composing tool, and there was little sense from my interactions that in its current form it suggests a performance tool. Once again this was not explicitly tested.

7. Conclusion and future work

The use of the application in the two composing experiments brought valuable new insights to the project, even though I was both designer and test subject. Once I immersed myself in a composing experiment more substantial than those used during development, I discovered that the need to deploy this tool for other users was less continuously engaging than I imagined it might be.

While there were valuable insights to be learnt from the inclusion of other participants in the process, the personal use of the application as a creative tool has been the most stimulating, and looks to be the focus of the interest I would have in further development of the project. The trend of text-based large language model steering interfaces for such generative applications is already well established in publicly available applications, which is also less engaging as a future prospect than I expected it to be. As both designer and expert user of the system, I am more interested in exploring the details of the instrument/instrumentalist ecology described by Rodger et al. (2020), and the use of a generalised text-based interface is not the most compelling communication possibility for me in that paradigm. My inclination relates more to the interaction offered by systems for modular synthesis which engage the user with a large and sometimes mysterious set of control parameters. My use of text descriptions for feedback are more akin to the labelling of controls than the insertion of layers of human-like language understanding between user and generation algorithm.

7.1. Extending the form

An immediate next step in this project is to introduce a parameter system that relates the phrases generated by the algorithm as it stands for organisation into longer-form works, essentially systematising the process described in Figure 30 and attaching a variation and steering mechanism to it.

7.2. Data gathering

A significant further development would be resolving the behaviour of the data gathering process. Several approaches suggest themselves. The idea of few shot learning, as mentioned by Ng (2023), is compelling. In such a model, a more general-purpose deep learning model is

trained on the knowledge domain to be addressed. During use, the model is able to accept examples (generally a small number) that guide its behaviour for that particular task. The nature of a pre-trained AI for such an implementation would require some research and could consist of a significant research outlay were I to attempt to develop it myself. While this aligns with the general trend for harnessing the power of modern deep learning models, it is not the only option.

This project suggests that there is much that can be done to improve the way I approached sentiment capture, by increasing the resolution of feedback, inferring this higher resolution from the questions asked, exposing the workings of this decision-making process to an interactive development process of its own, and expanding the options for delivering feedback, in particular in the form of musical artefacts. This leads to the idea of enabling the system to ingest and analyse musical artefacts, which was an area of experimentation in the current project that was excluded from the final version due to time constraints. This mechanism has a variety of applications in the ecosystem of this research. There are different technical issues for ingesting this material into the system depending on whether it is delivered as symbolic or audio, but once a musical artefact is expressed in the same set of parameters that are used for generation it represents new information to adjust the behaviour of the system. This could encompass feedback along the lines of 'Do something more like this clip', or the user manually editing a generated artefact in their editor of choice and offering it as the next step in a co-creative interaction, or giving the system the ability to capture live played material from the user and generate a response as a real-time musical collaborator.

7.3. Explainable AI

Mental models, long-term memory and argumentation (Llano et al., 2020) are techniques employed to address the issue of explainable AI behaviour. In this application, a *mental model* of the user's preferences was built up by the application to allow it to act in a way that could balance the output of the automated composing process against these preferences by outputting material that was more pleasing or provoking to the user. During testing, where this had not yet been implemented, it was interesting to note that there was intermittent sense that the application was aware of the direction perceived by the user. Later observations by myself in the composing exercises suggested that in situations where this mechanism was implemented, there did not seem to be an improvement in this sense of a shared goal. This

points to future work to more clearly understand how this mechanism can be made to palpably improve the interaction.

In terms of *argumentation*, the application provides specific points (music-specific generation parameters) to allow a user to communicate a traversal vector during the interaction, but the output of the robot is a black box moment. Due to the somewhat ineffective steering of the robot output by considering past user preferences, as mentioned above, the process whereby the robot generates new material becomes a liminal point at which musical direction might change in an unexpected, and sometimes unwanted, direction. An approach to examining this problem with the current system would be to ascertain whether it is a case of a flawed conceptual model for steering the robot variation generation or whether it is a matter of tuning the existing parameters to achieve different (and better) results. Future work would point to analysing the difference in the data related to the generation by the user – as opposed to robot variations. While the data for this difference exist, they are quite voluminous and in their current state the application has no way of communicating this in a precise and succinct way such that it may be understandable or useable in a creative interaction.

7.4. Accompaniment material

The need for accompaniment material to shape the genre, mood and narrative arc of a composition suggests there is considerable discovery to be made in how an algorithm generating such material can be made to intersect with various accompaniment instruments, playing styles and music production techniques. It is impossible to say whether a rule-based or data-driven system might be favoured going forward, but a model that caters for the blending of concepts, as suggested by Boden's (2004) combinatorial or transformational creativity (described in 2.1.1) is a promising approach.

7.5. Last thoughts

This research has been compelling for me as a technology maker and as a music practitioner, especially as I did not expect that any composing of my own would form part of this project. While the last few years have seen a lot of writing and coding, and less music making, this project has already precipitated a change to more of the latter, where my interests and past achievements in technology and music begin a new stage of my creative journey.

A last sobering thought relates to the notion of wasteful production. Systems such as these have tremendous power to generate large volumes of material, much more than can be made sense of by a single musical life. Reshaping the behaviour of the computational creative process to better reflect a human creative attention span – to create works that carry the imprint of the artist's hand (so to speak) – will be among my aspirations for future work.

8. Bibliography

Akama, T. (2021). A Contextual Latent Space Model: Subsequence Modulation in Melodic Sequence. International Symposium on Music Information Retrieval. https://doi.org/10.48550/arXiv.2111.11703

Alkaissi, H., & Mcfarlane, S. I. (2023). Artificial Hallucinations in ChatGPT : Implications in Scientific Writing. Cureus, 15(2), e35179. https://doi.org/10.7759/cureus.35179

Amabile, T. M. (1996). Creativity in context: Update to "The Social Psychology of Creativity." In Creativity in context: Update to "The Social Psychology of Creativity." Boulder, CO, US: Westview Press.Anguiano, D. (2023, 27 September). Hollywood Writers Agree to End Five-Month Strike after New Studio Deal. The Guardian. https://www.theguardian.com/culture/2023/sep/26/hollywood-writers-strike-ends-studio-deal (Accessed 15 January 2024).

Armstrong, D. (2017). Parasitic Frankensteinism: Extraction of Repetition Schemas of Musical Features for Use in an Algorithm for Composing Music. Unpublished master's thesis, AFDA, The School for the Creative Economy, Cape Town.

Bel, B. (1999). Portrait of an Extraterrestrian. In J. Tabor (Ed.), Otto Laske: Navigating new musical horizons (pp. 81–92). Westport: Greenwood press.Biles, J. (1994). GenJam: A Genetic Algorithm for Generating Jazz Solos. Proceedings of the International Computer Music Conference 1994, IGMA, SanFransisco.

Bitar, J. (2023, 2 May). Les 5 dangers de l'IA selon Geoffrey Hinton, un de ses pionniers.The [The Five Dangers of AI According to Geoffrey Hinton, One of Its Pioneers]. Le Point. https://www.lepoint.fr/sciences-nature/les-5-dangers-de-l-ia-selon-geoffrey-hinton-un-de-ses-pionniers-02-05-2023-2518646_1924.php#xtmc=intelligence-artificielle&xtnp=1&xtcr=8 (Accessed 31 December 2023).

Boden, M. A. (2004). The Creative Mind: Myths and Mechanisms. Hove: Psychology Press.

Boden, M. A. (2014). Creativity and Artificial Intelligence: A Contradiction in Terms? (E. S. Paul & S. B. Kaufman, Eds.). The Philosophy of Creativity: New Essays, p. 0. https://doi.org/10.1093/acprof:oso/9780199836963.003.0012

Encyclopaedia Britannica. (2021). Johann Sebastian Bach summary. https://www.britannica.com/summary/Johann-Sebastian-Bach (Accessed 8 February 2024)

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G.,

Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., & Armodei, D. (2020). Language Models Are Few-Shot Learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds), Advances in Neural Information Processing Systems (pp. 1877–1901). Retrieved from https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

Burleson, W. (2005). Developing Creativity, Motivation, and Self-Actualization with Learning Systems. International Journal of Human–Computer Studies, 63(4–5), 436–451. https://doi.org/10.1016/j.ijhcs.2005.04.007

Carlini, N., Lee, K., Ippolito, D., Tramèr, F., Jagielski, M., & Zhang, C. (2023). Quantifying Memorization across Neural Language Models. The Eleventh International Conference on Learning Representations, 1–19. https://openreview.net/forum?id=TatRHT_1cK (Accessed 8 February 2024)

Chadabe, J. (1977). Some Reflections on the Nature of the Landscape within which Computer Music Systems Are Designed. Computer Music Journal, 1(3), 5–11.

Chiticariu, L., & Reiss, F. R. (2013). Rule-based Information Extraction is Dead! Long Live Rule-based Information Extraction Systems! Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 827–832. https://aclanthology.org/D13-1079 (Accessed 27 December 2023)

Chomsky, N., & Polychroniou, C. J. (2023, 3 May). Noam Chomsky Speaks on What ChatGPT Is Really Good For. Noam Chomsky Interviewed by C.J. Polychroniou. https://chomsky.info/20230503-2 (Accessed 20 Nivember 2023)

Cook, M., Colton, S., Pease, A., & Llano, M. T. (2019). Framing in Computational Creativity: A Survey and Taxonomy. In K. Grace, M. Cook, D. Ventura, & M. L. Maher (Eds), Proceedings of the 10th International Conference on Computational Creativity (pp. 156–163). https://www.possibilityspace.org/papers/iccc19.pdf (Accessed 21 February 2024)

Csikszentmihalyi, M. (1990). Flow: The Psychology of Optimal Experience. New York: Harper & Row.

Csikszentmihalyi, M. (2014). Flow and the Foundations of Positive Psychology. London: Springer.

Dai, S., Jin, Z., Gomes, C., & Dannenberg, R. B. (2021). Controllable Deep Melody Generation via Hierarchical Musical Structure Representation. Proceedings of the 22nd International Society for Music Information Retrieval Conference, 2021. https://doi.org/10.48550/arXiv.2109.00663

Davis, E. (1998). Techgnosis: Myth, Magic, and Mysticism in the Age of Information. New York: Harmony Books.

Di Nunzio, A. (2013). Csound. www.musicainformatica.org/topics/csound.php (Accessed 13 February 2024.)

Dilmegani, C. (2023). Top 100+ Generative AI Applications/Use Cases in 2023. AI Multiple Research. https://research.aimultiple.com/generative-ai-applications/ (Accessed 25 November 2023).

Ebcioğlu, K. (1988). An Expert System for Harmonizing Four-Part Chorales. Computer Music Journal, 12(3), 43–51.

Edwards, M. (2011). Algorithmic Composition: Computational Thinking in Music. Communications of the ACM, 54(7), 58–67. https://cacm.acm.org/magazines/2011/7/109891-algorithmic-composition/fulltext#F3 (Accessed 13 December 2019)

Fallman, D. (2008). The Interaction Design Research Triangle of Design Practice, Design Studies, and Design Exploration. Design Issues, 24(3), 4–18. https://doi.org/10.1162/desi.2008.24.3.4

Fernandez, J., & Vico, F. (2013). AI Methods in Algorithmic Composition: A Comprehensive Survey. Journal of Artificial Intelligence Research, 48, 513–582.

Ferrara, E. (2023). Should ChatGPT Be Biased? Challenges and Risks of Bias in Large Language Models. First Monday. https://doi.org/10.5210/fm.v28i11.13346

Finke, R. A., Ward, T. B., & Smith, S. M. (1992). Creative Cognition: Theory, Research, and Applications. Cambridge, MA: MIT Press.

Fokkink, W. (2007). Introduction to Process Algebra (2nd edition). Cham: Springer.

Frich, J., Vermeulen, L. M., Remy, C., & Biskjaer, M. M. (2019). Mapping the Landscape of Creativity Support Tools in HCI. CHI Conference on Human Factors in Computing Systems. https://dl.acm.org/doi/10.1145/3290605.3300619 (Accessed 24 January 2020)

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software. Upper Saddle River: Addison-Wesley Longman.

Gibson, P. M., & Byrne, J. (1991). NEUROGEN, musical composition using genetic algorithms and cooperating neural networks. Proceedings of the International Con-Ference on Artificial Neural Networks, 309–313. Retrieved from https://api.semanticscholar.org/CorpusID:58990202

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Nets. Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014), 2672–2680. https://papers.nips.cc/paper_files/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html (Accessed 8 February 2024)

UCL News. (2023, 13 December). Google DeepMind co-founder discusses how AI can benefit humanity at UCL. https://www.ucl.ac.uk/news/2023/dec/google-deepmind-co-founder-discusses-how-ai-can-benefit-humanity-ucl (Accessed 15 January 2024)

Gothié, J.-D., Demeneix, B., & Remaud, S. (2017). An Introduction to Petri Nets and How to Analyse Them . . . Tutorial. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiKr5qomeqEAxWBQUEAHWhWAU0QFnoECA8QAQ&url=https%3A%2F%2Fverif.ulb.ac.be%2Fggeeraer%2FTutorial-Perti-Nets-Geeraerts.pdf&usg=AOvVaw1CmrItZGRpjgLx8m7Q0miV&opi=89978449 (Accessed 6 March 2020)

Green, T. (1989). Cognitive Dimensions of Notations. People and Computers, V, 443–460. https://doi.org/10.1109/vlhcc.2005.26

Hamman, M. (1999). Structure as Performance: Cognitive Musicology and the Objectification of Compositional Procedure. In J. Tabor (Ed.), Otto Laske: Navigating new musical horizons (pp. 37–52). Westport: Greenwood press.

Heaven, W. D. (2023, 2 May). Geoffrey Hinton Tells Us Why He's Now Scared of the Tech He Helped Build. MIT Technology Review. https://www.technologyreview.com/2023/05/02/1072528/geoffrey-hinton-google-why-scared-ai/ (Accessed 31 December 2023).

Hewett, T. T., Baecker, R., Card, S., Carey, T., Gasen, J., Mantei, M., Perlman, G., Strong, G., & Verplank, W. (1992). ACM SIGCHI Curricula for Human–Computer Interaction. New York: Association for Computing Machinery. https://dl.acm.org/doi/book/10.1145/2594128 (Accessed)

Hsu, K. J., & Hsu, A. (1991). Self-Similarity of the "1/f Noise" Called Music. Proceedings of the National Academy of Sciences of the United States of America, 88(8), 3507–3509. https://doi.org/10.1073/pnas.88.8.3507

ISO/IEC. (2019). ISO 9241-210: Ergonomics of Human-System Interaction Part 210: Human-Centred Design for Interactive Systems.

James, G., Hastie, T., Tibshirani, R., & Witten, D. (2013). *An introduction to statistical learning : with applications in R*. New York: Springer.

Jarvelainen, H. (2000). Algorithmic Musical Composition. Seminar on Content Creation, Telecommunications Software and Multimedia Laboratory, Helsinki University of Technology.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D. A. N., Xu, Y. A. N., & Ishii, E. (2022). Survey of Hallucination in Natural Language Generation. ACM Computing Surveys, 1(1).

Jordanous, A. K. (2012). Evaluating Computational Creativity: A Standardised Procedure for Evaluating Creative Systems and Its Application (University of Sussex). https://www.academia.edu/831189/Evaluating_Computational_Creativity_A_Standardised_Procedure_for_Evaluating_Creative_Systems_and_its_Application?auto=download (Accessed 16 January 2020)

Kantosalo, A. (2019). Human-Computer Co-Creativity: Designing, Evaluating and Modelling Computational Collaborators for Poetry Writing. PhD thesis, University of Helsinki. http://hdl.handle.net/10138/303928

Kantosalo, A., Thattai Ravikumar, P., Grace, K., & Takala, T. (2020). Modalities, Styles and Strategies: An Interaction Framework for Human–Computer Co-Creativity. In A. Cardoso, P. Machado, T. Veale, & J. M. Cunha (Eds), Proceedings of the Eleventh International Conference on Computational Creativity (pp. 57–64). http://computationalcreativity.net/iccc20/papers/ICCC20_Proceedings.pdf (Accessed 3 March 2022)

Kantosalo, A., & Toivonen, H. (2016). Modes for Creative Human-Computer Collaboration: Alternating and Task-Divided Co-Creativity. Proceedings of the Seventh International Conference on Computational Creativity (June), 82–89.

Karimi, P., Grace, K., Maher, M. Lou, & Davis, N. (2019). Evaluating Creativity in Computational Co-Creative Systems. (July 2018). https://www.researchgate.net/publication/326646917_Evaluating_Creativity_in_Computational_Co-Creative_Systems (Accessed 16 January 2020)

Kippen, J., & Bel, B. (1989). Can the computer help resolve the problem of ethnographic description? Anthropological Quarterly, 62(3), 131–140.

Kronfeldner, M. (2018). Explaining creativity. In Berys Gaut & M. Kieran (Eds.), Routledge Handbook on Creativity and Philosophy (pp. 213–229). Routledge.

Lerdahl, F., & Jackendoff, R. (1983). An Overview of Hierarchical Structure in Music. Music Perception: An Interdisciplinary Journal, 1(2), 229–252. https://doi.org/10.2307/40285257

Lo, M., & Lucas, S. M. (2006). Evolving Musical Sequences with N-Gram Based Trainable Fitness Functions. Proceedings of the IEEE Conference on Evolutionary Computation, 601–608.

Li, C. (2020, 3 June). OpenAI's GPT-3 Language Model: A Technical Overview. Lambda. https://lambdalabs.com/blog/demystifying-gpt-3 (Accessed 29 November 2023).

Li, X. (2020). The Application of Computer Technology in Modern Entertainment Media. IOP Conference Series: Materials Science and Engineering, 750(1), 12068. https://doi.org/10.1088/1757-899X/750/1/012068

Llano, M. T., D'Inverno, M., Yee-King, M., McCormack, J., Ilsar, A., Pease, A., & Colton, S. (2020). Explainable Computational Creativity. In F. A. Cardoso, P. Machado, T. Veale, & J. M. Cunha (Eds.), Proceedings of the 11th International Conference on Computational Creativity ICCC'20, Coimbra, Portugal, 334–341. Retrieved from http://computationalcreativity.net/iccc20/papers/ICCC20_Proceedings.pdf (Accessed 3 March 2022)

Lubart, T. (2005). How Can Computers Be Partners in the Creative Process: Classification and Commentary on the Special Issue. International Journal of Human Computer Studies, 63(4–5), 365–369. https://doi.org/10.1016/j.ijhcs.2005.04.002

Marcus, G., & Southen, R. (2024, January). Generative AI Has a Visual Plagiarism Problem: Experiments with Midjourney and DALL-E 3 Show a Copyright Minefield. https://spectrum.ieee.org/midjourney-copyright (Accessed 6 February 2024).

Maurer, J. A. (1999). A Brief History of Algorithmic Composition. https://ccrma.stanford.edu/~blackrse/algorithm.html (Accessed 28 July 2019)

McIntyre, R. A. (1994). Bach in a box: the evolution of four part Baroque harmony using the genetic algorithm. Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence, 852–857. Retrieved from https://api.semanticscholar.org/CorpusID:31407823

McKay, C., Cumming, J. E., & Fujinaga, I. (2018). jSymbolic 2.2: Extracting Features from Symbolic Music for Use in Musicological and MIR Research. Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, 348–354.

Mishra, P. (2019). Constraint Satisfaction Problems: Definition and Examples. https://study.com/academy/lesson/constraint-satisfaction-problems-definition-examples.html (Accessed 29 January 2020).

Murphy, M. G. (2023, 19 March). E-scaping Responsibility and Enjoyment through ChatGPT: A New Unconscious? Sublation Magazine https://www.sublationmag.com/post/chatgpt-a-new-unconscious (Accessed 17 January 2024)

Murray, J. H. (2012). Inventing the Medium: Principles of Interaction Design as a Cultural Practice. Cambridge, MA: MIT Press.

Nakakoji, K. (2006). Meanings of Tools, Support, and Uses for Creative Design Processes. Proceedings of International Design Research Symposium'06, 2006, 156–165.

Natale, S., & Ballatore, A. (2020). Imagining the Thinking Machine: Technological Myths and the Rise of Artificial Intelligence. Convergence: The International Journal of Research into New Media Technologies, 26(1), 3–18. https://doi.org/10.1177/1354856517715164

Ng, A. (2023). Andrew Ng: Opportunities in AI – 2023. https://www.youtube.com/watch?v=5p248yoa3oE&t=932s (Accessed 4 December 2023).

Nierhaus, G. (2009). Algorithmic Composition Paradigms of Automated Music Generation. Germany: Springer-Verlag/Wien.

Nolan, B. (2023, 2 November). The "Stakes Are Too High" to Ignore Extinction Risks of AI, AI Godfather Warns. Business Insider https://www.businessinsider.com/meta-ai-chief-yoshua-bengio-risk-yann-lecun-ai-godfather-2023-11?r=US&IR=T#:~:text=Claims by Meta's chief AI,of his fellow AI godfathers (Accessed 15 January 2024).

Norman, K. L. (2008). Cyberpsychology: An Introduction to Human–Computer Interaction. Cambridge: Cambridge University Press.

O'Brien, M. (2023, 5 December). AI's Future Could Be 'Open-Source' or Closed. Tech Giants Are Divided as They Lobby Regulators. AP News. https://apnews.com/article/ai-opensource-meta-ibm-chatgpt-dd61e99ac8135b36872b3987601067ec (Accessed 15 January 2023).

Pachet, F. (2002). The Continuator: Musical Interaction with Style. Proceedings of the International Computer Music Conference ICMC 2002, Gotheburg, Sweden.

Papadopoulus, G., & Wiggins, G. (1999). AI Methods for Algorithmic Composition: A Survey, a Critical View and Future Prospects. Proceedings of the AISB '99 Symposium on Musical Creativity. 110–117. AISB (Society for the Study of Artificial Intelligence and the Simulation of Behaviour).

Parnami, A., & Lee, M. (2022). Learning from Few Examples: A Summary of Approaches to Few-Shot Learning. https://doi.org/10.48550/arXiv.2203.04291

Paul, E. S., & Stokes, D. (2018). Atrributing Creativity. In B. Gaut & M. Kieran (Eds.), Creativity and Philosophy. Routledge.

Reddin, J., Mcdermott, J., & Neill, M. O. (2009). Elevated Pitch : Automated Grammatical Evolution of Short Compositions Elevated Pitch : Automated Grammatical Evolution of Short Compositions. Conference on Applications of Evolutionary Computation, 579–584. https://doi.org/10.1007/978-3-642-01129-0

Rodger, M., Stapleton, P., Walstijn, M. Van, Ortiz, M., & Pardue, L. (2020). What Makes a Good Musical Instrument ? A Matter of Processes , Ecologies and Specificities. Proceedings of the International Conference on New Interfaces for Musical Expression, (September), 405–410. https://doi.org/10.5281/zenodo.4813438

Runco, M. A., & Jaeger, G. J. (2014). The Standard Definition of Creativity The Standard Definition of Creativity. (January 2012). https://doi.org/10.1080/10400419.2012.650092

Schick, T., & Schutze, H. (2022). True Few-Shot Learning with Prompts: A Real-World Perspective. Transactions of the Association for Computational Linguistics, 10, 716–731.

Shneiderman, B. (2007). Creativity Support Tools: Accelerating Discovery and Innovation. Communications of the ACM, 50(12), 20–32. https://doi.org/10.1145/1323688.1323689

Simonton, D. K. (2014). Hierarchies of creative domains: Disciplinary constraints on blind variation and selective retention. In The philosophy of creativity: New essays. (pp. 247–261). https://doi.org/10.1093/acprof:oso/9780199836963.003.0013

Sunray, E. (2021). Sounds of Science: Copyright Infringement in AI Music Generator Outputs. The Catholic University Journal of Law & Technology, 29(2), 185–218.

Tan, H. H., & Herremans, D. (2020). Music Fadernets: Controllable Music Generation Based on High-Level Features via Low-Level Feature Modelling. International Symposium on Music Information Retrieval.

Tanaka, A. (2009). Sensor-Based Musical Instruments and Interactive Music. In R. T. Dean (Ed.), The Oxford Handbook of Computer Music (pp. 233–257). Oxford: Oxford University Press.

There's an AI for That. (2024). https://theresanaiforthat.com/ai/amper-ai/ (Accessed 15 January 2024).

Trinh, T. H., Wu, Y., Le, Q. V, He, H., & Luong, T. (2024). Solving Olympiad Geometry without Human Demonstrations. Nature, 625, 476–482. https://doi.org/10.1038/s41586-023-06747-5

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. Proceedings of the 31st Conference on Neural Information Processing Systems, 6000–6010.

Velardo, V. (2017). Towards a Music Systems Theory: Theoretical and Computational Modelling of Creative Music Agents. https://www.academia.edu/34363613/Towards_a_Music_Systems_Theory_Theoretical_and_Computational_Modelling_of_Creative_Music_Agents?email_work_card=view-paper (Accessed 16 January 2020)

Velkova, J. (2017). Media Technologies in the Making. PhD thesis, Södertörn University. Stockholm.

Voss, R. F., & Clarke, J. (1975). '1/fnoise' in music and speech. Nature, 258(5533), 317–318. https://doi.org/10.1038/258317a0Waltl, B., Bonczek, G., & Matthes, F. (2018). Rule-Based Information Extraction: Advantages, Limitations, and Perspectives. Conference on Empirical Methods in Natural Language Processing. https://api.semanticscholar.org/CorpusID:216006964

Weisstein, E. W. (n.d.). Self-Similarity. MathWorld: A Wolfram Web Resource. http://mathworld.wolfram.com/Self-Similarity.html (Accessed 4 February 2020).

Wiggins, G. A. (2003). Categorising Creative Systems. https://www.researchgate.net/publication/2902663_Categorising_Creative_Systems (Accessed 1 Jan 2020)

Wiggins, G. A. (2006). A Preliminary Framework for Description, Analysis and Comparison of Creative Systems. Knowledge Based Systems, 19(7), 449–458. https://doi.org/10.1016/j.knosys.2006.04.009

Yi, S., Lazzarini, V., & Costello, E. (2018). WebAssembly AudioWorklet Csound. Proceedings of the International Web Audio Conference. Retrieved from https://webaudioconf.com/_data/papers/pdf/2018/2018_24.pdf (Accessed 7 January 2023)

Yim, K., & Manickavasagam, H. (2023). Turn ideas into music with MusicLM. Retrieved January 3, 2024, from https://blog.google/technology/ai/musiclm-google-ai-test-kitchen/ (Accessed 3 January 2024)

Yin, Z., Reuben, F., Stepney, S., & Collins, T. (2023). Deep Learning's Shallow Gains: A Comparative Evaluation of Algorithms for Automatic Music Generation. In Machine Learning, 112, 1785–1822. https://doi.org/10.1007/s10994-023-06309-w

Young, M., & Bown, O. (2010). Clap-Along: A Negotiation Strategy for Creative Musical Interaction with Computational Systems. Proceedings of the International Conference on Computational Creativity, ICCC-10, 215–222.

Zhao, J., & Xia, G. (2021). Accomantage: Accompaniment Arrangement via Phrase Selection and Style Transfer. International Symposium on Music Information Retrieval, 833–840.

Zimmerman, J., Forlizzi, J., & Evenson, S. (2007). Research Through Design as a Method for Interaction Design Research in HCI. Conference on Human Factors in Computing Systems - Proceedings, (May 2014), 493–502. https://doi.org/10.1145/1240624.1240704

Žižek, S. (2023a, 7 April). ChatGPT Says What Our Unconscious Radically Represses.

Sublationmag. https://www.sublationmag.com/post/chatgpt-says-what-our-unconscious-radically-represses (Accessed 9 November 2023).

Žižek, S. (2023b, 7 April). The Post-Human Desert. Project Syndicate. https://www.project-syndicate.org/commentary/ai-post-human-future-by-slavoj-zizek-2023-04?barrier=accesspaylog (Accessed 15 January 2024).

Appendix A: First version of the random generation output

Some editing of the text files has been done to make them more readable. The original versions had no gaps between sections, spaces between words and symbols, or line breaks in lists.

A1. The set of parameter options

This contains the list of options or ranges of output values. Square brackets indicate a comma separated list of options and the content has been shortened for readability. I have kept the camel case formatting of the names of the parameters as they appear in the code, and I have separated parameters and the available options from the descriptions using different font.

`phraseLengthOptions:  [4, 6, 8, 10]`

- in bars
-

`timeSignatureOptions:  [4/4, ¾, 7/8, 5/8, 6/8]`

- numerator/denominator as in a written score

`minimumPhraseStartPercent  -0.2`

`maximumPhraseStartPercent  0.0`

- a percentage range for the start of the area in the total bar length where structure tones will be placed. Structure tone placing is quantized based on the structure tone spacing options, so in this case, the choice of range -0.2 – 0.0 assures that the first structure tone will be on the beginning of the first bar.

`minimumPhraseEndPercent  0.45`

`maximumPhraseEndPercent  0.76`

- percentage range for the end of the melody area.

`structureToneSpacingOptions:  [1.0]`

- value used for positioning and quantizing the structure tones, expressed in FloatBars (where the whole number represents a bar count and the decimal a percentage positioning within the bar). In this case, the only option used was a spacing and quantization of 1 bar, so the structure tones would be placed at the beginning of each subsequent bar if that position was found to be within the start and end percent of the total bar length.

```
minimumTempo:      65

maximumTempo:      145
```

- tempo range in beats per minute

```
xmlKeyOptions:     [-6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6]
```

- key options expressed as in the .musicxml format, with positive numbers being sharp keys and negative flat keys. So, for instance, 0 is no sharps or flats, -4 is 4 flats, 3 is 3 sharps etc.

-

```
minimumStartNote: 50

maximumStartNote: 70
```

- range of the first note of the melody, expressed in MIDI pitches. The pitch of the first note would be the closest chord tone to this start note based on the chord tones of the prevailing chord for that position

```
structureToneContourOptions:

      [(0.0, 0.0), (1.0, -1.0)],

      [(0.0, 0.0), (1.0, 1.0)],

      [(0.0, 0.0), (0.5, -1.0), (1.0, 0.0)],

       [(0.0, 0.0), (0.5, 1.0), (1.0, 0.0)]
```

- - these are x, y points on a break point graph with x range of 0.0 to 1.0 and y range of -1.0 to 1.0. The x range is a percentage of the melody area in the phrase as delineated by phrase start and end percent.

```
structureToneContourRangeMinimum:   5

structureToneContourRangeMaximum:   14
```

- minimum and maximum multiplier for the y value of the structure tone contour, expressed in relative MIDI pitches. The y position of the contour graph in the previous point for each structure tone position multiplied by the number generated within these constraints, offset from the start note and then quantized to the nearest chord tone.

```
clearanceOfEmbellishmentFromPreviousStructureToneOptions:   1.0

clearanceOfEmbellishmentFromPreviousStructureToneOptions:   0.5
```

- embellishments are added before each structure tone, and run the risk of colliding with the previous structure tone. This value dictates the clearance of embellishments from previous structure tones, expressed in quarter notes

```
embellishmentCountOptions:    [1, 2, 3, 4]
```

- options for the number of embellishments applied to each structure tone

```
embellishmentRepetitionPatterns:    [0], [0,1], [0,1,2], [0,1,0,2],
[0,1,2,3], [0,1,0,2,3], [0,1,0,1,2], [0,1,0,2,1], [0,0,1], [0,0,1,1],
[0,0,1,2]
```

- these are lists of indices to sets of embellishments. The set of embellishments referred to by each subsequent index is applied to each subsequent structure tone, and loops around to the beginning if the list if there exist more structure tones than embellishment set indices.

```
embellishmentPitchOptions:

      (IS_ESCAPE_TONE,UNACCENTED,STEP_JUMP)

      (IS_ESCAPE_TONE,UNACCENTED,JUMP_STEP)

      (IS_ANTICIPATION)
```

- this is a list of pitch models that can be applied to an embellishment. It must be noted that what I label as an ‘escape tone’ is just a label for a particular computational handling of an embellishment pitch, which could describe what is generally known as an escape tone, but can also describe other patterns. The parameters available for my ‘escape tone’ are whether it is accented or not, and in what order the jump and the step are applied. An accented tone will displace a structure tone to a later, weaker beat and assume its position. A step-jump type will take a melodic step from the previous note in the opposite direction to the destination note, whereas the jump-step type will approach the destination note be a step from the direction opposite to the previous note. In cases where the previous and subsequent notes are the same, the resultant embellishment will look like a neighbour tone. Due to the notes being the same, a direction contour cannot be calculated and the usage needs a further parameter to handle this eventuality, which is not described in this iteration of the algorithm. Lastly, due to these embellishments being applied one at a time, the best available previous note at that time in the calculation may not be the actual previous note in the finished melody, as further embellishments may intervene, and result in the note bearing no resemblance to an escape tone at all.

```
embellishmentRhythmOptions:

      (0,0,0,-1),

      (0,0,-1,0),

      (0,0,-1,-1),
```

```
(0,0,-2,0)
```

- offset of a given embellishment from the note that it is embellishing. This format of four integers represents offsets at different hierarchical beat levels: bar, super tactus, tactus and sub tactus. These terms are easily described by referring to 4/4 time signature, where each subsequent level is half the time value of the previous: bar (whole note) = 2 x supertactii (half note), supertactus = 2 x tactii (quarter notes), tactus = 2 x sub tactii (eighth notes). Of course, when moving beyond the comfortable symmetry of 4/4 time, things begin to overlap. ¾ and 2/4 time has no separation between bars and supertactus, 7/8 and 5/8 time have uneven tactii, and when moving to possible long time signatures (13/4 was a test case during development) there are a set of possible sub hierarchies between bar and super tactus. The general approach taken was to in all cases be able to come up with a computed value, even if this value might be explained in several ways in this format.

A2. The random number file

The random number file is a version of the random number sequence used to generate the random output. Each number is in the range 0.0 to 1.0, and is assigned to the parameter descriptor to which it applies. In some cases, this number is multiplied by the expected range of output values, and in other cases is applied to the available indices of a list of options. For example, in the case of tempo, the range is 65 to 145 beats per minute (bpm), as can be seen in the 'minimumTempo' and 'maximumTempo' values in A1. The 'TEMPO' value below of 0.6789961803838208 is applied as percentage of this range, resulting in a value of 100.865248275051648 which in the case of this parameter would be rounded up to 101 bpm.

```
PHRASE_LENGTH:    0.5800475271730875,
TIME_SIGNATURE:   0.14223986942369216,
PHRASE_START_PERCENT:   0.19496652294592054,
PHRASE_END_PERCENT:     0.4896218460813774,
STRUCTURE_TONE_SPACING: 0.1364394600842831,
TEMPO:      0.6789961803838208,
XMLKEY:     0.4769883625643675,
START_NOTE: 0.06416573855186725,
STRUCTURE_TONE_CONTOUR:       0.08069426403277402,
STRUCTURE_TONE_MULTIPLIER:    0.9945418386985158,
CHORD_LIST_GENERATOR:
      0.01887135265807649,
      0.5094506928123447,
      0.15143826233344015,
      0.2261989897906248,
      0.0815076447820674,
      0.208712162033172,
      0.3582276314782874,
      0.8731902316449216,
EMBELLISHMENT_CLEARANCE:      0.42101280783551454,
EMBELLISHMENT_REPETITION_PATTERN:   0.5688402709022407,
MUG_LISTS:
      0.9460195100686747,
```

0.5561002435968414,

0.2697162063862051,

0.13191530821151243,

0.20554144907279615,

0.08582226308674368,

0.6005330549603213,

0.7846955880588793,

0.8339875143167081,

0.5294619695057202,

0.025692413102276435,

0.9616406988185803,

0.7949735476623866,

0.7560053563952916,

0.5453871504646813,

0.9563123320897414,

0.42133985121541173,

0.5921141820025018,

0.6997438316791661,

0.758247275467003,

0.5907791446651697,

A3. The parameter file

```
phraseLength = 8

timeSignature = 4/4

phraseStartPercent = -0.16100669541081591

phraseEndPercent = 0.601782772285227

structureToneSpacingInFloatBars = 1.0

tempo = 119

xmlKey = 0

startNote = 51

structureToneContour:

      0.0, 0.0

      1.0, -1.0

structureToneMultiplier = 5

chordListGenerator = FloatBarChordProgression:

                        lengthInFloatBars = 8.0

                        0.0:C,

                        1.0:F,

                        2.0:C,

                        3.0:Dm,

                        4.0:C,

                        5.0:Dm,

                        6.0:Em,

                        7.0:Am,

clearanceOfEmbellishmentFromPreviousStructureTone = 1.0

embellishmentRepetitionPattern = 0, 1, 0, 1, 2,

embellishmentIndex = 0

      MuG_EscapeTone_RRP:     JUMP_STEP

                        relativeRhythmicPosition = 0:0:-1:0
                        UNACCENTED

                        UPPER_NEIGHBOUR

      MuG_EscapeTone_RRP:     STEP_JUMP

                        relativeRhythmicPosition = 0:0:0:-1
                        UNACCENTED
```

```
                        UPPER_NEIGHBOUR

      MuG_EscapeTone_RRP:     STEP_JUMP

                        relativeRhythmicPosition = 0:0:-1:-1
                        UNACCENTED

                        UPPER_NEIGHBOUR

      MuG_Anticipation_RRP: relativeRhythmicPosition = 0:0:-2:0


embellishmentIndex = 1

      MuG_EscapeTone_RRP:     STEP_JUMP

                        relativeRhythmicPosition = 0:0:-2:0
                        UNACCENTED

                        UPPER_NEIGHBOUR

      MuG_Anticipation_RRP: relativeRhythmicPosition = 0:0:-2:0

      MuG_EscapeTone_RRP:     JUMP_STEP

                        relativeRhythmicPosition = 0:0:-2:0
                        UNACCENTED

                        UPPER_NEIGHBOUR


embellishmentIndex = 2

      MuG_EscapeTone_RRP:     JUMP_STEP
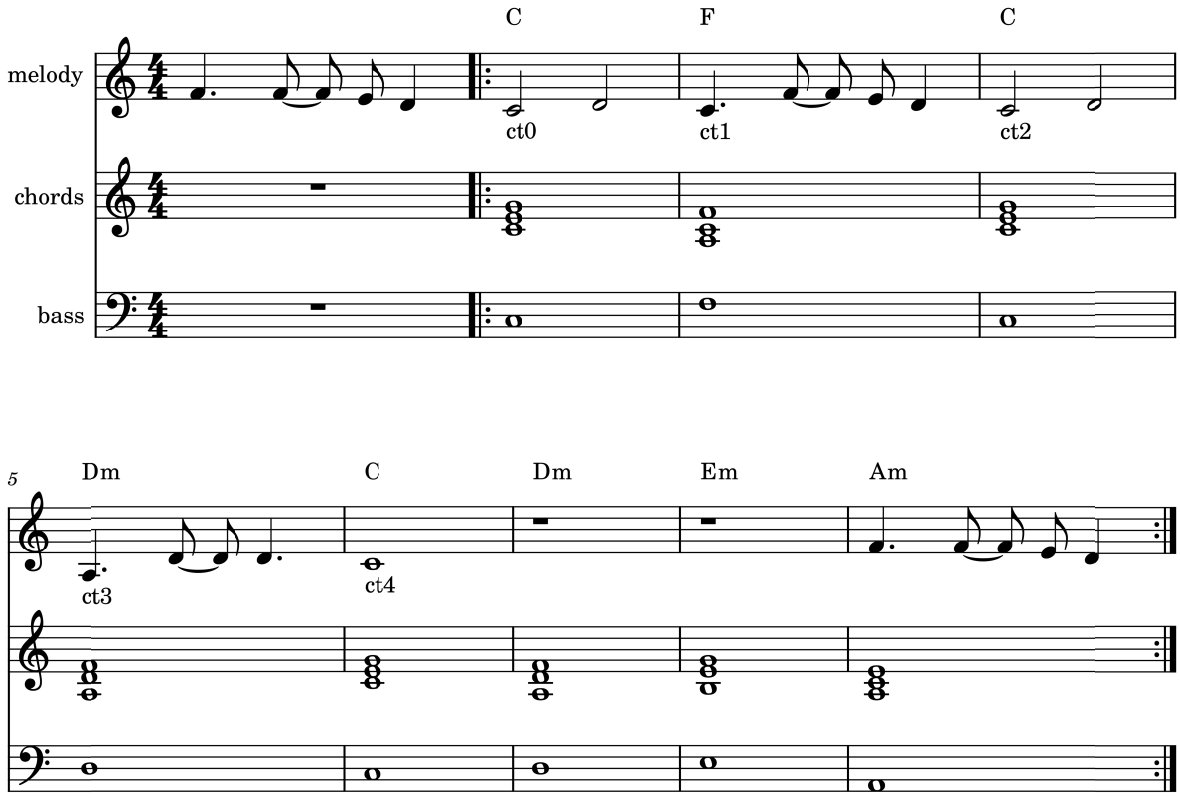
                        relativeRhythmicPosition = 0:0:-1:-1
                        UNACCENTED

                        UPPER_NEIGHBOUR

      MuG_Anticipation_RRP: relativeRhythmicPosition = 0:0:-1:-1
```

A4. The printed output

In this case, note that the generation of rhythm section material was without variation: full bar chords and bass notes, and a rudimentary drum part (not shown) based on the accents of the time signature.



A5. Feedback options for pre-alpha testing

This list is derived from the descriptive class names for the feedback objects in the Java code, and edited for clarity

1. Global feedback options
    a. Tempo_Slower
    b. Tempo_Faster
    c. PhraseLength_Longer
    d. PhraseLength_Shorter
    e. PhraseLength_MoreEven
    f. PhraseLength_LessEven
    g. TimeSignature_MoreEven
    h. TimeSignature_LessEven
    i. Key_Flatter
    j. Key_Sharper

2. Melody feedback options
    a. PhraseStart_Earlier
    b. PhraseStart_Later
    c. PhraseEnd_Earlier
    d. PhraseEnd_Later
    e. StartNote_Higher
    f. StartNote_Lower
    g. MelodyRange_Larger
    h. MelodyRange_Smaller
    i. Melody_EmbellishmentCount_More
    j. Melody_EmbellishmentCount_Less

3. Chord part feedback options
    a. Chord_LessLegato
    b. Chord_MoreLegato
    c. Chord_FavourInnerBeatStrengths
    d. Chord_FavourOuterBeatStrengths
    e. Chord_FavourStrongBeatStrengths
    f. Chord_FavourWeakBeatStrengths
    g. Chord_InterOnsetInterval_Longer
    h. Chord_InterOnsetInterval_Shorter
    i. Chords_MoreEmbellishments
    j. Chords_LessEmbellishments
    k. Chords_NoEmbellishments

4. Bass part feedback options
    a. Bass_MoreLegato
    b. Bass_LessLegato
    c. Bass_MoreEmbellishments

d. Bass_LessEmbellishments
e. Bass_NoEmbellishments
f. Bass_Embellishment_InterOnsetInterval_Longer
g. Bass_Embellishment_InterOnsetInterval_Shorter

5. Drum part feedback options
    a. KikSnare_LessDrive
    b. KikSnare_MoreDrive
    c. HiHat_LessDrive
    d. HiHat_MoreDrive
    e. KikSnare_LessSyncopation
    f. KikSnare_MoreSyncopation
    g. HiHat_LessSyncopation
    h. HiHat_MoreSyncopation

Appendix B: The online survey

Start of Block: Participant info

Q1 Select the items which apply to you (multiple options can be selected)

- ☐ I am a composer (1)
- ☐ I am a performing musician (2)
- ☐ I am a DJ (3)

Q2 I have formal music training

- ○ Yes (1)
- ○ No (2)

Q3 My music experience is mostly classical

- ○ Yes (1)
- ○ No (2)

Q4 My music experience is mostly jazz

- ○ Yes (1)
- ○ No (2)

Q5 My music experience is mostly popular music

- ○ Yes (1)
- ○ No (2)

Q6 I have used an algorithmic music composing tool before

- Yes (1)
- No (2)

Skip To: Q9 If I have used an algorithmic music composing tool before = No

Q7 I use an algorithmic composing tool regularly

- Yes (1)
- No (2)

Skip To: Q9 If I use an algorithmic composing tool regularly = No

Q8 An algorithmic composing tool is a large part of my regular music practice

- Yes (1)
- No (2)

Q9 I use a digital audio workstation in my general musical practice

- Yes (1)
- No (2)

Q10 I use music notation software in my general musical practice

- Yes (1)
- No (2)

Q11 I use a visual coding language for music generation in my general musical practice

- Yes (1)
- No (2)

Q12 I use a coding language for music generation in my general musical practice

- Yes (1)
- No (2)

Q13 I use a conventional musical instrument in my general musical practice

- Yes (1)
- No (2)

End of Block: Participant info

Start of Block: Interaction

Q14 The structure of the interaction was easy to understand

- Yes (1)
- No (2)

Q15 The application offered clear choices

- Strongly agree (1)
- Somewhat agree (2)
- Neither agree nor disagree (3)
- Somewhat disagree (4)
- Strongly disagree (5)

Q16 The musical feedback choices made (not the like/dislike mechanism) generated expected changes to the existing material.

- All the time (1)
- Most of the time (2)
- As often as not (3)
- Seldom (4)
- Never (5)

Q17 The control of the melody instrument was easy to understand

- Strongly agree (1)
- Somewhat agree (2)
- Neither agree nor disagree (3)
- Somewhat disagree (4)
- Strongly disagree (5)

Q18 The control of the chord instrument was easy to understand

- Strongly agree (1)
- Somewhat agree (2)
- Neither agree nor disagree (3)
- Somewhat disagree (4)
- Strongly disagree (5)

Q19 The control of the bass instrument was easy to understand

- Strongly agree (1)
- Somewhat agree (2)
- Neither agree nor disagree (3)
- Somewhat disagree (4)
- Strongly disagree (5)

Q20 The control of the drums was easy to understand

- Strongly agree (1)
- Somewhat agree (2)
- Neither agree nor disagree (3)
- Somewhat disagree (4)
- Strongly disagree (5)

Q21 The musical terminology was easy to understand

- Strongly agree (1)
- Somewhat agree (2)
- Neither agree nor disagree (3)
- Somewhat disagree (4)
- Strongly disagree (5)

Q22 There was a sense of musical continuity in the development of a musical idea

- Strongly disagree (1)
- Somewhat disagree (2)
- Neither agree nor disagree (3)
- Somewhat agree (4)
- Strongly agree (5)

Q23 As an interaction continued, the musical material became more to my liking.

- Strongly disagree (1)
- Somewhat disagree (2)
- Neither agree nor disagree (3)
- Somewhat agree (4)
- Strongly agree (5)

Q24 The musical idea progressed rapidly to a point where I thought further development was unnecessary

- Strongly agree (1)
- Somewhat agree (2)
- Neither agree nor disagree (3)
- Somewhat disagree (4)
- Strongly disagree (5)

Q25 The creative contribution of the bot was helpful in the interaction

- Strongly agree (1)
- Somewhat agree (2)
- Neither agree nor disagree (3)
- Somewhat disagree (4)
- Strongly disagree (5)

Q26 There was a strong sense of collaboration with the bot

- Strongly agree (1)
- Somewhat agree (2)
- Neither agree nor disagree (3)
- Somewhat disagree (4)
- Strongly disagree (5)

Q27 Based on this interaction, I would use a co-creative interaction such as this to develop musical material in the future

- Yes (1)
- Maybe (2)
- No (3)

Q28 I enjoyed the interaction with the application

- Strongly disagree (1)
- Somewhat disagree (2)
- Neither agree nor disagree (3)
- Somewhat agree (4)
- Strongly agree (5)

End of Block: Interaction

Start of Block: Block 2

Q29 Optional: please write any further comments
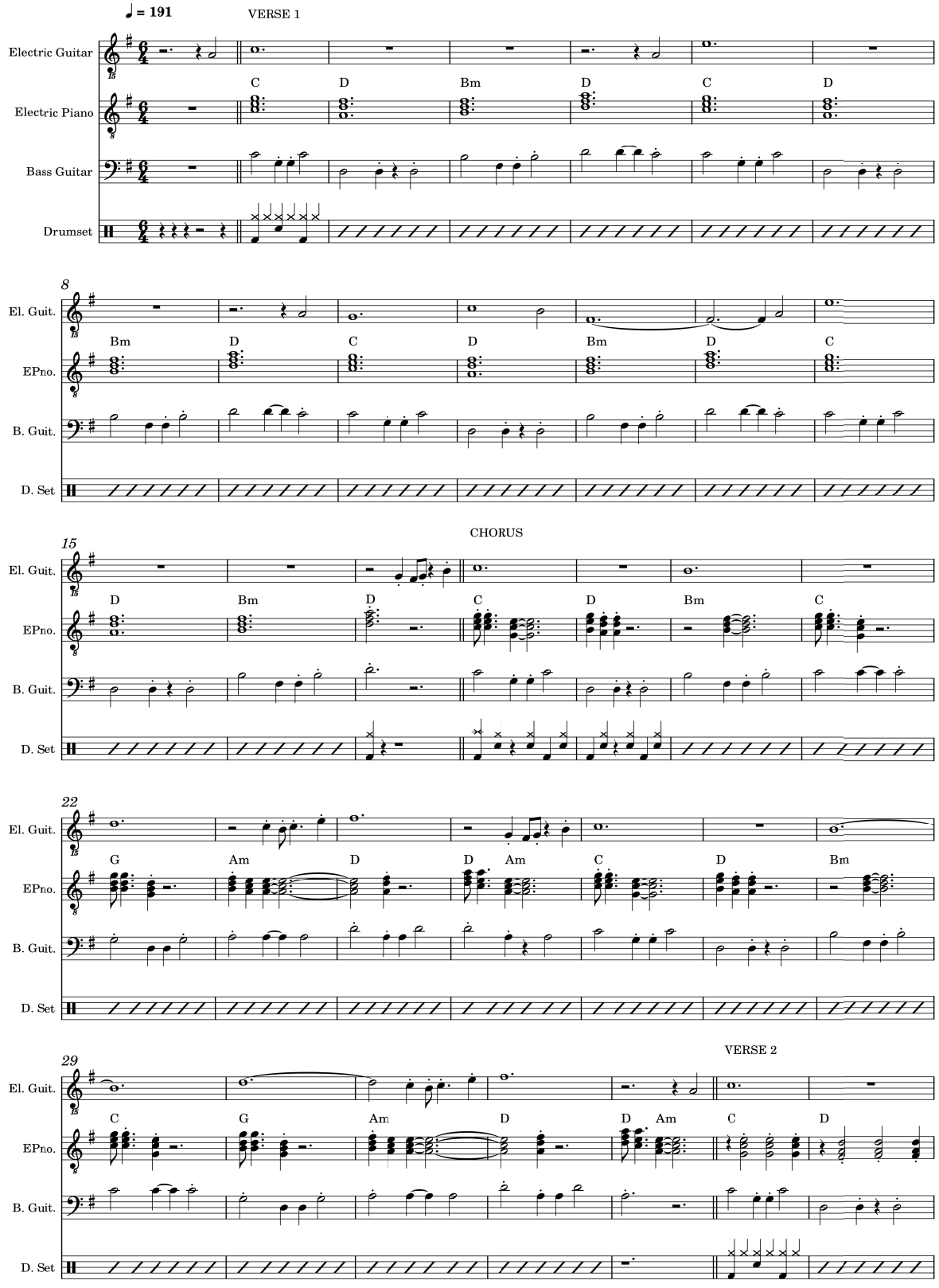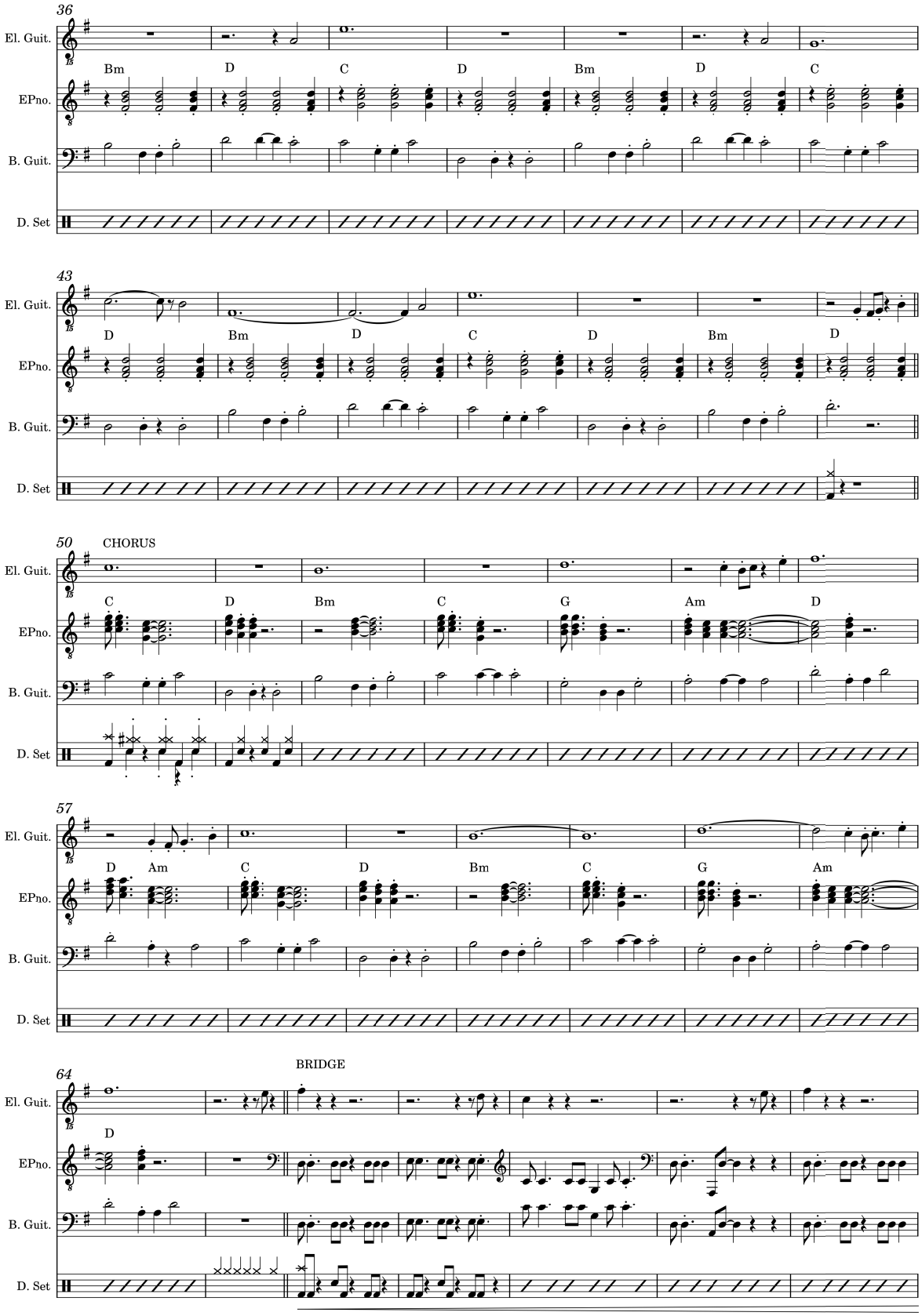
_____

_____

_____

_____

_____

End of Block: Block 2
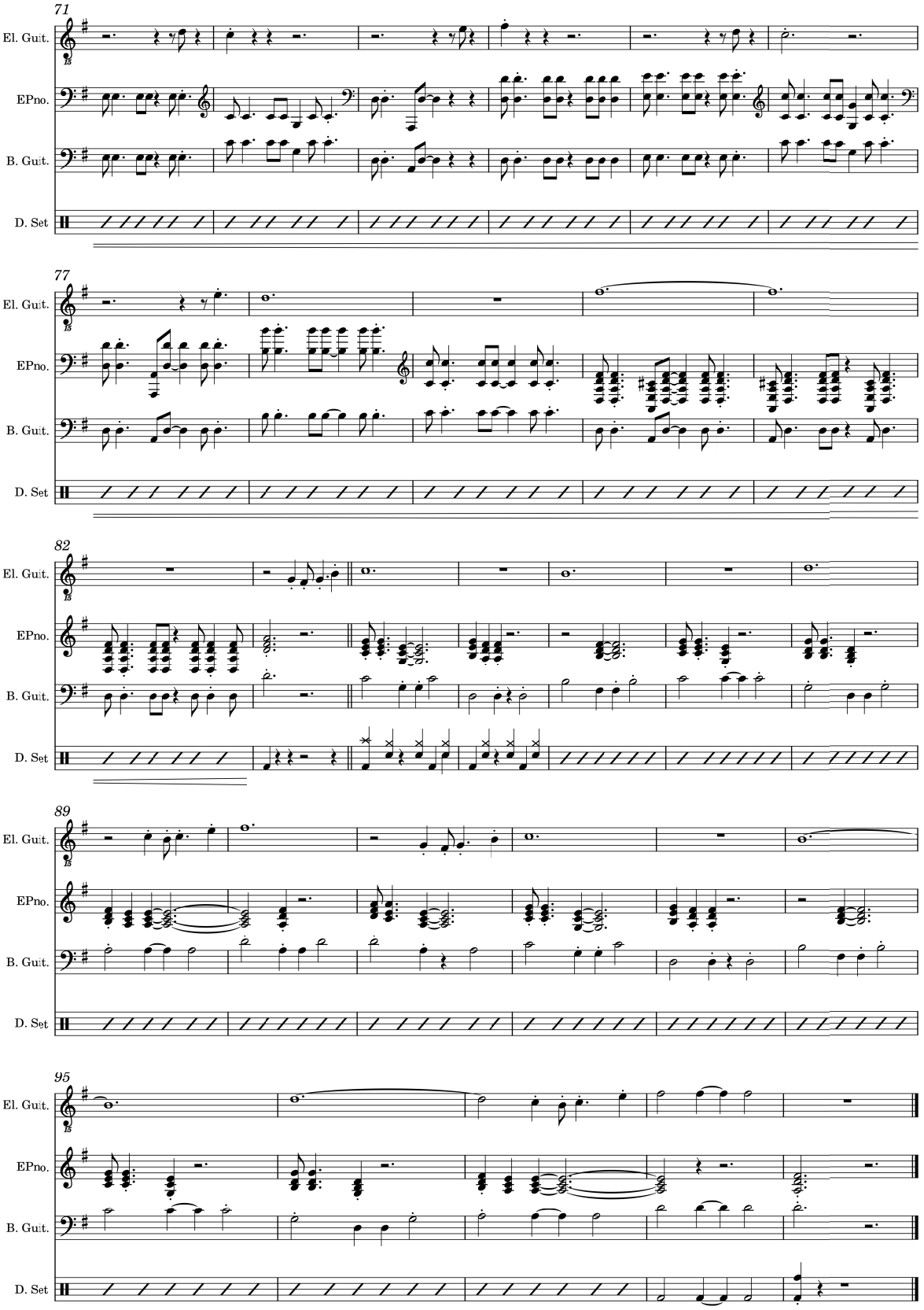
End of Block: Block 2

Appendix C: Score of the first composing experiment
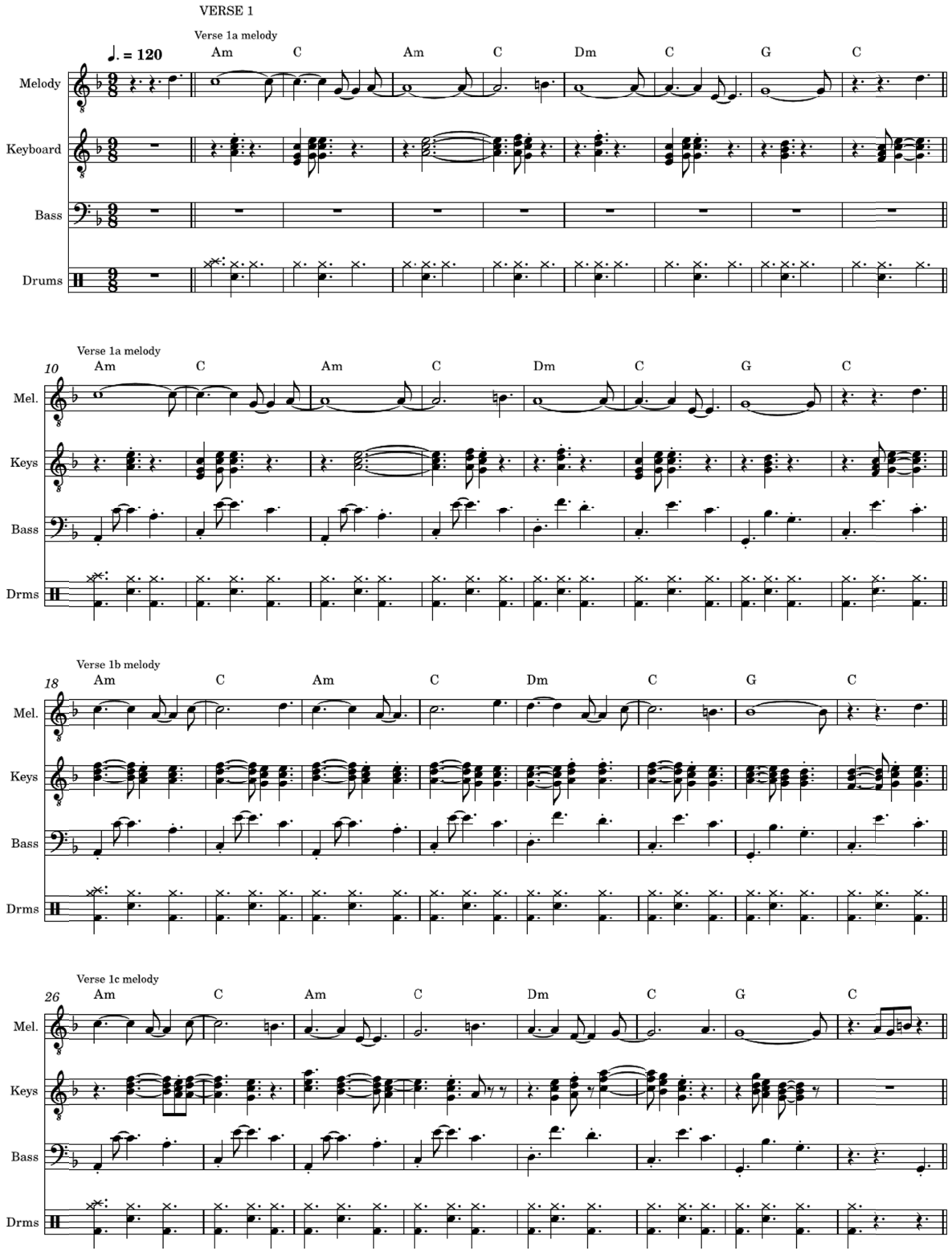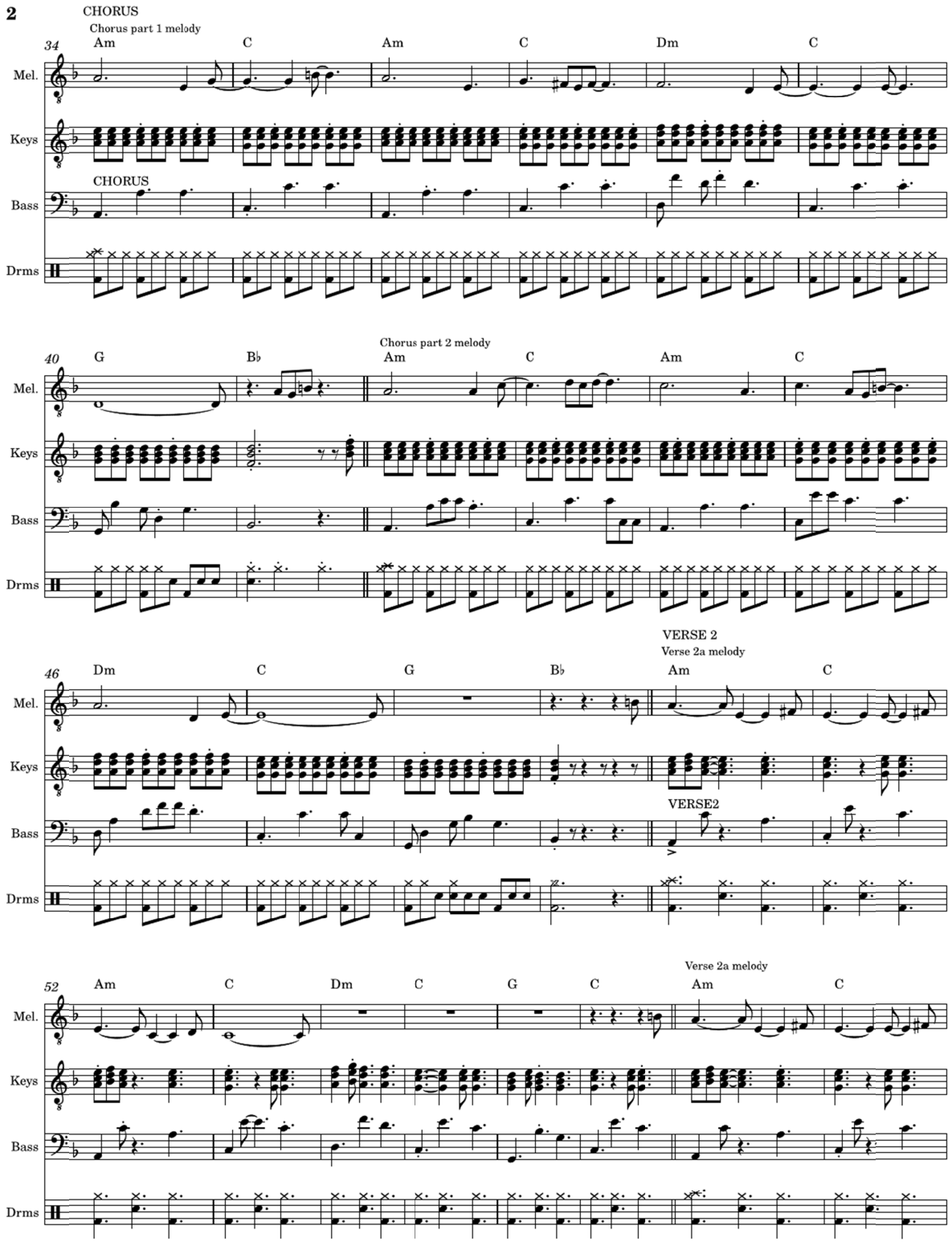


2
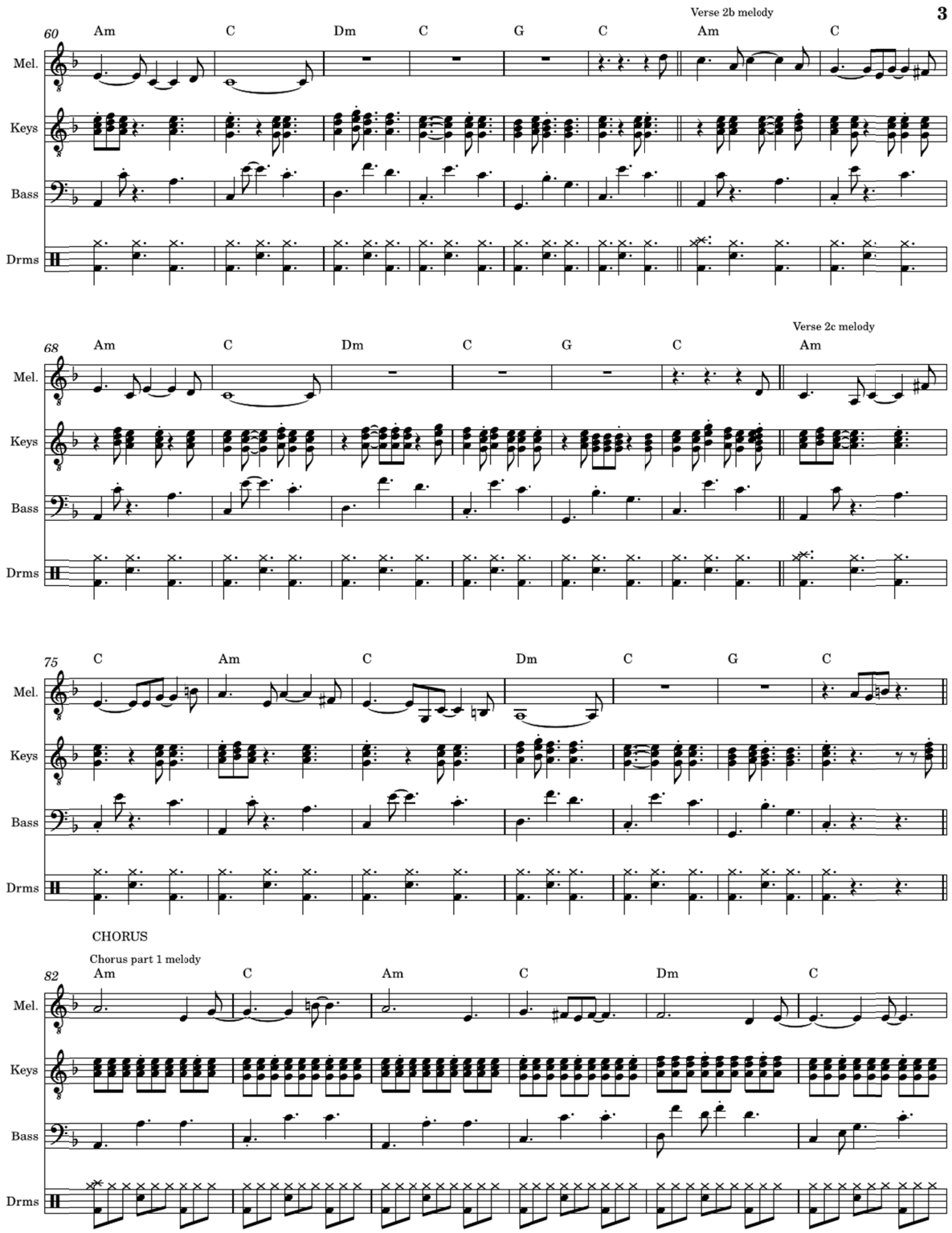

36
El. Guit.
EPno.
B. Guit.
D. Set
Bm D C D Bm D C
43
D Bm D C D Bm D
50
CHORUS
C D Bm C G Am D
57
D Am C D Bm C G Am
64
BRIDGE
D
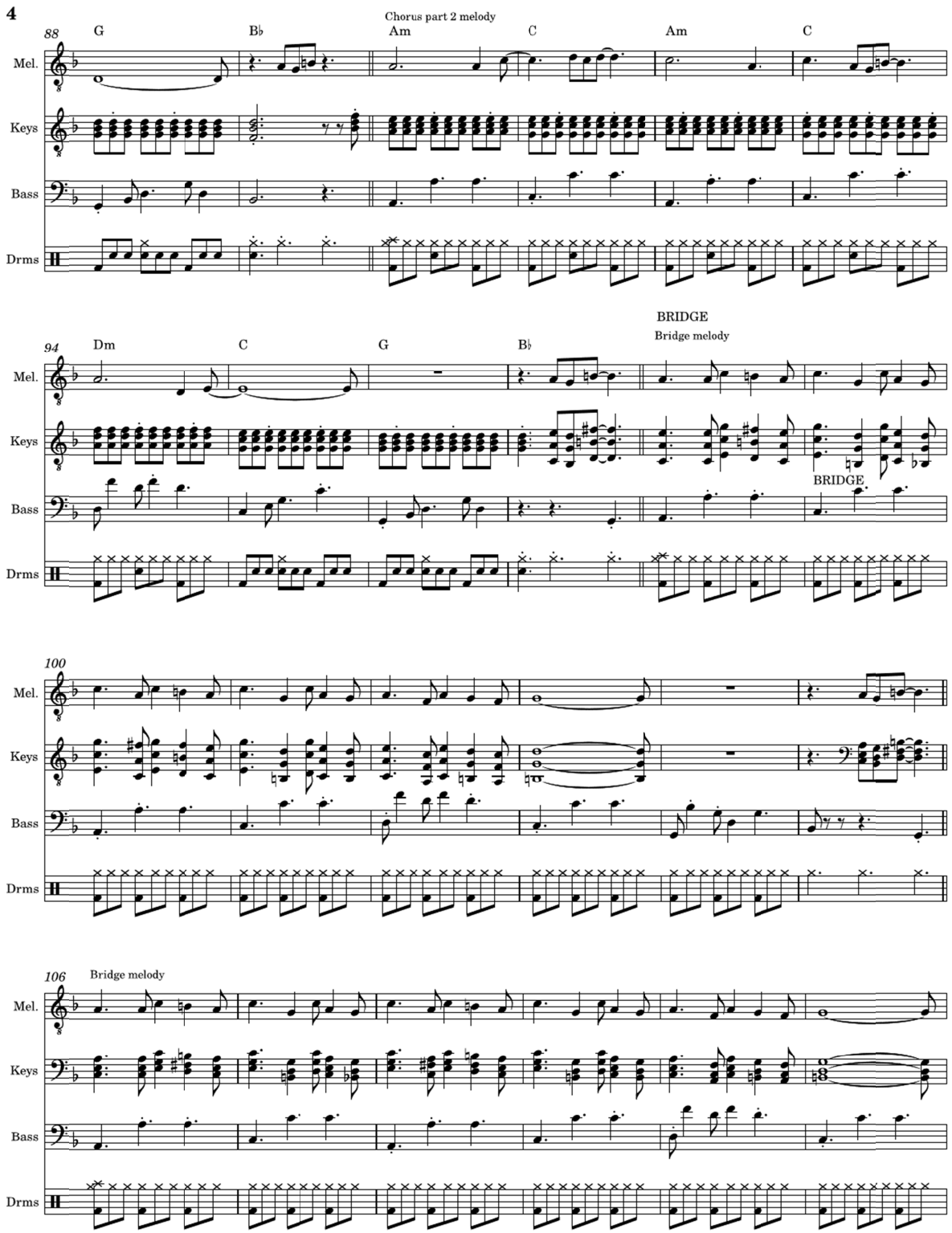

71
El. Guit.
EPno.
B. Guit.
D. Set
77
El. Guit.
EPno.
B. Guit.
D. Set
82
El. Guit.
EPno.
B. Guit.
D. Set
89
El. Guit.
EPno.
B. Guit.
D. Set
95
El. Guit.
EPno.
B. Guit.
D. Set
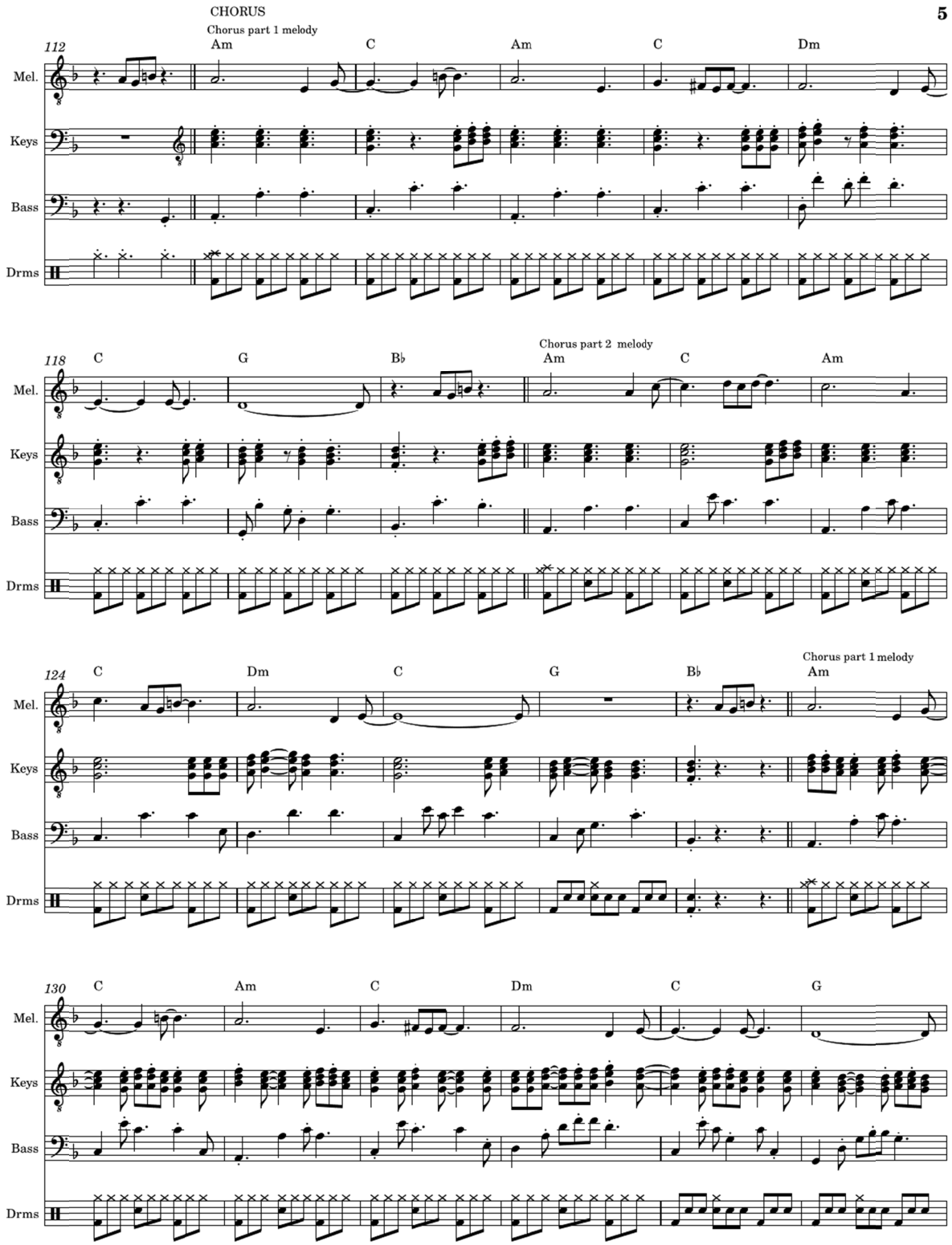
Appendix D: Score of the second composing experiment
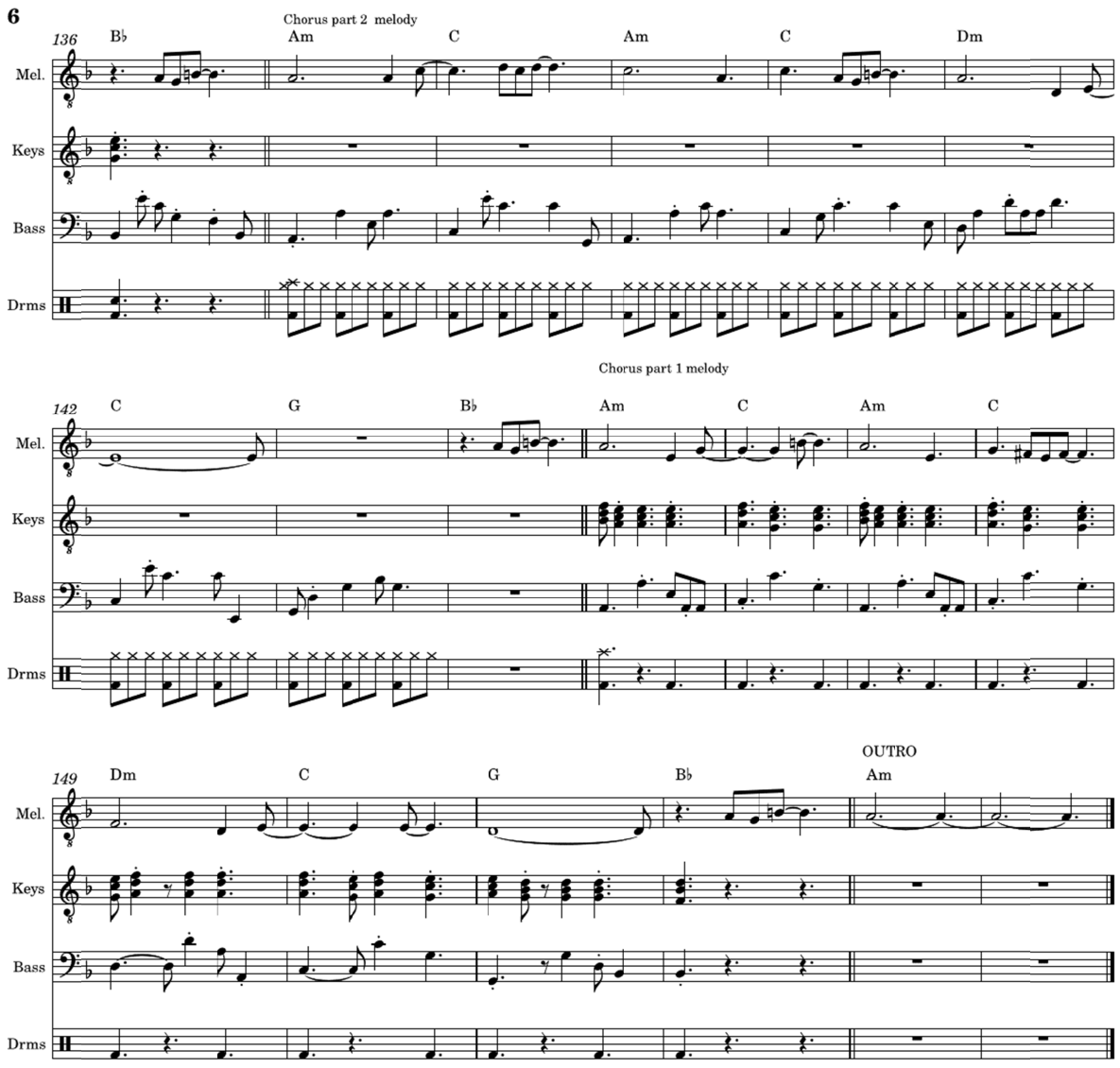



2
CHORUS
Chorus part 1 melody
34
Am
C
Am
C
Dm
C
Mel.
Keys
Bass
Drms
CHORUS
Chorus part 2 melody
40
G
B♭
Am
C
Am
C
VERSE 2
Verse 2a melody
46
Dm
C
G
B♭
Am
C
VERSE2
Verse 2a melody
52
Am
C
Dm
C
G
C
Am
C


Verse 2b melody
60
Am C Dm C G C Am C
Mel.
Keys
Bass
Drms
Verse 2c melody
68
Am C Dm C G C Am
75
C Am C Dm C G C
CHORUS
Chorus part 1 melody
82
Am C Am C Dm C


4
88
G
B♭
Chorus part 2 melody
Am
C
Am
C
Mel.
Keys
Bass
Drms
94
Dm
C
G
B♭
BRIDGE
Bridge melody
BRIDGE
100
106
Bridge melody


CHORUS
5
Chorus part 1 melody
112
Am
C
Am
C
Dm
Mel.
Keys
Bass
Drms
118
C
G
B♭
Chorus part 2 melody
Am
C
Am
124
C
Dm
C
G
B♭
Chorus part 1 melody
Am
130
C
Am
C
Dm
C
G


6
136
B♭
Am
C
Am
C
Dm
Chorus part 2 melody
Mel.
Keys
Bass
Drms
142
C
G
B♭
Am
C
Am
C
Chorus part 1 melody
149
Dm
C
G
B♭
Am
OUTRO