Developing Methods for Predicting Affect in Algorithmic Composition

Daniel Pitman

B.Mus. (Hons) 2012

Submitted in partial fulfillment of the requirements for the degree of

Master of Philosophy

Elder Conservatorium of Music
Faculty of Arts

University of Adelaide

June 2015

Contents

Abstract

Affective Algorithmic Composition (AAC) is a field that focuses on the algorithmic generation of music specifically to affect its audience in a targeted way.

This thesis presents a novel method for developing AAC systems based on collecting both perceived and induced affect data from human participants using multiple biosensor and surevey approaches, and modelling the resulting data in a predictive function based on a neural network. This in turn is used to drive the musical algorithm to generate music that can invoke any specified affective target.

These various approaches to affect measurement can be assessed and compared by their respective predictive error when used to train a neural network, providing an assessment tool for further refinement and development.

A pilot study of this method is also presented, The Affective Algorithmic Composer (AACr). AACr's predictive functions are trained using multiple forms of affect data collected from a group of participants, and can generate original music to invoke specific emotional states, physiological states, perceived content, and themes. Several generated compositions are included to demonstrate the abilities of the AACr to invoke affective states defined manually or directly taken from the user via biosensors.

The thesis concludes by reflecting on the method's strengths, areas for further development, and methods that could be used to determine the success of future AAC systems.

Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I give consent to this copy of my thesis when deposited in the University Library, being made available for loan and photocopying, subject to the provisions of the Copyright Act 1968.

The author acknowledges that copyright of published works contained within this thesis resides with the copyright holder(s) of those works.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

Daniel Pitman

2015

Acknowledgements

My supervisors Dr. Luke Harrald and Mr. Stephen Whittington, for their immense efforts and patience

Assc.Prof. Kimi Coaldrake whose guidance was so often critical

The Electronic Music Unit and Elder Conservatorium staff who offered assistance, critique, or patience

The Pain and Anaesthesia Research Clinic, Royal Adelaide Hospital, for access to and training with their EEG facilities

Friends and Family: Barry and Therese, Candice, Robert, Iran, Dan, and Meredith, for enduring support and/or proofreading

Finally I must express an immense appreciation to the volunteers (who must remain anonymous) who patiently auditioned and auditioned musical samples with uncanny resolve and no reward, yet were more critical to the project's success than anyone. -This project is literally made of you, and I can't appreciate you all enough for trusting me with your 'is-ness' in this whole process.

List of Acronyms

AAC: Affective Algorithmic Composition. A field of study where algorithmic composition is specifically designed to invoke a target affective state in the listener.

AACr: Affective Algorithmic Composer. The software and hardware developed in this study.

BCI: Brain Control Interface: A field of study, separate to AAC, where brain sensors are used to control computer systems.

BCMI: Brain Control Musical Interface. A field of study, separate to AAC, where brain sensors are used to control computer systems for music.

BIO: Biosensor (not including EEG). A descriptive abbreviation used in the AACr GUI.

BR: Bayesian Regularisation. A learning function used in neural networking

CAT: Computed Axial Tomography scanning. A brain scanning technology used in hospitals and laboratories involving X-rays.

DIY: Do It Yourself. A common term for enthusiast projects built at home or as a hobby.

ECG: Electrocardiogram. (sometimes EKG from Latin, ‘kardia’) A sensor that reports heart rate by measuring the field generated from the electrical impulses of the heart muscle.

EEG: Electroencephalogram. A sensor (or collection of sensors) that measures electric fields created by the brain’s neurons firing. Also used in the AACr GUI as a descriptive abbreviation.

EMG: Electromyography. A technique for measuring the electrical field changes created by muscles and their associated nerves.

EOG: Electrooculogram. A device that measures electrical field changes created by movement of the eyeball.

ERP: Event Related Potential. In EEG, a specific and often expected change in signals in reaction to a stimulus.

EVM: Eulerian Video Magnification. A process where the most subtle temporal changes in a digital video file are magnified to become visible, such as the change in complexion due to heart beat, or a vibrating guitar string.

fMRI: functional Magnetic Resonance Imaging. A brain scanning technology that is very common, but requires relatively large equipment and a magnetically isolated room.

GP: Genetic Programming. Specifically in this paper in regards to GP as a method of implementing symbolic regression, where solutions to a curve are formed using random symbols as pieces, and improved using a fitness routine.

GSR: Galvanic Skin Response. Another name for skin conductance.

GUI: Graphical User Interface. That part of a program that presents controls and information to the user via the screen.

HRV: Heart Rate Variance. The amount of variation from the average period of heart beats from a given sample.

IBI: Inter-Beat Intervals. The period in between each heartbeat.

LED: Light Emitting Diode. Simple, polarised, light emitting electrical component.

LIK: Likert. A descriptive abbreviation used in the AACr GUI.

LM: Levenberg-Marquardt. A learning function used in neural networking.

MEG: Magnetoencephalogram. A brain scanning technology measuring magnetic field changes, also requiring a magnetically isolated room.

MIDI: Musical Instrument Digital Interface. A simple and very common language/protocol for controlling instruments and synthesizers via digital commands.

MSE: Mean Squared Error. The average squares of the errors between data and a function trying to fit that data.

NN: Neural Network. An umbrella term for machine learning systems that emulate neural processing as found in organic brain networks.

NNS: Neural Network Server. A module of the Affective Algorithmic Composer system, described herein, that uses neural networks to calculates potential musical candidates for a given affective targets.

OSC: Open Sound Control. A sound control interface protocol that uses TCP-IP addresses and can communicate via ethernet.

R: (or R value) The correlation coefficient of two variables. If R is close to zero, the two variables in question are not related. As R approaches 1, the more the variables are related.

SCG: Scaled Conjugate Grading. A learning function used in neural networking.

SCL: Skin Conductance Level. The mean value of conductance between two electrodes on the skin of a set period of time.

SCR: Skin Conductance Response. A time measurement of the period between the beginning of a skin conductance event and the point it reaches half way back to the original level.

SQL: SQLite. An open and simple database language, natively implemented in Cycling’74’s *Max* software development environment.

Glossary of Terms

Affect: To have influence on, or potential to cause change.

Affective Target: A defined change in affective state that is required. In the AACr this involves a shopping cart type system, offering specific changes the user might like to attempt to invoke in the listener.

Algorithmic Composition: A method of composition, typically experimental, that involves defining an audible rendering of a mathematical formula or procedure rather then defining the notes themselves.

Arduino: One of many brands of small microprocessor\circuit boards that enables an electronic circuit to interface with a computer via USB or wirelessly. Other brands mentioned include *Teensy* and *OpenEEG*.

Array: A computer programming term for a list of numbers, similar to a matrix or multi-dimensional grid.

Biomusicology: A field of study that focuses on biological musical phenomena, including evolution, physiology, and empirical studies.

Biosensor Battery: Often a collection of different bio-sensors are collectively referred to as a battery, much like artillery.

Boolean: A method of mathematical analysis commonly used in programming that returns either ‘true’ or ‘false’. For example, if a = 1, and b = 2, then “a < b” is true and “a > b” is false.

Computational Critic: A part of a computer program that is responsible for assessing potential output (in this case musical passages) against the context of a defined target, usually using machine learning or some form of regression.

Induced Affect: That affect which causes changes in the listener both physiologically and psychologically. For example, a song that triggers the emotion of sadness in a listener whether the listener thinks the song is about being sad or not.

External Factors/Confounding Factors: May refer to anything which causes measured change in the participant other than the musical stimulus itself (distraction), influences that change (such as alcohol), or which causes the participant to react in ways beyond their personality/heritage (traumatic bias towards piano, illness).

Heuristics: A term for methods by which a problem can be approached and solved, typically used in computer programming and machine learning. For example, different search engines use different heuristics and return different results to the same search line.

Hierarchical Analysis: A traditional approach for analysing music using a range of levels or tiers of structure. Each tier influences the tier below it from the highest, overall form, right down to the lowest, individual motifs and notes.

Likert: A method of survey that employs answers using scales.

Music Affect: Both a contemporary and historical field of study, involving the influence of music on the human being, as well as a term for the phenomenon itself. For practical differentiation, in this study, the phenomenon studied is referred to as musical affect.

Music Algorithm: A system (typically a computer program) specifically designed to procedurally generate music.

Neutral Period: A period of time where no musical stimulus is presented at all.

Perceived Affect: Affect which is reported by the listener in terms of being expressed by the music. For example, “A song about being sad” does not necessarily trigger sadness in the listener.

Phasic: An event that occurs in phases, involves a series of smaller events, or is temporal in nature. A phasic change might involve the change of the temporal, periodical, or dynamic nature of the phenomena in question.

Predictive Function: Any mathematical or programming action that analyses a data set and extrapolates possible solutions to new enquiries, typically employing regression and machine learning techniques.

Salient: Noticeable or important details

Seed: See Structural Array

Stochastic Data: Data using samples, which inherently contains errors or noisy variation to some degree.

Structural Array: Also called a ‘seed’. Both terms are specific to this project. A structural array is the term used to refer to a list of numbers that represent all of the hierarchical variables needed to have the music algorithm generate a piece of music.

Synaesthesia: A physiological condition where various receptive systems (sight, hearing, smell etc.) are confused or influence other receptive systems in the brain. Commonly symptoms include associating colour with certain aspects music or timbre, or having texture strongly associated with certain smells.

Tag-word: A single word associated with a data entry, effectively forming a category so that all entries with a specific tag-word can be recalled using that tag-word.

List of Figures & Tables

1 Introduction

Generative algorithmic music, the basis of so much modern ground breaking work and experimentation in musical composition, is, at its core, audible representation of formula, or sonic graphing. The complexity of generative algorithmic composition has increased exponentially with advancements in computing technology, and large amounts of quite sophisticated musical material can be almost-instantly generated, leaving the composer with a task of deciding what or how each result may be valuable or useful.

Generative algorithmic composition has a different approach to traditionally composed music; this is especially true for expression, communication, and context. Where traditional composers have engaged human emotional concepts through a long established and delicately applied set of compositional protocols, harmonic movement, counterpoint, form, algorithmic approaches are much less able to accommodate human expressionistic protocols. Many composers have embraced this opportunity to explore the unfamiliar, but for certain purposes where emotional affect or reinforcement is paramount, such as in soundtracks for film or gaming, generative algorithmic music has found limited application, usually involving severe limitations on an algorithm's possible outcomes or laborious supervision by a human agent. It is within this realm of affective targeted music, -akin to film/media/gaming composition- that this generative algorithmic music project is focused.

Algorithmic music essentially strives for automated composition; "…the process of using some formal process to make music with minimal human intervention" (Alpern, 1995. p. 13). Affective Algorithmic Composition (AAC) aims not only to relieve the human agent of this inherent dependence, but also to create a system that can invoke affect musically, in ways that can be understood by a human audience.

It is worth discussing Williams' theme for the movie *Jaws*, (1975): two repeating notes, one semitone apart, with a long pause in between. The pause becomes progressively shorter dissonant embelishments increase as the protagonist approaches its prey until the rhythm is

relentless; a simple and functional composition specifically designed for reflecting and reinforcing the tension present in the film. This functional reinforcement approach to composition is reflected in the goals of AAC systems that "… are distinct from traditional algorithmic composition systems that do not consider an intended affective trajectory in the generated material: in AAC, the algorithm is always informed by an intended affective response" (Williams et al., 2014, p. 2). AAC is not concerned with experimental aesthetics; rather, any musical outcomes that can affect an audience.

AAC systems typically use simple categorical emotion models or machine learning functions to predict the affective outcome of unheard passages of music. This gives rise to a musical algorithm that can search for a piece of music to suit a certain emotional or affective target.

In this thesis, a method for affective algorithmic composition is proposed and demonstrated, based on training neural networks to recognise the affective qualities of musical structures, from the reactions of a group of volunteers. The method is focussed on reducing predictive error, and demonstrates the use of predictive error to assess and compare various methods of biofeedback measuring techniques, survey development, and implementation of the AAC for the user.

This thesis is structured into an introduction, two main sections, and a conclusion. The introduction includes a background information section which is an overview of the slightly perilous topic of music affect as a field of study, its relationship to generative algorithmic composition and the two concepts coming together in the field of affective algorithmic composition. This background information helps to outline the current state of research, and sets the scene for the developments in the following two main sections.

The first main section, *Developing Methods for AAC*, is a discussion of approaches and the related literature used in developing each aspect of this method. Topics include discussion of development of appropriate musical algorithms, equipment and software, collection and

analysis of affect data from volunteers, and the development and assessment of predictive functions using this data.

The second main section is a detailed outline of the *Affective Algorithmic Composer* system (AACr); a pilot study and functioning AAC system developed to demonstrate the method proposed. This section contains detailed descriptions and discussion for each component of AACr system, including the music algorithm, the sensors and auditioning process, the creation and assessment of the neural network system, and discussion of the musical outcomes.

In the conclusion, relative successes and future directions for research are discussed.

Several generated pieces are included as demonstrations of the AACr's output. Recordings of unsupervised real-time music are presented, generated from both manual and biosensor defined affective target states. Among these examples, three relatively intricate pieces are generated, with some supervision, for three famous scenes from Fritz Lang's 1927 silent film *Metropolis.* These pieces demonstrate the AACr's abilities for real-time media applications and a non-real-time compositional aide role.

1.1 Background Information

The earliest computerised musical algorithms were considered as aids for composers. Iannis Xenakis, used a computer to 'deduce' a score from a list of note densities and weights in *Atrées* (Xenakis, 1968). However, Xenakis was thoroughly involved in developing the final result. Cope critiques *Atrées*, "…the computer has not actually produced the resultant sound [or notation]; it has only aided the composer by virtue of its high speed computations."(Cope, 1976, p. 259).

Hiller and Isaacson were responsible for what is commonly regarded as the first unsupervised algorithmic composition, *The Illiac Suite*, and also demonstrated that the algorithm responsible could be given new variables to generate new material (Hiller & Isaacson, 1979; Roads, 1996).

Algorithmic Composition

Musical theory is inherently quantified and easily encoded, which on its own is generally enough shape for an algorithm to generate something that at least sounds like music. However, the re-useable nature of algorithms like *The Illiac Suite* has given to rise to a complex medium where composers capture an entire process for composing, rather than just rules for a song in itself. Thus 'composition theory' is a more relevant focus for algorithmic music than 'music theory' (Laske, 1989, p.46). This is a significant distinction, suggesting algorithm and music have perhaps never been exclusive of each other.

The Illiac Suite chose the most suitable result via a mathematical set of rules acting as a filter. These mathematical rules were previously defined, and thus these choices were informed by the author. Automating this inference is a core focus for AAC developers.

David Cope, speaking about the earliest development of *EMI (Experiments in Musical Intelligence)* eludes to his own human inference, speaking of his earliest attempts at algorithmic musical part writing,

> While some of the music composed using this approach did prove fairly successful, most of its output was equally uninteresting and unsatisfying. Having an intermediary –myself– form abstract sets of rules for composition seems artificial and unnecessarily premeditative (Cope, n.d.).

Cope's solution was to automate the hierarchical analysis of other composer's works, taking the underlying structures and shapes to inform the algorithm in the creation of new pieces (Cope, 2005). In effect, *EMI* still relies on human inference, just not from the author. By imitating the structures that other composers most often deemed to be valuable and emulating them, *EMI* produced 'original' music that featured uncanny similarities to the contributing composers.

It is worth taking a moment here to consider that Cope's *EMI* stands out as one of the most famous musical algorithms of all time, and one of the most controversial for his misuse of the

word ‘creativity’, which Wiggins criticises Cope for in his review of *Computer Models of Musical Creativity* (Wiggins, 2008).[1]

Regardless, to develop an algorithmic system that can filter material that is affectively valuable from the uninteresting still means bridging a ‘gap’ between symbolic mathematics and the communicational conventions of an emotional human audience.

The Gap between an Algorithm and its Audience

Composition is a human phenomenon of communication (Fedorenko, McDermott, Norman-Haignere, & Kanwisher, 2012), and as such it often contains levels, structures, hierarchies, and syntaxes, not dissimilar to written language (Cogan, 1984, p. 111; Levitin, 2007, p. 154; Pinker, 1999, p. 532). Composed music typically elicits meaning or metaphor, and relies on certain points of mutual context to relay this metaphor (London, 2007; Mannes, 2011; Petsche, Linder, Rappelsberger, & Gruber, 1988; Scruton, 2009; Thompson & Biddle, 2013; Zbikowski, 2008). Throughout a lifetime of exposure a human composer naturally develops a semantic framework of generalisations about how an audience will be affected by certain musical sounds, informing their decisions on how to fill the physical framework of their compositions. It could be argued that a composer uses a form of inductive reasoning, taking previous musical interactions as samples, and developing generalisations from them for use in predicting the effect of each musical mechanism as they compose (“Deductive and Inductive Arguments,”). It is conceptually difficult to relate such abstract human compositional reasoning to an algorithmic compositions system.

This highlights a ‘gap’ between mathematical music and a human audience’s communicative standard. This gap may be one reason why generative algorithmic

1 While misusing words like ‘creativity’ in the field of artificial intelligence is at least highly problematic, Wiggins is ruthless, accusing Cope of crimes in “pseudo-science”. It seems unnecessary on Wiggins’ part, or at least forgivable on Cope’s, but the reader should take from this a warning that terminology is especially delicate when discussing artificial intelligence and AAC. It is important to establish as early as possible that affective algorithmic composition is a method of filtration. Potential results are filtered from the finite but large number of possibilities an algorithm can produce, according to their affective potential, not created.

composition has not featured commonly in emotionally driven film/media/gaming musical application, as algorithmic music can difficult for an audience to contextualise emotionally.

Not surprisingly it was once a common assumption that musical composition must be an exclusively human function that cannot be synthesised (Cope, 2004). More recently musical neuroscience, bio-musicology, and computational neuroscience have been measuring and reverse-engineering musical affect in several different ways that make programming a function to resemble the composer's the inner critic a much more feasible proposition. However, music affect as a field can be difficult to navigate.

> …the structure of affect theory mirrors the ambiguity, open-endedness, and messiness of that which we might call affect. (Thompson & Biddle, 2013, p. 6)

The definition for 'music affect', both as a field and as a phenomenon, varies greatly from author to author. For this project it is very important to recognise that this 'stubbornly unrecognisable' term refers to a 'myriad of approaches: sometimes subtly differentiated, sometimes markedly conflicting in their differences' (Thompson & Biddle, 2013, p. 6). As soon as anyone uses the word 'affect' in a musical study, alarm bells ring, and they must commit to investigate the field and its definition for fear of being lost in the very ambiguity it suffers from.

Affect Origins

The Ancient Greeks wrote extensively on controlling human emotions with music, both in modes and with a system known as the four temperaments. But perhaps a more direct starting point for the study of musical affect is during the Renaissance, as German musicologists used the concept of *affect* (in German: *Affektenlehre*) to describe aesthetic concepts in Baroque music. Their idea of 'Affekt' stemmed from Latin doctrines of rhetoric and oratory, which at that point were largely theoretical and were being used for discussing music composition. In these Baroque music treatises the composer was to move the 'affects' of the listener much

like an orator moves his audience with rhetoric (Buelow, 2001). ‘The Affects’ in this case referred to the emotions or passions of the listener.

The *Doctrine of Affections*, also known as the *Theory of Affections*, became a common topic in the mid-17th century. Many theorists began to devote large parts of their treatises to categorising and describing types of affect as well as describing the ‘affective connotations’ of musical structures, instruments, scales, and rhythms (Buelow, 2001).

The result was a library of instruction on eloquence, proper musical grammar, and systems of emotion. Affect took on a sophisticated role as the core reason for composing, even a spiritual role, and was at times considered a universal. In 1597 Lorenzo Giacomini defined the affections as a “spiritual movement or operation of the mind in which it is attracted or repelled by an object it has come to know [as] a result of an imbalance in the animal spirits and vapours that flow continually throughout the body” (Giacomini-Tebalducci-Malespini, 1597).

Music is clearly something that has an effect, and many sought to further understand and harness this phenomenon. Hoyt describes Descartes’ (1649; Meyer, & Timmermans, 1990) *Les Passions de lâme (The passions of the Soul)* as one of the most decisive influences on the art of music. Descartes believed that there was a “rational, scientific explanation for the physiological nature of the passions” and that he had discovered the objective nature of emotion (Wilson, Buelow, & Hoyt). Descartes fostered a dualistic approach to bodily substances; extended matter on one side, and the immaterial mind on the other. While he stated that that mental experience could not emerge from pure mechanical dynamics, he treated the body as merely an organic machine.

In *L'homme, et la formation du foetus (Man and the formation of the foetus)*(Descartes et al., 1677), he writes of the immaterial mind controlling the otherwise automated body, but that the passions of the body (not of the mind) can influence the mind in return if left unchecked:

> Thus, I say, when you reflect on how these functions follow completely naturally in this machine solely from the disposition of the organs, no more nor less than those of a clock or other automaton from its counterweights and wheels, then it is not necessary to conceive on this account any other vegetative soul, nor sensitive one, nor any other principle of motion and life, than its blood and animal spirits, agitated by the heat of the continually burning fire in the heart, and which is of the same nature as those fires found in inanimate bodies. (Descartes et al., 1677, p. 8)

These early attempts to find understanding of human being's unusual engagement with these periodical and harmonious sounds remained a core aspect for Western musical theorists and critics, even though the word 'affect' fell out of favour through the 18th century. 'Rhetoric' remained a fundamental term in theoretical studies, to the point where many considered the, "…musical surface as saturated with rhetorical symbols." (Wilson et al.).

Attitudes to the concept of rhetoric had flipped by the beginning of the twentieth century. Hoyt summarises that investigations of the time returned very little evidence that the classical composer's had actively sought training in rhetoric, or actively employed the concepts in their compositions. He instead suggests

> …that rhetoric did not provide models for composers; rather, writers on music seem to have adapted rhetorical concepts to conform –however tenuously– to musical practice. (Wilson et al.).

Conflicting Approaches to Affect

The human of the twenty-first century is still reacting emotionally and physiologically to music (Coutinho & Cangelosi, 2011; P. N. Juslin & Västfjäll, 2008). Music as a biological phenomenon is witnessed and studied throughout the world's cultures. Physiological and psychological reactions to music now have several fields of study, approached by both the musician and the scientist (Wallin, 1991).

The exploitation of 'the passions' can still be found in commercial music, creating simple, reliably affective, emotionally driven music. The American company, Muzak, renowned for their ambient elevator music cliché, was a massive entity in the music industry, exclusively selling music that purposefully promoted brain states that were more susceptible to commercial marketing and advertising techniques (Brown & Volgsten, 2006, p. 110).

Despite its modern use in the fields of pharmacology, neuroscience, psychology, and physiology, the term ‘affect’ in music studies has remained for some an awkward or even pseudo-scientific concept amongst the fields of musicology. The modern music author, concerned with the musical studies on a physiological or psychological level, is forced to consider that the term ‘affect’ has a certain history and air about it, something which renders its employment potentially undermining. It is perhaps better to address this blurred line, rather than skirt around it:

The field of music affect suffers from inbuilt vagaries, an inability to explain experience with any authority, and a constant re-definition and re-stating of terms that never concretize. “The literature presents a confusing picture with conflicting views on almost every topic in the field” (P. N. Juslin & Västfjäll, 2008, p. 559). Even the Webster Dictionary’s entry is contradictory,

> 2: the conscious subjective aspect of an emotion considered apart from bodily changes; also: a set of observable manifestations of a subjectively experienced emotion… (“Definition of Affect”)

The reader will often find that the word *affect* refers to both the music’s influence on the listener as well as the change in the listener’s state. Webster Dictionary also states that a thing which is *affective*, is “relating to, arising from, or influencing feelings or emotions” (“Definition of Affective”).

More specific to the field, affect is also considered as a potential that something (i.e. music) can harbour, thus a passage of music may also contain affective potential which is yet to be heard and realised. This potential has at times been poetically liberated of a grounded definition.

A work, which despite being titled *Affect and Embodied Understanding in Musical Experience*, actually only presents a series of anecdotes of musical euphoria,. This was apparently evidence for the solidarity of the audience being a catalyst to the creation of a field of “affective and transformative energies” (DeChaine, 2002). As pleasant and descriptive as these anecdotes were, it certainly wasn’t empirical or a form of evidence or research. It serves

to highlight a kind of speculative approach that is at times unhelpful for the field of music affect and certainly offers nothing of value to an AAC project. We all know first-hand the power of music, but it might be better studied without poetic speculation, if only from an AAC point of view.

Julie Reiser (2012), in a review of what was meant to be a definitive reader on affect, *The Affect Theory Reader* (Gregg and Seigworth, 2010), summed up perfectly the disarray that the field has suffered.

> …I also approached the volume wanting to be able to use it in a class on affect theory; wanting to be charmed by its inclusion of more persuasive, interesting theoreticians than I had yet encountered; wanting to be converted by that one pivotal essay that could somehow explain to me why so many of the best minds of my generation have been seduced by this literary phlogiston; and wanting, albeit totally unfairly, to see it create a sense of order and coherence for what otherwise appears to be an untidy, excessive exploration of that *je ne sais quoi* of human experience that has seemingly evaded complete explication by the whole of theory and philosophy for close to two millennia. (Reiser, 2012)

Affect as an "ology" seems as elusive as the very experiences it hopes to study, and for fear of being lost in its own reputation for the internalistic and hopeless standing as an eccentric subfield of musicology, it might only be safely used when referring specifically to the historical concept, explicitly using the German 'k'; *affekt*. The needs of developing an AAC system are not satisfied by any conceptual model or internalised descriptions.

> It could appear that our claim that musical emotions must be investigated with regard to their underlying mechanisms is uncontroversial, and that all music researchers would agree. Yet, this is not how research has been conducted, which is ultimately what counts. Studies thus far have produced data that are collectively confusing and internally inconsistent, mainly because researchers have been considering only the induced emotions themselves, instead of trying to manipulate the underlying mechanisms in a systematic manner. (P. N. Juslin & Västfjäll, 2008, p. 574)

This fresh approach in the field of music affect from musicologists, musical psychologists, and neuroscientists, is empowered with the by-product of more modern methodologies, approaches, and standards. For encoding a system of affect within a composition algorithm,

the focus is clearly shifted towards these mechanisms; a more consistent source of evidence and, by nature, empirically definable.

Empirical Approaches: Perceived and Induced Affect

It is important here to distinguish between two categories of empirical affect: perceived and induced affect (Lundqvist, Carlsson, Hilmersson, & Juslin, 2009, p. 61). A song that sounds like it is (or is perceived as being) about the heart racing could be very different to a song that induces a change in heart rate. A song that is perceived as sad could induce similar physiological changes as a happy song. A survey that has a tick box next to "heart racing" will not necessarily be ticked at the same time as the listener's heart races. (P. N. Juslin & Sloboda, 2011, Chapter 11.3.2; Williams et al., 2014, p. 4).

There is a long history of empirical research into musical affect using biosensors and surveys. The following two cases studies, Hevner and Ellis, demonstrate perceived and induced affect literature, and are among early examples of a massive amount of empirical affect literature produced up to this day, critical for AAC development:

Perceived Affect

In 1937, Hevner published the last in a series of experiments measuring the expressiveness of music, where a group of participants selected tag words like "dignified, spiritual, poetic, sparkling" (Hevner, 1937, p. 622). The survey relayed information about perceived affect. It also led to some odd sounding conclusions like "High pitch shows its largest effects on the humorous-sparkling-playful tone and low pitch divides its effectiveness over sad, dignified, and vigorous-majestic groups." (Hevner, 1937, p. 625-626).

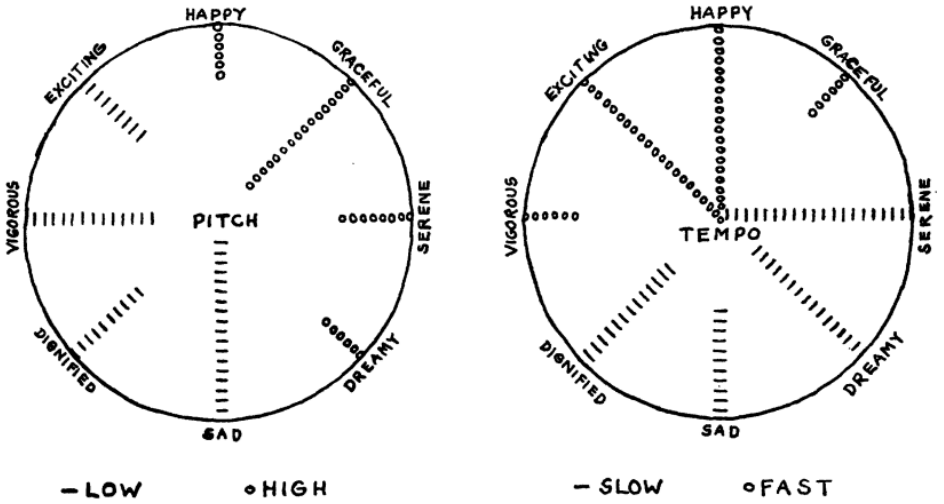


Figure 1 - Example of Hevner's Results (Hevner, 1937, p.624)

It's easy to suggest that this is a limited approach; the participants were likely inclined to interpret each of the survey words differently; the various nuances and endless combinations of musical structures in use are hardly accounted for, and this categorical structuring of emotions seems difficult to legitimise. Hevner does acknowledge these limitations:

> Against the too hasty conclusions of a casual reader who may expect that we have undertaken to reveal a formula by means of which music may be written to fulfil the requirements of any desired mood effect, we cannot emphasize too often the limitations which must be applied to our results. (Hevner, 1937, p. 627)

Hevner's work is still far more useful to AAC than that of Matthesson or DeChaine. It is impersonal and unbiased, and most importantly, an empirical approach. A quantified 'formula' for emotion like this could -and has been (Hoeberechts, Demopoulos, & Katchabaw, 2007)- implemented as a model that can outline the structural inputs to a musical algorithm to fulfil 'any desired mood effect'.

An AAC system based on this perceived affect data might tend towards higher pitched notes when instructed to express humorous/sparkling/playful concepts, or in the case of the chart from Figure 1, use higher tempos when instructed to generate exciting, happy, or graceful music. It may be a limited model as Hevner suggests, but prediction stems from

generalisation: where a group of specific examples exist, generalisation allows a prediction of varying degrees of strength regarding future examples ("Deductive and Inductive Arguments"; Giere, 1997; Herms, 2014, p. 2).

Induced Affect

In contrast, the literature of biomusicology embraces neural-physical, neural-psychological, comparative, and evolutionary musicological perspectives (Wallin, 1991). The relatively new term, coined in 1995 by Wallin, could be considered a completely separate field to affect, or perhaps an empirical manifestation of it. The line is blurred at points as both fields consider the functions and uses of music, musical influence on behaviour, and both physiological and psychological processes of music in the human body (Wallin, Merker, & Brown, 2000). However, biomusicology does have a pre-established terminology and a consistency that the field of affect does not often enjoy.

The 1952 publication, *Effects of Music on Respiration and Heart Rate* (Ellis & Brighouse, 1952), although not the first study to assess musically induced changes to physiological functions (Ellis & Brighouse 39), was among the first to do so with a well-defined procedure and a reasonable number of cases. Like Hevner's study, Ellis and Brighouse's study also provides data that could be modelled in an AAC system.

There are of course many external factors that could affect a subject's cardio and respiration rate, and Ellis takes definite measures, to the extremes of locking subjects in a cage with a mattress and interacting via speaker alone (Ellis & Brighouse 40), to ensure that external factors are insignificant. Ellis also outlines a stringent set of requirements that could be considered critical in a similar endeavour:

1. Individual rather than group treatment of Ss
2. Careful control and adequate specification of the conditions under which Ss are exposed to music
3. Use of several music selections
4. Experimental design allowing for adequate statistical treatment of results
5. Provisions for obtaining adequate measure before, during, and after music

(Ellis & Brighouse, 1952, p. 40)

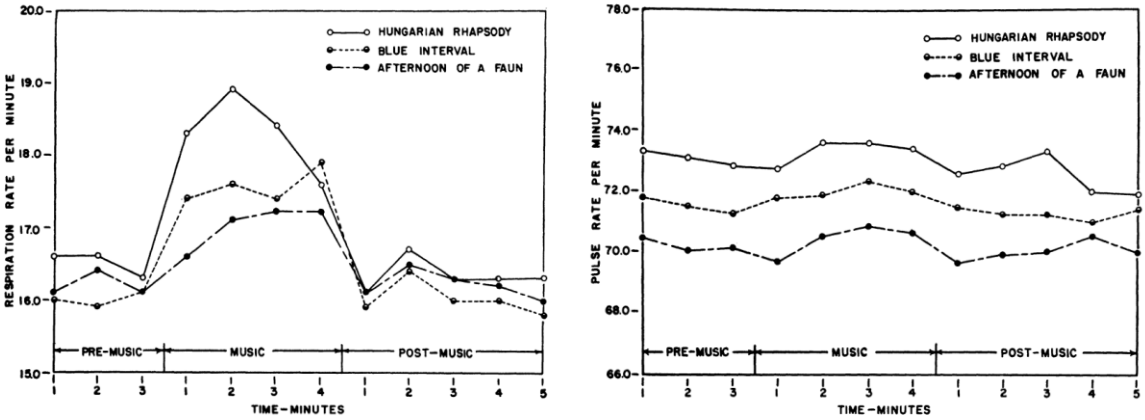


FIG. 1. MEAN RESPIRATION-RATE BEFORE, DURING, AND AFTER MUSIC (Every point based on the results of 36 Ss.)

FIG. 2. MEAN HEART-RATE BEFORE, DURING AND AFTER MUSIC (Every point based on the results of 36 Ss.)

Figure 2 - Cardio and Respiratory Reactions to Musical Stimuli (Ellis & Brighouse, 1952, p.42-43)

Figure 2 shows clear increases for the breath and heart rate change over the period of the musical stimuli, and a fairly consistent difference between each of the three songs used. This study is exemplary of many later studies that outline the potential for measuring induced musical affect via its cardiac and respiratory manifestations (Bernardi, Porta, & Sleight, 2006; Birnbaum, Boone, & Huschle, 2009; R. J. Ellis, Sollers III, Havelka, & Thayer, 2009).

Approaches to Affect in Algorithmic Music

At the time these studies were written, the terms ‘perceived’ and ‘induced’ affect had not yet become as prominent as they are now in fields studying musical affect. The audience ‘perceives’ affect consciously, but physiological changes are also ‘induced’, and how the two are related remains elusive (P. N. Juslin & Sloboda, 2011, Chapter 11.3.2; Williams et al., 2014, p. 4).

In the field of AAC, the distinction between perceived and induced affect has become a primary concern. In 2014, Williams reviewed thirty AAC systems and categorised them according to their abilities, structure, and feature sets (Williams et al., 2014, p. 10):

- Are they capable of compositional and performative music making?
- Do they use an entirely generative algorithm process?
- Are they useable in both unsupervised real-time and compositional aide applications?
- Are they also able to adapt to input directly from the biofeedback sensor array?
- Do they utilise induced affect?

It is a brief categorical structure for comparing AAC systems, and it may be useful for an aspiring AAC developer to assess their system by these criteria as a standard for comparison.[2]

Of those reviewed Jiang and Zhou's system, *Automated Composition System using GA (Genetic Algorithm)* is perhaps the most relatable for this project, as it uses a neural network to analyse potential outcomes as well as a pre-established model of emotion quantification and categorisation (Jiang and Zhou, 2010). Jiang and Zhou describe the "PAD" system; three polar dimensions, "pleasure-displeasure", "arousal-non-arousal", and "dominance-submissiveness". These emotional categories are mapped to musical variables that are more descriptive than a direct representation of the music data; "the density of notes", "the biggest interval", or "the stability of pitch" etc.

It's not well discussed if having a machine learning function as well as an emotional model benefits prediction. It is simply stated, "The experiment results show that our method can

2 Categories such as "composition or performance" are often only fickle matters of user interface design rather than fundamental limitations of the AAC method in question. The "supervised vs unsupervised" category makes categories like "composition versus performance" seem a little obsolete. Finally, it is curious to have an "induced affect" category only to put no entry into it, and then claim no such examples exist, more so when such examples do. Williams did not perhaps intend to devise a thorough or overarching standard, although he does inadvertently show that there is a requirement for such a thing in the field.

yield music which is pleasant to ordinary listeners" (Jiang and Zhou, 2010, p.380). This is a somewhat unsatisfying conclusion.

Little discussion of measured success in AAC studies seems uniform at this date.

> Indeed, affective evaluation in the surveyed AAC systems is sparse. There is a significant amount of further work in such evaluations. (Williams et al., 2014, p. 18)

Success in musical composition is difficult to define, but perhaps not so much for a researcher already wielding affect measuring equipment, and with the specific context that affect is ideally invoked accurately by the resulting compositions. In reality, this is more complex than just "seeing if it was right" (see Assessing the Predictive Functions p74).

Another interesting aspect of Williams' overview is that among the AAC's that were reviewed none used induced affect, a point which Williams also stresses:

> A system for the real-time, adaptive induction of affective responses by algorithmic composition (either generative or transformative), including the affective evaluation of music by measurement of listener responses to such a system also remains a significant area for further work. (Williams et al., 2014, p. 18)

This might be misleading, or involve complications of definition that Williams doesn't discuss, but outside of Williams' overview Chung and Vercoe developed the *Affective Remixer*, that adapts to user movements, like head nodding, foot tapping, clapping, air conducting, dancing, and other gestural information to control the real-time (re)arrangement of musical pieces (Chung & Vercoe, 2006). The system was trained by monitoring foot tapping, skin arousal, and a survey, and organised its data in a two dimensional arousal/valence space. This accounts for both perceived and –in the case of skin arousal-induced affect, despite Williams' concern that no such thing exists.[3]

Chung and Vercoe also use a diversified approach to data collection: "Three unique sets of data were selected to provide diverse measures of affective response" (Chung & Vercoe, 2006, p. 395). Chung and Vercoe do not go as far as using their multiple affect measuring

3 The EEG precursor to this project, (Pitman 2012), also counts as an AAC utilising induced affect.

approaches for assessment and comparison, providing grounds for further refinement of each individual affect measurement method, but this extrapolated premise will be shown to be a significant development for this project.

1.2 Aims and Method

Considering all the developments in AAC systems presented thus-far, it may be implied that the greater goals of AAC researchers still involve developing the processes by which practical and robust AAC systems can be created AND assessed.

An ideal AAC system would be:

- Robust enough to be used in unsupervised real time tasks such as musical therapy, computer gaming, film, affective environment music, or other dynamic situations.
- Capable of predicting both perceived and induced affect.
- Endowed with multiple approaches to measuring and predicting affect.
- Able to entertain increasingly greater variations in musical structure.
- Less dependent on categorical emotional models and more inclined towards machine learning or regression.
- Somehow assessable or comparable in terms of success.

This particular project hopes to contribute directly to these greater goals in AAC, but is also more specific in several ways:

- Developing an algorithm focusing on musical structure (without timbre).
- Developing and comparing multiple methods for collecting both perceived and induced affect data.
- Establishing and exploiting predictive functions from this data using machine learning.

To achieve this, the following method is proposed:

1. Develop a musical algorithm that is specifically capable of a broad range of musical structures.
2. Play randomly structured passages from the algorithm to a volunteer audience and collect affect data using an array of biological sensors and surveys simultaneously.
3. Use neural networks to develop a predictive function between musical structures and how those structures affected the human audience.
4. Use neural networks to build predictive functions that can help us to automatically select music for any given affective target.
5. Through an interface, generate affective musical passages as a compositional aide, unsupervised real-time system, and both in response to a user's current affective state and to manually define affective targets.
6. Analyse, compare, and discuss the relative successes and failures of each element.

Novel Approaches

Several novel approaches in this method and differences to typical AAC methods can be observed in this work. This system is primarily focussed on statistical relevance, that is, it is designed to have as little prediction error as possible in its neural networks, which has several ramifications.

Firstly the music algorithm, *Two Hands*, accepts an array of variables that each define an aspect of musical structure expected to be present in the passage generated. However, it is implied here that every variable that can affect the musical outcome should be considered by the predictive function. It can be argued that some small changes might only have a negligible influence on the potential affects within a passage; however it can also be argued that in certain cases these small changes have a significant influence on potential affect. This forces the algorithm design to use relatively few variables overall, to avoid having neural networks relatively complex. Further, the musical algorithm is forced to rely on quite straightforward

implementations, so that the musical structures intended are apparent to the listener in each passage.

Next, this system does not prefer to transpose all of the biological sensor values via emotional models if possible. Instead, physiological changes are analysed directly by the neural networks, and as a result, are available for the end user to directly describe the target affective state. This means that rather than asking for a higher arousal, the user asks for increasing heart rate variations or increasing skin conductance. It was considered that categorical emotional models may provide an unnecessary and slightly abstracted layer of complication, which may negatively affect prediction error. The electroencephalogram (EEG) analysis is an exception to this, as it is unreasonable to consider EEG signal without a significant amount of analysis (discussed further in A Note on EEG Complexity p35).

The Likert survey focuses on subjective aspects of the music, rather than perceived emotional content. This has some interesting ramifications on the interface; a user can ask for a piece that seems “thematic” or that “ends well” or “is easy to dance to”. This approach is intended to compliment, rather than conflict with, the affective aspects of the physiological sensors, and the same could be said for the tag word section of the survey.

Limitations

This project does not intend to reconcile or reveal the relationship between the biological mechanisms it measures and higher intellect or emotional content. Instead this project relies on existing evidence in literature to guide what mechanisms are relevant to musical influence, and how to best monitor them.

EEG measurements are treated by Emotiv’s proprietary analysis software that translates the physiological measurements into affective categories automatically. Critiquing this translation is not an intention for this project.[4]

4 While such a translation seems ideal, it could be argued, and should be investigated, whether such a translation is actually beneficial in terms of predictive error. While simple biological mechanisms of the body seem quite reasonably predictable in

This project focuses on musical structure, not timbre. Timbre is a significant part of the musical affect phenomenon, but due to the much larger number of variables needed to implement broad-spectrum timbre into the music algorithm, which in previous research has been found to totally dilute musical structure in terms of predictive relevance, it was deemed necessary to separate the two. [5,6]

'Success' in AAC requires more investigation. It will be evident that 'success' here (and in other AAC literature) is represented by predictive error margins in the neural network, tested by comparing predictions against withheld data. It's not clear yet what the golden standard for 'success' is in AAC neural network training, nor can we yet describe if or how well this translates to satisfactory music composition.

Other limitations are present, and are best discussed in the relevant sections.

---

relation to musical affect, higher functions of the mind and emotional activity become increasingly complex and individualistic.

5 For this project, the piano is used as something of a 'vanilla' timbre, to which this AAC is now permanently tied. Whilst using this system, changing instruments (or timbre) would introduce a large change in affects, which are not allowed for by the predictive functions of this particular system.

6 For a study considering timbre that is highly relevant to AAC, see (Klügel & Groh, 2013)

2 Developing Methods for Affective Algorithmic Composition

Armed with Hevner's perceived affect example and Ellis and Brighouse's example for induced affect procedure, the development of an empirical affect study with an Affective Algorithmic Composition (AAC) implementation in mind is investigated. The following aspects, most important for making such an empirical affect study relevant for AAC development, are covered:

- A music algorithm specifically designed for AAC
- A range of surveys and biofeedback devices to collect data
- And a predictive function based on machine learning

2.1 Musical Algorithm Considerations

There are several approaches to musical algorithms in AAC literature. Chung and Vercoe's *Affective Remixer* recombines pre-defined sections of music into new arrangements, one of the more common approaches to algorithmic music often found in immersive computer game music (Chung & Vercoe, 2006). This category of algorithmic music is referred to as transformative. To generate music from scratch without preconceived musical pieces is categorised as purely generative (Williams et al., 2014, p. 9).[7]

The requirements for a musical algorithm within an AAC are very specific and in many ways limiting. For this project, total control over all levels of musical structure is required, thus a purely generative approach is the focus, so that all levels of detail, down to the individual details of each note, can be traced back directly to the algorithm itself.

To get a picture of how each musical structure relates to affect, the algorithm must be capable of a broad, if not extensive, range of structural outcomes, rhythmically, harmonically, and stylistically. Any variable in the code that influences musical structure should be included in the data presented to the neural networks. In effect, the complete list of these variables should describe a passage of music entirely; a structural array or seed.

The number of variables used and the number of different values for each variable increases the number of possible outcomes that can be created with the algorithm. For instance, ten variables with ten values each means that there are in total 10^{10} possible outcomes. At one minute each, hypothetically, this would be already more music than can be heard in two hundred and fifty human lifetimes. The predictive error of the system can be improved if the number of possible outcomes is low, at the cost of algorithm complexity.

With a clear focus on prediction error, there is less room for long Markov chains, advanced hyperlinking networks, or complex linguistic models. Manzolli's *Roboser/Emotobot*

7 Definitions and categories of computer generated music are not yet concretised, and seem to be fairly open to debate. There are many contributers (Ariza, 2005; Burns, 1994, Cope, 1991; Gerhard & Hepting, 2004; Miranda, 2000). However, various papers from America, Europe, and Asia still differ or do not specify exact definitions. For this work, William's example is followed.

composition system demonstrates the generation of an expansive range of novel musical variation using a minimum number of variables (Manzolli & Verschure, 2005, p. 55).

Either way, the number of possible outcomes is still going to be much longer than human lifetimes, and finding ways to navigate this large number of possible outcomes is one of the primary motivations for developing an AAC in the first place. One might argue that the algorithm could be limited in other ways, such as with mathematical filters or stylistic limitations, but rather than exploring the possible outcomes such measures would only be a form of human inference; predictably sacrificing certain kinds of solution in favour of others.

Natural limitations, such as restricting the notes to the range of two hands on a keyboard, or to a certain number of voices and ranges, can help reduce the total number of outcomes in more natural ways, making approaches to creating a broad-spectrum algorithm a very individualistic and creative endeavour, despite these inherent limitations.

2.2 Collecting Affect Data

The studies of Hevner and Ellis were early examples of nearly a century of empirical affect studies that can inform the AAC developer in choosing biological phenomenon to monitor. There are several imminent technologies, overviewed here, which are quite practical, accessible, and ready to be deployed in AAC development, as well as some emerging technologies that show much promise for AAC endeavours. Some factors that may influence the choice of sensors may include finance, expertise, and access to facilities and equipment needed to measure the phenomenon. Measuring the phenomenon may also have ethical issues that need to be considered (e.g. invasive surgery, privacy invasion). Finally, the apparatus may influence the subject in such a way as to undermine the results (blocking ears, intimidating or uncomfortable apparatus).

Once a phenomenon has been chosen, further consideration must be given to implementing the sensors and their results. As Ellis and Brighouse outline, a biological measurement taken during musical stimuli often needs be compared to the same biological measurement taken

during a neutral period, usually immediately before or after the musical stimuli. Once connectivity, method, and procedure have been established, an analysis of the data must be considered so that concise data is presented to the predictive functions and later to the interface.

An excellent and thorough overview of various physiological reactions to music can be found in the *Oxford Handbook of Musical Psychology* (Hallam, Cross, Thaut, & Hodges, 2008, Chapter 11), which even includes information on biochemical responses and gastric motility. Particularly practical apparatuses are discussed here, along with considerations for developing musical affect surveys, and considerations for developing the procedure through which this data is collected:

Cardio

As already discussed, cardio-rate was shown to have a strong relationship to tempo in music and this is confirmed in more modern studies. It has also been shown that many musical structures contribute to changes in heart rate variability or variance (Bernardi et al., 2006; Birnbaum et al., 2009). It is therefore essential that any biofeedback study that seeks to ascertain any manifestation of musical affect consider both the change in heart rate and heart rate variability.

Facilities/Equipment

There are many different kinds of cardio monitors available which are simple and effective. Particularly common are electrocardiograms (ECG or EKG from Greek: Kardia) that are worn around the chest and monitor the considerable electrical field changes caused by the heart as it beats. Blood oxygen detectors monitor the blood's oxygen content as it fluctuates with each beat, the colour change easily detected at the fingertip's or earlobe's surface using a green LED and a light sensor. Recently both ECG and oxygen monitors have become very common commercial fitness products, often integrating easily into computer systems via USB or wirelessly.

Most devices automate the calculation of heart rate or send a regular real-time beat signal (the latter being advantageous). Further, there is a range of effective manufactured products that integrate into microprocessor systems like Arduino or Teensy. Assuming the subject cannot see the heart beat GUI, the sensor shouldn't impact results or deter the participant in anyway, thus there should be no unusual ethical concerns apart from minor discomfort.

Analysis

Typically, heart rate is measured in beats per minute. It has also been shown that sudden or tiny changes can be masked by such a simple analysis (R. J. Ellis et al., 2009, p. 1), thus it is also valuable to also calculate heart rate variability or variance (HRV). An increase in HRV suggests that the heart is less consistent in its rhythm or inter-beat intervals (IBI). Mathematically, HRV is the square root of the mean of squared successive differences in IBIs and is typically measured in milliseconds (Task Force of the European Society of Cardiology the North American Society of Pacing Electrophysiology, 1996).

$$For\ N\ IBIs, \qquad HRV = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(\mathrm{IBI}_i - \mathrm{IBI}_{i+1})^2}$$

It is certainly possible to analyse cardio-rhythm and blood flow in more and more depth, including the oxygenation of blood, cerebral artery flow, blood pressure, and baroreflex (Bernardi et al., 2006, p. 445), however these measurements require increasing expertise.

Respiration

We are concerned here with physiologic respiration, specifically natural breathing. The effect of music on breathing rate has been anything but conclusive in literature. Breath rate has been associated with musical preference (Ries, 1969, p. 62), tempo (Bernardi et al., 2006, p. 448; Ellis & Brighouse, 1952, p. 42), or not with music at all (Davis & Thaut, 1989). Ellis and Brighouse state that some musical influence on breath rate might not even become present until minutes after the stimulus (Ellis & Brighouse, 1952, p. 47). It seems a necessity to

preface any discussion of music's effect on respiration with a disclaimer that research has not been conclusive.

Facilities/Equipment

Much research has been done by manually counting breaths as the participant's torso expands and contracts, however AAC development requires a more autonomous solution. Several devices that attach to the face or even inside the nose or throat, monitoring airflow directly, are found in medical applications. Such intrusive devices are not ideal for AAC.

Eulerian Video Magnification (EVM) is a method for remotely breathing (and also heart rate) using video analysis (Wu et al., 2012). Subtle or minute temporal variations, normally impossible to see with the naked eye, are magnified in video data. This has led to applications where infant breathing can be monitored remotely via video, as well as pulses, structural vibrations, and other minute but repetitive movements. EVM is implemented using Matlab.

Another unique approach to monitoring respiration is a rubber stretch sensor (Coyle, Mitchell, O'Connor, Ward, & Diamond, 2009). A rubber cord or strap is cast with fine iron particles throughout that allow the rubber to modestly conduct electricity. As the rubber cord is stretched the resistance offered increases. This creates a very sensitive and useful stretch sensor that can be attached around the chest and/or abdomen to give feedback about the expansion or contraction of the torso while breathing.

Analysis

A breath sensor will typically track the number contractions or expansions over a minute (bpm) and also the relative size of each breath, potentially as a ratio or percentage of the maximum expansion of the torso.

It is useful to allow for appropriate calibration of maximum and minimum thresholds in any breath counting solution and to perform this calibration diligently with each new participant, as lung size, torso size, and breathing style varies throughout the population.

Skin Conductance

Also known as Galvanic Skin Response (GSR), skin conductance was very popular as a simple and effective psychological marker in the sixties and seventies, and is famous for its use in polygraphs (lie detectors). This bioelectrical phenomenon is strongly associated with emotions such as preference, fear, anger, disorientation, and anxiety. Chills, shivers, or pilomotor responses (goose bumps) can also be detected through skin conductance (Guhn, Hamm, & Zentner, 2007; Harrison & Loui, 2014). Not surprisingly, skin conductance is a core physiological cue in many music affect studies, and considering its simplicity, is an excellent candidate for AAC development (Coutinho & Cangelosi, 2008, 2011).

The underlying mechanism revolves around measuring the ability for the skin, usually across the palm between two electrodes, to conduct or resist a small electric current. This resistance changes with the amount of fluid present in and on the skin, mostly due to sweat. Where voltage is constant, the current change is linearly proportional to sudomotor activity; a sympathetic nervous response (Lykken & Venables, 1971, p. 659).

Facilities/Equipment

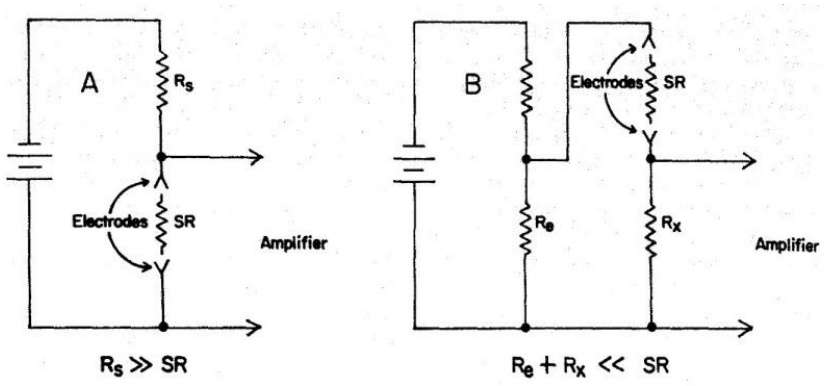


Figure 3 – Measuring voltage (left) and current (right) (Lykken & Venables, 1971, 656)

Due to their simplicity, skin conductance sensors are commonly constructed, rather than bought. In Figure 3, Lykken outlines two common circuits used. In the resistance loop circuit on the left is commonly found in many DIY guides and Arduino kits and varies the voltage as skin resistance changes. However, Lykken argues that current and not voltage should be

measured, as the typical resistance based voltage divider only returns a small amount of variation compared to the voltage at the load-bearing resistor. In the right diagram the voltage across the electrodes is kept roughly equal to the voltage at Re, thus the voltage at Rx varies directly with current across the electrodes.

Either circuit will work to a degree with an Arduino, even unamplified, however the larger dynamic range of the conductance circuit can improve the resolution of the digitalised signal significantly, as would signal amplification with an OP-amp (a small IC based amplifier). More sophisticated circuits build on these basic concepts (MacPherson, MacNeil, Marble, & Reeves, 1976).

The electrodes themselves must also be considered. In development, metal film electrodes that wrap around the finger provoked extra sweating. Certain “obvious-choice” metals such as aluminium foil perform poorly as electrodes. Ideally nickel or copper should be used, and fastened either by a conductive gel or a breathable binding so as not to induce sweating. Hand movements or the touching of conductive materials should also be avoided by participants.

Finally, some consideration of room temperature may help to avoid unexpected skin conductance changes.

Analysis

There are two main concerns for measuring skin conductance, skin conductance level (SCL) or tonic level, and skin conductance response (SCR). The SCL is a long term standard, often taken as a mean value, or isolated entirely from stimulated changes. Phasic events are temporary spikes in response to emotional activity, such as being surprised or scared. SCR refers to the length in time these phasic changes last for, or more accurately, from an event’s beginning through to the point where it returns to half its peak. These phasic events also have a long-term effect on the SCL (Lykken & Venables, 1971, p. 657).

SCR can vary from seconds to minutes, thus is often difficult to measure during short passage of music. Further, a phasic change or an uncommonly high number of phasic changes

will affect the SCL reading, thus it could be considered that for short passages of music such as the ones in some AAC projects, SCL alone is a reasonable measure.

Eye Tracking/Pupillometry

Pupillometry involves measuring the diameter of the pupil of the eye. Since the 1960's pupil diameter has been shown to relate to psychological stimulus in many animals from birth including humans (Goldwater, 1972; Laeng, Sirois, & Gredeback, 2012). There is strong evidence that pupil dilation and contraction is related to appreciation or disapproval (Kuchinke, Trapp, Jacobs, & Leder, 2009; Mudd, Conway, & Schindler, 1990), as well as changes to timbre and pitch (Hallam et al., 2008).

Electrooculography (EOG) is the measurement of electrical fields created as the eye, with a considerable inherent electrical potential, moves about. This is often measured with electrodes placed at the front of the temples.

Facilities

The Emotiv *Epoc* EEG headset can return some information about eye movement, likely using EOG methods (although this is only speculation, as the workings are proprietary) with mild reliability, but no measure of pupil dilation.

A pupillometer measures pupil dilation using electronic or infrared cameras that take a very close up picture or film of the eye. The apparatus requires that the head of the subject be held very still, usually with a brace or clamp. This is very uncomfortable for any period of time, and despite the huge potential for pupillary information in an AAC study, no pupillometry apparatus has yet been suitable or adopted for this project.

There is an emerging technology most notably implemented in prototype smart eyewear systems such as Google Glass or the ASL Mobile Eye Device. When worn these devices can track the user's eye movement and pupil dilation as well as the environment the user is viewing, so as to be able to distinguish changes which are reactions to environment from

those which are elicited by cognition (Józsa, 2010). Technology such as this, while intended for web based applications, would be ideal for future AAC developments.

Electroencephalogram

Neurons in the brain quickly build up a positive charge by absorbing ions from the fluid about them. When the neurons are activated, the charged is released to travel down the axiom inducing the release of various chemicals to receptors of other neurons. The neurons in the cerebral cortex are organised into 6 layers, parts of which are known to fire somewhat cohesively and in waves. The location and frequency of these waves gives us some remote information about what part of the brain is being used and to what degree.

There are several devices that can detect and monitor the tiny electric fields produced by the brain such as electroencephalogram (EEG), magnetoencephalogram (MEG), functional magnetic resonance imaging (fMRI), and computed axial tomography scanning (CAT).

fMRIs and MEGs produce much better imaging resolution than EEG, however require magnetically shielded rooms, large and expensive equipment, expertise, and can potentially isolate or intimidate a participant. CAT scans are x-ray based so are unethical for use in music research. Thus AAC typically focuses on EEG.

EEG uses small electrodes to measure electric field changes around the skull. Electrodes work best when placed as close to the neurons as possible and surgically implanted electrodes have even enabled some control over robotic limbs and the interception and reconstruction of vision (Regalado, 2014; Stanley, Li, & Dan, 1999). However, most research applications only ethically warrant placement of electrodes on the outside of the skull, naturally including AAC.

Even the best medical EEG systems suffer from a low spatial resolution. Each electrode can only measure the mean field of the tens of thousands of neurons beneath it. Penetration is also limited as the skull inhibits the signal, and muscle activity in the scalp causes interference. EEG is still by far the cheapest brain scanning technology, ranging from DIY

kits, toys, consumer oriented prototypes, through to the full laboratory grade apparatus with hundreds of electrodes. It also is relatively portable, requires relatively less expertise to operate, and has a long history of use in music related research.

Knowledge of bioelectrical fields in the body and the brain were established by Galvani and Canton in the seventeenth century, although invention of the EEG is credited to Hans Berger in 1924 (Collura, 1993). Berger and his invention were oppressed by the Nazi regime, and the EEG went without acknowledgement by the scientific community until 1934. EEG research has since spread all over the globe, particularly praised for its uses in diagnosing epilepsy and other abnormalities of the brain.

Rosenboom used EEG extensively in the 1970s to study, "… information processing modalities of the nervous system as they relate to aesthetic experience and creative activity" (Rosenboom, 1976). In 1988 detailed studies to concretise brain processes elicited by music found that a number of EEG parameters related to specific musical tasks (Petsche et al., 1988, p. 133), including a range of consistencies between groups of specific gender (Petsche et al., 1988, p.142), and those with similar levels of musical training (Petsche et al., 1988, p.139).

From the end of the twentieth century to the present, the EEG has experienced a wealth of musical research attention. There are many biomusicological studies using EEG to analyse manifestations of musical affect specifically (Behroozmand, Korzyukov, & Larson, 2012; Fedorenko et al., 2012; Janata & Petsche, 1993; Miranda, Sharman, Kilborn, & Duncan, 2003; Pitman, 2012; Schaefer, Desain, & Suppes, 2009; Steinbeis & Koelsch, 2008; Tan, 2012). Since 2000, EEG technology has taken centre stage in a lot of music affect research, propelled by Miranda, and others as well, who show that EEG can not only be used to analyse musical affect, but should also be able to control aspects of music creation as a brain control music interface (BCMI) (Miranda, 2010; Miranda & Brouse, 2005; Miranda et al., 2003).

Facilities/Equipment

Essentially the AAC researcher has three options: 1) Build an EEG, 2) collaborate with a hospital department, 3) use one of several prototype consumer level devices that are now quite readily available.

Through the nineties, as personal computing advanced rapidly, a surge in non-medical and non-academic interest by the general public created a small demand for consumer EEG technology. Hobbyists began to develop DIY systems, and formed communities such as the OpenEEG project, hosting an online forum and a database of free techniques and plans to build functional electrodes, amps, filters and even some software ("OpenEEG,"). A European company called Olimex produces several relatively affordable kit versions of some OpenEEG multi-electrode designs.

This public demand also resulted in a number of commercial products becoming available. In 1992, Biocontrol Systems released a range of practical physiological sensors for the arts, including single electrode EEG, as well as EMG, EOG, and ECG, called the *Biomuse System* (Knapp, R. B., Lusted, H. S., & Lloyd, A. M., 1993).

In 2007 Neurosky released the *MindSet*. It sports a single electrode and IC that was notably used in a series of toys such as the *Starwars Force Trainer* by Milton and the *Mindfle*x by Mattel. These were often cannibalised by DIY enthusiast for multi-electrode systems. Single electrode systems have been employed in several products to date, many utilising NeuroSky technology, mobile phone apps, and simple Bluetooth connectivity. Single electrode systems can be useful in AAC development, but there are more sophisticated options.

In 2011, Emotiv began releasing developer kits for their *Epoc* headset. The *Epoc* featured a new level of sophistication and was designed with both medium level research and computer gaming in mind. It featured 14 electrodes around the head, bluetooth connectivity, basic head tracking accelerometers, software with range of automated brain and EMG analysis features for the non-expert user, and a research edition that reports raw electrode data of a quality suitable for intermediate research applications (Badcock et al., 2013; Duvinage et al., 2013).

The ‘research edition’, though more expensive, is also compatible with dedicated EEG software platforms such as *OpenVibe*, Mathworks’s *EEG Toolbox* for Matlab, or *BCI2000*. Emotiv is planning to release a new 5 electrode headset, the *Insight,* in 2015.

In 2014, the *OpenBCI* system was released. It is an open source microprocessor hobby/development board similar to an Arduino or Teensy, and features 8 electrode inputs per board. The boards interact with computers via USB and Bluetooth. As a research tool such a flexible system might be quite useful, however may require more expertise in both EEG implementation as well as an understanding of programmable hardware. The relatively new community has not developed a huge range of firmware, software, or analysis options yet, however this is likely to improve in the future and may soon offer an excellent, non-laboratory, alternative option to the infamously proprietary Emotiv headsets.

Analysis

Electrodes are fitted to the skull of participants using various systems, including rubber caps, conductive gels, or plastic clamp designs. The positions of the electrodes are usually identified by the “10-20” system, named after the series of measurements, in degrees, that a practitioner uses to find each location. 21 key points are located around the scalp.

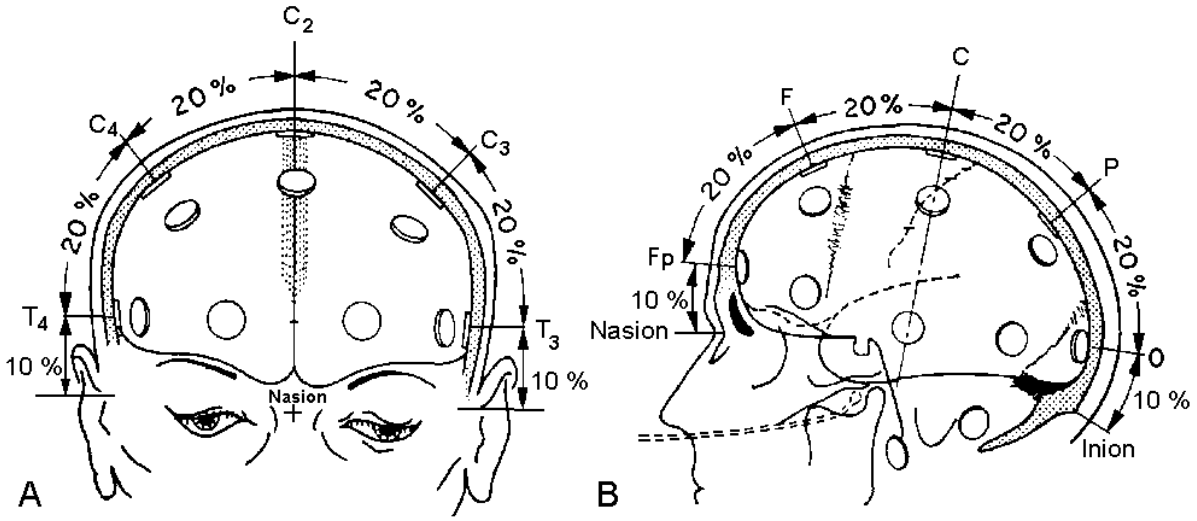


Figure 4 – The 10-20 Electrode System of the International Federation (Jasper, 1958, p.371)

EEG signal suffers from a high noise ratio. As the electrodes are placed on the exterior of the skull, muscle movement, eye movement, or a general physical disturbance of the electrodes will ruin the signal. To combat this, the recording of electrical fields is divided into overlapping segments called epochs; usually 1 or 2 seconds in length. Any epoch that contains artefacts or noise is ignored. The overall value of each electrode is compared against a reference or earth electrode (often at the earlobe) or sometimes against the mean value of all electrodes. Often a range of temporal, frequency, and impulse filters are used to finally attain useable data.

This data can be analysed in several ways. Most common is a simple frequency analysis in which each area of the brain is described in terms of its frequency of operation (see Figure 5), as initially demonstrated by Berger (Collura, 1993, p. 485).

Label	**Delta**	**Theta**	**Alpha**	**Beta**
Range	1-4Hz	4-8Hz	8-13 Hz	13-30Hz
Associated Behaviour	Deep sleep	Light sleep, waking	Meditating, resting	Awake, perceptive, problem solving

Figure 5 - EEG Frequencies and Brain States

Coherence is used to describe two areas of the brain sharing a similar frequency or phase. If the electrodes are neighbouring they share local coherence. If they are hemispherically opposite, they are said to have interhemispherical coherence. Coherence has been shown to be quite common and useful in identifying musical activity (Petsche et al., 1988, p. 135).

An Event Related Potential (ERP) is a signal change in response to a particular stimulus or event, for instance, a reaction to pain or succeeding in a test. Most ERPs are specific to the individual, but there are ERPs which have been found to be consistent throughout the population. For example, the P300 ERP is a specific ERP phenomenon localised to the occipital and central areas, and occurs after novel or salient stimuli (Polich, 2007, p. 2128).

The N100 ERP is elicited by unpredictable stimulus or a null result for an expected event and is particularly sensitive to aural stimuli (Näätänen & Picton, 1987, p. 375). ERPs are often exploited in BCI programming.

With some expertise, more detailed analysis can be performed on raw data using a range of software. Matlab features an EEG toolbox, BCI2000 is a dedicated EEG analysis program, and OpenVibe is an open source modular programming environment.[8]

A Note on EEG Complexity

When considering an EEG as part of an AAC project, laboratory equipment is perhaps assumed to be ideal, yet requires some considerable expertise. During development of this project, some time was spent by this researcher at the Royal Adelaide Hospital, investigating and being trained to use the NeuroCart biofeedback system (specifically the EEG aspects) at the Pain and Anaesthesia Research Clinic. Even though the laboratory grade NeuroCart system is utmost in terms of signal clarity, and having completed the training required to collect the data, there was no end to complexities of analysing EEG data.

Jacques Vidal, established as a pioneer of brain control interface using EEG in the 70s, reminds us of the complexity one faces in EEG analysis. He frequently criticised his peers and the equipment of the day:

> '…it had become obvious that most current methods and practices of EEG data acquisition and processing were utterly inadequate for the level of discrimination that was required…' (Vidal, 1973, p. 164).

Even the most extensive analysis still seems unable to identify musical affect with any authority, which is a major hurdle in the field of AAC. Much research in the field seems to be grinding against the complexities of EEG analysis with some faith that in the very near future

8 For an overview of EEG software see (Brunner et al., 2013).

EEG will somehow untangle its self and become considerably more robust. This is not entirely unlikely, as the Emotiv *Epoc* headset has been shown to provide a reasonable EEG affect analysis "out of the box" that is satisfactory as a starting point (Badcock et al., 2013). This analysis is very simple, targeting states of alertness rather than emotional categories, but is unfortunately not music specific.

The promising BCMI-MIdAS project (Williams et al., 2014, p. 18), involving EEG and fMRI experts (and from other fields) promises to provide more insight into gaining musical affect information from EEG signal. EEG in its current state doesn't qualify as being any more useful for AAC development than biosensors that focus on simple and more predictable mechanisms throughout the body. This implies that AAC development requires a multiple sensor approach.

Body Temperature

Early music therapy experiments showed no significant relationship between music and skin temperature (Guzzetta, 1989; Zimmerman, Pierson, & Marker, 1988). In more recent times, body temperature has been shown to react to musical stimulus, but those studies have been conflicting about how (Lars-Olov Lundqvist, 2000; Rickard, 2004). Never the less, Rickard points out that these reactions are at times relatively large and the apparatus for measuring body temperature is considerably cheap and convenient, a clear argument for the inclusion of a body temperature sensor in an AAC system.

Facilities/Equipment

There are a large range of digital thermometers on the market for very little cost although it is worth specifying a higher degree of accuracy. The equipment chosen should measure at least tenths if not hundredths of a degree. Some digital thermometers will connect via Bluetooth or USB, and a significant number of devices for Arduino or other microprocessor systems are on the market, including simple individual components such as thermistors and the TMP36 integrated circuit which returns a voltage signal linear to the Celsius scale.

Analysis

Body temperature is considered here in degrees Celsius (℃). The temperature of a human body can range from 36.0℃ to just under 38.0℃, and baseline measurements can vary based on where the measurement is taken. 37.7℃ would be considered a fever in most adults, however half degree variation are common over the day, and the body temperature can also be affected by hunger, sleeplessness, or illness. 37℃ is considered normal internal body temperature, 36.8℃ under the tongue, and increasingly lower when measured externally on the skin away from the torso (Longo, 2012).

Movement Tracking

The link between human movement and music has been seen as so complex that some researchers conclude there must be more than one mechanism involved. Some tripartite-category systems have been proposed (Berlyne, 1971; Dowling & Harwood, 1981; P. N. Juslin & Sloboda, 2011; Sloboda, 1998) that suggest that music is linked to movement through a formal similarity in the emotional signals. While there is a considerable need for empirical research in this area (P. N. Juslin & Västfjäll, 2008, p. 570) the mere existence of dancing and the nodding-to-music phenomenon suggests a strong if not fundamental link between musical rhythms and body movement. However, do not disregard the archetype classical music experience where the audience sits utterly still despite reporting to have been ‘moved greatly’ (P. N. Juslin & Sloboda, 2011), uncannily similar to the participant’s experience listening to algorithmic music in an empty room surrounded by a restrictive battery of biosensors.

Several relevant movement-capturing technologies are worth mentioning. EVM analysis and body tracking video technologies similar to Microsoft’s *Kinect* are immediately implementable, however more subtle measurements of body movement or rhythmic acknowledgement can be gained through contact microphones, bend and stretch sensors via microprocessor technology, or chairs with inbuilt pressure/stress sensors. Further,

electromyography (EMG) is an electrode-based technique for picking up muscular neuron activity in a very similar way to EEG.

Survey

A survey (while not technically a biofeedback element) is the primary method for collecting perceived affect as shown in the Hevener case study (see Perceived Affect p11). Any of the many established survey styles is useable in AAC, but some consideration of the resulting interface should be given. Williams shows that this is the dominant form of AAC system to date (Williams et al., 2014, p. 11), and perceived affect is not going to become obsolete or without potential as induced affect systems emerge. Survey design also provides some of the most creative opportunities in AAC development.

The questions used in the survey should ideally reflect the contents of the interface elements desired. Where a Likert scale survey is employed, the resulting interface control will likely reflect a scaled control like a dial or slide. Where binary check box type options are used, the resulting interface will follow a similar form. It could be argued that the interface design will be largely informed by the survey design or vice versa.

A survey can target perceived emotional content such as happiness or darkness, but it can also identify passages that the participants subjectively find very impressive or utterly boring. Music that suggests certain themes can also be identified through a tagging system. A survey can also target subjective musical features that are otherwise difficult to quantify. Pieces which are cadence-like can be identified by asking the participants subjective questions about whether or not the piece “felt like it ended well”. Further, and this is particularly relevant to those who might study or experience synaesthesia, the survey might be able to enquire about colours or smells, that will then reflected in a palette of colours and smells in the end user interface itself.

There is no lack of creative possibilities for survey based AAC, however the more complex or abstracted the relationships between the questions and the musical stimulus, the more likely

that the predictive function will suffer struggle to find statistical relationships between the more abstracted survey results and the music played. For example, the survey that asks about subjective rhythmical content may potentially have a much lower prediction error than the survey that tries to ascertain what kind of animal the music invokes.

Developing Auditioning Procedure

The auditioning procedure is the process by which participants are exposed to musical passages and return their perceived and induced affect feedback data. It should initially be noted that assessing the reactions to music via physiological changes is an indirect approach. Among the many factors that affect our physiological states, reactions to music are but one subtle factor among many. Participants may have had too many coffees that day, or may have endured considerable exercise beforehand. They may have not slept well, or suffered an emotional blow such as a relationship breakdown. Even some basic medical conditions, such as arrhythmia of the heart or asthma, may need to be identified before the procedure can begin, and may also require some ethical consideration. Furthermore, for the duration of the experiment, the participants will need to be denied any stimulation other than the musical algorithm's output. Nevertheless some external factors may still influence the results and need to be identified later.

In addition to the guidelines laid out by Ellis and Brighouse (1952, p. 40), the following checklist may also be considered:

1. Does the participant feel fairly normal today? Tired? Hyperactive? Sick?
2. Did the participant sleep relatively normal hours the night before?
3. Has the participant enjoyed a fairly normal routine diet on this day?
4. Has the participant endured any recent emotional disturbances or trauma recently?
5. Does the participant have any medical conditions that might affect the results or hinder the sensors themselves? This might include a pacemaker or heart condition, asthma or breathing difficulties, an abnormal skull, an inability to sweat, hearing loss, or a diagnosed mental health issue such as irrational mood swings or depression.
6. Is the participant under the influence of or recovering from any significant substances such as drugs, anaesthetics, alcohol, or anti-depressants?

Further, consider the space in which the procedure takes place. All forms of external stimulation like televisions, paintings, interesting equipment or instruments, internet, mobile phones, intrusions from other people, background noises, or music need to be removed. The changing state of the participants must be considered as well. With only algorithmic music to listen to for more than forty five minutes, participants often become bored, numb or sleepy, thus disengaging from the musical stimulus. Breaks should be scheduled regularly.

As Ellis and Brighouse outline, each musical stimulus should be compared to a neutral period directly before or after the musical stimulus is played (Ellis and Brighouse, 1952, p. 40). The biological measurements need to be considered as relative to this neutral period, measuring the difference in physiological state from before stimulus to after stimulus, rather than an overall biological values the may not be stable or consistent for each individual. Even so, a passage that would otherwise increase a participant's heart rate may have less or no affect if the participant's heart rate is already peaked, thus a period after hearing each passages is recommended to allow the participant to return to a more neutral physiological state.

Overall, the goal is to collect enough stochastic data for more generalised patterns to emerge. How many samples are required to achieve this involves many factors, and as yet an effective ratio or standard has yet to be defined.

2.3 Predictive Analysis- Creating a Computational Critic

The next problem, having collected the appropriate amount of affective data, is modelling this data into a predictive function. Alpern describes a *Computational Critic*:

> For a simple problem with a well-defined solution, such as performing symbolic regression (the process of deriving a function to produce a set of data, given that set of data), creating the critic is easy -it merely has to calculate how close each data point generated by a program is to the target set. For a complex domain such as music, where judgements must be made about what is good music, something which is very hard to qualify, creating this critic can be a very difficult process. (Alpern, 1995, p 13)

As Alpern implies, a form of regression analysis is called for. There are several approaches to regression analysis in AAC literature. In earlier empirical affect literature, results were often calculated manually and were presented as isolated or linear relationships, but modern biomusicological studies have found that the relationships between musical structures (including timbre) and how they affect humans is more complex.

Affect data also has a very specific nature. The relationship between music structure and affect is likely to be non-linear and covariance may exist between all variables, that is, no one structure variable will have an isolated relationship with any single affect outcome. The population (all possible outcomes) of an algorithm, broad enough to cover a large number of combinations of musical structures, is typically very large (hundreds of billions) and the number of training samples is bound to be relatively very small (hundreds).

The data is stochastic in nature, that is, has a greater error when the population is larger or when the sample size is smaller. Therefore a useful solution will be generalised, that is to say, only the most common trends are relevant or helpful and less common trends are increasingly unhelpful. Further, as stochastic data sets become larger, the number of functions that may operate as solutions approaches infinity.

Both the musical structure and affect aspects of the data contains discrete and continuous numerical values, and both nominal and ordinal categorical values. Typically musical structure data is more detailed than affect data (i.e. more variables) thus statistically predicting affect given musical structure is going to be more reliable than predicting musical structure given affect.

Finally, the data is unique and unrepeatable. Different groups of participants, or even the same group at a different time, give different results. Similar to Hevner's disclaimer (see Perceived Affect p11), no overarching statements about 'how music works' can be formulated even where a formula is revealed, as this will only represent the tiny sample of people used to audition the algorithm in the first place.

To analyse data of this kind requires a very careful and methodical approach, and to do this well, an aspiring AAC developer is likely to need a firm grasp of statistics, as well as advanced regression techniques (such as machine learning) as suggested by Alpern.

Symbolic Regression

Symbolic regression is the reduction of an observed pattern into a symbolic mathematic function. As an example, it may seem straightforward to see a suggestion of a straight 45 degree line in your data, and to describe it as $X = Y$. As the example increases in complexity so does the mathematical language, until the current limitations of symbolic mathematical language is reached.

Symbolic regression was first implemented in genetic programming by John Koza (1994). A library of mathematical formula 'pieces' are randomly arranged and then altered using a 'survival of the fittest' routine until a best fit (given a specified level of generalisation) is found. The resulting formula can then be used as a function to predict Y given X. Each time the symbolic regression is performed a random starting point is used, and in complex scenarios, an almost infinite number of results conforms to the same data given a margin of error.

The result is limited to that which is describable in symbolic mathematics, the size of the library used, and how many pieces can be employed. Developing a describable relationship is often desirable in many applications, such as developing engineering models, or generating electronic circuits (Dabhi & Vij, 2011; Zdaniuk, Walters, Luck, & Chamra, 2011).

Neural Networks

Neural Network (NN) is an umbrella term for a class of various statistical models that use layers of adaptive weights joined by nodes and are capable of approximating non-linear functions. NNs were initially based on the simulating neuron pathway organisation in organic brains; computational studies focussing on functional simulation, biological studies focussing

on simulating the entire biology of the neurons. This project uses functional studies for statistics and machine learning.

NNs have been in development since 1943, when (McCulloch & Pitts) published *A logical calculus of the ideas immanent in nervous activity*. NN research would be fairly slow until the end of the twentieth century, when suitable computing power, the development of error back propagation (Rumelhart, Hinton, & Williams, 1986; Werbos, 1974), recurrent networks, and deep learning models gave rise to a series of NNs with beyond-human performances in tasks such identifying handwriting or traffic signs (Amara & Schmidhuber, 2012; Ciresan, Meier, & Schmidhuber, 2012).

A typical supervised feed forward network, used for matching functions to observations, involves nodes arranged into an input layer, a number of 'hidden layers', and an output layer. Each node in the input layer reflects an input variable from the data set to be modelled, and each output node reflects a target variable from the dataset. The nodes in the hidden layers contain activation functions which activate when its collective inputs surpass a threshold. Activation functions can include binary 'on-off' functions like a transistor, a sigmoid gradient, or other logical functions, allowing for a range of possible behaviours.

It is the pathways themselves that 'store' the information. Every possible pathway through the network exists initially with a random weight, but these weights get adjusted to match the inputs and outputs during back propagation. As the dataset is run through the network, more common trends in the data begin to manifest as increasingly weighted pathways. Once the network is trained, making predictions simply involves presenting a set of inputs that in turn gives a set of the most likely outputs according to the most weighted paths.

The function derived is totally abstracted in the weights of the pathways, leaving it nearly impossible to decode. Further, NNs can build mathematical representations that defy symbolic description. A well-structured and well-trained network can be very robust, given reasonable data, although there can be a lot of tweaking of architectural variables to achieve robustness, hence a reasonable degree of expertise is required.

Comparison

Several studies comparing GP and NN performance show that these two methods can be more accurate than linear regression (LR), however in complex cases, both GP and NN provide different solutions each time they are generated (Dabhi & Vij, 2011; Dolado & Fernandez, 1998; Zdaniuk et al., 2011). GP has the advantage of providing a writable formula of its solution, but this feature is not of much significance in AAC, where only functional implementation is required. It would be naive to assume any formula derived from such limited data could be authoritative enough to create a conceivable rule or guide (Hevner, 1937, p. 627).

While these studies suggest that GP predictions tend to have less error than NN predictions, there is a tendency to focus on fairly empirical industrial and engineering domains and problems. Where GP constructs its results from pieces of pre-written symbolic mathematical formula, NN solutions are completely abstract, and able to form relationships that are not describable with symbolic mathematical language (Dolado & Fernandez, 1998, p. 157) thus one might expect that severely abstracted domains that defy description (such as AAC data) may benefit NN analysis.

Feed forward NNs are well-established as universal approximators (Hornik, Stinchcombe, & White, 1989) that can handle deeply complex, noisy, and abstracted data, and are particularly common in biomusicological and similar AAC studies (Coutinho & Cangelosi, 2011; Klügel & Groh, 2013; Korhonen, 2004; Minjun Jiang, 2010; Miranda et al., 2003; Schubert, 1999).

NN and GP represent the most common ways of dealing with such interdependent and multivariate data, however, current developments in deep learning networks, non-linear support vector machines, and other machine learning systems suggest more approaches could soon be available (Meyer, Leisch, & Hornik, 2003; Patil, Pressnitzer, Shamma, & Elhilali, 2012). Further, the field of neural computing itself is expanding quickly.

A thorough investigation of machine learning and regression techniques, specifically for such abstracted data such as is found in AAC studies, is part of a clear direction for future research.

Training of a Neural Network

A feed forward neural network's architecture is defined in many ways and often much tweaking is involved in achieving a robust system. There are also many considerations in formatting the data to work in a neural network environment. NNs are typically implemented with statistical software such as *Matlab* by Mathworks, IBM's *SPSS*, or the open source, *R*.

Each input of the NN will correspond to a value in musical structure array. While tempo (a numerical value) can be represented by one input variable, a number reflecting one of ten musical scales to use (a categorical value) requires a binary input for each musical scale. The input values range from either 0-1 or -1 to 1 and are mapped to the numerical values as to a degree of accuracy in decimal points, or as binary integers for the categorical values. The scale of these mappings does not need to be interdependent. Similarly, the output variables must be scaled in the same way, and are mapped to the affect results.

The layer of nodes between the inputs and outputs is called the hidden layer. Deep learning systems use multiple, task oriented, hidden layers (Ciresan et al., 2012), but in a feed forward system one hidden layer, given enough nodes, can theoretically represent any mathematical function (Hornik et al., 1989). It is difficult to ascertain how many nodes are required when the complexity of the function is unknown. There is no set rule, but a rough guide may be found by averaging the number of input and output nodes, adding half again, and experimenting from there, trading processing efficiency for complexity. A simple relationship might only require ten nodes, or hundreds of nodes might be appropriate. Some experimentation should be anticipated by the AAC developer.

The weights for the pathways between each layer are initialised in a random fashion. When training begins, a sample is chosen from the data, and the inputs (in this case musical

variables) are entered into the network. The resulting outputs are calculated and compared to the known variables of the sample. In the back propagation phase, the weights of the pathways are adjusted according to the error between the outputs and the known result. This process is then repeated with the next sample, the entire data set often used several times over (each time being an epoch). Simply put, if the error stops decreasing significantly after each epoch, training is complete.

Too much adjustment during back propagation can create an unstable network, so the amount of adjustment made is often a function of the error and the weightings themselves. Further, as network training becomes less and less generalised, there is a threshold beyond which it is no longer useful to train the network, as it will only begin to entertain outliers and stochastic error. Thus random sections of the data are often kept aside for comparison so that changes to the network weights can be stopped when they no longer positively affect both the training data and the test (or validation) data. **Error! Reference source not found.** shows plots of prediction versus known results over several stages, from untrained through to extremely overtrained. Weight adjustment and generalisation methods require much expertise and are usually prewritten by experts as learning or training functions. There are many available learning functions available that can guide the training of a network. (Mathworks, 2014) provides a functional overview and comparison of some common learning functions, two of which might be relevant here.

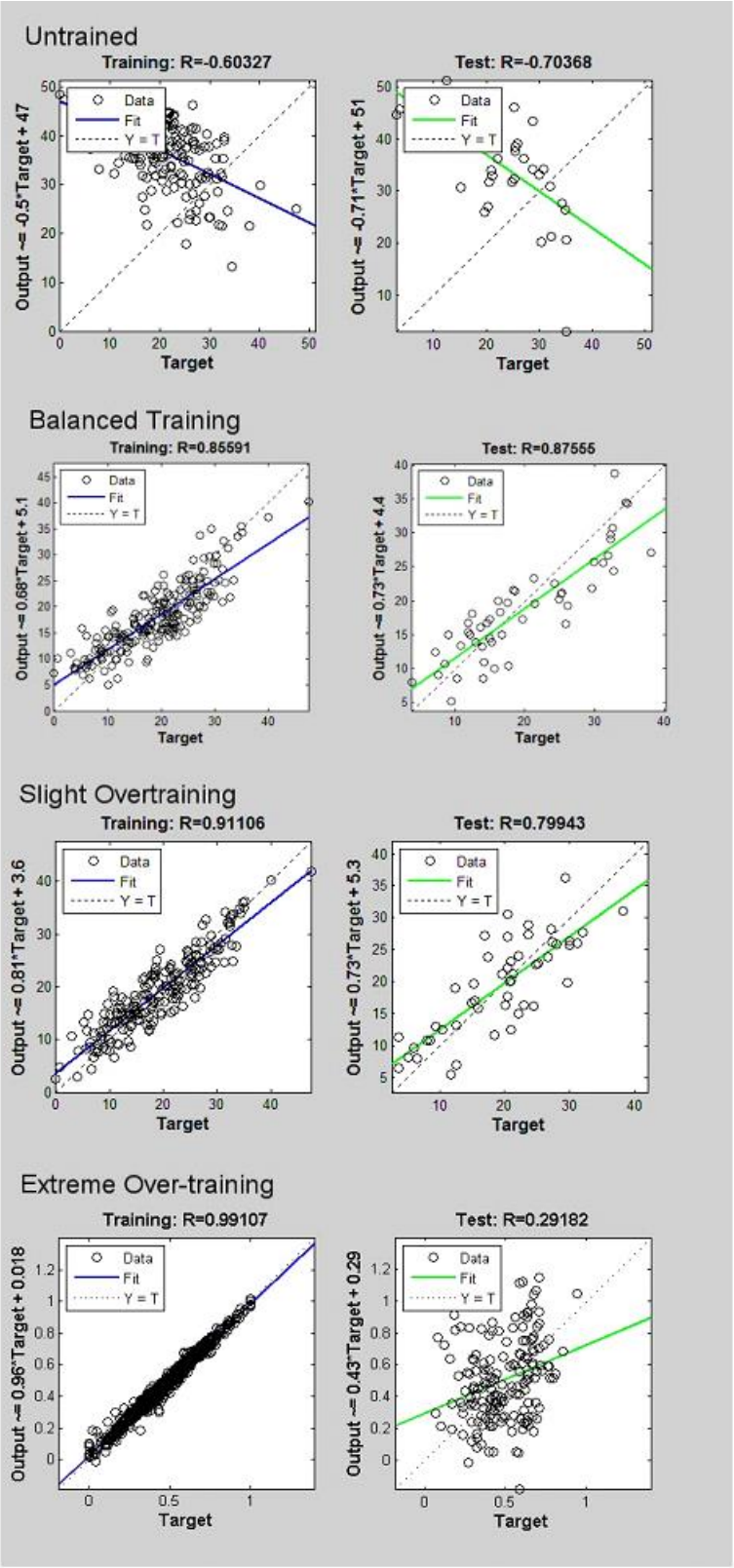


Figure 6 - Overtraining

The common Levenberg-Marquardt (LM) training function performs speedily with simple networks (less than one hundred weights) and becomes less competitive as network complexity increases. It excels at regression tasks (curve matching), but not categorisation tasks (pattern recognition). AAC networks are large enough that other learning functions might be worth considering.

Scaled Conjugate Grading (SCG) is much more suited to categorisation tasks and can handle networks with large numbers of weights, often as fast as LM functions can. SCG is very useful for analysing surveys.

Another commonly found learning function is Bayesian Regularisation (BR), which is excellent at dealing with data with small numbers of samples or stochastic noise (Burden & Winkler, 2008). During the development of the AACr, it was found that BR functions did tend to overtrain significantly, rectified by manually selecting the generalisation threshold.

Assessing a Predictive Function

Once trained, the NN (or GP) function is in a position to be "asked" to make predictions about hypothetical musical structures presented as inputs. Using a small sample of training data that was withheld from the entire training process, predictions of the NN can be compared to known results to calculate a Mean Squared Error (MSE). As the training data is scaled from 0 to 1, the test MSE of a successful network approaches 0, and an unsuccessful network approaches (or exceeds) 1. The R value is a measure of the correlation between the inputs and outputs. An R value of 1 suggests a direct relationship, where an R value of 0 means no relationship at all.

Some tweaking of architecture and different learning functions can result in big improvements. This work will demonstrate that it is possible, if the auditioning procedure and data treatment are carefully executed, to achieve a NN with MSE less than 37% of the standard deviation of training data, ranging from less than 0.1, to as low as 0.02, with unseen test predictions. Different affect measurements can also be trained in isolation to gauge their

correlation. R values for unseen data range from 0.04 for an abstract tag-word survey, to 0.38 for physiological bio feedback. It's not clear yet what the golden standard for "success" is in AAC NN training, nor can we know how well this translates to satisfactory musical composition; however this does provide some quantified assessment for further development of various aspects of AAC systems.

Further research investigating methods for measuring AAC success is anticipated (Williams et al., 2014, p.16, 18), meanwhile, MSE and R values are already common standards in assessing the performance of neural networks themselves and could be utilised in any AAC system where methods of regression or machine learning are employed.

Search Algorithms

The AACr NN functions can do hundreds of thousands of simulated auditions in seconds, but it is still not feasible to run through all the possible combinations of musical structure available for even a simple broad-spectrum AAC musical algorithm as outlined here. To find, for example, the ten top musical structure combinations most likely to induce sweaty hands, it should be assumed that either a certain time limit is involved, or that there are many good solutions. This limits how many musical structure combinations need to be auditioned in the search for a best solution. A search algorithm is employed here.

A 'brute force' approach uses random sample by sample (or in bulk) predictions, returning the best scoring examples found in the time available. This is a fairly reasonable approach given the low correlation values of these networks. There are more informed search algorithm approaches that use heuristics, genetic algorithms, or partial knowledge of the function to inform the selection of potential pieces, however it is not yet proven that the results could be any more reliable than brute force given the low correlation values expected in AAC applications, and the largeness of the search space.

There is a requirement for further investigation into the effectiveness of various search algorithms for use in AAC, however, such investigation is dependent on the definition of a good solution or a measurement of success.

3 Pilot Study: The Affective Algorithmic Composer

The Affective Algorithmic Composer (AACr) provides a working example of a complete AAC, demonstrating the methods outlined in *Developing Methods for AAC*. Each part of the AACr system is described:

- The graphical user interface
- *TwoHands* – the musical algorithm
- Data collection
- Predictive functions using neural networks
- Assessing the predictive functions

3.1 Overview

The AACr also directly addresses the aims of this project:

- Developing a broad-spectrum algorithm focusing on musical structure (not timbre) for use in an AAC.
- Developing and comparing methods for collecting both perceived and induced affect data.
- Establishing and implementing predictive functions from this data using neural networks.

AACr includes a generative musical algorithm, a biofeedback and survey audition interface, a predictive function in the form of a series of trained neural networks, and an interface that allows for real-time unsupervised music production and functions as an aide for a human composer.

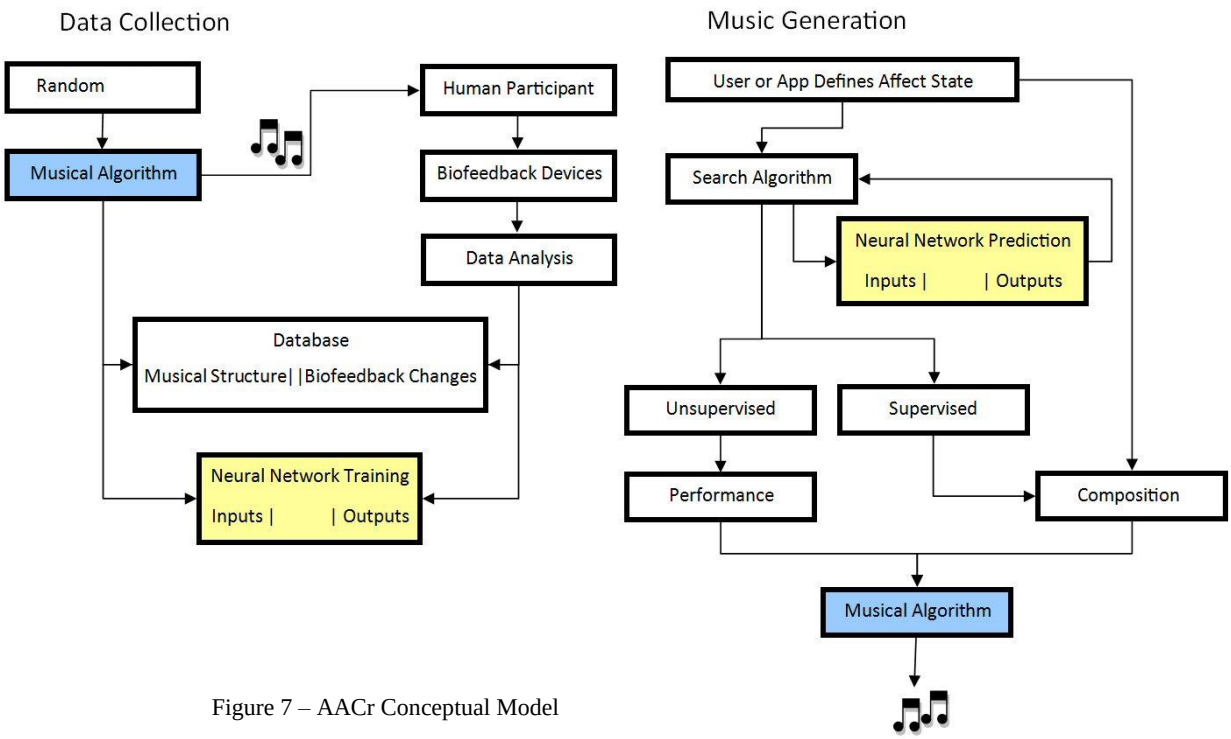


Figure 7 – AACr Conceptual Model

The AACr has two main functions, data collection and music generation, both referring to the same neural network and musical algorithm. The left side of Figure 7 outlines the basic logical process of data collection for the AACr.

The data collection task starts with the generation of random musical structures (in this case typical musical structures like scale, rhythm, chord progressions etc.) formatted as a string of variables or integers often referred to as seeds, structural arrays, or musical structures. This is sent to the music algorithm.

The algorithm receives a list of structural variables, decodes them into a musical passage that features those musical structures, and plays them to the human participant. The system then records induced biofeedback from the selected sensors, perceived affect data from survey results, and deposits them into a database alongside the related musical structural array. This database is later used to train several feed forward neural networks (one for biosensors, one for EEG, one for the Likert survey, and one for the tag survey). These neural networks, now trained to predict affect from musical structure, are used in the music generation task on the right side of Figure 7.

In the music generation task a target affective state is specified by either an external program, by using the biosensors, or manually. The neural networks immediately predict the affective values of every structural array in a pool of randomly selected passages as soon as the pool size is set. The search algorithm then only needs to score all the structural arrays relative to the specified target affective state. As the structural array pool is already all rated, this happens quite quickly and the top scoring seeds are quickly sent to the playlist ready for playback.

3.2 Graphical User Interface

The Graphical User Interface (GUI) for the AACr is written in Cycling 74's *Max MSP 5.0*. It is a graphical programming language frequently taught as a prelude to text programming in sonic arts courses, and it is efficient for establishing prototype user interfaces and audio

applications. It also has in-built compatibility with Ableton Live, Open Sound Control (OSC), SQLite, JavaScript, and serial communication with Arduino microprocessors; all of which are crucial in this project. Beyond this, in areas such as neural networking and the complex handling of arrays, Max is not ideal, and so this particular project uses several modules developed in different programming environments, including Supercollider and Matlab.

The GUI reflects a two sided approach to AAC, both the data collection, and the predictive playback, as seen in Figure 8 and Figure 9. On the left in Figure 8 is the 'Main Controls' section. Here the GUI allows the user to perform several key actions. The first thing that must be done before the AACr can play music is to connect the music algorithm script to the interface via OSC. This is done automatically when the algorithm script is loaded in *Supercollider*. When the music algorithm is connected it will automatically light up the 'Algorithm OSC' light. The user can also use the GUI to dictate the MIDI device to be used for play-back by the algorithm, and reset the algorithm in the rare case of an error.

Data Collection

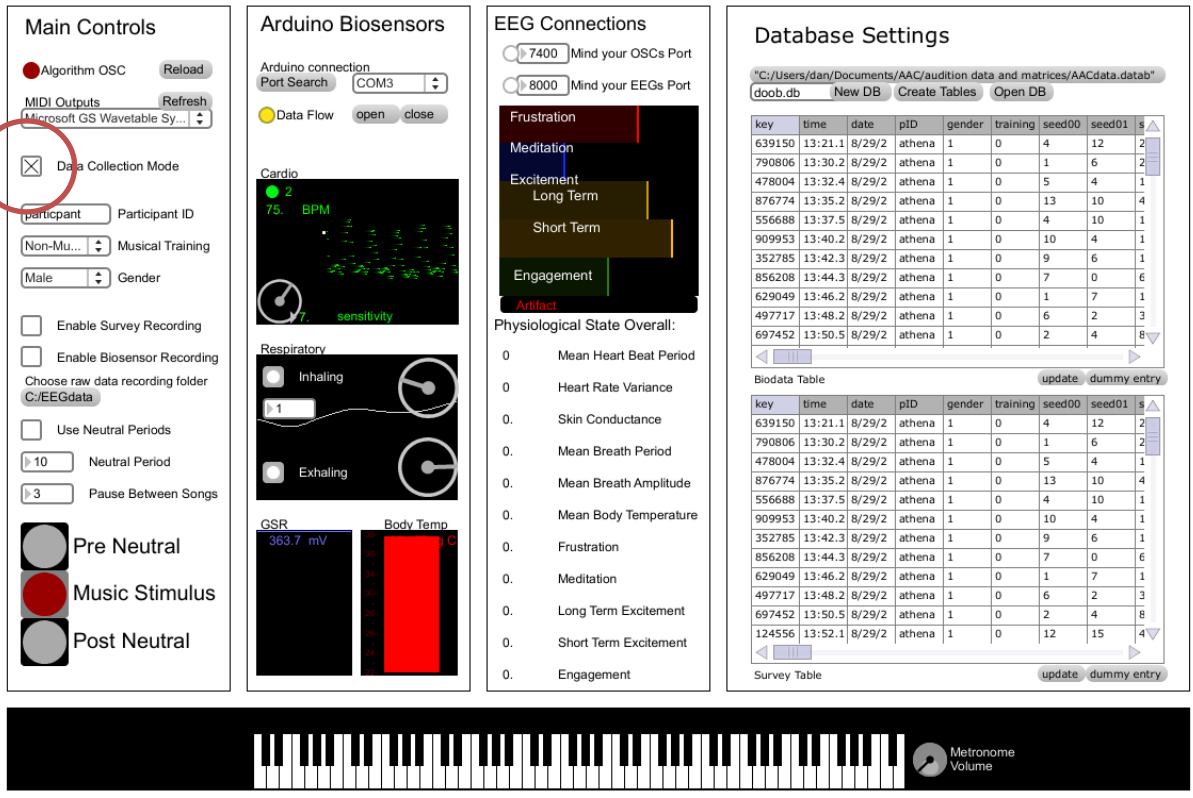


Figure 8 - The AACr GUI, in Data Collection mode.

The main controls section enables the user to enter basic participant information for the database, enable or disable the execution of biofeedback and survey recording, change which drive or folder raw biosensor data is to be stored in, or dictate the automation of neutral periods and pauses before and after music playback. It also allows the user to switch from data collection to playback control mode highlighted by the red circle in Figure 8 and Figure 9.

On the right hand side can be seen the ‘Database Setting’ panel. This panel allows the user to create new databases, or load an existing database. It uses a functional but limited native implementation of the SQLite database environment via JavaScript. Each database has two tables, one for ‘Biodata’, and one for ‘Survey’ data. If the ‘Enable Biodata Recording’ and ‘Enable Survey Recording’ boxes are checked, and a database has been initialised correctly, then every time a musical piece is played, affect data will be recorded during playback, neutral periods, and finally the survey will pop up after the passage has completed. This entry will contain the time, date, key, participant ID, musical training, gender, the musical structural array played (seed00, seed01 etc.) and the feedback values gathered. Raw time stamped data from the sensors is also recorded into separate files named according to participant, key, and sensor name. Although not used for this project, this raw data will prove useful for more in-depth analysis at a later date.

The ‘Arduino Biosensors’ panel is focussed on controlling the biofeedback sensors, which are implemented via the Arduino microprocessor. Once the Arduino is physically connected, the user can use the port search button and slide down menu to allocate which serial port is being used by the Arduino and open or close the flow of data from the Arduino to the GUI.

Below this are graphical analyses of the four biosensors. There are various controls to calibrate the heart rate monitor and thresholds of the breath counter to ensure the best possible results are being recorded at all times. The Galvanic Skin Response (GSR) and body temperature sensors are more robust and don’t depend on complex calibration.

The ‘EEG Connections’ panel has the controls for interfacing with the Emotiv *Epoc* EEG headset. From the top it shows the port selection slide down menus that have yellow activity monitors so that the user can monitor the connection. The EEG connects to the interface via OSC on two separate occasions. The first is via a free community written application called *Mind Your OSCs* by (bitrayne, 2013) that sends pre-analysed results from the *Affectiv Suite* directly to the interface via OSC, which are recorded to the database. The second occasion is via a similar app called *Mind Your EEGs* or *Mind Your OSCs 2* also by ‘bitrayne’. This app sends raw EEG data from the individual electrodes to be recorded and time stamped, similar to the raw data recording of the biosensors. A record of artefact events as dictated by Emotiv’s automated analyses is also recorded. The OSC ports can be selected in the GUI although the connection itself is initiated by these apps.

Once connected, a graphical display shows the affective values being read from the participant as well as an overall readout of the total physiological changes of the participant including the Arduino biosensors.

At the bottom of the GUI is a piano-like keyboard that reflects the note being played by the algorithm in blue for left-hand and red for right-hand. There is also a metronome volume control. It should be noted that the MIDI notes do not get generated by the interface itself but from the algorithm module. There is also an alarm that flashes at the bottom left of the keyboard section if ‘Enable Biorecording’ is enabled but no signal flow is detected.

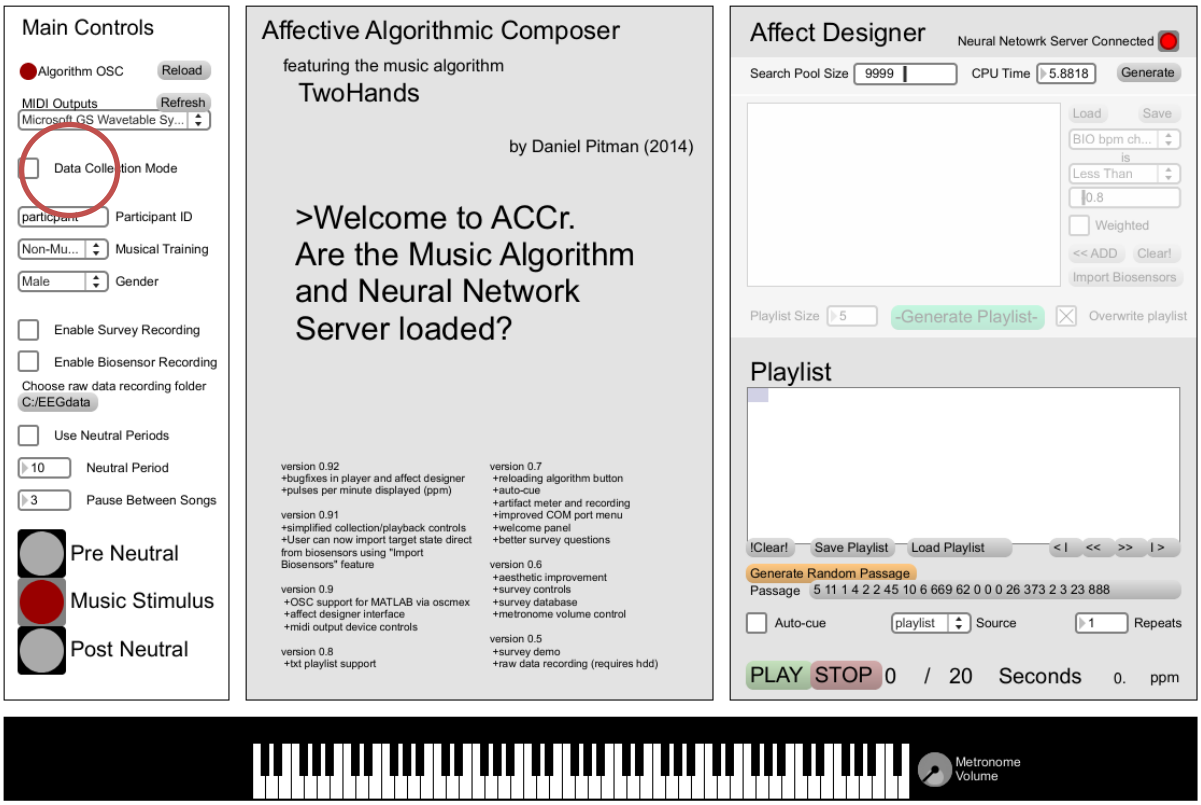


Figure 9 - The AACr GUI in Playback Mode

Figure 10 - Defining Target Affect Features

Playback Mode

An information panel and the ‘Affect Designer/Playlist’ control panel hide the database and sensor settings when the ‘Data Collection Mode’ setting is unchecked (circled in red in Figure 8 and Figure 9). The information panel includes a changing text display that describes the current action, version, and a list of updates from previous versions. This panel also serves as a handy cover to hide the biosensor and EEG feedback to avoid distracting the participant.

The ‘Playlist’ panel features the controls for sending random or predefined structural arrays to the musical algorithm, starting or stopping playback, loading and saving playlists, and automating looping and playlist progression. Playlists are the foremost composition tool in the AACr and can be constructed by the user or generated by the NN’s from the ‘Affect Designer’ panel. They are simply text files; listing structural arrays in the order they should be played.

The ‘Affect Designer’ takes a form similar to a shopping cart interface as demonstrated by (Ando, 2011). The user selects biological or survey paradigms, Boolean operators, and thresholds to describe the affective changes required. Once a defined target state is received, the neural network server can return a playlist of passages intended to invoke those same changes.

First, as with the other features, the OSC link must be initiated from the Neural Network Server app (the AACr’s predictive function) that contains the pre-trained NNs. The ‘Search Pool’ setting defines how many structural arrays are auditioned by the Neural Network Server to create the results with a fairly conservative estimated CPU time provided alongside. It’s estimated that 999999 structural arrays can be auditioned in about 588.23 seconds on a below average computer[9] so care must be taken to ensure that pool sizes are balanced against any real-time playing constraints.

9 This is a somewhat arbitrary calculation, but demonstrates the need to consider NN calculation times in realtime music generation.

It's important to specify that the user outlines required affective state changes rather than static states. For example, a user might specify 'Skin Conductance' and 'is greater than' and '0.3' as a feature[10] along with 'TAG hammer' as a feature (tag words don't have a Boolean component) and so-on until a list of affect requirements is complete. Optionally, by pressing the 'Import Biosensors' button, a brief analysis of the current state using only the biosensors and EEG can be used to create a target affective state, allowing the user to generate music reflecting the current bio-sensor wearer. As seen in Figure 10, paradigms are shown in one of four categories, BIO sensors, EEG, LIKert survey, and TAG word survey.

With a pool size and target state defined, the 'Generate Playlist' button causes the Neural Network Server to return those passages as a playlist which best matched the target, with the top scoring passages first in the list.

The interface can simply start playing through the list as provided, or the user is able to save these playlists as text files and recombine them as they see fit. Text playlists can be reloaded into the interface for playback or recording. New lists can also be generated while the current piece is playing. This allows for a fairly flexible system that can generate affective music on the fly, generate random musical passages (used for data collection), or act as a compositional aide.

3.3 Two Hands – The Music Algorithm

This project only deals with musical structure and not timbre. While timbre certainly has been shown to be an active part of musical affect (Klügel & Groh, 2013; Patil et al., 2012; Pitman, 2012; Schlemmer, Kulke, Kuchinke, & Van Der Meer, 2005) the algorithmic implementation of broad-spectrum timbre requires a whole new plethora of variables and considerations. To focus on musical structure the piano is adopted here as the main instrument; its broad pallet of

10 Read: 'Skin Conductance should increase by more than 0.3'. For induced affect, the emphasis is on change, as the current state of the listener is assumed to be unknowable, and regardless, inconsistent. This is not the case with all affect data.

expression is useful, but importantly, its position in western culture as something of a ‘vanilla’ timbre provides as close to an unbiased standard timbre as is perhaps possible.

For this project, the idea of only using notes that can be played within the reach of the two hands of a piano player provided a natural limitation and inspiration for the name *TwoHands.* It was developed using the open source *Supercollider* sound programming language by (James Mcartney). *TwoHands* accepts structural arrays with 40 variables.[11]

Figure 12 provides a brief overview of the variables and their functions within the algorithm.

TwoHands was not developed to be a cutting edge music algorithm on its own, and uses quite simple generative techniques to achieve its result with the minimum number of variables. When used with randomly generated variables it can output anything from incoherent hammering, to the sweetest four part lullaby. As one might assume, the total output often tends towards largely uninteresting and often awkward passages with the occasional “diamond in the rough”, hence the need to use well trained neural networks to navigate through the many possibilities.[12]

Receiving a structural array (or seed) via OSC triggers the algorithm to decode that seed and begin MIDI playback. Some other basic transport commands are provided for such as ‘play, stop, pause, reset’ and a function to reload the scripts in the rare case of an error.

I	IV	VI	I
VII	V	II	III
V	VI	IV	VII
I	II	III	V

Figure 11 - The ‘Chord Board’

Harmonic progression is dictated using a matrix of predefined chords, referred to as the “chord board” (see Figure 11). The chord board is designed so that any chord can be reached by any other chord in one move (including moving over the edges to the

11 The reader should note that 40 part structural arrays (or seeds) are quite difficult to wield within MAX and OSC, and for the sake of convenience are often compacted to 20 digits using binary or ternary modulus functions. When used in neural network training in Matlab, seeds must be expanded to 81 integers, as categorical values are expanded to become a number of binary options. Regardless, the algorithm produces reliable reproductions of pieces (albeit with minor aesthetic changes) whenever the same seed is given.

12 Again, this idea of value or usefulness is in the context of musical applications where affectivity is paramount as discuss on p. 1. Incoherent hammering is just as likely to be useful in a film/media/gaming context as any four part lullaby, so long as it produces the affective qualities desired.

opposite side). It features two of every diatonic chord, and three of the tonic and dominant. Passages define a number of turns, moving in a chess piece fashion, dictated by several different styles of movement: a directional preference (up, left etc.), preference for a numerical difference from one chord to the next (2, 4, 6 etc.), a preference for odds over even (or vice versa), and a preference for or against repeating chords. At the end of the passage, potential cadence possibilities are given slightly higher considerations as well.

Rhythms are described using a total number of pulses and two divisors (or factors) with probability environments that dictate which divisors are most likely to feature key rhythmic points and dynamics. A scale is selected from a list, and each hand is assigned a position and a mechanical mode by which to expand on the basic progression (chord playing, arpeggiating, melody, or bass line playing etc.). Each hand interprets the melody and rhythm in different ways according to their mode of operation and their own probability environments. The resulting sequence is then played out via the chosen MIDI device, intended for a real or virtual piano.

The algorithm has distinct hierarchical phases. At the highest order the harmonic progression is changed, which then requires that all the lower order attributes be recalculated. The next highest is the rhythm, which then requires only the hand executions to be recalculated. Finally, the execution of the hands can be recalculated without redefining the higher orders. This is reflected in the organisation of the structural array itself. At the time of writing, this feature hasn’t been established in the AACr interface module and currently structural arrays are triggered from scratch each time. The latent hierarchical isolation features of the algorithm are intended to be used for greater composition structures in future developments.

The algorithm is capable of 2.40734712102912e+25 possible outcomes, which have an average length of 30 seconds each. Even for this most straightforward of algorithms, to listen to all possible outcomes would take 32,715,632,760 (over 32 billion) planets with Earth’s

population 100 years of non-stop listening. This may be more people than will ever exist, a primary motivation for using AACr to help navigate these possibilities.

Anatomy of a Structural Array	
structural array in 20 part Max format	[5, 1, 1, 8, 1, 7, 45, 2, 8, 426, 652, 2, 0, 2, 26, 050, 1, 2, 10, 662]
same structural array in 40 part format	[1, 1, 1, 2, 0, 1, 2, 0, 0, 1, 1, 2, 0, 5, 4, 2, 8, 7, 3, 2, 2, 5, 6, 2, 0, 2, 2, 2, 2, 0, 5, 0, 1, 2, 1, 0, 1, 2, 6, 6]
Harmonic Variables	
~startpoint = [1, 1];	Matrix coordinates define the starting point on the chord board (Figure 11) and thus, the first chord.
~inertia= [1, 2];	Defines number of chords selected for first and second half of the passage (not including the start)
~vector=[0, 1];	Defines preferred direction (on the "chord board") for the first and second half is direction is the primary method. (up, down, left etc.)
~style=[2, 0];	Defines the preferred difference from one chord to the next for the first and second half.
~preferOdd=[0, 1];	Binary, defines a preference for odd or even chords for each half
~primarymethod= [1, 2, 0];	Defines which of the methods (vector, style, or preferOdd) is given precedence or ignored in considering the next chord.
~repeatness= [5, 4];	Increase or decrease preference for using chords that have already featured (first and second half).
~scale = 2;	Dictates a scale, (0 = ionian, 1 = dorian, 2 = phrygian, 3 = lydian and so on). Scales include all modes, harmonic minor, pentatonic, symmetrical scales, and a scale of octaves.
~tonic = 8;	Dictates the note to be tonic, as semitones above or below C

Rhythmic Variables	
`~rhythm = [7, 3, 2];`	Defines universal rhythmic characteristics, pulses per bar, two divisors (or factors) that denote key rhythmic and dynamic points. In this example there would be (7x3) 21 pulses per bar, with an underlying emphasis on every third and second beat.
`~probenv = [2, 5, 6];`	Define probabilities used to generate dynamics and rhythmic results.
`~runTime = 2;`	Total runtime is ~runtime*4+20 second (in this example 28 seconds), that in turn dictates pulse length.
Hand positioning and behavioural variables (right and left hands)	
`~lmode = 0;` `~rmode = 1;`	Operational mode: selected from basic chord playing, melodic, bass line melodic, and various kinds of arpeggiator. Here the left hand is playing basic chords, and the right a generated melody.
`~lpos = 0;` `~rpos = 2;`	~lpos defines how many octaves from the left the left hand is, and ~rpos defines how many octaves the right hand is up from the left hand. Here the left hand is at the lowest octave, and the right hand two octaves above it.
`~lrhythm = [2, 2, 2];` `~rrhythm = [1, 0, 1];`	Takes the universal rhythm characteristics and rearranges them to create new sub-rhythms. The numbers refer to the index of the universal rhythms. So the right hand is actually playing 3/7:3 and the left 2/2:2
`~lprob =[0, 5, 0];` `~rprob = [2, 6, 6];`	A probability environment, used in discerning rhythmic, dynamic, and harmonic variation in the execution of hand operations.

Figure 12 - The Structural Array in Detail

3.4 Data Collection

To collect and analyse EEG data, the Emotiv *Epoc* EEG headset was used alongside a battery of simple biosensors. The Arduino microprocessor was chosen as a platform to host four biosensors: heart rate and variance (HR, HRV), respiration rate and size (BR, BS), skin conductance level (SCL), and body temperature (BT).

For perceived data, two surveys methods were chosen. The first was a Likert survey that asks musically oriented but subjective questions, and the second part was a tag-word selection survey. Results from these sensors and surveys were recorded along with the structural array of the musical stimulus played. As well as this, the interface also recorded a unique key number for each sample, time and date, the ID of the participant (using non-identifying codenames), the gender of the participant, the musical training of the participant, and a raw recording of each of the biosensors, EEG electrodes, and EEG artefacts for future analysis/redundancy.

The sensors used in this project were chosen primarily for their simplicity in implementation, known relevance to musical affect, and to suit the project's scope and budget (see Collecting Affect Data p23).

EEG

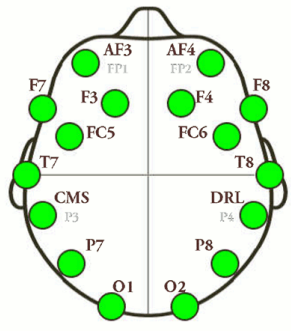


Figure 13 - Epoc Electrodes

Emotiv's *Epoc* uses interchangeable, saline soaked, passive electrodes and is easily positioned on the participants head without shaving hair, glues, or other impractical processes. Electrodes are placed at AF3, AF4, F3, F4, FC5, FC6, F7, F8, T7, T8, P7, P8, O1, and O2. The headset is connected wirelessly to the computer that presents both raw and analysed data through a control panel. A graphical guide helps the user to ensure each electrode has an appropriate connection with the scalp.

The software comes with three built in forms of analysis. The artefact, facial expression, and eye movement analysis is called ‘Expressiv Suite’. The infamously fickle brain control interface system, ‘Congnitiv Suite’, relies on training the system to recognise certain brain states for use as commands. Finally,‘Affectiv Suite’ delivers a reasonable but proprietary emotion/mood affect analysis in the paradigms of meditation, frustration, engagement, long and short term excitement(Lang, 2012), the main focus for this project.

Affectiv Suite provides a surprisingly well-implemented overview of induced affect (“legitimising affective suite,”). These analysis technologies are unfortunately totally proprietary. More open approaches to deriving emotional content from physiological sensors are still in development (Miranda, 2010).

The results of these analyses are exported via Open Sound Control (OSC) using two community made apps, *Mind your OSCs* that deals with pre-analysed data and *Mind Your EEGs* that deals with raw electrode data (bitrayne, 2013). OSC ports are selected and connections are simply initiated using these apps’ interfaces.

Biofeedback

For this project, a battery of four biosensors was built around an Arduino microprocessor. The Arduino firmware runs a simple script that measures each of the data pins (one for each sensor), and reports the values to the interface. The Arduino doesn’t perform any of the analyses itself, but sends raw data direct from the sensors to the interface via a USB serial connection. This serial connection is initiated in the interface.

A small prototype shield was developed that contains connections for the heart beat sensor, thermometer component, and simple resistance loop circuits for the skin conductance electrodes and stretch/breath sensor.

The heart beat sensor is a simple open source product called the *PulseSensorAmped* by Joel Murphy and Yury Gitman (Murphy & Gitman, 2014). It uses a simple light dependant diode and a bright green LED, placed against the finger or earlobe, to sense colour fluctuations

caused by oxygenation of the blood as the heart pumps. It can be attached directly to the Arduino. A simple signal analysis in the GUI provides cardio rate and variance. Should period between the beats be more irregular than an acceptable threshold, the cardio sensor interface will become visibly dimmed and brighten again once regular heartbeats have been detected.

Similarly, the temperature monitor is a single component called the TMP36. It requires 2.5-5.5v to operate and the output voltage is linear to the temperature in degrees Celsius, making analysis of the signal as simple as it is to connect the component straight up to the Arduino.

Respiration rate is measured using a stretch sensor around the participant's rib cage. It was found that using rubber all the way around the participant's chest resulted in a very small range, where as a solid strap held together with 10-15 cm of rubber improved range considerably. Less length would result in the too much tension so the rubber did not perform well. It has been useful to confirm that the sensor is calibrated well by counting participant breaths by sight, comparing this to the interfaces count, and adjusting the sensitivity accordingly. Adjusting the length of the rubber section can also improve the accuracy in some cases.

The skin conductance sensor was perhaps the most complicated sensor to implement for this project. Several versions were built using various materials. For the electrodes, aluminium proved to be less than ideal and having the contact wrap around the finger induced sweating. Eventually two steel/nickel coated snap-buttons were found suitable.

Stiff connection wires proved to be too prone to bumping the various sensors and distrubing their signal. These were replaced by braided wire with soft rubber as the isolation material.

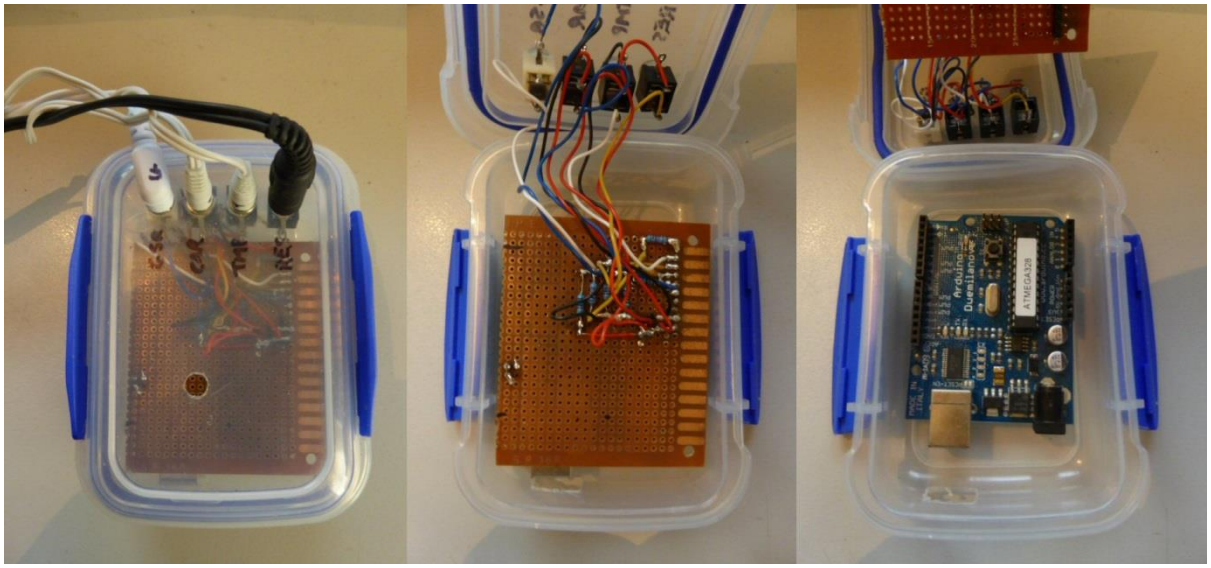


Figure 16 - Left: Biosensor connections. Centre: Revealing the prototype shield. Right: The underlying Arduino microprocessor

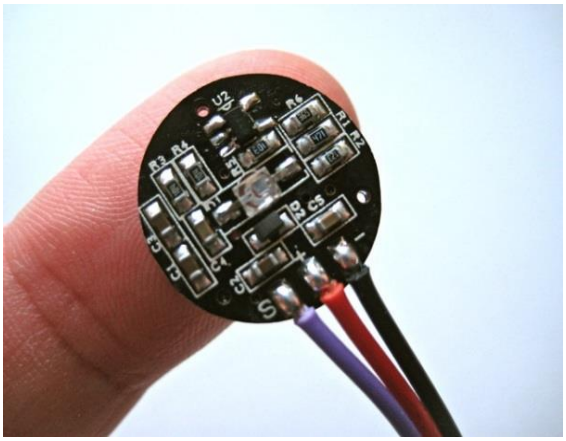


Figure 14 - A Simple Open Source Heart Monitor for Arduino (Murphy and Gitman)

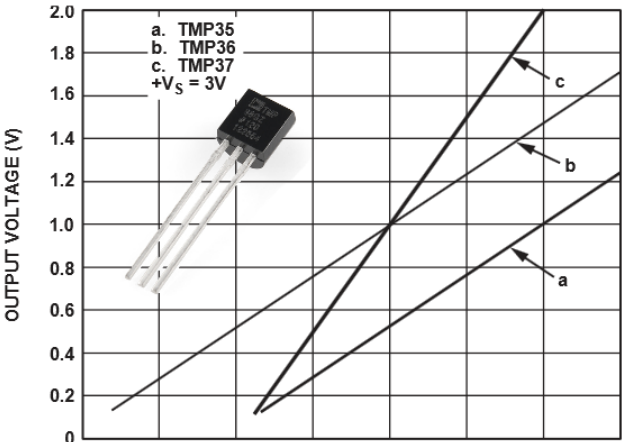


Figure 15 - TMP36 Voltage to temperature response ("TMP36 Datasheet" 5)

Figure 17 - An ideal length for the stretch sensor

Survey

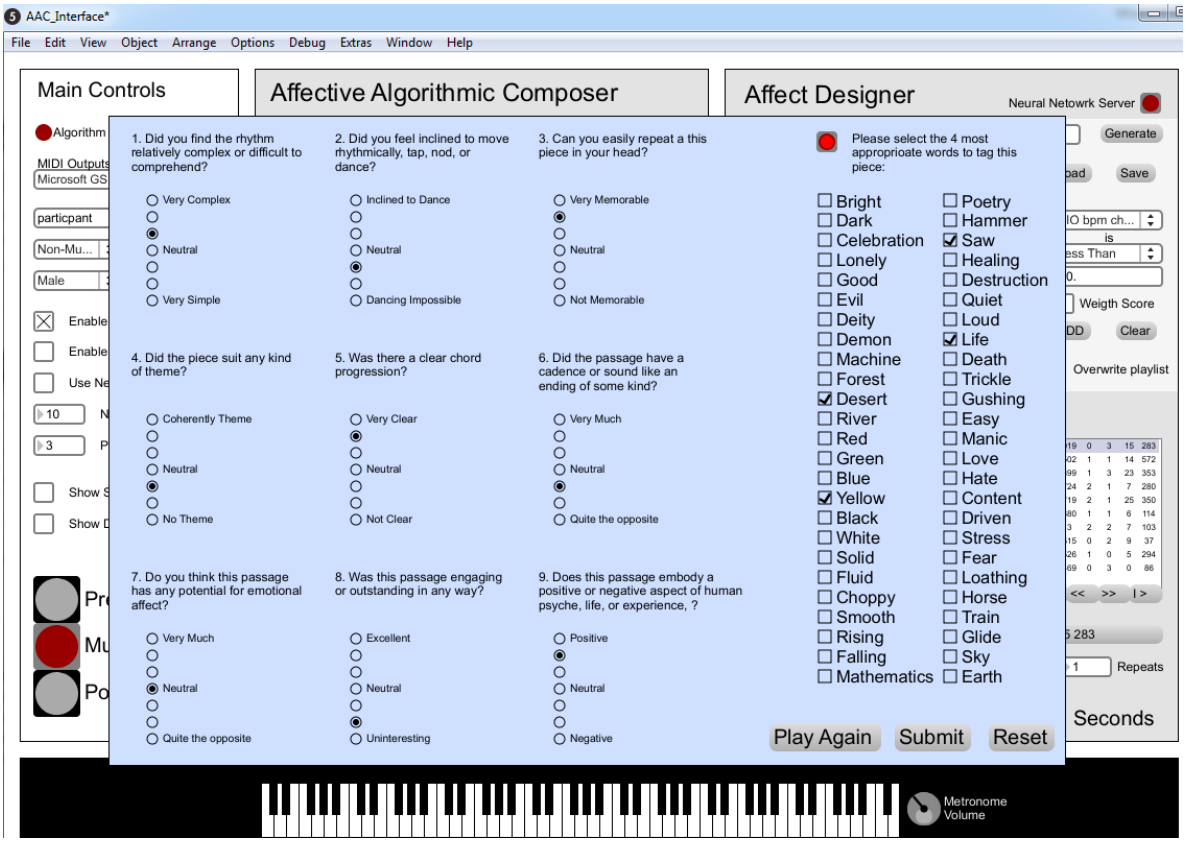


Figure 18 - The AACr survey

The survey is configured to come into view on the GUI after the musical stimulus and neutral periods have completed, as long as the 'Enable Survey Recording' check box is enabled in 'Main Controls'. The survey used in this project has two sections.

The first was nine questions on the subjective qualities of the music using Likert scales. These questions were designed to become tools that composing users might find useful such as feeling inclined to nod or dance, a potential for thematicism, a catchy phrase, or a positive or negative emotion. The second part of the survey required participants to choose four out of fifty possible tag words. The words featured a series of opposites such as bright and dark, a selection of colours (very popular with some participants), and several directly and indirectly descriptive words such as hammer, horse, fluid, and choppy.

Whilst not dealing strictly with affect, it was intended that users using the affect designer could use some of these tag words to help find a passage to suit a particular theme or environment.

As a point of interest, some participants found the first section quite straightforward to complete, but often that the second section, with tag words, was quite difficult or even stressful to complete. This might relate to the level of mental abstraction, which may have contributed to the higher predictive error that the predictive function reported for the second survey.

Participants

Volunteers were recruited from various locations around the city of Adelaide using flyers. The main ethical consideration in AAC data collection is participant anonymity. In line with the ethical requirements outlined by the University of Adelaide, each volunteer was given a participant information sheet, consent form, and completed a brief meeting with the researcher regarding expectations. No reward was made available to volunteers apart from remuneration of travel expenses, and each volunteer chose a non-identifying nickname for the purposes of the project.

There were several volunteers who were unable to participate due to medical reasons. One volunteer, diagnosed with hypomania, was prone to swinging from extremely energetic and alert to tired and sleepy, and would likely be unable to provide reliable affect data over time. A second volunteer suffered hypertension, a condition that can involve unexplained changes in a biological state, particularly heart rate, and would also render a participant unable to provide reasonable affect data. Eleven volunteers did participate and auditioned a total of three hundred and seventy five passages.

Typically each participant would attend for around forty five minutes to one hour per day, once or twice per week, with short breaks as required. Most participants reported considerable mental fatigue after thirty to forty minutes. There was a broad variation in volunteer attendance, some visiting once, others five or more times.

Participants were seated facing a blank wall, alone, and with as few distractions available as possible. Once sensors and EEG equipment were fitted and calibrated by the researcher,

participants were often quietly monitored until comfortable with the activity and then often left unsupervised for up to 45 minutes without a break.

The activity itself, largely automated, involves several periods. *preNeutral*: a silent neutral period of ten seconds during which biosensors measurements labelled ‘preNeutral’ are taken. *music1, music2, music3*: the collective period of one random musical stimulus from the algorithm (during which three biosensor measurements are taken labelled ‘music1’ through ‘music 3’). *postNeutral*: a final neutral period of ten seconds.

Data Treatment

The data is recorded into a simple SQLite database implementation native to Max 5.0, which is far more reliable than using ‘col’ objects. SQLite features were implemented by Cycling’ 74 for the developers more than users, and the feature was not originally documented. It is accessible using a simple JavaScript object and a script to report the contents of the database to a 'jit.cellblock' object for viewing.[13]

Not all the measurements taken are necessarily suited for training of the NN, and many outliers needed to be excluded. Several errors rose from hardware problems, participant behaviour, physiology problems, or software bugs.

One participant was discovered to have an incredibly low and arrhythmic heart rate and it was later confirmed that a doctor had already diagnosed arrhythmia. The biosensor data for that particular participant had to be excluded (although the survey data remained). Other samples featuring measurements that were extremely unrealistic indicated probable equipment failure and were excluded. At one point, many of the measurements taken for “music1” and “music2” periods were erroneous due to a coding mistake that was soon fixed, however

13 For more information regarding this lesser known but powerful SQL feature see C74: Data Collection: Building Databases Using SQLite https://cycling74.com/2008/09/05/data-collection-building-databases-using-sqlite/

“music1” and “music2” are not used in training NNs. Overall 90 samples were excluded from biosensor data and 33 from EEG data. All the survey data remained valid.

The biosensor and EEG data was defined by the difference between “music3” and “preNeutral” periods.

3.5 Predictive Functions using Neural Networks

There are four neural networks trained using Matlab, each focussing on a different aspect (biofeedback, EEG, Likert survey, and Tag word survey) of the data collected. This was a process of some experimentation in order to find the optimal architecture for each data type (see Training of a Neural Network p45). Best results included using a single hidden layer, and a straightforward feed forward architecture. This is a very simple implementation compared to the most recent developments in machine learning, but for this project, simple NN architectures prove effective when handled correctly and do lie within the scope of available expertise.

Using a simple script for interfacing with the GUI via OSC, and the four neural networks as predictive functions, a Neural Network Server app (NNS) was created. It is intended that, as a standalone app, the NNS can be used with the AACr GUI or could potentially work with other applications via Open Sound Control protocol (OSC).

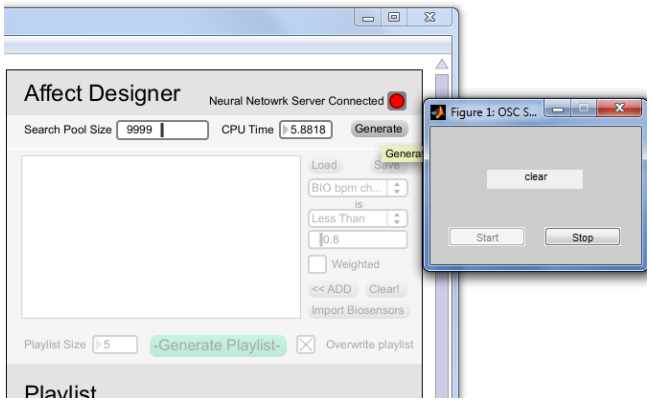


Figure 19 - The Neural Network Server

Once the executable is run, clicking the ‘start’ button will cause the NNS application to connect itself to the GUI via a predefined port in OSC (see Figure 19). When a pool size is defined in the interface, the NNS creates an array of the defined number of random seeds. Each seed is immediately given a predicted score by the neural networks for each value of affect. This can take some time initially, however when an affective target is defined by the user and sent to the neural network sever app, the seeds that score best, according to the affective target, can be returned to the interface almost immediately. A one-at-a-time approach to analysing passages was considered, as opposed to this bulk system, but in real time music generation tasks it became quite complex to manage time limitations and so the bulk system was preferred.

OSC functionality is provided using *oscmex*, an open source library of functions for Matlab that enable sending and receiving of OSC data (“oscmex,” 2012).

Neural Network Limitations

The NNs currently only consider the affect data, and not the other consequential data of gender, ID, and musical training, despite being included in the auditioning process. If implemented, this might involve a kind of “handshake” format where the listener can volunteer information about themselves in order to improve the results from the NNs.

Further, the NN training tools are as of yet able to be compiled, so a solution that allows real-time learning has not yet been developed. It is certainly a worthwhile investigation as it would allow unwanted pieces to be identified and ignored earlier in the training process, and potentially improve both the efficiency of the training process and the experience for the participants. Real-time learning would also allow long-term training programs that could build on the training progress made to date. Future projects may benefit from using a server/client architecture via the internet, allowing more broadly and recently trained NNs and other predictive systems to be available to users from the one client interface.

3.6 Assessing the Predictive Functions

The chart in Figure 20 shows architecture specifications and test results for each of the four neural networks created: biosensors (BIO), the electroencephalogram (EEG), the Likert part of the survey (LIK), and the tag part of the survey (TAG). The ‘Method’ column describes the number of hidden nodes and the learning function used, noticeably different for the TAG data.

Of all the samples provided for training the networks, around 20% were randomly isolated for use in testing the networks for success. The results shown in blue in Figure 20 represent the regression R value (R) and mean squared error (MSE) of the training data at the generalisation threshold, and the pink columns show the results for the test data at the same threshold. Although MSE makes for a useful description of success, it can be easily misinterpreted as a small error if the data only has a relatively small deviation in the first place. The last column, ‘MSE%’, shows the MSE as a percentage of standard deviation. A value of 100% would suggest that there is as much error as there is deviation in the predictions, a fairly hopeless scenario. A value of 0% suggests there is no error at all, which would be ideal if not very unlikely. The results of 30.61%, 31.76 and 36.33% for BIO, LIK, and EEG respectively are very promising, and exceeded expectations given the small number of samples available.

The larger MSE% of 72.01% for TAG data deserves some discussion. The TAG implementation was intended to be one of the most useful aspects of the ‘Affect Designer’. It has been considered that the high level of mental abstraction (where any individual is able to interpret the tag words differently) may have contributed to the high level of prediction error, or that perhaps the number of words available is too high for such a small sample to be able to extrapolate on.

In Figure 21, the blue line represents the NN’s number of correct guesses and the red line shows the number of times that participants chose to use that word. It’s clear that tags used

Per-Data Comparison for Neural Network Training								
				TRAINING DATA		TEST DATA		
DATA	SAMPLES	METHOD	ST DEVIANCE	R	MSE	R	MSE	MSE%
BIO	285*	Pattern Fitting: bayesian regulation(40 hidden nodes)	3.016E-01	6.278E-01	6.812E-02	3.815E-01	9.233E-02	30.61473477
LIKERT	375	Pattern Fitting: bayesian regulation(40 hidden nodes)	2.837E-01	7.245E-01	4.106E-02	3.082E-01	9.010E-02	31.76111523
EEG	342	Pattern Fitting: bayesian regulation(40 hidden nodes)	5.891E-02	4.505E-01	1.499E-02	8.327E-02	2.140E-02	36.32653405
TAG**	375	Categorisation: scaled conjugate gradient (200 nodes)	1.151E-01	3.363E-01	6.540E-02	4.884E-02	8.285E-02	72.00852307

Regression R Values measure the correlation between outputs and targets. An R value of 1 means a close relationship, 0 a random relationship.

Mean Squared Error (MSE)is the average squared difference between outputs and targets. Lower values are better. Zero means no error.

MSE% is the Mean Squared Error expressed as a percentage of the standard deviation of the samples. Small MSE's can be misleading if the typical range of the data is not considered, so a smaller MSE% is a better indiciation.

Test data is a reserved percentage (10-20%) of available samples selected randomly, and totally isolated from training. This gives a more practical idea of the neural networks effectiveness to predict data.

*due to an unfortunate software bug, only 145 samples were available for respiratory measurements.

** TAG neural network correctly tagged around 23.47% of the samples. So even though using an MSE is not particularly relevant for describing error in classifying, it is reasonable to state the the TAG neural network is wrong 76.53% of the time, which is comparable to the MSE% of 72.01%

Figure 20 - NN architecture and results

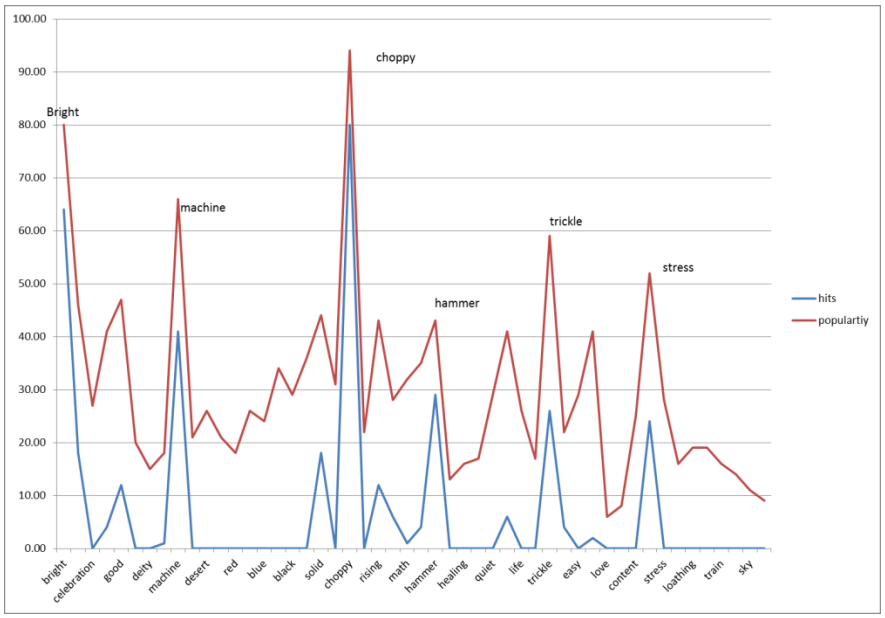


Figure 21 - Predictability vs. Popularity in TAG data

more often by participants are more successfully predicted. It might then be reasonable to assume that the error percentage of 72.01% might be improved upon greatly should less tag words be provided in the survey (increasing the popularity of those remaining), and/or more samples be provided (although this can be taken for granted in most neural network applications).

3.7 Interpreting Results

When a playlist is returned, the Neural Network Server (NNS) has given what it deems to be the most likely passages of music to achieve the target affect state change desired by the user, as structural arrays or seeds. Each array represents a variable (or set of variables) that is critical to musical generation in the *Two Hands* algorithm.

Affective Target	**Playlist Generated**
0, "BIO Bpm Change Close to 0.301284 1 / 1 "; 1, "BIO Breath Rate Close to 0.301284 1 / 1 "; 2, "BIO Skin Cndct Close to 0.301284 1 / 1 "; 3, "EEG Frustration Less Than -0.336023 1 / 1 "; 4, "EEG Meditation Less Than 0.301495 1 / 1 "; 5, "EEG Engagemnt Close to 0.332305 1 / 1 "; 6, "LIK Rhthm Complex Less Than 0.332305 1 / 1 "; 7, "LIK Outstanding Greater Than 0.200201 1 / 1 "; 8, "LIK Positivity Close to -0.460319 1 / 1 "; 9, "TAG machine Is Tagged -0.460319 1 / 1 "; 11, "TAG stress Is Tagged -0.460319 1 / 1 ";	1, 11 4 06 5 03 16 17 00 03 396 386 5 1 1 16 254 2 0 09 111; 2, 15 3 05 5 02 13 13 08 12 885 276 4 2 5 17 421 0 3 11 452; 3, 11 2 06 3 15 05 05 12 00 852 686 1 2 0 15 459 1 1 00 442; 4, 14 2 01 0 00 18 47 14 05 870 710 5 1 2 13 524 1 3 23 305; 5, 11 8 09 1 01 13 83 10 06 852 481 1 2 5 10 238 0 0 26 802; 6, 09 1 06 3 02 10 55 09 05 844 551 4 0 5 05 115 0 0 21 964; 7, 11 2 07 5 00 13 24 06 07 995 038 1 0 1 17 715 1 0 04 683; 8, 11 2 15 5 01 10 08 04 04 797 011 1 2 0 21 660 2 2 26 177;

Figure 22 - Affective Target and Resulting Playlist

Some rudimentary analysis can be done at a glance in Figure 22. The majority of playlists in this list have 11 as their first value, which denotes a VII chord (not specifically major, minor, or diminished) as the first chord to be played. Several entries have 2 as their second value, denoting the number of chords in the harmony progression, which would be quite short here.

Some analysis can also be accomplished using the post reports in Supercollider[14] as the algorithm decodes the arrays. For the example playlist from Figure 22, Supercollider shows that the use of scales is varying (eighth column), but the method by which chords are selected is tending towards preferring a consistent number of pitches between chords (sixth column). Rhythmically the system seems to have preferred using varying time signatures but with consistent emphasis on the 3 beat (8xx values are common in the ninth column). For a detailed description regarding decoding seeds, see Figure 12 - The Structural Array in Detail.

Further analysis requires playing back the passages audibly using the algorithm. This can be done conveniently using the "load" and "save" features of the GUI (audio files are also supplied on the digital accompaniment).

The examples provided (appendix C-E) are intended to demonstrate the potential for the AACr's key intended uses as a real-time unsupervised musical agent, receiving affective state instructions from media or biosensor wearers and providing immediate music to reinforce or counter that state (appendix C and D), and as a composer's aid, rapidly producing suggestions for static but affective compositions (appendix E).

For the most part the neural networks seem to respond more satisfactorily to lengthier affective designs, however, it is problematic to claim success from the various musical demonstration examples included. To claim success in this context would be fairly naive, and when listening to passages alongside media counterparts, notions of affective success may be potentially open to suggestion (perhaps most apparent in the *Metropolis* demonstrations).

The demonstrations do show that these features are functioning as intended, however the problem of stating to what degree they function remains.

14 The software environtment that the musical algorithm was coded in. Most programming enviornments use a post window to show status, debugging information, and present information from 'postln' or 'print' commands.

3.8 Defining Success

A thorough investigation of potential methods for defining AAC success is required in the field for comparison and refinement of future systems. An investigation into the success of the AACr as a musical system in regards to a any human audience is the immediate intended direction for further research. There are many factors to be considered, including the complications of induced affect, which are often focused on mechanics that are easily influenced by external factors. Where logistically possible, it might prove helpful to analyse the audience with biosensors, however in such an environment uncontrollable external influences would likely confound results beyond worth.

Often, even with identical architectures, NNs can develop a potentially infinite number of different functions for such complex stochastic data. It may be found that from two identical NN models, one might be more pleasing than the other. This –somewhat ironic- individuality on the computer's behalf may need formal assessment as well. Once a success level is defined, the "sample size/success" ratio might identify the most efficient number of training samples for future AAC endeavours.

In order to define a method for measuring success, there are several potential methods to consider. A Turing test is unsuited at this point, as the music algorithm inherently suffers from simplicity (see Musical Algorithm Considerations p22) such that even when being successfully affective, the AACr's music is consistently recognisable.

A perceived affect study may be simply implemented, for example, where volunteers knowing the target affective state listen to two corresponding playlists; one generated randomly and one generated from NN predictions. How often the volunteers guess correctly which is the affect driven playlist could determine a measure of success.

4 Conclusion

AAC systems have an enormous potential to provide a method to help us navigate the unthinkably massive realm of possibilities presented by even the most simple of musical algorithms, without the use of stylistic limitations or mathematical filters that restrict the algorithm in overly simplified ways that amount to human inference. These systems essentially do this by attempting to model or imitate the human audience in some way, which provides a context to filter (not create) potential musical passages. This is no small feat, but failing to systematically assess the effectiveness of these systems on an audience outside of the training participants, as is the current trend in the field (Williams et al., 2014, p. 18), cannot advance the field past its initial speculations. A clear direction for future research will be to determine if success can be defined, and if so, how this success relates to predictive error measurement.

The aims of this project have been successfully achieved, and many new directions for improvement have arisen in the process: A broad-spectrum algorithm focusing on musical structure (without timbre) has been developed, the development and comparison of multiple methods for collecting both perceived and induced affect data was successful, a predictive function trained from this data was developed and implemented, and music was successfully generated, targeting specified affective states.

If we rate the AACr using Williams' system of categorisation, the AACr is unique in satisfying all of Williams' categories: It is capable of compositional and performative music making, it does use an entirely generative algorithm process, it is capable of both unsupervised real-time and compositional aide applications, it is able to adapt to input directly from biofeedback in real-time, and it does utilise both perceived and induced (and is among the first entirely generative AAC systems to do so).

The musical algorithm, *Two Hands*, has proven to be adequate for the job at hand, abut has outlined some issues as a result of the limited numbers of variables variables. The forced

simplistic nature of the algorithm causes a consistently recognisable quality throughout the passages generated, which brings into question the “breadth of its spectrum” (see p58). This is not to say the algorithm isn’t capable of appropriate results, and this may also be an issue to consider in future survey design as well, but some further development in implementing simple musical structures is anticipated.

There are several bugs, which are still present in the *TwoHands* system. Most interesting of these ‘bugs’ is silence. During training the music algorithm was unintentionally left capable of producing tracks with one or zero notes, leaving the participants to quite frequently experience musical silence. Whilst silence is perfectly acceptable, the participants likely did not always engage with these silences musically or were not “free to enter into the act of listening” (Cage, 1961); frustrated that the machine might not be working, that the silences were too frequent and annoying, or that they were left to be distracted by their own thoughts and surroundings. It cannot be proven either way but it seems likely that as a result, silent passages may well be used by the AACr as a strategy to achieve affective targets featuring high frustration. Rather than removing these silences, perhaps a more enlightened approach would be to reduce their frequency, and brief the participants of the relevance of silence in modern music.

Other bugs that were not addressed (despite great lengths being taken) before training took place include the algorithm’s ability to exceed the limitations of pianos and player in pitch and tempo under certain circumstances. It is impossible to rectify these bugs in the musical algorithm in hindsight, without voiding the extensive training and predictions of the neural network.

The physiological measurements from the biosensor data have not been translated into a typical emotional model (with the exception of the EEG data), but the AACr demonstrates that this is not actually necessary. The predictive error of the (non-EEG) biosensors was lower than all other groups, thus the AACr is currently able to define induced physiological target states with less predictive error than any of the other methods for predicting affect. This

suggests some interesting investigations are due where categorical emotional models are currently in use, such as the arousal and valence models commonly found in AAC projects to date.

Not only do the relatively simple neural networks used here provide the predictive abilities that drive the generation of new affective music, they also provide one quantifiable method of defining success in terms of the system's ability to predict affect given musical structure. By adhering to simple measurement principles, addressing as many external factors as possible during auditioning, and using multiple simultaneous approaches for affect data collection, the AACr has improved hundred-fold over its predecessor (Pitman, 2012) in terms of predictive error despite only using a similar number of training samples. It should be expected that much more can yet be done to improve these predictions.

The neural network's prediction error also allows for a comparison of the multiple approaches to collecting affect data, including relative successes (for EEG, Biosensors, and the Likert survey) and relative failures (in the case of the TAG word survey). This method can provide a unique opportunity in AAC for guiding the refinement of the individual methods in future incarnations.

New breakthroughs in machine learning systems, such as deep learning neural networks and support vector networks, may have massive ramifications for AAC development, and will be a considerable focus for future research as well.

In closing, this method of AAC development emphasises a clear principle: a focus on prediction statistics allowing for assessment and refinement of each individual component. Other methods of assessment are still required, but the AACr pilot can already provide a working example of the most important aspects of predicting affect in algorithmic composition. Future research on more complex biosensor analysis, machine learning systems, and broad-spectrum algorithmic implementations will no doubt lead to the development of AAC systems capable of more frequent applications in film, online media, and gaming,

especially where quantity, adaptivity, or constant originality, are concerned, as well as medical applications such as assisting with biological function control, and many aspects of musical therapy.

Appendices

Appendix A- AACr Code Examples

The digital accompaniment includes all the commented code and patches for the GUI, music algorithm, Arduino, and neural network server required to run the AACr system. Standalone versions of the GUI and neural network server are provided as conveniences, which only require free runtime environments (see below) to use.

Useful links to coding environments and runtimes:

Supercollider (3.5.6 or newer)

http://supercollider.github.io/download.html

Cycling 74 Max 5.19 Runtime (Mac or Windows)

https://cycling74.com/downloads/older/

Matlab Runtime (2012a or newer)

http://au.mathworks.com/products/compiler/mcr/

Mind Your OSCs (Free)

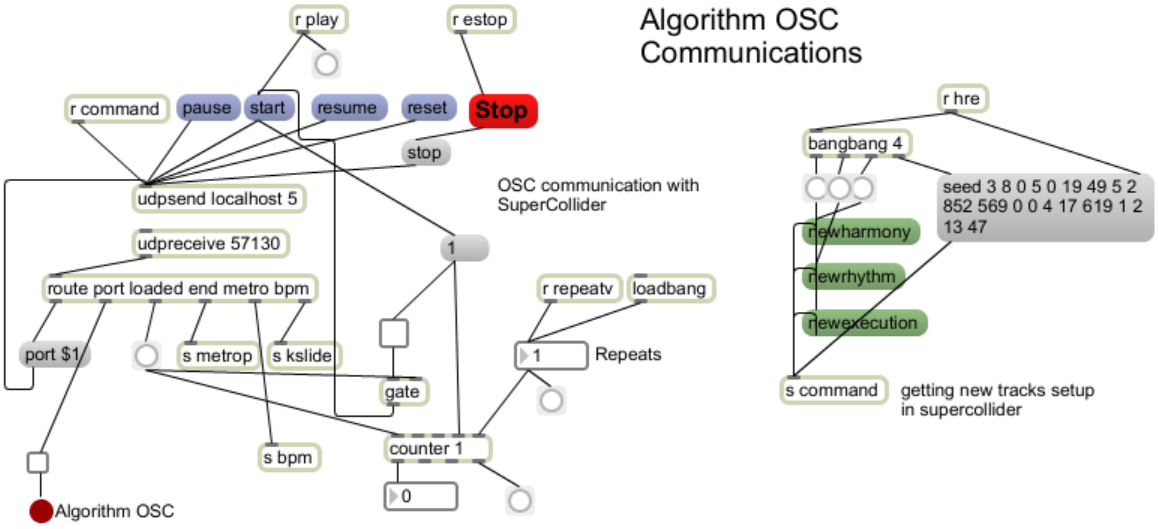
https://emotiv.com/store/product_85.html

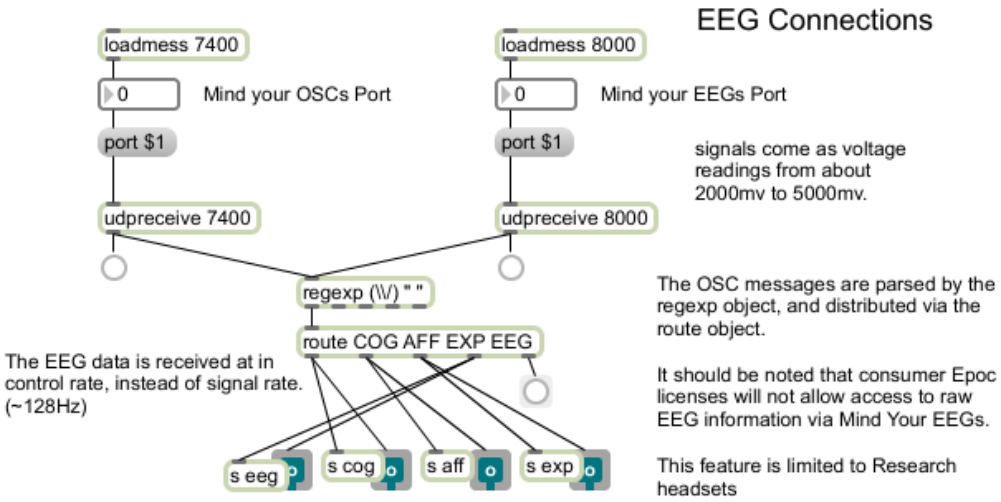
Arduino 1.6.3

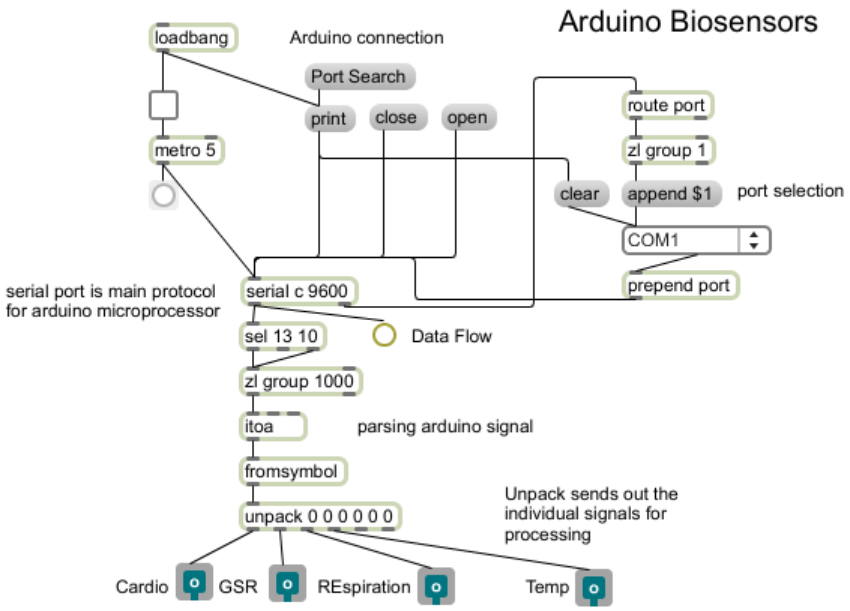
http://arduino.cc/en/Main/Software

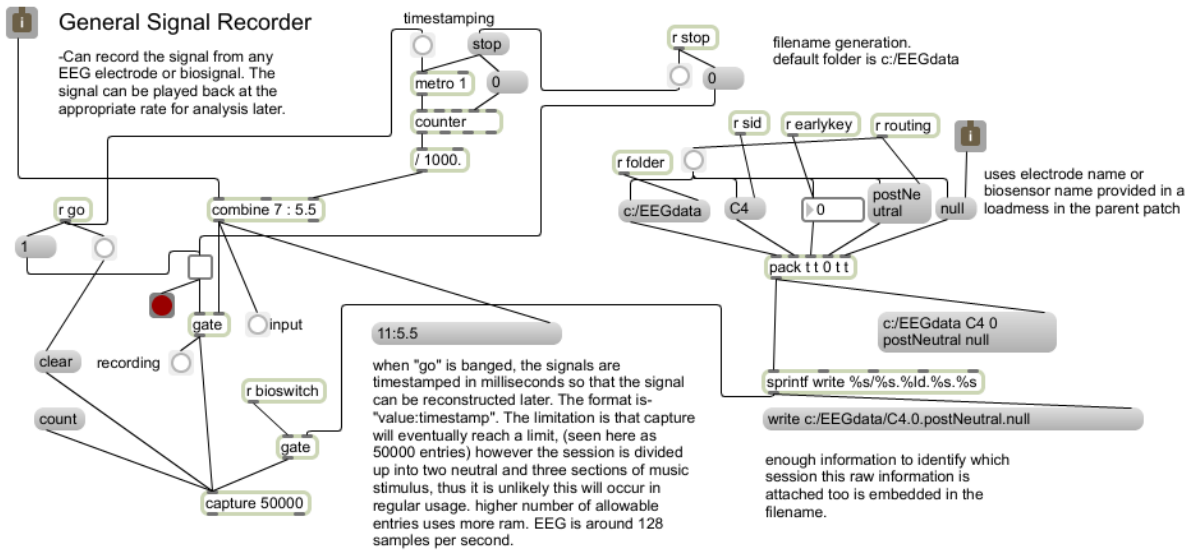
GUI Code Examples

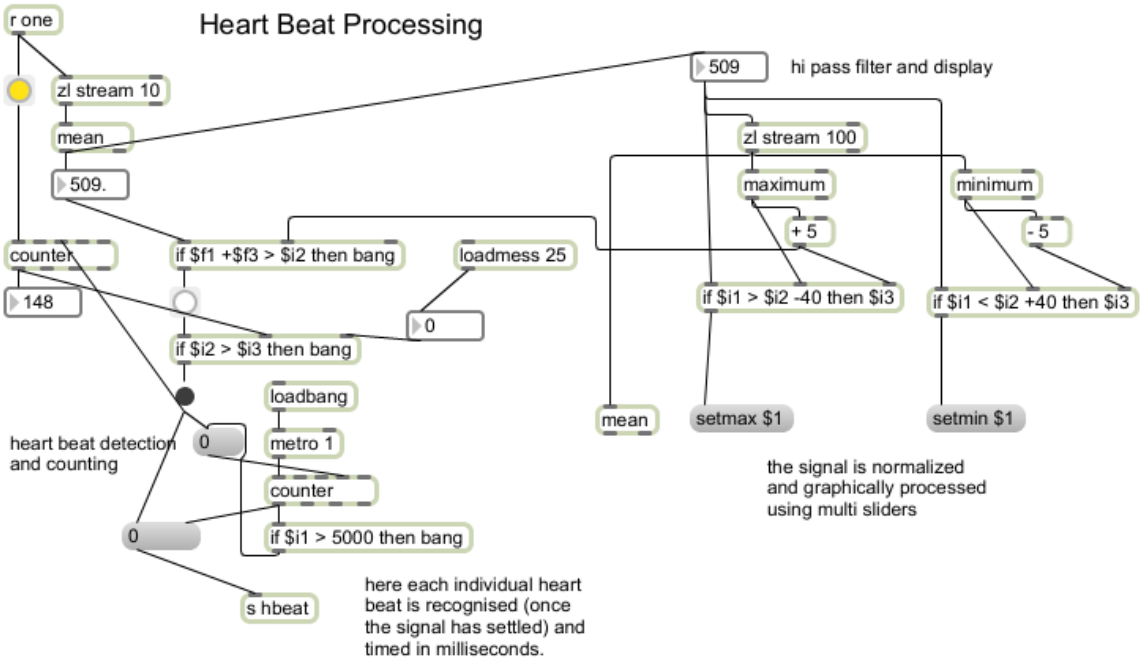
It is impractical to display all of the max patching in a readable way here on paper, thus important or novel aspects have been isolated and demonstrated here. To review the patching in full, refer to the digital accompaniment.

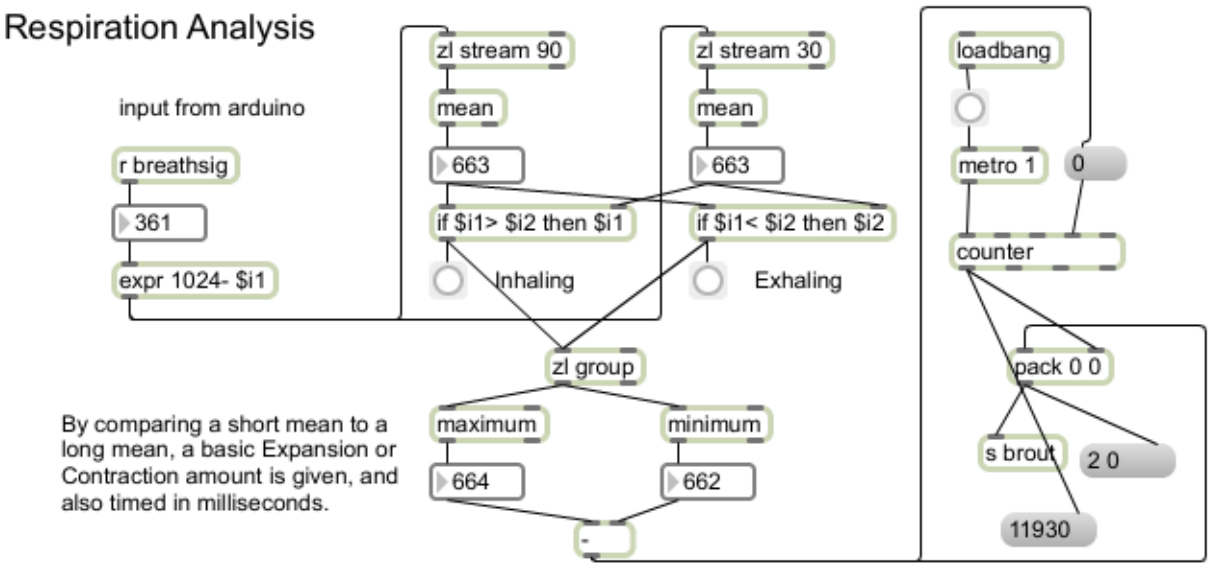


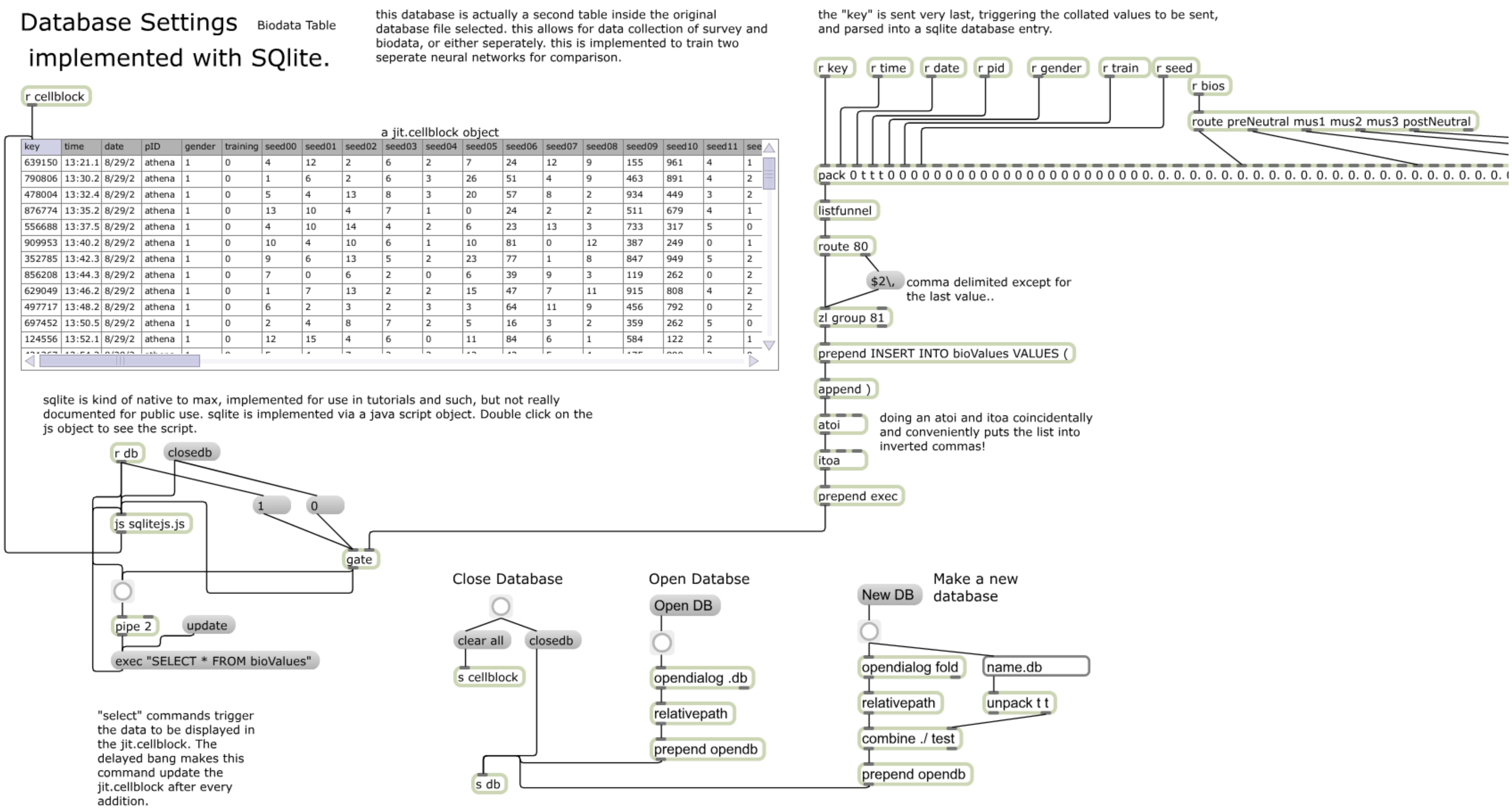



Arduino Biosensors
loadbang
Arduino connection
Port Search
print
close
open
route port
metro 5
zl group 1
clear
append $1
port selection
COM1
prepend port
serial port is main protocol for arduino microprocessor
serial c 9600
sel 13 10
Data Flow
zl group 1000
itoa
parsing arduino signal
fromsymbol
unpack 0 0 0 0 0 0
Unpack sends out the individual signals for processing
Cardio
GSR
REspiration
Temp


General Signal Recorder
-Can record the signal from any EEG electrode or biosignal. The signal can be played back at the appropriate rate for analysis later.
timestamping
stop
metro 1
0
counter
/ 1000.
r stop
0
filename generation.
default folder is c:/EEGdata
r sid
r earlykey
r routing
r folder
c:/EEGdata
C4
0
postNeutral
null
uses electrode name or biosensor name provided in a loadmess in the parent patch
pack t t 0 t t
r go
combine 7 : 5.5
1
gate
input
clear
recording
r bioswitch
count
gate
capture 50000
11:5.5
when "go" is banged, the signals are timestamped in milliseconds so that the signal can be reconstructed later. The format is- "value:timestamp". The limitation is that capture will eventually reach a limit, (seen here as 50000 entries) however the session is divided up into two neutral and three sections of music stimulus, thus it is unlikely this will occur in regular usage. higher number of allowable entries uses more ram. EEG is around 128 samples per second.
c:/EEGdata C4 0 postNeutral null
sprintf write %s/%s.%ld.%s.%s
write c:/EEGdata/C4.0.postNeutral.null
enough information to identify which session this raw information is attached too is embedded in the filename.

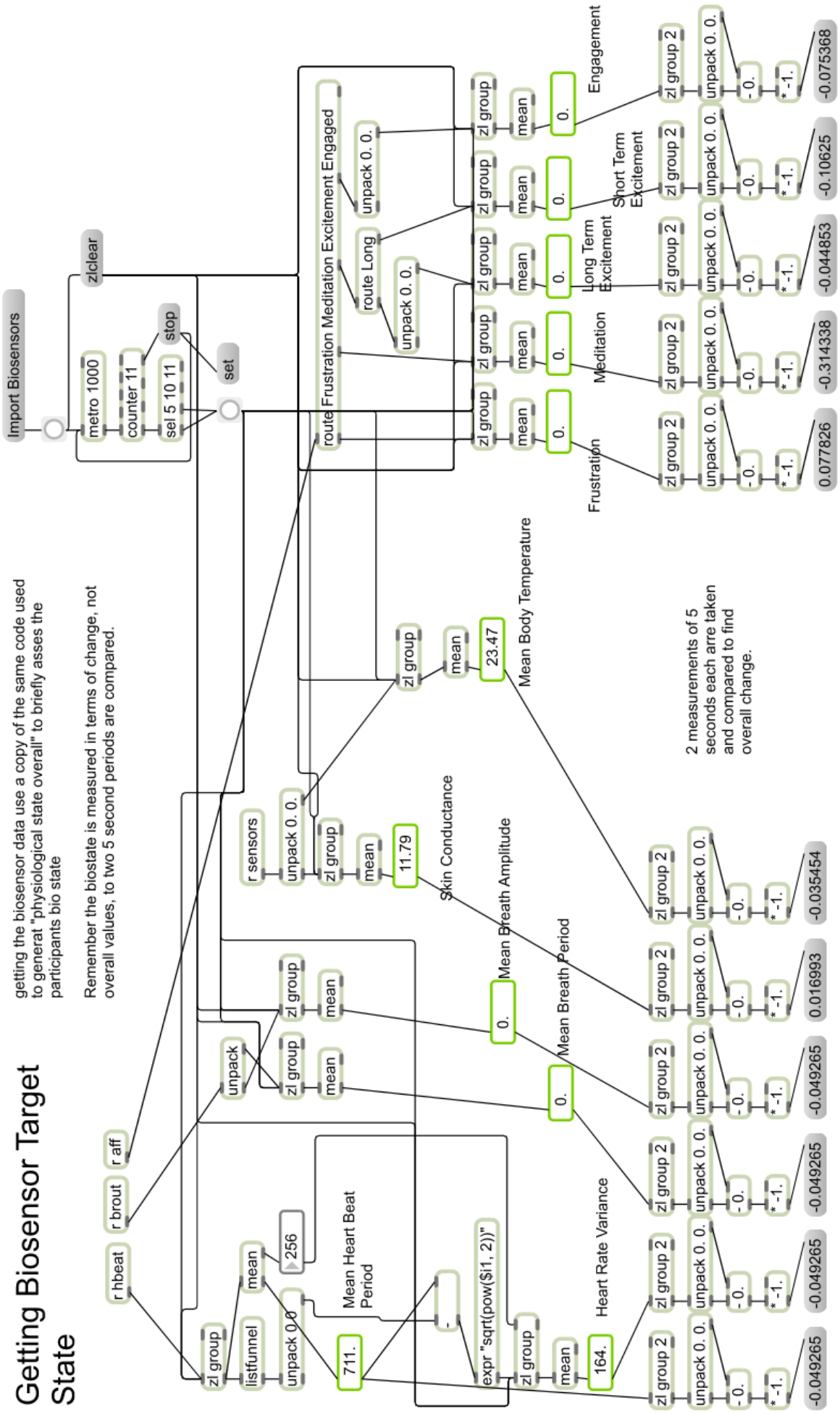

Heart Beat Processing
r one
zl stream 10
mean
509.
509
hi pass filter and display
zl stream 100
maximum
minimum
+ 5
- 5
counter
148
if $f1 +$f3 > $i2 then bang
loadmess 25
0
if $i2 > $i3 then bang
if $i1 > $i2 -40 then $i3
if $i1 < $i2 +40 then $i3
loadbang
heart beat detection and counting
0
metro 1
counter
0
if $i1 > 5000 then bang
s hbeat
mean
setmax $1
setmin $1
the signal is normalized and graphically processed using multi sliders
here each individual heart beat is recognised (once the signal has settled) and timed in milliseconds.

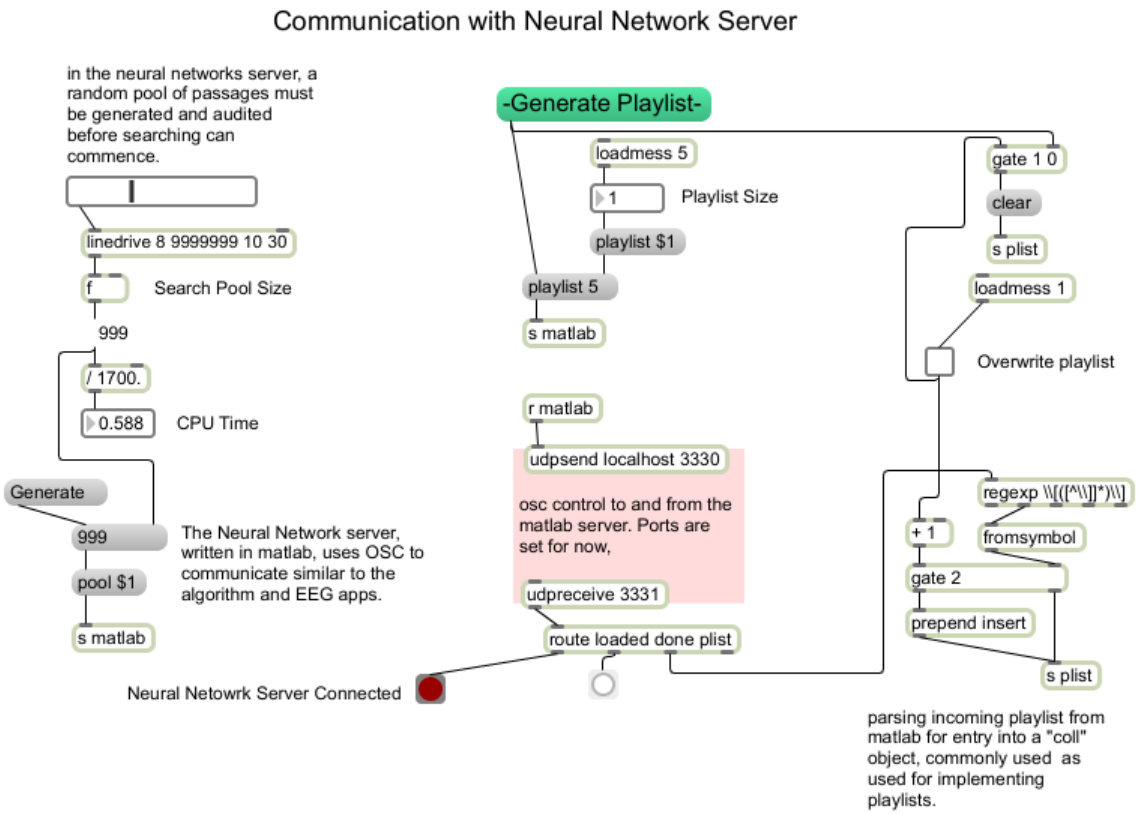

Respiration Analysis
input from arduino
r breathsig
361
expr 1024- $i1
zl stream 90
mean
663
if $i1> $i2 then $i1
Inhaling
zl stream 30
mean
663
if $i1< $i2 then $i2
Exhaling
zl group
maximum
664
minimum
662
-
By comparing a short mean to a long mean, a basic Expansion or Contraction amount is given, and also timed in milliseconds.
loadbang
metro 1
0
counter
pack 0 0
s brout
2 0
11930

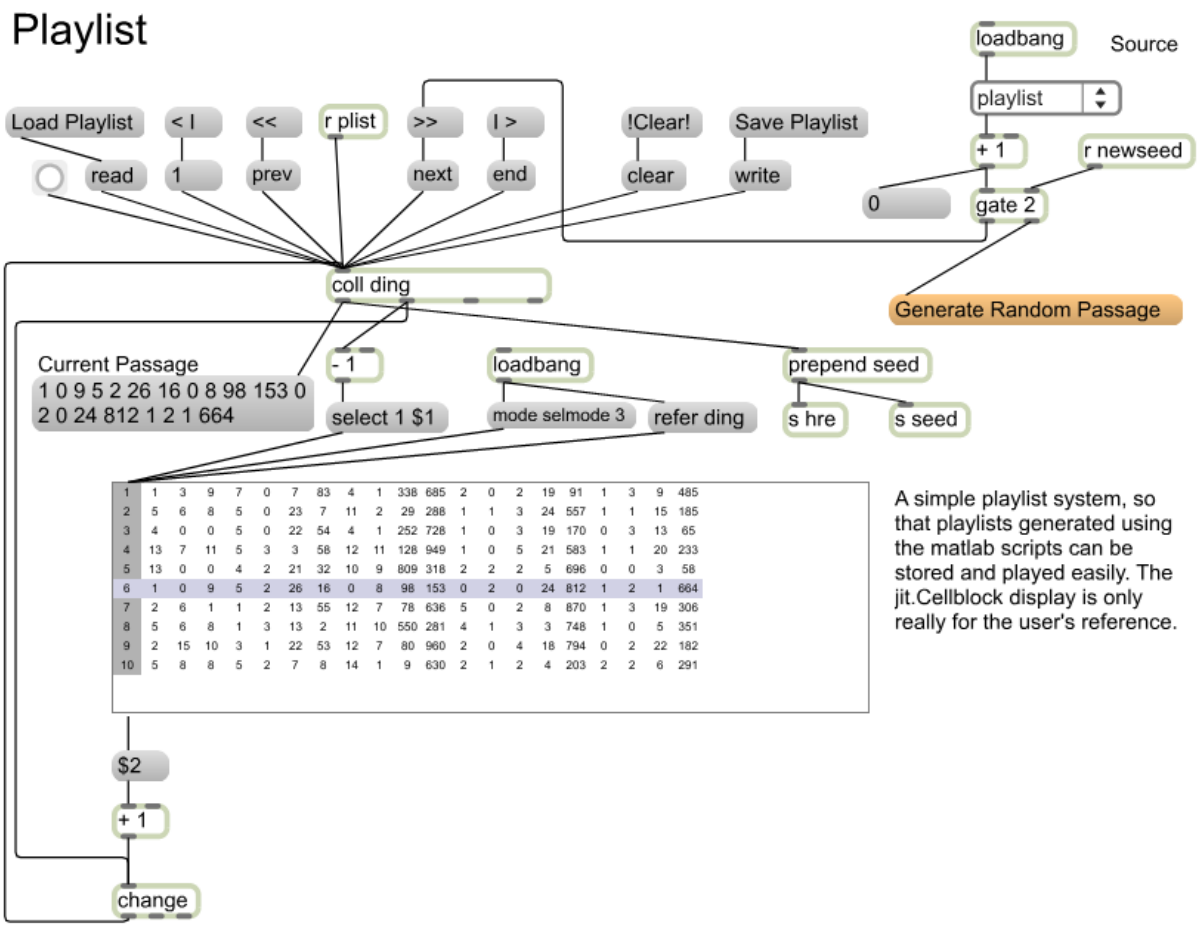

Database Settings implemented with SQlite.
Biodata Table
this database is actually a second table inside the original database file selected. this allows for data collection of survey and biodata, or either seperately. this is implemented to train two seperate neural networks for comparison.
the "key" is sent very last, triggering the collated values to be sent, and parsed into a sqlite database entry.
r cellblock
a jit.cellblock object
key time date pID gender training seed00 seed01 seed02 seed03 seed04 seed05 seed06 seed07 seed08 seed09 seed10 seed11
r key
r time
r date
r pid
r gender
r train
r seed
r bios
route preNeutral mus1 mus2 mus3 postNeutral
pack 0 t t t 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
listfunnel
route 80
$2\,
comma delimited except for the last value..
zl group 81
prepend INSERT INTO bioValues VALUES (
append)
atoi
itoa
doing an atoi and itoa coincidentally and conveniently puts the list into inverted commas!
prepend exec
sqlite is kind of native to max, implemented for use in tutorials and such, but not really documented for public use. sqlite is implemented via a java script object. Double click on the js object to see the script.
r db
closedb
1
0
js sqlitejs.js
gate
pipe 2
update
exec "SELECT * FROM bioValues"
"select" commands trigger the data to be displayed in the jit.cellblock. The delayed bang makes this command update the jit.cellblock after every addition.
Close Database
clear all
closedb
s cellblock
s db
Open Databse
Open DB
opendialog .db
relativepath
prepend opendb
Make a new database
New DB
opendialog fold
name.db
relativepath
unpack t t
combine ./ test
prepend opendb

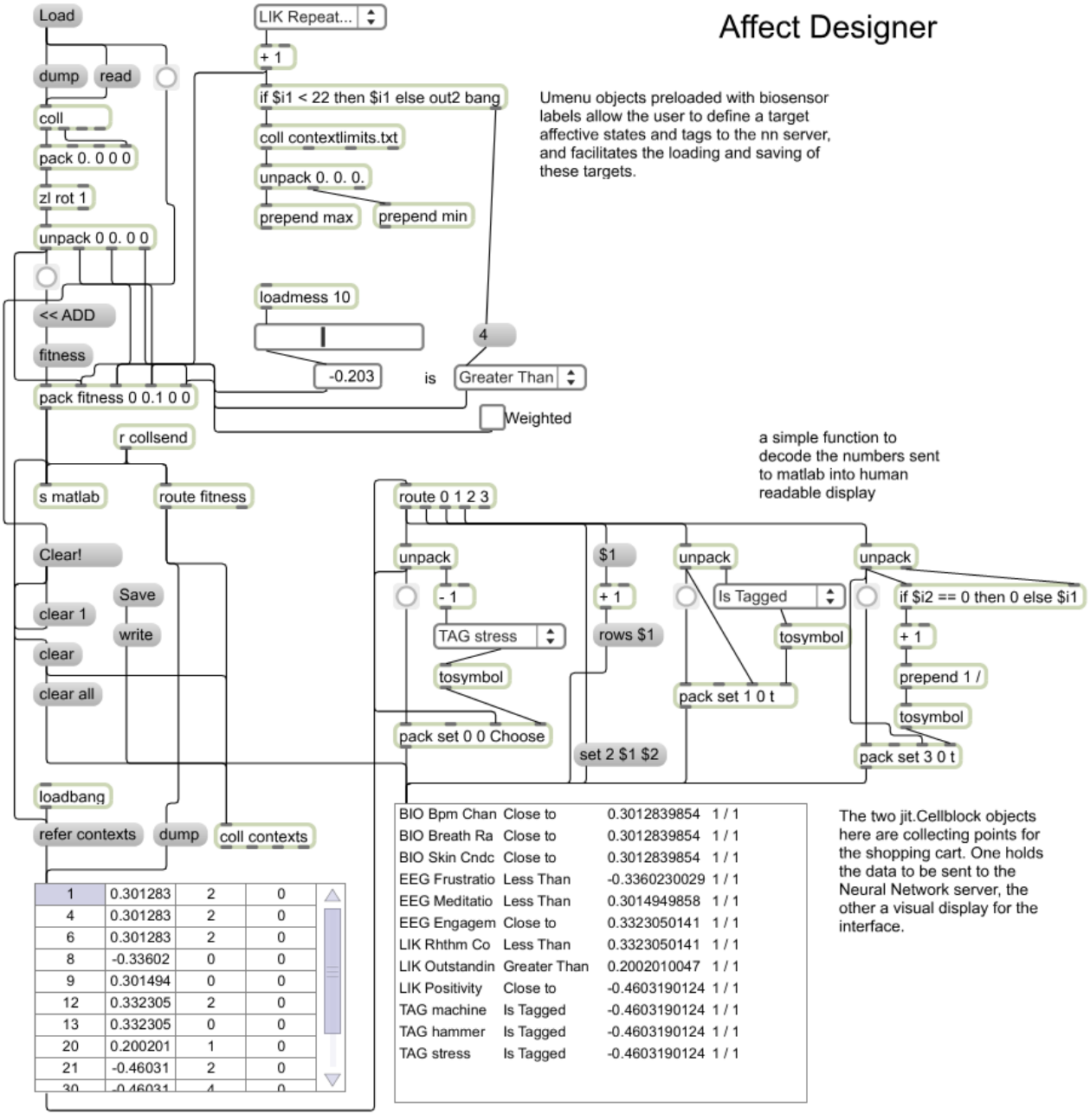
Javascript for SQLite in Max

```
var sqlite = new SQLite;
var result = new SQLResult;

function opendb(x)
{
    sqlite.open(x, 1);
}

function closedb()
{
    sqlite.close();
}

function exec(arg)
{
    sqlite.exec(arg, result);
    formatResultForCellblock();
}

function formatResultForCellblock()
{
    var numfields = result.numfields();
    var numrecords = result.numrecords();
    var fieldnames = new Array(numfields);
    var values = new Array(numfields);
    outlet(0, "clear", "all");
    outlet(0, "cols", numfields);
    outlet(0, "rows", numrecords + 1);
    for(var i=0; i<numfields; i++)
        outlet(0, "set", i, 0, result.fieldname(i));
    for(var i=0; i<numrecords; i++){
        for(var j=0; j<numfields; j++)
            outlet(0, "set", j, i+1, result.value(j, i));
    }
}
```


Getting Biosensor Target State
getting the biosensor data use a copy of the same code used to generat "physiological state overall" to briefly asses the participants bio state
Remember the biostate is measured in terms of change, not overall values, to two 5 second periods are compared.
2 measurements of 5 seconds each arre taken and compared to find overall change.
Import Biosensors
zlclear
metro 1000
counter 11
stop
sel 5 10 11
set
r hbeat
r brout
r aff
r sensors
unpack
unpack 0. 0.
zl group
mean
listfunnel
256
711.
164.
Mean Heart Beat Period
Heart Rate Variance
expr "sqrt(pow($i1, 2))"
11.79
Skin Conductance
23.47
Mean Body Temperature
0.
Mean Breath Amplitude
Mean Breath Period
route Frustration Meditation Excitement Engaged
route Long
Frustration
Meditation
Long Term Excitement
Short Term Excitement
Engagement
zl group 2
unpack 0. 0.
- 0.
* -1.
0.077826
-0.314338
-0.044853
-0.10625
-0.075368
0.016993
-0.035454
-0.049265


Communication with Neural Network Server
in the neural networks server, a random pool of passages must be generated and audited before searching can commence.
linedrive 8 9999999 10 30
f
Search Pool Size
999
/ 1700.
0.588
CPU Time
Generate
999
pool $1
s matlab
The Neural Network server, written in matlab, uses OSC to communicate similar to the algorithm and EEG apps.
Neural Netowrk Server Connected
-Generate Playlist-
loadmess 5
1
Playlist Size
playlist $1
playlist 5
s matlab
r matlab
udpsend localhost 3330
osc control to and from the matlab server. Ports are set for now,
udpreceive 3331
route loaded done plist
gate 1 0
clear
s plist
loadmess 1
Overwrite playlist
regexp \\[([^\\]]*)\\]
+ 1
fromsymbol
gate 2
prepend insert
s plist
parsing incoming playlist from matlab for entry into a "coll" object, commonly used as used for implementing playlists.


Playlist
loadbang
Source
playlist
Load Playlist
< l
<<
r plist
>>
l >
!Clear!
Save Playlist
read
1
prev
next
end
clear
write
+ 1
r newseed
0
gate 2
coll ding
Generate Random Passage
Current Passage
1 0 9 5 2 26 16 0 8 98 153 0 2 0 24 812 1 2 1 664
- 1
select 1 $1
loadbang
mode selmode 3
refer ding
prepend seed
s hre
s seed
1 1 3 9 7 0 7 83 4 1 338 685 2 0 2 19 91 1 3 9 485
2 5 6 8 5 0 23 7 11 2 29 288 1 1 3 24 557 1 1 15 185
3 4 0 0 5 0 22 54 4 1 252 728 1 0 3 19 170 0 3 13 65
4 13 7 11 5 3 3 58 12 11 128 949 1 0 5 21 583 1 1 20 233
5 13 0 0 4 2 21 32 10 9 809 318 2 2 2 5 696 0 0 3 58
6 1 0 9 5 2 26 16 0 8 98 153 0 2 0 24 812 1 2 1 664
7 2 6 1 1 2 13 55 12 7 78 636 5 0 2 8 870 1 3 19 306
8 5 6 8 1 3 13 2 11 10 550 281 4 1 3 3 748 1 0 5 351
9 2 15 10 3 1 22 53 12 7 80 960 2 0 4 18 794 0 2 22 182
10 5 8 8 5 2 7 8 14 1 9 630 2 1 2 4 203 2 2 6 291
$2
+ 1
change
A simple playlist system, so that playlists generated using the matlab scripts can be stored and played easily. The jit.Cellblock display is only really for the user's reference.


Affect Designer
Load
LIK Repeat...
+ 1
dump
read
if $i1 < 22 then $i1 else out2 bang
coll
coll contextlimits.txt
pack 0. 0 0 0
unpack 0. 0. 0.
zl rot 1
prepend max
prepend min
unpack 0 0. 0 0
loadmess 10
<< ADD
4
fitness
-0.203
is
Greater Than
pack fitness 0 0.1 0 0
Weighted
r collsend
s matlab
route fitness
Clear!
Save
clear 1
write
clear
clear all
loadbang
refer contexts
dump
coll contexts
Umenu objects preloaded with biosensor labels allow the user to define a target affective states and tags to the nn server, and facilitates the loading and saving of these targets.
a simple function to decode the numbers sent to matlab into human readable display
route 0 1 2 3
unpack
- 1
TAG stress
tosymbol
pack set 0 0 Choose
$1
+ 1
rows $1
set 2 $1 $2
unpack
Is Tagged
tosymbol
pack set 1 0 t
unpack
if $i2 == 0 then 0 else $i1
+ 1
prepend 1 /
tosymbol
pack set 3 0 t
1 0.301283 2 0
4 0.301283 2 0
6 0.301283 2 0
8 -0.33602 0 0
9 0.301494 0 0
12 0.332305 2 0
13 0.332305 0 0
20 0.200201 1 0
21 -0.46031 2 0
BIO Bpm Chan Close to 0.3012839854 1 / 1
BIO Breath Ra Close to 0.3012839854 1 / 1
BIO Skin Cndc Close to 0.3012839854 1 / 1
EEG Frustratio Less Than -0.3360230029 1 / 1
EEG Meditatio Less Than 0.3014949858 1 / 1
EEG Engagem Close to 0.3323050141 1 / 1
LIK Rhthm Co Less Than 0.3323050141 1 / 1
LIK Outstandin Greater Than 0.2002010047 1 / 1
LIK Positivity Close to -0.4603190124 1 / 1
TAG machine Is Tagged -0.4603190124 1 / 1
TAG hammer Is Tagged -0.4603190124 1 / 1
TAG stress Is Tagged -0.4603190124 1 / 1
The two jit.Cellblock objects here are collecting points for the shopping cart. One holds the data to be sent to the Neural Network server, the other a visual display for the interface.

Music Algorithm Code Examples

AACr_Twohands.scd:

```
/*
-------------------------------------------------------
TwoHands - By Daniel Pitman
A music algorithm, part of the Affective Algorithmic Composer (AACr)

                      ******

This script is the loading code, which boots a Supercollider server, initialises MIDI, audio,
and loads all the music algorithm and OSC functions. This is the only script which
needs to be evaluated manually. OSC communications are initiated automatically.

Comments are denoted with a "//"
-------------------------------------------------------

*/

//boot a server, and then determine the path for each module.

s.waitForBoot{

      var
      apppath=thisProcess.nowExecutingPath,
      osc=apppath.replace("AACr_TwoHands", "TwoHands.OSC2MAXinterface"),
      env=apppath.replace("AACr_TwoHands", "TwoHands.Core.EnvironmentVariables"),
      har=apppath.replace("AACr_TwoHands", "TwoHands.Core.HarmonyGeneratorRoutines"),
      ply=apppath.replace("AACr_TwoHands", "TwoHands.FormAndMidiPlayer"),
      hnd=apppath.replace("AACr_TwoHands", "TwoHands.HandModes"),
      arp=apppath.replace("AACr_TwoHands", "TwoHands.HandModes.arpegiator");

// setup memory allocations
      Server.local.options.memSize = 2 ** 20;
      Server.internal.options.memSize = 2 ** 20;
      ("Local:"++Server.local.options.memSize).postln;
      ("Internal:"++Server.internal.options.memSize).postln;


//load each module
      osc.load;
      env.load;
      har.load;
      ply.load;
      hnd.load;
      arp.load;


//initialize MIDI details
      MIDIClient.init;
//device can be altered later via the GUI
      p=MIDIOut(0);
      16.do({arg i;
         p.allNotesOff(i);
      });

//read metronome sound into memory (comment out to disable metronome).
//this is the only part which actually requires the supercollider server be booted.
      m = Buffer.read(s, Platform.resourceDir +/+ "sounds/SinedPink.aiff");

//test audio
      SynthDef(\metro, {| out = 0, bufnum = 0, speed = 1, vol=0.5 |
         var peep;
         peep = PlayBuf.ar(1, bufnum, speed*BufRateScale.kr(bufnum), doneAction:2);
         Out.ar([out, out+1],peep*vol);
      }).play(s, [\out, 0, \bufnum, m, \vol, 0.8,]);

      "Two Hands Loaded".postln;

//an emergency reset option which allows the scripts to be reloaded from the GUI
      ~reeset.free;~reeset = OSCresponder.new(nil, "reeset",{
         arg time, resp, command;

         var
         apppath=thisProcess.nowExecutingPath,
         osc=apppath.replace("AACr_TwoHands", "TwoHands.OSC2MAXinterface"),
         env=apppath.replace("AACr_TwoHands", "TwoHands.Core.EnvironmentVariables"),
         har=apppath.replace("AACr_TwoHands", "TwoHands.Core.HarmonyGeneratorRoutines"),
         ply=apppath.replace("AACr_TwoHands", "TwoHands.FormAndMidiPlayer"),
         hnd=apppath.replace("AACr_TwoHands", "TwoHands.HandModes"),
         arp=apppath.replace("AACr_TwoHands", "TwoHands.HandModes.arpegiator");

         //RE-load modules
         osc.load;
         env.load;
         har.load;
         ply.load;
```

```
         hnd.load;
         arp.load;

         //silence any preexisting MIDI notes
         16.do({arg i;
                  p.allNotesOff(i);
         });
      }).add;


};
```

TwoHands.OSC2MAXinterface.scd

```
/*
-------------------------------------------------------
TwoHands - By Daniel Pitman
A music algorithm, part of the Affective Algorithmic Composer (AACr)

                        ******

This script contains the OSC communication commands that can be sent
from the GUI to the algorithm.

Each OSCresponder object has a particular sciprt that is run when that
command is received from the GUI.

Comments are denoted with a "//"
-------------------------------------------------------
*/
// note: "z" is the main playback task.

// some commands


~pause.free;~pause = OSCresponder.new(nil, "pause",{
      arg time, resp, command;
      "pause".postln;
      z.pause;
}).add;

//midi all notes off and channel setting
~midic.free;~midic = OSCresponder.new(nil, "midic",{
      arg time, resp, command;

      16.do({arg i;
         p.allNotesOff(i);
      });

      p=MIDIOut(command[1]);  //select device

}).add;

//more playback control commands
~start.free;~start = OSCresponder.new(nil, "start",{
      arg time, resp, command;
      "start".postln;

      //initialize MIDI details
      //MIDIClient.init;
      //p=MIDIOut(0);  //select device
      16.do({arg i;
         p.allNotesOff(i);
      });
      z.stop;
      z.reset;
      z.start;
}).add;

~resume.free;~resume = OSCresponder.new(nil, "resume",{
      arg time, resp, command;
      "resume".postln;
      z.resume;
}).add;

~reset.free;~reset = OSCresponder.new(nil, "reset",{
      arg time, resp, command;
      "reset".postln;
      z.reset;
      16.do({arg i;
         p.allNotesOff(i);
      });
}).add;

~stop.free;~stop = OSCresponder.new(nil, "stop",{
      arg time, resp, command;
      "stop".postln;
      z.stop;
      /*16.do({arg i;
```

```
        p.allNotesOff(i);
    });*/
}).add;

//received a seed from the GUI
~newseed.free;~newseed = OSCresponder.new(nil, "seed",{
    arg time, resp, seed;

    z.stop;
    "1".postln;
    seed.removeAt(0);
    ~decodeSeed.reset;
    ~decodeSeed.value(seed);
    "new seed".postln;
}).add;

//Initialising a new passage
~newharmony.free;~newharmony = OSCresponder.new(nil, "newharmony",{
    arg time, resp, command;

    "2".postln;
    ~genpat.reset;
    ~chordpattern=nil;
    ~chordpattern=~genpat.value;  //to generate a new chord pattern
    //"new chord pattern".postln;
}).add;

~newrhythm.free;~newrhythm = OSCresponder.new(nil, "newrhythm",{
    arg time, resp, command;

    "3".postln;

    ~genrhythm.reset;
    ~coreharmony=nil;
    ~coreharmony=~genrhythm.value;  //to generate a new chord pattern
    ~coreharmony.postln;
}).add;


//metronome volume control
~metrovolume.free;~metrovolume = OSCresponder.new(nil, "metrovolume",{
    arg time, resp, command;

    ~metvol=command[1];
    command[1].postln;

}).add;

//a command which recalculates the existing seed from scratch
~newexecution.free;~newexecution = OSCresponder.new(nil, "newexecution",{
    arg time, resp, command;

    "4".postln;
    z.stop;
    z.reset;
    //"z reset".postln;
    //b.sendMsg("recalculated execution",);
}).add;


//some basic port setup and "loaded" message
a = NetAddr.langPort;
b = NetAddr.new("127.0.0.1", 57130);
b.sendMsg("port", a);
b.sendMsg("loaded", 1);
//b.sendMsg("OSC connected",);
```

TwoHands.Core.EnvironmentVariables.scd:

```
/*
-------------------------------------------------------
TwoHands - By Daniel Pitman
A music algorithm, part of the Affective Algorithmic Composer (AACr)

                    ******

Environment variables: These scripts are concerned with establishing environmental variables
(such as the chord board, scales etc.) and also interpreting incoming seeds from OSC into the
musical variables to which they relate.

Comments are denoted with a "//"
-------------------------------------------------------

*/

//midi channel
~channel = 1;
```

```
//a 2d matrix of chords to help navigate possible chord patterns. some chords are not
available to all scales (ie no 7 in pentatonic scales etc.)

~chordboard=[
      [1, 4, 6, 1],
      [7, 5, 2, 3],
      [5, 6, 4, 7],
      [1, 2, 3, 5]
];


//a selections of of scales by interval (must equal 12)
~scales = [
      [2, 2, 1, 2, 2, 2, 1], // ion
      [2, 1, 2, 2, 2, 1, 2], // dor
      [1, 2, 2, 2, 1, 2, 2], // phryg
      [2, 2, 2, 1, 2, 2, 1], // lydi
      [2, 2, 1, 2, 2, 1, 2], // mixo
      [2, 1, 2, 2, 1, 2, 2], // aeolian
      [1, 2, 2, 1, 2, 2, 2], // loch
      [2, 1, 2, 2, 2, 2, 1], // har minor
      [1, 2, 2, 2, 2, 1, 2], //loch with raised 5th
      [2, 2, 3, 2, 3],       //pents
      [2, 3, 2, 3, 2],
      [3, 2, 3, 2, 2],
      [2, 3, 2, 2, 3],
      [3, 2, 2, 3, 2],

      [3, 3, 3, 3],          //symmetrical weird scales
      [2, 2, 2, 2, 2, 2],
      [12, 12, 12, 12]       //octaves
];

//coordinates, up right down left, used on the chordboard during generation of core harmony.
~possibilities= [[0, 1], [1, 0], [0, -1], [-1, 0], [1, 1], [1, -1], [-1, -1], [-1, 1]];

 //multiple passages did combined to make greater forms, however this is currently redundant.
~form=["a"];



//************************************************************************
//an initial set of variable array elements


    //core harmony variables are often divided into first half and second half options,
approaching 'dominant' and departing from 'dominant'. Dominant is used loosely.

~startpoint = [3, 2];               //matrix coordinates
~inertia= [2, 3];                   //defines number of turns for first and second half
~vector=[[1, -1], [0, -1]];         //defines preferred direction first and second half
~style=[0, 2];                      //defines the preferred amount of change (notes) for each
chord, first and second half.
~preferOdd=[0, 0];                  //defines preferred odd or even chords
~primarymethod= [0, 2, 0];          //defines rates methods (int. 0=vector, 1=style,
2=preferodd) by importance (multiplier, thus 0 equals NO importance)
~repeatness= [0.1, 0.4];            //increase or decrease probability of using chords which
have already featured

    //basic rhythm and timing variables

~scale = [2, 1, 2, 2, 1, 2, 2];     //chosen scale (see environment variables)
~tonic = 4;                         //changes the tonic from C in semitones
~rhythm = [6, 4, 3];                //universal rhythmic characteristics

~probenv = [0.2, 0.2, 0.3];         //define probabilities used to generate core harmony
results
~runTime = 35;                      //run time

    //hand position and behavior variables

~lmode = 0;                         //operational mode
~lpos = 4;                          //octaves up from the far left
~lrhythm = [6, 3, 2];               //rhymthm pattern
~lprob = [0.2, 0.2, 0.1];           //three variables for controlling hand modes


~rmode = 1;                         //operational mode
~rpos = 3;                          //octaves up from the left hand
~rrhythm = [4, 6, 2];               //rhymthm pattern
~rprob = [0.3, 0.1, 0.6];           //three variables for controlling hand modes


~currentseed = [ 14, 0, 14, 2, 1, 7, 13, 7, 4, 867, 833, 3, 0, 0, 8, 130, 0, 2, 5, 733 ];
~oldseed = [1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2];
~reallyoldseed = [2, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2];
```

```
//*****************************************************************************

//Processing a new seeds (20 numbers) into 40 variables.

(

~decodeSeed.reset;
~decodeSeed=Routine({ arg seed;

	seed.postln;

	~reallyoldseed=~oldseed;
	~oldseed=~currentseed;
	~currentseed=seed;

	//startpoint (0-15)
	~startpoint[0]= floor(seed[0]/4);
	~startpoint[1]= seed[0]%4;
	["startpoint", ~startpoint].postln;

	//inertia (0-15)
	~inertia[0]= (floor(seed[1]/4))+1;
	~inertia[1]= (seed[1]%4)+1;
	["inertia", ~inertia].postln;

	//vector (0-15)
	~vector[0]=~possibilities[floor(seed[2]/4)];
	~vector[1]=~possibilities[seed[2]%4];
	["vector", ~vector].postln;

	//style (0-8)
	~style[0]=floor(seed[3]/4);
	~style[1]=seed[3]%4;
	["style", ~style].postln;

	//preferOdd (0-3)
	~preferOdd[0]=floor(seed[4]/2);
	~preferOdd[1]=seed[4]%2;
	["preferOdd", ~preferOdd].postln;

	//primary methd (0-26)
	~primarymethod[0]=seed[5]%3;
	~primarymethod[1]=floor(seed[5]/3)%3;
	~primarymethod[2]=floor(floor(seed[5]/3)/3)%3;
	["primarymethod", ~primarymethod].postln;

	//repeatness 99
	~repeatness[0]=(seed[6]%10)*0.1;
	~repeatness[1]=(floor(seed[6]/10))*0.1;
	["repeatness", ~repeatness].postln;

	//scales 0-15
	~scale=~scales[seed[7]];
	["scale", ~scale].postln;

	//tonic 0-12
	~tonic=seed[8];
	["tonic", ~tonic].postln;

	//rhythm 0-999
	~rhythm[0]=(floor(floor(seed[9]%10)*0.5))+4;
	~rhythm[1]=(floor((floor(seed[9]/10)%10)*0.5))+2;
	~rhythm[2]=(floor((floor(seed[9]/100)%100)*0.3))+1;
	["rhythm", ~rhythm].postln;

	//probability environment
	~probenv[0]=(seed[10]%10)*0.1;
	~probenv[1]=(floor(seed[10]/10)%10)*0.1;
	~probenv[2]=(floor(seed[10]/100)%100)*0.1;
	["probenv", ~probenv].postln;

	//runtime 0-5
	~runTime=seed[11]*4+20;
	["runTime", ~runTime].postln;

	//left hand
	~lmode=seed[12]; //(0-1)

	["lmode",~lmode].postln;


	//left hand rhtyhm 0-26
	~lrhythm[0]=~rhythm[seed[14]%3];
	~lrhythm[1]=~rhythm[floor(seed[14]/3)%3];
	~lrhythm[2]=~rhythm[floor(floor(seed[14]/3)/3)%3];
	["lrhythm", ~lrhythm].postln;

	//left hand probability environment
	~lprob[0]=(seed[15]%10)*0.1;
	~lprob[1]=(floor(seed[15]/10)%10)*0.1;
	~lprob[2]=(floor(seed[15]/100)%100)*0.1;
```

```
	["lprob", ~lprob].postln;

	//right hand
	~rmode=seed[16]; //(0-1)
	["rmode", ~rmode].postln;

	//hand positions
	~rpos=((seed[17])/2).ceil;   //(0-5)
	~lpos=seed[13]+1;   //(0-5)

	~rpos.do({arg i;
	   if (~rpos > 5,
			{
				if (~lpos > 1,
					{
						~rpos=~rpos-1;
						~lpos=~lpos-1;
					},{
						~rpos=~rpos-1;
				});
	   });
	});



	["rpos", ~rpos].postln;
	["lpos",~lmode].postln;


//right hand rhtyhm 0-26
	~rrhythm[0]=~rhythm[seed[18]%3];
	~rrhythm[1]=~rhythm[floor(seed[18]/3)%3];
	~rrhythm[2]=~rhythm[floor(floor(seed[18]/3)/3)%3];
	["rrhythm", ~rrhythm].postln;

//right hand probability environment
	~rprob[0]=(seed[19]%10)*0.1;
	~rprob[1]=(floor(seed[19]/10)%10)*0.1;
	~rprob[2]=(floor(seed[19]/100)%100)*0.1;
	["rprob", ~rprob].postln;


	seed.yieldAndReset;
});

)
```

TwoHands.Core.HarmonyGeneratorRoutines.scd

```
/*
-------------------------------------------------------
TwoHands - By Daniel Pitman
A music algorithm, part of the Affective Algorithmic Composer (AACr)

                      ******

This script contains the two functions which implement the harmony and rhythmic
data that will be used to guie the individual hand implementations.

Comments are denoted with a "//"
-------------------------------------------------------

*/


//*********************************************************************
//harmonic progression generator


//GENerate a PATtern of chords from the chord board, based on the scoring routines above and
the structural array data
// It calls the ~fourpossible routine (in turn calls the ~notecounter routine)

~genpat=Routine({
	var
	half0=Array.newClear(~inertia[0]),
	half1=Array.newClear(~inertia[1]),
	result=k,
	romnumresult=l;

	half0.size.do({arg i;     //generate first half of chord pattern
	   var moves, motion;
//score each possible new chord (routine)
		if (i == 0, {
			moves=~fourpossible.value([~startpoint[0], ~startpoint[1], 0, ~inertia[0]-i
]);
			},{
				moves=~fourpossible.value([ half0[i-1][0], half0[i-1][1], 0,
~inertia[0]-i ]);
		});
```

```
         moves=moves.windex;

//[i, "h0", moves].postln;
//repeat or move?
         if (moves==8, {
                  motion= [0,0];
                  },{
                           motion=~possibilities[moves];
         });
         half0.put(i, motion);
      });


//generate second half of chord pattern
      half1.size.do({arg i;
         var moves1,
         motion1;

//score each possible new chord
         if (i==0, {
                  //["i",i].postln;
                  moves1=~fourpossible.value([ half0[half0.size-1][0], half0[half0.size-1][1],
1, ~inertia[1]-i ]);
                  },{
                           moves1=~fourpossible.value([ half1[i-1][0], half1[i-1][1], 1,
~inertia[1]-i ]);
         });


         moves1=moves1.windex;


         if (moves1==8, {
//repeat or move?
                  motion1= [0,0];
                  },{
                           motion1=~possibilities[moves1];
         });

         half1.put(i, motion1);
      });

      result= [~startpoint] ++ half0 ++ half1;
      romnumresult=Array.newClear(result.size);

//compile all the coords for the chords and make sure coordinates wrap
      result.size.do({|i|
         if (i==0, {
                  romnumresult[0] = result[0];
                  romnumresult[i].size.do({|j|
                           if (romnumresult[i][j] >3 ,{romnumresult[i][j] =
romnumresult[i][j]-4});
                           if (romnumresult[i][j] <0 ,{romnumresult[i][j] =
romnumresult[i][j]+4});
                  });
                  },{
                           romnumresult[i] = romnumresult[i-1]+result[i];

                           romnumresult[i].size.do({|j|
                                    if (romnumresult[i][j] >3 ,{romnumresult[i][j] =
romnumresult[i][j]-4});
                                    if (romnumresult[i][j] <0 ,{romnumresult[i][j] =
romnumresult[i][j]+4});
                           });
         });
      });

//look up coords on the chord board
      romnumresult.size.do ({|i|
         romnumresult[i] = ~chordboard[romnumresult[i][0]][romnumresult[i][1]];
      });


      ["romnum",romnumresult].postln;
      romnumresult.yieldAndReset;


});



//***********************************************************************
//Rhythmic Application of the chord pattern (ie when the chords change)
//dependant on yield from ~genpat routine


~genrhythm=Routine({ arg val;
      var chordresult, lapCount=0, length;

//make a new array for the core chord pattern

      length = ~chordpattern.size*~rhythm[0]*~rhythm[1];
```

```
	chordresult = Array.newClear(length);


//populate with correct number of chord positions

	length.do({|i|

	   if (i % (length/~chordpattern.size) == 0, {


			if (i==0, {
				chordresult[i]= ~chordpattern[lapCount];
				},{
					chordresult[i+((~rhythm[2].bilinrand)*~rhythm[1])]=
~chordpattern[lapCount];
			});
			lapCount=lapCount+1;
		 });
	});

	chordresult.yieldAndReset;
});



//***********************************************************************
//chord scoring routine (called by ~fourpossible)

//a routine to yield the number of similare notes between two triad chord numbers. cha is an
array with two numbers representing chords

~notecounter= Routine ({arg cha;


	var scalenumber=~scale.size, notecounter, chorda, chordb;


//populate chord with note positions

	chorda = [cha[0], cha[0]+2, cha[0]+4];
	chordb = [cha[1], cha[1]+2, cha[1]+4];

//keep it in one octave
	chorda.size.do({|i|
	   if (chorda[i]> (scalenumber-1), {chorda[i] = chorda[i]-scalenumber});
	   if (chordb[i]> (scalenumber-1), {chordb[i] = chordb[i]-scalenumber});
	});

//count the notes
	notecounter=0;
	chorda.size.do({|j|
	   chordb.size.do({|k|
			if (chorda[j]==chordb[k], {notecounter = notecounter + 1});
	   });
	});

//yield a result
	notecounter.yieldAndReset;


});

//***********************************************************************
//score collecting (called by ~genpat)

//a routine that considers all possible movements on chordboard and returns a normalized
probability array (should be called eightpossible now)

//required argument is an array with x coord, y coord, half, current inertia,
~fourpossible = Routine ({ arg startcoord;
	var startchord, half, inertia, previous, score, repeats, wrap;

	wrap = [startcoord[0], startcoord[1]];
	wrap.size.do({ |i|
	   if (wrap[i] > 3, {wrap[i]=wrap[i]-4});
	   if (wrap[i] < 0, {wrap[i]=wrap[i]+4});
	});


	startchord = ~chordboard[wrap[0]] [wrap[1]];
	half = startcoord[2];
	inertia = startcoord[3];
	previous = startcoord[4];
	score= [0, 0, 0, 0, 0, 0, 0, 0];
	repeats=0;

	~possibilities.size.do({arg i;
	   var thischord, thiscoord=[nil, nil], stylescore=0, thisint;

	   thiscoord.put(0, startcoord[0] + ~possibilities[i][0]);
```

```
		thiscoord.put(1, startcoord[1] + ~possibilities[i][1]);

//wrap the coordinates if outside the matrix
		thiscoord.size.do({arg j;
			if (thiscoord[j] > 3, {thiscoord[j]=thiscoord[j]-4});
			if (thiscoord[j] < 0, {thiscoord[j]=thiscoord[j]+4});
		});

		thischord = ~chordboard[thiscoord[0]] [thiscoord[1]];

		if (thischord > ~scale.size, {stylescore = 0},
			{

//get number of different notes changing
				stylescore = ~notecounter.value([startchord, thischord]);

// difference from wanted number changing
				stylescore = (2 - (~style[half] - stylescore)) * ~primarymethod[1];

				if (~possibilities[i] == ~vector[half], {stylescore = stylescore +
(2*~primarymethod[0])});

//get score regarding target chord (dominant in first half, tonic in second)
				if (half == 0,{
					thisint = ~notecounter.value([thischord, 5]);
					},{
						thisint = ~notecounter.value([thischord, 1]);
				});

				if (~inertia[half] == inertia, {if (thisint == 3,
{stylescore=stylescore-1}) });
				if (~inertia[half] - inertia == 0, {if (thisint == 3,
{stylescore=stylescore+2},{stylescore=stylescore-2}) });

				if (~inertia[half] - inertia == 1, {
					if (thisint == 3, {stylescore=stylescore-2});
					if (thisint == 2, {stylescore=stylescore+2});
				});

				if (thischord.odd && ~preferOdd[half] == 1, {stylescore=stylescore
+ (1 * ~primarymethod[2])}, {
					if (thischord.even && ~preferOdd[half] == 0,
{stylescore=stylescore+(1 * ~primarymethod[2])});
				});


//check previous chords
				repeats=0;

				previous.size.do({arg j;
					if (j != previous.size, {
						if (previous[j] == thischord,
{repeats=repeats+1});

					});
				});

				if (repeats > 1, {stylescore = stylescore - (0.5-
~repeatness[half])});
				if (repeats > 3, {stylescore = stylescore - (0.8-
~repeatness[half])});
		});

//post score!
//score becomes an array with an entry for each possibile movement
		score.put(i, stylescore);

	});


//normalize the scores to 0-1 floating numbers
	score = score.normalize (0);

//add the possibility of not changing chord
	score=score ++ ~repeatness[half];

//make the floating numbers add up to 1 in total as probability weights.
	score=score.normalizeSum;

	score.yieldAndReset;


});
```

TwoHands.HandModes.scd

```
/*
-------------------------------------------------------
TwoHands - By Daniel Pitman
A music algorithm, part of the Affective Algorithmic Composer (AACr)

                        ******

A pair of routines that perform two potential hand modes, interpetting
the heirarchical structure laid out in the core harmony routines, and
elaborating on them in further detail.

It is expected that future hand modes can be added to the system.

Comments are denoted with a "//"
-------------------------------------------------------

*/

//basic rhythmic triads

~basicchords=Routine({ arg hand;
      var chordresult, markovCount=0, corestep, handprob, handrhythm, handpos;

//set variables for hands

      if (hand=="left", {
         handprob= ~lprob;
         handpos= ~lpos*12;
         handrhythm = ~lrhythm;
      });
      if (hand=="right", {
         handprob= ~rprob;
         handpos= (~lpos + ~rpos)*12;
         handrhythm = ~rrhythm;
      });

//check for existing coreharmony

      if (~coreharmony == nil, {"ERROR = no coreharmony found!".yieldAndReset;
         },{

//make new array

                  chordresult = Array.newClear(~coreharmony.size);

                  chordresult.size.do({arg i;

                           var
                           attempt=0,
                           nowchord= Array.newClear(3),
                           challenge;

//setup the markov counter
                           if (i==0, {markovCount= handprob[1];});

//will the chord be changing
                           if (~coreharmony[i] == nil, {},{corestep=~coreharmony[i]});

//asses strength of attempt for chord to exists here
                           handrhythm.size.do({ arg j;
                                    if (i % handrhythm[j] == 0, {attempt = attempt +
(j+3).reciprocal});
                           });

//asses challenge
                           challenge=0.2.rand + handprob[0] + markovCount;

//attempt vs challenge

                           //play chord?
                           if (challenge <= attempt,{

//basic triad from coreharmony
                                    nowchord=[corestep, corestep+2, corestep+4];

//Include position and scale data to define actual chords:

                                    nowchord.size.do({|j|
                                             var knote=0;
                                             if (nowchord[j]==0, {nowchord[j]= ~tonic+handpos;
```

```
                                                        },{
                                                                nowchord[j].do({|k|
                                                                        knote = knote +
~scale.wrapAt(k-1);
                                                                });

      nowchord[j]=[knote+~tonic+handpos, (attempt*0.5)+0.5];
                                                        });
                                        });

//decrease likihood of chord immediately after
                                        markovCount=markovCount + handprob[1];

//tell pulse before this one to noteoff
                                        chordresult.put(i, [nowchord, "noteOn"]);  //NEED TO
CHAnge format!~!!
                                        if (i==0, {}, {chordresult[i-1][1]=("noteOff")});

                                        },{
//don't play chord, put a nill entry, and slightly increase liklihood of playing on next pulse
                                                markovCount=markovCount- handprob[2];
                                                chordresult.put(i, [[[nil, nil]], nil]);
                                });


                        });
                        ["chords", chordresult].postln;
                        chordresult.yieldAndReset;

      });

});


//***************************************************************


// really simple bass line hand mode

~basicbassline=Routine({ arg hand;
      var chordresult, markovCount=0, corestep, handprob, handrhythm, handpos;

//set variables for hands
      if (hand=="left", {
         handprob= ~lprob;
         handpos= ~lpos*12;
         handrhythm = ~lrhythm;
      });
      if (hand=="right", {
                handprob= ~rprob;
                handpos= ~lpos+~rpos*12;
                handrhythm = ~rrhythm;
      });

//check for existing coreharmony
      if (~coreharmony == nil, {"ERROR = no coreharmony found!".yieldAndReset;
         },{

//make new array
                chordresult = Array.newClear(~coreharmony.size);

                chordresult.size.do({arg i;

                        var
                        attempt=0.0,
                        nowchord= Array.newClear(3),
                        challenge;

//setup the markov counter
                        if (i==0, {markovCount= handprob[1];});

//will the chord be changing
                        if (~coreharmony[i] == nil, {},{corestep=~coreharmony[i]});

//asses strength of attempt for chord to exists here
                        handrhythm.size.do({ arg j;

//strengthening bond to the "1" of each bar
                                if (i % handrhythm[0] == 0, {attempt = attempt + (1/(j+2))

                                        },{
                                                if (i % handrhythm[j] == 0, {attempt =
attempt + (1/(j+3))});

                                });
                        });

//asses challenge
                        challenge=0.2.rand + handprob[0] + markovCount;

//attempt vs challenge
```

```
                        if (challenge <= attempt,{
                                //play chord?

                                var neg = 0;
//bassnote
                                nowchord=[corestep];
                                if (i % handrhythm[0] == 0,{},{

//if we arnt on the "one", we have several alternatives
                                        nowchord = [[corestep,(corestep-3),(corestep-
1),(corestep+4),(corestep+7)].choose];

//if chosen note is below tonic, ie below 0, we can:
                                        if (nowchord[0] < 0, {

//stay positive by adding seven scale notes
                                                nowchord = nowchord + 7;

//offset by later subtracting 12 semitones
                                                neg = -12;
                                                },{
                                                        neg=0;
                                        });

                                });

//Include position and scale data to define actual chords (obviously taken from basictriads
but effective):
                                nowchord.size.do({|j|
                                        var note, knote=0, vello=0;

//establish actual note
                                        if (nowchord[j]==0, {
                                                note= ~tonic+handpos+neg;
                                                },{
                                                        nowchord[j].do({|k|
                                                                knote = knote +
~scale.wrapAt(k-1);
                                                        });
                                                        note=knote+~tonic+handpos+neg;
                                        });

//establish velocity
                                        if ((attempt*0.3)+0.5 > 1, {vello = 1}, {vello =
(attempt*0.3)+0.5 });

//translate to our "midi" format
                                        nowchord[j]=[note, vello];
                                });

//decrease liklihood of chord immediately after
                                markovCount=markovCount + handprob[1];

//put the note in the array
                                chordresult.put(i, [nowchord, "noteOn"]);

//include a note off for the previous note
                                if (i==0, {}, {chordresult[i-1][1]=("noteOff")});
                                },{

//don't play chord, put a nill entry, and slightly increase liklihood of playing on next pulse
                                        markovCount=markovCount- handprob[2];
                                        chordresult.put(i, [[[nil, nil]], nil]);
                        });


                });
                ["bass", chordresult].postln;
                chordresult.yieldAndReset;

        });

});
```

TwoHands.HandModes.arpegiator.scd

```
/*
-------------------------------------------------------
TwoHands - By Daniel Pitman
A music algorithm, part of the Affective Algorithmic Composer (AACr)

                        ******

A more hand mode routines that perform a series of differing arrpegiator techniques,
interpreting the hierarchical structure laid out in the core harmony routines, and
elaborating on them in further detail.

This is an example of new hand modes added to the system.
```

```
Comments are denoted with a "//"
-------------------------------------------------------

*/

//arpegiator

~basicarpeg=Routine({ arg hand;
      var handprob, handrhythm, handpos;

      //set variables for hands
      if (hand=="left", {
         handprob= ~lprob;
         handpos= ~lpos*12;
         handrhythm = ~lrhythm;
      });
      if (hand=="right", {
         handprob= ~rprob;
         handpos= (~lpos+~rpos)*12;
         handrhythm = ~rrhythm;
      });

//check for existing coreharmony
      if (~coreharmony == nil, {"ERROR = no coreharmony found!".yieldAndReset;
         },{
                  var
//how many pulses per note
                  restartchord,
                  ascending,
                  startnote,
                  leap,
                  interval,
                  currentint=0,
                  currentchord=~chordpattern[0],
                  thing=handrhythm[0]*handrhythm[1],
                  run,
                  sizec=~coreharmony.size,
                  resultpeg,

//make new array
                  chordresult = Array.newFrom(~coreharmony);

//does the run restart on each new coreharmony chord or when run finishes?
                  if ((handprob[0]*10).asInteger.odd, {restartchord=true},
{restartchord=false});

//does the run start on the coreharmony note or use a leap interval?
                  if ((handprob[1]*10).asInteger.odd, {startnote=true}, {startnote=false});

//otherwise what interval is used for leaping back at the end of a run?
                  leap=((handprob[2]+0.1)*10).ceil;

//arpegiating interval
                  interval=((handprob[1]+0.1)*5).ceil;

//ascending or descending?
                  if ( handprob[0] >0.5, {ascending=true},{ascending=false});
                  if (ascending,{leap=leap * (-1)},{interval=interval * (-1)});

//inserting starting points
                  if (restartchord, {

                           if (startnote,
                                    {
//reset rhythm coreharmony start on corehamrony chord?
                                             run=0;
                                             chordresult.size.do({arg i;
                                                      if (~coreharmony[i]!=nil,
                                                               {

      currentchord=~coreharmony[i];
                                                                        run=i;
                                                      });
                                                      if (i-run ==
(handrhythm[0]*handrhythm[1]),
                                                               {

      chordresult[i]=currentchord;
                                                                        run=i;
                                                      });
                                             });
                                    },
                                    {
//reset with coreharmony notes, but using own own chords
                                             run=0;
                                             chordresult.size.do({arg i;
                                                      if (~coreharmony[i] != nil,
                                                               {

      chordresult[i]=currentchord;

      currentchord=currentchord+leap;
```

```
                                                                run=i;
                                                        });
                                                        if (i-run ==
(handrhythm[0]*handrhythm[1]),
                                                                {

      chordresult[i]=currentchord;
                                                                run=i;
                                                        });
                                                });
                                });
                                },{
                                        if (startnote,
                                                {
//restart when run finishes only, but use coreharmony notes
                                                run=0;
                                                chordresult.size.do({arg i;
                                                        if (~coreharmony[i] != nil,
                                                                {

      currentchord=~coreharmony[i];
                                                        });
                                                        if (i == 0,
                                                                {

      chordresult[i]==currentchord;
                                                                run=i;
                                                        });
                                                        if (i-run ==
(handrhythm[0]*handrhythm[1]),
                                                                {

      chordresult[i]=currentchord;
                                                                run=i;
                                                        });

                                                });
                                                },
                                                {
//ignore coreharmony completely
                                                run=0;

                                                chordresult.size.do({arg i;
                                                        if (i == 0,
                                                                {


      chordresult[i]=currentchord;

      currentchord=currentchord+leap;
                                                                run=i;
                                                        },{
                                                                if ((i-run) ==
(handrhythm[0]*handrhythm[1]),

      {

      chordresult[i]=currentchord;

      currentchord=currentchord+leap;

      run=i;
                                                                });
                                                        });

                                                });
                                        });
                        });

//insert runs
                        currentchord=0;
                        currentint=0;
                        run=0;

                        chordresult.size.do({arg i;
                                var knote=0,
                            essex=0;

                                if (chordresult[i]!= nil,
//each time a starting note is encountered, a run is constructed from that point onwards.
                                {
                                        currentchord=chordresult[i];
                                        currentint=0;
                                        run=i;

                                        knote=currentchord;
                                },
                                {
                                        if (i-run == handrhythm[1],
                                                {
                                                        currentint= currentint+interval;
                                                        knote=currentchord+currentint;
                                                        run=i
```

```
                                                        },
                                                        {
                                                                knote=nil;
                                                        });
                                        });

                                        if (knote!=nil,
                                                {
                                                        knote.do({|k|
                                                                essex = essex + ~scale.wrapAt(k);
                                                        });
                                                        chordresult[i]=essex+~tonic+handpos;
                                        });
                        });

//Formatting the data with velocity and noteOn/noteOff data
                        chordresult.size.do ({arg i;

                                        var velo=0, length=0;
                                        if (chordresult[i] != nil,
                                                {
//caluculate velocity
                                                        velo=0.2;
                                                        if (i % ~rhythm[0]==0, {velo = velo +0.3});
                                                        if (i % ~rhythm[1]==0, {velo = velo +0.2});
                                                        if (i % ~rhythm[2]==0, {velo = velo +0.1});

//calculate note length
                                                        length=handrhythm[1]-
((handrhythm[1]*(handrhythm[2]*0.1)).ceil);

//insert parsed noteon
                                                        chordresult[i]= [[[chordresult[i], velo]],
"noteOn"];

//and noteoff  (in apregiator, note lengths are predictable)
                                                        if (i-length > 0,
                                                                {
                                                                        chordresult[i-
length][1]=("noteOff")
                                                        });
                                                },
                                                {
                                                        chordresult[i]= [[[nil, nil]], nil];
                                        });
                        });

                        ["arpeg",chordresult].postln;
                        chordresult.yieldAndReset;
        });
});
```

TwoHands.FormAndMidiPlayer.scd

```
/*
-----------------------------------------------------
TwoHands - By Daniel Pitman
A music algorithm, part of the Affective Algorithmic Composer (AACr)

                        ******

The chronologically last script to run. This code caluculates and performs
the final result. The "z" playback routine is often controlled via OSC commands.

It is expected that future hand modes can be added to the system.

Comments are denoted with a "//"
-----------------------------------------------------
*/

z= Task({
        var thisLeft, thisRight, pulsetime, bpm;

//check all is in order

        if (~basicchords == nil, {"Handmode (~basicchord) not loaded".postln;});
        if (~basicbassline == nil, {"Handmode (~basicbassline) not loaded".postln;});
        if (~basicarpeg == nil, {"Handmode (~basicarpeg) not loaded".postln;});

        ~basicbassline.reset;
        ~basicchords.reset;
        ~basicarpeg.reset;

//get the hand modes from the variable ~lmode and ~rmode organised for form parts a, b, and c.
        if (~lmode== 0, {~aleft = ~basicbassline.value("left")});
        if (~lmode== 1, {~aleft = ~basicchords.value("left")});
        if (~lmode== 2, {~aleft = ~basicarpeg.value("left")});
```

```
      if (~rmode== 0, {~aright = ~basicbassline.value("right")});
      if (~rmode== 1, {~aright = ~basicchords.value("right")});
      if (~rmode== 2, {~aright = ~basicarpeg.value("right")});

//play according to the form selected (note that forms using multiple seeds as sections
//were initiated used but are not implemented for AAC training purposes.
      pulsetime=~runTime/(~form.size*~aleft.size);

//caluclate bpm and report to GUI
      bpm=60/pulsetime;
      b.sendMsg("bpm", bpm);
      bpm.wait;

//so in this script, form[h] always = a, however forms may be implemnted later for
//greater composition systems where several forms such as "b" or "c" might refer to
//the combination of several seeds in a rondo or ternary format, for example.
//for now, only single seeds are required.
      ~form.size.do({arg h;
         if (~form[h]=="a", {thisLeft = ~aleft; thisRight = ~aright});


         p.allNotesOff(1);
         ~onleft = [0];
         ~onright = [0];

//Find and play noteOffs for this pulse
         thisLeft.size.do({arg i;

                  if (thisLeft[i][1]=="noteOff", {
                           ~onleft.size.reverseDo({|j|
                                    if (j==0, {},{
                                             p.noteOff(1, ~onleft[j]);
                                             ~onleft.removeAt(j);
                                    });

//each note commnad is reported to GUI for the keyboard display
                                    b.sendMsg("lnoteoff");
                           });
                  });

                  if (thisRight[i][1]=="noteOff", {
                           ~onright.size.reverseDo({|j|
                                    if (j==0, {},{
                                             p.noteOff(1, ~onright[j]);
                                             ~onright.removeAt(j);
                                    });
                                    b.sendMsg("rnoteoff");
                           });
                  });

//and noteons for this pulse
                  ["l", thisLeft[i][0][0]].postln;
                  ["ll",thisLeft[i][0][0][0]].postln;
                  if (thisLeft[i][0][0][0] != nil,   {
                           thisLeft[i][0].size.do ({|j|
                                    p.noteOn (
                                             1,
                                             note: thisLeft[i][0][j][0],
                                             veloc: 127*thisLeft[i][0][j][1]
                                    );
                                    b.sendMsg("lnoteon", thisLeft[i][0][j][0]); //
                                    ~onleft = ~onleft ++ thisLeft[i][0][j][0];
                           });
                  });
                  ["r",thisRight[i][0][0]].postln;
                  ["rr",thisRight[i][0][0][0]].postln;
                  if (thisRight[i][0][0][0] != nil,  {
                           thisRight[i][0].size.do ({|j|
                                    p.noteOn (
                                             1,
                                             note: thisRight[i][0][j][0],
                                             veloc: 127*thisRight[i][0][j][1]
                                    );
                                    b.sendMsg("rnoteon", thisRight[i][0][j][0]);
                                    ~onright = ~onright ++ thisRight[i][0][j][0];
                           });
                  });

                  if (i % ~rhythm[0] == 0, {

//plays  metronome on beats (delayed slightly to match midi)
                           {Synth(\metro, [\out, 0, \speed, 1, \bufnum, m, \vol,
~metvol]);}.defer(0.1);
                           b.sendMsg("metro", 1);
                  });
                  if (i % ~rhythm[1] == 0, {

//plays  metronome on beats (delayed slightly to match midi)
                           {Synth(\metro, [\out, 0, \speed, 2, \bufnum, m, \vol,
~metvol]);}.defer(0.1);
                           b.sendMsg("metro", 2);
                  });
```

```
//wait the appropriate time before starting the next pulse.
                pulsetime.wait;

                if (i+1 == thisLeft.size, {b.sendMsg("end", 1);});
        });

    });
});
```

Neural Network Server Code Examples

Function name	Format	Description
AAC_nnserver.m (.exe)		Creates an OSC server based on oscmex ("oscmex"). Upon receiving affective target matrices, result sizes, and search pool sizes, can use the following functions to return top scoring seeds to the interface playlist.
newseed.m	b=newseed(a)	Creates a matrix 'b' with 'a' random structural arrays in it in 20 part format
seed2nn.m	c=seed2nn(b)	If 'b' contains 20 part arrays, converts all the 20 part structural arrays in 'b' into 81 part neural network compatible arrays.
judge.m		Is responsible for coordinateing the neural networks addressing the seed pool.
simBIOFnet.m simEEGFnet.m simLIKFnet.m simTAGCnet.m	d=simBIOFnet(c)	If 'c' contains 81 part arrays, will audition all these via the biosensor neural network and return matrix 'd', with all the affective predictions from each network. The neural networks are formatted as functions, for use in compiled applications (hence no ability to train on the fly as yet).
scoring.m	e=scoring(c, f, g)	Scoring takes the auditioned results 'c', compares them to the target affective state provided as matrix 'f'. Scoring will return the top 'g' number of results with their index location and overall score as matrix 'e'.
report.m	h=report(e, b)	Report looks up the index numbers provided by 'e' in the original seeds provided as 'b' and lists the correlating arrays (in 20 part format) as matrix 'h'.

Included on the accompanying digital media is the scripts for the neural network server, as well as the oscmex library (which facilitates OSC functionality in Matlab), a collection of AACr data files and error analysis files in the file "AACdata.mat", and the trained neural networks in a native matlab format in a file, "AACnns.mat".

AAC_nnserver.m

```
function osc_server()

%define a gui
    handles = createGUI();

%OSC server and client
    osc = [];
    oscS = [];
    Acontext=[];
    Aaudited=[];
    Aseed=[];

%create GUI
    function h = createGUI()
        h.fig = figure('Menubar','none', 'Resize','off', ...
            'CloseRequestFcn',@onClose, ...
            'Name','OSC Server', 'Position',[100 100 220 140]);
        movegui(h.fig, 'center')
        h.start = uicontrol('Style','pushbutton', 'String','Start', ...
            'Callback',{@onClick,'start'}, ...
            'Parent',h.fig, 'Position',[20 20 80 20]);
        h.stop = uicontrol('Style','pushbutton', 'String','Stop', ...
            'Callback',{@onClick,'stop'}, ...
            'Parent',h.fig, 'Position',[120 20 80 20]);
        h.txt = uicontrol('Style','text', 'String','', ...
            'Parent',h.fig, 'Position',[60 80 100 20]);
        set(h.stop, 'Enable','off');
        drawnow expose
```

```
%with a timer for checking the OSC server for inputs
        h.timer = timer('TimerFcn',@receive, 'BusyMode','drop', ...
            'ExecutionMode','fixedRate', 'Period',0.11);
    end

%when user is ready, start the server and client (ports 3330 and 3331
%respectively) or free up addresses when stop is clicked

    function onClick(~,~,action)
        switch lower(action)
            case 'start'
                set(handles.start, 'Enable','off')
                set(handles.stop, 'Enable','on')
                osc = osc_new_server(3330);
                oscS = osc_new_address('127.0.0.1', 3331);
                d = struct('path','loaded', 'tt','i', 'data',{{1}});
                osc_send(oscS, d);
                start(handles.timer);
            case 'stop'
                set(handles.start, 'Enable','on')
                set(handles.stop, 'Enable','off')
                osc_free_server(osc); osc = [];
                osc_free_address(oscS); oscS = [];
                stop(handles.timer);
        end
        drawnow expose
    end

%receive an OSC message, check the path for an idea of how to treat it
    function receive(~,~)
        if isempty(osc), return; end

        m = osc_recv(osc, 0.1);
        if isempty(m), return; end
        n=[m{1}.path, m{1}.data];

        set(handles.txt, 'String',m{1}.path)
        drawnow expose

%generate a random seed pool
        if strcmp (n(1), 'pool') == 1

            Aseed=newseed([n{2}]);
            Ainputs=seed2nn(Aseed);

%and pass them through the neural networks for scoring
            Aaudited=judge(Ainputs);
            d = struct('path','done', 'tt','i', 'data',{{1}});
            osc_send(oscS, d);

        end;

%add to or create a target affect state
        if strcmp (n(1), 'fitness') == 1
            if isempty(Acontext)
                Acontext =[[n{2}], [n{3}], [n{4}], [n{5}]];
                disp ('single');
            else
                o=[[n{2}], [n{3}], [n{4}], [n{5}]];
                Acontext=[Acontext;o];
                disp('multi');
            end;


        end;

%or clear the existing affect state
        if strcmp (n(1), 'clear') == 1
            Acontext=[];
        end;

%score the random seeds according to target affect state,
%and then generate a playlist from the top scores

        if strcmp (n(1), 'playlist') == 1
%find our top scoring indexes
            Ascores=scoring(Aaudited, Acontext, [n{2}]);
            disp ('one');

%match them to the original seeds
            Areport=report(Ascores, Aseed);
            disp ('two');

%and populate the Max interface playlist with results.

%parse
            Plist=(1 : [n{2}]);
            Plist=[Plist;Areport];
            disp ('three');

%and deliver, one track at a time
            for i = 1:[n{2}]
```

```
                track=mat2str(Plist(:, i)');


                d = struct('path','plist', 'tt','s', 'data',{{track}});
                osc_send(oscS, d);
            end;
            disp ('four');


       end;

    end

%on closing the server, free up the addresses
    function onClose(~,~)
         if ~isempty(osc)
            osc_free_server(osc);

         end
         if ~isempty(oscS)
             osc_free_address(oscS);
         end
        stop(handles.timer); delete(handles.timer);
        delete(handles.fig);
        clear handles osc
    end
end
```

newseed.m

```
function x = newseed(num)

    %random seed generator
    %seed is a term used to described the compact form of the structural array.
    %where the integers are combined into groups. sdim will determine the size
    %of a seed, which can later be expanded into a fully neural network
    %compatible structural array with ordinals and nominals.

    sdim = [16 16 16 9 4 27 100 16 13 1000 1000 6 3 6 27 1000 3 4 27 1000];

    %from this we can generate "num" random seeds in a matrix

    x=zeros([num, 20]);

    for j=1:num
        for i=1:20
            x(j, i) = randi(sdim(1, i))-1;
        end;
    end;
   x=x';

end
```

seed2nn.m

```
function x = seed2nn (z)

%this function takes the seed and parses it into a neural netowrk
%compatible format. nominals and ordinals are handled in binary or scaled
%binary in neural networks, however in this expanded form, are difficult
%to handle. s is a matrix containing z seeds of 20 integers.

%find the array dimensions
z=z';
h=size(z, 1);

%create a new array in the 81 integer format

x = zeros([h, 82]);

%parse

for j=1:h
    %z(1)H
    if floor(z(j, 1)/4) == 0
        x(j, 1) = 1;
    elseif floor(z(j, 1)/4) == 1
        x(j, 2) = 1;
    elseif floor(z(j, 1)/4) == 2
        x(j, 3) = 1;
    elseif floor(z(j, 1)/4) == 3
        x(j, 4) = 1;
    end;
    if mod(z(j, 1),4) == 0
        x(j, 5) = 1;
    elseif mod(z(j, 1),4) == 1
        x(j, 6) = 1;
```

```
elseif mod(z(j, 1),4) == 2
    x(j, 7) = 1;
elseif mod(z(j, 1),4) == 3
    x(j, 8) = 1;
end;
%z(2)I
x(j, 9) = (floor(z(j, 2)/4)+1)/4;
x(j, 10)= (mod(z(j, 2), 4)+1)/4;
%z(3)J
if floor(z(j, 3)/4) == 0
    x(j, 11) = 1;
elseif floor(z(j, 3)/4) == 1
    x(j, 12) = 1;
elseif floor(z(j, 3)/4) == 2
    x(j, 13) = 1;
elseif floor(z(j, 3)/4) == 3
    x(j, 14) = 1;
end;
if mod(z(j, 3),4) == 0
    x(j, 15) = 1;
elseif mod(z(j, 3),4) == 1
    x(j, 16) = 1;
elseif mod(z(j, 3),4) == 2
    x(j, 17) = 1;
elseif mod(z(j, 3),4) == 3
    x(j, 18) = 1;
end;
%z(4)K
if floor(z(j, 4)/4) == 0
    x(j, 19) = 1;
elseif floor(z(j, 4)/4) == 1
    x(j, 20) = 1;
elseif floor(z(j, 4)/4) == 2
    x(j, 21) = 1;
end;
if mod(z(j, 4),4) == 0
    x(j, 22) = 1;
elseif mod(z(j, 4),4) == 1
    x(j, 23) = 1;
elseif mod(z(j, 4),4) == 2
    x(j, 24) = 1;
elseif mod(z(j, 4),4) == 3
    x(j, 25) = 1;
end;
%z(5)L
x(j, 26) = floor(z(j, 5)/2);
x(j, 27) = mod(z(j, 5), 2);
%z(6) M
if mod(z(j, 6), 3) == 0
    x(j, 28) = 1;
elseif mod(z(j, 6), 3) == 1
    x(j, 29) = 1;
elseif mod(z(j, 6), 3) == 2
    x(j, 30) = 1;
end;
if mod(floor(z(j, 6)/3),3) == 0
    x(j, 31) = 1;

elseif mod(floor(z(j, 6)/3),3) == 1
    x(j, 32) = 1;

elseif mod(floor(z(j, 6)/3),3) == 2
    x(j, 33) = 1;
end;
if mod(floor(floor(z(j, 6)/3)/3),3) == 0
    x(j, 34) = 1;
elseif mod(floor(floor(z(j, 6)/3)/3),3) == 1
    x(j, 35) = 1;
elseif mod(floor(floor(z(j, 6)/3)/3),3) == 2
    x(j, 36) = 1;
end;
%z(7)N
x(j, 37) = mod(z(j, 7), 10)*0.1;
x(j, 38) = floor(z(j, 7)/10)*0.1;
%z(8) O
x(j, (39+z(j, 8))) = 1;
%z (9) P
x(j, 55)=z(j, 9)/12;
%z (10) Q
x(j, 56)= mod(z(j,10), 10)*0.1;
x(j, 57)=mod(floor(z(j, 10)*0.1), 10)*0.1;
x(j, 58)=floor(z(j, 10)*0.01)*0.1;
%z (11) R
x(j, 59)= mod(z(j,11), 10)*0.1;
x(j, 60)=mod(floor(z(j, 11)*0.1), 10)*0.1;
x(j, 61)=floor(z(j, 11)*0.01)*0.1;
%z (12) S
x(j, 62) = z(j, 12)/5;
%z(13) T
x(j, (63+z(j, 13))) = 1;
%z(14) U
x(j, 66) = z(j, 14)/5;
%z(15) V
```

```
    y=[x(j, 56), x(j, 57), x(j, 58)];
    x(j, 67)=y(mod(z(j, 15), 3)+1);
    x(j, 68)=y(mod(floor(z(j, 15)/3),3)+1);
    x(j, 69)=y(mod(floor(floor(z(j, 15)/3)/3),3)+1);
    %z(16)W
    x(j, 70)=mod(z(j, 16), 10)*0.1;
    x(j, 71)=mod(floor(z(j, 16)*0.1), 10)*0.1;
    x(j, 72) = floor(z(j, 16)*0.01)*0.1;
    %z(17)X
    x(j, (73+z(j, 17))) = 1;
    %z(18)Y
    x(j, 76)=z(j, 18)/3;
    %z(19) Z
    x(j, 77)=y(mod(z(j, 19), 3)+1);
    x(j, 78)=y(mod(floor(z(j, 19)/3),3)+1);
    x(j, 79)=y(mod(floor(floor(z(j, 19)/3)/3),3)+1);
    %z(20) AA
    x(j, 80)=mod(z(j, 20), 10)*0.1;
    x(j, 81)=mod(floor(z(j, 20)*0.1), 10)*0.1;
    x(j, 82) = floor(z(j, 20)*0.01)*0.1;

end;
x=x';
end
```

judge.m

```
function x = judge (z)

%this script takes a matrix of NN compatible inputs (in the 82 integer format) and
%passes them through each of the neural networks,
%daniel pitman 2014

h=size(z);
w=h(2);
h=h(1);

error=0;

%is z in 82 format?
if (h==82) && (w == 82)
    disp('82 interger format assuming variables are vertical');
elseif (h~=82) && (w == 82)
    disp ('82 integer format but will be transposed');
    z=z';
elseif (h==82) && (w ~= 82)
    disp ('82 integer format but no transposing neccesary');
elseif (h~=82) && (w ~= 82)
    disp ('Error - data is in the wrong format');
    error=1;

end;



if error ~= 1
    %function based version of neural net simulation compatible with compiler
    a=simBIOFnet(z);
    b=simEEGFnet(z);
    c=simLIKFnet(z);
    d=simTAGCnet(z);

    %a bit of parsing
    for j = 1:  size(d, 2)
        M=d(:, j);
        [ bb, ix ] = sort( M, 'descend' );
        d(:, j)=0;
        for k = 1:4
            d(ix(k), j)=1;
        end;
    end;
    %bring the tables together and report
    x=[a ; b ; c ; d];
end;
end
```

simBIOFnet.m

```
function [y1] = simBIOFnet(x1)
%SIMBIOFNET neural network simulation function.
%
%
% [y1] = simBIOFnet(x1) takes these arguments:
%   x = 82xQ matrix, input #1
% and returns:
%   y = 7xQ matrix, output #1
```

```
% where Q is the number of samples.

%#ok<*RPMT0>

  % ===== NEURAL NETWORK CONSTANTS =====

  % Input 1
  x1_step1_xoffset =
[0;0;0;0;0;0;0;0;0.25;0.25;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0
;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0];
  x1_step1_gain =
[2;2;2;2;2;2;2;2;2.66666666666667;2.66666666666667;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2
;2;2;2;2;2.22222222222222;2.22222222222222;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2.22222222222222;
2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2;2;2;2;2
;2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2.222222
22222222;2;2;2;2;2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2.2222222
2222222;2.22222222222222];
  x1_step1_ymin = -1;

%(the list of weights for the following layers are extremely long, and have been excluded in
print for practical reasons)

  % Layer 1
  b1 = [0.0083569848519827808;-0.0031274966019145486;… ];
  IW1_1 = [0.0024100434761767904 0.0060097415606493888… ];


  % Layer 2
  b2 = [-0.060311067158879211;0.031319003088078702;… ];
  LW2_1 = [-0.05092044603298657З 0.023251668878365883…  ];


  % Output 1
  y1_step1_ymin = -1;
  y1_step1_gain = [1;1;1;1;1;1;1];
  y1_step1_xoffset = [-1;-1;-1;-1;-1;-1;-1];

  % ===== SIMULATION ========

  % Dimensions
  Q = size(x1,2); % samples

  % Input 1
  xp1 = mapminmax_apply(x1,x1_step1_gain,x1_step1_xoffset,x1_step1_ymin);

  % Layer 1
  a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*xp1);

  % Layer 2
  a2 = repmat(b2,1,Q) + LW2_1*a1;

  % Output 1
  y1 = mapminmax_reverse(a2,y1_step1_gain,y1_step1_xoffset,y1_step1_ymin);
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings_gain,settings_xoffset,settings_ymin)
  y = bsxfun(@minus,x,settings_xoffset);
  y = bsxfun(@times,y,settings_gain);
  y = bsxfun(@plus,y,settings_ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n)
  a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings_gain,settings_xoffset,settings_ymin)
  x = bsxfun(@minus,y,settings_ymin);
  x = bsxfun(@rdivide,x,settings_gain);
  x = bsxfun(@plus,x,settings_xoffset);
end
```

simEEGFnet.m

```
function [y1] = simEEGFnet(x1)
%SIMEEGFNET neural network simulation function.
%
%
% [y1] = simEEGFnet(x1) takes these arguments:
%   x = 82xQ matrix, input #1
% and returns:
%   y = 5xQ matrix, output #1
% where Q is the number of samples.
```

```
%#ok<*RPMT0>

  % ===== NEURAL NETWORK CONSTANTS =====

  % Input 1
  x1_step1_xoffset =
[0;0;0;0;0;0;0;0;0.25;0.25;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0
;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0];
  x1_step1_gain =
[2;2;2;2;2;2;2;2;2.66666666666667;2.66666666666667;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2
;2;2;2;2;2.22222222222222;2.22222222222222;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2.22222222222222;
2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2;2;2;2;2
;2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2.222222
22222222;2;2;2;2;2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2.2222222
2222222;2.22222222222222];
  x1_step1_ymin = -1;

%(the list of weights for the following layers are extremely long, and have been excluded in
print for practical reasons)

  % Layer 1
  b1 = [-0.00026837059421779707;0.0027490534340656589;  …];
  IW1_1 = [0.00012976728762139716 -0.014158279982624873  …];

  % Layer 2
  b2 = [-0.0017919281790146299;-0.086698114187520894;  …];
  LW2_1 = [0.090439168098039702 -0.0093490772034676133  …];

  % Output 1
  y1_step1_ymin = -1;
  y1_step1_gain =
[1.37931034482759;4.44444444444444;2.40963855421687;1.33333333333333;4.44444444444444];
  y1_step1_xoffset = [-0.71;-0.2;-0.29;-0.73;-0.25];

  % ===== SIMULATION ========

  % Dimensions
  Q = size(x1,2); % samples

  % Input 1
  xp1 = mapminmax_apply(x1,x1_step1_gain,x1_step1_xoffset,x1_step1_ymin);

  % Layer 1
  a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*xp1);

  % Layer 2
  a2 = repmat(b2,1,Q) + LW2_1*a1;

  % Output 1
  y1 = mapminmax_reverse(a2,y1_step1_gain,y1_step1_xoffset,y1_step1_ymin);
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings_gain,settings_xoffset,settings_ymin)
  y = bsxfun(@minus,x,settings_xoffset);
  y = bsxfun(@times,y,settings_gain);
  y = bsxfun(@plus,y,settings_ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n)
  a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings_gain,settings_xoffset,settings_ymin)
  x = bsxfun(@minus,y,settings_ymin);
  x = bsxfun(@rdivide,x,settings_gain);
  x = bsxfun(@plus,x,settings_xoffset);
end
```

simLIKFnet.m

```
function [y1] = simLIKFnet(x1)
%SIMLIKFNET neural network simulation function.
%
%
% [y1] = simLIKFnet(x1) takes these arguments:
%   x = 82xQ matrix, input #1
% and returns:
%   y = 9xQ matrix, output #1
% where Q is the number of samples.

%#ok<*RPMT0>
```

```
  % ===== NEURAL NETWORK CONSTANTS =====

  % Input 1
  x1_step1_xoffset =
[0;0;0;0;0;0;0;0;0.25;0.25;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0
;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0];
  x1_step1_gain =
[2;2;2;2;2;2;2;2;2.66666666666667;2.66666666666667;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2
;2;2;2;2;2.22222222222222;2.22222222222222;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2.22222222222222;
2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2;2;2;2;2
;2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2.222222
22222222;2;2;2;2;2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2.2222222
2222222;2.22222222222222];
  x1_step1_ymin = -1;

%(the list of weights for the following layers are extremely long, and have been excluded in
print for practical reasons)

  % Layer 1
  b1 = [0.0015126609741011507;-0.01378324579748802; …];
      IW1_1 = [-0.0088382264038172759 -0.0048981897555301825 …];
  % Layer 2
  b2 = [-0.029921079401091049;0.0049106138259782453; …];
  LW2_1 = [-0.090793906886305772 0.10568794194880675  …];

  % Output 1
  y1_step1_ymin = -1;
  y1_step1_gain = [2;2;2;2;2;2;2;2;2];
  y1_step1_xoffset = [0;0;0;0;0;0;0;0;0];

  % ===== SIMULATION ========

  % Dimensions
  Q = size(x1,2); % samples

  % Input 1
  xp1 = mapminmax_apply(x1,x1_step1_gain,x1_step1_xoffset,x1_step1_ymin);

  % Layer 1
  a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*xp1);

  % Layer 2
  a2 = repmat(b2,1,Q) + LW2_1*a1;

  % Output 1
  y1 = mapminmax_reverse(a2,y1_step1_gain,y1_step1_xoffset,y1_step1_ymin);
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings_gain,settings_xoffset,settings_ymin)
  y = bsxfun(@minus,x,settings_xoffset);
  y = bsxfun(@times,y,settings_gain);
  y = bsxfun(@plus,y,settings_ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n)
  a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings_gain,settings_xoffset,settings_ymin)
  x = bsxfun(@minus,y,settings_ymin);
  x = bsxfun(@rdivide,x,settings_gain);
  x = bsxfun(@plus,x,settings_xoffset);
end
```

simTAGCnet.m

```
function [y1] = simTAGCnet(x1)
%SIMTAGCNET neural network simulation function.
%
%
% [y1] = simTAGCnet(x1) takes these arguments:
%   x = 82xQ matrix, input #1
% and returns:
%   y = 50xQ matrix, output #1
% where Q is the number of samples.

%#ok<*RPMT0>

  % ===== NEURAL NETWORK CONSTANTS =====

  % Input 1
  x1_step1_xoffset =
[0;0;0;0;0;0;0;0;0.25;0.25;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0
;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0];
```

```
  x1_step1_gain =
[2;2;2;2;2;2;2;2;2.66666666666667;2.66666666666667;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2
;2;2;2;2;2.22222222222222;2.22222222222222;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2.22222222222222;
2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2;2;2;2;2
;2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2.222222
22222222;2;2;2;2;2;2.22222222222222;2.22222222222222;2.22222222222222;2.22222222222222;2.2222222
2222222;2.22222222222222];
  x1_step1_ymin = -1;

%(the list of weights for the following layers are extremely long, and have been excluded in
print for practical reasons)

  % Layer 1
  b1 = [-1.6063606855947747;-1.6202590147823288; …];
  IW1_1 = [0.13441176976767255 0.18281823804068265 …];

  % Layer 2
  b2 = [0.26242223005678944;-0.89590065634591398; …];
  LW2_1 = [-0.57796284381775742 -0.2324463588647622  …];

  % Output 1
  y1_step1_ymin = -1;
  y1_step1_gain =
[2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2
;2;2;2];
  y1_step1_xoffset =
[0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0
;0;0;0];

  % ===== SIMULATION ========

  % Dimensions
  Q = size(x1,2); % samples

  % Input 1
  xp1 = mapminmax_apply(x1,x1_step1_gain,x1_step1_xoffset,x1_step1_ymin);

  % Layer 1
  a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*xp1);

  % Layer 2
  a2 = repmat(b2,1,Q) + LW2_1*a1;

  % Output 1
  y1 = mapminmax_reverse(a2,y1_step1_gain,y1_step1_xoffset,y1_step1_ymin);
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings_gain,settings_xoffset,settings_ymin)
  y = bsxfun(@minus,x,settings_xoffset);
  y = bsxfun(@times,y,settings_gain);
  y = bsxfun(@plus,y,settings_ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n)
  a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings_gain,settings_xoffset,settings_ymin)
  x = bsxfun(@minus,y,settings_ymin);
  x = bsxfun(@rdivide,x,settings_gain);
  x = bsxfun(@plus,x,settings_xoffset);
end
```

scoring.m

```
function R = scoring(db, contextfile, resultsize)

%Here the context file which contains the affective state is intepretted.
%
%Scoring takes the current pool and organises a list of the top entries
%in order acording to the affective state's outlines.

score=zeros(1, size(db, 2));

   for i= 1: size(contextfile, 1)

       feature=contextfile(i, 1);
       threshold = contextfile(i, 2);
       sign = contextfile(i, 3);
       if contextfile (i, 4) == 1
           weight = 1;
       else
           weight = i;
       end;
```

```
    %If the boolean is Less Than
       if sign == 0
           for j = 1: size(db, 2)
               if db(feature, j) < threshold
                   score(1, j)=score(1, j)+weight/i;
               %else

               end;
           end;
       end;
    %If the boolean is Greater Than
       if sign == 1
           for j = 1: size(db, 2)
               if db(feature, j) > threshold
                   score(1, j)=score(1, j)+weight/i;
               %else

               end;
           end;
       end;

     %If the boolean is Close Too
       if sign == 2
           for j = 1: size(db, 2)
               if db(feature, j) - threshold < 0.1 && db(feature, j) - threshold > -0.1
                   score(1, j)=score(1, j)+weight/i;
              % else

               end;
           end;
       end;

       %If the boolean is Exactly Equal Too
       if sign == 3
           for j = 1: size(db, 2)
               if db(feature, j) == threshold
                   score(1, j)=score(1, j)+weight/i;
               %else

               end;
           end;
       end;

    %If the boolean is Tagged
       if sign == 4
           for j = 1: size(db, 2)
               if db(feature, j) > 0
                   score(1, j)=score(1, j)+weight/i;
              % else

               end;
           end;
       end;

    %put results in order
       [ bb, ix ] = sort( score, 'descend' );
       R=[ix(1:resultsize);bb(1:resultsize)];


   end;

end
```

report.m

```
function x = report(score, seed)

%Results from the scores are matches to their original 20 part seeds.

ss=size(score, 2);

    for i = 1 : ss
        x (:, i)=seed(:, score(1, i));
    end;
end
```

Appendix B- Data Collected

It is impractical to present such a huge database in print, however on the accompanying digital media can be found the database file created by the GUI after all data was collected, an excel spread sheet which overviews the data used and highlights excluded entries, and a Matlab file containing the edited training data used to train the neural networks.

Appendix C- Generating Affective Music

Instructions for generating affective music:

1. **Run the AACr GUI** - Use either Cycling 74's Max 5, or the Max 5 runtime.
2. **Run algorithm loading script** - load the main algorithm script into the Supercollider IDE. Evaluate the file. This will load all the required algorithms scripts and initiate a connection to the GUI. The "Algorithm OSC" light will light up on the GUI.
3. **Select a MIDI device in the GUI** - This may be tested by generating a random seed and playing it in the Playback controls of the GUI. A virtual Piano device is recommended.
4. **Load the Neural Network Server** - Either run the "AAC_NNserver.exe" executable file (for windows) or run the uncompiled script 'AAC_NNserver.m' in Matlab itself. Click start, and confirm that the "Neural Network Server Connected" light is on in the GUI. Select a pool size, and wait for the analysis process to complete. The process is complete when the "Affect Designer" interface is no longer greyed out.
5. **Define an affective state** - Do this by either manually choosing affect aspects and thresholds from the shopping cart interface, or by clicking on the "Import Biosensors" button (preferably while some biosensors are being worn)
6. **Generate a Playlist** - By selecting a playlist size, and then clicking on the "Generate Playlist" button, the required affective state is sent to the Neural Network Servers and

compared to the scores of the seeds within the current pool. The returned playlist will contain the highest scoring seeds first.

7. **Playback** – In the “Playlist” control panel, select the “<|” button to force the playlist to return to the first track. Checking “autocue” will make the AACr play each track in turn without prompting, or tracks can be selected using the mouse. “Repeats” defines the number of times each track is played before stopping playback or proceeding to the next track in the list.

In the digital accompaniment is a demonstration video. It covers some basic instructions for setting up the AACr, demonstrates manually and automatically generating the desired affective targets from biosensors, and using the Neural Network Server to generate a playlist to suit that affective target.

The resulting piece is included as audio, with a playlist and affective state file that can be loaded back into the AACr interface, also shown here:

Affective Target
0, "BIO Bpm Change Close to 0.301284 1 / 1 ";
1, "BIO Breath Rate Close to 0.301284 1 / 1 ";
2, "BIO Skin Cndct Close to 0.301284 1 / 1 ";
3, "EEG Frustration Less Than -0.336023 1 / 1 ";
4, "EEG Meditation Less Than 0.301495 1 / 1 ";
5, "EEG Engagemnt Close to 0.332305 1 / 1 ";
6, "LIK Rhthm Complex Less Than 0.332305 1 / 1 ";
7, "LIK Outstanding Greater Than 0.200201 1 / 1 ";
8, "LIK Positivity Close to -0.460319 1 / 1 ";
9, "TAG machine Is Tagged -0.460319 1 / 1 ";
11, "TAG stress Is Tagged -0.460319 1 / 1 ";

Playlist Generated

1, 15 3 4 6 1 5 94 5 4 689 437 4 2 3 9 747 1 1 16 639;
2, 15 12 3 8 2 1 58 8 0 409 253 5 2 1 14 812 1 2 18 131;
3, 3 15 12 8 1 5 72 12 3 167 555 1 2 0 5 608 1 3 6 569;
4, 7 13 6 0 2 17 59 14 1 995 770 3 2 4 25 345 0 3 9 850;
5, 14 12 6 8 2 10 83 13 11 329 511 5 2 1 4 291 1 1 21 463;
6, 14 13 1 0 0 2 78 1 12 389 478 5 0 5 23 701 2 1 22 184;
7, 15 4 5 2 1 4 95 10 9 13 372 4 2 5 23 785 1 3 10 716;
8, 13 15 13 0 3 4 8 13 4 567 646 5 2 2 1 231 2 3 18 138;

Appendix D- Unsupervised Output

These pieces were generated by the AACr without supervision, as a demonstration of the AACr's immediate output. A brief description, discussion, and specifications for each track are provided here. Copies of the playlists, affect settings, and recordings are provided on the accompanying digital media.

It's worth noting that several gaps are apparent, where passages with no notes (or few notes) have been generated. It was not intended by the programmer that passages have no notes, but the algorithm has proven to be quite capable of breaking out of expected behaviour, and in the case of silence, this might be a more of a feature than a bug, in keeping with Cage's philosophy on silence. Several of the arpeggio hand-modes are known to leave the range of a typical keyboard, tempos often defy the maximum speed of the action of a piano, and some passages may have notes but with no velocity.

It most cases, addressing such bugs is a straight forward and largely anticipated developmental task, but in the case of an AAC algorithm, it is essential that the implementation of variables to musical output remain consistent once training has taken place, limiting what bugs can actually be fixed without interfering with the predictive aspects of the system. Thus, these demonstrations must unfortunately be presented flaws-and-all, in order to justly relay the underlying ability of the system.

Searching for Machine

Method

Manually defined affective state, playlist consists of top ten responses.

Description

A composition generated by the AACr in response to a manually entered affective target. This piece is a collection of ten passages (selected entirely without supervision) that are supposed to invoke an emotionally positive response, not increase your hand sweat, but are very likely to feature the tag "Machine".

Playlist generated

1, 2 5 0 7 3 7 80 14 10 660 355 5 0 3 7 919 0 3 15 283;
2, 1 9 2 7 3 24 18 14 1 43 672 0 0 3 8 502 1 1 14 572;
3, 15 1 3 5 1 9 50 14 6 54 176 2 1 2 25 899 1 3 23 353;
4, 8 3 3 5 2 3 4 7 8 72 952 1 2 0 25 724 2 1 7 280;
5, 5 2 15 6 0 6 93 1 3 417 361 1 2 5 24 719 2 1 25 350;
6, 12 14 3 7 0 6 59 11 3 230 70 0 0 1 18 880 1 1 6 114;
7, 9 1 15 4 2 24 11 10 1 585 968 4 1 4 1 3 2 2 7 103;
8, 2 9 2 0 0 24 40 12 6 148 445 0 0 2 20 515 0 2 9 37;
9, 6 0 3 8 0 25 75 7 0 172 29 4 0 2 7 526 1 0 5 294;
10, 1 0 9 5 3 24 14 11 1 223 614 4 1 1 0 469 0 3 0 86;

Good vs. Evil

Method

Manually defined affective state, playlist consists of top ten responses.

Description

A composition generated by the AACr in response to a manually entered affective target. This piece is a collection of 10 passages (selected entirely without supervision) that are supposed to be theatrical, mildly engaging, and are very likely to invoke the tag words "Good" or "Evil".

Playlist generated

1, 1 3 9 7 0 7 83 4 1 338 685 2 0 2 19 91 1 3 9 485;
2, 5 6 8 5 0 23 7 11 2 29 288 1 1 3 24 557 1 1 15 185;
3, 4 0 0 5 0 22 54 4 1 252 728 1 0 3 19 170 0 3 13 65;
4, 13 7 11 5 3 3 58 12 11 128 949 1 0 5 21 583 1 1 20 233;
5, 13 0 0 4 2 21 32 10 9 809 318 2 2 2 5 696 0 0 3 58;
6, 1 0 9 5 2 26 16 0 8 98 153 0 2 0 24 812 1 2 1 664;
7, 2 6 1 1 2 13 55 12 7 78 636 5 0 2 8 870 1 3 19 306;
8, 5 6 8 1 3 13 2 11 10 550 281 4 1 3 3 748 1 0 5 351;
9, 2 15 10 3 1 22 53 12 7 80 960 2 0 4 18 794 0 2 22 182;
10, 5 8 8 5 2 7 8 14 1 9 630 2 1 2 4 203 2 2 6 291;

Appendix E- Soundtrack for Fritz Lang's *Metropolis*

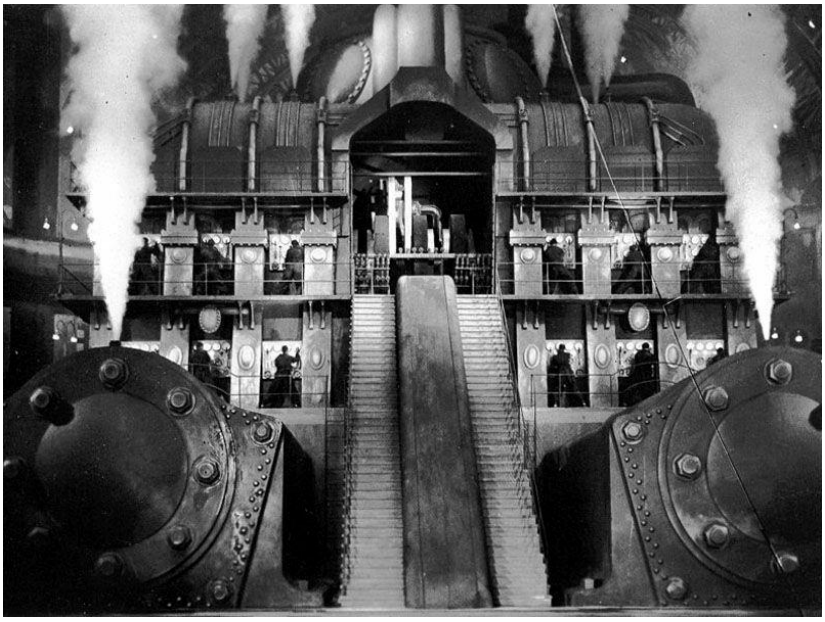


Figure 23 - A scene from Fritz Lang's *Metropolis*

The pieces in this section are provided as demonstration of the AACr as a compositional aide. The three compositions here are intended to accompany three famous scenes from the silent film, *Metropolis* by Fritz Lang, (1927). The version referred to here is the restored version by Universum Film, (2003).

For each moment in the film, a very detailed target affective state was defined to attempt to match the film's intent, and the most appropriate of between 3-7 suggestions was chosen for each moment. The whole result is a combination of each of these affective targets. Despite being 'supervised', these compositions are still intended to reflect the more immediate results of the AACr to the affective states provided, thus the foremost passages outputted by the system were preferred unless flawed somehow.

There was some minor editing to line up the music with the film but nothing that changes the music generated. The score arrangements attempt to reflect the pieces as much as possible, although have been 'cleaned up' to be readable. There are often no clear time signatures (thus 4/4 is often used by default) and key signatures are occasionally unclear, often being modal, chromatic, or atonal.

There are a few passages that leave the playable range of a lot of manual or small pianos, and passages that are probably too fast to play. It was decided to leave these passages in place, rather than re-arrange them for human players as digital instruments will have no trouble replicating these scores. Justly demonstrating the abilities of the system is the priority here. Players should simply ignore out of range notes, or play every alternate chord when the tempo exceeds the maximum speed of the piano's action.

The accompanying digital media has the video files of the film scenes (where copyright permits) with music audio synced, audio only files, PDF versions of the scores, and AACr compatible playlists and affect design files.

The timestamps included reference the 2003 restored version of the film, 2.30:11 in length.

Moloch!

(Starting at 13:29) Freder has descended into the worker's city only to be confronted by a horrific accident that wounds and kills workers. Freder is confronted by hallucinations of the machine taking the likeness of Moloch; a demon of metal and soulless industry who demands human sacrifices to be appeased.

This particular scene was chosen, for its dramatic scenes, action, and mechanical rhythmic movement.

Playlist
1, 15 12 2 1 3 16 30 13 0 175 821 5 2 5 9 499 0 0 4 925;
2, 6 7 11 7 1 26 62 5 12 900 549 0 2 0 5 343 0 2 16 812;
3, 4 11 9 7 1 6 44 10 4 939 645 2 2 4 26 132 2 3 17 887;
4, 8 14 10 1 3 25 92 14 1 742 746 4 1 1 0 497 0 3 0 35;
5, 4 4 5 7 0 9 0 8 12 621 54 3 1 1 11 347 2 0 16 804;
6, 8 13 7 2 3 7 99 15 7 681 568 3 1 0 24 725 1 0 6 579;
7, 0 4 8 2 3 24 1 5 3 513 740 3 2 3 3 675 1 3 7 835;

Affective Target for Part 1
0, "BIO Hrv Change Close to 0.000000 1 / 1 ";
1, "BIO Breath Sizes Close to 0.000000 1 / 1 ";
2, "BIO Skin Cndct Close to 0.000000 1 / 1 ";
3, "EEG Frustration Less Than -0.090000 1 / 1 ";
4, "EEG Meditation Greater Than 0.410000 1 / 1 ";
5, "EEG Engagemnt Greater Than 0.410000 1 / 1 ";
6, "LIK Rhthm Complex Close to 0.020000 1 / 1 ";
7, "LIK Repeatblty Greater Than 0.640000 1 / 1 ";
8, "LIK Outstanding Greater Than 0.640000 1 / 1 ";
9, "TAG dark Is Tagged 0.800000 1 / 1 ";
10, "TAG lonely Is Tagged 0.800000 1 / 1 ";
11, "TAG machine Is Tagged 0.800000 1 / 1 ";
12, "TAG forest Is Tagged 0.800000 1 / 1 ";

Affective Target for Part 2
0, "BIO Bpm Change Close to 0.300000 1 / 1 ";
1, "BIO Breath Sizes Close to 0.300000 1 / 1 ";
2, "BIO Skin Cndct Close to 0.300000 1 / 1 ";
3, "EEG Meditation Less Than 0.000000 1 / 1 ";
4, "EEG ST Exctmnt Close to 0.510000 1 / 1 ";
5, "EEG Engagemnt Close to 0.350000 1 / 1 ";
6, "LIK Rhthm Complex Close to 0.352000 1 / 1 ";
7, "LIK Repeatblty Close to 0.954000 1 / 1 ";
8, "LIK Thematicism Close to 0.520000 1 / 1 ";

9, "LIK Outstanding Close to 0.730000 1 / 1 ";
11, "TAG machine Is Tagged -0.098000 1 / 1 ";
12, "TAG dark Is Tagged -0.098000 1 / 1 ";
13, "TAG hammer Is Tagged -0.098000 1 / 1 ";
14, "TAG trickle Is Tagged -0.098000 1 / 1 ";

Affective Target for Part 3
0, "BIO Bpm Change Close to 0.300000 1 / 1 ";
1, "BIO Breath Sizes Close to 0.300000 1 / 1 ";
2, "BIO Skin Cndct Close to 0.300000 1 / 1 ";
3, "EEG Meditation Less Than 0.000000 1 / 1 ";
4, "EEG ST Exctmnt Close to 0.510000 1 / 1 ";
5, "EEG Engagemnt Close to 0.350000 1 / 1 ";
6, "LIK Rhthm Complex Close to 0.352000 1 / 1 ";
7, "LIK Repeatblty Close to 0.954000 1 / 1 ";
8, "LIK Thematicism Close to 0.520000 1 / 1 ";
9, "LIK Outstanding Close to 0.730000 1 / 1 ";
11, "TAG machine Is Tagged -0.098000 1 / 1 ";
12, "TAG dark Is Tagged -0.098000 1 / 1 ";
13, "TAG hammer Is Tagged -0.098000 1 / 1 ";

Affective Target for Part 4
0, "BIO Bpm Change Greater Than 1.235272 1 / 1 ";
1, "BIO hrv Percent Greater Than 1.235272 1 / 1 ";
2, "BIO Breath Sizes Greater Than 1.235272 1 / 1 ";
3, "BIO Skin Cndct Greater Than 1.235272 1 / 1 ";
4, "EEG Frustration Greater Than 1.235272 1 / 1 ";
5, "LIK Rhthm Complex Greater Than 1.235272 1 / 1 ";
6, "LIK Positivity Less Than -0.210000 1 / 1 ";
7, "TAG choppy Is Tagged -0.210000 1 / 1 ";
8, "TAG stress Is Tagged -0.210000 1 / 1 ";
9, "TAG machine Is Tagged -0.210000 1 / 1 ";
10, "TAG loud Is Tagged -0.210000 1 / 1 ";

Affective Target for Part 5
0, "BIO Bpm Change Precisely 0.000000 1 / 1 ";
1, "BIO Breath Rate Precisely 0.000000 1 / 1 ";
2, "BIO Skin Cndct Precisely 0.000000 1 / 1 ";
3, "BIO Temp change Greater Than 0.300000 1 / 1 ";
4, "EEG Meditation Greater Than 0.300000 1 / 1 ";
5, "EEG LT Exctmnt Greater Than 0.300000 1 / 1 ";
6, "EEG Engagemnt Greater Than 1.235272 1 / 1 ";
7, "LIK Repeatblty Greater Than 1.109300 1 / 1 ";
8, "LIK Thematicism Close to 0.000000 1 / 1 ";
9, "LIK Cadential Greater Than 0.760000 1 / 1 ";
11, "TAG demon Is Tagged 0.760000 1 / 1 ";
12, "TAG poetry Is Tagged 0.760000 1 / 1 ";

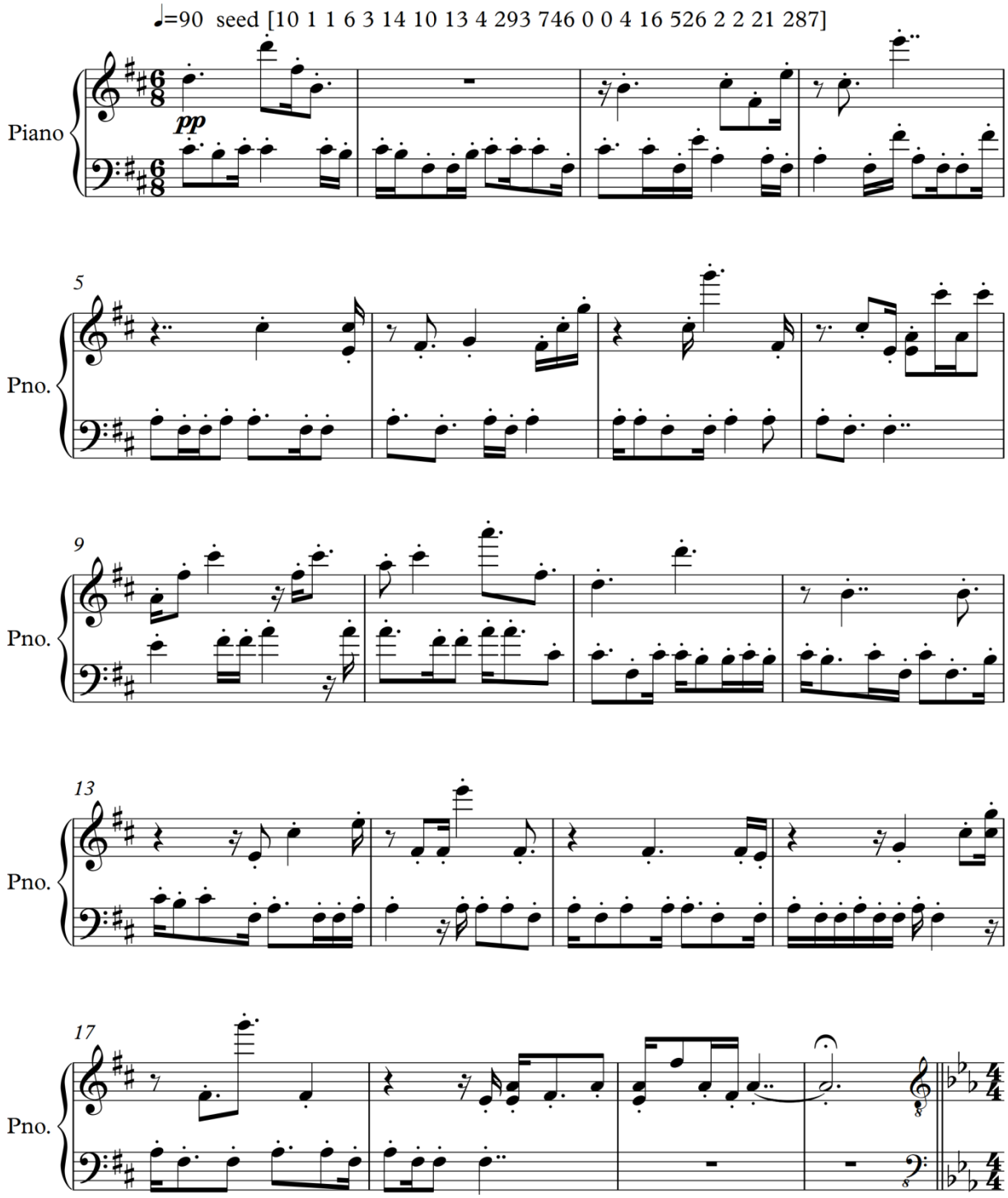
13, "TAG manic Is Tagged 0.760000 1 / 1 ";
14, "TAG train Is Tagged 0.760000 1 / 1 ";

Affective Target for Part 6
0, "BIO Bpm Change Precisely 0.000000 1 / 1 ";
1, "BIO Breath Rate Precisely 0.000000 1 / 1 ";
2, "BIO Skin Cndct Precisely 0.000000 1 / 1 ";
3, "BIO Temp change Greater Than 0.300000 1 / 1 ";
4, "EEG Meditation Greater Than 0.300000 1 / 1 ";
5, "EEG LT Exctmnt Greater Than 0.300000 1 / 1 ";
6, "EEG Engagemnt Greater Than 1.235272 1 / 1 ";
7, "LIK Repeatblty Greater Than 1.109300 1 / 1 ";
8, "LIK Thematicism Close to 0.000000 1 / 1 ";
9, "LIK Cadential Greater Than 0.760000 1 / 1 ";
11, "TAG demon Is Tagged 0.760000 1 / 1 ";
12, "TAG poetry Is Tagged 0.760000 1 / 1 ";
13, "TAG manic Is Tagged 0.760000 1 / 1 ";
14, "TAG train Is Tagged 0.760000 1 / 1 ";

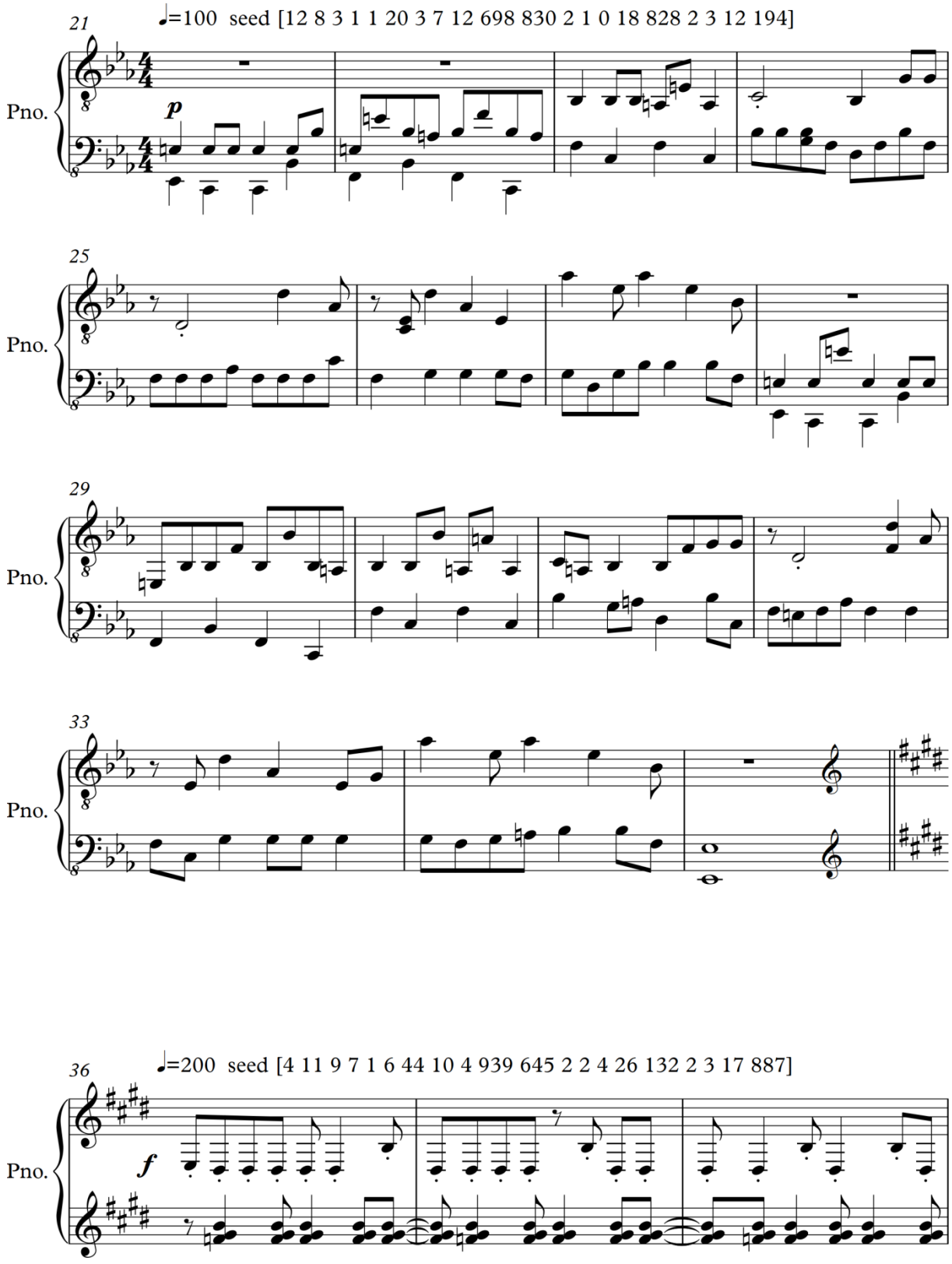
Affective Target for Part 7
0, "BIO Bpm Change Close to -0.211896 1 / 1 ";
1, "BIO Skin Cndct Close to -0.211896 1 / 1 ";
2, "BIO Breath Rate Greater Than 0.140000 1 / 1 ";
3, "BIO Temp change Less Than 0.000000 1 / 1 ";
4, "EEG Meditation Greater Than 0.362000 1 / 1 ";
5, "EEG ST Exctmnt Less Than 0.000000 1 / 1 ";
6, "LIK Rhthm Complex Less Than 0.000000 1 / 1 ";
7, "LIK Affective Close to 0.817000 1 / 1 ";
8, "TAG smooth Is Tagged 0.817000 1 / 1 ";
9, "TAG poetry Is Tagged 0.817000 1 / 1 ";
11, "TAG death Is Tagged 0.817000 1 / 1 ";
12, "TAG fear Is Tagged 0.817000 1 / 1 ";

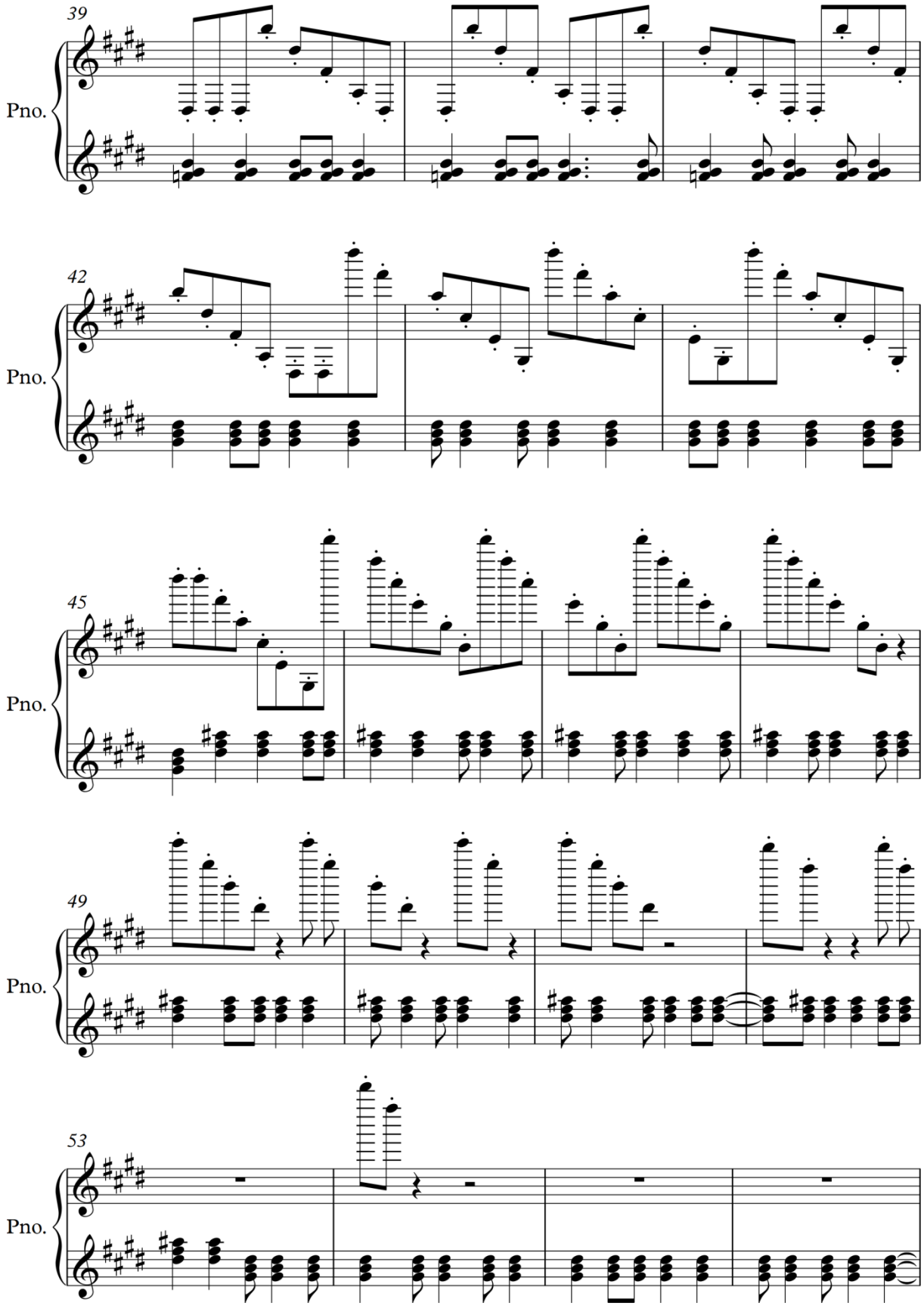
Moloch! - Metropolis

Generated by the Affective Algorithmic Composer
by Daniel Pitman 2015

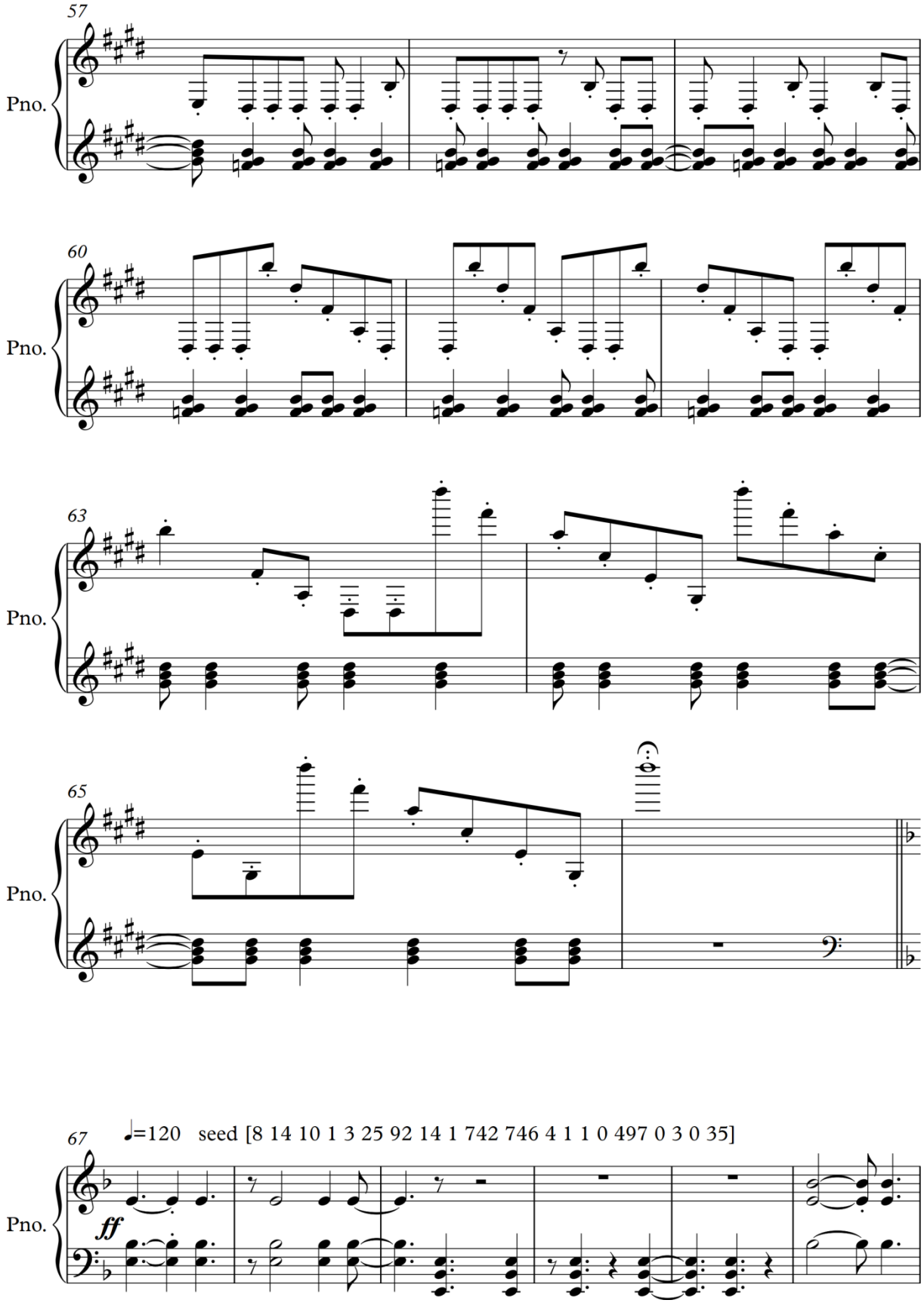


2

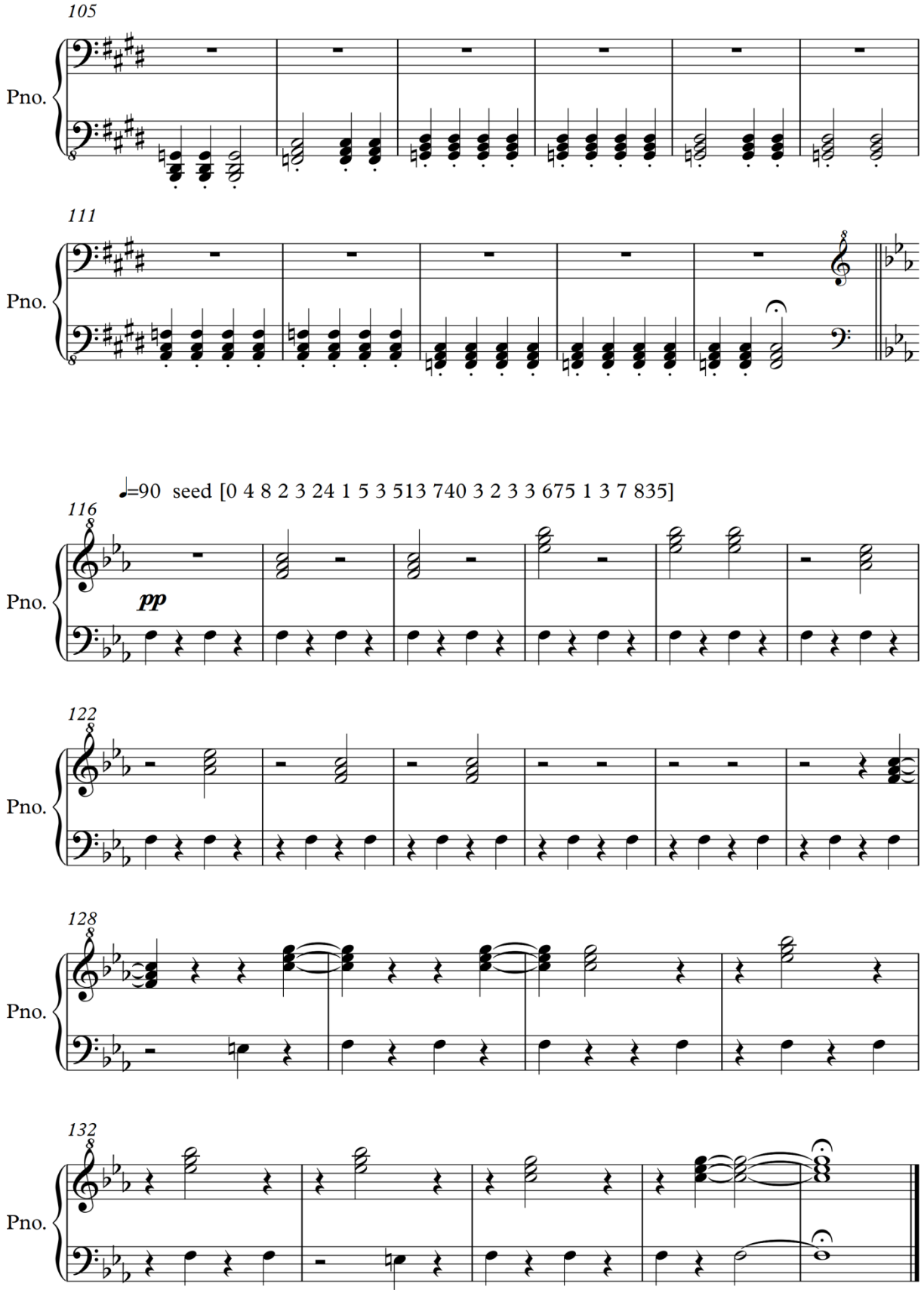

21
♩=100 seed [12 8 3 1 1 20 3 7 12 698 830 2 1 0 18 828 2 3 12 194]
Pno.
p
25
Pno.
29
Pno.
33
Pno.
36
♩=200 seed [4 11 9 7 1 6 44 10 4 939 645 2 2 4 26 132 2 3 17 887]
Pno.
f


39
Pno.
42
Pno.
45
Pno.
49
Pno.
53
Pno.

4


57
Pno.
60
Pno.
63
Pno.
65
Pno.
67
♩=120 seed [8 14 10 1 3 25 92 14 1 742 746 4 1 1 0 497 0 3 0 35]
Pno.
ff

6


105
Pno.
111
Pno.
♩=90 seed [0 4 8 2 3 24 1 5 3 513 740 3 2 3 3 675 1 3 7 835]
116
Pno.
pp
122
Pno.
128
Pno.
132
Pno.

Maria's Dance

(Starting 1:30.11) A robot designed to imitate Maria, the prophet of the workers, is sent to seduce and corrupt the wealthy and powerful men of Metropolis with a hypnotic dance. Simultaneously, bed ridden Ferder suffers fever induced hallucinations of the figure of Death and a coming apocalypse.

The combination of hypnotic dancing themes and undertones of madness and apocalyptic visions makes this a particularly interesting challenge for affective target design.

Playlist
1, 3 8 0 5 0 19 49 5 2 852 569 0 0 4 17 619 1 2 13 47;
2, 1 7 0 6 2 5 67 8 11 558 901 2 1 2 6 929 1 1 11 702;
3, 15 13 14 8 2 20 77 8 6 192 593 1 0 5 16 863 1 3 19 180;
4, 6 0 12 7 1 5 33 11 12 126 420 1 2 3 12 617 1 3 21 635;
5, 2 14 13 6 3 14 5 15 10 284 673 3 2 0 4 191 0 1 11 840;
6, 12 15 14 0 2 19 99 12 5 76 337 2 2 2 21 732 2 3 9 955;
7, 1 7 2 7 2 22 26 14 12 339 881 3 0 4 5 558 1 1 9 310;
8, 7 12 9 8 2 16 58 1 0 648 448 4 1 1 25 448 0 2 17 201;

Affective Target for Part 1
0, "BIO Bpm Change Less Than 0.100000 1 / 1 ";
1, "BIO Skin Cndct Less Than 0.100000 1 / 1 ";
2, "EEG ST Exctmnt Close to 0.190000 1 / 1 ";
3, "LIK Dancey Close to 0.209800 1 / 1 ";
4, "LIK Positivity Close to 0.000000 1 / 1 ";
5, "TAG bright Is Tagged 0.650000 1 / 1 ";
6, "TAG glide Is Tagged 0.650000 1 / 1 ";
7, "TAG green Is Tagged 0.650000 1 / 1 ";
8, "TAG quiet Is Tagged 0.650000 1 / 1 ";
9, "LIK Repeatblty Close to 0.800000 1 / 1 ";
10, "TAG trickle Is Tagged 0.800000 1 / 1 ";

Affective Target for Part 2
0, "BIO Hrv Change Close to 0.450000 1 / 1 ";
1, "BIO Breath Rate Less Than 0.000000 1 / 1 ";
2, "BIO Temp change Less Than 0.000000 1 / 1 ";
3, "BIO Skin Cndct Greater Than 0.137000 1 / 1 ";
4, "EEG ST Exctmnt Greater Than 0.378000 1 / 1 ";
5, "EEG Engagemnt Greater Than 0.378000 1 / 1 ";
6, "EEG Meditation Greater Than 0.000000 1 / 1 ";
7, "LIK Rhthm Complex Close to 0.291000 1 / 1 ";
8, "LIK Outstanding Close to 0.588000 1 / 1 ";
9, "LIK Positivity Close to 0.000000 1 / 1 ";

11, "TAG deity Is Tagged 0.384000 1 / 1 ";
12, "TAG machine Is Tagged 0.384000 1 / 1 ";
13, "TAG white Is Tagged 0.384000 1 / 1 ";
14, "TAG quiet Is Tagged 0.384000 1 / 1 ";
15, "TAG sky Is Tagged 0.384000 1 / 1 ";

Affective Target for Part 3
0, "BIO Hrv Change Close to 0.461000 1 / 1 ";
1, "BIO Skin Cndct Close to 0.461000 1 / 1 ";
2, "BIO Temp change Less Than 0.000000 1 / 1 ";
3, "EEG ST Exctmnt Close to 0.297000 1 / 1 ";
4, "LIK Dancey Greater Than 0.833000 1 / 1 ";
5, "LIK Rhthm Complex Less Than 0.440000 1 / 1 ";
6, "LIK Thematicism Close to 0.503000 1 / 1 ";
7, "EEG Frustration Less Than 0.000000 1 / 1 ";
8, "LIK Positivity Close to 0.000000 1 / 1 ";
9, "TAG celebration Is Tagged 0.000000 1 / 1 ";
11, "TAG yellow Is Tagged 0.000000 1 / 1 ";
12, "TAG rising Is Tagged 0.000000 1 / 1 ";
13, "TAG sky Is Tagged 0.000000 1 / 1 ";

Affective Target for Part 4
0, "BIO Hrv Change Greater Than 0.290000 1 / 1 ";
1, "BIO Breath Sizes Greater Than 0.290000 1 / 1 ";
2, "BIO Skin Cndct Greater Than 0.610000 1 / 1 ";
3, "EEG Frustration Greater Than 0.610000 1 / 1 ";
4, "EEG Meditation Close to 0.610000 1 / 1 ";
5, "EEG ST Exctmnt Less Than 0.279000 1 / 1 ";
6, "EEG Engagemnt Greater Than 0.279000 1 / 1 ";
7, "LIK Thematicism Greater Than 0.940000 1 / 1 ";
8, "LIK Cadential Greater Than 0.600000 1 / 1 ";
9, "LIK Positivity Less Than 0.000000 1 / 1 ";
11, "TAG demon Is Tagged 0.000000 1 / 1 ";
12, "TAG poetry Is Tagged 0.000000 1 / 1 ";
13, "TAG loud Is Tagged 0.000000 1 / 1 ";

Affective Target for Part 5
0, "BIO Hrv Change Close to 0.000000 1 / 1 ";
1, "BIO Breath Rate Close to 0.000000 1 / 1 ";
2, "BIO Skin Cndct Close to 0.000000 1 / 1 ";
3, "EEG LT Exctmnt Close to 0.000000 1 / 1 ";
4, "EEG Engagemnt Close to 1.235272 1 / 1 ";
5, "LIK Dancey Close to 1.100000 1 / 1 ";
6, "LIK Thematicism Close to 1.100000 1 / 1 ";
7, "LIK Cadential Close to 1.100000 1 / 1 ";
8, "LIK Outstanding Close to 1.100000 1 / 1 ";
9, "LIK Positivity Less Than 0.011000 1 / 1 ";

11, "TAG gushing Is Tagged 0.011000 1 / 1 ";
12, "TAG math Is Tagged 0.011000 1 / 1 ";
13, "TAG hammer Is Tagged 0.011000 1 / 1 ";
14, "TAG hate Is Tagged 0.011000 1 / 1 ";

Affective Target for Part 6
0, "BIO Bpm Change Greater Than 0.649000 1 / 1 ";
1, "BIO Breath Rate Less Than 0.010000 1 / 1 ";
2, "BIO Skin Cndct Close to 1.235272 1 / 1 ";
3, "EEG Frustration Less Than 0.150000 1 / 1 ";
4, "EEG ST Exctmnt Greater Than 0.750000 1 / 1 ";
5, "EEG Engagemnt Greater Than 0.750000 1 / 1 ";
6, "LIK Dancey Greater Than 0.750000 1 / 1 ";
7, "LIK Repeatblty Greater Than 0.750000 1 / 1 ";
8, "LIK Chordal Greater Than 0.400000 1 / 1 ";
9, "LIK Affective Close to 0.270000 1 / 1 ";
11, "TAG evil Is Tagged 0.270000 1 / 1 ";
12, "TAG deity Is Tagged 0.270000 1 / 1 ";
13, "TAG math Is Tagged 0.270000 1 / 1 ";
14, "TAG loud Is Tagged 0.270000 1 / 1 ";
15, "TAG manic Is Tagged 0.270000 1 / 1 ";

Affective Target for Part 7
0, "BIO Bpm Change Greater Than 0.960000 1 / 1 ";
1, "BIO Breath Sizes Greater Than 0.960000 1 / 1 ";
2, "BIO Skin Cndct Greater Than 0.960000 1 / 1 ";
3, "EEG Frustration Less Than 0.000000 1 / 1 ";
4, "EEG ST Exctmnt Greater Than 0.960000 1 / 1 ";
5, "LIK Thematicism Greater Than 0.960000 1 / 1 ";
6, "LIK Chordal Greater Than 0.960000 1 / 1 ";
7, "LIK Outstanding Greater Than 0.960000 1 / 1 ";
8, "LIK Positivity Close to -0.211896 1 / 1 ";
9, "LIK Affective Greater Than 0.940000 1 / 1 ";
11, "TAG choppy Is Tagged 0.940000 1 / 1 ";
12, "TAG hammer Is Tagged 0.940000 1 / 1 ";
13, "TAG train Is Tagged 0.940000 1 / 1 ";
14, "TAG horse Is Tagged 0.940000 1 / 1 ";

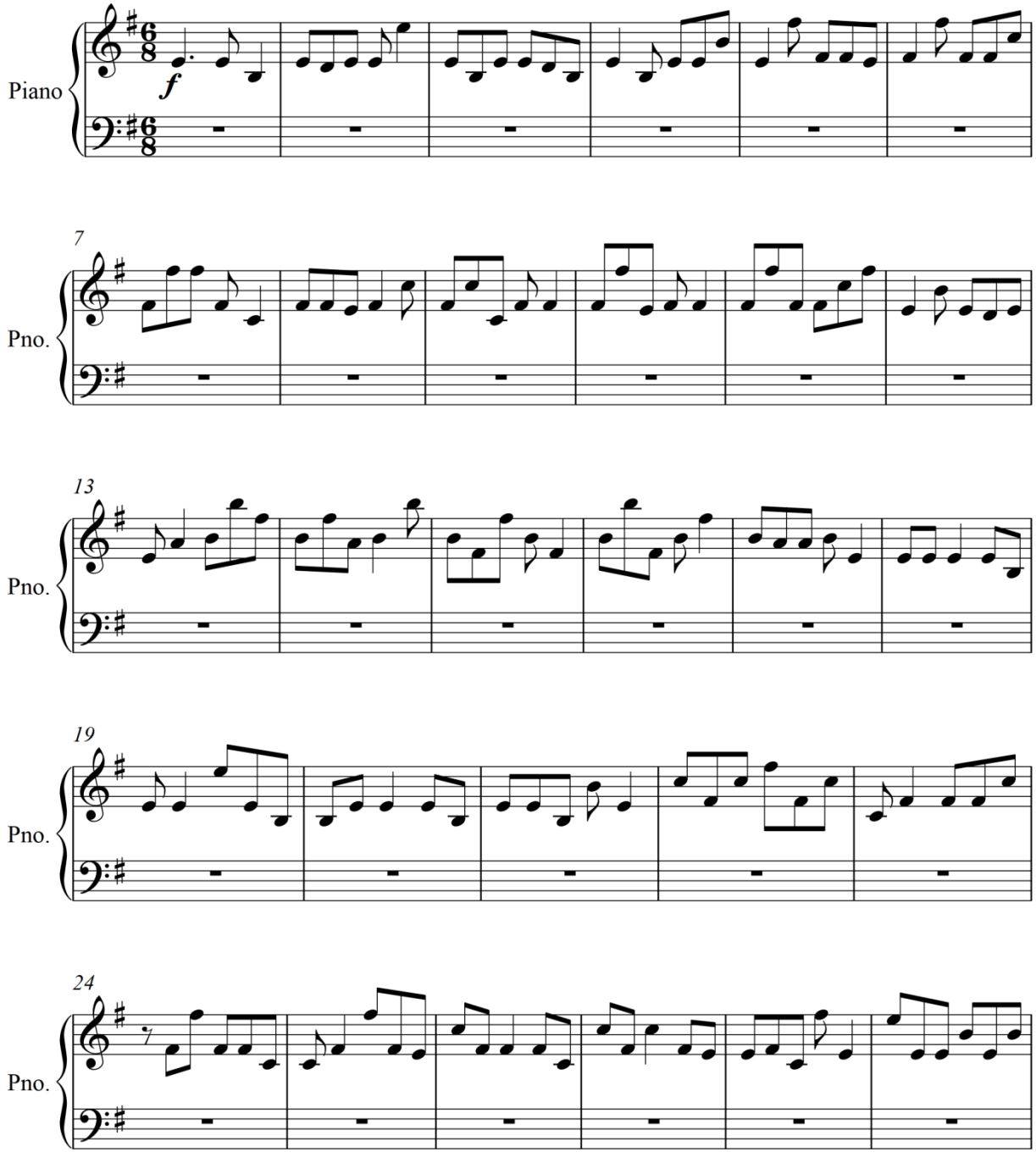
Affective Target for Part 8
0, "TAG death Is Tagged -0.211896 1 / 1 ";
1, "TAG poetry Is Tagged -0.211896 1 / 1 ";
2, "TAG hammer Is Tagged -0.211896 1 / 1 ";
3, "BIO Skin Cndct Close to 0.630000 1 / 1 ";
4, "BIO Bpm Change Close to 0.630000 1 / 1 ";
5, "EEG Frustration Close to -0.211896 1 / 1 ";
6, "LIK Positivity Close to -0.211896 1 / 1 ";
7, "TAG dark Is Tagged -0.211896 1 / 1 ";

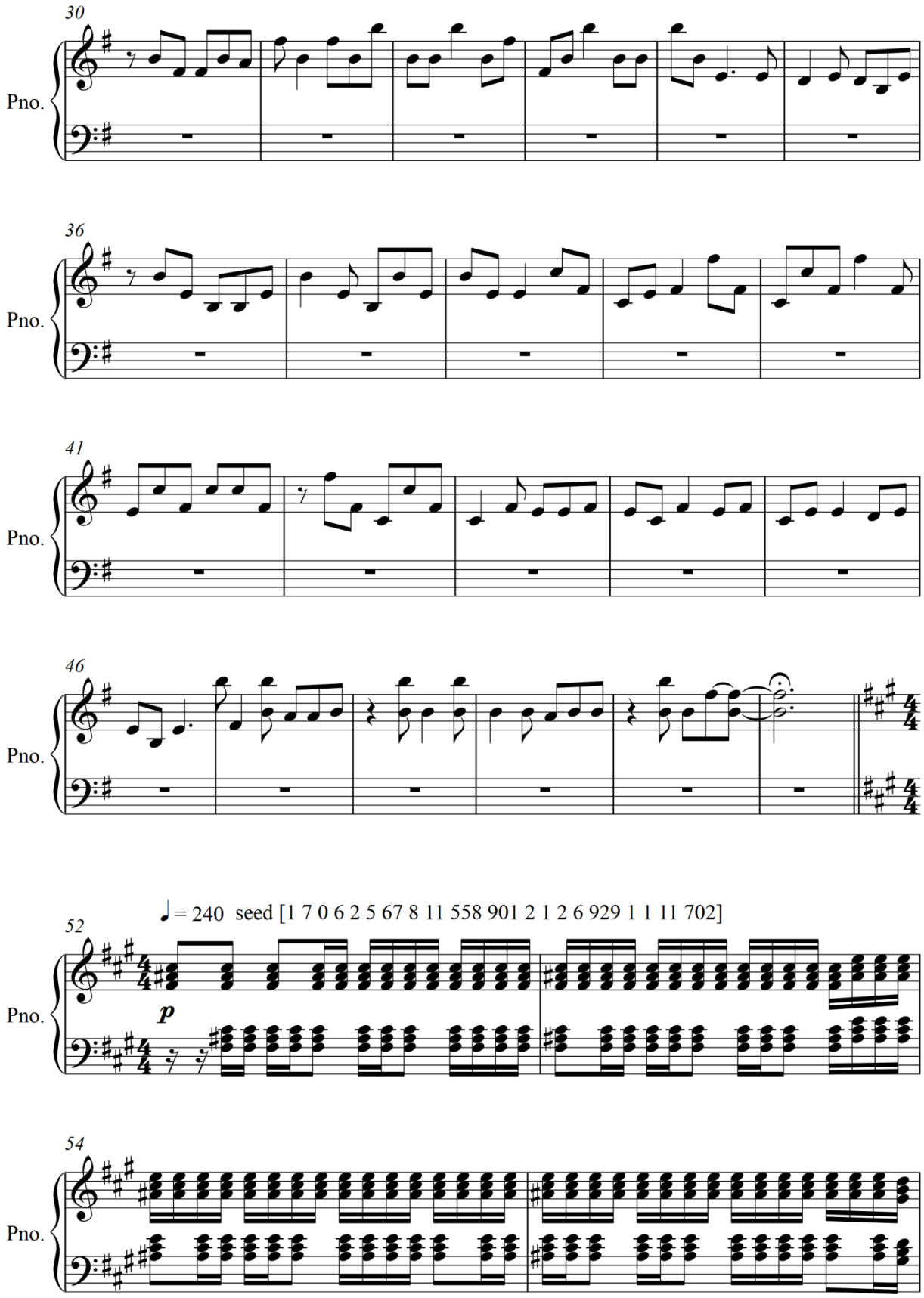
8, "LIK Rhthm Complex Close to 0.300000 1 / 1 ";
9, "TAG loud Is Tagged 0.300000 1 / 1 ";
11, "LIK Cadential Greater Than 0.300000 1 / 1 ";

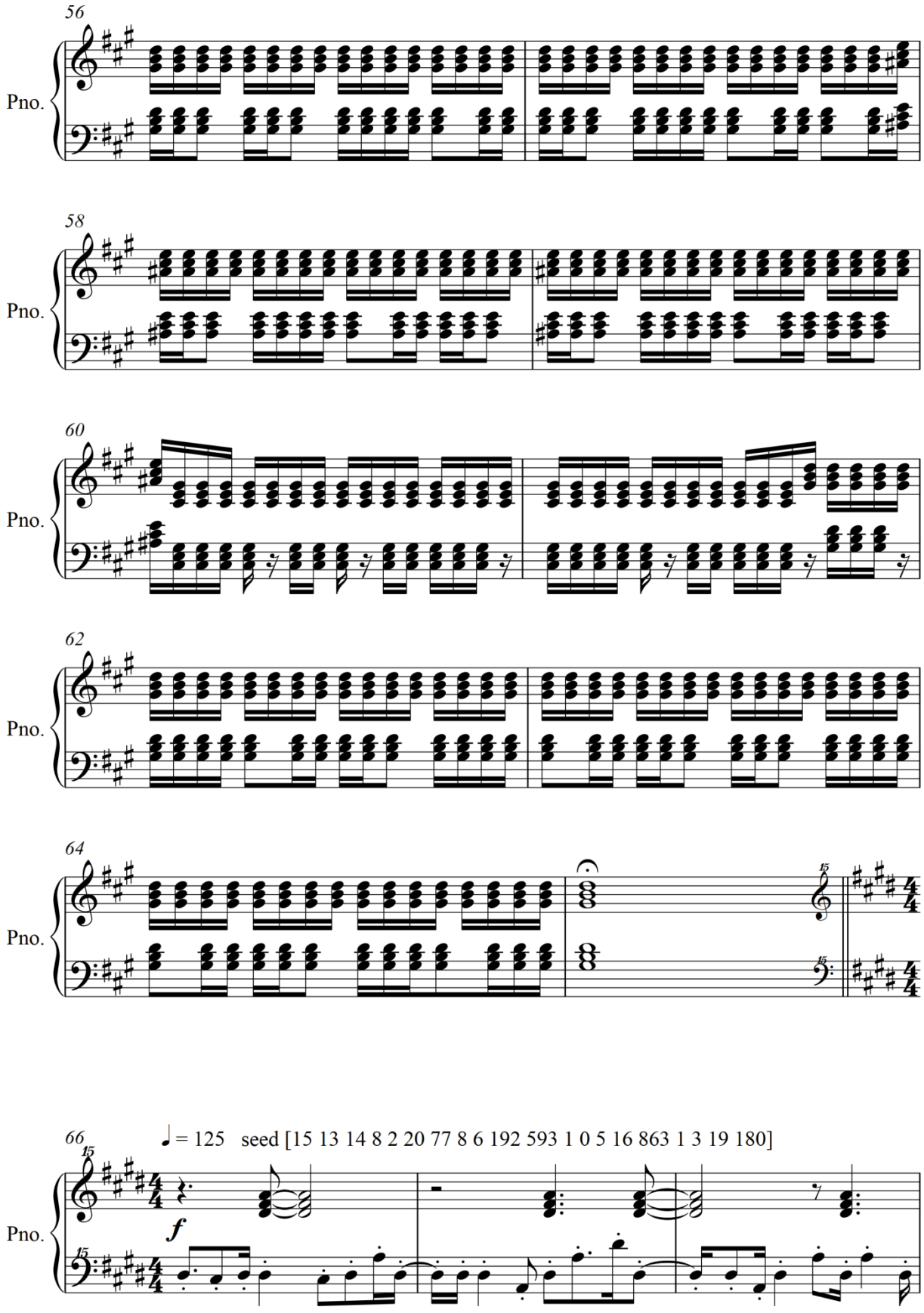
Maria's Dance - Metropolis

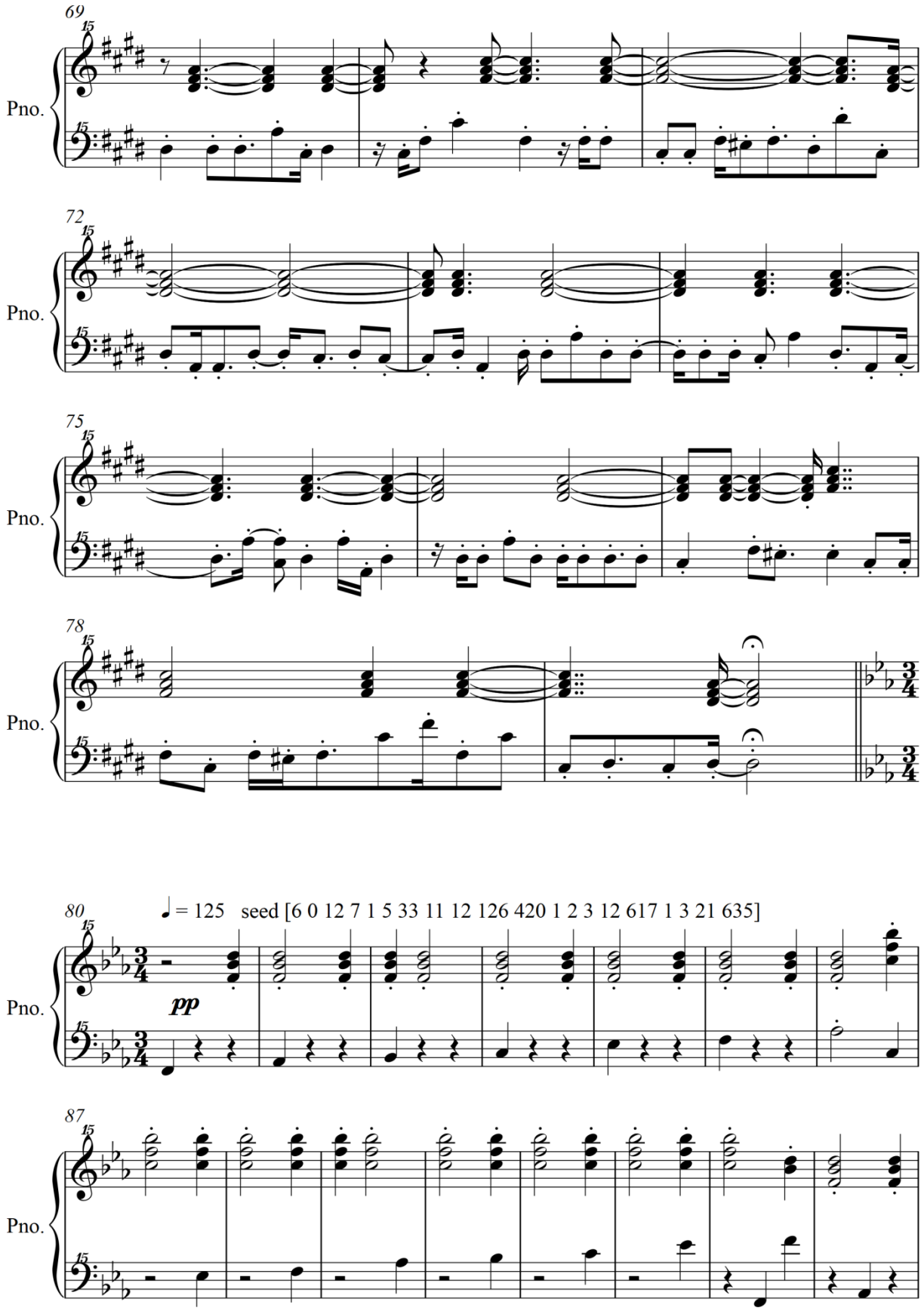
Generated by the Affective Algorithmic Composer
by Daniel Pitman 2015

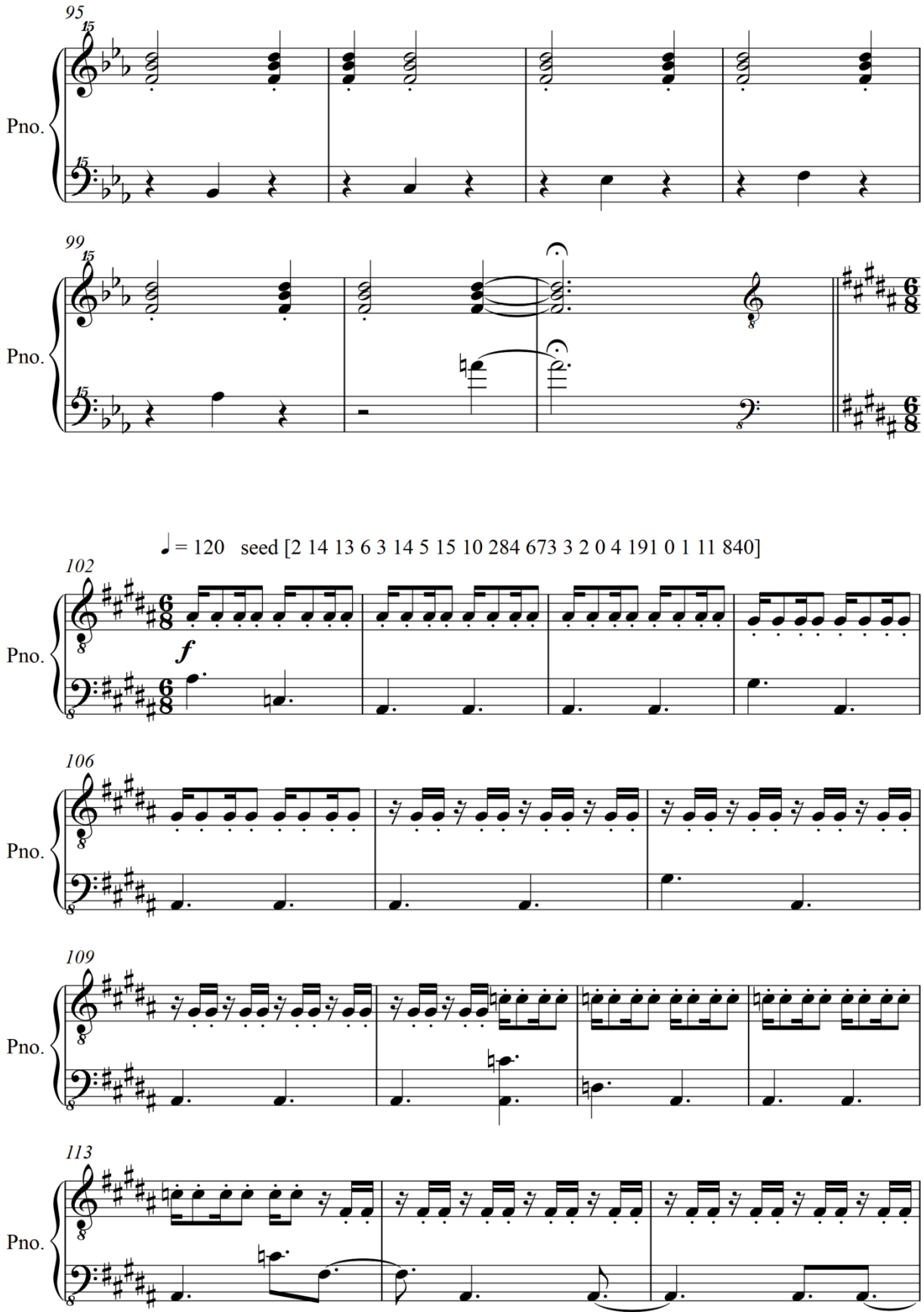
Ignore Common Accenting
♩ = 150 seed [3 8 0 5 0 19 49 5 2 852 569 0 0 4 17 619 1 2 13 47]

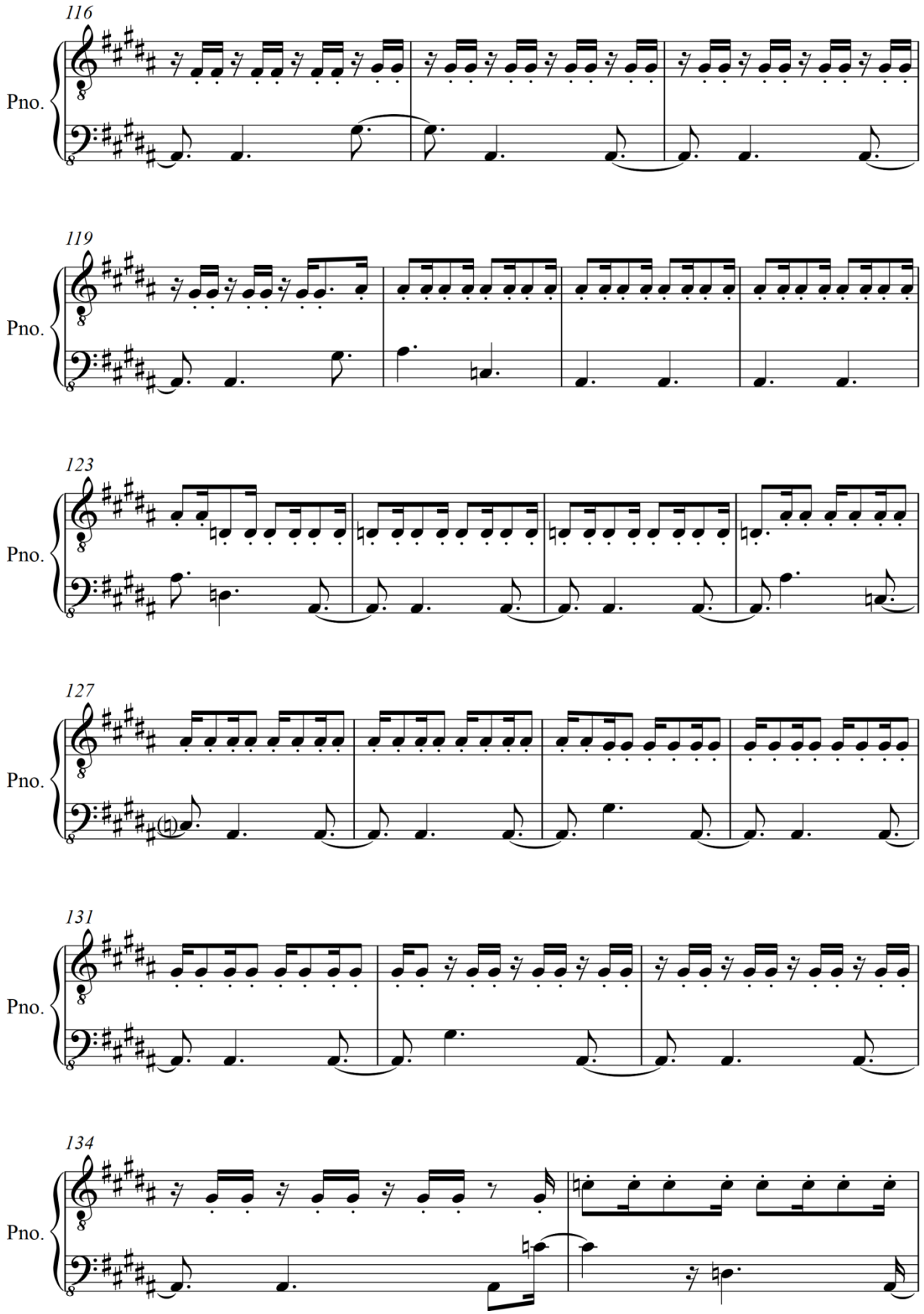


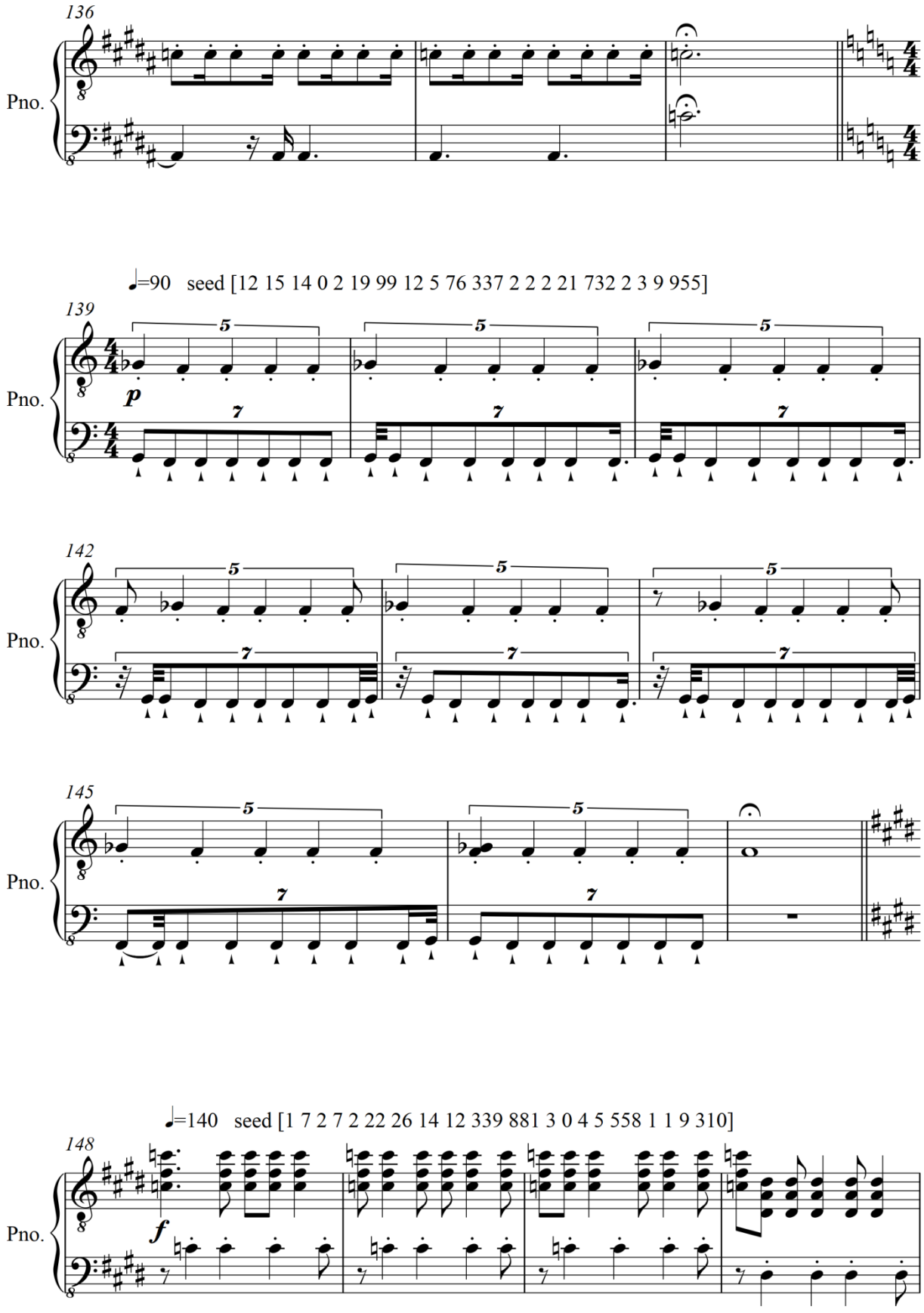

30
Pno.
36
Pno.
41
Pno.
46
Pno.
♩ = 240 seed [1 7 0 6 2 5 67 8 11 558 901 2 1 2 6 929 1 1 11 702]
52
Pno.
p
54
Pno.

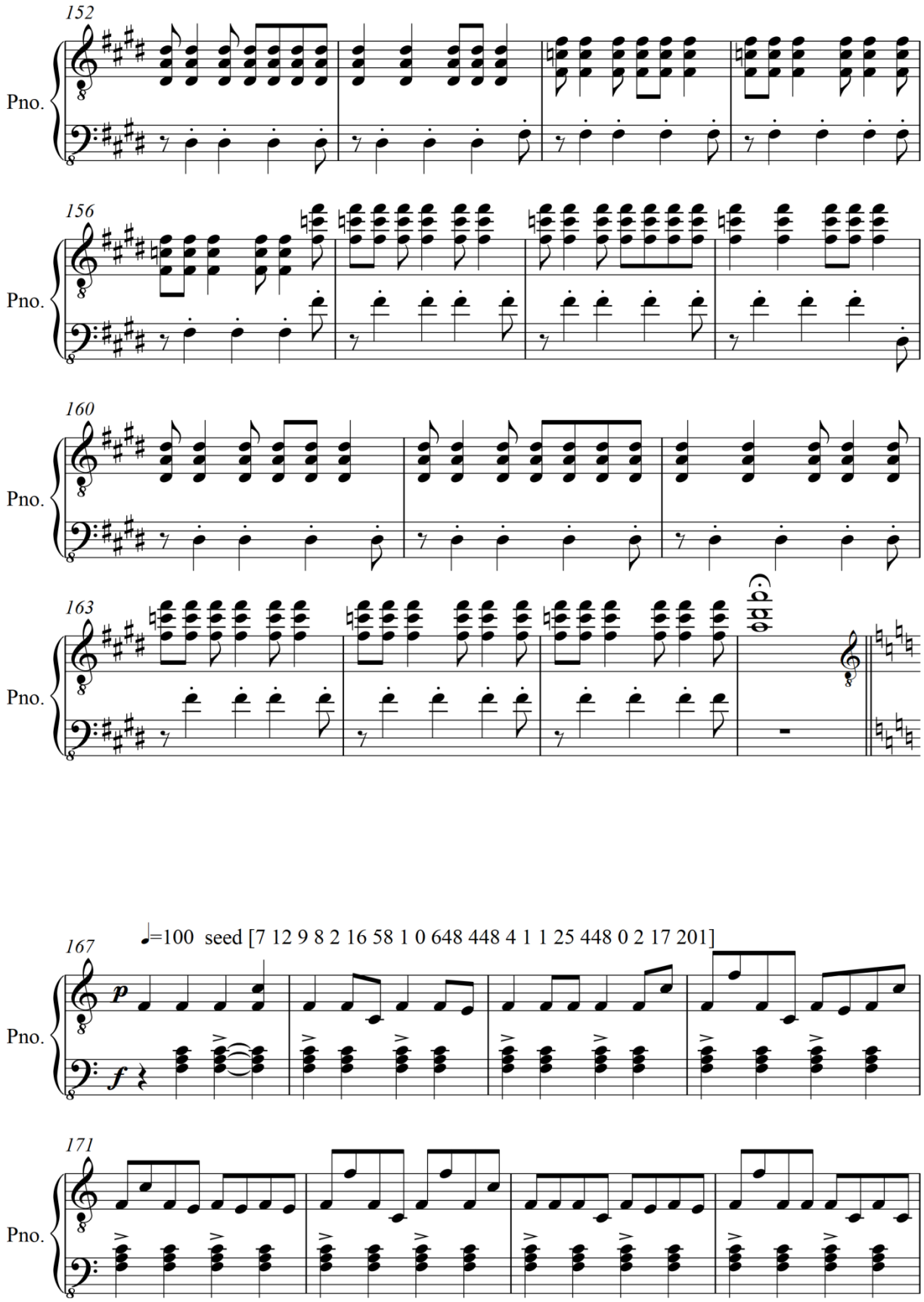

56
Pno.
58
Pno.
60
Pno.
62
Pno.
64
Pno.
66
♩ = 125 seed [15 13 14 8 2 20 77 8 6 192 593 1 0 5 16 863 1 3 19 180]
Pno.
f

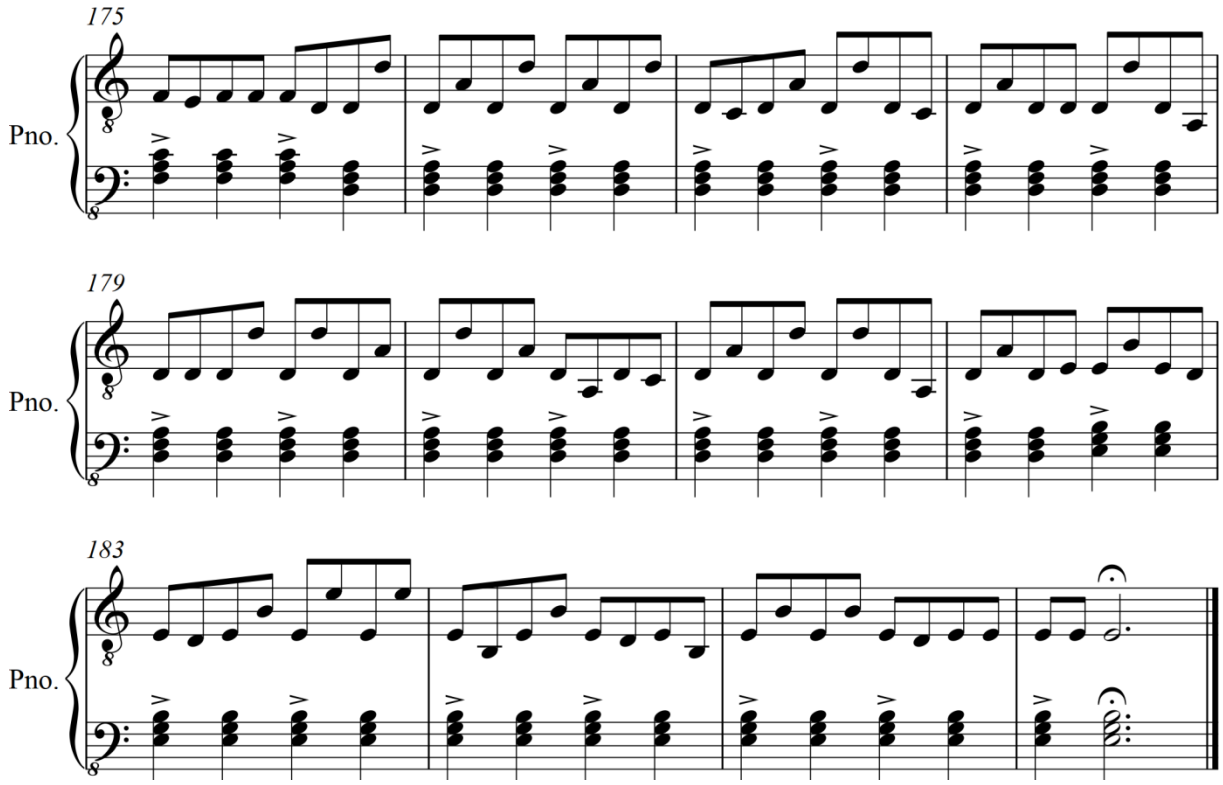

69
Pno.
72
Pno.
75
Pno.
78
Pno.
80
♩ = 125 seed [6 0 12 7 1 5 33 11 12 126 420 1 2 3 12 617 1 3 21 635]
pp
Pno.
87
Pno.


95
Pno.
99
Pno.
♩ = 120 seed [2 14 13 6 3 14 5 15 10 284 673 3 2 0 4 191 0 1 11 840]
102
Pno.
f
106
Pno.
109
Pno.
113
Pno.


116
Pno.
119
Pno.
123
Pno.
127
Pno.
131
Pno.
134
Pno.


136
Pno.
♩=90 seed [12 15 14 0 2 19 99 12 5 76 337 2 2 2 21 732 2 3 9 955]
139
Pno.
p
142
Pno.
145
Pno.
♩=140 seed [1 7 2 7 2 22 26 14 12 339 881 3 0 4 5 558 1 1 9 310]
148
Pno.
f


152
Pno.
156
Pno.
160
Pno.
163
Pno.
♩=100 seed [7 12 9 8 2 16 58 1 0 648 448 4 1 1 25 448 0 2 17 201]
167
Pno.
171
Pno.


175
Pno.
179
Pno.
183
Pno.

Reunion/Final Scene

(Starting 2.21:58) Rothwang the scientist, driven by an insane confusion between his now destroyed robot and Maria herself, falls to his death after fighting with Freder. Maria and Freder are finally reunited and are able to mediate a final truce between the workers and the city's elite.

For this last piece, the AACr's approach to more dramatic and emotional affect states is tested with the target affects designed to mimic or reinforce the subtler emotional drama of the couple as lovers and the two factions struggling to find grounds for peace.

Where the other pieces have been generated without limitations of producing music that is playable by a human performer, this piece was specifically limited to be playable by human performers.

Playlist
1, 15 11 13 5 3 5 76 3 5 610 801 2 1 1 1 362 2 0 15 420;
2, 3 1 15 6 1 16 5 2 0 569 354 5 0 3 20 875 0 2 15 262;
3, 3 3 13 3 0 1 64 15 5 41 649 5 1 1 20 16 2 1 20 802;
4, 15 3 13 7 2 17 60 7 1 649 433 5 0 5 3 667 1 1 8 267;
5, 5 0 11 7 3 25 94 3 5 812 601 4 2 1 26 432 1 0 24 871;
6, 11 3 15 2 0 26 72 13 3 410 249 1 2 2 11 6 2 3 24 198;
7, 5 5 13 6 1 11 9 13 10 555 779 5 2 4 0 781 2 2 20 520;
8, 2 6 1 7 0 13 90 11 3 630 930 5 0 2 10 303 0 3 23 579;
9, 1 0 14 0 1 16 38 9 10 95 273 4 1 2 9 899 0 1 3 224;

Affective Target for Part 1
0, "BIO Bpm Change Close to 0.000000 1 / 1 ";
1, "BIO Skin Cndct Close to 0.000000 1 / 1 ";
2, "BIO Breath Rate Close to 0.250000 1 / 1 ";
3, "EEG ST Exctmnt Close to 0.550000 1 / 1 ";
4, "EEG Frustration Close to -0.211896 1 / 1 ";
5, "LIK Outstanding Close to 0.250000 1 / 1 ";
6, "LIK Positivity Close to -0.211896 1 / 1 ";
7, "TAG machine Is Tagged -0.211896 1 / 1 ";
8, "TAG healing Is Tagged -0.211896 1 / 1 ";
9, "TAG falling Is Tagged -0.211896 1 / 1 ";
10, "TAG driven Is Tagged -0.211896 1 / 1 ";

Affective Target for Part 2
0, "BIO Bpm Change Close to 0.700000 1 / 1 ";
1, "BIO Skin Cndct Close to 0.700000 1 / 1 ";

2, "EEG ST Exctmnt Close to 0.700000 1 / 1 ";
3, "EEG Engagemnt Close to 0.700000 1 / 1 ";
4, "EEG Meditation Close to -0.054000 1 / 1 ";
5, "LIK Thematicism Close to 1.235272 1 / 1 ";
6, "TAG rising Is Tagged 1.235272 1 / 1 ";
7, "TAG stress Is Tagged 1.235272 1 / 1 ";
8, "TAG driven Is Tagged 1.235272 1 / 1 ";
9, "TAG good Is Tagged 1.235272 1 / 1 ";

Affective Target for Part 3
0, "BIO Skin Cndct Close to 1.235272 1 / 1 ";
1, "BIO hrv Percent Close to 1.235272 1 / 1 ";
2, "EEG Frustration Close to 1.235272 1 / 1 ";
3, "EEG ST Exctmnt Close to 1.235272 1 / 1 ";
4, "LIK Thematicism Close to 1.235272 1 / 1 ";
5, "LIK Cadential Close to 1.235272 1 / 1 ";
6, "LIK Positivity Close to 0.000000 1 / 1 ";
7, "LIK Outstanding Close to 1.235272 1 / 1 ";
8, "TAG falling Is Tagged 1.235272 1 / 1 ";
9, "TAG destruction Is Tagged 1.235272 1 / 1 ";
11, "TAG trickle Is Tagged 1.235272 1 / 1 ";

Affective Target for Part 4
0, "TAG love Is Tagged 1.235272 1 / 1 ";
1, "TAG healing Is Tagged 1.235272 1 / 1 ";
2, "TAG good Is Tagged 1.235272 1 / 1 ";
3, "TAG smooth Is Tagged 1.235272 1 / 1 ";
4, "BIO Bpm Change Close to -0.170000 1 / 1 ";
5, "BIO Skin Cndct Close to 0.000000 1 / 1 ";
6, "EEG Engagemnt Close to 0.720000 1 / 1 ";
7, "LIK Outstanding Close to 0.720000 1 / 1 ";
8, "LIK Positivity Close to 0.210000 1 / 1 ";
9, "LIK Thematicism Close to 0.730000 1 / 1 ";

Affective Target for Part 5
0, "TAG gushing Is Tagged 1.000000 1 / 1 ";
1, "TAG stress Is Tagged 1.000000 1 / 1 ";
2, "TAG poetry Is Tagged 1.000000 1 / 1 ";

Affective Target for Part 6
0, "TAG poetry Is Tagged 1.000000 1 / 1 ";
1, "TAG healing Is Tagged 1.000000 1 / 1 ";
2, "TAG easy Is Tagged 1.000000 1 / 1 ";
3, "TAG earth Is Tagged 1.000000 1 / 1 ";
4, "TAG good Is Tagged 1.000000 1 / 1 ";
5, "TAG green Is Tagged 1.000000 1 / 1 ";
6, "BIO Bpm Change Close to 0.000000 1 / 1 ";

7, "BIO Skin Cndct Close to 0.000000 1 / 1 ";
8, "BIO Breath Rate Close to 0.000000 1 / 1 ";
9, "LIK Rhthm Complex Close to 0.460000 1 / 1 ";
11, "LIK Thematicism Close to 0.810000 1 / 1 ";
12, "LIK Positivity Close to 0.810000 1 / 1 ";
13, "EEG Engagemnt Close to 0.810000 1 / 1 ";

Affective Target for Part 7
0, "TAG poetry Is Tagged 1.000000 1 / 1 ";
1, "TAG healing Is Tagged 1.000000 1 / 1 ";
2, "TAG easy Is Tagged 1.000000 1 / 1 ";
3, "TAG earth Is Tagged 1.000000 1 / 1 ";
4, "TAG good Is Tagged 1.000000 1 / 1 ";
5, "TAG green Is Tagged 1.000000 1 / 1 ";
6, "BIO Bpm Change Close to 0.000000 1 / 1 ";
7, "BIO Skin Cndct Close to 0.000000 1 / 1 ";
8, "BIO Breath Rate Close to 0.000000 1 / 1 ";
9, "LIK Rhthm Complex Close to 0.460000 1 / 1 ";
11, "LIK Thematicism Close to 0.810000 1 / 1 ";
12, "LIK Positivity Close to 0.810000 1 / 1 ";
13, "EEG Engagemnt Close to 0.810000 1 / 1 ";

Affective Target for Part 8
0, "TAG love Is Tagged 0.610000 1 / 1 ";
1, "TAG celebration Is Tagged 0.610000 1 / 2 ";
2, "LIK Cadential Greater Than 0.666000 1 / 3 ";
3, "LIK Positivity Greater Than 0.666000 1 / 4 ";
4, "LIK Outstanding Greater Than 0.666000 1 / 5 ";

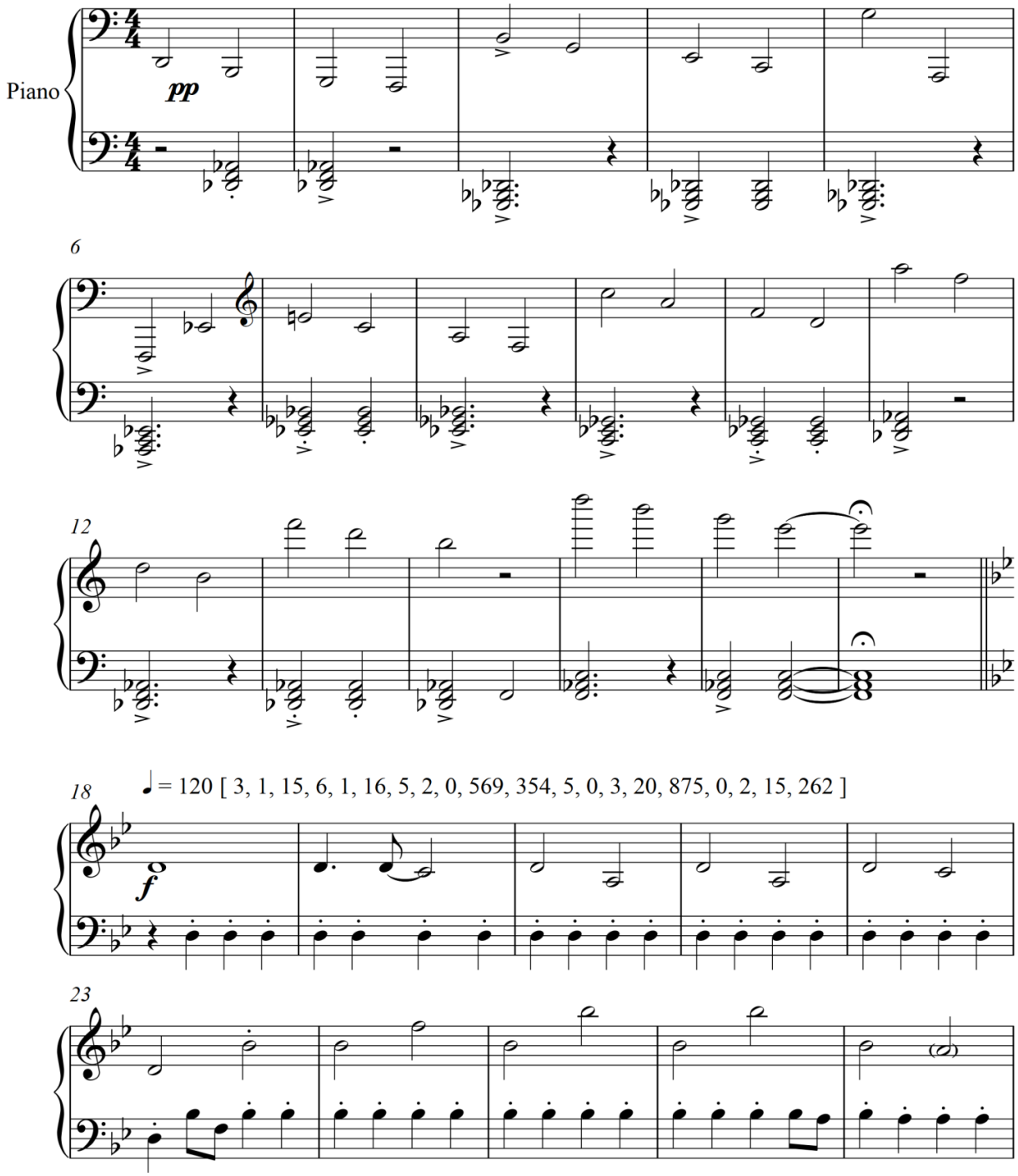
Affective Target for Part 9
0, "BIO hrv Percent Close to 0.520000 1 / 1 ";
1, "EEG LT Exctmnt Close to 0.520000 1 / 2 ";
2, "EEG Engagemnt Close to 0.520000 1 / 3 ";
3, "LIK Dancey Close to 0.520000 1 / 4 ";
4, "LIK Chordal Close to 0.520000 1 / 5 ";
5, "LIK Outstanding Close to 0.520000 1 / 6 ";
6, "LIK Positivity Close to 0.520000 1 / 7 ";
7, "TAG good Is Tagged 0.520000 1 / 8 ";
8, "TAG loud Is Tagged 0.520000 1 / 9 ";
9, "TAG easy Is Tagged 0.520000 1 / 10 ";

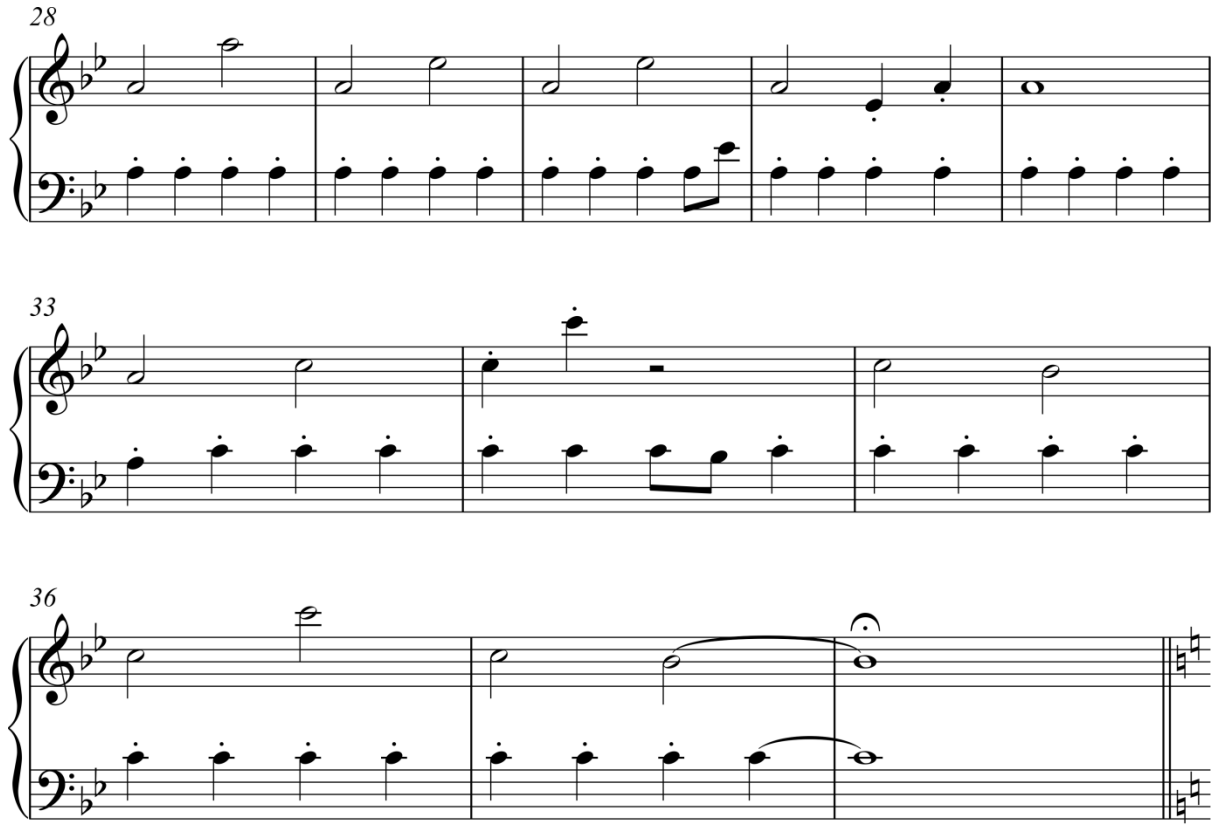
Reunion Scene - Metropolis

Generated by The Affective Algorithmic Composer
by Daniel Pitman 2015

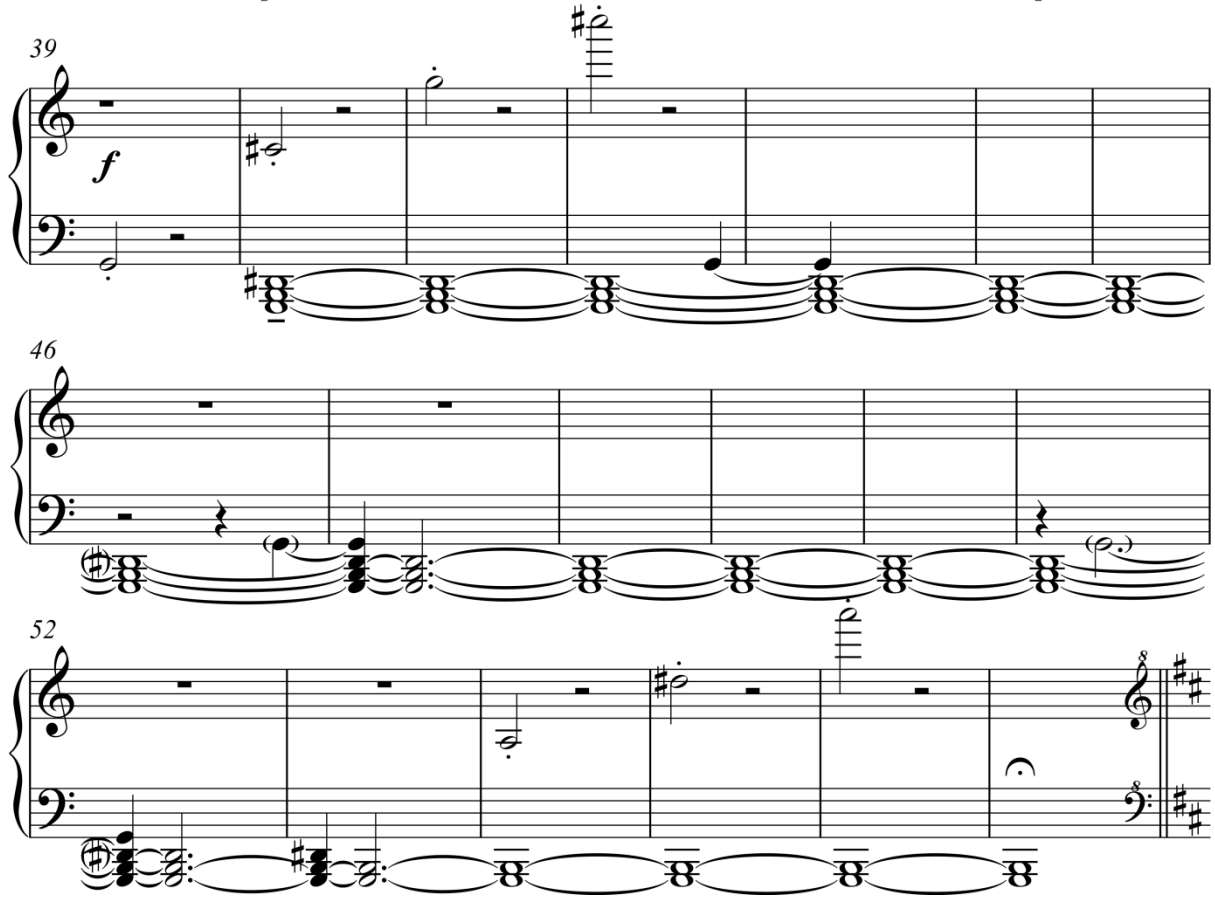
♩ = 90 [15, 11, 13, 5, 3, 5, 76, 3, 5, 610, 801, 2, 1, 1, 1, 362, 2, 0, 15, 420]

Ignore common accenting..

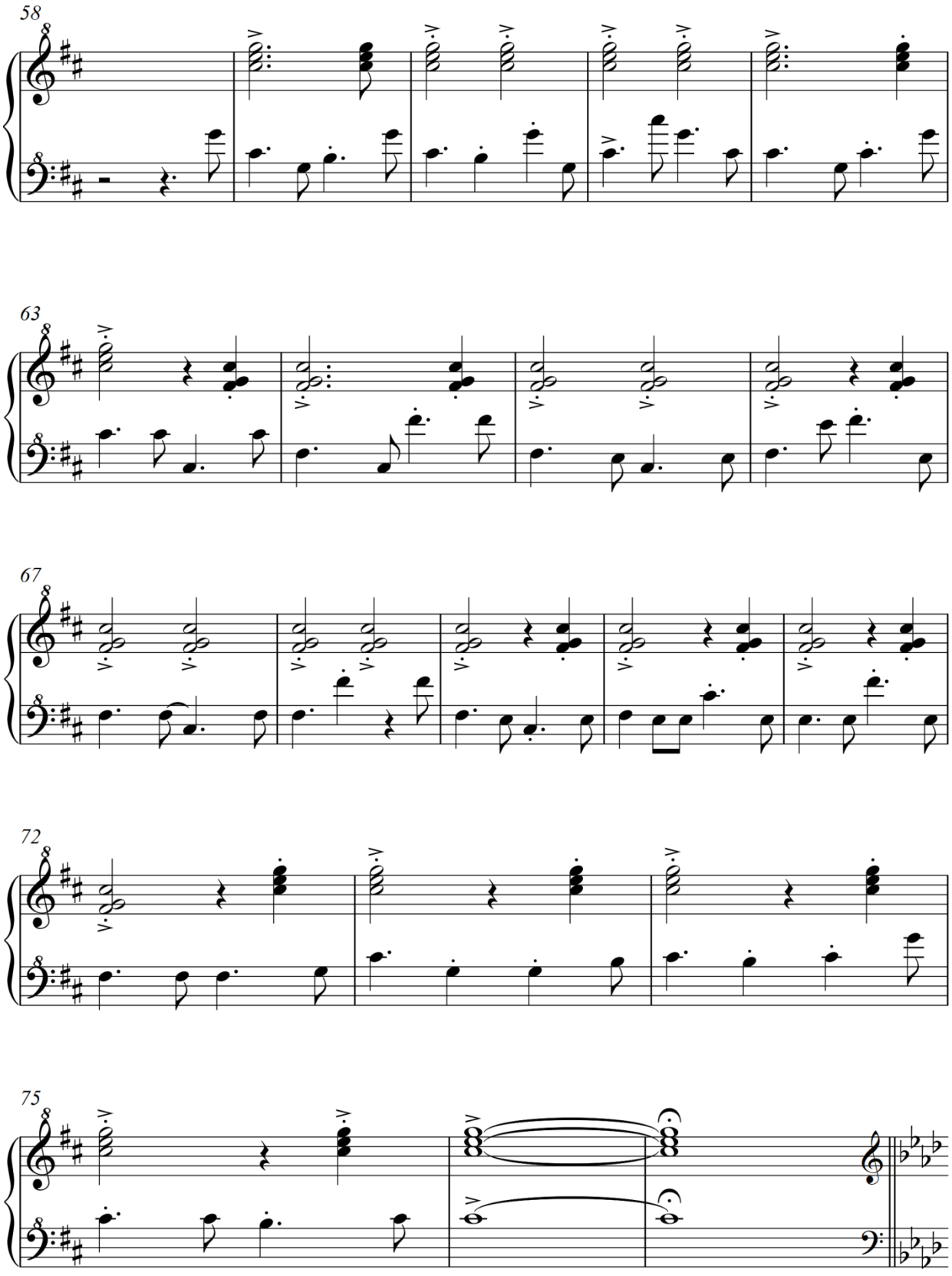




♩ = 150 seed [3, 1, 15, 6, 1, 16, 5, 2, 0, 569, 354, 5, 0, 3, 20, 875, 0, 2, 15, 262]



♩ = 150 seed [15, 3, 13, 7, 2, 17, 60, 7, 1, 649, 433, 5, 0, 5, 3, 667, 1, 1, 8, 267]



78 ♩ = 80 seed [5, 0, 11, 7, 3, 25, 94, 3, 5, 812, 601, 4, 2, 1, 26, 432, 1, 0, 24, 871]

pp

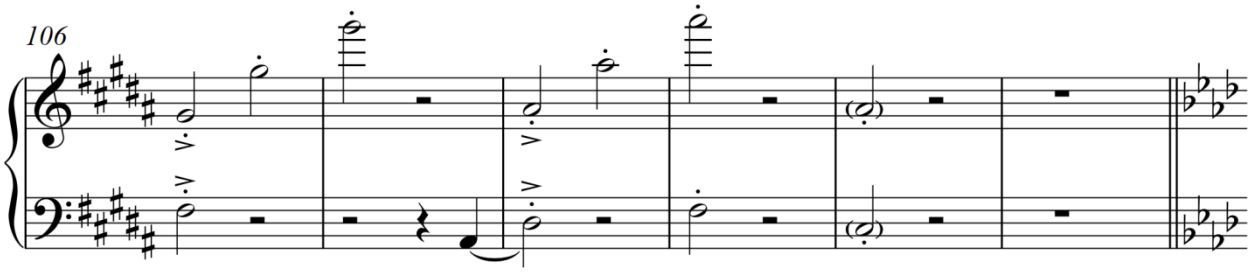
84

88

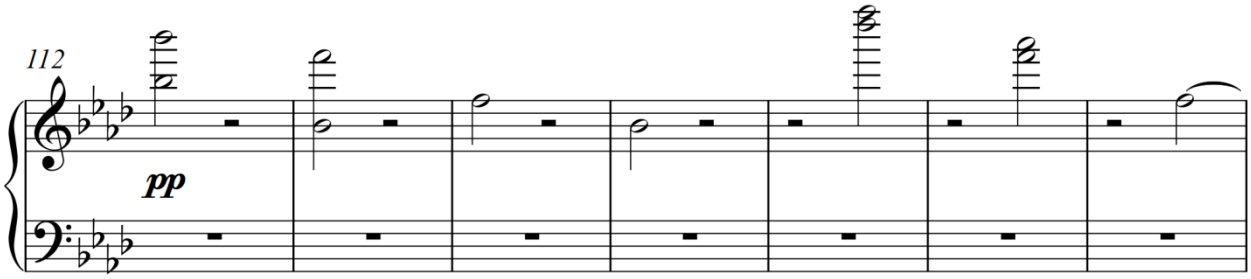
♩ = 120 seed [11, 3, 15, 2, 0, 26, 72, 13, 3, 410, 249, 1, 2, 2, 11, 6, 2, 3, 24, 198]

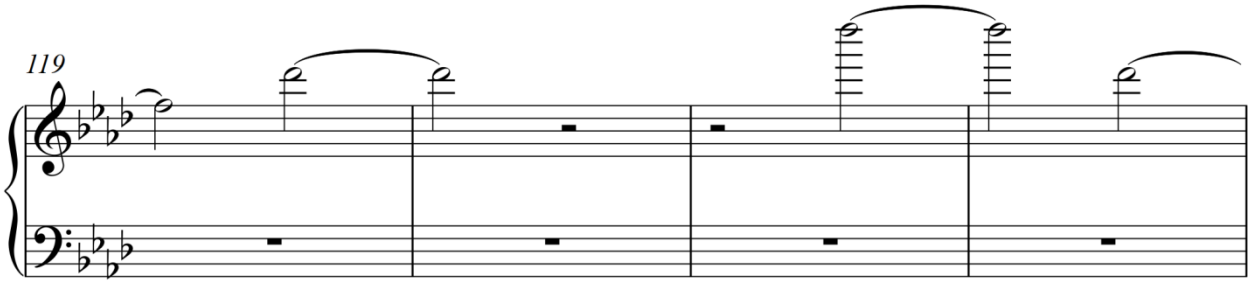
92

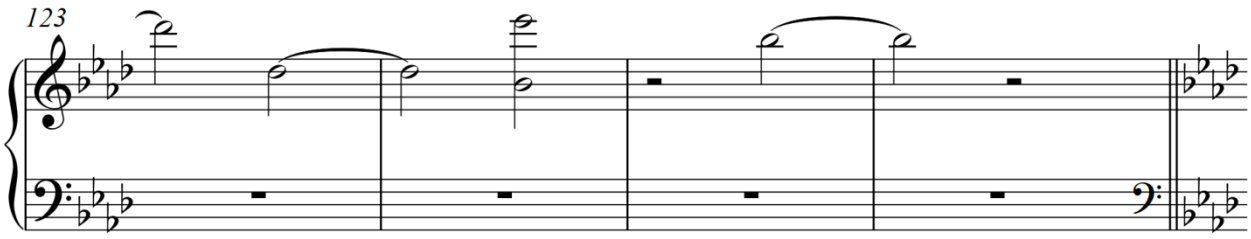
99



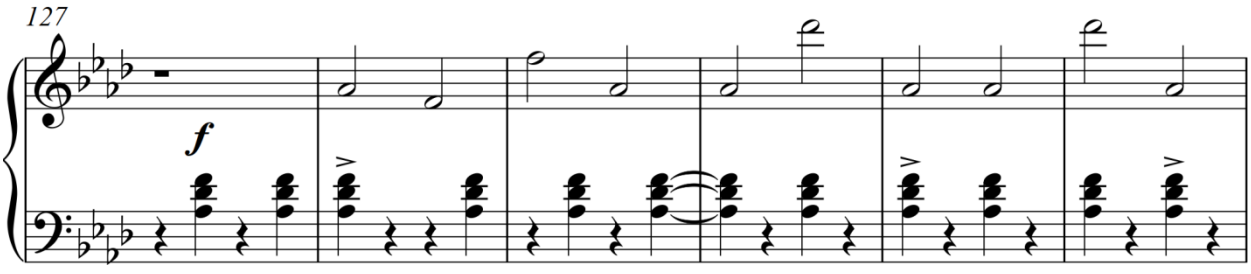
♩ = 100 seed [5, 5, 13, 6, 1, 11, 9, 13, 10, 555, 779, 5, 2, 4, 0, 781, 2, 2, 20, 520]

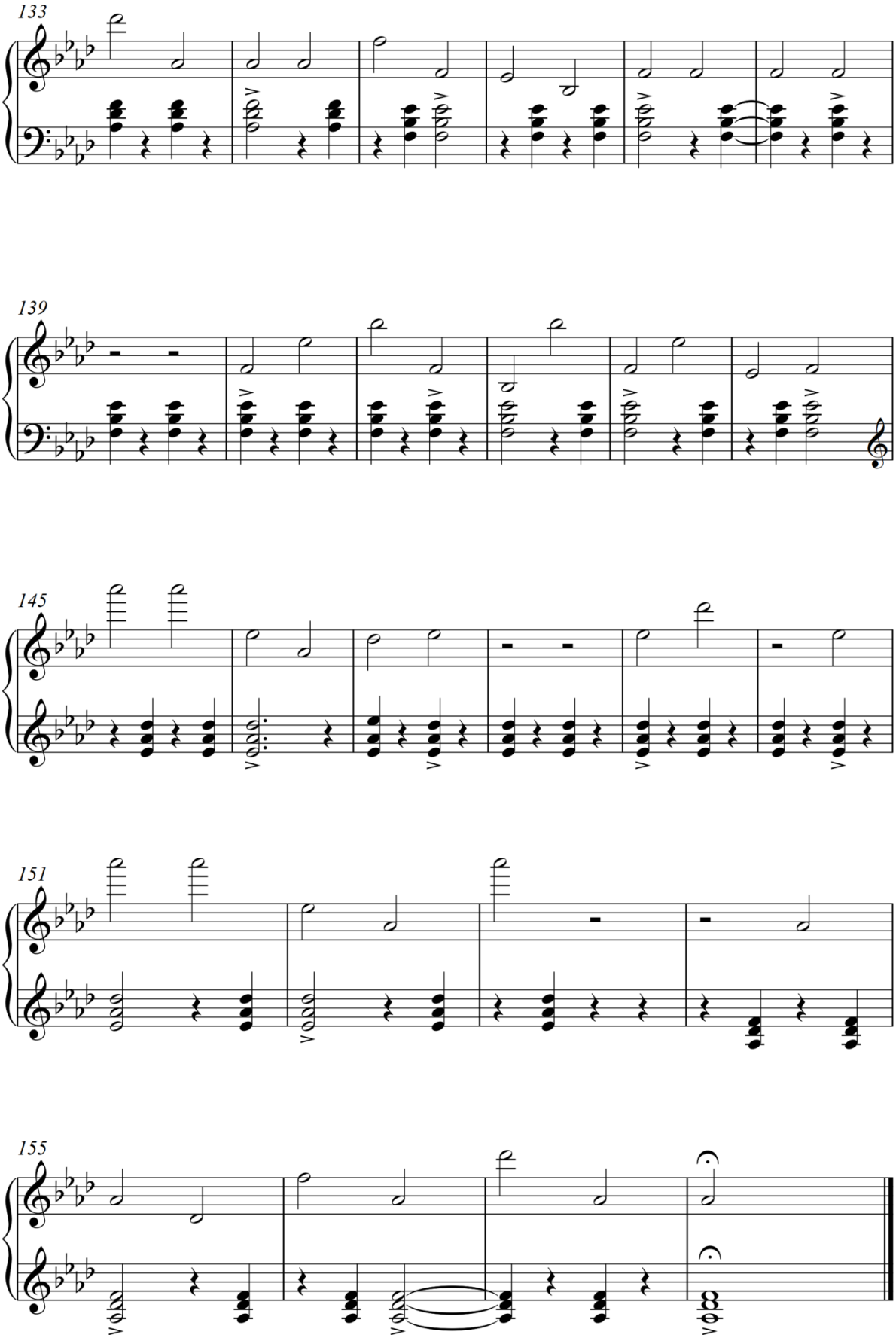






♩ = 200 seed [1, 0, 14, 0, 1, 16, 38, 9, 10, 95, 273, 4, 1, 2, 9, 899, 0, 1, 3, 224]




133
139
145
151
155

Appendix F- Ethics Certification

THE UNIVERSITY of ADELAIDE

RESEARCH BRANCH
OFFICE OF RESEARCH ETHICS, COMPLIANCE AND INTEGRITY

BEVERLEY DOBBS
EXECUTIVE OFFICER
LOW RISK HUMAN RESEARCH ETHICS REVIEW GROUP (FACULTY OF HUMANITIES AND SOCIAL SCIENCES AND FACULTY OF THE PROFESSIONS)
THE UNIVERSITY OF ADELAIDE
SA 5005
AUSTRALIA

TELEPHONE +61 8 8313 4725
FACSIMILE +61 8 8313 7325
email: beverley.dobbs@adelaide.edu.au

21 October 2013

Mr S Whittington
School: Elder Conservatorium of Music

Dear Mr Whittington

ETHICS APPROVAL No: **HP-2013-060**
PROJECT TITLE: **Using Biofeedback to Train Algorithmic Music Generators for Autonomous Use in Dynamic Media**

I write to advise that the Low Risk Human Research Ethics Review Group (Faculty of Humanities and Social Sciences and the Faculty of the Professions) has approved the above project. The ethics expiry date for this project is **31 October 2016**.

Ethics approval is granted for three years subject to satisfactory annual progress and completion reporting. The form titled *Project Status Report* is to be used when reporting annual progress and project completion and can be downloaded at http://www.adelaide.edu.au/ethics/human/guidelines/reporting. On expiry, ethics approval may be extended for a further period.

Participants in the study are to be given a copy of the Information Sheet and the signed Consent Form to retain. It is also a condition of approval that you **immediately report** anything which might warrant review of ethical approval including:

- serious or unexpected adverse effects on participants,
- previously unforseen events which might affect continued ethical acceptability of the project,
- proposed changes to the protocol; and
- the project is discontinued before the expected date of completion.

Please refer to the following ethics approval document for any additional conditions that may apply to this project.

Yours sincerely

ASSOCIATE PROFESSOR RACHEL A. ANKENY
Convenor
Low Risk Human Research Ethics Review Group (Faculty of Humanities and Social Sciences and Faculty of the Professions)

Bibliography

Affect. (n.d.). *Merriam-Webster Dictionary* (n.d.). Retrieved from http://www.merriam-webster.com/dictionary/affect

Affective. (n.d.). *Merriam-Webster Dictionary* (n.d.). Retrieved from http://www.merriam-webster.com/dictionary/affective

Alpern, A. (1995). Techniques for algorithmic composition of music. Amhusrt, Mass:*Hampshire College*.

Allison, B. Z., Dunne, S., Leeb, R., Millán, J. D. R., & Nijholt, A. (2012). *Towards Practical Brain-Computer Interfaces: Bridging the Gap from Research to Real-World Applications.* Heidelberg, Berlin:Springer

Amara, A., & Schmidhuber, J. (2012, November 28). How bio-inspired deep learning keeps winning competitions. *Kurzweil: Accelerating Intelligence*. Retrieved from http://www.kurzweilai.net/how-bio-inspired-deep-learning-keeps-winning-competitions

Ando, D. (2011.). Improving User-Interface of Interactive EC for Composition-Aid by means of Shopping Basket Procedure. In *Proceedings of the International Conference on New Interfaces for Musical Expression* 76–79.

Ando, D., Dahlstedt, P., Nordahl, M. G., & Iba, H. (2005). Computer Aided Composition for Contemporary Classical Music by means of Interactive GP. In *The Journal of the Society for Art and Science* 4(2)s. 77–87.

Andreassi, J. L. (2007). *Psychophysiology: Human Behavior And Physiological Response*. Abingdon:Routledge.

Angel, C. R. (2011). Creating Interactive Multimedia Works with Bio-data. In *Proceedings of the International Conference on New Interfaces for Musical Expression* 421–424. Retrieved from http://icemserv.folkwang-hochschule.de/~robles/papers_robles/Biointerfaces_CEIArtE_sm.pdf

Ariza, C. (2005). *An open design for computer-aided algorithmic music composition: athenaCL*. Boca Raton, Fla:Dissertation.com.

Ariza, C. (2005). Navigating the Landscape of Computer-Aided Algorithmic Composition Systems: A Definition, Seven Descriptors, and a Lexicon of Systems and Research. In *Proceedings of the International Computer Music Conference* 765–772.

Ariza, C. (2009). The interrogator as critic: The turing test and the evaluation of generative music systems. *Computer Music Journal*, *33*(2), 48–70.

Assayag, G., Rueda, C., Laurson, M., Agon, C., & Delerue, O. (1999). Computer-assisted composition at IRCAM: From PatchWork to OpenMusic. *Computer Music Journal*, *23*(3), 59–72.

Badcock, N. A., Mousikou, P., Mahajan, Y., de Lissa, P., Thie, J., & McArthur, G. (2013). Validation of the Emotiv EPOC EEG gaming system for measuring research quality auditory ERPs. *PeerJ*, *1*, e38.

Balzano, G. J. (1987). Review. *The American Journal of Psychology*, *100*(1), 135–139.

Behroozmand, R., Korzyukov, O., & Larson, C. R. (2012). ERP correlates of pitch error detection in complex Tone and Voice auditory feedback with missing fundamental. *Brain Research*, *1448*, 89–100.

Berlyne, D. E. (1971). *Aesthetics and psychology*. NY:Appleton Century Crofts.

Bernardi, L., Porta, C., & Sleight, P. (2006). Cardiovascular, cerebrovascular, and respiratory changes induced by different types of music in musicians and non-musicians: the importance of silence. *Heart*, *92*(4), 445–452.

Bettadapura, V. (2012). Face expression recognition and analysis: the state of the art. *arXiv Preprint arXiv:1203.6722*. Retrieved from http://arxiv.org/abs/1203.6722

Birnbaum, L., Boone, T., & Huschle, B. (2009). Cardiovascular Responses to Music Tempo during Steady-State Exercise. *Journal of Exercise Physiology Online*, *12*(1), 50-57.

bitrayne. (2013). Mind Your OSCs. Retrieved September 1, 2015, from http://sourceforge.net/projects/mindyouroscs/

Bongers, B. (2000). Physical interfaces in the electronic arts. Interaction theory and interfacing techniques for real-time performance *Trends in Gestural Control of Music*, 41–70.

Brown, S., & Volgsten, U. (2006). *Music And Manipulation: On the Social Uses And Social Control of Music*. NY:Berghahn Books.

Brunner, C., Andreoni, G., Bianchi, L., Blankertz, B., Breitwieser, C., Kanoh, S., … others. (2013). Bci software platforms. In *Towards Practical Brain-Computer Interfaces* 303–331. Heidelberg, Berlin:Springer.

Bruscia, K. E. (1991). *Case Studies in Music Theraphy*. Barcelona Publishers.

Buelow, G. J. (2001). *Affects, theory of the*. In Grove Music Online. Retrieved from http://www.oxfordmusiconline.com

Burden, F., & Winkler, D. (2008). Bayesian regularization of neural networks. *Methods in Molecular Biology (Clifton, N.J.)*, *458*, 25–44.

Burns, K. H. (1994). *The History and Development of Algorithms in Music Composition*, 1957-1993. Dissertation, Indiana:Ball State University.

Cage, J. (1961). *Silence*, Middletown, CT: Wesleyan University Press.

Chung, J., & Vercoe, S. (2006). The Affective Remixer: Personalized Music Arranging. Presented at *Human Factors in Computer Systems*. 393–398.

Churchland, P. S. (1994). *The computational brain*. Mass:MIT Press.

Ciresan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 3642–3649.

Cogan, R. (1984). *New Images of Musical Sound*. Mass:Harvard University Press.

Collins, N. (2008). Reinforcement learning for live musical agents. In *Proceedings of the International Computer Music Conference (ICMC), Belfast*. Retrieved from http://classes.berklee.edu/mbierylo/ICMC08/defevent/papers/cr1054.pdf

Collins, N. (2011). *The SuperCollider Book*. Mass:MIT Press.

Collura, T. F. (1993). History and Evolution of Electroencephalographic Instruments and Techniques. *Journal of Clinical Neurophysiology*, *10*, 476–504.

Cope, D. (1976). *New directions in music* (4th ed.). Iowa: W. C. Brown Co.

Cope, D. (1987). Experiments in Musical Intelligence. Presented at the *International Computer Music Conference, San Francisco*: Computer Music Association.

Cope, D. (1991). *Computers and Musical Style*. Oxford: Oxford University Press.

Cope, D. (2000). *The Algorithmic Composer*. Wisc: A-R Editions, Inc.

Cope, D. (2004). *Virtual Music: Computer Synthesis of Musical Style*. Mass:MIT Press.

Cope, D. (2005). *Computer models of musical creativity*. Mass:MIT Press.

Cope, D. (n.d.). Experiments in Musical Intelligence. Website. Retrieved August 26, 2014, from http://artsites.ucsc.edu/faculty/cope/experiments.htm

Coutinho, E., & Cangelosi, A. (2008). *Computational and psycho-physiological investigations of musical emotions*. Thesis. University of Plymouth. Retrieved from http://www.cse.salford.ac.uk /cogsys/EduardoCoutinho.pdf

Coutinho, E., & Cangelosi, A. (2011). Musical emotions: predicting second-by-second subjective feelings of emotion from low-level psychoacoustic features and physiological measurements. *Emotion (Washington, D.C.)*, *11*(4), 921–937.

Coyle, S., Mitchell, E., O'Connor, N., Ward, T., & Diamond, D. (2009). Wearable sensors and feedback system to improve breathing technique. In *UNCSR*. Dublin City University.

Dabhi, V. K., & Vij, S. K. (2011). Empirical modeling using symbolic regression via postfix Genetic Programming. In *2011 International Conference on Image Information Processing (ICIIP)* 1–6.

Dahlstedt, P., & McBurney, P. (2006). Musical agents: Toward Computer-Aided Music Composition Using Autonomous Software Agents. *Leonardo*, *39*(5), 469–470.

Davis, W. B., & Thaut, M. H. (1989). The Influence of Preferred Relaxing Music on Measures of State Anxiety, Relaxation, and Physiological Responses. *Journal of Music Therapy*, *26*(4), 168–187.

DeChaine, R. (2002). Affect and Embodied Understanding in Musical Experience. *Text and Performance Quaterly*, *22*(2), 79 – 98.

Deductive and Inductive Arguments. (n.d.). In *Internet Encyclopedia of Philosophy*. Retrieved from http://www.iep.utm.edu/ded-ind/

Definition of Affective. (n.d.). Retrieved March 3, 2015, from http://www.merriam-webster.com/dictionary/affective

Descartes, R., Meyer, M., & Timmermans, B. (1990). *Les Passions de l'âme*. Paris: Librairie générale française.

Descartes, R., Schuyl, F., La Forge, L. de, Clerselier, C., Hoym, K. H., & Descartes, R. (1677). *L'homme, et la formation du foetus*. Paris, T. Girard. Retrieved from http://archive.org/details/lhommeetlaformat00desc

Dolado, J., & Fernandez, L. (1998). Genetic Programming, Neural Networks and. Linear Regression in Software Project. Estimation. University of the Basque Country.

Dowling, W. J., & Harwood, D. L. (1981). *Music Cognition*. Mass:Academic Press.

Duvinage, M., Castermans, T., Petieau, M., Hoellinger, T., Cheron, G., & Dutoit, T. (2013). Performance of the Emotiv Epoc headset for P300-based applications. *Biomedical Engineering Online*, *12*(1), 56.

Edwards, M. (2011). Algorithmic composition: computational thinking in music. *Communications of the ACM*, *54*(7), 58.

Eigenfeldt, A., & Pasquier, P. (2012). Populations of populations: composing with multiple evolutionary algorithms. In *Evolutionary and Biologically Inspired Music, Sound, Art and Design* 72–83. Heidleburg, Berlin:Springer.

Ellis, D., & Brighouse, G. (1952). Effects of music on respiration-and heart-rate. *The American Journal of Psychology*, *65*(1), 39–47.

Ellis, R. J., Sollers III, J. J., Edelstein, E. A., & Thayer, J. F. (2008). Data transforms for spectral analyses of heart rate variability. *Biomedical Sciences Instrumentation*, *44*, 392–397.

Ellis, R. J., Sollers III, J. J., Havelka, B. M., & Thayer, J. F. (2009). The heart of the music: Musical tempo and cardiac response. In *Society for Psychophysiological Response*.

Fedorenko, E., McDermott, J. H., Norman-Haignere, S., & Kanwisher, N. (2012). Sensitivity to musical structure in the human brain. *Journal of Neurophysiology*, *108*(12), 3289–3300.

Gerhard, D. & D. H. Hepting. (2004) *Cross-Modal Parametric Composition*. In Proceedings of the International Computer Music Conference. San Francisco: International Computer Music association, 505-512.

Giacomini-Tebalducci-Malespini L. (1597). *Oratione e discorsi*. Fiorenza: Sermartelli

Giere, R. N. (1997). *Understanding Scientific Reasoning*. Canada:H B/Holt/Saunders.

Goldwater, B. (1972). Psychological Significance of Pupillary Movements. *Psychological Bulletin*, *77*(5), 340–355.

Gregg, M., & Seigworth, G. J. (2010). *The Affect Theory Reader*. NC: Duke University Press.

Guhn, M., Hamm, A., & Zentner, M. (2007). Physiological and musico-acoustic correlates of the chill response. *Music Perception*, *24*(5), 473–483.

Guzzetta, C. E. (1989). Effects of relaxation and music therapy on patients in a coronary care unit with presumptive acute myocardial infarction. *Heart & Lung: The Journal of Critical Care*, *18*(6), 609–616.

Hallam, S., Cross, I., Thaut, M., & Hodges, D. (2008). *Oxford Handbook of Music Psychology*. Oxford:Oxford University Press.

Hantz, E. C., Kreilick, K. G., Kananen, W., & Swartz, K. P. (1997). Neural Responses to Melodic and Harmonic Closure: An Event-Related-Potential Study. *Music Perception: An Interdisciplinary Journal*, *15*(1), 69–98.

Harrison, L., & Loui, P. (2014). Thrills, chills, frissons, and skin orgasms: toward an integrative model of transcendent psychophysiological experiences in music. *Frontiers in Psychology*, *5, 790*.

Herms, D. (2014). Logical Basis of Hypothesis testing in Scientific Research. Ohio: Ohio State University. Retrieved from http://www.dartmouth.edu/~bio125/logic.Giere.pdf

Hevner, K. (1937). The Affective Value of Pitch and Tempo in Music. *The American Journal of Psychology*, *49*(4), 621–630.

Hiller, L., & Isaacson, L. (1979). *Experimental music: composition with an electronic computer*. Westport, Conn:Greenwood Press.

Hoeberechts, M., Demopoulos, R. J., & Katchabaw, M. (2007). A flexible music composition engine. Ontario:University of Western Ontario

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.

Hsu, W., & Sosnick, M. (2009). Evaluating interactive music systems: An HCI approach. In *Proceedings of New Interfaces for Musical Expression (NIME)* 25–28. Retrieved from http://userwww.sfsu.edu/whsu/IMSHCI/NIME_Initial_Submission.pdf

Janata, P., & Petsche, H. (1993). Spectral Analysis of the EEG as a Tool for Evaluating Expectancy Violations of Musical Contexts. *Music Perception: An Interdisciplinary Journal*, *10*(3), 281–304.

Jasper, H. (1958). The ten twenty electrode system of the international federation. *Electroencephalography and Clinical Neurophysiology*, *10*, 371–375.

Jaws. (1975). [US]: MCA Records / Universal Music.

Johnson, C. G. (2012). Fitness in evolutionary art and music: what has been used and what could be used? In *Evolutionary and Biologically Inspired Music, Sound, Art and Design* 129–140. Heidleburg, Berlin:Springer.

Józsa, E. (2010). A potential application of pupillometry in web-usability research. *Periodica Polytechnica Social and Management Sciences*, *18*(2), 109.

Jiang, M., & Zhou, C. (2010). Automated composition system based on GA. In *2010 International Conference on Intelligent Systems and Knowledge Engineering (ISKE)* 380–383.

Juslin, P. N., & Sloboda, J. (2011). *Handbook of Music and Emotion: Theory, Research, Applications*. Oxford: Oxford University Press.

Juslin, P. N., & Västfjäll, D. (2008). Emotional responses to music: The need to consider underlying mechanisms. *Behavioral and Brain Sciences*, *31*(05). 559-621.

Juslin, P., & Sloboda, J. A. (Eds.). (2001). *Music and Emotion: Theory and Research* (1 edition). Oxford:Oxford University Press.

Kim, K.-J., & Cho, S.-B. (2006). A Comprehensive Overview of the Applications of Artificial Life. *Artif. Life*, *12*(1), 153–182.

Klügel, N., & Groh, G. (2013). Towards Mapping Timbre to Emotional Affect. In *New Interfaces for Musical Expression*. 23. 525-530

Knapp, R. B., Lusted, H. S., & Lloyd, A. M. (1993). Biosignal Processing and Biocontrollers. *Virtual Reality Systems*, *1*(1), 38–39.

Korhonen, M. D. (2004). *Modeling continuous emotional appraisals of music using system identification.* In University of Waterloo 89–109. University Press. Retrieved from https://uwspace.uwaterloo.ca/handle/10012/879

Koza, J. R. (1994). *Genetic programming II: automatic discovery of reusable programs*. Cambridge, Mass:MIT Press.

Kuchinke, L., Trapp, S., Jacobs, A. M., & Leder, H. (2009). Pupillary responses in art appreciation: Effects of aesthetic emotions. *Psychology of Aesthetics, Creativity, and the Arts*, *3*(3), 156–163.

Laeng, B., Sirois, S., & Gredeback, G. (2012). Pupillometry: A Window to the Preconscious? *Perspectives on Psychological Science*, *7*(1), 18–27.

Legitimising affective suite. (n.d.). online forum. Retrieved May 21, 2014, from http://emotiv.com/forum/forum10/topic316/

Lang, M. (2012). Investigating the Emotiv EPOC for cognitive control in limited training time. Honours Thesis. Uniersity of Canterbury Retrieved from https://corpus.canterbury.ac.nz /research/reports/HonsReps/2012/hons_1201.pdf

Lars-Olov Lundqvist, F. C. (2000). Facial electromyography, autonomic activity and emotional experience to happy and sad music. *International Journal of Psychology*, *35*, 225.

Laske, O. (1989). Composition theory: An enrichment of music theory. *Journal of New Music Research*, *18*(1-2), 45–59.

Levitin, D. J. (2007). *This Is Your Brain on Music: The Science of a Human Obsession* (1 Reprint edition). NY:Plume/Penguin.

London, J. (2007). Conceptualizing Music: Cognitive Structure, Theory, and Analysis. *Music Theory Spectrum*, *29*(1), 115–125.

Longo, D. L. (Ed.). (2012). *Harrison's principles of internal medicine* (18th ed). NY:McGraw-Hill.

Lundqvist, L.-O., Carlsson, F., Hilmersson, P., & Juslin, P. (2009). Emotional responses to music: Experience, Expression, and Physiology. *Psychology of Music*, *37*(1), 61–90.

Lykken, D. T., & Venables, P. H. (1971). Direct Measurement of Skin Conductance: A Proposal for Standardization. *Psychophysiology*, *8*(5), 656–672.

MacPherson, R. D., MacNeil, C., Marble, A. E., & Reeves, J. L. (1976). Integrated circuit measurement of skin conductance. *Behavior Research Methods & Instrumentation*, *8*(4), 361–364.

Mannes, E. (2011). *The Power of Music: Pioneering Discoveries in the New Science of Song*. NSW:Walker Books.

Manzolli, J., & Verschure, P. (2005). Roboser: A Real-World Composition System. *Computer Music Journal*, *29*(3), 55–74.

Mathworks. (2014). Choose a Multilayer Neural Network Training Function *Mathworks Website*. Retrieved from http://au.mathworks.com/help/nnet/ug/choose-a-multilayer-neural-network-training-function.html

Mattheson, J., & Harriss, E. C. (1981). *Der vollkommene Capellmeister: a revised translation with critical commentary*. UMI Research Press.

Maurer IV, J. (1999). The History of Algorithmic Composition. Website. Retrieved September 9, 2013, from https://ccrma.stanford.edu/~blackrse/algorithm.html

Mcartney, J. (2013). SuperCollider. website. Retrieved January 7, 2015, from http://supercollider.sourceforge.net/

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*(4), 115–133.

Metropolis (1927) DE: Universum Film. Dir. Lang, F.

Metropolis (2003) restored version. DE: Universum Film. Dir. Lang, F.

Meyer, D., Leisch, F., & Hornik, K. (2003). The support vector machine under test. *Neurocomputing*, *55*(1–2), 169–186.

Minjun Jiang, C. Z. (2010). Automated composition system based on GA. In *Institute of Electrical and Electronics Engineers Conference. 2010.* 380-383

Miranda, E. R. (2000). *Composing Music with Computers*. Burlignton:Focal Press.

Miranda, E. R. (2003). On the evolution of music in a society of self-taught digital creatures. *Digital Creativity*, *14*(1), 29–42.

Miranda, E. R. (2010). Organised Sound, Mental Imageries and the Future of Music Technology: a neuroscience outlook. *Organised Sound*, *15*(01), 13–25.

Miranda, E. R. & Brouse, A. (2005). Interfacing the Brain Directly with Musical Systems: On developing systems for making music with brain signals. *Leonardo*, *38*(4), 331–336.

Miranda, E. R., Sharman, K., Kilborn, K., & Duncan, A. (2003). On harnessing the electroencephalogram for the musical braincap. *Computer Music Journal*, *27*(2), 80–102.

Mudd, S., Conway, C. G., & Schindler, D. E. (1990). The eye as music critic: Pupil response and verbal preferences. *Studia Psychologica*, *32*(1-2), 23–30.

Murphy, J., & Gitman, Y. (2014). PulseSensor. Retrieved June 16, 2014, from http://pulsesensor.myshopify.com/products/pulse-sensor-amped

Näätänen, R., & Picton, T. (1987). The N1 Wave of the Human Electric and Magnetic Response to Sound: A Review and an Analysis of the Component Structure. *Psychophysiology*, *24*(4), 375–425.

Nishino, H. (2011). Cognitive Issues in Computer Music Programming. In *Proceedings of the International Conference on New Interfaces for Musical Expression* 499–502. Retrieved from http://www.nime2011.org/proceedings/papers/M21-Nishino.pdf

Oliwa, T., & Wagner, M. (2008). Composing Music with Neural Networks and Probabilistic Finite-State Machines. In M. Giacobini, A. Brabazon, S. Cagnoni, G. Di Caro, R. Drechsler, A. Ekárt, … S. Yang (Eds.), *Applications of Evolutionary Computing* (Vol. 4974, 503–508. Berlin Heidelberg:Springer.

OpenEEG. (n.d.). Retrieved December 16, 2014, from http://openeeg.sourceforge.net/doc/

oscmex. (2012). Retrieved January 14, 2015, from http://sourceforge.net/projects/oscmex/

Patil, K., Pressnitzer, D., Shamma, S., & Elhilali, M. (2012). Music in Our Ears: The Biological Bases of Musical Timbre Perception. *PLoS Comput Biol*, *8*(11), e1002759.

Petsche, H., Linder, K., Rappelsberger, P., & Gruber, G. (1988). The EEG: An Adequate Method to Concretize Brain Processes Elicited by Music. *Music Perception: An Interdisciplinary Journal*, *6*(2), 133–159.

Pinker, S. (1999). *How the Mind Works*. Penguin Books Ltd.

Pitman, D. (2012). *Developments and limitations in adapting electroencephalography for music.* Honours Thesis. University of Adelaide.

Polich, J. (2007). Updating P300: An integrative theory of P3a and P3b. *Clinical Neurophysiology*, *118*(10), 2128–2148.

Regalado, A. (2014, June 17). A Brain-Controlled Robotic Arm. Retrieved December 12, 2014, from http://www.technologyreview.com/featuredstory/528141/the-thought-experiment/

Reiser, J. (2012). Blind Hope: A Review of Gregg and Seigworth's The Affect Theory Reader. Website. Retrieved August 16, 2013, from http://www.electronicbookreview.com/thread/endconstruction/affective

Rickard, N. S. (2004). Intense emotional responses to music: a test of the physiological arousal hypothesis. *Psychology of Music*, *32*(4), 371–388.

Ries, H. A. (1969). GSR and breathing amplitude related to emotional reactions to music. *Psychonomic Science*, *14*(2), 62–62.

Roads, C. (1996). *The Computer Music Tutorial*. Mass:MIT Press.

Rosenboom, D. (1976). Brainwave Music. Website. Retrieved March 17, 2014, from http://davidrosenboom.com/gallery/brainwave-music

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Letters to Nature, 323(9), 533–536.

Schaefer, R. S., Desain, P., & Suppes, P. (2009). Structural decomposition of EEG signatures of melodic processing. *Biological Psychology*, *82*(3), 253–259.

Schlemmer, K. B., Kulke, F., Kuchinke, L., & Van Der Meer, E. (2005). Absolute pitch and pupillary response: effects of timbre and key color. *Psychophysiology*, *42*(4), 465–472.

Schneck, D. J., & Berger, D. (2005). *The Music Effect: Music Physiology and Clinical Applications*. UK:Jessica Kingsley Publishers.

Schön, D. A., (1983) *The Reflective Practitioner: How Professionals Think in Action. NY:*Basic Books.

Schubert, E. (1999). *Measurement and time series analysis of emotion in music*. Thesis (Ph. D.). University of New South Wales. Retrieved from http://unsworks.unsw.edu.au/fapi/datastream/unsworks:500/SOURCE01

Scruton, R. (2009). *Understanding Music: Philosophy and Interpretation* (1 edition). NY:Bloomsbury Academic.

Siwiak, D., Berger, J., & Yang, Y. (2009). Catch Your Breath-musical biofeedback for breathing regulation. In *Audio Engineering Society Convention 127*. Retrieved from http://www.aes.org/e-lib/browse.cfm?elib=15065

Sloboda, J. (1998). Does Music Mean Anything ? *Musicae Scientiae*, *2*(1), 19–31.

Soibelman, D. (1948). *Therapeutic and Industrial Uses of Music: A Review of the Literature*. UK:Cambridge University Press.

Stanley, G. B., Li, F. F., & Dan, Y. (1999). Reconstruction of natural scenes from ensemble responses in the lateral geniculate nucleus. *The Journal of Neuroscience*, *19*(18), 8036–8042.

Steinbeis, N., & Koelsch, S. (2008). Comparing the Processing of Music and Language Meaning Using EEG and fMRI Provides Evidence for Similar and Distinct Neural Representations. *PLoS ONE*, *3*(5), e2226.

Supper, M. (2001). A few remarks on algorithmic composition. *Computer Music Journal*, *25*(1), 48–53.

Tanaka, A. (2000). Musical performance practice on sensor-based instruments. *Trends in Gestural Control of Music*, *13*, 389–405.

Tan, B. H. (2012). *Using a Low-cost EEG Sensor to Detect Mental States*. Thesis. Pittsburg:Carnegie Mellon University. Retrieved from http://www.cs.cmu.edu/afs/cs/Web/People/listen2/pdfs /Lucas_Tan_CS_MS_thesis_2012.pdf

Task Force of the European Society of Cardiology the North American Society of Pacing Electrophysiology. (1996). Heart Rate Variability Standards of Measurement, Physiological Interpretation, and Clinical Use. *Circulation*, *93*(5), 1043–1065.

Thompson, M., & Biddle, I. D. (Eds.). (2013). Sound, music, affect: theorizing sonic experience. NY:Bloomsbury Academic.

Valdes, C. X., & Thurtle, P. (2005). Biofeedback and the arts: listening as experimental practice. Presented at the *REFRESH: First International Conference for Media Arts, Sciences and Technolgies*, Banff Center. Retrieved from http://pl02.donau-uni.ac.at/jspui/handle/10002/325

Vidal, J. J. (1973). Toward direct brain-computer communication. *Annual Review of Biophysics and Bioengineering*, *2*, 157–180.

Wallin, N. L. (1991). *Biomusicology: Neurophysiological, Neuropsychological, and Evolutionary Perspectives on the Origins and Purposes of Music*. NY:Pendragon Press.

Wallin, N. L., Merker, B., & Brown, S. (Eds.). (2000). *The origins of music*.Mass:MIT Press.

Werbos, P. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Thesis. Harvard University.

Wiggins, G. A. (2008). Computer Models of Musical Creativity: A Review of Computer Models of Musical Creativity by David Cope. *Literary and Linguistic Computing*, *23*(1), 109–116.

Wigram, T., Pedersen, I. N., & Bonde, L. O. (2002). *A Comprehensive Guide to Music Therapy: Theory, Clinical Practice, Research, and Training*. UK: Jessica Kingsley Publishers.

Williams, D., Kirke, A., Miranda, E. R., Roesch, E., Daly, I., & Nasuto, S. (2014). Investigating affect in algorithmic composition systems. *Psychology of Music*, Online, 1 –24. Retrieved from http://pom.sagepub.com/cgi/doi/10.1177/0305735614543282

Wilson, B., Buelow, G. J., & Hoyt, P. A. (n.d.). Rhetoric and music. *Grove Music Online*. Retrieved from http://www.oxfordmusiconline.com

Wu, H.-Y., Rubinstein, M., Shih, E., Guttag, J. V., Durand, F., & Freeman, W. T. (2012). Eulerian video magnification for revealing subtle changes in the world. *ACM Trans. Graph.*, *31*(4), 65.

Xenakis, I. (1968). *Atrées: for 11 instruments:[ST/10-3, 060962]*. Editions Salabert.

Zbikowski, L. M. (2005). *Conceptualizing Music: Cognitive Structure, Theory, And Analysis*. Oxford: Oxford University Press.

Zbikowski, L. M. (2008). Metaphor and music. In *The Cambridge handbook of metaphor and thought*. UK:Cambridge University Press. 502–524

Zdaniuk, G. J., Walters, D. K., Luck, R., & Chamra, L. M. (2011). A Comparison of Artificial Neural Networks and Symbolic-Regression-Based Correlations for Optimization of Helically Finned Tubes in Heat Exchangers. *Journal of Enhanced Heat Transfer*, *18*(2), 115–125.

Zimmerman, L. M., Pierson, M. A., & Marker, J. (1988). Effects of music on patient anxiety in coronary care units. *Heart & Lung: The Journal of Critical Care*, *17*(5), 560–566.