

A COGNITION-ANALOGOUS APPROACH TO
EARLY-STAGE CREATIVE IDEATION SUPPORT IN
MUSIC COMPOSITION SOFTWARE

A Thesis Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Jeffrey A. Smith

©Jeffrey A. Smith, February 2011. All rights reserved.

PERMISSION TO USE

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

ABSTRACT

Examination of the underlying principles of creativity reveal theoretical aspects that have not been well explored in creativity facilitation software. Most significantly of these, there has been little investigation into exploiting the distinctions between early- and late-stage creative processes and the attendant differences in cognitive processing active at those times, nor into employing the structural scaffolding embedded within creative works and the manner in which these can be extracted and harnessed to define levels of abstraction through which the material can be viewed and manipulated.

The Wheelsong project was conceived to exploit these principles, in the service of devising more creatively facilitative music composition tools, by focusing on these earlier, exploratory stages of the creative process, and by privileging structure over minutiae, in alignment with the mode of cognition (and corresponding user needs) that dominate the exploratory phase.

Explorations conducted with Wheelsong demonstrate that the platform embraces broad stylistic and cultural ranges of output. Experiments comparing the creative merits of early-stage, fragmentary outputs produced by Wheelsong against those produced by traditional representation schemes show a substantial improvement in both subjective quality and diversity indicators adhering to the structurally produced candidates, as measured by human judges.

ACKNOWLEDGEMENTS

This document would be empty if it were not for the support and encouragement provided by my supervisors, David Mould and Mark Daley; the ongoing oversight and feedback provided by my committee at the various stages of my work's development, Peter Stoicheff, Michael Horsch, Dean McNeill and Gord McCalla; and the financial support contributed by the University of Saskatchewan, the Department of Computer Science and by the National Sciences and Engineering Research Council of Canada (NSERC).

More importantly, my life would be empty if it were not for the love, patience and good humour shown to me every day by Fleur, Brinnameade, Merridew, Rigel and Tayna Smith. Life is a wild safari adventure into the unknown, and I can't imagine having a better group of people crammed into the jeep with me than you guys.

I'd like to dedicate this work to two of my most important teachers:

To Ethel Smith, for the lesson of wonder;

and to Sabina Meade, for the lesson of family.

CONTENTS

| | |
|--|-------------|
| Permission to Use | i |
| Abstract | ii |
| Acknowledgements | iii |
| Contents | v |
| List of Tables | viii |
| List of Figures | ix |
| 1 Introduction | 1 |
| 1.1 Terms of Reference | 1 |
| 1.2 Hypothesis | 2 |
| 1.3 Contribution Highlights | 4 |
| 1.4 Outline | 4 |
| 2 Background | 6 |
| 2.1 Creativity and Cognition | 6 |
| 2.1.1 Defining Creativity | 6 |
| 2.1.2 Creative Cognition | 9 |
| 2.1.3 Creative Contexts | 10 |
| 2.2 Creativity Facilitation | 11 |
| 2.2.1 General Frameworks and Principles | 11 |
| 2.2.2 Facilitative Software | 13 |
| 2.2.3 Sketching | 16 |
| 2.2.4 Conceptual Focus | 17 |
| 2.3 Music Composition Tools | 19 |
| 2.3.1 Notation Editors | 19 |
| 2.3.2 Template Tools | 20 |
| 2.3.3 Sequencers | 20 |
| 2.3.4 Signal Processors | 21 |
| 2.3.5 Automated Composers | 21 |
| 2.3.6 Exaptive Systems | 23 |
| 2.3.7 Music Programming Systems | 23 |
| 2.4 The Composing Process | 24 |
| 2.4.1 Professional Symphonic Composition | 24 |
| 2.4.2 Advice for Novices | 27 |
| 2.4.3 Non-western Composition | 29 |
| 2.4.4 Common Themes | 31 |
| 2.5 Synthesis | 32 |
| 2.5.1 Prime Ideator | 33 |
| 2.5.2 The P-Judgement Conundrum | 34 |
| 2.5.3 Bimodality Blindness | 35 |
| 2.6 Summary of Premises | 35 |
| 3 Program of Research | 39 |
| 3.1 Preliminary Research Hypothesis | 39 |
| 3.2 Terminology | 41 |

| | | |
|----------|---|------------|
| 3.2.1 | Creativity | 41 |
| 3.2.2 | Measurability | 42 |
| 3.2.3 | Creative Richness | 45 |
| 3.2.4 | Users and Musical Expertise | 45 |
| 3.2.5 | Musical Sketches | 46 |
| 3.2.6 | Software Composition Tools | 46 |
| 3.2.7 | Musical Rules | 48 |
| 3.2.8 | Musical Competence | 49 |
| 3.2.9 | Variety of Ideas | 49 |
| 3.3 | Scope | 50 |
| 3.4 | Formal Research Hypothesis | 50 |
| 3.5 | Experiment Plan | 51 |
| 3.5.1 | Implementation | 51 |
| 3.5.2 | Validation of Platform | 51 |
| 3.5.3 | Examination of Hypothesis | 52 |
| 3.6 | The Judgement Conundrum Revisited | 52 |
| 4 | Wheelsong | 54 |
| 4.1 | WSNG File Format | 56 |
| 4.1.1 | Unitlessness | 57 |
| 4.1.2 | Convenience | 59 |
| 4.2 | Transformation and Generation Filters | 61 |
| 4.3 | WSNT File Format | 64 |
| 4.4 | User Interfaces | 66 |
| 4.5 | Visualization and Audition Filters | 67 |
| 5 | Exploratory Compositions | 69 |
| 5.1 | The Explorations | 70 |
| 5.1.1 | Historical Western Music | 70 |
| 5.1.2 | Popular Western Music | 77 |
| 5.1.3 | Non-western Traditions | 80 |
| 5.1.4 | Tabula Rasa | 85 |
| 5.1.5 | Algorithmic Composition | 87 |
| 5.1.6 | Rejections | 89 |
| 5.2 | Analysis | 90 |
| 5.2.1 | Designed Affordances | 90 |
| 5.2.2 | Emergent Observations | 94 |
| 5.3 | Other Users | 101 |
| 5.3.1 | User A | 103 |
| 5.3.2 | User B | 103 |
| 5.3.3 | User C | 104 |
| 5.3.4 | User D | 104 |
| 5.4 | Conclusions | 105 |
| 6 | Richness Test | 106 |
| 6.1 | Introduction | 106 |
| 6.2 | Generation Phase | 107 |
| 6.2.1 | Experiment Design | 108 |
| 6.2.2 | Results | 114 |
| 6.3 | Variation Phase | 120 |
| 6.3.1 | Experiment Design | 120 |
| 6.3.2 | Results | 124 |
| 6.4 | Hybridization Phase | 126 |
| 6.4.1 | Experiment Design | 126 |
| 6.4.2 | Results | 128 |

| | | |
|----------|--|------------|
| 6.5 | External Validation | 129 |
| 6.6 | Conclusions and Implications | 132 |
| 7 | Conclusions | 135 |
| 7.1 | Contributions | 135 |
| 7.2 | Limitations | 137 |
| 7.3 | Future Directions | 138 |
| 7.3.1 | Music Composition | 138 |
| 7.4 | In Summary | 141 |
| | References | 148 |
| A | WSNG Operators | 149 |
| A.1 | Generative | 149 |
| A.1.1 | wsngscale | 149 |
| A.1.2 | wsngrandom | 149 |
| A.2 | Transformative | 150 |
| A.2.1 | wsngfromscale | 150 |
| A.2.2 | wsngrandomize | 151 |
| A.2.3 | wsnginterpolate | 151 |
| A.2.4 | wsngretrograde | 151 |
| A.2.5 | wsnginvert | 152 |
| A.2.6 | wsngsequence | 152 |
| A.2.7 | wsngloft | 152 |
| A.2.8 | wsngharmonize | 153 |
| A.2.9 | wsngbeatstuffer | 153 |
| A.2.10 | wsngstomp | 154 |
| A.2.11 | wsngmetronome | 154 |
| A.2.12 | wsngmetrify | 154 |
| A.3 | Evaluative | 155 |
| A.3.1 | wsnt2wsng | 155 |
| A.3.2 | wsngtomidi | 155 |
| A.3.3 | wsngtoly | 156 |
| A.3.4 | wsngpianoroll | 156 |
| A.3.5 | wsngvoiceflow | 156 |
| A.3.6 | wsnt2dot | 157 |
| B | Compositions | 158 |
| B.1 | Resolving Our Voices | 158 |
| B.2 | The Lament | 159 |
| B.3 | Telco Jackhammer | 161 |
| B.4 | Invention Sketch | 164 |
| B.5 | Tokyo Invention | 168 |
| B.6 | Bach Alley Shuffle | 172 |
| B.7 | Morning Traffic | 176 |
| B.8 | River of Light | 184 |
| B.9 | Clockhouse | 194 |
| B.10 | Incident on Pier 6 | 204 |
| B.11 | Ladrang Wilujeng | 214 |

LIST OF TABLES

| | | |
|------|---|-----|
| 4.1 | Table of default values for WSNG components | 59 |
| 5.1 | Cumulative operator occurrences | 91 |
| 5.2 | Effort measures for six sample compositions | 97 |
| 5.3 | Efficiency measures for six sample compositions | 97 |
| 5.4 | Age distribution of users | 101 |
| 6.1 | Effort metrics, e, for the generative fragment population | 109 |
| 6.2 | Inter-judge score correlations per candidate for generation phase | 115 |
| 6.3 | Distribution of <i>newness</i> score for generated fragments, by scheme | 116 |
| 6.4 | Distribution of <i>simplicity</i> score for generated fragments, by scheme | 117 |
| 6.5 | Distribution of <i>validity</i> score for generated fragments, by scheme | 118 |
| 6.6 | Distribution of <i>distinctness</i> score for generated fragments, by scheme | 119 |
| 6.7 | Distribution of <i>applicability</i> score for generated fragments, by scheme | 119 |
| 6.8 | Distribution of <i>validity</i> score for variations, by scheme | 125 |
| 6.9 | Distribution of <i>applicability</i> score for variations, by scheme | 125 |
| 6.10 | Distribution of <i>distinctness</i> score for variations, by scheme | 126 |
| 6.11 | Distribution of <i>validity</i> score for hybrids, by scheme | 128 |
| 6.12 | Distribution of <i>applicability</i> score for hybrids, by scheme | 129 |
| 6.13 | Distribution of <i>distinctness</i> score for hybrids, by scheme | 129 |
| 6.14 | Composition contest results | 130 |

LIST OF FIGURES

| | | |
|------|--|-----|
| 2.1 | Hyperscore user interface | 15 |
| 2.2 | Ingres' <i>Study for the Portrait of the Comtesse de Haussonville</i> [39] | 18 |
| 2.3 | Ending of Bartók's <i>Piano Sonata, Movement I</i> [107, p. 175] | 18 |
| 2.4 | A Schenkerian sketch of a Brahms score [11, p. 9] | 19 |
| 2.5 | Bartók's overall process [107, adapted from p. 29] | 24 |
| 2.6 | Bartók's primary creative workflow [107, adapted from p. 34] | 25 |
| 2.7 | Fragmentary memo [107, p. 20] | 26 |
| | | |
| 4.1 | Successive decompositions of melodic contours | 56 |
| 4.2 | First measure of Bach's <i>Invention №1 in C Major</i> | 58 |
| 4.3 | Bach's <i>Invention №1 in C Major</i> in compact WSNNG form | 60 |
| 4.4 | Repeated arpeggio | 61 |
| 4.5 | More simple arpeggio transforms | 62 |
| 4.6 | Repeated arpeggio | 62 |
| 4.7 | More complex arpeggio transforms | 63 |
| 4.8 | Generative filters | 64 |
| 4.9 | First measure of <i>Invention №1 in C Major</i> , expressed structurally | 65 |
| 4.10 | <i>Clockhouse</i> encoding in the Vim editor | 67 |
| 4.11 | Visualizing the first bar of Bach's <i>Invention №1 in C Major</i> | 68 |
| | | |
| 5.1 | Naive visual patterns in Bach's <i>Invention №1 in C Major</i> | 71 |
| 5.2 | WSNT encoding fragments from the <i>Invention Sketch</i> | 72 |
| 5.3 | Coding changes to produce the <i>Tokyo Invention</i> | 74 |
| 5.4 | Coding changes to produce the <i>Bach Alley Shuffle</i> | 75 |
| 5.5 | Coding changes to produce <i>Melancholy</i> | 76 |
| 5.6 | Coding changes to produce <i>Danger Baby</i> | 77 |
| 5.7 | Simple example of <code>wsngharmonize</code> | 78 |
| 5.8 | Perlin's Blues | 79 |
| 5.9 | Important arguments for <code>wsngrandom</code> | 80 |
| 5.10 | Voice flow diagram for <i>Ladrang Wilujeng</i> fragment | 81 |
| 5.11 | Original and derived encodings for GenPan1 | 82 |
| 5.12 | Voice flow diagram for <i>Clockhouse</i> fragment | 83 |
| 5.13 | Partial voice-flow diagram for <i>River of Light</i> | 84 |
| 5.14 | Partial voice-flow diagram for <i>Incident on Pier 6</i> | 84 |
| 5.15 | Voice partitioning for the final bars of <i>O Canada</i> | 86 |
| 5.16 | Reverse-Schenkerian composition of <i>The Lament</i> | 87 |
| 5.17 | Genetically spawned <i>Arhythmia</i> | 88 |
| 5.18 | Voice flow for <i>Your Sister's Depression</i> | 89 |
| 5.19 | Rudimentary editor interfaces | 102 |
| 5.20 | Rapid-fire arpeggios | 103 |
| 5.21 | Unexpected result | 103 |
| | | |
| 6.1 | Pseudo-code of atomic generation algorithm | 111 |
| 6.2 | Pseudo-code of structural germination algorithm | 112 |
| 6.3 | Evaluation form for generation phase candidates | 114 |
| 6.4 | Pseudo-code of variation algorithm | 122 |
| 6.5 | Evaluation form for variation candidates | 124 |
| 6.6 | String crossover | 127 |
| 6.7 | Tree crossover | 127 |
| 6.8 | Evaluation form for hybridization candidates | 128 |

| | | |
|------|--|-----|
| 6.9 | Component scores for <i>The Lament</i> and <i>Clockhouse</i> | 131 |
| 6.10 | Recapping the high-score production levels | 132 |

CHAPTER 1

INTRODUCTION

Computers are now being used in almost every form of artistic expression, from visual arts to music, from dance to architecture, in literature, and even in newer forms of media that have only become possible since the advent of computerization and networking. But despite that proliferation of engagement, many researchers agree (e.g. [28, 92, 103]) that much remains to be done in our quest to facilitate the full breadth of the creative process in software.

In an article on creativity tools, Ben Shneiderman stated that, “The grand challenge for creativity support tool designers is to enable more people to be more creative more of the time.” [103, p. 1]. Within the context of the article, the statement was not merely a definition of the goals of facilitated creativity software, but further, that in his view, we have not yet achieved all that can be achieved. More specifically, Shneiderman’s declaration carries several implications: that people who are not yet creative can become so; that people who already are creative can become moreso; that the frequency with which either group produces output judged to be creative can be increased; and that these aims can all be achieved to some degree through better software design. In this document, I will explore one approach to meeting this challenge, within the specific domain of music composition.

1.1 Terms of Reference

In order to undertake such an investigation, I must first be careful to define several key terms.

Types of Creativity As I will discuss in Chapter 2, creativity can be defined in many different ways, but recent cognition theory points to two relevant types of creativity, characterized by the standards to which a candidate idea is held. In the case of psychological creativity (p-creativity), to be judged creative, a work must pass muster within the mind of its creator, judged by his system of values and against his understanding of historical precedents. This is the mode of creativity that most directly informs an individual artist’s sense of his own creative abilities, and is the most directly related to his individual process of artistic investigation.

Historical creativity (h-creativity), on the other hand, pertains to a cultural judgement in which a specific group evaluates the merits of an idea, within the context of their own system of values

and their collective historical legacy. In a sense, h-creativity is simply p-creativity, expanded to the collective mode, and judged by gate-keepers.

Neither mode, however, can be taken as authoritative. Just as different people can have conflicting views on the creative merit of any particular work, cultures and societies can also differ, nor are these distinctions limited to geographical or language divisions. Time can also play a crucial role in the shifting evaluations of creative merit. The most extreme example I have encountered is that of J. S. Bach, who in his own time was regarded primarily as an accomplished organist — his reputation as a composer having fallen into decline even before his death. It was not until more than a century later, when Mendelssohn and Schumann revived interest in his work, that Bach’s reputation as a composer was restored [77, 93].

Phases of Creativity In addition to there being different contexts in which creativity can be judged, there are also different processes by which creative ideas are conceived and developed, and different stages to those processes. In general, early stages are characterized by unstructured exploration and association-making, while later stages most typically consist of refinement and polishing activities. My work is specifically aimed at facilitating the less thoroughly explored dimension of early-stage creative ideation.

Intended Users As implied in Shneiderman’s challenge, it would be advantageous if my approach were able to facilitate creativity for both beginner and expert users alike, but what do I mean by “beginner” and “expert?” There are in fact, two relevant dimensions of expertise involved. A user can have arbitrary experience in the realm of music composition, and that degree of experience is independent of his experience with any particular tool.

In order for my solution to be most widely applicable, it should be efficacious regardless of the user’s experience in either dimension. By this I do not mean that it must offer benefit to all categories of users in the same way, but only that for each such category, the measured creative effectiveness is increased.

1.2 Hypothesis

With these terms now clarified, I can present my research hypothesis, which is founded upon several related observations that I have extracted from the literature on creativity theory, media tools, creative people and software facilitation, which I will summarize here and then expand upon more fully in Chapters 2 and 3.

Observation 1 Gabora tells us [36] that there are two different modes of cognition involved at opposite ends of the creative process. The early, exploratory stage, she tells us, is dominated by

associative-style thinking, in which wide leaps of analogy are made, attempting to link disparate concepts. Conversely, the later stages of the process, in which ideas are refined and polished, is increasingly dominated by the analytical mode of thought.

Given that the manner in which users are thinking differs between those two opposed ends of the creative process, it stands to reason that their needs within those two different contexts might differ as well, and that, by extension, the features and affordances provided by their tools might likewise need to differ, in order to best accomodate the relevant thinking style.

Observation 2 From such diverse fields as cognitive psychology [52], musicology [66] and evolutionary psychology [49], we have an increasing body of theory suggesting that music is a fundamental construct within the configuration of human cognition, and that, like other forms of human thought, music is inherently structural in its cognitive organization. From that, I am led to wonder whether it might be advantageous to represent musical ideas in a similarly structural manner, to better orient the content with the composer’s cognitive processes.

Observation 3 There are many methodological approaches to music composition. For example, one can begin by choosing a style and genre, borrowing from standard musical forms, and can work down from there by introducing successively finer-grained, novel variations on the standard key signatures, rhythms and so on. Alternately, one could begin with an invented melodic fragment or chord progression, and work upward, aggregating elements into successively larger structures, themes, verses, movements and the like. Most commonly, these two approaches are applied in combination, as the composer alternates his attention between the top-down, and bottom-up levels of abstraction.

Observation 4 Meichenbaum tells us [75] that ideative creativity can be enhanced by shifting the user’s inner mental dialogue away from fine-grained detail, and toward more abstract levels of construal of the subject material. This suggests that we can similarly enhance musical creative ideation by finding ways to present the content to users in a form that emphasizes structure over minutiae.

Observation 5 Eaglestone et al. [28, 86, 26] criticize existing composition tools for their relative inability to present users with multiple ideas and threads of development simultaneously. This aligns with Csikszentmihalyi’s observation [21] that creativity flourishes where multiple cultures and ideas are brought into juxtaposition.

Synthesizing these observations, I produced my research hypothesis, which in thumbnail form, states that:

Hypothesis 1 (Thumbnail) *Music composition tools can be devised that produce measurably more creative ideas, more frequently, at the early stages of creative exploration, by organizing the content structurally, in terms of the musical transformations used to aggregate the elements, as compared to traditional tools that place emphasis on the note-event details.*

This hypothesis was examined by creating a testbed implementation in software, called Wheel-song, that was subjected to a series of experiments to demonstrate its general expressive reach, and to assure its suitability as a platform for further analysis of the hypothesis.

The claims of the hypothesis were then tested in a series of experiments using sample musical fragments created within the testbed system.

1.3 Contribution Highlights

This thesis reports a number of contributions, the highlights of which are as follows.

An implementation of my experimental composition platform, Wheelsong, is provided and experiments are described in which example outputs were produced to demonstrate the expressive power and variety of compositions achievable with the system.

Wheelsong is shown to be capable of representing and manipulating musical compositions in a “sketch” form that is suitable to rapid generation and evaluation of candidate musical fragments covering a broad expanse of music space.

Example composition sketches are offered to demonstrate Wheelsong’s stylistic and cultural diversity of output and its inherent ability to harness what might normally be called “user errors,” by transforming them into potentially inspiring variations, relevant to the subject matter at hand.

A compact collection of musical transformation operations are described that permit the creation of compositions conforming to a wide range of musical styles, cultures and genres with relative efficiency.

A set of metrics is proposed for measuring early-stage, p-creative richness in arbitrary artistic media.

Proceeding from the specific experiences with Wheelsong and music composition, a more general framework for early-stage creativity facilitation is provided, applicable to arbitrary media, along with discussion of how that framework might be applied to other types of artistic media.

1.4 Outline

In Chapter 2, I will survey the background literature from several related fields, and in so doing, will extract a number of theory-based premises to serve as axioms from which my specific facilitation hypothesis was constructed. The development of this hypothesis will be related in Chapter 3.

Following from this hypothesis, Chapter 4 will then give a detailed account of the experimental suite of inter-dependent software tools, called *Wheelsong*, that I developed to test my hypothesis.

In Chapter 5, I will describe a series of explorations that were conducted to demonstrate *Wheelsong*'s breadth and utility as an early-stage composition tool. This will then be followed by Chapter 6's description of the three staged experiments that were conducted to assess the degree to which *Wheelsong* addresses the original goal of facilitating early-stage creative musical ideation.

Finally, in Chapter 7, I will summarize the findings of this work, the contributions it makes, its limitations, and the various directions of continued research that it suggests, both in music and by extrapolation of the underlying theoretical model, into other creative media as well.

CHAPTER 2

BACKGROUND

There are several different disciplines that touch upon aspects of creativity that are germane to my work. In this chapter I will provide a brief overview and analysis of the most salient contributions within each of them. The contributing fields are organized from abstract theory to concrete human practice. I will begin in Section 2.1 with discussions of creativity theory arising from cognitive psychology, followed in Section 2.2 by the work on general creativity facilitation being explored in computer science. Section 2.3 will explore the variety of software tools built specifically for music composition, and then, finally, Section 2.4 will survey the variety of ways in which humans have traditionally composed their music, without computer assistance. These surveys will then be synthesized in Section 2.5.

My ultimate goal is to extract practical theoretical insights from these various fields and from them, to devise a theory-inspired approach to facilitating creativity in software. Given the breadth of the fields we will be covering, however, it would be easy to lose sight of the important points discussed along the way. For that reason, I will extract the most relevant of them and highlight them as *foundational premises*, expressed in terms appropriate to my intent of inspiring facilitative practices.

In Chapter 3, I will then resynthesize those premises to suggest a specific architectural approach to facilitating creativity that will then be developed and tested throughout the remainder of this work. To assist the reader in following these subsequent discussions, the last section (2.6) of this current chapter then will be a simple restatement of the entire set of premises, collected together for easy reference.

2.1 Creativity and Cognition

2.1.1 Defining Creativity

The literature is replete with accounts of many different creative processes employed in many different fields (e.g. [74, 112, 107, 37]), but while these descriptions illuminate the area I intend to study, the majority of them do so in a subjective, non-rigorous manner, couched in terms that are relatively imprecise and poorly quantified. There is not even any consensus over what class of

entity is being defined; some definitions focus on the ideating person, some are based in the process used, and others describe qualities of the artifacts produced. For example, consider the following definitions.

A product or response will be judged as creative to the extent that (a) it is both a novel and appropriate useful, correct or valuable response to the task at hand, and (b) the task is heuristic rather than algorithmic. [4]

Creative activity seems simply to be a special class of problem solving activity characterized by novelty, unconventionality, persistence and difficulty in problem formulation. [83]

Creative products elicit a distinct set of aesthetic responses from observers: surprise, satisfaction, stimulation, and savoring. [50]

Creativity results in the production of some novel result that is useful, tenable, or satisfying, and represents a real “leap” away from what has previously existed. [109]

Imprecise definitions like these, while descriptive, provide a poor basis for building software models. More recently, however, cognitive psychology has at last begun providing a greater degree of precision, giving us more concrete theories and definitions that can be transformed into software requirements. I now draw attention to two of these more recent definitions because they each emphasize different pieces of the puzzle: what creativity is, and how it is judged.

Boden defines creativity by its chief attributes as “the ability to come up with ideas or artefacts that are new, surprising and valuable.[9]” Csikszentmihalyi, on the other hand, describes it more in terms of its effect, saying that creativity is “when a person, using the symbols of a given domain¹, such as music, engineering, business or mathematics, has a new idea or sees a new pattern, and when this novelty is selected by the appropriate field for inclusion into the relevant domain.” [21]

The apparent conceptual consistency between the various definitions of creativity, however, does not carry over to the proposed models of how it behaves in practice. Some authors characterize the creative process as being comprised of two distinct phases [23, 22], others employ four phases [117, 36, 102], and others posit as many as five [19, 4]. One class of theorists, referred to by Shneiderman as the *structuralists*, characterize the creative process as the meticulous, methodical construction of analytical solutions, while another group — the *inspirationalists* — emphasize the role played by sudden flashes of insight.

Despite these differences, most commentators acknowledge Wallas’s model [117] as the first workable description of the creative process, and many of the most often cited models in use today owe an obvious debt to that origin. Wallas’s model construes the creative process as a journey taken by an individual thinker, passing through four stages. The Preparation Phase is one in which

¹Terms identified with a † are included in the glossary.

the individual must first be immersed in some knowledge domain relevant to the problem at hand, achieving some degree of expertise with the material. The Incubation Phase consists mostly of doing other things, while allowing some degree of subconscious or background rumination to take place. Illumination is the stage at which a hypothetical solution emerges, which is then consciously developed. This phase is often triggered by a sudden flash of insight, which is often referred to as the “a-ha” or “eureka” moment, in deference to the story of Archimedes and his sudden insight into the principle of hydrostatic buoyancy. Wallas’s final stage is the Verification Phase, in which the developed hypothesis is put to the test and either rejected or verified as a legitimate solution to the subject problem.

Most definitions follow Wallas’s lead in another important aspect as well, defining creativity in terms of solving a problem, although in the case of artistic creativity, it is often difficult to quantify what problem is actually being solved unless we appeal to the rather uninformative problem of “creating something beautiful.”

The Wallas model is still used today, in one form or another, especially with respect to what Boden refers to as the p-creative mode. In assessing p-creativity, or *psychological* creativity, all that matters are the personal, subjective impressions of the artist. An idea that is new, unexpected, simple, etc. *in the mind of the artist* qualifies as being p-creative.

Beyond p-creativity, however, cognitive psychologists have lately moved to a broader, more societally informed definition, marginalizing the personal judgement aspects of p-creativity in favour of historical, cultural judgements. This broader, h-creative interpretation posits that until ideas have been evaluated by gatekeepers in the community, they cannot be considered truly creative. Tying the two forms together, Boden acknowledges that, in practice, before an idea can be offered up for h-creative judgement, it must first pass some preliminary p-creative value judgement in the mind of the originator; otherwise, it would never have been presented to the broader community.

The P-H Sequencing Premise: H-creative ideas must generally be judged as p-creative by their originator before becoming available to the field for h-creative scrutiny.

A consequence of this new, historical/cultural perspective on creativity is an acknowledgement of the immensity of influence played by the surrounding cultural context on any new creative discovery. No idea is wholly new, in and of itself. Everything is informed by what has come before and by the works of contemporaries. Even if the new work stands as a complete rejection of the status quo, that in itself is a form of constructive influence.

The Contextual Dependence Premise: No matter how original, no idea is created in a vacuum of influence. New ideas can only be conceived and understood within the context of what has gone before.

Boden goes on further to acknowledge that it is entirely logical to seek to facilitate h-creative production by focusing attention on facilitating p-creative work [9, p. 47].

The P-Leveraging Premise: H-creativity might be enhanced by increasing p-creative output.

That of course raises a question that I should deal with immediately. Can creativity actually be enhanced? Fortunately, the answer appears to be, “Yes.” One especially intriguing mechanism, reported by Meichenbaum [75], is to shift the focus of the subject’s inner dialogue away from minutiae, toward more abstract issues, which induces a cognition shift toward associative thinking and higher levels of construal, thereby triggering more abstract, exploratory and creative behaviour.

The Feasibility Premise: It is feasible to intervene in the creative process in a manner that enhances the measured creativity of the result.

2.1.2 Creative Cognition

In addition to there being two tiers of creative judgement, Gabora proposes [36] that there are also two fundamental modes of thought involved in the creative process: the associative and analytic modes. In the associative mode, ideas are juxtaposed as the brain compares and contrasts possible memory patterns, looking for what might be termed “inspiration.” Analytical mode, on the other hand, involves the more constructive development of patterns and ideas already in place. Gabora concludes that these two modes equate naturally to the two ends of the creative process, with associative thinking dominating the early stages of creation, and analytical thinking predominant during the later stages. This distinction is echoed repeatedly in subjective descriptions of the creative process, perhaps most clearly in the description given by Wyndham, quoted in Section 2.4.

If there are indeed different modes of cognition involved at different stages of the creative process, then it seems reasonable to conclude that software designs should consider the relevant modes of cognition that will be active when the tool is employed, and that doing so may improve the quality of interaction experienced by the user, thereby improving the tool’s creative effectiveness.

The Cognitive Context Premise: Creativity may be enhanced by designing tools with an awareness of the cognitive modes that will be employed by the user at the time of use.

Following from these observations, Trope and Liberman note more explicitly [114] that a thinker’s perceived distance from the material at hand strongly influences the level of abstraction at which he construes it, and Jia et al. have demonstrated [53] that creative thinking is usually associated with the more abstract, or higher, construal levels. Furthermore, Jia et al. go on to suggest that the association is bi-directional: creativity induces abstract construals, and conversely, abstract construals induce creative thinking. So therefore, by leading a subject to consider a prob-

lem at a higher level of abstraction, it may be possible to direct their thinking toward more creative associations and outputs.

The Cognition Induction Premise: Early-stage exploratory thinking may be inducible by directing a thinker’s attention toward high-level, abstract representations, suggesting a greater psychological distance from the material.

2.1.3 Creative Contexts

Csikszentmihalyi adds two important, additional observations. First, he refers repeatedly [21] to the notion that when cultures collide (in whatever form that may take) artists immersed in the collision are more readily able to view and contrast constructs underlying the source cultures. This crucible of diverse comparison, he says, frequently results in an explosion of creative diversity.

The Richness Premise: Creativity may be enhanced when large numbers of diverse ideas from diverse contexts can be placed in close mutual proximity.

He also gives us the notion of flow [20], a state of creative engagement so thorough that it precludes awareness of any external concerns from the mind of the thinker, including awareness of self and time. Flow, he goes on to say [21], is a crucial component of the creative process, because it enables fluidity of thought in which more of the problem domain can be held in the mind at once, and manipulated effectively.

The Smoothness Premise: Creativity may be enhanced when manipulations of the subject material are quick and easy to achieve, minimizing cognitive disruptions to the process, and facilitating flow.

In a related finding from the context of management studies, Shalley et al. note a strong correlation between levels of creative satisfaction among workers and their willingness to stay engaged in a job or task [100], although they do not claim causality.

The Satisfaction Premise: Creative productivity might be increased by processes that improve the level and/or rate of creative satisfaction achieved.

In addition to the work being done on creativity *per se*, there are other tangential contributions that I feel are important as inspiration for my approach. David Huron tells us about the role of anticipation and its function in the cognitive process of understanding music [49]. He describes the process of listening to music as one of building mental models of the underlying principles of the composition. These models are used to predict the future course of the music, by extrapolation. Anticipation then builds as the mind awaits confirmation or refutation of that prediction, which is then fed back into the process to refine the model and another cycle begins. These structures that

are built are not individual notes, but instead take the form of more abstract or even algorithmic relationships between them. Speaking about similar mental structures, Levitin suggests [66] that they are every bit as complex as the structures that underpin language, and in fact, are processed by the same areas of the brain, using the same cognitive mechanisms.

This notion of structure-building also resonates very deeply with an idea popularized by Jaynes [52], which holds that all conscious thought can be characterized as the construction of hierarchical systems of metaphor, relating lower level ideas to successively higher level abstractions, through transformative relations, resulting in a sort of constructed idea scaffold or framework.

Results like these suggest to me that not only might humans be hard-wired for music, but that music, as well as other forms of idea expression, are inherently structural in their cognitive organization. And from that, I am led to wonder whether it might be advantageous to represent musical ideas in a similarly structural manner.

2.2 Creativity Facilitation

2.2.1 General Frameworks and Principles

There are a number of lines of research that fall under the general banner of creativity facilitation, such as the creative design of programming tools [40], enhancing creativity in corporate decision-making [72], or even the creative analysis of accounting data [54]. But each of these efforts tends to focus on domain-specific solutions, and little of it seems directly relevant to the specific problem of facilitating creativity of software users.

Much of what has been written on methods for harnessing human creative powers in software has been written from a business perspective. Cougar, for example, offers a five-stage model of the process [19] consisting of problem delineation, information collection, ideation, evaluation, and implementation planning. Superficially, this appears similar to other models, such as those of Wallas or Shneiderman (see below), but in the details, Cougar’s model emphasizes what Shneiderman calls the “structuralist” perspective, which characterizes creativity in its most analytical form, espousing the methodical and exhaustive examination of prior art and potential candidate solutions. This structuralist approach is common among business creativity models ([55, 56, 72]) and seems well suited to a corporate style of research, where schedule predictability and evidence of thoroughness are perhaps of greater import than is generally the case for artistic explorations. But while there are undoubtedly many who employ similarly methodical techniques, Shneiderman acknowledges that this is the least common approach taken by artists.

In searching for truly domain-agnostic frameworks that are able to support artistic modes of creativity as well as those of engineering and business, there is not much to be had. There are some

models that focus on very specific issues in creativity, such as collaboration support [14], or idea dissemination [61], but very few that try to embrace the broader challenge of creativity as a whole.

One truly general approach is Shneiderman’s Genex model [101], in which he posits four sequential phases to the creative process, known as Collect, Relate, Create and Donate, which owe their origins to Wallas’s model. The most significant departure Shneiderman makes from Wallas is with his Donate phase, in which he privileges the h-creative mode by requiring that the creative output be shared with others. While the Genex framework never becomes prescriptive, Shneiderman does make some other useful observations. In particular, he emphasizes the value of visualization in the creative process — recommending that an ability to see the subject material in different ways and from different perspectives be provided.

The Visualization Premise: Creativity can be enhanced by providing multiple views of the content, emphasizing different structures or interpretations.

In their COSTART project, which assessed visual artists working with computers, Candy and Edmonds report [13] a commonly perceived requirement for flexibility (in every dimension of their tools: software, hardware, interfaces, etc.), a need for collaborative engagement with others, and support for a structural engagement with their subject material.

At a National Academy of Science workshop on creativity support tools held in 2005, a number of leading researchers in the field drafted a joint statement [92] outlining the design principles for creative facilitation. They cite such things as a need for low-cost experimentation (“costs” meaning impediments to user workflow), a low threshold, a high ceiling, wide walls, multiple modes of interaction, multiple styles of working, collaboration support, open file formats, open architectures, extreme simplicity, careful choice of abstractions, and above all else, they say, software should be designed by people who actually intend to use the programs themselves.

Particular from among that list, the ceiling, threshold and walls requirements, which I refer to as the “nice house” guidelines, have achieved a degree of support in the field, and are quoted repeatedly in the literature [58, 79, 59, 43, 91, 92, 116], despite the fact that I have not encountered any substantiating research behind them, other than anecdotes and appeals to common sense. This is an area that clearly needs better study, which will hopefully provide better insight into how and when those attributes are best manifested, as well as substantiating the work that has already been based upon them.

Substantiation aside, these guidelines *are* more specific than the other, more general frameworks, in that they move beyond a description of the stages that must be supported, and begin to speak about principles that ought to underlie actual software architectures. But as prescriptive design recommendations, they fall short, in that they do not recommend any specific architectural features. How does one design for low-cost experimentation? What architectural components provide low thresholds or high ceilings? How is collaboration support introduced without interrupting solitary

exploration? On practical details such as these, the nice-house guidelines leave much as an exercise for the reader.

2.2.2 Facilitative Software

In their study of electro-acoustic composers, Eaglestone et al. [28, 86, 26] decry the scarcity of concrete data relating the subjective experience of artists into corresponding requirements for software tools, especially composers, and they issue a call for others to recognize this laxity and begin shoring up this deficiency in the field.

While their work is specific to a sub-genre of composers who are concerned primarily with signal processing and synthesis, their findings may be more broadly applicable — at least to other modes of composition. Their chief criticism of existing tools is the lack of support for multiple, parallel threads of exploration. In other words, they cite the inability of current tools to manage a multitude of musical fragments and projects simultaneously. The multiplicity of exemplars is important, they note, because each candidate, and the transformations applied to them, inform all of the others, and a much more efficient exploration of the space can be conducted if the inter-influencing set of ideas can be surveyed collectively.

The Simultaneity Premise: Creativity may be enhanced by tools that permit multiple content fragments to be developed and examined simultaneously.

A number of authors (e.g. [27, 69, 12, 84]) have commented upon the notion of serendipity and the importance of introducing appropriate elements of surprise or randomness into the creative process. A simple approach that is used often is to employ pseudo-random number generators to simulate non-deterministic decision-making within an operation, or to generate unpredictable content. MacCrimmon and Wagner, for example, use this technique to select random candidate ideas from a pool of relevant ideas for generating novel business solutions [68]. A common practice in this approach, however, is to present randomness as a selectable operation — a sort of “planned serendipity” — which is not really serendipitous at all, so I prefer to call these techniques “chaos tools.”

The Chaos Tools Premise: Creativity might be enhanced by providing users with access to unpredictable, pseudo-random content and/or operators.

True serendipity, as Boden defines it, is “the finding of something of value without its being sought [9, p. 234],” which seems somewhat at odds with the practice of asking users to elect when and where to employ it. This suggests that if true serendipity is to be employed in software, the tool itself must somehow be made to present alternative content while the user is otherwise engaged. The danger, of course, is that the distinctions between “spontaneously generated alternative content”

and “software bug” are both very small and entirely subjective, so great care would have to be taken to avoid offending the user if such tools were to be offered.

The Serendipity Premise: Creativity might be enhanced if the software can be made to spontaneously suggest valuable alternative solutions without irritating the user.

Wiggins gives us an elaborated view of Boden’s work on creativity, specifically applying it to the domain of music. In his essay, Wiggins describes both the exploratory and transformational forms of creativity as operations guiding a search through what Boden referred to as the search space of possible outcomes, and in this vein, Wiggins makes the specific point [120, p. 17] that for the transformational style of creative exploration to be effective, the searcher must be “in some sense” aware of the rules of the domain that are being applied.

More specifically, whether users have any formal training or theoretical foundation in the musical genre being explored or not, they are mostly likely to at least have a subconscious understanding of the rules, simply by virtue of being an experienced audience of that genre. We know that humans do not require formal training to be able to classify familiar music into genres. According to Fiske [33], this is done by innate absorption of the rules and structures of the genre. Clearly, once the features of a genre or style have been learned, audiences are then able to recognize them, even if they cannot articulate the rules, or consciously construct original exemplars of their own.

The Domain Familiarity Premise: Creative exploration can only be guided effectively within domains for which the user has at least an experiential understanding of its rules and traditions.

Fiske appears to be providing us with a definition of Wiggins’s required domain awareness that opens a new door. It may not be the case, as Wiggins implies, that the composer must have a conscious understanding of how to *apply* the rules of the genre, so long as they are somehow present in the constructive process, and he can recognize their presence or absence when he hears them.

The Rule Presence Premise: Creativity is possible in a domain for which a user has no theoretical training, so long as its rules are present in some other form.

Hyperscore [30] is a commercial product arising out of research at MIT to specifically address creativity of score composition for both novice and master user alike. Its primary interface (shown in Figure 2.1) presents a two-dimensional, visual sketch metaphor in which precomposed musical fragments can be positioned onto a time/pitch canvas by drawing contour lines. These contour lines are used to loop the fragments in time, according to the horizontal extent, as well as to modulate the pitch and/or key in response to variations in its vertical extent. To use Myers’s terminology, Farbood et al.’s Hyperscore has a low threshold, but, by its creators’ own admission, Hyperscore’s ceiling is not particularly high, nor are its walls especially wide. Auditioning half a dozen sample outputs available on the Hyperscore web site (<http://www.hyperscore.com>) and on

the Toy Symphony project’s web site (<http://www.toysymphony.org>) confirms this.

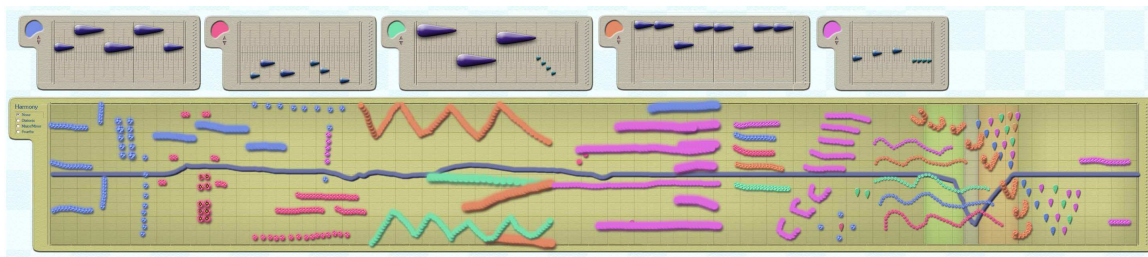


Figure 2.1: Hyperscore user interface

These compositions, however, seem to rely heavily on a very limited number of transformations — pitch modulation and looping — and while greater variety can be achieved by writing additional atomic fragments to work with, doing so in a manner that is artistically consistent with the other fragments in the working set requires a higher degree of musical skill than is assumed in their target user audience, who tend to be children. The fact that the auditioned works all seem to have at least a basic harmonic consonance is a testament to the rules of harmony that Hyperscore embeds in its contour curve interpretation algorithm, but this is also a weakness, because the harmonies are all very similar and safe. Clearly, rules of music theory can be successfully embedded in a tool, and exploited by its users in a “low threshold” manner, without any real understanding of what they’re doing, but Hyperscore does not seem to offer a sufficient variety of other such well embedded rules to sustain the diversity and flexibility that are necessary for full creative facilitation.

Sam Reese surveyed the world of creative composition software available for children [89], and found what he felt were several promising entries. Some, like *Songworks II*, were able to offer creative suggestions, such as melodies to fit a hand-entered harmony, but such auto-suggesters are extremely limited, and tend to produce extremely similar suggestions. Other systems, such as *Making More Music*, offer a library of static melody fragments that can be positioned, modulated and overlaid.

In essence, these tools all seemed to fall into varying combinations of notation editors and sequencing systems and are consequently quite limited in the types of experimentation and exploration that can be undertaken. Furthermore, tools such as Hyperscore that *do* provide domain rule guidance, do so in a manner that hard-codes the rules into the workings of software itself, thus ensuring tight stylistic constraints on what types of music can be sketched which limits the utility of the tool as the user gains experience.

The Rules-as-Content Premise: To maximize creative expressivity, tools should treat domain rules as content, rather than as design constraints.

Bruce Jacob made the observation [51] that the process of creative elaboration and exploration in music is inherently algorithmic in nature. In his argument, Jacob observed that many musi-

cally valid variations can be produced from fragmentary content by applying simple algorithmic transformations to them, and that the composition as a whole can then be artfully expanded by a recursive, hierarchical application of this process.

2.2.3 Sketching

Contrary to creativity frameworks — which tend to be built from the top down, using cognitive psychology’s theories of creativity — sketching systems are more bottom up in origin, starting from observed artistic practice. As their justifications for this approach, researchers in this area typically emphasize the various types of low-fidelity experimentation that pervade the traditional creative processes, though they are not blind to the psychology arguments.

Gabora characterizes the human creative cognitive process [36] as beginning in its earliest stages with the net widely cast, suppressing finer details in favour of broad interpretations and a generous filter for selecting those ideas that are tangentially relevant to the thought at hand. This view corresponds well with the characterization of sketching given by Wong [121], in which she describes the process as offering low fidelity representations that support broader interpretations and more general or higher-level associations.

Other researchers [59, 2] point to the rapidity with which sketching allows ideas to be created and evaluated as its chief value. This, too, corresponds with Gabora — particularly with her notion that associative ideation takes large, rapid strides through the mental concept space. Additionally, this attribute of speed reinforces Csikszentmihalyi’s observation [21] that such rapid experimentation and evaluation cycles are essential to the establishment and maintenance of flow.

The Rapid Generation Premise: Creativity may be enhanced by processes that produce candidate ideas quickly.

The Rapid Evaluation Premise: Creativity may be enhanced by processes that permit users to quickly evaluate candidates and decide whether to keep or reject them.

Another issue mentioned by both Wong [121] and Landay [58] is the notion I call completion cuing in which high-fidelity representations displayed during the earlier stages of creative exploration have a tendency to short-circuit the creative process by inappropriately conveying the suggestion that the work is more mature than it really is. According to Wong [121], the power of sketches is that they deliver crude representations, which, in addition to properly signalling the incompleteness of the idea, can also, by their lack of refined detail, represent a broad class of ideas, rather than just one. This aligns well with Boden’s notion of the minutia-agnostic nature of the associative-thinking comparison function, which is also reiterated by McCloud in his discussion of the general principles behind the universality of sketch representations in the visual arts [73].

According to Landay and Myers [58], overly specified representations, when introduced too soon into the creative process, discourage continued revision and refinement of the expressed ideas.

The Completion Cueing Premise: Early stage creative exploration is better served by low-fidelity representations that convey a less polished, incomplete impression of the content, encouraging further development.

Several commentators [34, 59, 71] also highlight the requirement that sketch representations incorporate multimodal linguistic data, in order to provide annotative and historical depth to the more abstract representations of the sketch itself. Furthermore, by allowing users to contribute this annotative content in the language and vocabulary most relevant to themselves and their peers, its explanatory power is increased greatly, along with its potential influence over the creative process.

Extrapolating from this, the value of annotations to the creative process appears to lie in the fidelity with which they allow contextual information to be recorded, shared and re-absorbed by the artist, which in turn minimizes false starts and time-consuming interruptions when changing from one context to another, thus helping to preserve flow.

The Annotation Premise: Creativity may be enhanced by allowing users to attach annotative notes to the content at various levels of granularity, and in an unconstrained manner.

Verstijnen et al. articulate two different modes of creative thought: restructuring and combination making [115]. In their analysis of the two modes, sketching proves particularly well suited to restructuring, although their analysis is entirely in the context of visual sketching, and whether or not it extends to broader interpretations is unexplored.

2.2.4 Conceptual Focus

When we look at sketching behaviours in the different arts, we generally see a restriction of the sketch to a single conceptual subject. A painter's sketch, for example, might focus on a specific detail, such as the expression on the face in a portrait (see Figure 2.2(a)), or the billowing pleats of a dress (see Figure 2.2(b)). Such sketches often emphasize a particular detail from the overall work, with other elements either simplified or omitted altogether.

In other cases, the sketch might address the entire canvas, demonstrating that the manner in which the sketches simplify the content need not take the form of spatial culling. Instead, other conceptual dimensions of the work might be constrained. In the case of a whole-canvas cartoon, such as that shown in Figure 2.2(c), the subject of the sketch tends to focus attention on one or two global features of the work, such as the approximate relative positioning of masses and blocks of shape or colour, or the implications of light and shadow play across the canvas.

Within musical sketching, we see a similar narrowing of focus to a limited set of ideas. A sketch might explore a localized bit of rhythmic by-play (see Figure 2.3), or the more global symmetry of

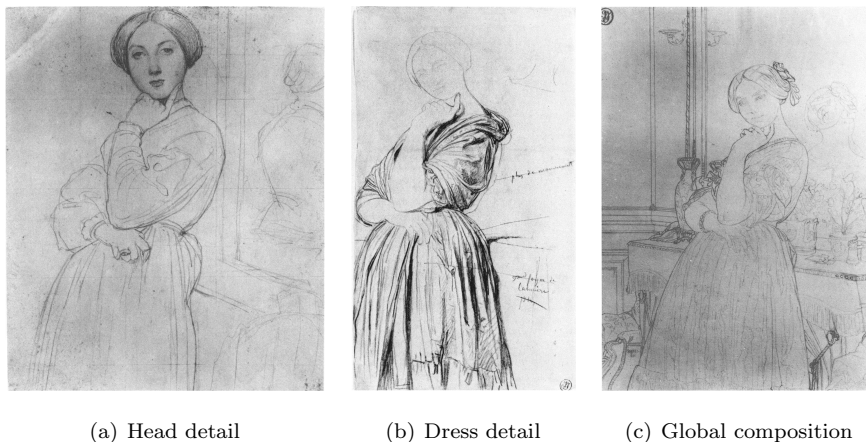


Figure 2.2: Ingres' *Study for the Portrait of the Comtesse de Haussonville* [39]

movements throughout the piece, such as that shown in Figure 2.7.

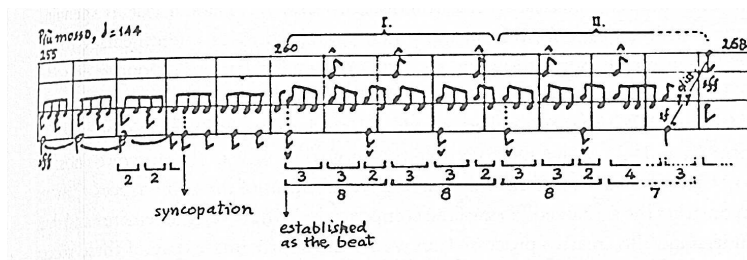


Figure 2.3: Ending of Bartók's *Piano Sonata, Movement I* [107, p. 175]

The layered devolutions of musical complexity expressed in Heinrich Schenker's musical analyses [99] can be seen as analogous to the composition sketches used by painters. For example, the fragment in Figures 2.4(a) and 2.4(b) are successive simplifications of a 16-measure excerpt from Brahms's *Intermezzo, Op. 117, No. 2*. At least one artist [10] has reversed this process of simplification to compose new works using successive embellishment — a technique that is described as being a reverse Schenkerian process.

While sketches can encompass the entire spatial (or in this case, temporal) span of a work, they usually do so with greatly simplified content, emphasizing the contrasts between the major components, rather than conveying the exhaustive detail of the full score, which would only serve to obscure the larger-scale contrasts that are the point of the sketch.

The Sketch Simplicity Premise: Early-stage sketches should focus on a specific issue within the work, localized either temporally or conceptually.

(a) Simplified

(b) Further simplified

Figure 2.4: A Schenkerian sketch of a Brahms score [11, p. 9]

2.3 Music Composition Tools

The history of composition — even before the appearance of computers — is full of attempts to produce tools that reduce the onerous barrier of required experience and guide novices toward compositions of their own. I have characterized these tools into a number of (somewhat overlapping) categories, which I will now discuss.

2.3.1 Notation Editors

Notation editors are one of the main tools in the modern digital composer’s toolkit. Tools such as Finale, Sibelius and Rosegarden present the user with one or more blank musical staves and permit them to place and manipulate notes, rests, time signatures, dynamics, tempos, key signatures and even instrument assignments. The result is that, in skilled hands, almost anything that can be played on an instrument by one or more musicians can be encoded in a publishable and repeatable form. Since most of these systems integrate well with MIDI devices (both as input and as output), the user is able to input music easily on a MIDI-enabled instrument, and even to audition his work automatically, by having the MIDI system play the score back for him.

The downside, of course, is that with all of this power and flexibility, there is almost no hand-holding, and there are just too many ways for the unskilled user to violate those rules and structures by which cacophony is transformed into music. When presented with a blank slate, the novice user will find it difficult to create anything that is not horrible, unless it is trivially short and/or boring. Conversely, if novices are given pre-existing scores as input, the output they are generally able to

achieve is only trivially divergent from the source material. As a class, notation editors tend to expect too much sophistication from their users to be useful for neophyte composers.

2.3.2 Template Tools

Template tools provide a template into which the user is expected to insert musical fragments, and a simple process for doing so. One of the earliest such designers was Guido d’Arezzo, who published the *Micrologus* in 1026, which produces melodies for church music [5]. His method assigned pitches for each vowel in an input liturgical text, and relied on the reading of the text to provide the rhythm. The resulting music was entirely monophonic and quite dull by modern standards, but it did permit unskilled clergy to create singable melodies, and imparted a consistency to the material sung in the various churches in which it was employed.

Another template tool, usually attributed to Mozart, is the *Musikalisches Würfelspiel* [78], which was popular enough to inspire a number of other composers to create *würfelspiels* of their own. A *würfelspiel* consists of a pool of musical fragments, a metrical template of some kind, and a rule for placing fragments from the pool into the template most commonly involving the roll of dice. Mozart’s *würfelspiel* offers over eleven trillion possible compositions, so each one produced is likely to be an original, in the sense that it has never been heard before.

The music generated by a *würfelspiel* is much more complex than that of the *Micrologus*, and to listen to any one output, the average listener would be unlikely to identify its synthetic origin. Their downfall, however, is that all the pieces produced by a given *würfelspiel* are identifiably similar, and their novelty soon wears thin, as can be heard in *Dice Game #1*² and *Dice Game #2*³.

2.3.3 Sequencers

A class of software used widely today is the digital audio workstation (DAW); examples include Cubase (www.steinberg.net), Ardour (www.ardour.org), and Ableton Live (www.ableton.com). DAWs typically allow users to manipulate both MIDI and raw audio signal data together, in a timeline-oriented interface of some kind. I have divided them into two distinct parts: the sequencing part, and the signal processing part. A sequencer, fundamentally, is a tool that permits multiple fragments of music to be organized over time. The fragments can be played simultaneously, sequentially or both, and larger fragments can be assembled by repeating and layering smaller sections into larger ones.

While some sequencers can be used effectively by unskilled users, the bottom-up, quotation-based construction scheme makes it difficult to achieve results that differ much from the general sound of the constituent samples. Using a tabla drum pattern and sitar phrase as inputs, an

²Refer to “*Dice Game #1*” in the *Wheelfragments* folder of the companion CD.

³Refer to “*Dice Game #2*” in the *Wheelfragments* folder of the companion CD.

unskilled user's output will very likely feel strongly Indian, and will quote the source material directly and frequently. Since the only melodic elements available are those encoded within the input phrases, it would take quite a bit of skill to evoke new melodies by segmenting and recombining the inputs.

Generally speaking, these tools are easier for novices to use than notation editors, but they still require too much experience to produce original music in a wide variety of styles and genres.

2.3.4 Signal Processors

The signal processing features generally treat music as digital wave-forms, allowing users to apply a variety of transforms and filters, to yield artistic audio effects such as reverberation, echo, pitch bending, or distortion.

They can also be used to do quite weird and wonderful things to the sound that are almost impossible to describe. Tools like Kyma [98] or Max/MSP (www.cycling74.com), while not limited only to signal-based manipulations, seem particularly powerful in this dimension, and their interfaces for managing the complexity of structured signal manipulations were a contributing influence in my own thinking on constructural encodings, which will be developed in the next chapter.

Other tools, such as Xenakis's UPIC [122] or Roads's Pulsar Generator [95] embrace quantization of sound not as discrete pitches, but in microsound bundles known as granules. But granulated or otherwise, the increased complexity introduced by wave-form manipulation makes this class of tool rather likely to produce frequent barrages of explosive cacophony, and thus, renders them more likely to frustrate novice composers.

2.3.5 Automated Composers

The field of artificial intelligence has explored many approaches to synthetic composition. Automated composers are my classification for those systems that attempt to build structure from a working knowledge of the rules of music.

Some operate from whole cloth, attempting to build viable works from nothing but a set of rules, such as Hirata's attempts [44] to build compositions from Lerdahl and Jackendoff's generative grammar [63]. Others take in fragmentary information and attempt to construct fuller works around them. For example, Biles's GenJam [8] takes harmonic fragments as input and attempts to create appropriate melodic accompaniment using evolutionary algorithms, whereas Ebcioğlu takes the opposite tack [29], generating harmonic fragments to complement input melodies, using BSL, a first-order predicate logic programming language.

Perhaps the most successful of the automated composers is Cope's EMI system [17], which blends the two approaches, taking a number of compositions from one artist as input, and producing complete new works as output that convincingly mimic the style of the input group.

Music can be composed from the top down, starting with choices of genre and musical forms, and working from there down eventually to choices of key, theme and ultimately motifs and specific notes. It can also be composed from the bottom up, beginning with a particular chord or melodic fragment, and building upward using repetition, variation, modulation, etc, until an entire work has been crafted. Most commonly, some combination of these two approaches are used, as the attention of the composer bounces back and forth between the lower and higher levels of abstraction.

This grouping of lower level fragments into larger upper level passages represents a kind of structural framework for the composition, and these structures form the basis of David Cope’s work [16, 17] with *EMI*. By extracting these types of structures, which he calls *signatures*, from a composer’s works, he is able to build a type of template from which new works in that same style and genre can be composed through recombination. To many listeners, both master and novice alike, the new works sound authentic [18]. They also sound representative—not only of the genre for which they were produced, but for the artist himself. In fact, if works from other genres or other artists are included in the set of input examples examined, the signatures extracted tend to become simplistic and the music resulting from recombining those signatures are generally considered to be failures.

This suggests that it is not the notes themselves in a musical score that make for compelling, successful, creative music, but rather, it is the relationships between them. Western music is built strongly on the traditions first formalized in counterpoint, and anyone familiar with counterpoint will know that it has rules, such as those articulated by Fux [35], and that they are quite restrictive. Compositions that deviate very far from the established rules quickly begin to sound unsuccessful — or at least not much like counterpoint. In the early years of the 20th century, Heinrich Schenker [99] predicated much of his theory of musical analysis on deconstructing these musical structures, and examining how those structures differed among composers and between different genres and eras. This notion was taken even further by Lerdahl and Jackendoff, who suggest [63] that much of western tonal music can be represented by a hierarchical grammar that encodes music using similar structures.

By extension, I think it is instructive to view music not as a function of the notes themselves, but rather, as a function of the relationships that pertain between and among them, and that are embedded within the musical signal itself. In addition to Cope’s *EMI*, Hoover and Stanley’s *NEAT Drummer* [46] also demonstrates that such embedded signals can be extracted procedurally and repurposed.

The Constructural Representation Premise: Hierarchical transformations of fragmentary content can be used to construct structural representations of musical content that explicitly encode the relevant domain rules used in their construction.

Each of these rule manipulating systems succeeds, to some degree, in producing non-trivial

output music, but none of them gives the user much control over the output. While these systems may strike deeply into questions of how computers can be made to compose music, they do not offer much artistic satisfaction to those who would seek to use them as a tool for composing music of their own.

2.3.6 Exaptive Systems

Biologists use the term “exaptation” to describe the process whereby some evolutionary structure or feature is co-opted as a solution to a different problem, for which it was not originally developed. The commonly cited example is that of feathers, which originated as a temperature regulating structure and were later repurposed to assist in flight. Researchers into artificially intelligent music composition systems do something similar, developing algorithms that take their structure from other, non-musical sources, processes or systems. I call such approaches “exaptive” composing systems. Gogins, for example, extracts fractal structures from iterated function systems [38], whereas Fischman’s investigations [32] create music by borrowing structure from Schrödinger’s equation for atomic potential with radial symmetry. Both of them then attempt to apply their borrowed structures as foundations for musical compositions.

The outputs of these systems can often be compelling, and they are certainly not likely to have been heard before, but they are quite hard to predict and control. Furthermore, the structures borrowed from such host processes do not generally conform to any familiar rules from musical culture, which gives the outputs resulting from them a distinctly experimental sound that, in my experience, tends to be rather limited in stylistic breadth.

The Loose Typing Premise: Creative inspiration can be achieved by exapting structural patterns from other domains into musical structures. Hence, musical creativity might be enhanced by treating content as being loosely typed.

2.3.7 Music Programming Systems

Perhaps the most powerful composition tools available, in terms of the expressive range achievable and the compactness of the representations, are the programming languages, such as Haskore [48], Common Music [110], C Sound, Max/MSP [123], ChucK [118], or the LISP front-end GUI systems like Patchwork [60] and OpenMusic [6]. These systems permit music to be manipulated to produce almost any output imaginable. But just as signal processors increased the domain of knowledge needed to use them effectively, the programming languages go even further in this regard, adding the need for familiarity with computer programming in order to exploit the tool productively.

2.4 The Composing Process

If I am to use these above-summarized, general theoretical premises of creativity to facilitate music composition, it is perhaps prudent to first confirm that they actually apply to the specific case of *musical* creativity, and that they do so for a large cross-section of experience levels and genres, since my goal is to facilitate such a diverse group.

I have therefore chosen three representative cases to investigate: western professional composition, beginning song-writers, and composition from non-western traditions. For each case, I will present a summary of the artistic process employed by one or two practitioners of that description, and assess the fit between abstract theory and the specific practices summarized.

2.4.1 Professional Symphonic Composition

Béla Bartók

Béla Bartók was a Hungarian composer working in both Europe and the United States throughout most of the first half of the 20th century. Figure 2.5 shows the sequential stages of creative development employed by Bartók, as reconstructed by Somfai [107]. This process, documented by the many fragmentary and intermediate drafts left in the composer’s archives, begins with a principal creative process, which then passes on to a number of successive refinement and correcting phases. These subsequent stages involved some degree of content refinement — particularly in stages marked 3_A and 3_B in the figure — but they were primarily engaged in more typographic and editorial concerns than they were musical.

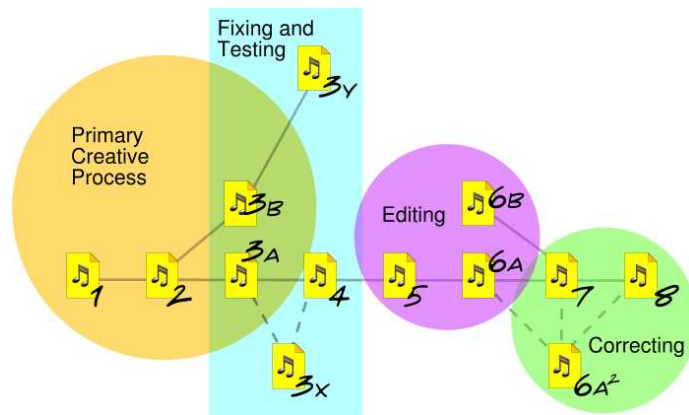


Figure 2.5: Bartók’s overall process [107, adapted from p. 29]

Examining the primary creative process in more detail (see Figure 2.6), note that the process begins with two stages of raw ideation. First were the fragmentary memos — brief melodic or thematic ideas, or notions for large-scale structure — that Bartók wished to explore. For example, consider the symmetry of movements diagram shown in Figure 2.7. Note especially that this sketch

has absolutely no references to pitch, rhythm or any other note-event attributes. The scope of interest is at a much higher level of abstraction.

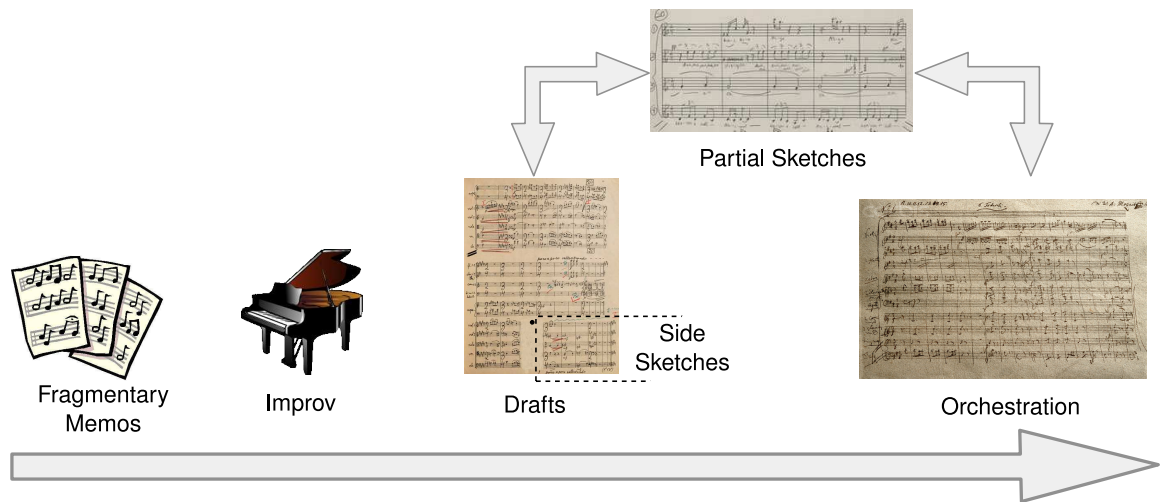


Figure 2.6: Bartók’s primary creative workflow [107, adapted from p. 34]

The second stage of primary creation were the improvisations, used to explore the ideas within, or extending those hinted at by the memos. This process yielded a series of what Somfai calls drafts, which were typically focused on particular sections or voices of the intended work. Often, these drafts would contain side-sketches, which included very specific illustrations of particular elements from the drafts, which themselves were less rigidly and thoroughly detailed.

Building from those drafts, Bartók moved toward a full-scale orchestration, but often employed what Somfai calls partial sketches along the way, which were more thematic than detail oriented, somewhat analogous to the different layers of abstraction produced in a Schenkerian analysis. Information flow at this stage tended to be bi-directional, as Bartók moved back and forth between the different levels of abstraction. It is interesting to note that the side-sketches were extremely detailed, but quite fragmentary in that they focused on specific note sequences for one or two voices over a very short range of time, whereas the partial sketches were much more abstract, in terms of note-event details, but were much broader in scope, often encompassing entire movements. Quite clearly, we can see Bartók relying equally on sketches constrained in duration as well as those constrained in resolution.

The Multi-resolution Premise: Experienced composers need to be able to work at multiple levels of resolution or abstraction, and to move between them arbitrarily.

Bartók’s process stands as a fairly typical example of the general creative process, outlined by Gabora, in which broad early-stage creative explorations are refined in a continuing, but not necessarily linear, progression toward later stage refinement and polish.

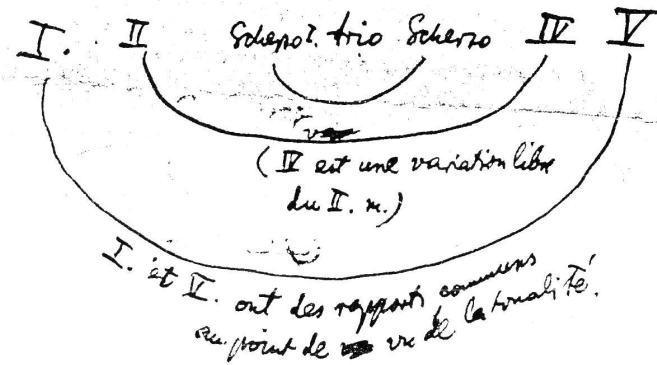


Figure 2.7: Fragmentary memo [107, p. 20]

Wyndham Thomas

Wyndham Thomas is a composer and musicologist at the University of Bristol who has worked throughout the later half of the 20th century. He gives several insights into his own creative process [112], which he describes as proceeding through two distinct phases: conception and development. He describes the conception phase as follows:

Let us suppose that I am beginning work. I am hearing sounds and colours in my mind's ear. I sense the shape, the progress of the piece as a whole, and I hear and see the rise and fall of the lines. I also have an overall sense of what is possibly best described as the music's harmonic orbit — its internal tensions, internal rate of change and its inner energy. [112, p. 3]

Notice that the terms used to describe this early process are indicative of a high level of construal. Development, on the other hand, is more localized and linear in scope.

At this stage, I move from an aerial or global hearing and view of the piece to a temporal one; I am walking along a road, rather than memorising its width and contour from the air. In other words, I am composing sequentially — time becomes linear, as opposed to the 'frozen time-frame' of Stage One. [112, p. 4]

While this comment does not move to more note-centric terminology, it *does* emphasize a shift of perceived distance, toward a closer, more detail-oriented awareness, echoing the creativity construal notions of Jia et al.

All artists sketch. To all of them their sketches have the meaning of a tryout, of initial planning. The painter chalks out form and space on the canvas, and distributes tentatively the colors of his palette. Certain sculptors first make a sketch in clay or wax, experimenting with the qualities of mass, contour and flow which they expect to integrate into the envisioned work. The architect plans his areas and dimensions on testing drafts before he can entrust them to the final blueprint, and later proceeds to the real stage of building.

The musician, too, feels his way through similar stages of testing. He sketches his tonal ideas in terms of melody, harmony and rhythm. He outlines the form and marks the tone color. To him, as to all artists, the sketch is a means of seeking and finding:

it serves the purpose of a preliminary orientation prior to further and final stages of synthetic work. [24, p. 207]

Again, we see evidence of the importance of high-level, abstract engagement in the early exploratory phase. It is also worth noting Thomas's emphasis on the importance of sketching as an apparent manifestation of this process.

2.4.2 Advice for Novices

Multi-tiered Engagement

In a brief tutorial published online by the Bloomington School of Music [85], composer Nadje Noordhuis offers some practical, albeit simplistic, advice for beginning composers. In it, she encourages artists to immerse themselves in the kind of music they wish to compose, paying particular conscious attention to the structures of the material.

The initial steps of her recommended process privilege the global-scale decisions first, such as mood, tempo, genre and movement structure. She then advises moving into a fragmentary ideation stage, in which short elements, suitable to the declared global goals, are invented and manipulated. This is to be followed by selecting a single idea to serve as the primary motif, which is then elaborated and developed to create larger and larger structures, for which a contrasting second motif is then selected and developed, culminating in a final integration and polishing phase.

This process clearly demonstrates the notion that both top-down and bottom-up processes are employed simultaneously, demonstrating that the Multi-Resolution premise applies to novices as well as to experienced composers. It illustrates the process of general ideation progressing into more analytical development and refinement, and makes clear the necessity for basing the work in a historical context, adapting existing high-level structures from a culture or genre to frame the development of a new individual work within that same context.

While the threshold is indeed kept low in this recipe, it does, however, assume a degree of musical facility on the part of the composer. This is evidenced by the advice to settle on an instrument and tonality early in the process, to select appropriate musical structures, and by the recommendation given in the sidebar to work with a common-staff notation tool.

Noordhuis's recommended process differs from those of previously discussed composers only in detail. She still advocates for more abstract, high-level decision-making in the early phase, proceeding to more detailed, concrete activities later. Where it appears to differ most is in the assumed musical sophistication — both of the materials being developed, and of the composer's knowledge of music theory.

Iterative Deviation

A different process, commonly offered to aspiring songwriters, is the method I call iterative deviation, which assumes even less musical training than Noordhuis's approach. In this second protocol, novice composers are advised to start by selecting an existing work from the genre they wish to pursue, and to make successive, incremental changes to the melody, harmonic progression, and/or the rhythm of the music, until they have achieved something sufficiently distinct, and then to write new lyrics to suit the new material.

On the surface, this is an exceptionally simplistic protocol, but it in fact formalizes a process that many composers use, borrowing and adapting ideas from their predecessors and contemporaries. As Stravinsky is reported to have said, "Lesser artists borrow, great artists steal." An attribute of iterative derivation that is rarely discussed is that, by employing completed works as its input, most of the relevant rules of music theory and aesthetics that apply to the genre are already embedded and therefore do not have to be explicitly provided by the novice composer. Not only will this result in output compositions that are more mature than a bottom-up process might have delivered in the hands of a true novice, but the resulting process of close-up study and experimental modification is almost certain to teach him some of the rudimentary rules of structure and style relevant to the chosen genre along the way.

On first analysis, this iterative derivation process seems to contradict creativity theory, in that it appears to start immediately with modifications to specific details of the work, but we need to look more closely at several assumptions. First, we are dealing here with composers who are presumed to have little or no musical theory training. Consequently, abstractions such as tone, mood, symmetry, etc., are not likely to be especially meaningful to them. Instead, all of these considerations are buried within the very first step: selecting the source work from which they will deviate. This step may not be particularly time-consuming, but it does occur, and it encapsulates a great many issues of abstract decision making.

The second limitation such users have is that they are not expected to have sufficient musical experience to know which kinds of changes to the material are likely to produce what kinds of results. So while the advice to try changing the rhythm, or the melody are indeed more specifically focused than the early steps would be for an experienced composer starting from scratch, they are still more broad and abstract than suggestions to try changing specific notes, or specific timing values, which are in fact the sorts of activities that will come into play in the later stages of this process.

In practice, then, this process still exhibits an abstract-to-concrete transition. The gap between construal levels may seem slight to an experienced musician, but in the face of relative musical

ignorance, they are much less so.

The Incremental Deviation Premise: Novice composers can be facilitated by a process of incremental modifications to existing works.

It is also important to observe that this iterative deviation process clearly offers the neophyte composer something that the professional processes do not: concrete guidance regarding the rules of music appropriate to the genre in question. That guidance is embedded directly into the source material and this idea of embedded, implicit knowledge will be used again later.

The Domain-savvy Targeting Premise: Novice composers can be further facilitated by a system that focuses their attention on creatively fruitful targets for modification, within the rules of the musical domain.

2.4.3 Non-western Composition

Michael Robinson

Michael Robinson is a modern composer working primarily in the genre of Indian classical raga music, and he has published several first-person accounts of his working process. He describes his process as very traditional, despite the fact that his chosen instrument is a computer program that performs his works, under his control.

My creative process begins with the conceptualization of a composition, frequently inspired by a live performance, or recording of a raga by an Indian master. This conceptualization may take anywhere from minutes to years. [96]

Robinson’s description of this process paints his first phase as one similar to Bartók’s, in which the starting elements relate to high-level issues such as a form, mood or even a specific timbral flavours.

Once I am ready to proceed with my new composition, I feel like the music has already been internalized, though the specific details need to be rendered. Now the second phase begins, and I set about composing a complete, fully notated score using a personalized form of Western musical notation. I compose using a mechanical pencil, music paper, and metronome, without the aid of any musical instrument. [96]

Clearly, a progression is evident from the abstractions of the first phase to a more note-specific development phase in which the “specific details need to be rendered.” Note also that, like many other composers, Robinson’s early phases do not make use of computer software. This is especially noteworthy, given that the computer is not only used later in his process, but is in fact his performance instrument of choice.

When the score is completed, I enter the third phase, translating the musical score note-by-note into a numerically based software program. All of the musical elements,

including pitch, rhythm, timbre, tuning, dynamics, articulation, tempo, and spatial placement, are programmed in this third phase of the creative process.

[96]

And here we see Robinson's final, most concrete, detail-oriented phase, in which he transcribes and adjusts every aspect of each note. Despite the non-western traditions of the genre, Robinson's process appears no less conformant to the descriptions of creativity theory than those of western composers.

Gamelan

In their summation of modern compositional practices in the genre of Indonesian and Javanese gamelan music, Sadra et al. describe two distinct approaches [97]. In one, new works are heavily influenced by older traditional works, although they are quick to point out that this practice does not necessarily translate into derivative works. Instead, they describe it as a necessary, continual advancement of the traditions. This practice aligns neatly with the theoretical premise that all acts of creativity are inherently collaborative, borrowing from, and building upon prior works in the field.

But this tradition-building process does not lack for creative inventiveness. The authors cite many examples in which the very foundations of the traditions are now being employed in ways that were not permitted by the historical rules. For example, multiple traditional tuning systems are now sometimes placed into the same composition, as are rhythmic elements taken from different gamelan sub-cultures. Such practices would have amounted to cultural heresy in the past, but today these are perhaps best seen as examples of Csikszentmihalyi's juxtaposition principle in action.

The second school of composition, according to Sadra et al., is a more avante-garde, experimental approach which self-consciously seeks to set aside the historical rules and traditions in an attempt to explore more fully the range of effects and moods that can be achieved with the instruments. Such compositions, they tell us, are much more personal and expressive of the individual composer than are the more socially-informed traditional works.

Descriptions given of these experimental practices repeatedly begin with issues of structure, form, exploration and trial and error — precisely those high-construal level issues that theory posits for early stage creativity. Subsequent phases also follow theory, proceeding to integration of ideas and moving then towards refinement, then performance, notation and revision.

Gamelan composition, it would seem, can also be placed in alignment with theory. And while this section's examination of different compositional practices has not been exhaustive (and therefore cannot be construed as proving that *all* composing processes conform) there seems to be sufficient correspondence to justify using the tenets of general creativity theory as a foundation for my attempt to facilitate creativity in musical composition.

2.4.4 Common Themes

One of the most vexing things about the subjective discourses on the creative process is how different they are from one another. Josef Haydn, for example, wrote about being inspired by a nature hike to transform a bird, a tree or a rock into music [24, p. 68], whereas Brahms reported a much less deliberate or conscious process, in which ideas flowed into him “directly from God, with complete harmonies and orchestration [80, p. 437].” Another composer I have spoken with talked about having experimented with a laborious and exhaustive process of auditioning all possible combinations of two or three piano keys, searching patiently for a new sound that he had not heard before.

Yet out of this chaos of disparate methodologies come the same results: competent, creative music. These processes are not sleights of hand or circus tricks — they are known to be successful, and not just in a strict, technical sense of success, but apparently as repeatable and recurring processes, at least for the artists who have reported them.

What is it about these wildly disparate recipes that somehow allow them all to reach the same end? I will not speculate on why they all work, nor on what other processes might be feasible, but I have noted a similarity among them—a way of looking at them that reveals a common characteristic.

Each of the processes I’ve examined (with the exception of those rare flashes of inspiration that deliver completed works in final form) operates on the same principle, in which a number of potential ideas are continually presented to the artist, who then employs his expert judgement to pick and choose those ideas that are worthy of further examination and refinement. In a sense, it is not the artist’s conscious, deliberate will that finds the ideas and notices their merit, but rather, his passive but ever-present and highly refined sense of artistic appreciation for the domain. Will and intent only come into play in the act of choosing the process and deciding when to engage in it. So long as it exposes his artistic sensibilities to a steady stream of ideas, it should work.

Artist #2 marches himself through keyboard combinations, #3 auditions tunes from his jukebox, and #4 picks up his instrument and lets his subconscious out to play. Very different practices, superficially, but in every case, there is a perceptive and attentive audience watching the entire process and waiting to pounce when the proper conditions are recognized. It is in the later stages, however, during development and refinement of those ideas, that focused will and intent take the dominant role.

The Act of Awareness Premise: Creative ideation is not an act of will — it’s an act of awareness; of attending to a stream of input and judging its potential in the context of a creative domain.

2.5 Synthesis

While I do not take Myers’ “nice house” dicta as having yet been proven necessary, I do agree that facilitating a wider number of potential users within a single tool is advantageous, if for no other reason than simple tool-building economics. To provide these attributes, we must strike a balance between the needs of novices and those of masters, which is something that few of the reviewed approaches have managed. For example, the full acoustic expressivity offered by the signal processing tools are of use to the masters, but provide too little guidance for novices. On the other hand, simple sequencing systems allow novices to stay safely within the bounds of musicality, but they are unlikely to lead master composers into new creative territory.

It seems plausible that a tool aimed at musical sketching might provide the proper balance — providing for the development of novel creative fragments to support the masters, but also exploiting that simplicity for the benefit of the novices. In considering the four basic degrees of musical freedom — pitch, time, volume and timbre — I propose that this sketching tool construe music in a very simplified form, limiting expression to just two of those dimensions: pitch and time. My justification for this choice lies in the observation that music from almost every genre and level of complexity has at one time or another been reduced to an “easy piano” version, which are very popular with novices; while at the same time, many of the most complex orchestral works created by composers, began their development as experimental fragments performed in sketch form on the composer’s piano. The easy-piano target, then, appears to be a sketch representation that will suit both ends of the musical experience spectrum.

If we look at the tools surveyed in Section 2.3, all the ones that seem appropriate to sketching these reduced-form composition fragments are atomic in nature, by which I mean that they are strongly oriented toward representing and manipulating the atoms of their medium — the note events within music — but not to working directly with the higher level structures.

By contrast, the systems that seem to have been the most successful at algorithmically synthesizing high quality musical scores were the structural ones, such as Mozart’s *Musikalisches Würfelspiel* and David Cope’s *EMI* systems, in which the object of manipulation was not so much the note events themselves, but the structures by which they were assembled and related to one another.

The reason for the success of these systems appears to be because the structures being manipulated encode many of the rules for artistic merit intrinsically. For example, a structure which interprets all pitches in a composition with respect to some base palette of pitches (such as a musical key) is more likely to produce musically appealing output than is a system in which any pitch is as likely to occur as any other.

Cope describes *EMI* as a sort of automatic würfelspiel builder that examines extant works in a given genre by a given composer, and extracts a set of structural heuristics that can be recombined

in different sequences, using different inputs, to produce works similar in style and complexity to the originals, without repeating any recognizable note sequences from any of the input works.

The question that immediately arises in my mind is whether we could facilitate human creativity in a similar way, by allowing humans (instead of AI algorithms) to manipulate music structurally, as constructive hierarchies of note transformations, rather than atomically. This will form the basis for my approach, outlined in the next chapter.

2.5.1 Prime Ideator

Looking at the foregoing efforts in creativity facilitation, I find that much of it has been influenced by the recent trends toward h-creativity and its twin components of collaborative artifact development and external, cultural validation. For example, consider how Shneiderman’s Genex model [101] transforms the fourth stage of Wallas’s original model from a mode of validation to be performed by unspecified agents, into a donation mode in which acceptance of the donation must be vetted by external parties — thus disenfranchising the artist himself from the verification process.

Mamykina et al. [71] acknowledge the contribution of individuals, but primarily in the context of their participation on interdisciplinary teams. They in fact go so far as to suggest that the dimension of p-creative facilitation is a “done” field, although no evidence is offered for this view.

Perhaps most distressingly, Fischer et al. state [31] that the role of the individual in creativity is highly overrated, and that it is the combined efforts of the society that are the true source of creativity. While I concede that an idea produced by any individual is strongly informed by a resynthesis of ideas that have gone before, drawn from a variety of sources, and that it might be improved upon by peers, human beings are not hive-minds – they do not share cognitive experiences. It seems obvious, therefore, that before the group can seize upon and develop an idea, it must first arise in the mind of one, single primary ideator who makes that first connection, develops it, evaluates it and then must decide whether to share it with others. The end result of many iterations of this process may indeed be collaborative, but those first, essential steps are taken alone. And it is this dimension of the creativity problem that I intend to explore.

So while the facilitation community seems currently disinclined to embrace the individual as an individual, I take heart from Boden’s acknowledgement [9, p. 2] that h-creativity is indeed a subset of p-creativity, in that before they can be judged by the community, ideas must first have passed muster with the primary ideator. I believe that this fact alone justifies continued research into facilitating p-creativity in software.

2.5.2 The P-Judgement Conundrum

There is, however, a challenge to overcome in facilitating the lone ideator in his or her quest for p_a -creative⁴ ideas. The problem is that no test of creativity can measure p_a -creativity unless it is applied and judged subjectively. That is, we can only know if we've succeeded in helping a user achieve p_a -creativity if he himself judges the work to be p_a -creative. Some research appears to acknowledge this view [27], whereas others seem to confuse the issue by inviting readers to evaluate the output produced, but never clarifying which style of creativity they were seeking to facilitate in the first place [30].

Any judgement of creativity made by an individual is, by definition, an assessment of p-creativity, since an individual is only able to assess merit within the context of his own knowledge of the field. Due to its complete subjectivity, such a judgement made by one observer in no way validates or invalidates the p-creative assessment made by another, although it may cause others to reassess their own views. Consequently, h-creativity can only be seen in the aggregate, as opinions in the field accumulate, interact and converge. During the process of creative production, however, the artist does not have access to the aggregate evaluation, and so must rely on his own opinion, or perhaps those of a small number of confidantes. Within the actual cognitive events of creation, however, even those confidante opinions are less accessible, leaving the artist, ultimately, alone in his own skull with his p_a -creativity. How then do we facilitate the creativity of a solitary artist while he is actually creating? It seems our best choice, in that moment, is to engage with his associative thought processes, attempting to facilitate his immediate p_a -creative sense of accomplishment.

The conundrum arises when we try to decide how we, the observers, can judge the process and any facilitations it may have afforded, after the fact. The instinct is to observe the artifacts produced and judge the facilitation in terms of our own assessment of their creative merit, but this would be a mistake, because *our* context of evaluation — our aesthetic values and experience — played no role in the artist's cognitive feedback loop during the creative act. Even if we were unanimous in our rejection of creative merit, at best, this would be an assessment of h-creativity — not p-creativity.

So, I cannot stress enough, that while artifacts produced by a creative process can be offered to illustrate creative merit or the presence of creative facilitation, it cannot be taken as evidence — neither supporting nor refuting. There is only one judge whose opinion of the work has any bearing during the creative process: it is that of the artist himself, at that time, and even his own later assessment cannot change the p_a -creative assessment made at the time.

The P-Judgement Premise: Only the artist can judge the p-creative merit of his own output,

⁴Judgements made by the artist during the act of creating, as opposed to those made after the fact. Discussed in further detail in Chapter 3

and any evaluations made by others are, by definition, either irrelevant, or are evaluations of h-creative merit.

This dilemma will have to be considered in the methodologies of any research aimed at exploring p-creative facilitation.

2.5.3 Bimodality Blindness

It is perhaps surprising that the distinct change in the user's mode of thinking (from the associative mode of the early stages to the analytical mode of later stages) is not discussed in the literature of software facilitation, nor is any mention made of the need for interfaces to acknowledge this shift in creative cognition by adapting the types of tools presented, or to modifying the manner in which the user interacts with them.

Perhaps this blindness goes some way to explaining the claim that p-creative facilitation is "done." [71]. When viewed through the lens of the more analytical processes of late stage creativity, which are emphasized in the h-creative model, creativity can be seen as extremely collaborative. And since the facilitation field is currently organized around h-creative facilitation, it may seem that the diminished role of the prime ideator has indeed been fully served. It is only when we turn our attention to the associative cognition of early stage creative exploration that the role of the primary ideator becomes more clearly essential and ill-served.

But regardless of any rationales for why it has not yet been addressed, it seems self-evident that if users' needs and styles of working are known to change throughout the course of their creative process, then the tools developed to facilitate those processes should be designed to support and exploit this shift.

2.6 Summary of Premises

Throughout this chapter I have extracted a number of observations relating to how creativity might be better facilitated in software. These premises will form the basis for the next chapter, in which I will develop my theoretical approach and methodologies. The premises are collected here for easy reference.

P-H Sequencing Premise: H-creative ideas must generally be judged as p-creative by their originator before becoming available to the field for h-creative scrutiny. (See p. 8.)

Contextual Dependence Premise: No matter how original, no idea is created in a vacuum of influence. New ideas can only be conceived and understood within the context of what has gone before or been done by peers. (See p. 9.)

P-Leveraging Premise: H-creativity might be enhanced by increasing p-creative output. (See p. 9.)

Cognitive Context Premise: Creativity may be enhanced by designing tools with an awareness of the cognitive modes that will be employed by the user at the time of use. (See p. 9.)

Cognition Induction Premise: Early-stage exploratory thinking may be inducible by directing a thinker's attention toward high-level, abstract representations, suggesting a greater psychological distance from the material. (See p. 9.)

Richness Premise: Creativity may be enhanced when large numbers of diverse ideas from diverse contexts can be placed in close mutual proximity. (See p. 10.)

Smoothness Premise: Creativity may be enhanced when manipulations of the subject material are quick and easy to achieve, minimizing cognitive disruptions to the process, and facilitating flow. (See p. 10.)

Satisfaction Premise: Creative productivity might be increased by processes that improve the level and/or rate of creative satisfaction achieved. (See p. 10.)

Visualization Premise: Creativity can be enhanced by providing multiple views of the content, emphasizing different structures or interpretations. (See p. 12.)

Simultaneity Premise: Creativity may be enhanced by tools that permit multiple content fragments to be developed simultaneously. (See p. 13.)

Chaos Tools Premise: Creativity might be enhanced by providing users with access to unpredictable, pseudo-random content and/or operators. (See p. 14.)

Serendipity Premise: Creativity might be enhanced if the software can be made to spontaneously suggest valuable alternative solutions without irritating the user. (See p. 14.)

Domain Familiarity Premise: Creative exploration can only be guided effectively within domains for which the user has at least an experiential understanding of its rules and traditions. (See p. 14.)

Rule Presence Premise: Creativity is possible in a domain for which a user has no theoretical training, so long as its rules are present in some other form. (See p. 14.)

Rules-as-Content Premise: To maximize the creative facilitation of novice users, tools should treat domain rules as content, rather than as design constraints. (See p. 16.)

Rapid Generation Premise: Creativity may be enhanced by processes that produce candidate ideas quickly. (See p. 16.)

Rapid Evaluation Premise: Creativity may be enhanced by processes that permit users to quickly evaluate candidates and decide whether to keep or reject them. (See p. 16.)

Completion Cueing Premise: Early stage creative exploration is better served by low-fidelity, representations that convey a less polished, incomplete impression of the content, encouraging continued development. (See p. 17.)

Annotation Premise: Creativity may be enhanced by allowing users to attach annotative notes to the content at various levels of granularity, and in an unconstrained manner. (See p. 17.)

Sketch Simplicity Premise: Early stage sketches should focus on a specific issue within the work, localized either temporally or conceptually. (See p. 18.)

Constructural Representation Premise: Hierarchical transformations of fragmentary content can be used to construct structural representations of musical content that explicitly encode the relevant domain rules used in their construction. (See p. 22.)

Loose Typing Premise: Creative inspiration can be achieved by exapting structural patterns from other domains into musical structures. Hence, musical creativity might be enhanced by treating content as being loosely typed. (See p. 23.)

Multi-resolution Premise: Experienced composers need to be able to work at multiple levels of resolution or abstraction, and to move between them arbitrarily. (See p. 25.)

Incremental Deviation Premise: Novice composers can be facilitated by a process of incremental modifications to existing works. (See p. 29.)

Domain-savvy Targeting Premise: Novice composers can be further facilitated by a system that focuses the user's attention on creatively fruitful targets for modification, within the rules of the musical domain. (See p. 29.)

Act of Awareness Premise: Creative ideation is not an act of will — it's an act of awareness; of attending to a stream of input and judging its potential in the context of a creative domain. (See p. 31.)

P-Judgement Premise: Only the artist can judge the p-creative merit of his own output, and any evaluations made by others are, by definition, either irrelevant, or are evaluations of h-creative merit. (See p. 34.)

CHAPTER 3

PROGRAM OF RESEARCH

In the preceding chapter, I extracted a number of premises from the literature that provide clues to different techniques and theories that might be employed in support of creativity, but they are not yet organized into any semblance of a facilitation theory. Most of them are about creativity in general, and are not yet crafted in terms of musical composition. They are also general enough that, through selection and emphasis, any number of theoretical approaches to facilitation might be marshalled together from them.

The next step in developing my own research agenda is to winnow those general premises down to a specific set of musically targeted, mutually consistent premises that will form the core of my own facilitation hypothesis, which I can then implement and evaluate. That process will be the focus of this current chapter.

I will begin, in Section 3.1, by articulating my basic research objective, expressed in relatively plain terms, and based upon a synthesis of the extracted premises. In Section 3.2, I will then define and clarify the terms of that preliminary objective, followed in Section 3.3 by a discussion of the issues of scope, culminating in Section 3.4, with a final, formal research hypothesis.

3.1 Preliminary Research Hypothesis

Music can be an intimidating field of creative expression [1, p. 326], and most people view themselves as entirely incapable of composing, preferring instead to leave such work to the experienced professionals [47].

Given this climate of perceived personal incompetence, it would seem foolhardy then, if we wish to take up Shneiderman's challenge, to follow the current fashion of facilitation research by pursuing collaboration-based solutions. Consider that – whether we're talking about neophytes approaching music for the first time, or experienced composers looking for new perspectives – artists of any calibre are reluctant to expose their ignorance to others [3], as would likely be required by any collaborative process.

This is not to suggest that our hypothetical users are unwilling to learn from the historical and contemporary sources that define the various musical domains in which they are interested, but

they must be free to do so in relative privacy to avoid the creative impulse being extinguished by the fear of premature scrutiny.

What appears to be needed then, is a tool that allows individual users to explore musical ideas on their own, drawing and/or building upon input and ideas from the historical/cultural context, but in a manner that leaves the user in total control and complete privacy. Furthermore, this tool should provide sufficient guidance about the rules of music to protect novices from the discouragement of frequent atrocious cacophonies, but at the same time, it should be flexible enough to allow more experienced users to violate or redefine those rules as needed.

I therefore propose to take a large step toward reaching this goal by designing and evaluating an architecture for a music composition tool guided by the following specific premises.

- The p-leveraging premise suggests that we may be able to improve h-creative output by facilitating p-creativity
- The under-facilitated phase of the creative process is that of early-stage ideation
- The cognitive context premise suggests that to facilitate early-stage creativity we should embrace associative and/or exploratory processes and tools
- By the act of awareness premise, one approach to facilitating early-stage creativity is to support the production of experimental candidates
- The rapid generation and evaluation premise and the smoothness premise both tell us that such candidate production should be quick and efficient
- By the richness premise, that candidate production should embrace a wide range of ideas, but the smoothness premise cautions us to temper that breadth by minimizing the cognitive intrusions of disruptively a-musical ideas
- Implementing the constructural representation premise should inherently conform to both the rule presence and rules-as-content premises, and should also induce creative reasoning, as per the cognitive induction premise

Taking these ideas together, I can now state a preliminary hypothesis for how musical composition creativity might be facilitated for both novice and master composers.

Hypothesis 2 (Preliminary) *A software architecture that encodes early-stage musical ideas as structural sketches, expressed in terms of the musical rules used to construct them, will generate a greater richness of p-creative ideas for users to work with, than are generated from atomically-focused systems, thereby producing measurably more competent and creative output, regardless of the degree of musical expertise of the user.*

This hypothesis, in this preliminary form, captures the spirit of what I propose to do. Before it can serve as the foundation for my research, however, it must first be developed more fully.

3.2 Terminology

The above-stated preliminary hypothesis uses a number of imprecise terms. In this section, I will refine these terms so that they can be used to specify the proposed research clearly. The following subsections are organized around the key concepts cited in that preliminary hypothesis.

3.2.1 Creativity

As discussed earlier, creativity can be divided into two different types, depending upon who is assessing the merit and when, with p-creativity judged relatively, in the context of what the creator perceived as new and valuable discovery, compared to h-creativity, which is judged by others in the wider context of what has already been done in the field. There is, however, a potential for vagueness in exactly what Boden means by the p-creative mode, which she does not address.

If an artist declares a work to be creatively satisfying at the moment of its creation, and then dismisses it the next day as derivative rubbish, are both statements to be taken as equally valid declarations of its p-creative merit? Both assessments speak to the artist's own subjective impression of the work and both are (presumably) grounded in the same awareness of the domain and its history.

The difference, in my view, is the notion of agency. The considerations that inform his judgement in the first case relate to subjective assessments that he made while he was immersed in the creative process—the instantaneous decisions that guided his creative feedback loop as it unfolded. As such, these considerations had a direct influence on the course of the creative development of the work. His subsequent rejection of the work arises from a more reflective assessment, and likely includes considerations beyond those that were psychologically active at the time of creation.

These two sub-species of p-creativity seem to hold importance differences. The earlier, active form has direct impact on the artist's creative output and is the part of creativity that Gabora equated with associative cognition. Conversely, the latter, reflective form seems more analytical in nature, and though it is less directly involved in creation itself, it seems to govern what I call the artist's *gate-keeping* function, in which he continually deliberates over the work, deciding whether or not it should be pursued further and/or whether it should ever leave his shop to be subjected to further, external scrutiny.

I will refer to these two sub-classes of p-creativity as psychologically *active* creativity and psychologically *reflective* creativity, and will reference them as p_a -creativity and p_r -creativity, respectively. In cases where I mean either or both forms, the subscript will be omitted. It is unclear to

me whether Boden intended her definition of p-creativity to include both cases or not, but I would like to be clear that the style of creativity referenced in my hypothesis is p_a -creativity. Given that this mode of creative judgement has the greatest influence over the immediate course of creative development, it seems the most potentially valuable form to try to facilitate.

3.2.2 Measurability

In order to measure success or failure in facilitating creativity, I first had to settle on a way to measure it. Surprisingly though, despite creativity’s long history of study, and attempts at measurement, there did not appear to be any widely accepted metrics. By 1989, Torrance and Goff [113] were able to identify more than 250 distinct testing tools in the literature. Plucker and Renzulli [88] comment on this apparent fragmentation, suggesting that it may be due to the lack of agreement on the basic definition of creativity. Despite the lack of agreement, however, they did note that of all the various approaches to measurement, expert evaluation was by far the most common scheme.

Within that broad category, however, there is still much to differentiate the candidates. Some, such as Besemer and O’Quin’s method [7], are very controlled, providing an explicit definition of creativity and set of criteria to guide the experts in their assessments. Other schemes, like Amabile’s CAT test [4], are less prescriptive, leaving the definition of creativity and related terms to the individual interpretation of each judge, in an attempt to capture the “I don’t know what it is, but I know it when I see it” nature of the subject.

None of the measurement protocols I’ve explored or seen references to have attempted to measure the ability of a system to facilitate creativity. Instead, they have all focused on evaluating personalities, practices, ideas, or artifacts for creative merit of some kind—except for one. In 2009, Carroll et al. [15] introduced their Creativity Survey Instrument (CSI), which they describe as the first tool specifically designed to measure creative facilitation in software—a tool they felt was needed to replace the more generic task loading and productivity measurement instruments that had been employed previously within the Human-Computer Interaction (HCI) field, and that they dismissed as being generally ill-suited to the specific peculiarities of evaluating creativity factors. Their approach is to solicit users’ subjective impressions of their experience with the subject software, evaluating such things as ease of use, the degrees to which the user felt cognitively engaged, creatively fulfilled, and able to express themselves effectively.

I have two problems with the CSI. First, their work makes little mention of any of the above-cited classes of pre-existing measurement schemes, demonstrating little awareness of the field. Perhaps these antecedents have been omitted because they are measurements of creative products, rather than of facilitative environments, but the lack of acknowledgement of such prior art outside the relatively new field of HCI is cause for concern regarding the depth of foundation upon which the

CSI is built.

Provenance aside, my second concern is that the CSI appears to measure creative facilitation indirectly, by assessing conditions known to correlate with creatively productive environments and tools, rather than measuring the creative merit of the output actually produced. In taking this approach, the CSI seems to commit the fallacy of affirming the consequent. If creative environments are known to exhibit attributes X, Y and Z, it does not follow that the presence of those attributes in another environment is indicative of creative facilitation. In the absence of accepted causal factors of creativity (and not simply correlated factors), it seems to me that the only way to assess the creative facilitation of a system is to measure the creative merit of its output.

The CSI leads users through a subjective assessment of the tool-using experience, after the fact, scoring environmental and behavioural factors that are known to correlate with creatively productive environments. At no time, apparently, were the subjects asked whether or not they considered any of their outputs to be creative. The closest approach to a direct question was in the final of their six questions, in which subjects are asked to score the degree to which they felt “...able to be very expressive and creative while doing the activity.”

As a measurement of a tool’s ability to make its users *feel* creative, the CSI may have merit, and I agree that *feeling* creative is an important aspect of *being* creative, but *feeling* creative and *being* creative are not isomorphic states, so I do not believe the CSI can be used to measure the facilitation of creative output, which is my goal.

Ultimately, the decision of whether or not to use the CSI was made for me, since it had not yet been published when I began my trials. Of the other existing measurement tools I reviewed, none acknowledged the differences between early- and late-stage creative processes, nor the distinct cognitive differences adherent to the two phases. Seeing no directly applicable options in the literature, I elected to develop my own metric, basing it on the prior systems, but also mindful of recent cognition theory, and of the early-stage nature of the creative context I was intending to assess.

For my measurement system, I employed the common expert-evaluation scheme. In Section 2.1, I presented a number of definitions of creativity that each highlights different aspects of its makeup, but citing a multiplicity of definitions only serves to incite ambiguity. These various definitions tend to circle one another, sharing some terms in common, omitting others, and sometimes clouding the issue further by the use of overlapping concepts. From them, I extracted five concepts that appear to be largely independent of one another, are exhaustive of the concepts embraced by other definitions, and appear to be measurable, in the context of early-stage experimentation.

The concept of “newness” is present in more complex concepts like “originality,” “novelty,” and possibly also “surprise,” which are commonly used in definitions of creativity. I define the metric of newness to mean that, in the opinion of a specific judge, the artifact in question has not previously

appeared in the domain.

Terms like “surprise,” “novelty,” and “originality” all seem to connote a degree of unexpectedness. This I define as being, in the experience of a specific judge, an idea that cannot be readily constructed from obvious extrapolations or combinations of ideas already in the domain.

The notion of “simplicity” is contained within the broader concepts of “elegance” and “practicality.” My functional definition for it within the context of creativity assessment is that it describes ideas that can be expressed compactly, with a minimum of caveats, conditions or qualifications.

“Validity” appears within concepts such as “correctness,” “effectiveness,” and “valuable,” which are often used to describe creative ideas. Specifically, I define validity to mean that the described idea produces correct results as defined by the constraints of the problem or the domain in which it is assessed.

Lastly, “applicability” is a quality that hides within “usable,” “productive,” and “effective,” and which I take as meaning that the idea in question can be applied readily. At first glance, applicability appears to be a combination of simplicity and validity, but an idea can be simple and valid without being applicable. For example, a solution to the problem of highway traffic collisions would be to equip each vehicle with a device that allows them to pass harmlessly through one another. This solution is both simple and valid, but it is not applicable.

If the various definitions of creativity can be constructed from terms that can themselves be constructed from these five component qualities, then it stands to reason that by measuring an artifact for any of these five qualities, we are in effect measuring some aspect or component of its creative merit. I must reiterate, though, that such assessments are subjective self-assessments. Recall that, from the judgement conundrum, p -creativity can only be assessed within the context of the artist’s experience and opinion, and that the assessment of one judge in no way effects the validity of another’s. So, to demonstrate p_a -creative facilitation, we are not looking for outputs that are generally agreed by a number of judges to be creative; we are looking for a process that is judged by a number of users to have produced p_a -creative results for them.

Apparently confounding the discussion, Kreitler and Casakin [57] tell us that there is no reliable correlation between self-assessed creativity and external expert assessment measured by Amabile’s CAT test. The apparent conflict between these two measures fades, however, when we realize that they are in fact measuring two distinct attributes: p_a - vs. h -creativity, respectively.

Bringing this together, I define the degree of creative merit exhibited by an artifact as an aggregate measure of its newness, unexpectedness, simplicity, validity and applicability, within some specific context of evaluation.

It should be noted, however, that this measurement can only be used to compare two specific ideas. To compare the relative power of different facilitative tools and contexts, we must employ a different measure, such as the notion of creative richness, which is discussed next.

3.2.3 Creative Richness

In general usage, the term “richness” tends to refer to a relative measure of the proportional presence, or “density,” of some quality, such as flavour or money. My preliminary hypothesis uses the term *creative richness* to denote some measurable attribute of a population of ideas. In this context, the term “density” can also be interpreted in several ways. Ideas can be collected densely in space, or densely in time. Whether a number of ideas are experienced concurrently in close spatial proximity, or serially, in rapid succession, the effect is approximately the same: each idea impinges upon the consciousness before the memory of the previous one has faded, thus facilitating associative cognitive comparison. Therefore, both spatial and temporal density seem relevant.

Nemeth and Ormiston suggest [81] that richness can be measured as a function of quantity, quality and diversity. I intend to follow their lead, but will explore several different measures of the quality dimension.

By creative richness, then, I mean a comparative measure between two or more populations of ideas, expressing the rate of occurrence of high quality candidates within those populations. The meaning of “high quality” is dependent upon which of the five attributes of creativity discussed in Section 3.2.2 have been identified as relevant. Depending upon the purpose of any particular experiment, creative richness might be measured as the relative proportion of simple ideas to complex ones, or their distinctness from one another, or even the speed with which the user can shift his attention among the candidates.

3.2.4 Users and Musical Expertise

The literature demonstrates a degree of muddiness when using terms “beginner” and “expert.” Are “beginners” lacking in experience with the software tool or with the artistic domain? Clearly, a user can be an expert composer, while also being a complete newcomer to computing or to a particular program. Conversely, many expert computer users will be entirely unschooled in music composition theory.

The distinction is a crucial one in discussions of facilitation where the target user audience is widely construed. Therefore, for clarity, I will avoid using terms like “beginner” and “expert” in favour of “novice” and “master,” by which I refer specifically to the degree of expertise in the relevant artistic domain. (In the case where one wishes to describe the degree of familiarity a user has with a particular tool, I would suggest using the terms “newbie” and “wiz,” which capture the notion in terms common to the culture of software development.)

3.2.5 Musical Sketches

If we are to facilitate early-stage creative ideation, then according to the cognitive context premise, we must consider the fact that users will have to be presented with candidate materials and tools that are appropriate to the associative mode of cognition active at that time.

Musical compositions, however, are experienced serially. Unlike images or sculptures, which can be absorbed — at least, superficially — in their entirety, music must be auditioned from start to finish in order to be understood and evaluated. This presents a challenge for musical ideation software tools. In order to facilitate the richness and simultaneity premises, users must be able to embrace a number of candidate pieces as nearly concurrently as possible. So in order to evaluate ideas rapidly enough to allow them to resonate with one another cognitively, they must be relatively brief.

In addition to brevity, another clue to a possible formulation for those fragments is given by the rules-as-content and structural representation premises, which tell us that our musical ideas might best be encoded as hierarchical networks of musical transformations acting on note sequence data. By emphasizing the basic relationships between elements of the composition, and offering little in the way of note-by-note manipulation tools, this encoding scheme should also help to keep attention on the broad, general principles of musical ideas, rather than on the specific notes resulting from them, thus helping to avoid the completion cueing problem. Specific choices for what is meant by such terms as “networks,” “musical transformations,” and “note sequence data” in this research will be developed more fully in Chapter 4.

3.2.6 Software Composition Tools

While musical performance and extemporization are important manifestations of creativity, I am not concerning myself with them. I specifically limit myself to the more deliberate, protracted process of composing musical scores.

Most software programs designed to support the composition and/or notation of musical scores focus on the user’s need to make fine-grained changes to the content. Consequently, every note must be presented for potential change, and every possible change to those notes must be supported by the tool set. This is an entirely appropriate orientation when the purpose of the tool is to either notate a composition that has already been developed to some degree of maturity, or to polish one that is nearly so, but as a number of musicians have told me anecdotally, early-stage composition is seldom done on a computer. Instead they tend to “play around” with ideas elsewhere, most often on their instrument of choice. It is only after some initial inspiration has come to them, and been developed to some degree, that they might then turn to software to help them with further development and refinement. The problem seems to be that most of the existing software tools are

just not suited to the kind of free-form thinking and exploration that the composer finds necessary for creative thought.

Novice musicians, on the other hand, do not yet have the musical knowledge necessary for competent original composition, and so their efforts tend toward either triviality or imitation, neither of which tends to be creatively satisfying. When they try to compose using notation software, the musical resolution achievable with such systems is so fine-grained, and so unconstrained, that just about any conceivable sequence of note events can be expressed. Without musical training to guide their choices, these users are far more likely to achieve cacophony than consonance.

One of the features of atomic editors is that the incremental variation produced by each successive edit differs only trivially from its immediate antecedent, and any substantive diversity encountered when using such tools in the course of developing an idea will likely be smeared out across a long sequence of incremental, localized changes. As Gabora tells us, however, a key feature of early-stage cognition is that it is widely associative, seeking to make connections across large leaps within the problem domain, and it is specifically desensitized to fine-grained differences between ideas.

When we look at other tools, such as sequencers, they too are essentially atomic in this regard. While changing a loop counter or instrument assignment may make large-scale changes to the specific audio signal generated, perceptually speaking, those changes still tend to be small, and would not be characterized by the average listener as constituting a new composition – rather, they would likely still be seen as minor variations on the previous candidate.

From this, I conclude that most musical score composition tools available today are essentially atomic in nature, and consequently, the incremental changes of content to which users are exposed during their use are creatively invisible for the purposes of associative, early-stage ideation. A differently constituted tool, however, that provided more dramatic changes between incremental steps might be able to provide such inspiration where atomic tools fail.

Atomic editors are also limited by virtue of the fact that they must present every detail of the artifact to the user. Consequently, they tend to be real-estate hungry, requiring a large allotment of screen resources to display their content in accessible form. Generally speaking, this makes them weak choices as a means of displaying a multitude of ideas, since, in order to exploit the richness premise, the ideas presented must be able to stimulate the consciousness of the artist more or less concurrently. By contrast, hierarchically expressed musical fragments lend themselves to compact, recursively inspectable presentation, which will permit a relatively large number of such candidates to be manipulated in a relatively small screen area.

3.2.7 Musical Rules

Rules of music composition come in many forms. There are rules of compulsion, such as “Blues guitar solos must resolve to the tonic;” and rules of prohibition, such as the one prohibiting the use of tri-tone harmony in Medieval ecclesiastical music. But such rules are unstable, applying usually to a limited context — either stylistic, cultural or even being composer-specific. They are rarely ironclad, even within their particular domain of application. I have been unable to find any rule of music that is truly universal, and indeed, I suspect that any such universal rule would simply stand as a challenge to composers, compelling them to search for a context in which the rule could be broken for beneficial effect.

For this reason, it seemed entirely inadvisable to build a creative composing environment in which any particular rule — either compulsive or prohibitive — was strictly enforced. But it is also clear that most musical compositions are bound by certain rules — the rules or logic of the composition itself. As researchers such as Huron [49] or Lerdahl and Jackendoff [64] suggest, musical compositions are inextricable from the patterns and structures from which they are constructed. These pattern and structure rules are not something applied *to* the music—in a very real sense, they *are* the music — and therefore, they should be encoded as part of the musical content, rather than held separate from it.

There are perhaps many ways to construe these rules of music, and no doubt, many different sorts of computer facilitation could be explored by simply choosing different ways to characterize what those rules are like and how they should be expressed, but there is one characterization that seems particularly attractive to me. Recall that Cope’s EMI system extracts recurring patterns from existing works — the so-called *signature* constructions that are common to a particular composer working in a particular form. Recall also that a common approach to composition cited by many composers is the notion of iterative experimentation and aggregation or construction of larger structures from smaller ones.

In this context, then, I am not concerning myself with abstract rules of music that might hold some formal theoretical or artistic legitimacy in and of themselves, but rather, I am concerned with the internal rules that govern the logic and construction of the specific work in which they are embedded. Rather than *rules of music*, I will call these *constructive transforms*.

Such transforms can either create simple musical passages out of nothing, or can take one or more fragments of music, along with some number of parametric controls, and transform those inputs into some new music, synthesized from the inputs.

As a simple example, consider a *loop* operator, which takes a single musical passage as input, along with a repetition count, and produces a longer passage of music by concatenating the input passage to itself as many times as are indicated by the counter. Although simple in conception, this transform is extremely common, used, for example, to construct the ubiquitous “verse, chorus,

verse, chorus” structures in modern pop music.

Another simple example is the *transpose* operator, which takes an input musical fragment and a pitch-offset parameter dictating how much the output copy should be raised or lowered in pitch. This could be used, for example, in the construction of counterpoint melodies to restate the original theme, raised a perfect fifth above its initial statement, or it could be used to create a duplicate parallel octave melody one octave above the original.

By encoding compositions as a network of these constructive transformations and their inputs, we can encode the resulting music of a composition indirectly, as the consequence of the rules used to define it. Furthermore, by declaring that these musical transforms be treated as extensible plug-in modules to the architecture, I can allow for almost any such rule to be added to the system, as our needs and understanding evolve.

3.2.8 Musical Competence

The preliminary hypothesis references the notion of *competence*, meaning musical competence. More specifically, it suggests that such musical competence will be measurably improved when the hypothesis is put to the test. This might lead one to believe that I will be employing some form of software evaluation of musical competence, but fortunately, that will not be necessary. Recall that one of the five criteria of creativity is the notion of *validity*, and in the case of musical compositions, I take this to mean that the candidates produced will have to conform to the broad tenets of what is valid (or competent) music, but since these rules are to be applied and assessed subjectively by the artist, we do not need to define an absolute definition for what is or is not musically competent. Both the functional definition and the assessment of its presence in the candidates is left as a responsibility of the assessor.

3.2.9 Variety of Ideas

The richness premise embeds a crucial assumption: that its juxtaposed materials are sufficiently diverse that they each present different insights and possibilities to the perceptions of the artist. It is through combining these differing elements that the creative process is fuelled. Exploring a set of identical or indistinguishably similar artifacts would not offer the same creative inspiration.

But like competence, variety need not be judged in the absolute by our tools. It too is a subjective attribute and must be left to the judgement of the artist, because only the artist himself can assess whether the candidates presented to him during the exploration were sufficiently diverse from one another to serve as indicators of distinct potential avenues he might explore, or whether they were so mutually inter-similar that they all sounded alike to his ear, and thus excited no diversity of choice.

3.3 Scope

The ultimate goal of this line of research is to one day present a robust set of software tools to composers of all backgrounds that will enable them to be more creative, but tackling that all in one step is far more work than can be done responsibly in a single doctoral thesis. In the course of my work, I have come to realize that before we can build such user interfaces, we will first need a much stronger foundation of facilitative theory than is currently available. We have to gain a handle on the problems of how best to express creative media in computational terms, and how to go about measuring and comparing creative outputs before we can then turn our attention to exploring user interfaces that exploit these foundations.

Paraphrasing from Greenberg and Buxton [41], we must get the right design — by which I mean selecting our foundation and approach — before we can proceed to getting the design right — meaning the specifics of any particular interface or tool set.

3.4 Formal Research Hypothesis

This thesis, then, will focus on the nearer-term goals of providing some of those necessary foundations. As suggested in the preceding discussions, the avenue I think will be most fruitful will be to focus on facilitating the relatively under-explored area of personal p_a -creativity during the process of early stage ideation.

I propose to do this by exploiting the structures of the musical content that are most relevant to that early, exploratory stage, rather than attending to the note-by-note details. Further, I propose doing so in a manner that allows ideas to be both generated and evaluated quickly, and that allows multiple, conceptually disparate ideas to be juxtaposed easily against one another.

My contention is that this structural sketching approach will expose the user to a creatively richer set of intermediate forms that will be more distinct from one another than would be the case with typical atomically oriented tools, but which will also still be strongly related to the material at hand, so that they will be seen as relevant ideas, and not as pointless distractions.

These goals are summed up in the following research hypothesis:

Hypothesis 3 (Formal) *The p_a -creative richness of musical sketches produced by composition software tools can be increased over that of pitch notation-based tools by representing musical compositions as constructive hierarchies of note-sequence transformations, thereby providing stronger facilitation of early stage personal creative ideation and better musical guidance for novice composers, without sacrificing expressive breadth.*

3.5 Experiment Plan

There are three distinct components to the experimental regimen: developing a platform for conducting the experiments; validating the utility of the developed platform; and challenging the hypothesis itself, using the platform. In this section I will now briefly describe the necessary experiments in each of those areas, but this overview is intended to be cursory, highlighting their relationship to the overall hypothesis rather than expounding on the minutiae of their execution. Those details will be provided and discussed in subsequent chapters.

3.5.1 Implementation

To conduct our experiments, we will need a way to encode and manipulate both structural and atomic representations of the music. For the purposes of experimental isolation, these tools must be shorn of as many distractions and extraneous influences as possible. In my survey of the existing tools and applications, I found none that seemed well suited to my needs, so I created my own, called *Wheelsong*.

In order to justify using the *Wheelsong* system as my testing substrate, I will have to demonstrate that it can effectively encode both constructural and atomic musical compositions; that such encodings can be reliably rendered into audio; that those encodings can be effectively modified; and that the resulting renderings logically correspond with the changes made.

The implementation of *Wheelsong* and the confirmations of its basic representation and manipulation fidelity will be discussed in Chapter 4.

3.5.2 Validation of Platform

Sufficiency of Representation Breadth

The claims implicit in the hypothesis are not simply that structural representations will yield more creative bebop jazz melodies, or hip-hop rhythm tracks. The claim is that structural representations will improve the creativity of music in general, construed much more broadly than any one culture or genre. To that end, I will have to demonstrate that the *Wheelsong* system is capable of representing music from a variety of genres and cultures.

Furthermore, while the stated intent is to facilitate early stage experimentation, and the discovery or construction of creative idea fragments, it is also important that the substrate not exhibit any arbitrary limits on complexity or length of the subject compositions, because in keeping with our emphasis on subjectivity, it is entirely up to the artist to define what he or she means by a fragmentary idea. In support of this, I will demonstrate that *Wheelsong* is capable of arbitrary precision, complexity and length in its compositions.

Discussion of the experimental compositions created to demonstrate these points will be provided in Chapter 5. Specifically, I will demonstrate that constructive hierarchies of note-sequence transformations can be used to encode music compositions in software; that such a system can be used to encode a wide range of musical cultures, genres and styles; and that such a system can be used to encode both complete compositions and fragmentary sketch compositions.

Sufficiency of Workflow Breadth

As cited in Section 2.1 and 2.4, there are several different basic approaches to the creative composing process. Some works are derived seemingly from whole cloth, while others are more obviously derived from existing works. If Wheelsong is to demonstrate more general applicability, it will be important to show that both approaches are supported. Examples of both creative paths will be shown in Chapter 5.

3.5.3 Examination of Hypothesis

User Completeness

The research hypothesis also stipulates that structural representation schemes should offer creative advantages to both novice and master composers. Example compositions created by both classes of user will be provided and discussed in Chapter 5, demonstrating that the Wheelsong architecture can permit experienced composers to create self-satisfyingly creative new composition ideas; and can also assist inexperienced composers to compose original, musically competent, and self-satisfyingly creative compositions.

Creative Richness

The hypothesis predicts that tools like Wheelsong will not only allow users to produce more creatively rich content than they would otherwise do with atomic-style tools, but also that doing so is inherent to the structural nature of the representation and toolset themselves.

To confirm this, a series of experiments will be conducted in which a panel of evaluators will be asked to judge the creative merits of a series of samples that were composed at random using either atomic or structural representation systems. Chapter 6 will describe these experiments and analyse the results.

3.6 The Judgement Conundrum Revisited

The judgement conundrum (see Section 2.5.2) creates a distinct complication for these assessments, in that it asserts that there is no way for an objective third party to validate the results. The

experiment can be repeated, but because the crucial factor being judged is both subjective and extremely sensitive to context, judgements made by a given subject at a given time cannot be used to substantiate or refute judgements made either by another subject or at another time — not even if the experiment is performed again with the same subjects.

This means that, while I can present outputs from the system as illustrations in this dissertation, they are poor evidence, because the opinions regarding those illustrations formed by myself or by the reader, after the fact, have little bearing on the psychological state of the creator at the time of creation. Such opinions are an evaluation of h-creativity or p_r -creativity, not of p_a -creativity, and so fall outside the scope of this investigation.

At the same time, I recognize that this will require the presentation of some evidence beyond just the qualitative assessments reported by the subjects themselves. To that end, I will be including example outputs from all the composition experiments, but with the understanding that these are to be taken as illustration, rather than as evidence of merit.

Now that I have laid out my hypothesis and outlined the steps needed to examine it, the following chapters will proceed to do so. In Chapter 4 I will describe the software platform developed to support examination of the hypothesis. Chapter 5 will examine that platform's ability to provide the support necessary to explore each of the targetted issues, and Chapter 6 will then describe three experiments conducted with that system to evaluate the hypothesis.

CHAPTER 4

WHEELSONG

Before I could begin investigating my hypothesis, I first needed to choose a platform— the workbench tool(s) that would be used. Requirements for the platform were somewhat conflicted. I needed a system that could represent both structural and atomic musical constructs with equal facility, which eliminated most notation-based systems; it had to be extremely broad in its definition of structural encodings, like a programming language, yet it had to be easy to understand and manipulate for non-technical users, which seemed to preclude programming languages; and it had to present users with access to the richly connected structures of the representation, likely through a robust graphical user interface (GUI), but at the same time, I could not allow it to introduce the distractions and complexities that come with the dense feature sets common to most mature, existing desktop tools.

Perhaps the most insidious risk in using existing tools lies in the potential for subtle cognition bias effects. Since I had already concluded that most existing applications focused much of their attention on facilitating late-stage polish and refinement activities, it seemed clear that they would also contain an inherent bias in their designs and workflows toward that working context, and therefore toward supporting users who were engaged in analytical-style cognition. Since the literature has not yet acknowledged the distinct roles played by these two modes of cognition in creative facilitation, the safest course of action seemed to be to assume that they all had some level of this unconscious bias, and so I elected to develop my own tools. Such purpose-built tools not only freed me from potential cognition bias, but they also offered the added advantage of having complete access to all of the source code, should my investigations dictate the need to change any aspect of their function.

Since little of what works for late-stage activities could be relied upon as a model for working in the early-stage context, I was left with a number of conspicuous unknowns. It was not known which specific musical transforms might be best suited to the kind of structural composing indicated in my hypothesis; it was not known what sort of user interface might provide the best access to the resulting tools and content representations; it was not known what mode of audio output would be most effective, nor what types of inspections and visualizations might be of greatest use to users. To build tools despite these unknowns, all I could be certain of was that I required an architecture

that was extremely flexible, in which compositions could be encoded and those encodings could be manipulated, but with everything beyond that left as fluid as possible, for later specification.

The most fluid and adaptable design for my purposes, it seemed, was the pipeline architecture, in which there is no central, authoritative program, but rather, the entire system could be built as a cooperating population of utilities that shared a common data format. This struck me as highly adaptable, which suited the fact that my research was going to be exploratory in nature. With a pipeline architecture, I was even free to change programming languages from one module to the next. Not only did this allow wide flexibility in the sense of being able to add arbitrary operator nodes or interaction tools at a later time, but by implementing the pipeline at the shell level, it allowed all of those later additions to be completely independent in their own internal architectures — and since it was not known at design time how the various notions of musical structure would best be manifested into software, this deep flexibility seemed crucial.

As was developed in the previous chapter, my intention was to look at music as a hierarchical structure in which small snatches of musical information were aggregated into successively larger structures. The suite of tools I developed to conduct this investigation, called *Wheelsong*, therefore took an entirely black-box view of structure at the architectural level, positing fragmentary musical data, originating in atomic leaf nodes, which then flowed upward from there, through successive “aggregating transformations,” until finally emerging from the root node as a single cohesive entity. But what went on in those aggregative transforms was left entirely unspecified. In this way, *Wheelsong* makes it possible to explore encodings of musical structure without ever specifying exactly what musical structure is, or is not, beyond the notion that it is some inter-dependant network of musical ideas and operations. As long as the contract implied by the file format of the data stream is respected, such an architecture seems ideally suited to supporting the kind of experimental exploration I am seeking to facilitate.

The *Wheelsong* suite of tools is therefore composed of many parts, but they can be grouped into five main categories, which will be discussed in this chapter. First, and of primary importance, is the data stream format itself, called *WSNG*. This format defines how musical note events are encoded by the *Wheelsong* system and is used to represent all of the musical fragments used in a *Wheelsong* composition at every level of granularity. The *WSNG* format will be discussed in Section 4.1. Next are the various operators that produce or transform *WSNG* musical fragments. Examples of these are described in Section 4.2. Section 4.3 will then describe the *WSNT* file format, which is used to specify the various connections between and among those operator nodes and *WSNG* fragments for a composition. With the encoding scheme described, users also need a mechanism for creating and manipulating them, which will be the subject of Section 4.4. Finally, visualization and auditioning tools that allow the user to examine all or part of a composition in a variety of ways will be described in 4.5.

4.1 WSNG File Format

Music is commonly described (e.g. [66, 94, 64]) as ranging over four independent dimensions: pitch, time, amplitude and timbre, forming what I will call *music space*. A composition can be described, generally speaking, as a collection of musical note events, and each of these can be represented as a curve in music space. Most compositions, then, can be represented as a collection of finite curves in the 4-dimensions of music space.

This is, admittedly, an over-simplification. Timbre in particular probably requires more than four independent parameters of its own to adequately specify the complete range of its acoustic subtleties. But if we limit our discussion to the types of information normally found in common staff score notation, we see that timbre receives little more encoding than the choice of instrument and perhaps a few of the special modes or colours it can achieve. So, since the intent is to represent musical compositions and not their performances (which would certainly require much more detail), by employing these four conceptual dimensions, we are approximately as tightly constrained as we would have been in using common notation, and hence should have more than sufficient expressive range for our purposes.

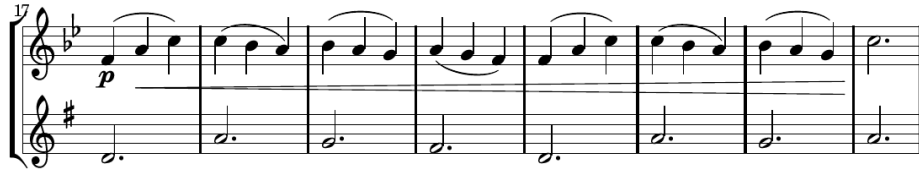


Figure 4.1: Successive decompositions of melodic contours

These musical curves can be arbitrarily complex, rising and falling in any of the three non-temporal dimensions independently. Such complexity is dealt with in common notation by breaking it down into smaller components that exhibit greater local stability, such as those shown in the 3-note passages of Figure 4.1, which are in turn, part of an even longer curve spanning the first seven measures.

In essence, this notation decomposes the curve in piecewise fashion, expressing the larger curve as a smoothly connected series of sequential sub-events. From mathematics, we know that any curve can be approximated with arbitrary precision by a piecewise linear approximation of appropriate resolution, so it seems reasonable that we can do the same with curves in music space. In order to represent arbitrary music-space curves, we need to be able to specify note events as finite linear segments in which the start and end points of each segment are clearly located in the four musical dimensions. An encoding system that allows us to encode such linear musical note event segments would seem to be at least powerful enough to approximate arbitrary compositions that can be expressed in common notation, which in turn should be sufficient for our purposes.

For each note event, we need to be able to specify a pitch, time, volume and voice for the start and the end of the event. This could be represented by the eight-tuple $(p_0, p_1, t_0, t_1, v_0, v_1, x_0, x_1)$, and this would work. In practice however, it was found that expressing time as an absolute start time and a relative duration was more manageable — especially when transcoding from common notation, in which durations are the most ready-to-hand values available. So, the actual encoding used is: $(p_0, p_1, t_0, \Delta t, v_0, v_1, x_0, x_1)$.

These are encoded as ASCII text files according to the form: $Pp_0Sp_1Tt_0D\Delta tVv_0Cv_1Xx_0Yx_1$, with white space delimiters signifying the end of one note event and the beginning of the next. This encoding is referred to as the WSNM format.

These tags are assigned mnemonically as **P**itch and **S**lur, **T**ime and **D**uration, **V**olume and **C**rescendo, and **voX**. In the absence of a good mnemonic for the second timbre value, the X of vox is simply paired with Y.

Before presenting an example encoding, however, we must still address the issue of which units are to be used in quantifying the various components, and as we shall see in the next section, this turns out to be somewhat problematic.

4.1.1 Unitlessness

A brief survey of musicology will turn up a cornucopia of different units for measuring pitch. One can express pitches absolutely, using acoustic frequency, measured in Hertz, or relatively, using the logarithmic scale of cents. They can be stated indirectly, either in terms of the tuning system (e.g. scientific notation's $C4$ or Helmholtz's C') or even in scale-relative terms, such as the tonic, supertonic, mediant, subdominant, or tonic, second, third, fourth systems often applied to diatonic scales.

Similarly, time can be referenced, at the very least, in terms of beats or measures or seconds. It can also be referenced indirectly, in terms of larger structures, such as the verses of a popular song, or the movements of a concerto. Dynamics, or volume, can be expressed quantitatively in terms, such as decibels or sones, or relative to other sections or passages of the composition, by using terms like pianissimo or fort e.

So with all these diverse interpretations to choose from for measuring the dimensional attributes of music, it becomes problematic to choose just one to serve as the foundation of all music, insofar as WSNM files will attempt to encode all music. In some contexts, it might be best to express and manipulate pitches in terms of their acoustic frequencies, whereas, at other times, working with them as indirect references to the degrees of a particular scale might be more advantageous. It all depends on what the composition is, and more specifically, which transformation operators are being applied.

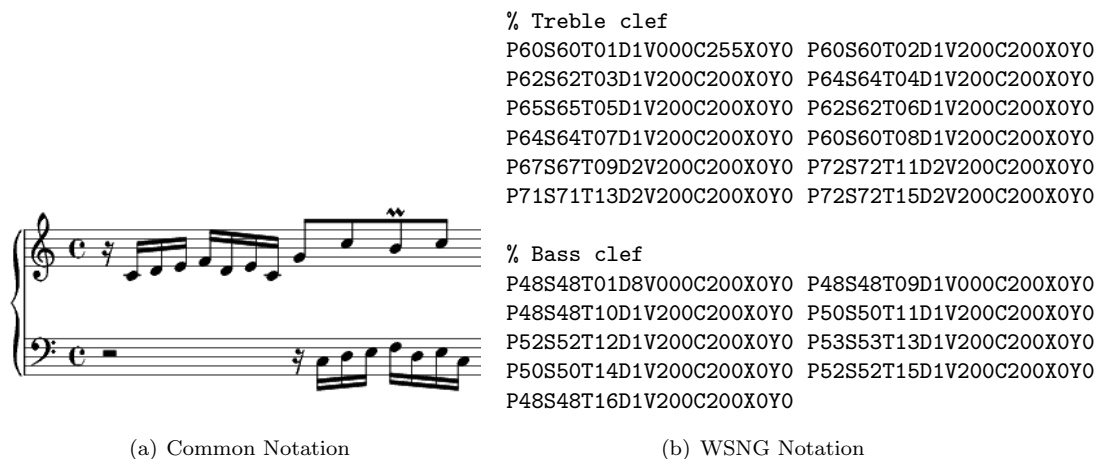
For this reason, WSNF files are declared to be inherently unitless. They are simply floating-point, scalar values. Any assertion of a sense of units to those scalars is left to the individual transformations, and there is no expectation that any two of them will share their expectations.

In practice, then, the first note from the bottom staff in Figure 4.1 could be encoded as P62S62T0D3V80C80X0Y0 if we were to infer MIDI pitch values for the pitch and slur components, timing values expressed in terms of a quarter-note having a duration of 1.0, volume measured as a percentage of the maximum, and vox values representing the staff number.

Alternately, we could equally justify encoding it as P293.665S293.665T17D1V60C60X9Y9. In this case, we've assumed that pitches are expressed as frequency values, time is counted in musical bars or measures, the volume is expressed in phons¹, and vox components are expressed as MIDI instrument numbers — in this case, selecting the glockenspiel. Clearly, there are many, many ways that any particular composition or fragment can be expressed in WSNF format, and the selection of appropriate encoding bases is one that can best be made by the person doing the encoding.

In practice, I have found that MIDI-based values are reasonable default assumptions, as they are readily understood by many people already, but this should not be seen as simply converting MIDI files into ASCII format, since there are so many other ways to quantify the components, which each give rise to different structural and creative contexts.

This fluidity of representation also offers the WSNF data streams great flexibility, allowing the output of any one transform to be readily ingested by another. What the results of interpreting the output milliseconds of one operator as the incoming beat numbers of another is anybody's guess, but this is exactly the kind of malleability lauded by the loose typing premise, and as we shall see later, it is a powerful feature.



(a) Common Notation

(b) WSNF Notation

Figure 4.2: First measure of Bach's *Invention No. 1 in C Major*

¹A perceived-loudness scale that compensates for differing sensitivity in the human ear depending upon the frequency of the sound

Figure 4.2 shows two versions of Bach’s *Invention №1 in C Major*: the first in common notation and the second in fully qualified WSNG. (Note that the wavy symbol above the musical notation is called a trill, which indicates that the note should be ornamented with a specific rapid succession of neighbouring notes. To make these examples understandable to those who are not familiar with trills, I have chosen to ignore it, although it should be obvious that encoding a trill would be a trivial matter of a few additional note events.) For consistency, and unless otherwise noted, all examples of WSNG files used in this document will follow the convention of using MIDI pitch values as the implied pitch units and beats as the implied time units. Volumes are expressed as a one byte, positive integer, with 0 meaning a silent note event, and 255 as the maximum value. Voice assignments are simple integers, and are usually allowed to retain their default value of 0, which simply means that all note events are to be voiced with the same instrument or timbral settings.

4.1.2 Convenience

As Figure 4.2(b) makes clear, however, “human-readable” does not necessarily mean “easy to read,” and while it might be perfectly acceptable for code-based parsers to swallow these lengthy music fragments, they are not particularly accessible at a casual glance.

This becomes particularly problematic in contexts where the user wishes to simply “type in some notes” — especially for those with any degree of music training. Having to type in the start and end values for every note, in each of the four dimensions, quickly becomes tedious. To facilitate this style of interaction, a number of short-hand conveniences have been implemented in the parser.

| Component | Default | 1 st Note |
|-----------|--------------|----------------------|
| Pitch | N/A | |
| Slur | Pitch | |
| Time | Prev End | 0.0 |
| Durat’n | Prev Durat’n | 1.0 |
| Volume | Prev Vol. | 255.0 |
| Cresc. | Volume | |
| Vox | Prev Vox | 0 |
| Y | Vox | |

Table 4.1: Table of default values for WSNG components

First, there is the notion of defaults. As shown in Table 4.1 all components of a note event have a default value, with the exception of P. Every note event must declare a starting pitch, but beyond that, nothing else need be declared. The default ending pitch is, naturally, the same as the starting pitch. Similarly, the default behaviour for volume and vox is to remain unchanged from the start of any given note event to its conclusion. This way, if a particular note event is stable—the most common circumstance—it can be encoded more compactly by leaving out the end values altogether.

In common musical practice, there is also a lot of note-to-note coherence. A quarter note is most often followed by another quarter note, and the volume of one note is most likely to be the same as its neighbours. Exploiting this fact frequently allows us to leave out the starting values as well. When unspecified, the Duration, Volume and Vox components all repeat whatever value was declared by the immediately previous note event in the stream, which cascades to the next previous note and so on, until a predecessor note is found with an explicit specification. In the case of the first note in the stream, for which there is no immediate predecessor, numeric constants ($T = 0$, $D = 1.0$, $V = 255$ and $X = 0$) are defined in the parser to serve as default values. So in most cases, a note stream can be specified very compactly, with component values only being explicitly noted when they change.

This deference to the previous value does not work well, however, when applied to start time. When we leave out explicit declaration for the start time of a note event, the most useful default to infer is not the same time value at which the previous note started, but instead, the time at which it ended, because it is much more common for notes to follow each other in time than it is for them to sound simultaneously. By employing these default behaviours, most continuous melodic passages can be encoded quickly by listing just the pitches for each, along with their duration or volume when those values change.

```

% Treble clef
P60T1D1V0 P60V200 P62 P64   P65 P62 P64 P60
P67D2 P72 P71 P72

% Bass clef
P48T1D8V0 P48D1V0 P48V200 P50 P52
P53 P50 P52 P48

```

Figure 4.3: Bach's *Invention №1 in C Major* in compact WSNG form

Figure 4.3 shows the same passage as that of Figure 4.2(b), but in much more compact, readable form. Note also that, while melodic rests could be encoded implicitly, by simply advancing the T component to a later time, it is often useful in practice to explicitly encode them, which has been done here by assigning note events with a volume of zero.

One last convenience implemented in the parser that becomes particularly useful when transcribing common notation scores into WSNG format is the support for fractional values. Instead of having to repeatedly enter six- or seven-digit floating point values for triplets, or having to remember the floating-point representation for the various sixteenth- and thirty-second-note fractions, the parser will accept floats in the form A/B. So, a three-sixteenths middle-C note event can be encoded quickly as P60D3/16.

These conventions and conveniences greatly simplify both the creation and inspection of WSNG fragments intended for human consumption, and so are presented here to assist the reader in reading

the examples cited later in this document.

On its own, however, the WSNG format is simply another way to encode musical notation. It does not become particularly useful as a platform for exploring musical structure until we add a mechanism for building longer fragments from smaller ones, which I will cover in the next section.

4.2 Transformation and Generation Filters

Now that we can specify musical fragments, we need to be able to make use of them. The second component of Wheelsong’s pipeline architecture is the set of modules that allow us to do so, variously referred to in this document as filters, generators, transforms or operators, depending on context.

Taking our cue from the structural representation premise, these modules should be capable of executing simple modifications to musical data, such as repeating it, or transposing it to a higher or lower pitch. They should also be capable of combining smaller pieces into larger pieces, through such mechanisms as concatenation or parallelization. In this section we will explore several examples to illustrate the various types of operations that can be performed and the role they play in the bigger Wheelsong picture.

Perhaps the simplest and most common operation employed by composers as they construct larger structures out of smaller elements is the notion of repetition. Figure 4.4 depicts just such a filter in action.²

```
> echo "P60 P64 P67" | wsngrepeat -c 2  
  
P60S60T0D1V255C255X0Y0 P64S64T1D1V255C255X0Y0 P67S67T2D1V255C255X0Y0  
P60S60T3D1V255C255X0Y0 P64S64T4D1V255C255X0Y0 P67S67T5D1V255C255X0Y0
```

Figure 4.4: Repeated arpeggio

This statement (following the “>” prompt) applies the `wsngrepeat` operator to a three-note input — C, E and G expressed in MIDI terms, with pitch values of 60, 64 and 67. These notes are declared sequentially (via the default timing assumptions of the WSNG format, which sequences them one after the other). The arguments to the filter (`-c 2`) cause that input sequence to be rendered twice in succession to the output, shown in the bottom of the figure.

Note that despite the absence of any explicit timing information in the compact input expression, the output has correctly interpreted the notes as being sequential, and the second statement of the arpeggio has been shifted appropriately to start-time values of 3, 4 and 5, succeeding the first statement which had start-time values of 0, 1 and 2. Note also that, despite the use of compact form for the original input, the transform expanded the output into fully qualified form.

²These examples were constructed in Linux, using the BASH shell, but can be replicated in most modern command shell environments.


```

> echo "P60 P64 P67" | wsngretrograde

P67S67T0D1V255C255X0Y0 P64S64T1D1V255C255X0Y0
P60S60T2D1V255C255X0Y0

(a) Retrograde

> echo "P60 P64 P67" | wsnstranspose -n 7

P67S60T0D1V255C255X0Y0 P71S64T1D1V255C255X0Y0
P74S67T2D1V255C255X0Y0

(b) Raised

> echo "P60 P64 P67" | wsnsgchanmult -T 0.3 -D 0.3

P60S60T0D0.3V255C255X0Y0 P64S64T0.3D0.3V255C255X0Y0
P67S67T0.6D0.3V255C255X0Y0

(c) Faster

```

Figure 4.5: More simple arpeggio transforms

Three more simple transforms are shown in Figure 4.5, one showing the arpeggio input in retrograde (played backward), another showing it raised in pitch by a perfect fifth (7 semi-tones), resulting in a GMaj, instead of the input CMaj chord, and the last showing the arpeggio played faster, which is achieved by multiplying the time values (both start-time and duration) by a constant fraction of 0.3.

Operations such as those shown above are common transformations to find within musical compositions and are especially apparent in counterpoint, which is the foundation of most modern western music, but there are, of course, many other transformations possible. In my own Wheelsong compositions, the five operators of transposition, concatenation, repetition, palette lookup, and random fragment generation account for more than 75% of the nodes, although I suspect other users' preferences will vary significantly. For a complete list of all the Wheelsong transform operations implemented to date, see Appendix A.

Each of the transforms shown above is a unary operator, taking a single stream as input and transforming it to a modified output. As mentioned in the previous chapter, however, the point of Wheelsong is that complex musical structures can be constructed from humble, fragmentary roots, but to achieve that, we need a slightly more complex class of operators, capable of combining multiple input WSNG fragments.

```

> wsnsgsequence -i arpeggio.wsng -i arpeggio-raise.wsng

P60S60T0D1V255C255X0Y0 P64S64T1D1V255C255X0Y0 P67S67T2D1V255C255X0Y0
P60S60T3D1V255C255X0Y0 P64S64T4D1V255C255X0Y0 P67S67T5D1V255C255X0Y0
P67S60T6D1V255C255X0Y0 P71S64T7D1V255C255X0Y0 P74S67T8D1V255C255X0Y0

```

Figure 4.6: Repeated arpeggio

The example shown in Figure 4.6 shows two input fragments (the outputs from Figures 4.4 and 4.5(b)) being concatenated together by the `wsgnsequence` operator.

Once all the obvious aggregation functions have been implemented, music offers a diverse assortment of more inventive ways to combine streams — either to transform the substance of the inputs or to expand upon them, enlarging them in potentially startling ways. For example, Figure 4.7(a) shows a transformation that treats one input as a palette of pitches and treats the other as a stream whose pitch values are array lookups into the palette. Or consider Figure 4.7(b) which repeats its arpeggio input once for each controlling note encountered in the second stream, raising the pitch of the repeated arpeggio by the pitch of the controlling note. These transformations do not seem to be drawn from classical music theory, but their power to transform musical fragments in inspiring ways has been demonstrated in several of the compositions that will be explored in Chapter 5.

```
> wsgnfromscale -s arpeggio-rev.wsgn -i arpeggio.wsgn

P412S60T0D1V255C255X0Y0 P436S64T1D1V255C255X0Y0 P463S67T2D1V255C255X0Y0
P412S60T3D1V255C255X0Y0 P436S64T4D1V255C255X0Y0 P463S67T5D1V255C255X0Y0

(a) Rescaled

> wsgnloft -C arpeggio.wsgn -i arpeggio.wsgn

P120S60T0D1V255C255X0Y0 P124S64T1D1V255C255X0Y0 P127S67T2D1V255C255X0Y0
P120S60T3D1V255C255X0Y0 P124S64T4D1V255C255X0Y0 P127S67T5D1V255C255X0Y0
P124S60T6D1V255C255X0Y0 P128S64T7D1V255C255X0Y0 P131S67T8D1V255C255X0Y0
P124S60T9D1V255C255X0Y0 P128S64T10D1V255C255X0Y0 P131S67T11D1V255C255X0Y0
P127S60T12D1V255C255X0Y0 P131S64T13D1V255C255X0Y0 P134S67T14D1V255C255X0Y0
P127S60T15D1V255C255X0Y0 P131S64T16D1V255C255X0Y0 P134S67T17D1V255C255X0Y0
P120S60T18D1V255C255X0Y0 P124S64T19D1V255C255X0Y0 P127S67T20D1V255C255X0Y0
P120S60T21D1V255C255X0Y0 P124S64T22D1V255C255X0Y0 P127S67T23D1V255C255X0Y0
P124S60T24D1V255C255X0Y0 P128S64T25D1V255C255X0Y0 P131S67T26D1V255C255X0Y0
P124S60T27D1V255C255X0Y0 P128S64T28D1V255C255X0Y0 P131S67T29D1V255C255X0Y0
P127S60T30D1V255C255X0Y0 P131S64T31D1V255C255X0Y0 P134S67T32D1V255C255X0Y0
P127S60T33D1V255C255X0Y0 P131S64T34D1V255C255X0Y0 P134S67T35D1V255C255X0Y0

(b) Lofted
```

Figure 4.7: More complex arpeggio transforms

Finally, while these various transforms are capable of achieving marvellous effects, so far they have all been shown operating on explicitly coded input fragments. This is often a desirable way to construct compositions, but it is also possible to work with synthetic fragments. Figure 4.8 demonstrates two simple examples of this. Figure 4.8(a) shows the creation of a simple scale, with the `-m` parameter selecting from various major, minor, or blues scales, or even foreign scales such as the pelog and slendro scales of gamelan music. Figure 4.8(b) demonstrates the application of Perlin noise [87] (a coherent noise function) to create a random note stream that exhibits some surprisingly melodic features. Like the other transforms discussed above, these too have been used

to great effect in some of the compositions that will be explored in upcoming chapters.

```
> wsngscale -m 7
P49 P51 P54 P55 P56 P58 P61
(a) Scale
```

```
> wsngrandom -e 8 -r 0.2 -g 0.2 -p 60 -P 65
P62.5T0D1X0V0 P62.5946T1D1X0 P62.424T2D1X0
P62.317T3D1X0V0 P62.2285T4D1X0 P61.976T5D1X0
P61.9771T6D1X0V0 P62.3136T7D1X0
(b) Random
```

Figure 4.8: Generative filters

But having a way to express individual musical fragments, and a suite of operations that can be performed upon any one of them are insufficient, in themselves, for representing complete compositions. For that, we also need a way to specify a collection of such fragments and a sequence of operations to perform upon them. This last component of the encoding scheme will be the subject of the next section.

4.3 WSNT File Format

Each of the examples from the previous section contained a single operation applied to a single fragment, which is simple and easy to understand, but very few worthwhile compositions can be expressed so simply. Most are comprised of dozens or even hundreds of aggregative and transformative steps that must all be executed in a specific sequence. Representing that sequence is the last step of the Wheelsong encoding scheme, and is the province of the Wheelsong Network (WSNT) file format that I will now describe.

In principle, all Wheelsong operations can be executed on the command line, as they were in the previous examples. Through the use of pipes and tee shunts, quite elaborate combinations can even be composed and executed as a single, lengthy shell statement. Doing so, however, would be an exercise in frustration for any but the simplest (and therefore least interesting) musical structures. Requiring lengthy manual input of command sequences would be a hideous impediment to flow, and would be a never-ending cause of transcription error. Clearly, in order to exploit the flexibility benefits of the pipeline architecture efficiently, there needs to be a way to express the operation sequence in a static, repeatable format.

The examples shown back in Figure 4.6 and 4.7 offered our first look at that problem. Note that instead of declaring the inputs directly on the command line, inputs to binary (or n -ary, where $n > 1$) transforms cannot be expressed with inline pipes, and had to be written out to data files, which are then referenced by name in the command-line construction.

In general, only very trivial transformations can be constructed on the command line. Significant structures only become evident when the operators are combined in longer chains, and even simple chains involve more typing than is convenient. In principle, this could be accomplished by using any

one of the many scripting languages that abound, but doing so would be problematic. Scripting languages are more than simple batch command processors. They are typically programming languages in their own right, and using them would conflict with my design intent in two ways. First, they would require the user to learn something about programming, but worse, using them would run the risk of embedding some of the structural logic of the composition in the programming constructs, rather than in the connectivity graph itself and the transforms employed by it.

To resolve this problem, a pipeline graph rendering file format was created, called WSNT, specifically to represent Wheelsong networks. In this next example, Figure 4.9(b) shows a real-world WSNT file in which two WSNG fragments are combined to create a sketch of the the first measure of Bach’s *Invention N°1 in C Major*. For comparison, Figure 4.9(a) shows the common notation for the same passage.

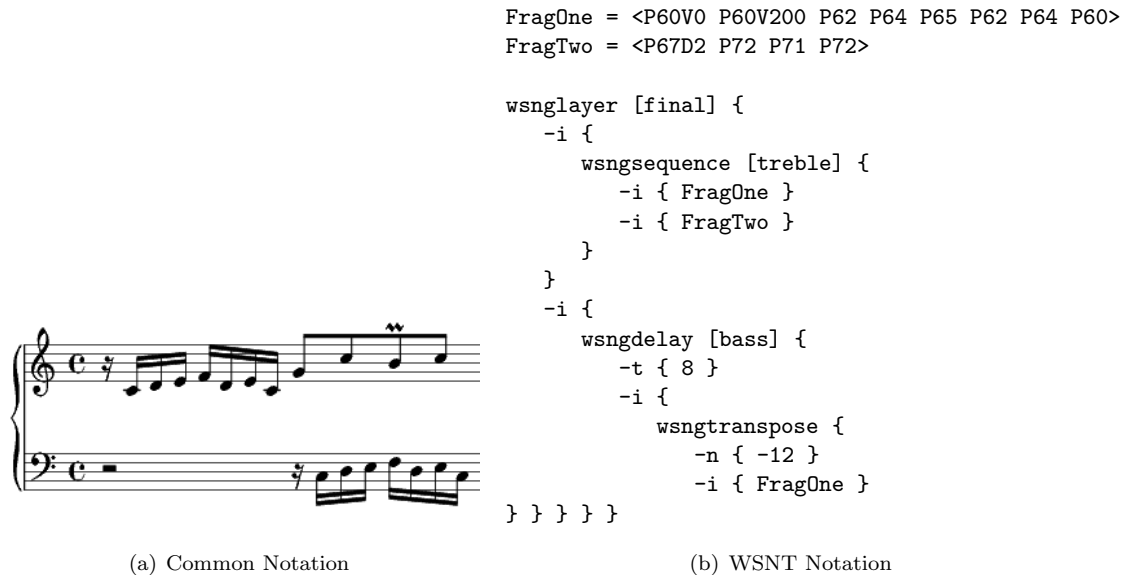


Figure 4.9: First measure of *Invention N°1 in C Major*, expressed structurally

The first two lines of the example show the declaration of explicit WSNG literal fragments, FragOne and FragTwo. Below that follows a hierarchically nested tree of transform operations and their arguments. Rendering of this file is a simple process of bottom-up traversal, assembling shell statements, executing them and storing their outputs in temporary caching files so that they can be used as inputs to higher level structures. The tokens presented in square brackets are labels for the output of the associated transform, which allows them to be referenced from other subtrees of the structure, just as though they had been declared as static literals at the top of the file.

It is possible to compose by writing WSNT files in a text editor, but such a process is unlikely to be attractive to non-technical users. While more elaborate interfaces have been sketched, it is worth noting that almost all of the example compositions discussed in later chapters were created using the text editor interface.

This concludes the discussion of Wheelsong’s encoding scheme, but there are still two more components to the overall system. Users must be able to see/hear the output of their composition structures, and they must be able to create and manipulate them. Those facilities will be the subject of the last two sections of this chapter.

4.4 User Interfaces

If users are to manipulate Wheelsong’s musical structures, they must have some control interface through which to do so, but as discussed in the previous chapter, it seems folly to compound the uncertainties of an experimental encoding scheme by building an experimental user interface on top of it. Doing so would only make it more difficult to determine which measured effects arose from which of the two experimental elements of the system. On the other hand, we cannot evaluate a tool if there is no way to manipulate it and its content.

The content of Wheelsong compositions, of course, consist of WSNG fragments and trees of transform operators that modify them, and there are any number of existing approaches to how such trees might be manipulated, such as expandable tree controls, 2D network editors, and so on. In the long term, exploring which approaches best deliver the affordances of constructural representations to which sorts of users may well be an important component of my research. For now, however, I wish to focus on measuring the benefits of the encoding scheme itself, so I elected to rely on a text editor—in particular, the Vim editor, which is commonly used in Linux consoles. While Vim is a very powerful and flexible editor, working with WSNT encodings as text files is a slow and awkward process, by comparison to the GUI techniques mentioned above.

WSNT files for any but the most simplistic compositions are much larger than will fit comfortably in a single display screen, so manipulating them becomes fraught with navigational issues. Furthermore, there is no way for a text editor to permit local probing, auditioning sub-structures within a work in progress to hear what they sound like, so this must be done by creating external files, or by restructuring the existing file to play only the sub-structure of current interest. All of this combines to make for a slow, awkward process that most certainly impedes flow.

Figure 4.10 shows the *Clockhouse* encoding displayed in the Vim editor. Less than 3% of the entire file is visible, which illustrates something of the navigational inefficiency of editing compositions this way. By choosing Vim for my interface, I felt it would be safe to conclude that any measured effect on creativity was much more likely to be contributed by the encoding than by this interface. Only after the encoding scheme’s affordances are reasonably well understood will I then feel it appropriate to begin considering more effective approaches to exposing those affordances to the users.

```

GenPan4Scale = <P7 P9 P11 P12 P13>
% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
wsngfromscale [GenPan1] {
  -i {
    wsngrandom [ii] { -c -S {13} -s {0} -e {8} -p {0} -P {4} -x {0} }
  }
  -s { GenPan1Scale }
}
wsngfromscale [GenPan2] {
  -i {
    wsngrandom [iii] { -c -S {23} -s {0} -e {8} -p {0} -P {5} -x {0} }
  }
  -s { GenPan2Scale }
}
wsngfromscale [GenPan3] {
  -i {
    wsngrandom [iiii] { -c -S {31} -s {0} -e {8} -p {0} -P {4} -x {0} }
  }
  -s { GenPan3Scale }
}
wsngfromscale [GenPan4] {

```

Figure 4.10: *Clockhouse* encoding in the Vim editor

4.5 Visualization and Audition Filters

There is more to working with Wheelsong than simply assembling constructive musical scaffolds. Users also need to be able to hear their work and visualize its structures, both as a whole and in various component parts. By finding different ways to present the data to the user, new and compelling insights might thereby be fostered.

All of the examples discussed to this point have had a built-in assumption that they were to be played by a MIDI sequencer, but what new insights might be gained if they were instead rendered via CSound? Or played on a theramin? Would displaying the data as common notation be of interest to any users? Would new patterns and structures become evident if the data was displayed as a piano-roll? Or if it was used as parameters to describe an abstract visual animation? The visualization premise suggests that any of these might conceivably be good ideas, in some cases, for some users. In recognition of that, a few experimental modules of this type have been developed, although they have not yet been used extensively in any composing processes, nor has their facilitative power been assessed.

The `wsngtocsound` module will export WSNG data into a CSound file, which is better suited to exploring melodies with non-quantized pitches or alternate tuning systems. Other audio generation systems or score encoding file formats such as MusicXML or Lilypond are also possible, but with the richness of issues that can be explored through the default MIDI conduit, there has yet been no need to pursue any of these other formats.

Figure 4.11 shows the same opening bar of Bach's *Invention №1 in C Major* that has been shown previously, although this time rendered with some of the Wheelsong visualization tools. Users who are familiar with MIDI editing systems may recognize the piano roll diagram of Figure 4.11(a), produced by the `wsngpianoroll` filter. This format mimics the idea of the punched rolls of

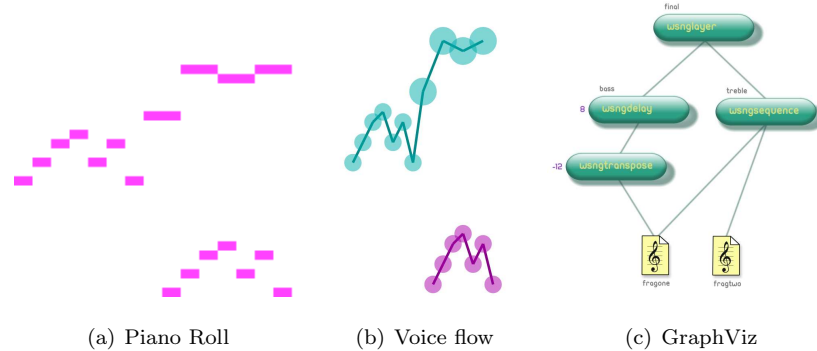


Figure 4.11: Visualizing the first bar of Bach’s *Invention №1 in C Major*

paper that are used to program player pianos. The bars of the diagram indicate notes to be played, where the vertical axis encodes pitch and the horizontal axis describes both start- and end-times.

The diagram of Figure 4.11(b) is what I call a “voice-flow”. Voice-flow diagrams were inspired by Stephen Malinowski’s unpublished experiments in visualizing MIDI data [70]. Though similar in layout to the piano roll, voice-flow diagrams (created using the `wsngvoiceflow` filter) encode more information than a piano-roll, replacing rectangular bars with circles, the radius of which represents the duration of the note, and its opacity represents the volume. Furthermore, each distinct melody or voice within the work is encoded with a different colour, and optionally linked in sequence by a similarly coloured “flow line.” I find that these diagrams are highly intuitive, and even the most untrained of users seem able to follow the progress of a musical piece easily, without any prompting or training required. It also serves to place the various contributing voices of a work into visual juxtaposition, while making many of their patterns instantly accessible to human visual processing.

While these visualizations present the final atomic representations of the work, another filter, called `wsnt2dot`, employs the powerful GraphViz engine to create a schematic representation of the WSNT structure, such as the one shown in Figure 4.11(c). So even in the absence of a graphical user interface, users can still readily explore several different visual interpretations of the work.

These tools, however, are simply preliminary explorations to demonstrate the feasibility of visualization filters. As future research begins to explore graphical user interface options in earnest, a number of different presentation schemes will have to be explored, and doing so through the creation of additional rendering filters seems a practical way to begin that process.

CHAPTER 5

EXPLORATORY COMPOSITIONS

In Section 3.5.2, I articulated a set of features that would be needed in a research platform to support the types of experimentation I expected to conduct. Specifically, those required features are: the ability to represent both atomic and structural compositions; the ability to encode musical sketch structures in a variety of genres, and cultural traditions; the ability to embrace a variety of compositional processes; the ability to present a rich assortment of creatively diverse ideas; the ability to represent content compactly and with leveragable power; and to offer utility to both novice and expert composers alike.

In this chapter I will probe the limits of Wheelsong, and the degree to which it delivers on these requirements, by using it to conduct a series of composition experiments. The experiments themselves were performed in a somewhat loosely structured fashion, guided by the need to investigate the success of supporting the stated goals, but also influenced by artistic curiosity and the exploitation of opportunities and ideas that arose *in situ*. To simplify the discussion, I have chosen to organize this chapter into conceptual sections, although it should be noted that this does not always conform with the experimental chronology.

Furthermore, as mentioned at the outset, if we are to succeed in peering into p_a -creative facilitation, we must engage, at least in part, in the subjective impressions of the users. So, in addition to discussing the fulfilment of the stated requirements, I also present relevant subjective insights arising from this first-hand usage experience.

Space precludes me from discussing all of the various works that have been composed to date, so I have chosen instead to discuss a strategic subset of them that speaks most directly to the issues at hand. Throughout this discussion, the reader is encouraged to consult the audio examples on the companion CDs, as indicated, and the full encodings listed in Appendix B.

In addition to assessing the success of my stated design goals, I will also extract observations from this first round of experimental use regarding some unexpected features and behaviours that became evident, which will help guide further development and the design of subsequent experiments.

Lastly, if it is true that early stage creativity will require different sorts of tools from those that are employed at later stages, then it would seem worthwhile to identify indicators that might be used to monitor the user's progress along this spectrum. Such an indicator, if accurate enough,

would allow the interface to be adapted automatically to the user’s changing needs. Consequently, based on my subjective assessment of the composing experience, I will introduce a number of simple candidate metrics that may have some predictive value, although that value will have to be assessed in subsequent research.

5.1 The Explorations

Each time an encoding is created, there are a number of decisions to be made about how to express the source material structurally. These decisions give rise to a variety of possible interpretations and diverging points for further exploration. The results discussed in this section have been organized into groups of related experiments, stemming from a common source, each of which addresses a different required feature. This organization also helps to illustrate the fluidity with which one compositional idea flows into the next.

5.1.1 Historical Western Music

Given that Wheelsong is intended to serve, in part, as a composition sketching tool, I thought it best to start by demonstrating its utility in this context. Rather than demonstrate a sketch of an original composition, however, which might raise questions regarding Wheelsong’s applicability to “real” compositions, I’ve elected to begin with a sketch of a real work from an acknowledged master composer.

Unfortunately, even with this stipulation, it is still fairly common practice in some branches of music software research to employ self-servingly simplistic examples, such as *Mary Had A Little Lamb* or *Twinkle Twinkle Little Star*. I felt that it would be a better test to build my sketch against the full score of the subject work, and not against a toy or “easy piano” reduction of the original.

To this end, I chose the example of J. S. Bach’s *Invention №1 in C Major*¹, which is neither trivial nor self-servingly simplistic. Furthermore, the piece offers structures that are quite evident in the score on a purely visual basis (see Figure 5.1), and should be understandable to readers who have no music theory training.

The first step in this experiment was to identify patterns from which the piece could be reconstructed. Not being an expert in composition or counterpoint theory, I opted to use naive, visual patterns, rather than attempting to identify the application of musicological rules of the genre. Also note that, for this exercise, trills² were completely ignored.

Figure 5.1 shows the identification of ten such patterns occurring within the first six measures, eight of which are stated in the first measure, with the remaining two appearing later, in the fifth and

¹Refer to “*Invention №1 in C Major*” in the *Wheelsfragments* folder of the companion CD.

²The wavy line that appears above some notes indicating that it should be embellished by rapidly alternating it with a neighbouring note



Figure 5.1: Naive visual patterns in Bach’s *Invention №1 in C Major*

sixth measures. Declarations of melodic patterns are indicated with a coloured rectangular over-bracket or under-bracket, and interval patterns are identified with a coloured arc. The rest of the score is annotated with coloured blocks indicating which passages can be reconstructed by applying an affine transformation to one of the declared patterns. This example was created by examining the score visually for naive patterns of recurring note contours, resulting in transformations that are limited to temporal translation, pitch translation (transposition), pitch reflection (inversion), temporal reflection (retrograde) and temporal scaling.

The literature does not appear to hold any advice for how we might gauge the quality of a musical sketch, but one metric that seems useful is the notion of coverage. The proportion of non-rest note events contained in the original score, to the number of notes accounted for by the pattern sketch, gives us an approximate measure of the sketch’s coverage of the original. In this case, the original score contains 456 distinct note events, 67 of which are not accounted for by the sketch, which leaves an approximate coverage of 85%. This suggests that the sketch probably conveys much of the essence of the original. The reader can judge this first hand by comparing the renderings of *Invention Sketch* and *Invention №1 in C Major* on the *Raw Wheeldata* companion CD.

It should also be noted that the notes produced by the sketch do not always match exactly. In some few cases, the sketch output differs from the original with respect to accidentals (sharps or flats). This is entirely in keeping with the practice of counterpoint composition, whereby the composer would construct a transformation outline not unlike the sketch, and then proceed to

correct various sour notes, bringing them in line with the conventions. Since my goal is to produce a sketch of the work, and not a finished product, this should not be viewed as a problem. Moreover, this is precisely the trade-off to be made between sketching systems and production systems, in that fine-grained precision is sacrificed for exploratory power.

```

1  % The identified patterns
2  green = <P22D1 P23 P24 P25>
3  dblue = <P23D1 P24 P22>
4  orange = <P26D2 P29 P28 P29>
5  red = <P22D2 P15>
6  brown = <P27D2 P23>
7  dgrey = <P28D0.5 P29 P30D1>
8  lgrey = <P29D3 P30D1>
9
10 % some patterns are derived
11 wsgsequence [purple] {
12   -i {green}
13   -i {dblue}}
14 wsgsequence [lblue] {
15   -i {purple}
16   -i {orange}}
17 wsgselect [pink] {
18   -e {3} -i {green}}
19
20 % some useful fragments
21 pause16 = <P1D1V-0>
22 down5 = <P0D1 P4>
23 drop = <P12 P10 P8 P6>
24 ...
25
26 % some useful aggregations
27 wsgsequence [lbluepat] {
28   -i {pause16}
29   -i {lblue}}
30 wsgsequence [purplepat] {
31   -i {pause16}
32   -i {purple}}
33 wsgsequence [purplepati] {
34   -i wsginvert { -i {purplepat} }}
35 ...
36
37 % assembling the treble patterns
38 wsgsequence [treble] {
39   % bar 1-2
40   -i { wsgloft {
41     -C {down5} -i {lbluepat} }}
42
43   % bar 3-4
44   -i { wsgloft [bar2t] {
45     -C {drop} -i {purplepati} }}
46   ...
47 }
48
49 % assembling the bass patterns
50 wsgsequence [bass] {
51
52   % bar 1-2
53   -i { pause2 }
54   -i { wsgtranspose {
55     -n {-7} -i {purplepat}}}
56   -i { wsgtranspose {
57     -n {-3} -i {red}}}
58   -i { pause4 }
59   -i { wsgtranspose {
60     -n {-3} -i {purplepat}}}
61   ...
62 }
63
64 wsgchanmult [play] {
65   -t {0.6} % Reduce total duration
66   -i { wsgfromscale { % resolve note palette
67     -s { wsgscale { -m {0}} % major scale
68     -i { wsgsequence { % combine parts
69       -s
70       -i { treble }
71       -i { bass }
72     } } } } }

```

Figure 5.2: WSNT encoding fragments from the *Invention Sketch*

Figure 5.2 shows details from the sketch encoding. Lines 2 – 8 show declaration of the patterns that are literal note sequences, and lines 11 – 18 show the purple, light blue and pink patterns as derivatives of one or more of those literal patterns. In lines 27 – 34, some of those patterns are built up into additional aggregate patterns of convenience, such as the `lbluepat` and `purplepat`, which prepend rest notes to the patterns, and `purplepati`, which creates a melodic inversion of `purplepat`. Each of these patterns are used regularly throughout the remainder of the sketch.

The section from line 38 to 47 illustrates how those patterns are sequenced to create the first few measures of the treble portion of the work, and lines 50 to 62 do the same for the bass component, although both of these sections are elided for the sake of brevity. (For complete details, see WSNT encoding in Appendix B.4.)

Finally, the section from line 64 to the end depicts three global operations: innermost, the treble

and bass sections are laid over top of one another with the `wsngsequence` filter; above that, the pitch values, which until this point have been expressed in palettized notation, are resolved into an actual scale by the use of the `wsngfromscale` operator; and uppermost, the tempo of the piece is accelerated using `wsngchanmult`.

Encoding pitches indirectly, through a palettized lookup table, is a technique I have found useful for structuring musical fragments in a way that better facilitates subsequent deviation. In the sketch, the `wsngfromscale` command (line 66) generates a seven-note major scale and uses that as a palette from which to resolve pitch references from the patterns employed throughout the sketch. A palettized pitch value of 1 will select the first element of the palette³, a pitch of 2 will select the second, and so on. To accommodate more than one octave, pitch values are indexed modulus the size of the palette, so an index of 8 will access the first note of a seven-pitch palette, but in the next higher octave.

The *Invention Sketch*⁴ demonstrates that WSNT files can indeed encode sketch representations of musical compositions. The next several examples will explore this sketch further, examining various ways in which this simple structure can be powerfully transformed.

The Variations

An examination of Figure 5.1 will reveal that, of the ten different patterns identified, the vast majority of the *Invention Sketch* is constructed using only half of them: the dark blue, light blue, purple, green and pink patterns. Furthermore, those five patterns are themselves dependent upon only three primary, literal patterns: the green, dark blue and orange note sequences. This raises some tantalizing questions. How much of the *Invention Sketch*'s musicality is defined by those three literal fragments, and how much of it resides in the constructural network used to aggregate them into their final structure? Does the structure of the piece admit only one fundamental composition, or can that same structure produce multiple, distinct-sounding outputs? If multiple, what are the limits of apparent stylistic diversity of the outputs that can be produced? To investigate these questions, I began a series of exploratory modifications.

Figure 5.3 shows a portion of the *Invention Sketch* encoding on the left. The right side shows the small set of changes that were applied to that encoding to produce the variant called the *Tokyo Invention*.⁵ Specifically, each of the three primary patterns has been replaced with melodic fragments of my own arbitrary specification^①, although I have been careful to preserve the total duration of each pattern. In addition, the tempo has been slowed down a bit, the note events have been slightly shortened to produce a more staccato effect, and the major scale palette from which

³To keep things familiar for non-technical users, palettes are 1-based arrays, not 0-based.

⁴Refer to “*Invention Sketch*” in the *Wheelsongs* folder of the companion CD.

⁵Refer to “*Tokyo Invention*.” in the *Wheelsongs* folder of the companion CD.

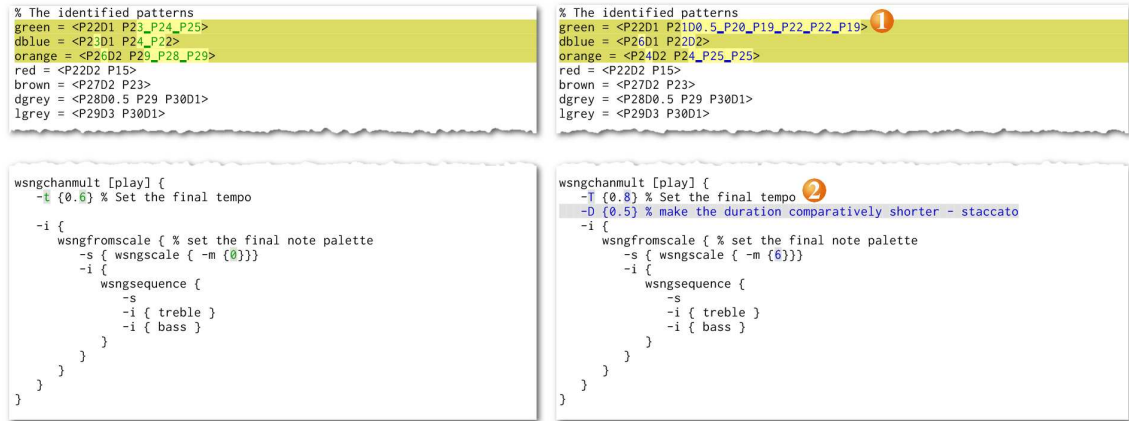


Figure 5.3: Coding changes to produce the *Tokyo Invention*

the pitch indices are resolved has been replaced with an Indonesian slendro scale², to give the piece a more eastern feel.

The result is a very pleasing and modern sounding composition with the expected slightly oriental feel. Like the *Invention Sketch*, *Tokyo Invention* demonstrates thematic development and even invokes a similar call-and-response structure, but it seems very unlikely that anyone encountering it without knowing its provenance would detect the influence. In my judgement, the important structural rules subconsciously embedded in the piece by Bach have been successfully repurposed to produce a new composition without requiring the composer to employ any understanding of those principles in doing so.

A second variation, called the *Bach Alley Shuffle*⁶, excerpted in Figure 5.4, employs different changes to produce an entirely different result. In addition to changing the primary literal patterns¹ and the palette⁵ (this time to a blues scale) several structural changes are also applied. The descending repetitions of the purple pattern in the original have been inverted to an ascending progression by changing the specification of pattern **drop**³ and several of the aggregate patterns from the *Invention Sketch* have been modified². For example, the **pink**, **doublegreen** and **doubledblue** patterns have been inverted from their original specification, while the construction of the **purple** pattern has reversed the order in which the **green** and **dblue** patterns are assembled. The only other change made was a slight increase in the overall playing time⁴.

In character, *Bach Alley Shuffle* is quite different from both the *Invention Sketch* and *Tokyo Invention*, conveying a very bluesy feel that also suggests a sense of big-city nightlife. Note also that in both examples, the duration of each pattern was preserved after the alterations. This is because, in the case of this particular series of encodings, synchronization of the treble and bass parts was managed implicitly, and relies on the sequence orders and durations of corresponding

⁶Refer to “*Bach Alley Shuffle*” in the *Wheelsongs* folder of the companion CD.

```

% The identified patterns
green = <P22D1_P23_P24_P25>
dblue = <P21D1_P24_P22>
orange = <P26D2_P25_P28_P29>
red = <P22D2_P15>
brown = <P27D2_P23>
dgrey = <P28D0.5_P29_P30D1>
lgrey = <P29D3_P30D1>

wsngsequence [purple] { -i (green) -i (dblue) }
wsngsequence [lblue] { -i (purple) -i (orange) }
wsngselect [pink] { -s (3) -i (green) }
wsngchanmult [doublegreen] { -t (2) -i (green) }
wsngchanmult [doubledblue] { -t (2) -i (dblue) }
wsnginvert [green1] {-i (green)}

wsngchanmult [doublegreen1] { -t (2) -i (green1) }

% The identified patterns
green = <P29D0.5_P22_P21D0.5_P28D1_P19D0.5_P22D1>
dblue = <P26D1_P22D2>
orange = <P24D2_P24_P25_P25>
red = <P22D2_P15>
brown = <P27D2_P23>
dgrey = <P28D0.5_P29_P30D1>
lgrey = <P29D3_P30D1>

wsnginvert [green1] {-i (green)}
wsngretrograde [dbluer] {-i (dblue)}
wsnginvert [dblue1] {-i (dblue)}
wsnginvert [dbluer1] {-i (dbluer)}
wsngsequence [purple] {-i (dbluer1) -i (green1) }
wsngsequence [lblue] {-i (purple) -i (orange) }
wsngselect [pink] {-s (3) -i (green1) }
wsngchanmult [doublegreen] { -t (2) -i (green1) }
wsngchanmult [doubledblue] { -t (2) -i (dblue1) }
wsngchanmult [doublegreen1] { -t (2) -i (green1) }

lowdrop11 = <P6_P4_P2_P0>
greendrop = <P7_P5_P3>
drop = <P12_P10_P8_P6>

wsngsequence [lbluepat] {-i (pause16) -i (lblue)}
wsngsequence [lbluepat1] {-i (pause16) -i (wsnginvert (-i (lblue)))}
wsngsequence [purplepat] {-i (pause16) -i (purple)}
wsngsequence [purplepat1] {-i (pause16) -i (wsnginvert (-i (purple) )))}
wsngretrograde [dbluer] {-i (dblue)}
wsnginvert [dblue1] {-i (dblue)}
wsnginvert [dbluer1] {-i (dbluer)}
wsngsequence [pausepurple] {-i (pause2) -i (purplepat)}
wsnginvert [pausepurple1] {-i (pausepurple)}
wsngtranspose [climb2] {-n (46) -i (climb)}
wsngchanmult [greendouble] { -t (2) -i (green) }

lowdrop11 = <P6_P4_P2_P0>
greendrop = <P7_P5_P3>
drop = <P6_P8_P10_P12>

wsngsequence [lbluepat] {-i (pause16) -i (lblue)}
wsngsequence [lbluepat1] {-i (pause16) -i (wsnginvert (-i (lblue)))}
wsngsequence [purplepat] {-i (pause16) -i (purple)}
wsngsequence [purplepat1] {-i (pause16) -i (wsnginvert (-i (purple) )))}

wsngsequence [pausepurple] {-i (pause2) -i (purplepat)}
wsnginvert [pausepurple1] {-i (pausepurple)}
wsngtranspose [climb2] {-n (46) -i (climb)}
wsngchanmult [greendouble] { -t (2) -i (green) }

wsngchanmult [play] {
-t (0.6) % Set the final tempo
-i {
wsngfromscale { % set the final note palette
-s { wsngscale (-m (8)) }
-i {
wsngsequence {
-s
-i { treble }
-i { bass }
}
}
}
}

wsngchanmult [play] {
-t (0.7) % Set the final tempo
-i {
wsngfromscale { % set the final note palette
-s { wsngscale (-m (8)) }
-i {
wsngsequence {
-s
-i { treble }
-i { bass }
}
}
}
}

```

Figure 5.4: Coding changes to produce the *Bach Alley Shuffle*

sections being identical. If these are modified unequally, the two parts fall rapidly out of step with each other and the result, while intriguing at times, is generally less pleasant. It seems likely that such structural decisions heavily influence which types of subsequent alterations will produce the most satisfying results.

A third variation of the *Invention №1 in C Major* produces a piece called *Melancholy*,⁷ and its modifications are shown in Figure 5.5. Unlike the previous two pieces, *Melancholy* is not particularly pleasing over its entire length, but the first four bars seem to evoke a degree of sadness and of waiting for change, which inspired the title. In creating this variation, the first three bars seemed promising, but the fourth was uninspired. By making a simple substitution¹ (replacing the final iteration of the purple pattern in measure 4 with the dgrey and brown patterns) the short passage was brought to a pleasing closure. While previous examples have demonstrated purpose-driven modifications to parametric elements such as tempo and pitch palette, this example demonstrates that the WSNT system supports purpose-driven structural modifications as well.

The fourth and final variation of *Invention №1 in C Major* in this exploration is one called *Morning Traffic*.⁸ After observing the great mood shift achievable by simply changing the pitch

⁷Refer to “*Melancholy*,” in the *Wheelfragments* folder of the companion CD.

⁸Refer to “*Morning Traffic*,” in the *Wheelsongs* folder of the companion CD.

```

% The Identified patterns
green = <P2201_P23_P24_P25>
dblue = <P23D1_P24_P22>
orange = <P26D2_P29_P28_P29>
red = <P2202_P15>
brown = <P2702_P23>
dgrey = <P2800.5_P29_P30D1>
lgrey = <P29D3_P30D1>

wsngsequence [purple] { -i {green} -i {dblue}}
wsngsequence [lblue] { -i {purple} -i {orange}}
wsngselect [pink] { ~e (3) -i {green}}
wsngchannmult [doublegreen] { -t (2) -i {green}}
wsngchannmult [doubledblue] { -t (2) -i {dblue}}
wsnginvert [greeni] {-i {green}}
wsngchannmult [doublegreeni] { -t (2) -i {greeni}}

% The Identified patterns
green = <P2200.5_P2200.5_P2201_P26D2>
dblue = <P24D2_P22D1>
orange = <P24D1_P24_P24D2_P25D2_P24D2>
red = <P2202_P26D1_P22>
brown = <P2702_P23>
dgrey = <P2800.5_P29_P30D1>
lgrey = <P29D3_P30D1>

wsngsequence [purple] { -i {green} -i {dblue}}
wsngsequence [lblue] { -i {purple} -i {orange}}
wsngselect [pink] { ~e (3) -i {green}}
wsngchannmult [doublegreen] { -t (2) -i {green}}
wsngchannmult [doubledblue] { -t (2) -i {dblue}}
wsnginvert [greeni] {-i {green}}
wsngchannmult [doublegreeni] { -t (2) -i {greeni}}

lowclimb9 = <P5_P6>
lowdrop11 = <P6_P4_P2_P0>
greendrop = <P7_P5_P2>
drop = <P12_P10_P8_P6>

wsngsequence [lbluepat] {-i {pause16} -i {lblue}}
wsngsequence [lbluepati] {-i {pause16} -i {wsnginvert {-i {lblue}}}}
wsngsequence [purplepat] {-i {pause16} -i {purple}}
wsngsequence [purplepati] {-i {pause16} -i {wsnginvert {-i {purple}}}}
wsngretrograde [dbluer] {-i {dblue}}
wsnginvert [dbluei] {-i {dblue}}
wsnginvert [dblueri] {-i {dbluer}}

wsngsequence [pausepurple] {-i {pause2} -i {purplepat}}
wsnginvert [pausepurplei] {-i {pausepurple}}
wsngtranspose [climb9] { -n (46) -i {climb7}}
wsngchannmult [greendouble] { -t (2) -i {green}}

lowclimb9 = <P5_P6>
lowdrop11 = <P6_P4_P2_P0>
greendrop = <P7_P5_P2>
drop = <P6_P8_P10>

wsngsequence [lbluepat] {-i {pause16} -i {lblue}}
wsngsequence [lbluepati] {-i {pause16} -i {wsnginvert {-i {lblue}}}}
wsngsequence [purplepat] {-i {pause16} -i {purple}}
wsngsequence [purplepati] {-i {pause16} -i {wsnginvert {-i {purple}}}}
wsngretrograde [dbluer] {-i {dblue}}
wsnginvert [dbluei] {-i {dblue}}
wsnginvert [dblueri] {-i {dbluer}}
wsngsequence [purplepatr] {-i {pause16} -i {wsngretrograde {-i {purple}}}}
wsngsequence [pausepurple] {-i {pause2} -i {purplepat}}
wsnginvert [pausepurplei] {-i {pausepurple}}
wsngtranspose [climb9] { -n (46) -i {climb7}}
wsngchannmult [greendouble] { -t (2) -i {green}}
wsngchannmult [dgreyi2] { -t (2) -i {wsnginvert {-i {dgrey}}}}

% bar 3-4
-i { wsngloft [bar2t] { -C {drop} -i {purplepati} }}

% bar 5
-i { brown }
-i { lgrey }
-i { wsngtranspose { -n (5) -i {purplepati} }}

% bar 3-4
-i { wsngloft [bar2t] { -C {drop} -i {purplepati} }}
-i { wsngtranspose { -n (3) -i {dgreyi2} }}
-i { wsngtranspose { -n (3) -i {brown} }}

% bar 5
-i { orange }
-i { wsngtranspose { -n (5) -i {purplepati} }}

wsngchannmult [play] {
-t {0.6} % Set the final tempo
-i {
wsngfromscale { % set the final note palette
s { wsngscale { -m (0) }}
-i {
wsngsequence {
s
-i { treble }
-i { bass }
} } } }
} ) ) ) }

wsngchannmult [play] {
-t {1.0} % Set the final tempo
-i {
wsngfromscale { % set the final note palette
s { wsngscale { -m (1) }}
-i {
wsngsequence {
s
-i { treble }
-i { bass }
} } } }
} ) ) ) }

```

Figure 5.5: Coding changes to produce *Melancholy*

palette, I began to wonder what the effect would be of introducing a more extreme palette. So instead of using a pre-defined scale from `wsngscale`, I drew from the prelude movement of Bach's *Cello Suite No. 1 in G Major* and used the melody from the first four bars of that work as the palette. This turned out to be the simplest modification of all the variations on *Invention No. 1 in C Major* that I have tried to date and at the same time is both the most stylistically divergent result and the most personally satisfying. *Morning Traffic* has been likened by one knowledgeable commentator to the modern minimalist work of Steve Reich. The fact that this can be achieved by such a simple-seeming intercombination of one Baroque work with another very clearly illustrates the transformative power of working with music structurally.

It should be stressed that, contrary to the examples shown here, it is not proposed that users of *Wheelsong* should be manipulating musical constructions through their textual representations. These experiments are aimed at probing the abilities and affordances of the representation scheme itself, and so must necessarily be manipulated through these crude, expert-only methods until the merits of constructural design have been soundly established. At that time, a more appropriate interface can be developed.

These examples demonstrate clearly that multiple outputs can be achieved from a single source,

and that all of those results can reflect quite different styles. They also serve to show that, in some cases, simplistic sketch inputs do not yield simplistic or sketchy outputs. *Morning Traffic*, for example, is very close to a finished composition.

5.1.2 Popular Western Music

With encouraging results based on Baroque pieces, the next logical step was to explore a more modern source. For this, I chose a piece that is simple in construction (so that the voices and motifs are relatively easy to work with and to discuss) but that also seems to be timeless, which suggests that it cuts across a number of genre labels and is perhaps therefore more generally representative.

Mbube Sketch

This piece is a sketch of the African traditional song *Mbube*, made popular in North America by Solomon Linda under the title *The Lion Sleeps Tonight*. Where the Bach piece and its variants all had two simultaneous voices, *Mbube* has three, making it a bit more complicated. The important elements of the WSNT encoding are shown in Figure 5.6, along with the corresponding elements of the transformed piece, entitled *Danger Baby*.⁹



Figure 5.6: Coding changes to produce *Danger Baby*

The first two lines are components of the familiar bass motif, usually referred to as the *wim-o-weh* part, which underpins the entire song. In the transformation, a simple change is made by shifting the rest from the fourth note event to the second¹. The fragment representing the descant-like counter-melody is modified² and then the result is inverted³ along with the lyric melody, which is also retrograded. Lastly, the key is swapped from major to a minor mode⁴ and the tempo is accelerated⁵ slightly.

⁹Refer to “*Danger Baby*.” in the *Wheelfragments* folder of the companion CD.

These changes result in a piece altogether different in feel from the original, with a more urgent rhythm and a pessimistic tone, putting one in mind of a cautionary tale or warning, hence the title.

Blues

In this experiment, we will probe Wheelsong's extensibility and its ability to sketch harmonic progressions.

Since the previous compositions were all based on sketches of existing works, I next wanted to explore Wheelsong's ability to sketch new works from a blank slate. For this experiment, I chose the blues form as one that is immediately familiar to most western listeners, and which conforms to a relatively simple harmonic progression format. Consequently, the composition process began with encoding the chord progression and accompanying rhythms.

```
roots = <P60D2 P67 P65 P60>
shapes = <P0 P0 P0 P2>

wsngharmonize [play] {
-p {shapes}
-i {roots}
}
```

Figure 5.7: Simple example of `wsngharmonize`

This marks the first attempt to work directly with chords. One tool for doing so is the `wsngharmonize` filter, shown in a simple construction in Figure 5.7. In keeping with the loose typing premise, `wsngharmonize` takes two input melody fragments and recombines them to create chord progression specifications. The first input stream describes the root notes and durations of the chords in the progression. Chord forms for each root are then specified by a second WSNM stream, where the pitch values of that second stream are interpreted as index values, selecting one of nine chord shapes: Major, minor, Major 7th, minor 7th, dominant 7th, augmented, diminished, augmented 7th and diminished 7th. These are not intended as exhaustive. They are simply nine relatively common chords used by rhythm guitar players. A more thorough treatment of harmony specification is still needed.

With the rhythmic and harmonic pattern established, the piece still sounded incomplete, but rather than construct a melody manually, I wanted to see whether there might be a quick and efficient way to simply sketch some broad details of the sort of melody I wanted, and let the system create something suitable to serve as a stand in.

Some researchers in synthetic composition algorithms (e.g. Biles [8], Thom [111]) have explored the prospect of generating synthetic melodies for blues and jazz compositions, employing advanced AI algorithms for the job. I wondered if it might be possible to mimic such results with an easy-

```

1  ProgressionRoots = <P1D4 P1 P1 P1      41  } } }
2                                P4 P4 P1 P1      42
3                                P5 P4 P1 P5>      43
4  ProgressionChords = <POD4 P0 P0 P4      44  wsngrtranspose [bass] {
5                                P0 P0 P0 P4      45    -n { 12 }
6                                PO P0 PO P4>      46    -i {
7                                                47      wsngrfromscale [skeleton] {
8  wsngrswing [swungline] {      48        -s { wsngrscale { -m {0} } }
9    -S { 1.5 }      49          -i { swungline }
10   -S { 0.5 }      50      }
11   -i {      51    }
12     wsngrinterpolate {      52  }
13     -n { 1.0 }      53
14     -g { 0.5 }      54  wsngrsequence [intro] {
15     -i { ProgressionRoots }      55    -s
16   }      56    -i { chords }
17 }      57    -i { bass }
18 }      58  }
19      59
20  wsngrharmonize [chords] {      60  wsngrrepeat [accomp] {
21    -p { ProgressionChords }      61    -c {4}
22    -i { skeleton }      62    -i {
23  }      63      wsngrsequence {
24      64        -s
25  wsngrtranspose [melody] {      65        -i { chords }
26    -n { 36 }      66        -i { bass }
27    -i {      67      }
28      wsngrfromscale { % 8 = blues scale      68    }
29      -s { wsngrscale { -m {8} } }      69  }
30      -i {      70
31        wsngrandom {      71  wsngrsequence [play] {
32          -c      72    -i { intro }
33          -S { 23 }      73    -i {
34          -r { 0.1 }      74      wsngrsequence {
35          -p { 1 }      75        -s
36          -P { 14 }      76        -i { accomp }
37          -e { 144 }      77        -i { melody }
38          -x { 1 }      78      }
39        }      79    }
40      }      80  }

```

Figure 5.8: Perlin's Blues

to-use melody construction tool that could be employed by both master and novice composers alike.

Musical melodies are generally regarded as being monophonic, and a monophonic pitch sequence is more readily described as musical when its contour is locally stable, with relatively few large leaps in pitch [42, 25]. Superficially, this description seemed similar to a description of coherent noise functions, such as the one introduced by Ken Perlin [87], which has been exploited extensively in computer graphics for simulating natural textures and structures.

Inspired by this successful approach to simulating natural phenomena, it seemed reasonable to wonder if coherent noise might be applied to generating similarly convincing melodies.

Figure 5.9 shows the crucial parameters to the coherent noise melody generator, called `wsngrandom`. Using them, the user can specify the duration and several aspects of melodic stability, as well as a random number seed to control the repeatability of the output. If one generated melody is not quite right, a new candidate employing the same parametric constraints can be produced by simply

| Flag | Value | Description |
|------|---------|--|
| -S | numeric | Seed value for the pseudo-random number generator |
| -e | numeric | Time value for end of generated melody |
| -d | numeric | Default duration of note events |
| -g | numeric | Likelihood duration will change at each note event |
| -r | numeric | Likelihood a given note event is a rest |

Figure 5.9: Important arguments for `wsnrandom`

changing the seed value.

The resulting composition is called *Perlin’s Blues*¹⁰ and is surprisingly satisfying — especially given the simplicity with which it was specified. The tonality of the generated melody is certain to be at least approximately suited to the harmonic context, since the melody is expressed indirectly, via the pentatonic blues scale, which is known to harmonize well with the I-IV-V harmonic context of the 12-bar blues progression. The resulting melody meanders a bit more than a human-crafted melody would likely do, but this is because one long melody was generated to overlay the entire twelve measures. Typically, an extemporizing human artist would play shorter phrases, repeating and varying them over time, but for the purposes of providing a “place-holder” melody sketch, this longer piece works quite well.

Creating more structured melodies from smaller Perlin melodies will be discussed in the next experiment.

5.1.3 Non-western Traditions

For a non-western example, I turned to the Asian tradition of gamelan. A gamelan is an orchestra composed predominantly of percussion instruments, similar to xylophones, cymbals and gongs. As well as having a different orchestration, gamelan music employs a different scale and tuning system from western music, and uses what is called a “recursive” rhythmic structure, in that the voices are divided into several groups, each of which plays at twice the tempo of the previous group. Each part is relatively simple, to facilitate gamelan’s tradition of amateur players, but in the aggregate, a number of complex effects emerge.

A complicating factor of working with gamelan is that there is no real tradition of notating the parts, since they are traditionally taught by example. This is further complicated by the unfamiliar tuning and scale system, which means that the common notation scheme does not really apply. Fortunately, musicologists have invented various notation schemes and Sorrell uses these to provide transcriptions of several works [108], which I was able to use as the basis for a WSNT encoding.

¹⁰Refer to “*Perlin’s Blues*” in the *Wheelfragments* folder of the companion CD.

Ladrang Wilujeng

The subject piece is called (*Ladrang Wilujeng*¹¹) and is from the Indonesian sub-genre of gamelan. In my encoding, I have replicated parts for the genpan, genbar, gambang, bonpan, bonbar, sarbar, peking, kendang and gong instruments. I have chosen to exclude the rebab voice, as it is a fretless string instrument that plays highly unstable pitches similar to a theremin, and such wandering pitches do not reproduce well in MIDI, which currently serves as my rendering engine.

Due to the number of parts and the length of the composition, the encoding of *Ladrang Wilujeng* is extremely long (see Appendix B.11), and not well suited to excerpting here as an illustration. Instead, consider the voice-flow diagram, shown in Figure 5.10, which depicts approximately half of the first 32-beat colotomic cycle.

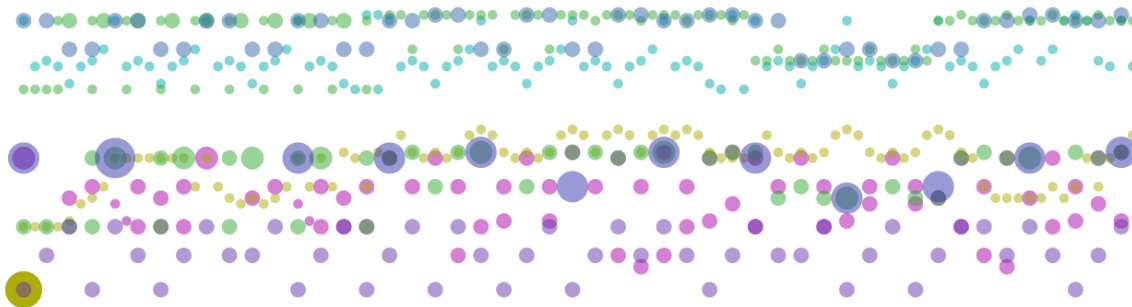


Figure 5.10: Voice flow diagram for *Ladrang Wilujeng* fragment

Comparing gamelan performances is difficult, because these pieces are best likened to jazz standards in that every performer interprets them in their own way and each performance tends to be a re-interpretation rather than a recitation [108]. Worse, there is no standardization of tuning in the gamelan world, so every orchestra plays each note at a slightly different pitch as well, and no two gamelans have identical tunings.

I have played the encoding for three people who are familiar with the style—one Indonesian colleague and two composers who have explored the form. Each of them reports that my encoding sounds like authentic gamelan music, although none of them were familiar enough with *Ladrang Wilujeng* to attest to its fidelity. I take this, at the very least, as corroboration of the claim that Wheelsong can encode a sketch of a composition from a non-western musical tradition.

Sorrell’s encoding notates the melodies with indirect palette references, where the designation 1 3 5 6 referred to the tonic, 3rd, 5th and 6th degrees of the scale. So naturally, the encoding for the WSNT followed suit, expressing the melodies as indirect literal note fragments, and then running the final composition through a palette lookup to assign those scale references to actual

¹¹Refer to “*Ladrang Wilujeng*” in the *Wheelfragments* folder of the companion CD.

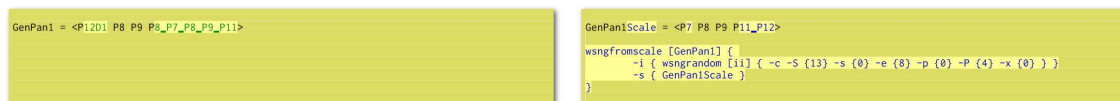
MIDI notes, using the approximate mapping Sorrell gives for rendering the Indonesian slendro scale into equal tempered tuning.

As a simple experiment, I then produced a variant of the piece in which the slendro scale was replaced with a pentatonic blues scale. The result, called *Blue Gamelan*¹² is stylistically similar to the original, but distinct nonetheless, feeling very much caught between two musical cultures. To my western-indoctrinated ears, the piece is eerie and foreboding, and successful enough to inspire a series of derivatives, which will be discussed in the next section.

Clockhouse

Inspired by the visual patterns evident in Figure 5.10, I wondered if it would be possible to combine those with the `wsnrandom` melody generator discussed earlier, to create synthetic gamelan music, along the lines of a Wüürfelspiel.

Each of the literal note sequences in the *Ladrang Wilujeng* encoding corresponds to a phrase from the original score, but very few phrases of the original composition use all of the notes available in the palette, so I elected to use that as the basis for my algorithm.



```
GenPan1 = <P1201 P8 P9 P8_P7_P8_P9_P11>

GenPan1Scale = <P7 P8 P9 P11_P12>
wsnfromscale [GenPan1] {
  -i { wsnrandom [ii] { -c -S (13) -s (8) -e (8) -p (8) -P (4) -x (8) } }
  -s { GenPan1Scale }
}
```

Figure 5.11: Original and derived encodings for GenPan1

I replaced each of the literal melodic fragments from the *Ladrang Wilujeng* with a synthetic fragment that had similar characteristics. For example, by examining the literal melody specification for **GenPan1**, shown on the left of Figure 5.11, I extracted the unique list of pitches it uses, shown on the right as **GenPan1Scale**. I also observed that the source phrase has eight note events, all of duration one, and there are no rests. I then wrote a corresponding **NewGenPan1**, based on the `wsnrandom` statement, in which notes are selected out of **GenPan1Scale** until it generates a fragment of length eight. (Since `wsnrandom` defaults to emitting notes of unit length, with no rests, there was no reason to specify explicit parameters to control these aspects.) The result is **NewGenPan1**, which is randomly generated, but superficially mimics the tonal and rhythmic characteristics of the original **GenPan1**. This process was then repeated for all of the literal phrases in *Ladrang Wilujeng*.

Throughout this process, if any new fragment seemed particularly inappropriate, generating a new variant conforming to the same constraints was accomplished by simply changing the random seed value given in the `-S` parameter of the associated `wsnrandom` statement. In practice, however,

¹²Refer to “*Blue Gamelan*” in the *Wheelsongs* folder of the companion CD.

this was only done three or four times in the development of this piece, usually to eliminate cases where the rhythmic instability setting produced rhythms that were excessively rapid and did not suit the overall mood.

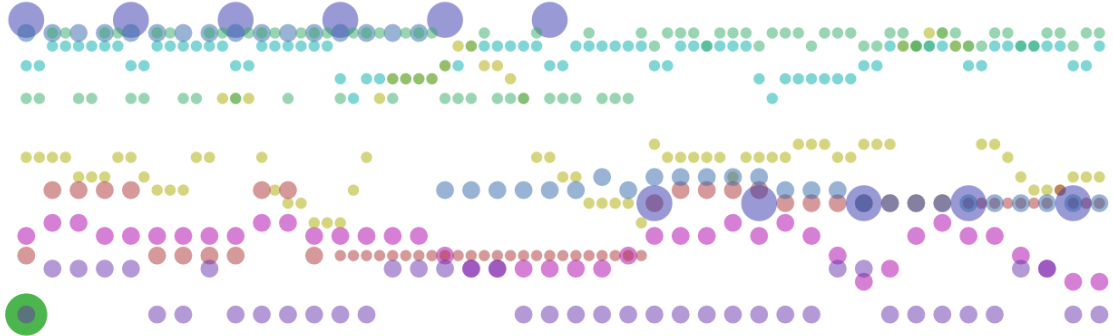


Figure 5.12: Voice flow diagram for *Clockhouse* fragment

The result is a piece called *Clockhouse*, which is similar in basic feel to *Ladrang Wilujeng*, but with its own unique melodic elements. Some of the features of the final composition, such as the darker and lighter note palette shifts at time codes 3:12, 6:24 and 9:36, were constructed intentionally after the first three-minute loop was achieved. But other features, such as the amusing little western-sounding back-fill that first occurs at 00:34, were completely serendipitous structure manifested from out of the coherent noise generator.

Figure 5.12 shows a voice flow diagram of the introductory portion of the composition. In comparison to the *Ladrang Wilujeng* on which it is based, *Clockhouse*¹³ seems more grid-like, with regular, clock-like melodic components. This suits the composition, but it also raises the question of whether such regularity is a feature of this particular composition, or whether it is more systemic, related to the `wsnrandom` filter, or whether it might perhaps even be some unconsidered consequence of structural composing itself.

To explore this, I made a few simple changes to the *Clockhouse* encoding. For each instrument, I kept the same set of phrase generators and pitch palettes, but I shuffled the order in which those pitch palettes were used. I also changed the random seed values used in all the `wsnrandom` filters, and gave some of them a slight duration instability (via the `-g` flag) so that each phrase had a small chance of doubling or halving the duration of notes emitted at some point within the phrase. Also, to more clearly expose the role played by the different voices, I introduced them separately in the final arrangement by muting the beginning of each part for a different length of time. I then balanced this with a similar staggered termination to create a similarly layered outro.

The result is a new composition, the introduction to which is visualized in Figure 5.13. To my ear, the temporal rigidity exhibited by *Clockhouse* has been greatly reduced in this new piece,

¹³Refer to “*Clockhouse*” in the *Wheelsongs* folder of the companion CD.

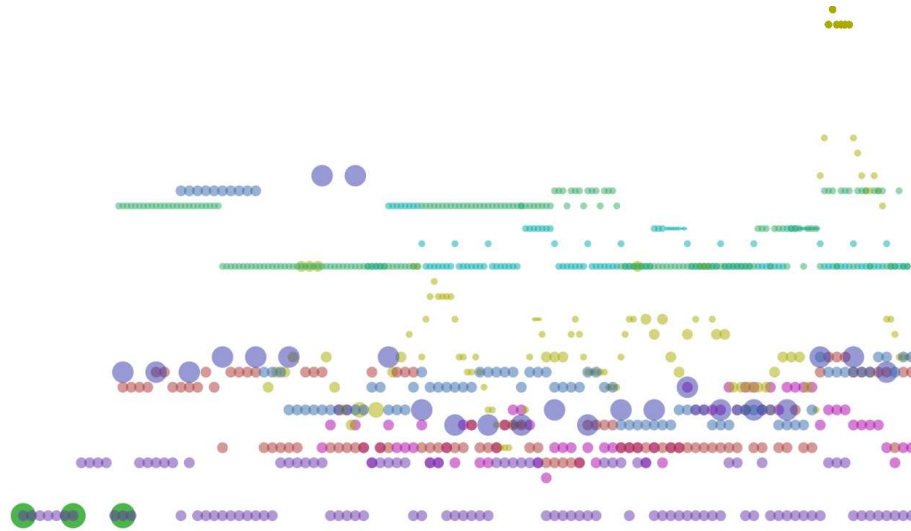


Figure 5.13: Partial voice-flow diagram for *River of Light*

which I call *River of Light*¹⁴. I take this as strong evidence that the regularity of *Clockhouse* was a consequence of the particular filters and arguments chosen, and is not an attribute of Wheelsong or of structural composing in general.

Somewhat unexpectedly, *River of Light* also sounds more gamelan-like than *Clockhouse* does — both to my admittedly inexpert ear and to the three experienced gamelan listeners I mentioned previously — despite the fact that it is one generation further removed from the original, gamelan source material.

The *River of Light* experiment also demonstrates that distinctly different sounding compositions can be achieved from a common scaffold, simply by modifying parameters. In one final experiment on this structure, I wondered how far that common structure could be pushed. Must all compositions derived from *Clockhouse* sound like gamelan?

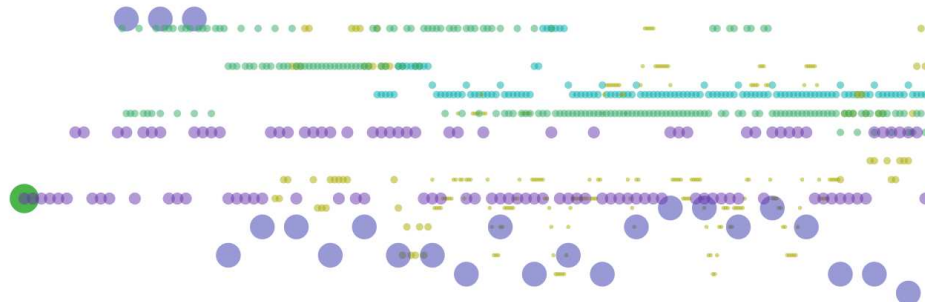


Figure 5.14: Partial voice-flow diagram for *Incident on Pier 6*

¹⁴Refer to “*River of Light*” in the *Wheelsongs* folder of the companion CD.

To create the piece I call *Incident on Pier 6*¹⁵ (visualized in Figure 5.14) I again changed the random number seeds from those in *Clockhouse* and employed slight duration instabilities, but this time the pitch palettes were left in their original, unshuffled order. I then removed half of the voices to create a sparser sound, and changed the voicings to more western instruments. The result of these relatively simple alterations is decidedly *un-gamelan-like*. Instead, it sounds like a score for a gritty waterfront detective movie, hence the title. It has moments of building and then resolving tension, melodic and thematic development, as it drives relentlessly toward an unsatisfying conclusion.

Taken together, these pieces suggest that a single structure can indeed spawn a variety of distinct compositions — within a specific cultural tradition as well as across such boundaries, into entirely different domains.

5.1.4 Tabula Rasa

To this point, the experiments discussed have all been based on encoding an existing work and then deviating that encoding to produce new works. I also wanted to confirm that *Wheelsong* can facilitate creation of original compositions and structures.

One approach to composition that seems well-suited to constructural representation is the notion of phase loops—a process whereby one or more static phrases are repeated in a non-synchronous manner, producing a constantly shifting juxtaposition of the component elements. Steve Reich’s *Clapping Music* [90] is a well-known example in which a single 12-beat percussion phrase is repeated constantly by one performer while the second performer shifts the pattern by one beat after every twelve iterations. This results in a rich assortment of “phantom” beats as the phases progress, until finally it resolves into perfect unison for the final section. (For an example of how this piece works, see my own *The Goddess of Fire*¹⁶, which is adapted from Reich’s original conception.)

In pursuing a more independent original composition, based on this concept of phased loops, I used six simple phrases of differing lengths, repeated continuously until they came into phase to produce some recognizable final passage. To create this composition, I started at the end, by choosing the climactic passage that the piece would resolve into, and working backward from there. For the climax, I selected the final four bars of a simple arrangement of *O Canada*.

The next step was to partition these notes into six different melodic voices, so that when all six are played simultaneously, they reconstitute the desired four bars of music. Note that the voice-flow diagram shown in Figure 5.15, depicts each voice with a different colour.

Rests were then added to the end of five of the phrases, so that all six phrases would have a unique duration. Consequently, when they are looped, they will be mutually out of phase for some

¹⁵Refer to “*Incident on Pier 6*” in the *Wheelsongs* folder of the companion CD.

¹⁶Refer to “*The Goddess of Fire*” in the *Wheelsongs* folder of the companion CD.

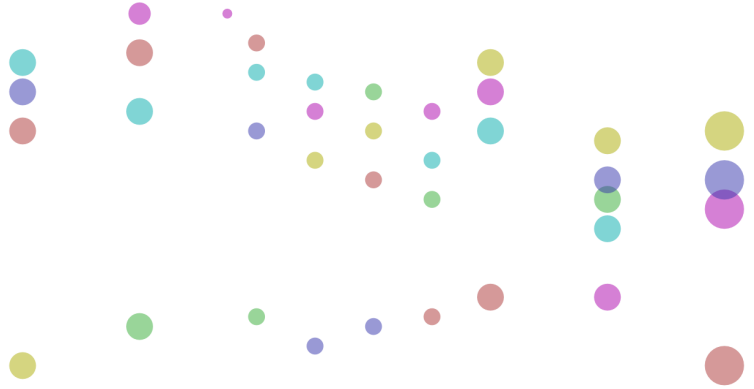


Figure 5.15: Voice partitioning for the final bars of *O Canada*

number of iterations before resolving back into phase and restating the desired finale. The final lengths of the phrases were 16, 18, 20, 24, 30 and 32 beats.

The result is a piece, called *Resolving Our Voices*,¹⁷ that begins in phase, emitting the desired finale, but then goes immediately out of phase and progresses for almost thirteen minutes before coming back into phase. In practice, however, I felt the result was rather monotonous, so I elected to trim the piece further, selecting the final three minutes as an appropriate length, to improve listenability.

This type of composition highlights some of the sketching features of *Wheelsong*, allowing for the quick construction of an algorithmically-oriented composition idea which would have been a time consuming chore to accomplish manually in notation-based editors.

Another approach to building original, structural compositions is suggested by the musical analysis process devised by Heinrich Schenker [99] and described in Chapter 2. Instead of iteratively simplifying a finished composition to reveal the inherent basic structures, I reversed this process, starting with some basic ideas and iteratively embellishing them to create the final work.

Figure 5.16(a) shows the fundamental bass motive upon which this composition, called *The Lament*,¹⁸ was built. A melody was then constructed by subdividing the intervals between successive bass notes with arpeggio forms, to create the situation shown in Figure 5.16(b). (Note that the arpeggios for the last two bass notes prior to the conclusion of the phrase were raised in pitch to create a more interesting harmony.) The melody line was then raised by one octave to create a greater distinction between the two voices, although the figure depicts them in the same octave to demonstrate the relationship more clearly.

This piece was originally composed before the development of *Wheelsong*, as I was exploring ideas of structural composition processes. During this later experimental exploration phase, I

¹⁷Refer to “*Resolving Our Voices*,” in the *Wheelsongs* folder of the companion CD.

¹⁸Refer to “*The Lament*,” in the *Wheelsongs* folder of the companion CD.

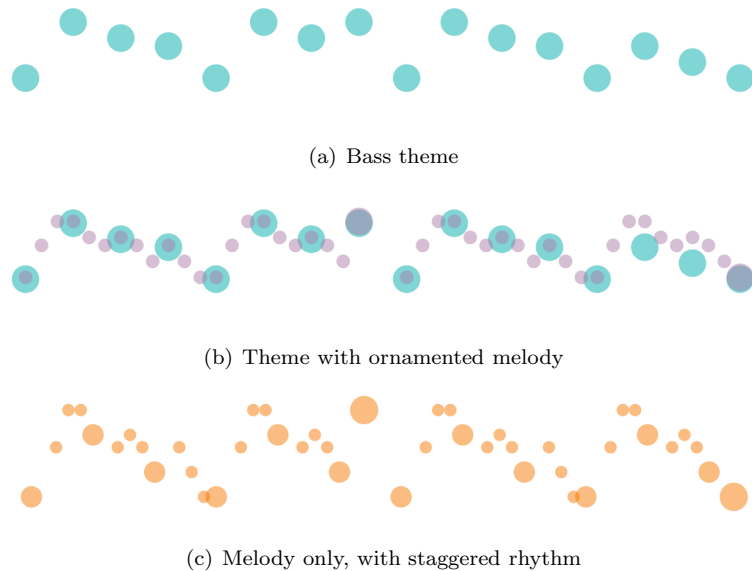


Figure 5.16: Reverse-Schenkerian composition of *The Lament*

encoded the original piece into Wheelsong form to demonstrate that the original approach was viable in this context and to permit subsequent Wheelsong-assisted variation upon the original theme, employing rapid exploration of rhythmic variations, using a palettization scheme similar to pitch palettization, implemented in the `wsngbeatstuffer` filter.

Figure 5.16(c) depicts one experiment in which the fourth of every eight beats is reduced in length and the eighth is extended, thus shifting the rhythmic emphasis. In this particular case, none of the modifications was selected as part of the final composition, but this experiment makes it clear that global changes to the rhythm of a piece can be accomplished easily. I have not experimented any further than this with palettized rhythm changes, and suspect that with further exploration, a number of worthwhile effects could be achieved.

5.1.5 Algorithmic Composition

Wheelsong offers support for a variety of genres and working processes, but it also supports purely algorithmic compositional processes as well. This should come as no surprise, given the extensible plug-in architecture for WSNG transform filters, but in addition to such black-box methods, the WSNT network structure is also conducive to direct algorithmic creation and manipulation. This last series of experimental results illustrates this facility and demonstrates the quality and variety of compositions that can be achieved in this manner.

In this vein, two different tools were developed, each of which treats the WSNT structures as arbitrary tree-like data to be manipulated in the abstract.

The first, called `wsntspawn` generates a random sketch by choosing filter operators pseudo-

```

wsngtranspose [play] {
-n {24}
-i { wsngfromscale [node39] {
-s { wsngscale [node38] { -m {5} }}
-i { wsngsequence [node2] {
-i { wsngloft [node26] {
-C { wsngscale [node25] { -m {8} }}
-i { wsngloft [node23] {
-C { wsngscale [node22] { -m {7} }}
-i { wsngrepeat [node20] {
-c {2}
-i { wsngtranspose [node19] {
-n {-1}
-i { wsnginterpolate [node18] {
-n {0.226658}
-g {0.617233}
-i { wsngfromscale [node17] {
-s { wsngscale [node16] { -m {1} }}
-i { wsngloft [node14] {
-C { wsngrandom [node13] {
-S {51}
-g {0.0725062}
-r {0.452963}
}}
-i { wsnginterval [node9] {
-n {46}
-i {-3}
}} }} }} }} }} }} }} }}
-i { wsngrandom [node30] {
-S {494}
-g {0.0380116}
-r {0.0335226}
}}
-i { wsngrandom [node34] {
-S {381}
-g {0.151985}
-r {0.177053}
}}
-i { wsngscale [node36] { -m {6} }}
}} }} }

```

Figure 5.17: Genetically spawned *Arhythmia*

randomly and assembling them into a syntactically correct WSNT encoding. An example output of this process, called *Arhythmia*,¹⁹ is shown in Figure 5.17. Despite its rather compact encoding, the interwoven use of `wsngloft` and `wsngfromscale` operators produces a rhythmic phrase that modulates in both pitch and key continually and manages to hold the listener’s ear for longer than might have been predicted from its simplicity. The ending, however, constructed by the last three block statements in the figure, are entirely superfluous and give no sense of closure. All of the intricacy derives from the first, deeply nested section of the encoding.

The second tool, called `wsnttreeswap`, combines two WSNT files to generate a hybrid of the two in which a randomly chosen subtree from one input file is replaced with a randomly chosen subtree from the other — a process often referred to as *genetic crossover* [45]. The example shown in Figure 5.18, entitled *Your Sister’s Depression*,²⁰ is a deceptively simple melody created from *Danger Baby* and a thumbnail melody constructure called *Hope*,²¹ yet somehow, the genetic recombination manages to oppose both the strident tension of the former, and the upbeat mood of the latter, presenting a slow, mournful refrain, evocative of its title.

As an extreme stylistic contrast, the piece called *Telco Jackhammer*²² (not illustrated) demonstrates an unusual-sounding marriage of Baroque and electronica styles by combining the *Invention Sketch* and *Arhythmia* structures. What begins as a modernist restatement of the *Invention*’s main theme quickly devolves into a series of tonal beeps and burbles, reminiscent of a ringing telephone, that still somehow manages to maintain a cohesive, thematic identity. Its allegiances to its two

¹⁹Refer to “*Arhythmia*,” in the *Wheelfragments* folder of the companion CD.

²⁰Refer to “*Your Sister’s Depression*,” in the *Wheelfragments* folder of the companion CD.

²¹Refer to “*Hope*,” in the *Wheelfragments* folder of the companion CD.

²²Refer to “*Telco Jackhammer*” in the *Wheelsongs* folder of the companion CD.

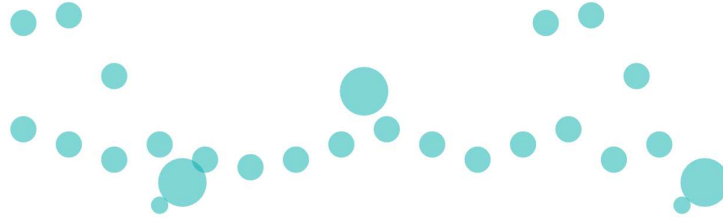


Figure 5.18: Voice flow for *Your Sister's Depression*

contributing predecessors is obvious, but it is an entirely independent composition, that further extends the boundaries of what can apparently be achieved with constructural composing tools.

It should also be noted that these are just three particular examples chosen from over forty sketches that were singled out as having merit over the course of two afternoons spent experimenting with these genetic tools, based on a seed population of only six encodings.

5.1.6 Rejections

Before moving on to analysis of these results, I should put these explorations into greater context by describing some of the pieces that were *not* accepted. For this, I will describe some of the rejections from the explorations of *Invention Sketch*, and the reasons for rejecting them.

The first piece, called *Craptastica 1*²³ was rejected because it sounded too much like the source encoding, and in light of some of the other pieces that had already been selected (such as *Tokyo Invention*), it offered no new charm or appeal of its own.

*Craptastica 2*²⁴ was rejected because it is fairly monotonic and boring and did not seem to venture very far from its core theme. It has some nice rhythmic elements, but compared to the rest of our experiments, it was a little too static.

The piece called *Craptastica 3*²⁵ has the opposite problem: it's too divergent from the source, and has no evident structure or musicality, sounding more like a series of pointless meanderings strung together with awkward pauses and a lot of monotony.

Overall rejection rates are not consistent, as they depend highly upon the nature of the piece(s) involved in the exploration, and the intent of the user. In practice, however, the rejections have not tended to swamp the quasi-acceptable candidates. In each generation, there tends to be one or two promising glimmers — usually enough to keep the user interested in trying another round. To date, sessions that were abandoned in frustration, without producing any satisfying result at all have been relatively infrequent; accounting for less than 10% of the sessions. It should be noted, however, that these results pertain only to myself, as the sole user to date, and were produced

²³Refer to “*Craptastica 1*” in the *Wheelfragments* folder of the companion CD.

²⁴Refer to “*Craptastica 2*” in the *Wheelfragments* folder of the companion CD.

²⁵Refer to “*Craptastica 3*” in the *Wheelfragments* folder of the companion CD.

under generous conditions in which no particular result was being sought. This will have to be examined more rigorously in future experiments.

I note, however, that even when rejected, many of these works still contain passages of musicality and structure, and in other contexts, might have been judged more favourably. I have found this to be true rather often when working with Wheelsong, and the question of which candidate attracts my attention for further development is more a question of what I'm looking for at the time, than it is about the acceptable musicality of the candidates themselves.

5.2 Analysis

Wheelsong was designed to deliver the specific range of features deemed necessary to permit proper examination of my research hypothesis. In this chapter, I have described the explorations conducted to confirm the presence of these features in Wheelsong. While pursuing that analysis, a number of additional observations emerged, suggesting other affordances that were not specifically anticipated. In this section, I will summarize both the confirmations and these additional findings.

5.2.1 Designed Affordances

The exploratory compositions described in Section 5.1 illustrate the degree to which Wheelsong embodies the original design requirements, which in turn carries implications for how subsequent exploration of the research hypothesis should proceed. Those findings are discussed in the following subsections.

Representational Breadth

In order to compare the benefits of manipulating structural representations of music versus manipulations of atomic representations, we must minimize any differences between the tools used in the production and manipulations of each, in order to minimize the potential number of explanations to which perceived qualitative differences can be attributed. Ideally, the same tool would be used for both representational modes, and would be capable of presenting and manipulating each with a common set of tools.

By employing a two-stage design, in which fragmentary, atomic musical fragments in WSNG format are manipulated by a higher-level, structural scaffold described in WSNT format, and then ultimately rendered back into WSNG format for final playback, Wheelsong provides exactly this kind of bi-modal architecture, capable of working seamlessly with either atomic or structural content. As we will see in the discussion of the *compactness* metric in Section 5.2.2, a given work expressed in WSNT might stand anywhere on the spectrum from almost entirely atomic to entirely structural. This suggests that Wheelsong is an encoding scheme that is well suited to serving the

representational needs of this research, and in so doing, sufficiently isolates the notions of structure and atomicity from other potentially complicating influences.

Furthermore, because we are working directly with WSNT and WSNL files, which contain no extraneous, distracting content such as graphical user interface elements, historical revision data, or any of the many other elements that clutter most production-quality composing tool data files, and because we are using non-graphical text editors to directly manipulate them, the Wheelsong experimental platform is well suited to focusing the composer’s attention on the the issues of structure vs. atomicity. It should also be noted that maintaining this purity of focus and bi-modal operation would be valuable considerations in the design of any subsequent user interface aimed at examining user reactions to structural vs. atomic encodings.

Within the set of possible transformational operators, however, it is clear that the current implementation is a work in progress. Just as `wsngrandom` and `wsngharmonize` arose from a discovered need during exploration, it seems likely that new ideas for operators will continue to arise for some time as new and different compositions are explored, and as more composers become involved, each with their own personal views of structure. It is difficult to say what impact future expansion of the operator set will have. New operators might make the system more effective, by offering more ways to construct compositional structures, or by permitting more compact and efficient sketches; but they may also weaken the system by increasing the choice complexity and navigational issues involved in constructing the sketches.

| Count | Filter | Cum. % | Count | Filter | Cum. % |
|-------|------------------------------|--------|-------|------------------------------|--------|
| 332 | <code>wsngtranspose</code> | 27.0 | 5 | <code>wsngchanadd</code> | 98.6 |
| 209 | <code>wsngsequence</code> | 44.0 | 3 | <code>wsngrandomize</code> | 98.8 |
| 160 | <code>wsngrandom</code> | 57.0 | 3 | <code>wsngpattern</code> | 99.1 |
| 156 | <code>wsngfromscale</code> | 69.7 | 3 | <code>wsnginterval</code> | 99.3 |
| 88 | <code>wsngrepeat</code> | 76.8 | 2 | <code>wsngornament</code> | 99.5 |
| 63 | <code>wsngchanmult</code> | 82.0 | 1 | <code>wsngswing</code> | 99.5 |
| 45 | <code>wsngloft</code> | 85.6 | 1 | <code>wsngmetronome</code> | 99.6 |
| 44 | <code>wsngscale</code> | 89.2 | 1 | <code>wsngmetrify</code> | 99.7 |
| 44 | <code>wsnginvert</code> | 92.8 | 1 | <code>wsnglayer</code> | 99.8 |
| 31 | <code>wsngretrograde</code> | 95.3 | 1 | <code>wsngharmonize</code> | 99.9 |
| 29 | <code>wsngselect</code> | 97.7 | 1 | <code>wsngbeatstuffer</code> | 100.0 |
| 6 | <code>wsnginterpolate</code> | 98.2 | | | |

Table 5.1: Cumulative operator occurrences

Table 5.1 shows the frequency of use for the 23 different operational filters that are employed throughout the 20 example compositions discussed in this chapter. More than 80% of the work was done by only six filters, all of which perform relatively abstract, generic operations that require no specific awareness of the musical nature of the data they are modifying. (Recall that although `wsngfromscale` sounds like a musical function, it is in effect just an array lookup oper-

ation.) In comparison, the more music-specific operators, such as `wsngharmonize`, `wsngornament` or `wsngbeatstuffer` are used rarely. This may be a function of my own particular predilections as a composer, but it still suggests that, while it is certainly likely that new niche-purpose tools will be desired in the future, the very fact of their niche focus will likely minimize their impact on the system as a whole. Consequently, I view this as an optimization issue rather than as substantive to the core hypothesis, and will use this observation to narrow the selection of operators presented in subsequent experimental user interfaces to those with the widest general applicability.

Stylistic Breadth

Music comes in an enormous number of styles and genres, existing in one recognizable form or another in every documented culture in human history [82]. To demonstrate that Wheelsong can represent the musical content from each of them would be a monumental undertaking, but this should not be necessary. For my purposes, it should suffice to demonstrate simply that Wheelsong is not limited to an especially narrow set of contexts, and that it can represent music from a variety of genres, styles and cultures. In this chapter I have demonstrated competent sketches from Baroque counterpoint, African traditional, Indonesian gamelan, blues, electronica and modern minimalism. So while it would be foolhardy to say that Wheelsong covers *all* the bases, it seems to support a sufficiently broad range of styles to support an exploration of composition within a western musical sensibility.

Furthermore, the frequency with which even these simple examples have leapt from one starting style into some other, unexpected domain suggests that there may be many more to discover as these experiments continue, perhaps even leading to entirely new styles or genres of music that have not yet been identified. For example, *Tokyo Invention* and *Incident on Pier 6* are both outputs that, in my view, while they are still musical, seem somewhat adrift in terms of their stylistic classification. A more experienced composer might well be able to exploit such beginnings and follow them into new stylistic territory.

Procedural Breadth

In this chapter I have discussed pieces that represent a variety of different compositional methods. Some, such as *Morning Traffic* or *Incident on Pier 6* were created as variations on pre-existing works. *Clockhouse* was created through a series of structural generalizations, while *The Lament* drew on a process of successive refinement or ornamentation. *Resolving Our Voices* derived from an abstract mathematical idea, in a somewhat top-down process, whereas the *Invention Sketch* was reached along a step-by-step, aggregative, bottom-up path. While this is by no means an exhaustive list of all the possible processes by which music can be composed, there are almost as many

approaches represented in this chapter as their are compositions, and this degree of diversity seems sufficient to our current purposes.

Also, to maintain applicability of the Wheelsong platform to the widest possible audience, it will be important to preserve this process-agnosticism as part of the design for any subsequent user interfaces.

User Breadth

In order to study the potential benefits of structural composition encodings to composers, we need to be able to assess those benefits over a wide range of users, representing a full spectrum of musical experience.

Since they were the result of only a single composer, the experiments described in this chapter do little to probe the range of musical experience required for users to exploit any benefits inherent to structural composing, other than to prove that at least one inexpert composer is able to use it. They do, however, suggest some likely areas to investigate further.

Examples such as *Tokyo Invention* or *River of Light*, which are relatively complete-sounding compositions created through simple experimental modifications to pre-existing encodings, seem to be of far higher complexity and quality than what one would typically expect a musical novice to be capable of producing through traditional tools and methods. It seems worthwhile to explore whether inexperienced composers are sufficiently engrossed in such a low-effort, high-reward process that they are willing to continue further along the creative track than they do when confronted by other, more atomic tools. Certainly the effort-reward ratio seems promising in terms of inducing Csikszentmihalyi's sense of creative flow.

Conversely, the dramatic stylistic departure from the source material achieved by examples like *Incident on Pier 6* or *Morning Traffic* suggests that more experienced composers might find that structural composing offers them a powerful avenue to discovering new thematic or stylistic departures from their traditional domains and work processes.

As well, the ability to explore a variety of structurally experimental composition processes, such as those used in the development of *Resolving Our Voices* or *Clockhouse*, without having to learn a computer programming language, might also be of perceived value to more experienced composers.

Regardless of the musical experience of the user, however, it likely that very few users would find the current text-editor-based interface to be desirable, or even tolerable. Consequently, these observations point only to a potential value, inherent to this type of structural encoding scheme. Evaluating that potential will require a more sophisticated and accessible user interface.

5.2.2 Emergent Observations

In this section I will discuss issues and observations that do not pertain directly to our intended features and expectations.

Subjective Evaluation

Recall that the research hypothesis focuses on facilitating the early-stage p_a -creative process, which is best measured subjectively by the users. To that end, I report that my subjective impression of these exploratory trials is one of almost stunned amazement. The quality and diversity of the compositions I have produced with these tools completely defies both my expectations of the tools themselves and my assessment of my own musical capabilities.

Wheelsong seems suited to taking structural ideas and exploring them, such as we saw with *Resolving Our Voices*. It also seems effective at allowing a user to sketch pre-existing scores, as seen with the *Invention Sketch*, or *Mbube*. Subsequently — as *Danger Baby*, and *Incident on Pier 6*, or even *Telco Jackhammer* demonstrate — it is then clearly able to suggest dramatically divergent extrapolations based upon those starting themes.

Where it falls short however — at least, in my own case as a user — is in allowing me to sketch ideas I hear in my head. This is largely due to my weak notation skills—I have very little practice at transcribing musical sounds into the discrete pitches and rhythms needed for notation. This could be addressed by providing performance-capture tools, such as a MIDI input module, but since my current focus is on examining the encoding scheme, rather than the authorship tools, I have not yet done so.

Apart from Wheelsong, my own historical practice for composing has always followed the popular “noodling” model, dominated by exploratory play to discover interesting fragments, and then aggregating those into more complex works. I have found Wheelsong very well suited to this process.

Leveragability

As discussed earlier, to sketch anything effectively, the user must be able to work quickly, making a small number of simple changes to produce an approximation of his intended result, in effect, sacrificing fidelity for speed. This I have translated into a need for compactness of representation, to minimize navigational delay; and leverage, by which I mean permitting local, individual edits to have larger-scale effects on the output.

As a representation increases in size, the artist must pay increasing attention to issues of navigating through the material, rather than on effecting modifications to it, and this is an impediment to flow. Similarly, a conceptual transformation that can only be accomplished by a long sequence of operations will also intrude upon flow, by distracting the user from the effect he is trying to

achieve with the mechanical details of applying it. The explorations described in this chapter have uncovered several such thicknesses in the design of the Wheelsong filters.

In examining the encodings of these experimental compositions, I have found that the number of structural operations required to produce them is higher than I expected, and perhaps higher than an ideal sketching tool would require. Too often I felt I was having to belabour a constructive idea through repetitive (and distracting) operations. For example, the *Clockhouse* encoding contains 26 different invocations of `wsngsequence`, each of which had to be conceived and executed separately.

I also found that, in some situations, I resented the amount of time spent making localized, exploratory adjustments to single input parameters on one or two operator nodes. However, there were other situations in which continued experimental modification of a single parameter was both rewarding and enjoyable. Upon reflection, it seems that the rewarding experiences related to situations in which the localized modifications had large-scale impact on the resulting composition, such as changing the scale against which the palettized compositions were resolved (an adjustment to the `-m` parameter in a `wsngscale` node) or adjusting the rhythmic stability (via the `-g` parameter) on a `wsngrandom` node that was in turn being used to drive the rhythmic cadence of an entire composition via the `wsngbeatstuffer` operation. The more tedious localized parameter explorations seemed more trivial by comparison, especially in the context of working on a preliminary sketch.

Spurred by this observation, I re-examined the set of filter operators and noted that many of them exhibit a sort of myopia in their design, in that their specified operational parameters were informed only by the expectation of what effect they would have on the specific output of the node, and not by what effect they would have on the composition as a whole. So in the cases where the WSNT scaffold placed an operation into a highly leveraged context, exploration of its input parameters was fruitful and enjoyable, but in contexts where the operation was not well leveraged, those explorations were just so much pointless fiddling.

As a consequence of these two perceived shortcomings, I have undertaken a new *leveragability* policy in the design of new operation filters, which dictates that all input parameters are to accept WSNG stream data as input, instead of single numeric fields. In the case where the input stream is single valued, then the operator will behave as before, but in cases where multiple note events are provided, they are to be leveraged into more expansive output. For example, the new `wsngloft` operator can now be used in place of a complex combination of `wsngtranspose` and `wsngsequence`. When the WSNG data on the `-C` param contains a single note event, its pitch is interpreted as a pitch interval by which the source WSNG stream on the `-i` param is to be shifted, so it behaves identically to the `wsngtranspose` operation. But when the `-C` stream contains N note events, the source stream is concatenated N times, with each iteration being transposed by the pitch interval specified by the corresponding note event.

Use of this interface should make for much more compact expressions and a much higher po-

tential to leverage small inputs for large gain. Several other new operators of this type have since been implemented, but as yet, no experiments have been conducted to confirm the expected value of more compact and leveraged constructions that should be achievable with them.

Measurements

Throughout this experimental process, I have attempted to monitor my activities, watching for indicators that might be used to measure or predict my degree of engagement and satisfaction with a work in progress, and one obvious possibility is to measure effort over time. Measuring the effort required to create a composition is difficult though, and presents some challenges of interpretation. If we were to discuss effort in terms of the time taken to achieve a composition, it is unclear whether a long time should be considered bad, because it conflicts with the notion of rapid sketching, or good, since that suggests that the artist was engrossed in the process. No doubt, the answer depends in part on the length and complexity of the piece being composed, since it stands to reason that longer pieces are likely going to take more time.

One measurement I've uncovered for characterizing effort is the number of abandonment decisions made. This kind of exploratory sketching is an iterative process, and when working with text encoding files directly in an editor, my process tended to rely on exploring parameters within one version of the file for some number of iterations until a substantive new idea arose, at which time a copy was made of the file, within which the next phase of parametric explorations could be conducted.

Table 5.2 shows the number of files generated in the construction of five of the above-cited compositions. More specifically, these are the number of iterative exploratory steps taken, or generations explored, after the original source material had been encoded. So, for example, the three steps²⁶ taken to produce *Morning Traffic* do not take into account the experimental constructions that were used to build the *Invention Sketch*.

Another possible measure of effort is the size of the composition, which can be measured either as the number of nodes in the encoding, or even, more simplistically, as the number of note events in the output. It can also be measured more indirectly, by counting the number of distinct WSNL literal streams that are encoded into the composition, which measures effort in terms of the number of input materials that were brought to bear, and speaks somewhat to the effort involved in weaving together a larger number of sources. These measurements are shown for six different compositions in Table 5.2.

²⁶The composition known as *Morning Traffic* actually occurred in the third iteration, but it was ignored at that time, since it violated what I thought I was looking for. It was only at iteration 10, when I realized how much farther afield these results were leaping, that I went back, re-auditioned the earlier works, and discovered *Morning Traffic* in the rejectamenta.

| Example | Generations | # Nodes | # Notes | # Literals |
|------------------|-------------|---------|---------|------------|
| Danger Baby | 6 | 16 | 16 | 16 |
| Invention Sketch | N/A | 96 | 96 | 96 |
| Morning Traffic | 3 | 118 | 118 | 118 |
| Resolving Voices | 5 | 9 | 9 | 9 |
| Wilujeng | 1 | 71 | 71 | 71 |
| Clockhouse | 3 | 153 | 153 | 153 |

Table 5.2: Effort measures for six sample compositions

Based on these measures, we can also compute several other indicators pertaining to leverage, which are shown in Table 5.3 for the same six examples.

The metric I call *compactness* is a measurement of the efficiency of a sketch, contrasting its overall output length against its structural size. It is computed by N/F where N is the number of note events found in the output WSNG stream and F is the number of filter nodes in the WSNT sketch. In principle, sketches are more powerful if fewer nodes are needed to generate the output.

| Example | Compactness | Fragment 'n | Expansion | Capture |
|------------------|-------------|-------------|-----------|---------|
| Danger Baby | 13.3 | 4.0 | 6.4 | 1.0 |
| Invention Sketch | 4.9 | 4.6 | 9.6 | 24.0 |
| Morning Traffic | 4.0 | 5.1 | 4.6 | 1.8 |
| Resolving Voices | 146.8 | 1.5 | 23.2 | 0.8 |
| Wilujeng | 58.9 | 1.2 | 4.3 | 1.1 |
| Clockhouse | 69.6 | 3.3 | 53.8 | 13.9 |

Table 5.3: Efficiency measures for six sample compositions

Fragmentation is a more subtle measurement of efficiency, this time contrasting the number of filter nodes used to construct a piece with the number of literal melodic fragments upon which it is built. Expressed as the number of filter nodes divided by the number of declared WSNG literals employed, fragmentation gives a sense of how much of a composition’s structure is influenced by any of the literal streams. A higher fragmentation score for a composition suggests that modifying or replacing any one of its literals will have less impact than if such a transformation were applied to a less fragmented work.

Similar to fragmentation, *expansion* relates the size of the work to its encoded literals, but in this case I am contrasting the number of output notes to the total number of input notes encoded into its literals.

In practice, any composition could be sketched by simply stating its entire content as a literal string. This would produce the same output as a more structurally mature encoding, but all of its musical logic would be locked into the literal fragment, and none of it would be rendered into manipulable form in the structural scaffolding of the WSNT network. The crucial concept here

is the ratio between the number of structural nodes used, against the length of the longest literal fragment, a measurement I call *capture*, which reflects how much of the musicality has been migrated from the literals into the structures. A work with a high capture score is one that I expect might be more likely to produce pleasing results in the hands of novice users who do not have enough musical training of their own to impart it to their work.

These measures presume that all of the literals and all of the independent node hierarchies declared within the file actually contribute to the final output, but this is not a requirement of the file format itself. In fact, it is quite common to accumulate orphaned elements as a work progresses. The measurements reported in Tables 5.2 and 5.3 were calculated after all such orphans were removed.

At this point, however, these metrics are simply plausible measurements, which, based on my experience, might be useful in predicting the value of an encoding being considered as raw input for a structural composition. These metrics will be examined in more detail in Chapter 6 where we will be able to assess their correspondence with compositions that have been evaluated by a panel of independent judges.

Another dimension in which the above metrics might prove useful is in estimating the state of creative maturity of the work in progress. If we are to devise software that presents features to the user that are appropriate to his current style of cognitive engagement, we will need a way to estimate his current location on the arc from early-stage to late-stage development. I suggest that early-stage manipulation will be evidenced primarily by growth and refinement of the structural network of the encoding, whereas late-stage work will be more focused on refinements to the parameter values and the literal fragment specifications. At this stage in the research, I do not have sufficient data to evaluate the utility of the metrics in this regard, but I hope to explore this issue more deeply in subsequent experiments.

Naivety

Examples like *Invention Sketch* and *Clockhouse* suggest that one does not need to base the structural design of a WSNT scaffold on a deep understanding of the musicological foundations of the piece in question. Despite the fact that I have very limited knowledge of the rules of counterpoint composition, and essentially none at all regarding gamelan, the sketches I made, based on my naively identified patterns, were still adequate to being transformed (again naively) into credible works in their own right.

One might surmise from this that the presence of pattern in the formation of a piece is, in itself, a more important component of its perceived musical quality than the inclusion of any particular set of “theory-based” patterns. This is, of course, a highly contentious position, and would require much more study before attempting to draw such a conclusion. I include it here only as a tantalizing

suggestion for further research.

Serendipity

One of the most rewarding and unexpected findings during this exercise was the frequency of serendipitous results. By this, I mean simply a welcome but unexpected discovery, made by chance, in the sense Boden described [9, p. 234]. The most obvious examples of such discoveries occurred when I was working toward some specific anticipated result, and achieved something totally unexpected along the way.

This was the case with *Morning Traffic*. In that exploration, I had completed the *Invention Sketch* encoding and was searching for new Baroque-style works. I found many of them, but like *Wüürfelspiels*, the new pieces sounded highly derivative of the source material, so I rejected them. It was not until I was exploring the 10th generation of derivative structures that I realized that the candidates I was auditioning represented a wider stylistic range than I was expecting. Some of those rejects were not bad compositions, they just were not very Baroque-sounding. A re-examination of the rejected material turned up *Morning Traffic*, which I had bypassed as early as the 3rd generation.

Another frequent cause of serendipity is the so-called “mistake.” In order to prepare the six phased voices for *Resolving Our Voices*, I first had to encode the concluding, four-bar *O Canada Fragment*²⁷ in WSNG format. The score from which I was working was an easy-piano arrangement consisting of a fairly plain, unornamented chord progression. A single mistake, in which an entered note was given the wrong duration, resulted in a completely new interpretation of the stodgy chord progression, called *O Funkada!*,²⁸ with a more exhilarating, jazzier cadence. If such a transcription error had occurred within a traditional atomic notation editor, the result would have been a single, mis-timed note, and in all likelihood, would have gone completely unnoticed. But the system of short-hand entry used in WSNG format encodings assumes that for any unspecified note event parameter, the value from the preceding note should be re-used, and this caused the single mistake to propagate through the remainder of the encoding. To demonstrate that this is not an isolated event, a very similar mistake in the encoding of Franz Liszt’s *Deuxieme Ballade*, resulted in a provocative re-interpretation of this romantic classic into a pounding boogie-woogie, that I call the *Bleuxieme Ballade*.²⁹

But “mistakes” can be made in many other aspects of the encoding and manipulation process as well. As part of my process for exploring the derivatives of *Clockhouse*, it was my habit to assign a different metallophonic instrument to each of the different voices. When I began the generation that ultimately resulted in *Incident on Pier 6*, I decided to switch from metallophones to stringed

²⁷Refer to “*O Canada Fragment*” in the *Wheelfragments* folder of the companion CD.

²⁸Refer to “*O Funkada!*” in the *Wheelfragments* folder of the companion CD.

²⁹Refer to “*Bleuxieme Ballade*.” in the *Wheelfragments* folder of the companion CD.

instruments, just for variety. As it happens, the MIDI instrument library I was using had a block of metallophone in voices 9 – 15, and a block of stringed instruments in voices #24 – 46. So the easiest way to effect the switch quickly was to simply apply a correction factor to the voice channel components in the WSNG-format output to move them all from one zone to the other. I could have done this with either the `wsngchanmult`, applying a factor of 3, or I could have done it with `wsngchanadd`, applying an increment of approximately 20. I elected to use the multiplier, but I was unhappy with the result, so I changed it back. Rather than remove the operator that I had just inserted into the tree, however, I simply set the increment back to 0, which effectively neutralized the node, but left it in place so that I could experiment with it later. Obviously, setting the increment to 0 would have been the correct thing to do if I had actually used the `wsngchanadd` operator, but since I had actually used `wsngchanmult`, in my temporary confusion, I ended up multiplying all the voice values by 0, instead of adding 0.

The result of this seemingly simple error was that all of the channels, which had previously been voiced by a variety of distinct instruments, and had been competing with each other for acoustic dominance, were now all being played on a single instrument, and where the separate voices had produced a random, chaotic sounding mess, placing all the voices on the same instrument allowed them to work together. No one of the voices alone had any particular melodic appeal, but together, a melody *was* completely evident, passing back and forth between one voice and the next. It was only by accidentally merging them into a single instrument that the melody was brought to light, and ultimately refined into *Incident on Pier 6*.

The important thing to observe about these examples is that, with structural encoding at the foundation, even simple transcription errors can often be magnified into valuable structural modifications of the material. Notation editors, on the other hand, with their atomic focus on individual note placement, rarely produce anything other than a sour note when the mouse slips. As a result of this phenomemon, with WheelSong, user errors can become a powerful source of creative diversity — whether they arise from clumsiness, forgetfulness, or even ignorance.

Rehabilitating Failures

Of the 18 file generations listed in Table 5.2, which produced five “good” compositions, there is a temptation to view the remaining 13 as unsuccessful or as failures. Indeed, I have called them exactly that, elsewhere in this document, since they were rejected for my purposes at the time. Like the sketches in a painter’s sketchbook, however, unsuccessful sketches are not always valueless. Often, they contain the germ of an idea, or an approach, that inspires us later. *Morning Traffic* is an example of this principle, in that it was inspired by a line of inquiry that was initially abandoned while creating *Clockhouse*, and was only re-examined and found valuable later, when the context of evaluation had shifted.

Every time I review the experimental archives, I am struck by the variety and diversity of the pieces I find there. Sometimes flashes of brilliance occur in the middle of an otherwise tedious work, and other times, I find entire pieces that somehow escaped my notice the first time around. For example, I’ve just pulled one out of the archive while writing this paragraph. For obvious reasons, I will call this piece *Resurrected*.³⁰ It was originally rejected as a result of my impatience. The opening bars are identical to one of the sources from which it was hybridized (*Tokyo Invention*), and during the original exploration, I neglected to give it enough time to demonstrate its potential, which emerges later.

What has become apparent from examples like these is that even the rejectementa of a Wheel-song session can have value, and in the context of inspiring early-stage creativity, this strikes me as a potent quality for a creativity facilitation tool to have.

5.3 Other Users

Wheelsong has been exposed to twenty-two casual users to date, ranging in age from ten years to well over seventy (see Table 5.4), and representing a diversity of musical and composing experience, from rank beginner to seasoned professional.

| Age Group | < 13 | 13-20 | 20-30 | 30-40 | 40-50 | 50-60 | 60-70 | > 70 |
|-----------|------|-------|-------|-------|-------|-------|-------|------|
| Count | 2 | 6 | 0 | 4 | 4 | 4 | 1 | 1 |

Table 5.4: Age distribution of users

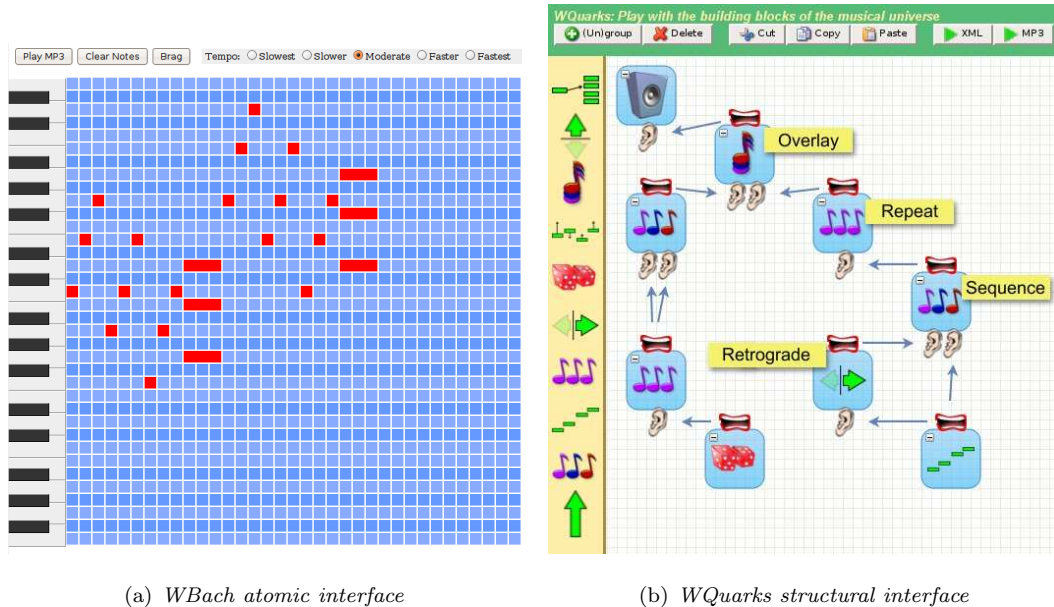
In these casual interactions, users are exposed to two different interfaces: the WBach interface (shown in Figure 5.19(a)) providing atomic interactions, and the WQuarks interface (shown in Figure 5.19(b)), which provides structural interactions.

WBach exposes users to the WSNG layer of Wheelsong, presenting a note-based, atomic interaction in which they can enable or disable the pitch and duration of note events by clicking on a simple grid to enable or disable notes of a given pitch (expressed on the vertical axis) and at a particular time (via the horizontal axis).

WQuarks, on the other hand, exposes the structural WSNT layer of Wheelsong, allowing users to manipulate generated or pre-encoded atomic fragments by connecting them with operators in a WSNT network graph, by dragging the outputs of one node to the inputs of others.

These interfaces were not well refined and introduced some uncertainty regarding the measured effects and whether they were attributable to the interfaces themselves or to the specific encoding schemes behind them, but I deemed them necessary, since I felt that the text-editor interface would

³⁰Refer to “*Resurrected*.” in the *Wheelfragments* folder of the companion CD.



(a) *WBach atomic interface*

(b) *WQuarks structural interface*

Figure 5.19: Rudimentary editor interfaces

have been too intimidating for the average user. Indeed, the majority of problems that users were observed to be having pertained directly to their attempts to manipulate the interface elements. Furthermore, these incidents were both more frequent and more severe in the WQuarks interface than they were in WBach, which likely contributed to a preference bias against WQuarks.

The awkwardness of these these interfaces made it difficult to assess any subtle aspects of comparison, although one particular signal seemed quite strong. Upon completing their explorations, users were asked which of the two systems would be better suited to developing musical ideas they already had in their heads, for which 17 of the 22 chose WBach (the atomics interface). Conversely, when asked which of the two tools would be more likely to lead them to new musical ideas, 16 of them chose WQuarks.

These two results are important to consider together. While there was a strong preference expressed by these users for the constructural scheme’s ability to inspire new musical ideas, taken on its own, this might be attributed to simple acquiescence bias [62]. I note, however, that users were not aware of the intent of the research, and both of these questions presented a positive hypothesis to which they could have acquiesced. The fact that they differentiated between the two, and strongly (approximately $\frac{3}{4}$), suggests that there is indeed a signal here being measured, one that strongly endorses the structural approach over the atomic for early stage exploratory ideation.

In addition to these more quantitative results, anecdotal comments made by several of the users serve to illustrate their general sentiments. These are discussed in the next few brief subsections.

5.3.1 User A

This first user, who is in her 50s, sight-reads piano and has played since childhood, but does not compose. She told me that she, “...enjoyed [WQuarks], where the idea of creating by ear, tone and variation was cool!” One of her composition fragments (shown in Figure 5.20) shows her fascination with rapid-fire arpeggios, which she laughed at when she first achieved it, because there was “no way” she would ever have thought to try composing something that fast because she’d never be able to play it.”

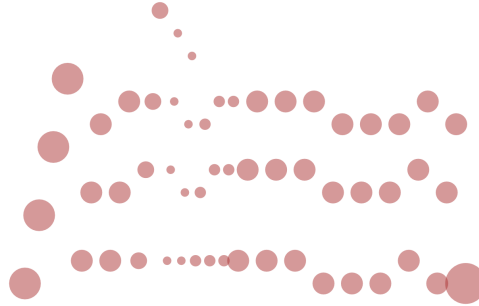


Figure 5.20: Rapid-fire arpeggios

5.3.2 User B

This user, in her 40s, has no musical or compositional experience beyond long-forgotten music lessons in grade school. She remarked that WQuarks seemed like it was easy to experiment with and to try lots of variations on musical ideas. By objective standards, her fragment (shown in Figure 5.21) would be judged as trivially simplistic, but subjectively, she stated that she had no idea she knew enough about music to compose anything this good.

This comment seems to perfectly illustrate the issue of the P-Judgment premise. The user was clearly pleased with the material and, had there been more time, would likely have continued to develop it further – even though an objective evaluation would almost certainly have brought that exploration to a halt, denying her the sense of creative satisfaction and stifling any p_a -creative achievements she might subsequently have made.

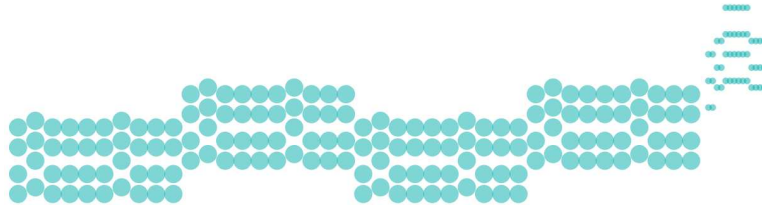


Figure 5.21: Unexpected result

5.3.3 User C

The benefits of structural tools are not necessarily limited only to novice composers. A full-time, professional musician and composer (in the 40 – 50 age range) remarked that WQuark’s balance of mystery and reward reminded him of his first tentative steps into music when he was a child, and that he had not felt that kind of fascination with music in a long, long time.

5.3.4 User D

To date, there has been only one user with both a composing background and a sufficient degree of experience with programming to be comfortable tackling Wheelsong through its primary, textual interface, other than the author. Unlike the other, casual users above, User D experimented with Wheelsong for one or two hours per day, over a period of two weeks.

When asked, after the fact, to describe his objectives during his experiments, he stated that he had gone in with the typical composing objectives, attempting to produce the music he heard in his head, but that it quickly became apparent that this was the wrong way to approach Wheelsong, and so he shifted to a more exploratory footing, attempting to simply sketch ideas and see where they went. This approach, he reported, was much more successful.

Regarding the creative diversity and/or quality of the outputs he was able to produce, he stated that “The creative diversity was great. I found it especially useful for exploring ‘process music’. Guys like Steve Reich would love this thing. It’s really geared towards that minimalist aesthetic of *generation* as opposed to exposition.”

In attempting to describe the difference in cognitive experience between Wheelsong and other composing tools with which he is familiar, User D reported that Wheelsong was “[u]tterly different from working with traditional notation based processes, either on paper or in Finale/Sibelius” and that, while it bore similarities to some other systems, such as IRCAM’s Patchwork, its limitations in comparison to such music programming environments, “made it a better tool for creativity.” When asked elaborate, he offered that, “A completely blank canvas is a terrifying thing for a mediocre artist (which is the most charitable possible classification for my compositional skills); having some initial constraints and directions makes creativity enormously easier for me.”

Judging by comments like these, it seems that, while Wheelsong can facilitate both novice and experienced composers in their creative ambitions, it does so via different mechanisms, and in different ways, for each group. The benefits reported by novice users seem best characterized as pertaining to the system’s ability to keep their musical wanderings legitimately musical, and not allowing them to wander too far into cacophony. With more experienced composers, however, the benefits seem more about the structural composing’s ability to take them in new and unexpected directions.

5.4 Conclusions

These exploratory experiments were an important step in probing the boundaries, limits and affordances of the system. Wheelsong appears to provide the kind of flexibility that will allow me to explore issues of p_a -creative facilitation without constraining that examination into a narrow style of music, user or process, any of which would weaken the potential conclusions that could be drawn from its use.

In addition to confirming Wheelsong's basic utility as my research platform, I have also uncovered a number of guidelines that will help shape subsequent feature development and experiment design, as well as highlighting a number of tantalizing possibilities that may bear further investigation in their own right.

With the results of this loosely structured exploration in hand, I am now able to devise more specific research experiments around more specific questions. In fact, I now have many more questions than I can possibly explore in a single thesis. For the remainder of this thesis document, I will explore specific measurements of the creative facilitation offered by Wheelsong, but in future experiments I hope to investigate many of the questions raised here, such as the optimal musical operator set to balance expressive breadth against functional compactness, or investigating the cost-benefit tradeoffs of black-box versus white-box construct manipulation. The list of questions seems boundless.

CHAPTER 6

RICHNESS TEST

6.1 Introduction

My formal research hypothesis states, in part, that, “The p_a -creative richness of musical sketches produced by composition software tools can be increased over that of pitch notation-based tools by representing musical compositions as constructive hierarchies of note-sequence transformations...” More compactly, the premise is that using constructural encodings will increase the p_a -creative richness of the candidates produced.

While the explorations in the previous chapter seem to support this in observations such as the wide stylistic diversity imparted by seemingly minor changes to the structure, and the frequency of serendipitous outcomes, these observations are merely suggestive. In this chapter I will investigate the hypothesis more directly by describing an experiment in which I have attempted to objectively measure a difference in the p_a -creative richness of musical compositions produced by means of constructural vs. atomic encodings. More precisely, I will investigate the following hypothesis.

Hypothesis 4 *There is a measurable difference in the distribution of creative richness attribute scores assessed on a population of musical candidates composed atomically, as compared to a population generated by a comparable constructural process.*

In this experiment, three significant considerations guided the design. First, recall that p_a -creative merit is an entirely subjective assessment. Second, my intended users come from all levels of musical experience. Finally, subjecting users to two different user interfaces would introduce a source of bias and uncertainty, since it is the affordances of the encoding schemes I wish to compare, not contributions from the interfaces built on top of them.

These considerations led to the following experimental framework, repeated for each of the three phases, under different conditions:

- A traditional composition methodology was chosen to serve as a model
- An algorithm was devised and implemented to simulate that methodology using atomic representation

- An algorithm was devised and implemented to simulate that methodology using constructural representation
- A number of example MP3 fragments were produced pseudo-randomly using the atomic method
- An equal number of example MP3 fragments were produced pseudo-randomly using the constructural method
- These two sets of MP3 fragments were pooled together, interleaved in a random order
- A subset of the creativity-analogous criteria (discussed in Section 3.2.2) were chosen for their relevance to the specific methodology being modeled
- A group of independent human judges was assembled, representing a wide range of musical and compositional experiences and tastes
- The judges were each presented with the combined pool of samples and asked to rate each sample on each of the criteria, on a 5-point Likert scale
- Auditioning of samples and attribute score inputs were managed through a web-based form, using the judges' own computer and audio equipment

The experiment was conducted three times—each time utilizing a different compositional methodology chosen from the discussion in Section 2.4. In the following sections, I will describe the three experimental protocols employed and the algorithms developed, and then present the actual experimental results, followed by an analysis of those results, and finally, some conclusions, in the context of the original hypothesis.

6.2 Generation Phase

The blank canvas can be a daunting prospect for even the most seasoned of artists, yet one of the more common approaches to composition is to work from scratch, building up successively more complex works from smaller, spontaneously invented elements. One avenue to supporting the creative efforts of a user would be to provide an environment in which that blank slate was measurably prone to producing a diverse collection of fragmentary ideas that are of higher creative merit, compared to other available tools. This is what I mean when I suggest “increasing the p_a -creative density” of a tool.

The primary goal of this phase of the experiment is to assess the p_a -creative density produced by atomic vs. constructural tools, when starting from a blank slate. In addition, I will also use the data collected in this phase to probe the orthogonality of the metrics asserted in Section 3.2.2 as being co-variant with creative merit.

6.2.1 Experiment Design

Starting from a blank slate in each case, a population of 12 atomic and 12 constructural compositions was generated at random from the set of all syntactically valid compositions in their respective representation schemes.

Those populations were then mixed in randomized order and presented to a series of human judges, via a web interface, through which each judge evaluated every candidate for the five different subjective attributes. The judges were not told the purpose of the experiment, nor were they aware that there were two classes of candidates being judged.

In the following subsections, I will now describe these various constraints and protocols in greater detail.

Effort Constraints

As discussed in Section 2.2.3, there are many types of musical sketches, encompassing many different scopes and classes of content. In an effort to refrain from privileging any one sub-species of sketch, I have characterized them as all working with “fragments.” How musicians limit the scope of their experimental sketches can be quite varied, and depends largely on the type of sketch in question. Some composers may work with specific melodic phrases, while others might work with a particular chord progression, or a repeating pattern of figured bass, or possibly a thematic recurrence pattern such as the Bartók example shown in Figure 2.7. Each of these is composed of different kinds of information, and explores a different aspect of the overall work in progress. Consequently, using terms like “bar,” “motif,” “theme,” or “melody” are each too narrow, and imply a particular kind of sketch. Hence, I use the more type-agnostic term “fragment” to mean any individual musical sketch.

But now that I come to assess them, it becomes necessary to measure them in some equitable manner that allows me to examine only ‘comparable’ candidates. On the one hand, I have atomic fragments, which are composed of individual note events, each having eight component scalar values. On the other hand, there are constructural fragments, composed of a network of generative and transformative musical operators, each of which has a potentially differing edge connectivity and number of numerical control parameters.

The unifying dimension I have chosen, which will permit a more apples-to-apples comparison between the two schemes, is that of the *effort* used to encode them. For this I use a definition similar to Levenshtein’s *edit distance* [65], which is used to compare strings, although instead of computing the distance between two fragments, as Levenshtein proposes, I compute the distance between each fragment and its corresponding null candidate.

In the case of atomic fragments, measuring this effort is fairly straightforward—every new note event is a new edit step, or quantum of effort, so $e_a = \#(\text{note events})$.

The case with constructural representations, however, is a bit more complex. Clearly, each operator node added to the graph of a WSNT file is a new quantum of effort in the encoding, but it seemed that assigning values to its input parameters was also a form of effort, so those are also included as part of the constructural effort calculation. So $e_c = \#(\text{operator nodes}) + \#(\text{node parameters})$.

It could be argued that the assignment of transformation operators is comparable to setting the parametric values of pitch, time, etc. on note events, but I reasoned that a note event is incomplete without its component values, whereas transform operations have default settings for each control parameter and are therefore fully specified without the additional numerical inputs. At worst, this decision privileges atomic fragments by allowing greater effort to be allocated to each candidate, in comparison to the constructural candidates to which they will be compared.

The next step was to establish some constraints around this notion of effort, to ensure that all generated candidates were at least approximately comparable to what a composer might think of as a useful sketch. Comparing an opera-length fragment to a six-note melody would essentially be meaningless. To establish appropriate bounds for this, I took my cues from the notion of a melodic phrase, which is a common formative element mentioned by composers when they are discussing their creative process. Examining my own archive of musical sketches, I found none with fewer than eight notes, and while some sketches were hundreds of notes long, I was unable to find any whose essence required more than fifty or sixty note events to express. So, for want of any better metric, I chose these as my limits. In this experiment, the random generation of both atomic and constructural fragments were constrained to effort scores lying between 8 and 60.

| | | | | | | | | | | | | | | |
|-------------|----|----|----|----|----|----|----|----|----|----|----|----|------------------|------------------|
| Atomic | 16 | 18 | 19 | 29 | 31 | 31 | 36 | 40 | 41 | 43 | 52 | 52 | $\Sigma e = 408$ | $\bar{e} = 34.0$ |
| Construct'l | 8 | 10 | 12 | 16 | 16 | 19 | 22 | 27 | 38 | 41 | 44 | 57 | $\Sigma e = 310$ | $\bar{e} = 25.8$ |

Table 6.1: Effort metrics, e , for the generative fragment population

Table 6.1 shows the effort score for the 24 total samples generated for the experiment, broken down by scheme. Note that they all fall well inside the prescribed range, and, as a curious consequence of the random process, there appears to be a significantly higher degree of effort allotted to the atomic samples, which, as a class, represented 31% more effort than the constructural group, which may have unduly privileged the atomic population in the evaluations.

Note Parameter Constraints

In addition to limiting the effort scores of each candidate, I also felt it was important to limit the acoustic and temporal ranges of the sample populations. Since this is intended as an examination

of creative musical value in a human context, it seemed reasonable to limit the pitch range to those pitches that are accessible and comfortable to human hearing. Watt and Quinn [119] provide us with a number of statistical measures of the distribution of pitches and durations in what they call “ordinary melodies,” but these measures are biased toward historically successful musical melodies, rather than all melodies that can be perceived comfortably by human hearing. Since I am attempting to explore the inherent affordances of the representation schemes themselves, it seemed inappropriate to bias the test toward successful exemplars. So instead of the Watt and Quinn values, I used the standard piano keyboard to define the experimental pitch range, reasoning that pianos are unlikely to include pitches that are generally unacceptable to the ear, since this would increase the cost of producing them, while adding no perceived value. All pitches generated were therefore limited to the standard 88 piano pitches, valued from MIDI 21 through MIDI 108, inclusive.

In the case of structural samples, the situation is more complicated. While it was still possible to limit the range of pitches generated by leaf nodes in the generative network, it proved ultimately infeasible to constrain their subsequent transformation, since doing so would require a sophisticated understanding of how the network of operations was constructed. As a result, while the WSNG samples were constrained to the standard keyboard range, the WSNT samples were effectively unconstrained, and the generated candidates employed notes from as low as MIDI 24 to as high as 266. As a safeguard, the `wsnt2midi` renderer has a built-in mechanism that limits pitch values to the range 0 through 127, mapping outlier pitches back into that range cyclically, using modulus arithmetic. So a pitch request of 128 will be transformed into a 0, pitch 128 will become 1, and so on, ad infinitum. Negative valued pitch requests are similarly treated.

Unfortunately, this permits many pitches outside of the standard keyboard range (0–20 and 109–127), which sound rather harsh and unpleasant when played with the default grand piano voice that was used for this experiment. This then offers another slight advantage to the atomic population in the assessment, since they are not burdened by the presences of these unpleasant sounds.

Another limit imposed on the candidates was that of time. While there is no reason why a note with a duration of 0.5 seconds should be considered any more or less musically creative than a duration of 2 seconds, there is a limit to the patience of the human judges — especially when they will have to audition 24 candidates — so an upper bound was imposed on the length of any one note event, to keep the aggregate lengths of the samples from becoming too lengthy. Again, by appealing to my own archive of sketch material, a somewhat arbitrary length of 5 seconds was chosen. None of the sketches I examined contained individual note events of any longer duration than that.

At the other extreme, a note of zero duration would obviously be degenerate, in the sense

that, without duration, it would add nothing to the composition, so a non-zero minimum positive duration was enforced as well. After auditioning a succession of shorter and shorter note events, it seemed that notes with durations of less than 0.02 seconds were effectively inaudible, so the minimum note duration limit was set at 0.02 seconds. This is in keeping with the general acoustic perception parameters summarized by Moore [76].

Again, while those same limits were applied to the generative construction operators, it was not feasible to track all the potential transformations that might modify note timing through the network. Instead, the final constructed melodies were free to span a potentially wider duration range, extending to extreme values that are most likely to be interpreted as either unpleasant or distracting. Despite this intended bias in favour of atomic candidates, however in practice, the population of atomic samples that was actually generated for this assessment contained note event durations between 0.02 and 4.3 seconds, while the notes of the structural candidates ranged between 0.14 and 2.1 seconds. It is unclear whether this discrepancy in note durations contributed significantly to the measured differences in the judges' scores, but I suspect that there were other, more substantial causes, and that any contribution made by this relative duration imbalance was swamped by those other factors.

Fragment Creation

These above constraints and considerations were then used to guide development of two utilities: *wsngspawn*, to generate random, valid WSNG-format composition fragments; and *wsntgerminate*, to emit random, valid WSNT-format fragments.

```

BEGIN
  minPitch = 21
  maxPitch = 108
  minDuration = 0.02
  maxDuration = 10
  minNumNotes = 8
  maxNumNotes = 60

  numNotes = randomIntegerBetween(minNumNotes, maxNumNotes)

  WHILE numNotes
    pitch = randomIntegerBetween(minPitch, maxPitch)
    duration = randomFloatBetween(minDuration, maxDuration)
    printNoteEvent(pitch, duration)
  END

```

Figure 6.1: Pseudo-code of atomic generation algorithm

Figure 6.1 shows the *wsngspawn* in pseudo-code form as a straight-forward loop that generates pitches and durations from within the range of permissible values and emits them in sequence as WSNG-formatted output.

By comparison to that atomic algorithm, *wsntgerminate*'s structural one (shown in pseudo-code form in Figure 6.2) is more elaborate, building a random tree of interconnected operator nodes.

Since the average number of numerical parameters supported by the different operator nodes was approximately 3, the minimum and maximum number of nodes permitted in the generated trees was divided by three, to approximately equate the total effort scores with those of the atomic fragments.

```

BEGIN
  minPitch = 21
  maxPitch = 108
  minDuration = 0.02
  maxDuration = 10
  minNumNodes = 2
  maxNumNodes = 20

  listOfTerminalOperators = [wsngrandom, wsngscale, wsnginterval]
  listOfNonTerminalOperators = [wsngfromscale, wsngcycle, wsngloft,
    wsnginterpolate, wsngstomp, wsngsequence, wsngretrograde, wsnginvert,
    wsngpatternweave, wsngrandomize, wsngharmonize]
  listOfAllOperators = listOfTerminalOperators + listOfNonTerminalOperators

  tree.initialize()

  WHILE tree.hasUnterminatedBranches()
    parentNode = tree.getFirstUnterminatedNode()
    FOR inputNum FROM 0 TO parentNode.getNumberRequiredNodeInputs()
      newNode = SpawnNewNode(tree)
      parentNode.appendInput(newNode)

  PRINT tree
END

FUNCTION SpawnNewNode(tree)
  IF tree.size < minNumNodes
    childOperatorPool = listOfNonTerminalOperators
  ELSE IF tree.size < maxNumNodes
    childOperatorPool = listOfAllOperators
  ELSE
    childOperatorPool = listOfTerminalOperators

  newNode = ChooseRandomOperatorFromList(childOperatorPool)
  newNode.assignRandomNumericParameters()

  RETURN newNode

```

Figure 6.2: Pseudo-code of constructural germination algorithm

Metrics

Recall that I am trying to measure the creative richness of a population of randomly chosen musical fragments, and am doing so in the specific context of being able to use those fragments to inspire and develop more complex creative works. This characterization exposes two issues that need to be evaluated: the creative merits of the musical candidates in their own right, and the density with which those various attributes occur in the population.

The notion of measuring creative merit has already (in Section 3.2.2) been described as correlating with five co-variant analogs: newness, unexpectedness, simplicity, validity and applicability, but these terms were developed in the abstract, irrespective of any particular medium. In order

to have them evaluated, I first had to define them in terms of the musical composition context in which they were to be employed.

The concept of newness, I captured by asking judges to assess the inverse quality, which I have labelled *familiarity*. Judges were asked to evaluate fragments on a 5-point Likert scale, with “0 meaning quite unlike other music you’ve heard in your life. 4 meaning very familiar or derivative sounding. (This is not in comparison to the other samples, but to the ‘real’ music you know.)” Note that this quality was inverted (as was simplicity) in an effort to control for acquiescence bias [62], and to disguise the true aim of the evaluation.

The co-variant notion of unexpectedness seemed irrelevant in this particular experiment, since the samples were not produced by the judges themselves, so there could have been no preconceived expectation for the samples to contrast against. I therefore did not attempt to measure it.

Like newness, simplicity was also inverted, with judges being asked to gauge fragments for their musical complexity, again on a 5-point Likert scale, with “0 meaning it is very simple sounding. 4 meaning it sounds very complex.”

The co-variant of validity was characterized in this context as *musicality*, reasoning that for a musical idea to be creatively valid, it must conform to the judge’s sense of what is or is not musical. Again, judges were given a 5-point Likert scale, and asked “Is the piece following a musical plan? 0 meaning this has no identifiable musical organization. 4 meaning it is following an obvious musical plan.”

I interpreted applicability, in this situation, to mean that the samples were applicable to the purposes for which they were created, meaning that they offered potential as building blocks for bigger and better compositions. The 5-point Likert scale for this attribute was presented with the question, “Does this fragment suggest an idea for a more complete composition? 0 meaning this piece has nothing interesting in it to make it worth further development. 4 meaning that there is at least a portion of it that sounds interesting enough that it could be developed further, although this need not be true for the entire sample.”

With the four of the five co-variant analogs now adopted, one last quality was presented for the judges to score, in an effort to address the additional notion of density. In particular, I have interpreted idea density in the sense that a population comprised of nearly identical candidates has a lower idea density than does a population consisting of mutually distinct candidates. With that in mind, judges were asked to score a 5-point Likert scale for “Distinctness: (As compared to the other samples.) 0 meaning that many of the other samples are just like it. 4 meaning that it is completely unlike all the other samples.

Evaluation Process

While the samples were each created in their distinct representation schemes (WSNG for atomic samples and WSNT for the constructural candidates), they were presented to judges uniformly in MP3 format, after having been rendered through a MIDI sequencer. All the candidates utilized the same grand piano voice at the same base tempo, in which one unit of WSNG time equated approximately with 0.5 seconds of clock time.

The evaluations were conducted through an unsupervised web page to which judges had unrestricted access, although they were required to sign in each time they visited. In addition to verbal instructions from me, they were also presented with textual instructions and the above-quoted definitions of terms on each of the web pages in the test. Judges were encouraged to complete the series as quickly as practical, but given the duration of the complete, three-phase test (ranging from 2 to 4 hours, depending on the judge) they were permitted to break their time into smaller sessions, as they saw fit.

In this generation phase of the test, 12 compositions were presented from each of the two representation schemes, for a total of 24 samples, which were presented in a randomly mixed list. A screen shot of the web form used to evaluate each candidate is shown in Figure 6.3.

| <u>Play #0</u> <u>(0:19)</u> | 0 | 1 | 2 | 3 | 4 |
|---|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Familiarity: | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Complexity: | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Musicality: | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Potential: | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Distinctness: | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

Figure 6.3: Evaluation form for generation phase candidates

Judges were advised to familiarize themselves with the range of candidates by pre-auditioning the population before beginning the actual evaluations, in an effort to reduce any calibration effect that might otherwise have occurred. Evaluation consisted of playing the candidates one at a time (by clicking on the “Play” link) and scoring each one in all attributes before moving on to the next fragment. There was no attempt made to enforce this protocol, nor was any data collected to monitor adherence.

6.2.2 Results

A total of ten judges completed this phase of the assessment, ranging in age from thirteen to sixty-five years, and comprised of six males and four females. Three of the judges had extensive prior composing experience and three had no musical training or experience at all, but all the selected participants expressed a desire to be able to compose music of their own, had a strong self-declared

interest in music, and perceived themselves to be discerning in their tastes. (An eleventh candidate, who claimed to listen to all music, from all genres, with no specific preferences for style or quality, was rejected on the basis that this may reflect an impaired ability to discern among qualitative musical features.)

Although there was no detailed timing information captured as part of the experimental protocol, judges reported needing between 45 and 90 minutes to complete this phase of the assessment.

Note: In an attempt to minimize confusion, and to reflect these findings back to the principles I am trying to study, I have elected to use the language of the five co-variant attributes, as originally discussed in Section 3.2.2, rather than the language of the music-specific qualities that were presented to the judges. To this end, the judges’ scores for the two qualities that were inverted (newness/familiarity and simplicity/complexity) have been re-inverted, back to their original sense, by subtracting the score from 4. This aligns all five attributes, making them more directly comparable, with greater creativity or density indicated by higher-valued scores in each case.

Inter-rater Correlation

A comparison of how judges scored each candidate shows very little agreement. For each attribute, a mean correlation score was computed by averaging the pairwise correlation scores (Pearson’s product-moment correlation) computed between each pair of judges. These aggregates are shown in Table 6.2. In four of the five attributes, the \bar{p} values are not significant at the customary 0.05 level, so we must conclude that in those cases, any agreement found can be explained by chance. The only attribute for which there was any significant correlation ($\bar{p} = 0.041$) was the newness attribute, but even there, the correlation was not particularly strong, with an \bar{r} value of 0.515. Clearly, the judges do not concur on which samples are “good.”

| Attribute | Pearson’s \bar{r} | \bar{p} |
|--------------|---------------------|-----------|
| Newness | 0.515 | 0.041 |
| Simplicity | 0.178 | 0.172 |
| Validity | 0.433 | 0.165 |
| Appliability | 0.304 | 0.219 |
| Distinctness | 0.250 | 0.254 |

Table 6.2: Inter-judge score correlations per candidate for generation phase

In hindsight, however, agreement among the judges on a sample-by-sample basis should not have been expected. The judges had no history of conducting this type of assessment, nor were they trained against a reference data set to harmonize and calibrate their mutual understanding of the attributes. This can only have been exacerbated by the wide variation in their musical backgrounds.

Ultimately, though, their failure to correlate on an individual sample level seems unimportant, in light of their strong correlation at the aggregate level. When we examine the implied preference for one encoding scheme over the other, as indicated by the distributions of high scores, the judges were in near total agreement.

Inter-scheme Distribution

The primary concern of this experiment is to assess whether or not there is a measurable difference in creative potential between candidates produced in one scheme vs. the other. In other words, we are looking for a detectable difference in subjective score distributions between the atomic and structural partitions of the fragment population.

As we saw in the previous section, comparing attribute scores on a fragment by fragment basis tells us little, since discrepancies between the judges' subjective impressions are amplified in that context. But in the aggregate, we can hope to see consistent trends differentiating the two classes.

To investigate this, the judges' scores were tallied into a separate contingency table for each attribute, showing the breakdown of how many times each of the five possible scores (0 through 4) was assigned to an atomic fragment versus a structural fragment, from each of the judges. With 10 judges and 12 samples in each of the two scheme-classes, each attribute was therefore assessed 240 times.

In this aggregated form, the data shows consistent trends, demonstrating a marked increase in the scores assigned to the structural fragments, as compared to the atomic samples. In the case of all but one of the attributes, the mean score rose by more than 1.0 on the five-point scale. The *complexity* rose in average score by 0.40.

Tables 6.3 through 6.7 show contingency tables for each attribute, along with an accompanying histogram¹ of the score distributions for that attribute. Results for all five attributes were highly significant, with Pearson's χ^2 producing p values in the range $[6.2 \times 10^{-18}, 2.16 \times 10^{-5}]$.

| | 0 | 1 | 2 | 3 | 4 | \bar{n} | σ | Skew |
|-----------|----|----|----|----|----|-----------|----------|-------|
| atomic | 1 | 2 | 10 | 60 | 47 | 3.25 | 0.75 | -1.15 |
| construct | 18 | 35 | 29 | 20 | 18 | 1.88 | 1.29 | 0.23 |

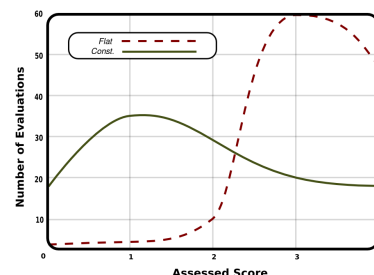


Table 6.3: Distribution of *newness* score for generated fragments, by scheme

¹The histograms are reproduced with Bezier splines instead of the traditional stair-step display, as this was found to minimize confusion of overlapping lines and made the trends more readily apparent.

Again the newness attribute stands somewhat apart from the others. Recall that in the previous section, we saw that the judges exhibited some degree of correlation regarding how new or unfamiliar the atomic candidates seemed. This level of correlation can be seen again here. As Table 6.3 shows, newness scores for the atomic candidates were very high, with a mean score, \bar{n} , of 3.25 ($p = 6.18 \times 10^{-18}$) and demonstrated a comparatively low variance ($\sigma_a = 0.75, \sigma_c = 1.29$), with fully 89.1% of atomic samples ranked highly ($n > 2$), and the most extreme skewness score (-1.15) of all the criteria assessed in this experiment.

Conversely, the constructural population was not seen as particularly new, having a \bar{n} of only 1.88, a relatively wide distribution with $\sigma_c = 1.29$, and a much less extreme skew of 0.23.

This consistency of the judges to view the atomic samples as new-sounding warrants comment. Recall that newness is seen as an important co-variant of creative merit, which, superficially, might be seen as an indication of higher creative merit. The problem with this interpretation is that, while newness may be a necessary condition of creative value, I have seen no claims in the literature that it is sufficient. I suspect that within the set of all possible compositions that are unfamiliar to a given audience, the bad compositions significantly outnumber the good ones. If this is true, then the number of false positives we could expect from a newness sieve would be very high – a point that seems to be supported by the data. While there were 107 cases in which an atomic sample was scored highly for newness, there were only 10 cases in which atomics were scored highly for both newness and their creative potential (applicability). Performance of the sieve in the constructural class was somewhat better, but still dominated by false positives, with 38 samples scored well for newness, of which only 9 were also highly scored for potential. With these rates of false positives, newness by itself seems a bad predictor for creative merit.

| | 0 | 1 | 2 | 3 | 4 | \bar{s} | σ | Skew |
|-----------|---|----|----|----|----|-----------|----------|-------|
| atomic | 6 | 37 | 30 | 27 | 20 | 2.15 | 1.18 | 0.14 |
| construct | 9 | 9 | 27 | 54 | 21 | 2.58 | 1.10 | -0.81 |

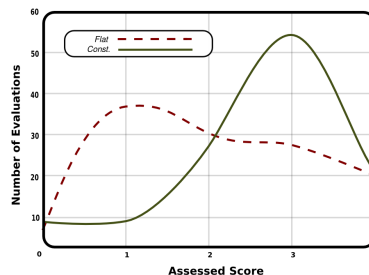


Table 6.4: Distribution of *simplicity* score for generated fragments, by scheme

A comparison of the standard statistical measures of mean and deviation for the simplicity attribute does not reveal much difference between the atomic and constructural populations. With $\bar{s}_a = 2.15$ and $\bar{s}_c = 2.58$, there is a slight edge given to the constructural examples, as a whole. The skewness score, however, shows a slightly different story, with atomic scores biased toward the low end of the scale and constructurals biased toward the higher end.

This distinction becomes most evident when we look at only those candidates that scored highly

for simplicity ($s > 2$). Here we see that the atomic group contributed 47 such examples, while the constructural population produced 75—almost 60% more frequently.

While this demonstrates a marked improvement in the performance of the constructural class over the atomics, assessing the power of this attribute as a predictor, we find that, like newness, simplicity does not appear to be a particularly important metric. Of the 47 cases in which an atomic candidate was found to be simple, there were only 2 cases in which it was also found to have creative potential. The constructural samples also showed a substantial, although less extreme imbalance, with 75 fragments highly scored for simplicity, from which only 18 were also rated highly for creative potential.

Like newness then, simplicity seems to be a poor predictor of creative merit, so while the constructural samples produced a substantially greater number of simple candidates, the increase in simplicity does not seem strongly enough coupled with an increase in creative merit to make it an important improvement.

| | 0 | 1 | 2 | 3 | 4 | \bar{v} | σ | Skew |
|-----------|----|----|----|----|----|-----------|----------|-------|
| atomic | 30 | 57 | 23 | 8 | 2 | 1.12 | 0.92 | 0.78 |
| construct | 9 | 14 | 43 | 29 | 25 | 2.39 | 1.16 | -0.28 |

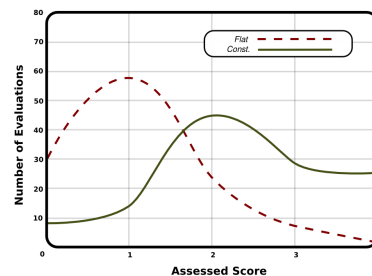


Table 6.5: Distribution of *validity* score for generated fragments, by scheme

While the previous two attributes (newness and simplicity) were found wanting in terms of their predictive power, this trend does not hold for the rest. As a predictor, validity seems much more powerful than either of the previous attributes examined. In the atomic population, 8 of the 10 highly valid examples were also highly-scored for potential, and from the constructural group, 34 of 54 were so rated. With a combined predictive success rate of 66%, it would seem that validity bears a strong association with overall subjective creative merit.

This makes validity an important dimension for us to improve if we wish to facilitate creativity, and this indeed appears to have been achieved with constructural representations. Table 6.5 shows that, while the atomic scheme produced 10 highly valid candidates, the constructural population produced more than five times as many, with 54 cases. The mean score assigned to the constructurals was more than 1.25 points higher on the five-point scale.

Like validity, distinctness is a relatively strong predictor, with more than half of the highly distinct candidates also having been scored highly for potential (4 of 6 atomics and 32 of 54 constructurals).

This comes as something of a surprise, in that distinctness was included as a measure of the

| | 0 | 1 | 2 | 3 | 4 | \bar{d} | σ | Skew |
|-----------|----|----|----|----|----|-----------|----------|-------|
| atomic | 22 | 61 | 31 | 5 | 1 | 1.18 | 0.81 | 0.51 |
| construct | 11 | 15 | 40 | 43 | 11 | 2.23 | 1.08 | -0.47 |

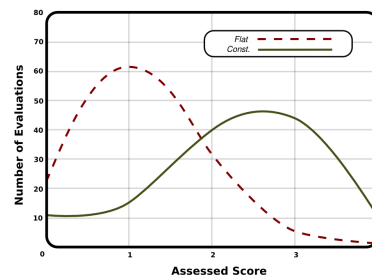


Table 6.6: Distribution of *distinctness* score for generated fragments, by scheme

diversity, rather than the quality of the samples. Finding this connection between the two suggests that the judges may have subconsciously prejudiced their evaluation of distinctness with their aesthetic judgement, but further investigation will have to be conducted to shed light on this connection.

Table 6.6 shows the contingency and histogram for distinctness scores, and once again, we see that structural examples substantially outperform atomics, with 54 highly scored examples, as compared to only 6, respectively. Again we see the mean scores for the structurals more than a full point higher than the atomics on the five-point scale. This tells us that there is measurably greater variety among the samples produced, which is an important component of creative diversity, since if more candidates are perceived as distinct from their peers, then the density of ideas they represent must also be higher.

| | 0 | 1 | 2 | 3 | 4 | \bar{a} | σ | Skew |
|-----------|----|----|----|----|----|-----------|----------|-------|
| atomic | 34 | 57 | 18 | 6 | 5 | 1.09 | 1.00 | 1.10 |
| construct | 12 | 24 | 37 | 27 | 20 | 2.16 | 1.22 | -0.08 |

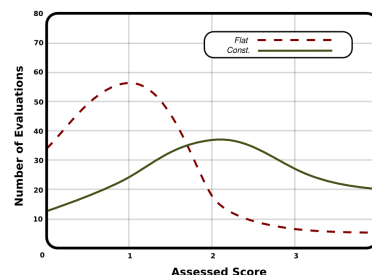


Table 6.7: Distribution of *applicability* score for generated fragments, by scheme

This leaves the most direct measure for last – the measure of creative potential. While previous attributes were examined for their power to predict creative merit, by asking judges to score samples for their potential to be developed further, into more complete compositions, this attribute comes the closest to being a direct assessment of p-creative merit.

In this regard, the structural examples still perform measurably higher than the atomic set. With an improvement of more than 1.0 for the mean score, and producing 47 high-scoring examples, as compared to 11 from the atomics, structurally representation seems significantly better suited to producing creatively engaging musical fragments at this early stage of creative development.

It is still possible, however, that the candidates being scored highly in one attribute are not

the same as those being scored highly in other attributes, and while it is encouraging to see these improved rankings for individual attributes, it is most important, in terms of facilitating creative exploration, that individual candidates perform well in multiple attributes, ideally in all of them.

When I combine the three important indicators, I find 4 cases in which a judge scored a single candidate highly for all three attributes, as compared to 23 such incidents in the constructural group. Or, stated another way, in a trial in which each of 10 judges examined 12 musical candidates, on average, only four of them found a single example worth pursuing when they were working with atomic samples, but when they were working with constructural examples, each of them found 2.

By increasing the measured subjective creative merit of musical fragments produced, as well as their diversity, it seems that constructural representations did indeed offer greater creative richness than did the atomic representations, in this experiment.

6.3 Variation Phase

The second phase of the experiment focuses on approximating another very common compositional process: deriving new works as a series of variations upon a pre-existing work. Instead of exploring the addressable music space around the null composition, this exploration shifts to examine the space proximate to some known work—in effect, gauging the potential success of leaping from one plausible variation and finding another, related to the first, but with its own appeal.

6.3.1 Experiment Design

To assess the relative ability of the two encoding schemes to facilitate this type of exploration, I examined the creative density of the two schemes in this specific context, reasoning that if the density and quality of ideas immediately proximate to an existing composition is higher, then users of such a system are more likely to find better ideas, more often in the course of their experimentation.

This scenario was modelled by first choosing a pool of five seed composition fragments, which I refer to as *archetypes*. From each archetype, four variations were generated in each of the two representation schemes for judges to evaluate, yielding eight variations per seed, and forty variations in total.

The archetypes were chosen for their stylistic variety, and relative brevity, to facilitate exploring a good mix of genres and styles while also keeping the judging process relatively short. The archetype fragments used were taken from *Danger Baby*, *Invention Sketch*, *Clockhouse*, the prelude from Bach's *Cello Suite #1*, and a blues fragment entitled *Slippery Blue Ice*.²

²Refer to “*Slippery Blue Ice*.” in the *Wheelfragments* folder of the companion CD.

Archetype fragments ranged in duration from 16 to 20 seconds and the population of candidate variations produced ranged from 8 to 117 seconds.

Variation Method

Varying a composition was defined as the application of value changes to the parameters of its encoding. Thus, the effort measure remains unchanged between an archetype and its family of variants, maintaining their homogeneous comparability.

For atomic encodings, a variation was defined to be a modification made to either the pitch or duration of any one note event. For structural encodings, modifications were permitted to either pitch or duration values in any of the literal melodic fragments encoded within them, or to any of the numeric parameters on the transform operators in the tree. Modification of the operator nodes themselves, by replacing one operator with another, was not permitted, since this style of more aggressive modification was conducted as part of the hybridization phase.

Since I wish to be able to aggregate the scores across all archetype variations, to assess the behaviour of the two schemes in the aggregate, it becomes important to ensure some degree of homogeneity to the effects produced by the variations in each archetype family, despite differences between them in their effort/complexity. In particular, to compare a very small, low-effort encoding to a larger, more complex example, it seemed inequitable to apply the same, fixed number of variations in both cases, as this would likely produce more extreme variations of the smaller archetypes and comparatively more minor variation on the more elaborate sources. To counter this, the number of variations applied to each seed fragment was calculated as a proportion of the size of the seed's encoding. For each seed, n variations were applied at random, where n was equal to 10% of the number of alterable parameter values present in the encoding, rounded to the nearest integer. So, for example, with a seed encoding that contained 16 modifiable parameters, 2 variations would be applied, whereas a candidate with 385 alterable parameters would be modified in 39 of them. Target parameters were selected randomly, with replacement, so the effective number of modifications applied in the production of each variation was at most n .

In hindsight, the decision to limit modifications to only the numeric parameters may have been overly restrictive, since, for structural candidates, the choice of operator assigned at each node of the graph is also a parameter to the representation. While this numeric-only policy can more readily be seen as equitable, in reality, I fear that it privileged the atomic class, since the structural examples were constrained from demonstrating the full range of their variability. It remains to be seen, however, to what degree that restriction compromised the overall performance of the structural population.

Constraints

The parameters being varied in this experiment were all numeric. Each variation step consisted of selecting an old value, v_{orig} , somewhere in the encoding, and replacing it with a new value, v_{new} , where $\frac{v_{orig}}{2} < v_{new} < 2v_{orig}$. In the cases where the string encoding of v_{orig} contained a decimal point, v_{orig} was assumed to be a floating point value, and v_{new} was left in float form. Where v_{orig} did not contain a decimal point, it was assumed to be an integer, and v_{new} was rounded to the nearest integer.

By using these string-based variation algorithms, it was possible to employ the same code in varying both WSNB and WSNT files, which eliminated a potential source of accidental bias between the two generated populations.

Fragment Creation

These above constraints and considerations were then used to guide development of a single utility, *wsntdeviate*, shown in pseudo-code form in Figure 6.4. Since all changeable numeric parameters expressed in either WSNB or WSNT files can be identified by the same set of regular expressions, the same code can be used to process either file type.

```
BEGIN
  sourceFragment = LoadFile(sourceFileName)
  numParameters = countChangeableParameters(sourceFragment)
  numChanges = numParameters/10;

  FOR changeCount FROM 0 TO numChanges
    targetParameterID = randomIntegerBetween(0, numParameters)
    oldValue = getParameterValueByID(targetParameterID, sourceFragment)

    deltaFactor = randomFloatBetween(0.5, 2.0)
    newValue = deltaFactor * oldValue
    IF oldValue.contains('.')
      newValue = roundToNearestInteger(newValue)
    setParameterValueByID(targetParameterID, newValue, sourceFragment)

  PRINT sourceFragment
END
```

Figure 6.4: Pseudo-code of variation algorithm

Metrics

As in the generation phase, this phase explored the creative richness of randomly constructed musical fragments, in the specific context of being able to use those fragments to inspire and develop more complex creative works. Again, there were two issues to be evaluated: the creative merits of the musical candidates in their own right, and the density with which those various attributes occurred in the population.

Repeating the practice used in the generation phase, the density issue was probed by asking judges to evaluate the distinctness of the samples, but whereas distinctness in the generation phase was assessed with respect to the entire population, the distinctness of these variations was assessed within family groupings only. Judges were asked to score a given fragment for its distinctness from the archetype upon which it was based, and from all the other variations created from that same source.

With almost twice as many samples to evaluate in this phase, I was mindful of trying to limit the time burden placed on judges, so I elected to capture fewer of the creativity attributes. Of the five attributes—newness, simplicity, unexpectedness, validity and applicability—I still felt that unexpectedness was inappropriate to the context of this trial, since the judges were being presented with these fragments as *fait accompli*, and therefore had no prior expectation against which to measure the result. Of the remaining four, I reasoned that validity and applicability were the most crucial indicators, as they seem to sum up the user’s overall impressions of the candidate.

Since newness had been presented in the previous experiment as being compared to all the music the judge was familiar with, I feared that in this experiment, the fact that candidates were derived from, and likely still similar to, existing works, would make it difficult for them to assess newness in any meaningful way. I therefore elected to omit newness from the evaluations.

Furthermore, I felt it would be similarly difficult to evaluate simplicity/complexity in the context of an existing, non-trivial composition — especially for judges who had little or no composing experience. By comparison, the fragments produced in the generation phase were much simpler and shorter than those in this phase, and their relative complexity was fairly easy to judge. Ultimately, I felt that insufficient value was likely to be gleaned from this attribute as well, so it was also omitted.

In light of the fact that these two dropped attributes (newness and complexity) were subsequently shown to have poor predictive power, I do not believe that their omission from this phase of the experiment was detrimental.

This left the judges with only three criteria to assess: validity, applicability, and distinctness, which were presented in the guise of musicality, potential and distinctness, to maintain consistency with the previous phase. Assessments were collected via the same 5-point Likert scale.

Judges were given the following advice on how to interpret the criteria, and the form shown in Figure 6.5 illustrates the manner in which data was collected for each of the family groupings in the population.

Regarding Musicality: *In general the variations are approximately as musical as the archetypes, so please focus your evaluation on the musicality of the changes from the original.*

Regarding Potential: *a sample that differs from the archetype in a particularly interesting way that might be worth exploring further should score higher, although the ‘good part’ need not run*

| Archetype 0: Play (0:20) | | | | | | | | | | | | | | | | | | | | | | | |
|-----------------------------------|---|---|---|---|---|-----------------------------------|---|---|---|---|---|-----------------------------------|---|---|---|---|---|-----------------------------------|---|---|---|---|---|
| Play #0 (0:35) | 0 | 1 | 2 | 3 | 4 | Play #2 (0:20) | 0 | 1 | 2 | 3 | 4 | Play #4 (0:20) | 0 | 1 | 2 | 3 | 4 | Play #6 (0:20) | 0 | 1 | 2 | 3 | 4 |
| Musicality | ○ | ○ | ○ | ○ | ○ | Musicality | ○ | ○ | ○ | ○ | ○ | Musicality | ○ | ○ | ○ | ○ | ○ | Musicality | ○ | ○ | ○ | ○ | ○ |
| Potential | ○ | ○ | ○ | ○ | ○ | Potential | ○ | ○ | ○ | ○ | ○ | Potential | ○ | ○ | ○ | ○ | ○ | Potential | ○ | ○ | ○ | ○ | ○ |
| Distinctness | ○ | ○ | ○ | ○ | ○ | Distinctness | ○ | ○ | ○ | ○ | ○ | Distinctness | ○ | ○ | ○ | ○ | ○ | Distinctness | ○ | ○ | ○ | ○ | ○ |
| Play #1 (0:32) | 0 | 1 | 2 | 3 | 4 | Play #3 (0:20) | 0 | 1 | 2 | 3 | 4 | Play #5 (0:20) | 0 | 1 | 2 | 3 | 4 | Play #7 (0:20) | 0 | 1 | 2 | 3 | 4 |
| Musicality | ○ | ○ | ○ | ○ | ○ | Musicality | ○ | ○ | ○ | ○ | ○ | Musicality | ○ | ○ | ○ | ○ | ○ | Musicality | ○ | ○ | ○ | ○ | ○ |
| Potential | ○ | ○ | ○ | ○ | ○ | Potential | ○ | ○ | ○ | ○ | ○ | Potential | ○ | ○ | ○ | ○ | ○ | Potential | ○ | ○ | ○ | ○ | ○ |
| Distinctness | ○ | ○ | ○ | ○ | ○ | Distinctness | ○ | ○ | ○ | ○ | ○ | Distinctness | ○ | ○ | ○ | ○ | ○ | Distinctness | ○ | ○ | ○ | ○ | ○ |

Figure 6.5: Evaluation form for variation candidates

through the entire sample.

Regarding Distinctness: samples should only be compared to other samples in the same groups, including the archetype, but definitely not to the samples in other groups.

Evaluation Process

As was the case in the generation phase, the two classes of samples (atomic versus structural) were interleaved at random, and presented to the judges in MP3 format, so that there was no way for them to infer which sample was produced via which method, nor were they informed as to the dual-class nature of the population.

Again, judges were given unsupervised access to the web evaluation form, were allowed to break their time into convenient sessions, were required to sign in at the beginning of each such session, and were asked to pre-audition the candidates in a given family before assessing them.

6.3.2 Results

This assessment was conducted with the same ten judges who participated in the previous phase, although two were unable to complete the trial, citing a lack of available time, so their partial contributions were discarded from this phase, leaving eight judges, ranging in age from sixteen to sixty-four years, and distributed as five males and three females. Three of the judges have extensive prior composing experience and three have no musical training or experience at all.

Although there was no detailed timing information captured as part of the experimental protocol, judges reported requiring between 45 and 120 minutes to complete this phase of the assessment, in the aggregate, and several judges reported that they spread their assessments out over a period of several days.

Inter-scheme Correlation

As with the previous experiment, the judges exhibited no consistent agreement on a fragment by fragment basis, though this is of little concern since our interest continues to lie in the aggregated

case.

For each attribute, the scores assigned by all judges were tallied into a contingency table showing the breakdown of how many times each of the five possible scores was assigned to an atomic fragment versus a constructural fragment. While the differences between the two populations are less pronounced than they were in the generation phase, there are still obvious and statistically significant distinctions.

Tables 6.8 through 6.10 show contingency tables for each attribute, along with accompanying graphs depicting the histograms of the score distributions. Results for all three attributes were highly significant, with Pearson's χ^2 producing p values in the range $[3.1 \times 10^{-13}, 1.6 \times 10^{-4}]$.

| | 0 | 1 | 2 | 3 | 4 | \bar{v} | σ | Skew |
|-----------|----|----|----|----|----|-----------|----------|-------|
| atomic | 5 | 46 | 58 | 41 | 10 | 2.03 | 0.96 | 0.15 |
| construct | 28 | 28 | 40 | 50 | 14 | 1.96 | 1.24 | -0.20 |

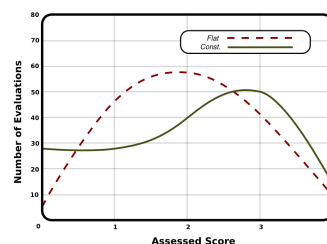


Table 6.8: Distribution of *validity* score for variations, by scheme

Scores for validity, as seen in Table 6.8, show very similar means, with atomic candidates scoring slightly higher than their constructural counterparts, although the scores for constructural fragments are negatively skewed in comparison. Regarding high-score performance, atomic candidates were assessed highly in 51 cases, while 64 cases were so designated from among the constructural group, which, while not as pronounced as some of the comparisons observed in the generation phase, is still a respectable 25% increase in the number of candidates judged as promisingly valid.

| | 0 | 1 | 2 | 3 | 4 | \bar{a} | σ | Skew |
|-----------|----|----|----|----|----|-----------|----------|-------|
| atomic | 15 | 64 | 47 | 22 | 12 | 1.7 | 1.06 | 0.52 |
| construct | 30 | 30 | 45 | 38 | 17 | 1.89 | 1.26 | -0.03 |

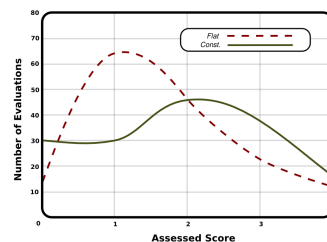


Table 6.9: Distribution of *applicability* score for variations, by scheme

As with validity, the mean scores for applicability are again similar, with constructural candidates having the slight edge in this case. (See Table 6.9.) And once again, the more negative skew within the constructural fragments results in a higher proportion of high-scoring candidates from that group: 55 as compared to only 33 atomic candidates.

In the dimension of distinctness (see Table 6.10) there is a stronger contrast between the mean scores, with constructural candidates scoring well more than half a point higher, on average, and again, the more negative skew of the scores produces a strong showing in the number of high-scoring

| | 0 | 1 | 2 | 3 | 4 | \bar{d} | σ | Skew |
|-----------|----|----|----|----|----|-----------|----------|-------|
| atomic | 9 | 82 | 53 | 12 | 4 | 1.50 | 0.82 | 0.79 |
| construct | 15 | 24 | 54 | 49 | 18 | 2.19 | 1.12 | -0.30 |

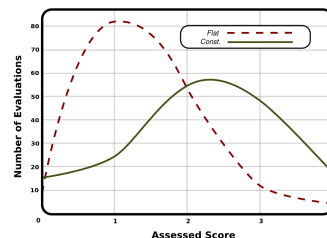


Table 6.10: Distribution of *distinctness* score for variations, by scheme

candidates produced: 67 highly distinct structural candidates as opposed to only 16 distinctive atomic fragments.

As with the generation phase, validity seems to remain a good predictor. Of the 115 times a sample was scored highly for validity, 74 of the cases were also judged highly for creative potential.

Distinctness also maintained its apparent but weaker value as a predictor, with 38 high potential candidates arising from the pool of 83 highly distinct samples.

Comparing multi-attribute high scores, as we did in the previous experiment, we find 10 cases of atomic candidates being scored highly for all three attributes of validity, applicability and distinctness, as compared to 22 such triple-crown samples from the structural population.

In summation, while the distinctions are less marked, there is still a clear increase in the number of structural candidates ranked highly by the judges, in comparison with the atomic candidates.

6.4 Hybridization Phase

The third and final phase of this experiment was designed to explore the last of three approaches to composition, this time based on juxtaposing and combining two contrasting ideas, a process sometimes referred to as a *mash-up*. This is a less common approach to composition than the other two, but has recently gained attention in popular culture, and it provides an opportunity to explore an approach to the creative richness of juxtaposition suggested by Csikszentmihalyi.

6.4.1 Experiment Design

To simulate this process, candidates were created by inserting randomly chosen content from one archetype encoding into a randomly selected location in a second, while maintaining syntactic validity.

Fragment Creation

Compositions were selected in random pairs from a pool of seed compositions and then combined into new hybrid candidates. The set of seed compositions was the same as that used in the variations phase, with the addition of a sixth candidate to offer more permutations for hybrid candidate pairs.

The sixth archetype used was one of the more interesting candidates spawned during the generation phase, and is entitled *Mysterious Fluid*.³ Contributing pairs for each candidate were selected without replacement, to ensure that each hybrid was generated from two distinct archetypes.

$$\begin{aligned}
 A &= a_1 a_2 a_3 a_4 \sqrt{a_5 a_6 a_7} a_8 a_9 \\
 B &= b_1 b_2 \underline{b_3 b_4 b_5 b_6 b_7 b_8} b_9 b_{10} b_{11} b_{12} b_{13} b_{14} \\
 \text{New} &= a_1 a_2 a_3 a_4 \underline{b_3 b_4 b_5 b_6 b_7 b_8} a_8 a_9
 \end{aligned}$$

Figure 6.6: String crossover

Hybridization was carried out using the crossover techniques employed in genetic algorithm research [45]. For atomic samples, candidates were created using string crossover, as illustrated in Figure 6.6. A randomly chosen substring in one archetype was replaced by a randomly chosen substring from the second archetype.

For structural samples, tree crossover was used, illustrated in Figure 6.7, in which a randomly chosen node and its dependant sub-tree, from one archetype was replaced with a randomly chosen node and dependant sub-tree from a second archetype.

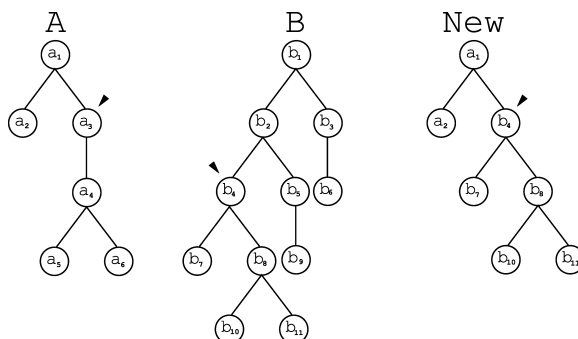


Figure 6.7: Tree crossover

A total of 24 hybrid candidates were created, twelve by atomic representation methods using string crossover, and twelve by hierarchical sub-tree crossover for the structural candidates. These were presented to the judges in randomly interleaved fashion, with no indication made of which representation scheme was used to produce which candidate.

Metrics

The same three qualities were assessed in this phase as were judged in the variation phase, and the same descriptions of each attribute were given, with the exception of distinctness, which was described as follows.

³Refer to “*Mysterious Fluid*.” in the *Wheelfragments* folder of the companion CD.

Distinctness: How clearly does the sample differ from the archetypes? 0 meaning not at all, and it is easy to identify which archetypes were used in the mashup. 4 meaning very different and you cannot identify the original archetypes.

| | | |
|-----------------------------|-----------------------------|-----------------------------|
| Archetype 0: Play (0:17) | Archetype 1: Play (1:55) | Archetype 2: Play (0:09) |
| Archetype 3: Play (0:18) | Archetype 4: Play (0:19) | Archetype 5: Play (0:20) |
| Play #1 (1:36) | Play #2 (0:27) | Play #3 (0:05) |
| Musicality | Musicality | Musicality |
| Potential | Potential | Potential |
| Distinctness | Distinctness | Distinctness |

Figure 6.8: Evaluation form for hybridization candidates

Figure 6.8 shows a detail from the form used to capture the evaluations, which, in full, presented 24 candidate evaluation panels rather than the 3 shown here.

6.4.2 Results

This assessment was conducted with the same ten judges as were used in the previous phases, with the same two judges withdrawing for time reasons, leaving a total of eight judges. Although there was no detailed timing information captured as part of the experimental protocol, judges reported requiring between 30 and 60 minutes to complete this phase of the assessment.

Inter-scheme Correlation

While the atomic candidates show the expected, quasi-normal distribution for all three attributes, the constructural population shows evidence of bimodal partitioning in all three attributes. This suggests that either the constructural fragments are divided into two distinct qualitative categories, one scoring highly, the other scoring weakly in each attribute, or else the field of judges are divided into two camps as to how to interpret the constructural candidates. Attempts to distinguish between these two explanations were inconclusive, leading me to believe that both sources played a role in producing the bimodality.

| | 0 | 1 | 2 | 3 | 4 | \bar{v} | σ | Skew |
|---------------|----|----|----|----|---|-----------|----------|-------|
| atomic | 9 | 27 | 34 | 24 | 2 | 1.82 | 0.98 | -0.11 |
| constructural | 16 | 23 | 20 | 31 | 6 | 1.88 | 1.22 | -0.11 |

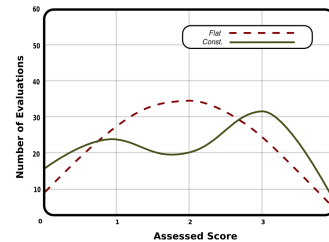


Table 6.11: Distribution of *validity* score for hybrids, by scheme

Table 6.11 shows validity scores for constructural fragments to be somewhat higher than those for the atomic group. While the mean score and skewness are virtually the same for both groups, the larger variance of the constructural scores reflects more candidates in the outlier positions, and when we compare the rates at which the two populations produced high-scoring candidates, we find 37 candidates from the constructural group versus 26 from the atomics.

| | 0 | 1 | 2 | 3 | 4 | \bar{a} | σ | Skew |
|---------------|----|----|----|----|----|-----------|----------|------|
| atomic | 11 | 35 | 30 | 12 | 8 | 1.7 | 1.10 | 0.47 |
| constructural | 14 | 35 | 20 | 16 | 11 | 1.74 | 1.23 | 0.40 |

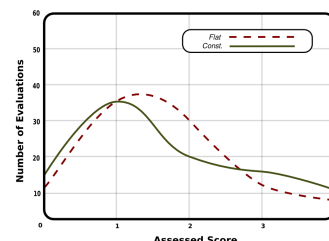


Table 6.12: Distribution of *applicability* score for hybrids, by scheme

With applicability, as illustrated in Table 6.12, the story is much the same. Very similar mean scores and skewnesses, combined with a slight difference in variance produce slightly more high-scoring constructural candidates than atomic, by a rate of 27 to 20.

| | 0 | 1 | 2 | 3 | 4 | \bar{d} | σ | Skew |
|---------------|----|----|----|----|----|-----------|----------|------|
| atomic | 11 | 42 | 35 | 5 | 3 | 1.45 | 0.88 | 0.57 |
| constructural | 9 | 28 | 22 | 14 | 23 | 2.15 | 1.33 | 0.11 |

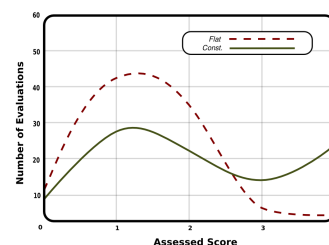


Table 6.13: Distribution of *distinctness* score for hybrids, by scheme

Table 6.13 shows greater differentiation between the two populations for distinctness scores, with constructural having the higher mean score and a more favourable skew. Comparing high-score production, the constructurals come out ahead with 37 as compared to only 8 distinctly-scored atomic candidates.

If we consider the production rate of high-score values, we see that, considering all three attributes, atomic candidates were merited highly 54 times, as compared to the 101 high scores given to the constructural population, and when we compare how often a single candidate was ranked highly for all three criteria, there were six such fragments from the constructural population and only 1 from the atomics.

6.5 External Validation

More recently, an unexpected opportunity arose to solicit external validation of my results when I discovered composition-contest.com. This web site is dedicated to providing critical feedback to

serious amateur composers and, as part of that mandate, conducts a monthly themed competition to which composers can submit their work and receive expert critiques, so I submitted one of the Wheelsong output compositions to see how it would compare against other works, created traditionally by other composers.

In addition to the general competition, the contest also states a thematic target, and awards secondary prizes for the work that is judged to most closely embody the stated theme. Rather than choose the best piece from the population of Wheelsong outputs, I elected instead to choose the one that most closely matched the stated theme, reasoning that this would at least partially mitigate any preferential selection bias on my part. The stated theme for that month (April 2010) was “clockwork,” so the piece I submitted was *Clockhouse*.

| (a) April 2010 | | (b) July 2010 | |
|----------------|-------------------|-------------------|-------------|
| Submission | Overall | Submission | Creativity |
| 24 | 8.75 | 39 | 9.00 |
| 36 | 8.69 | Clockhouse | 8.50 |
| 31 | 8.38 | 24 | 8.49 |
| 21 | 8.26 | 25 | 8.29 |
| 17 | 8.01 | 26 | 8.02 |
| 25 | 8.00 | 36 | 7.99 |
| 29 | 7.91 | 37 | 7.85 |
| 37 | 7.84 | 31 | 7.83 |
| 7 | 7.73 | 21 | 7.78 |
| 9 | 7.69 | 17 | 7.74 |
| 6 | 7.67 | 3 | 7.70 |
| 22 | 7.64 | 5 | 7.65 |
| 3 | 7.55 | 9 | 7.64 |
| 5 | 7.50 | 6 | 7.62 |
| 20 | 7.45 | 16 | 7.54 |
| 26 | 7.45 | 15 | 7.52 |
| 23 | 7.41 | 32 | 7.44 |
| 33 | 7.35 | 22 | 7.41 |
| 39 | 7.35 | 19 | 7.40 |
| 27 | 7.30 | 35 | 7.39 |
| 16 | 7.27 | 4 | 7.30 |
| 19 | 7.20 | 27 | 7.30 |
| 8 | 7.17 | 7 | 7.22 |
| 28 | 7.13 | 23 | 7.15 |
| 32 | 7.00 | 2 | 7.02 |
| 35 | 7.00 | 8 | 7.01 |
| 40 | 6.98 | 13 | 7.01 |
| 28/41 | Clockhouse | 14 | 7.01 |
| 15 | 6.95 | 20 | 6.93 |
| 2 | 6.87 | 28 | 6.92 |
| 34 | 6.80 | 12 | 6.88 |
| 4 | 6.73 | 33 | 6.82 |
| 38 | 6.60 | 29 | 6.81 |
| 30 | 6.59 | 34 | 6.71 |
| 10 | 6.48 | 40 | 6.61 |
| 18 | 6.41 | 38 | 6.55 |
| 14 | 6.32 | 10 | 6.47 |
| 13 | 6.27 | 11 | 6.34 |
| 1 | 6.25 | 18 | 6.21 |
| 11 | 6.16 | 30 | 6.17 |
| 12 | 6.13 | 1 | 5.77 |

| Submission | Overall | Submission | Creativity |
|--------------|---------------|---------------|-------------|
| 24 | 8.52 | 24 | 8.24 |
| 11 | 8.46 | 15 | 8.00 |
| 15 | 8.13 | 11 | 7.74 |
| 13 | 8.07 | 28 | 7.56 |
| 23 | 8.00 | 32 | 7.49 |
| 7 | 7.78 | 9 | 7.43 |
| 29 | 7.75 | 27 | 7.42 |
| 6 | 7.71 | 18 | 7.36 |
| 9 | 7.71 | 5 | 7.35 |
| 28 | 7.68 | 29 | 7.32 |
| 1 | 7.58 | 36 | 7.32 |
| 26 | 7.58 | 13 | 7.26 |
| 18 | 7.53 | 7 | 7.25 |
| 10 | 7.52 | 1 | 7.22 |
| 20 | 7.44 | 6 | 7.17 |
| 34 | 7.41 | 34 | 7.17 |
| 25 | 7.33 | 26 | 7.15 |
| 31 | 7.28 | 10 | 7.14 |
| 4 | 7.26 | 31 | 7.13 |
| 32 | 7.25 | 35 | 7.13 |
| 22 | 7.20 | 20 | 7.12 |
| 36 | 7.17 | 25 | 7.12 |
| 27 | 7.16 | 19 | 7.11 |
| 35 | 7.16 | 4 | 6.97 |
| 12 | 7.02 | 23 | 6.96 |
| 30 | 6.99 | 14 | 6.78 |
| 5 | 6.95 | 12 | 6.75 |
| 28/37 | Lament | 16 | 6.75 |
| 19 | 6.74 | Lament | 6.72 |
| 16 | 6.69 | 22 | 6.71 |
| 3 | 6.64 | 30 | 6.71 |
| 14 | 6.54 | 3 | 6.60 |
| 2 | 6.26 | 2 | 6.20 |
| 33 | 6.04 | 17 | 5.96 |
| 37 | 4.76 | 33 | 5.95 |
| 17 | 4.25 | 37 | 5.37 |
| 21 | 3.90 | 21 | 4.26 |

Table 6.14: Composition contest results

Judging of the competition was doubly blind. Judges had no information about the work, other than its name, and composers had no information about who was judging. 42 submissions were evaluated by 10 judges, all of whom are serious composers and are described (by the managers of the web site) as experts, although I have no independent corroboration of their credentials.

Of the 42 submissions (see Table 6.13(a)) *Clockhouse* finished in the lower half of the pack in 28th place, with a score of 6.98/10, averaged over the 10 judges. Within the guidelines given to

judges, a score of 7/10 represents a work of “average” quality. Comments from the judges said that they liked the idea of *Clockhouse*, but felt it lacked sufficient thematic development. They thought it was rather monotonous, and felt it was far too long a piece, given its lack of thematic progression.

The exciting part of their judgment does not appear until we look at the score breakdowns. Each piece was scored by each judge on a number of attributes, such as thematic development, orchestration, etc., one of which was “creativity.” Within the creativity dimension, *Clockhouse* scored 8.5/10 and finished in 2nd place.

Based on the evidence of this experiment alone, with a sample size of 1, it is conceivable that Wheelsong contributed nothing to *Clockhouse*, and that any creative merit it exhibits was provided by the “genius” of the composer, in spite of the tools. When this argument was pointed out to me, I immediately submitted a second composition, to the July 2010 competition, which had a theme of “pastoral scenes.” This time, however, I chose a work from among those that were composed with traditional tools, and not with Wheelsong.

The only work in my portfolio that came even close to “pastoral” was *The Lament*, which was composed as an experiment in reverse-Schenkerian composition, and later encoded into Wheelsong format for subsequent experimentation.

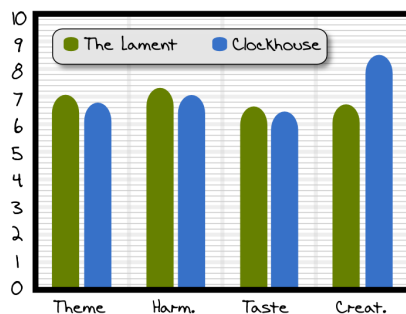


Figure 6.9: Component scores for *The Lament* and *Clockhouse*

Table 6.13(b) shows the results. *The Lament* received an overall score of 6.74, similar to the score achieved by *Clockhouse*. This time, however, the creativity score (6.72) was almost exactly the same as the other scores.

In fact, of the eight component scores shown in Figure 6.9 (each composition scored for themes, harmonies, taste and creativity) one stands out as a conspicuous outlier: the creativity score assigned to the Wheelsong-facilitated *Clockhouse*. All other scores fell within a fairly tight and consistent range. This seems to refute the suggestion that my own innate composing skills were the cause of *Clockhouse*’s high creativity score, suggesting instead that they are somewhat less than average, and that it was the introduction of Wheelsong to my creative process that was the more likely origin of the increased creativity.

6.6 Conclusions and Implications

At the beginning of this chapter, I set out to explore the question of whether or not I would be able to measure a difference between the distribution of subjective creativity scores assessed on two different populations of music fragments, produced through two distinct methods.

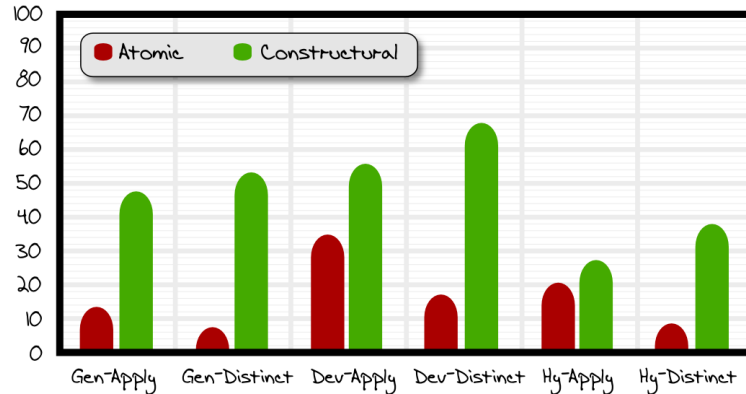


Figure 6.10: Recapping the high-score production levels

Figure 6.10 recaps the key results of this investigation, showing the numbers of candidates rated highly for applicability and for distinctness in each of the three phases, corresponding to the contextual quality and density of the fragments produced. In every case, the constructural methods significantly out-perform the atomic approach. The sought after distinction was indeed present and measurable. What is more interesting, however, is what we can infer from those differences.

In all three phases—which each corresponds to a well known composing methodology employed by both amateur and expert composers alike—the differences were more than just measurable, they were substantial. At the outset of this experiment, I would have been happy to measure a 2, 3 or 5% improvement in creativity scores. So long as those improvements were backed consistently by an indication of statistical significance, that would have been sufficient motivation to warrant further exploration, to see if those small differences might be exploited and amplified. I believed that if we could provide users with more satisfying creative results perhaps 10% or 20% more often than was currently the case, I might be able to make a worthwhile contribution. Instead, the measured improvements were between 40% and 500%, depending upon which particular attribute or collection of attributes was being compared, and which composing process was employed.

The benefits of constructural composition seem clearest in the first, blank-slate phase of the experiment, which, coincidentally, employed no pre-existing compositional content. In fact, we can view the three phases of the experiment as being a progression of contexts — beginning with no prior art at all and then progressing to working with a single piece of prior music, and then finally to working with two prior pieces at a time. As we progress through those contexts, I note a

corresponding decline in the statistical distinctness of the score distributions, although it is unclear whether this is an artifact of constructural versus atomic construction, or whether it might be an artifact of the experimental methods used in this study. I suspect the latter case.

Recall that in the previous chapter, producing exploratory compositions, the best outputs were produced as a result of combining two pre-existing works—a result which seems diametrically opposed to the trend of this experiment. The difference, I believe, lies in the presence or absence of conscious guidance from a human artist. The chain of modifications applied in the variation phase of this experiment were done blindly. If the first variation produced an unpleasant result, the sequence continued regardless, elaborating upon a non-optimal first step. Under human guidance, that chain would likely have been abandoned, and a more promising first variation would have been found before progressing to a second step.

In principle, human guidance would be of benefit to both constructural and atomic methodologies, but if there is a bias in the perceived creative value of constructural operations (as is suggested by the generative phase of this experiment) then that bias would likely be amplified as higher-merit operations become compounded by each successive step in the variation process.

The hybridization phase may also have suffered from the lack of conscious guidance, perhaps to an even greater degree. In the explorations described in Chapter 5, I note that when I was hybridizing compositions, my changes to the structures were not completely random. I consciously selected components to swap or change. While I was never completely certain of exactly which aspects of a piece were the result of which structures, I usually had some inkling. For example, I knew that the `wsngfromscale` operator was vitally connected to the tonal palette, and that the `wsngbeatstuffer` operator's impact was largely rhythmic. Even imprecise information like this provides important clues, allowing the artist to make better-informed choices about how to manipulate the compositions to creative benefit.

This seems to suggest that the next phase of investigation should introduce the dimension of conscious human guidance to the process, to determine what effect, if any, that has on the creative efficiency of constructural tools. This carries with it, however, additional problems of how to assess the encoding scheme without introducing further uncertainties adherent to an experimental interface, independent of the experimental encodings. Such an experiment would ideally employ an interface that did not expose the users to the internal representations, and that worked equally effectively with both atomic and constructural content. A first step toward that goal might be to repeat these experiments using a human-guided genetic exploration methodology in which each candidate was treated as a black box, and to which genetic operations could be applied. Such an experiment would permit conscious decisions to be introduced into the chain of modifications without actually exposing the user to the distinct details of how each candidate was encoded.

Regardless of which step comes next, though, the magnitude of the benefits measured in this

experiment seem to cry out for confirmation and further study. I would hope that these results serve as inspiration for other researchers to join this exploration, to see if we might be able, collectively, to introduce profound changes to the degree of creative richness offered to composers by their tools.

CHAPTER 7

CONCLUSIONS

In this chapter, I will summarize the findings and contributions reported in this thesis, and the implications they have for the ongoing quest to improve facilitation of creativity in media tools. I will also enumerate the known limitations of the current system and suggest potential avenues along which further research and development might be profitably pursued.

7.1 Contributions

I conducted a survey of creativity theory and facilitation literature, and from this, determined that:

- while there is general agreement on what attributes are common to creative ideas, in practice there are two distinct species: h-creativity and p-creativity, differentiated by the contexts in which the assessment is made. Of these, I concluded that the p-creative species is under-explored in software tools and is of high potential value as an avenue to supporting human creativity in software.
- the p-species of creativity should be further divided into two sub-species, which I call *active* and *reflective*, and I propose that of them, the active form is both the more powerful and more relevant form for guiding software design.
- there are a number of concrete theories and findings in our general understanding of creativity and psychology, which I have transformed into a set of foundational premises that can be combined in different ways to specify approaches to facilitating creativity in software that emphasize different qualities.
- there are a number of attributes by which creative ideas are often recognized, which I have re-framed into relatively orthogonal qualities that can be used as a starting point for developing indicators of creative merit.
- insufficient attention has been paid by software designers to the shift that occurs in the artist's cognition (and consequently, in his or her needs) as creative projects mature. Brain activity is known to shift from associative thinking at the early idea-seeking phase, to analytical thinking

at the later, idea refinement stage. Of these two, I concluded that the early, associative mode is both under-explored and of high potential value as an approach to facilitating creativity.

- the p-creative process is inherently subjective and is therefore not meaningfully assessable by external judges, since it relates to the internal cognitive state of the subject at a particular moment in time, rather than to any externally accessible attributes; and that this conflict, which I call the *judgement conundrum* presents a problem for experimental research into p-creative facilitation.
- there are a number of guidelines employed within the field regarding how creativity ought to be facilitated, and which are widely followed, but which do not appear to have been confirmed experimentally.

Upon examining the set of foundational premises, I then:

- articulated a coherent theory of constructural design, by which early-stage, p_a -creativity might be facilitated in software;
- devised a method to test for p_a -creative facilitation without running afoul of the judgement conundrum;
- created an implementation of the constructural design theory, called *Wheelsong*, providing proof that the theory can be implemented, as well as providing a test-bed by which the facilitative power of the theory can be tested;
- proposed four distinct measures of the compactness or efficiency of structural sketches.

The experiments conducted using these above-cited contributions were then used to produce:

- evidence to affirm the value of *Wheelsong* (and by extension, constructural design) for providing richer support for p_a -creative exploration than is offered by atomic tools such as notation editors;
- evidence to suggest that constructurally designed tools are capable of transforming the negatively-construed phenomenon of user error into a positively construed process of creative inspiration;
- a body of musical compositions which have passed the test for p_a -creative merit (and subsequently, some degree of p_r -creative merit as well) which might be used in later studies to explore the relationship between the various modes of p- and h-creative facilitation.

7.2 Limitations

Despite the above listed positive contributions, a number of limitations and weaknesses have also been noted.

- My experiments have been largely silent on the purpose-driven approach to composition in which composers seek some particular kind of result. Focus in this work has been placed on a more open-ended style of inspiration-seeking in which the composer has no destination in mind and is simply looking for appealing new ideas.
- In its current form, *Wheelsong* does not readily support the efficient specification of some kinds of thematic variation. Many forms of music employ repetition of key phrases and melodies, but often with minor variations for the sake of variety or to conform to subtly different musical contexts. For example, the tail end of a melodic phrase will often be modified to suit the harmonic or affective context in which it is being restated. *Wheelsong* currently requires the composer to re-build each variation as a separate entity. A solution to this would be to re-cast the command interface of all operators, adding provision for sub-stream selection patterns to limit the scope of effect for an operator. Selection should be possible on any combination of component values, such as all pitches between 60 and 70 that are on voice 2 with a volume of more than 65, or any note events prior to time 7 with a duration of 0.5. The selection interface should also offer the ability to either trim the unselected events from the data stream or to pass them through unmodified, while applying the associated effect only to the selected events.
- The current implementation of *Wheelsong* is also unnecessarily preoccupied by operations on pitch. While many operations most naturally apply to a given component or set of components (e.g. `wsngrtranspose` most logically applies to pitch), there should be no reason why they cannot be applied to other, less intuitive components, if only for the sake of experimental flexibility. Hence, the proposed selection interface should also be able to designate target component(s) for the associated operation.
- There is currently no provision in *Wheelsong* for dynamic structures. A mechanism that would allow a tree to be defined as a variation of another tree would be extremely powerful, and I suspect that such meta-modifications lie at the heart of what human composers like Bach often do. Ironically, this suggests that what is needed is a *constructural* language for building the operation graphs, rather than the static, atomic language that is currently implemented.

7.3 Future Directions

The work discussed in this thesis opens avenues to a number of different lines of further inquiry, both within the context of music and the existing *Wheelsong* system, as well as following the more general implications of constructural design into other software facilitated forms of media. Here I will outline some of the more tantalizing possibilities, couched in terms of questions that seem worth having answers to.

7.3.1 Music Composition

“Are the benefits of constructural design enhanced when the process of creating musical fragments is guided by human aesthetic judgement?” As discussed in Section 6.6, properly assessing the potential benefits of constructural design, as implemented in *Wheelsong*, may have been hampered by the lack of conscious human guidance in the process of creating musical fragments. Implementing a black-box interface through which users of arbitrary musical skill could intervene in and direct the development of new musical fragments would allow for a much better assessment of constructural design principles in the context of some compositional methodologies, without requiring the development of a detailed interface to expose users to the internal complexities within the constructs.

“Does having the ability to manipulate the internals of constructural representations provide for greater p_a -creative user satisfaction?” I have suggested that the lack of conscious human guidance may have contributed to the weaker score differentials noted in later phases of the richness test, but can that be substantiated?

“Do any measured advantages of giving users access to the constructural internals change depending on the musical and/or compositional experience of the user?” Users have told me that they want to be able to manipulate the internals of their encodings, and this was especially true of the musically sophisticated users. While providing such tools would likely translate into higher levels of user satisfaction with the process, I am not convinced that it would also translate into higher levels of satisfaction with the results produced.

“Can an optimal operation set be defined that maximizes *Wheelsong*’s expressive reach while minimizing the number of operators and parameters required?” There is a tension between the need to maximize the expressive reach of a tool and the need to minimize the complexity of its interface. Too little expressivity renders it irrelevant to all but casual users, whereas the complexity of having too many tools to choose from increases the cognitive burden of using the tool and interferes with flow. To maximize expressivity and minimize choice complexity, it would be worthwhile to seek a *minimal spanning set* of operators that could produce the greatest range of outputs with the fewest number of operations and parameters.

“Can we automate the process of building creatively exploitable constructural representations of arbitrary musical input?” Casual users who have been exposed to *Wheelsong* often ask if they can experiment with a favourite song, but creation of constructural representations has so far only been possible through a time-consuming, manual process. Finding a way to generate creatively valid constructural encodings quickly would be of immense practical benefit. Generating a constructural representation of an arbitrary piece of music is easy if we are willing to accept degenerate structures. For example, the entire input song could be encoded as a literal WSNG string, which was then emitted with a single `wsngsequence` operator, but this would be creatively degenerate, since no real structural information is being encoded into the graph, and would therefore be of no practical creative value. Conversely, my *Invention Sketch* shows that *musical* naivety can still be exploited for creative gain. This suggests that it might be possible to automate the process of constructing useful encodings from input MIDI data without having to invent automated musicological analysis algorithms first.

“Do indicators of encoding compactness have signals embedded within them that can be used to infer either the absolute or relative creative maturity of a work in progress?” By capturing the various compactness and leveragability metrics during the course of a piece’s development, and comparing them to explicit input from the user regarding her perception of her progress on the work, we might be able to find a correlation that can be used to drive an automated progress metric. If such an indicator can be extracted, it would permit more responsive user interfaces that were better able to adapt to the changing state of the work and the user’s corresponding cognitive state.

“Can *Wheelsong* be used to improve the comprehension and/or retention of music theory in students?” It seems clear that musical compositions contain within them some implicit encoding of the musical and aesthetic rules used to compose them. This is the principle on which Cope’s *EMI* [17] and Hoover and Stanley’s *NEAT Drummer* [46] systems are founded. It is also the principle that I believe accounts for the success of my own musical explorations.

These tools preserve those embedded rules and exploit them in the service of creating new works, whether the user is educated in music theory or not, but while *EMI* and *NEAT Drummer* typically keep the extracted structures hidden from the user, *Wheelsong* places them front and centre, making them the focal point of the user’s attention. I suspect that this shift in focus — away from the flurry of notes and toward the rules that produced them — may offer significant advantages for teaching music theory, perhaps even teaching elements of music theory passively, through repeated exposure.

“Can *Wheelsong* be used to improve the successful engagement of early-stage music students?” By increasing the perceived musicality of works produced by novice users, *Wheelsong* permits them to be aesthetically successful more often, and sooner in the course of their musical

practice than they otherwise might have been. When they are rewarded more often, and more richly for those first tentative steps, it is possible that they will be more likely to pursue the field with more confidence and less likely to abandon their studies in frustration.

“Can the principles of constructural design be applied to tools for other media?”

In the abstract, these principles seem to be applicable to a number of different media, and the idea has received some preliminary traction in the field [106]. In practice, Karl Sims’s work [104] has already demonstrated one way in which constructures might be applied to encoding textural images, and I suspect that it might also be applied to more representational imagery as well.

Another, less obvious area that seems to have potential is that of creative writing. Full compliance with constructural design would require that the media elements encoded by the software be represented in a way that permits constructive generation of the output media artifact from the encoded structures. In *Wheelsong*, this was accomplished by formulating the structure nodes as musical fragment transformations, but such an approach would not work in the field of creative writing. Computer science is not yet at the point where a simple “cultural milieu” transform could be applied to *Romeo and Juliette* to produce *West Side Story* as an output, nor could the atomic elements of “stuffed toys,” “lonely boy,” and “a bear at the zoo” be combined through a network of software transforms to construct a manuscript for “Winnie the Pooh.”

This does not mean, however, that constructural design cannot be applied to the creation of creative writing tools, but it was not until my experiments in another field — collaborative text analysis — began to bear fruit that I understood how it might be done.

In many ways, text analysis is the opposite of creative writing. While a writer starts with some ideas and then slowly weaves them into an elaborate textual matrix, the analyst starts at the other end, with an extant text, and then slowly tries to tease its constituent ideas and structures back out of it. Both need to be able to represent a large number of interdependent ideas, both need to be able to represent any number of revisions/witnesses/drafts of a text, both need to be able to make linkages and assignments between one or more ideas and passages of text, and both need to be able to sprinkle commentary arbitrarily throughout the entire network. The only thing that differs between the two practitioners (at this level of abstraction) is the order in which the various elements are placed into the system.

These are precisely the kinds of things that can be used as the “structures” of a creative writing project: the themes, ideas, and other abstract notions that combine to form the writer’s artistic materiel. My experiments with collaborative text analysis tools [67] have shown me that this annotative model is a powerful way to encode the literary scholars’ view of the world, even allowing multiple, conflicting viewpoints to be expressed entirely independently of one another within the same data network. I have already begun to explore [105] how these principles might be extended to provide tools for both writers and literary scholars alike.

“Can we confirm the apparent utility constructural tools have for domain theorists, computer theorists, domain practitioners and domain pedagogues?” And if so, does juxtaposing these distinct domain interests within a single, common tool present any opportunities to facilitate richer, more creative advances within that domain? It seems at least plausible that a constructural interfaces can act as a catalyst for interdisciplinary confluences between the various facets of the domain. Consider the music-specific case. Experimental operators created by computer science theorists become tools that can be used by composers; visualization tools created for composers become teaching aids for pedagogues; rapid sketching tools allow musicologists to explore structural theories rapidly, etc. These seem to potentially form interdisciplinary feedback loops in which each discipline inspires and drives the others, and it is not difficult to imagine similar exchanges occurring in the context of other media, such as writing tools, visual arts, sculpture and the like.

7.4 In Summary

By embracing the mode of cognition known to dominate early-stage creative cognition, a different approach to architecting software tools—privileging structure and pattern over note-specific details—is suggested, which can be used to facilitate more satisfyingly creative and diverse exploratory music composition than is achieved with traditional encoding schemes.

Compositions encoded in this form embed not only the notes of the composition, but much of the internal logic and rules governing its construction as well, the presence of which helps to guide novice users toward more complex and satisfying compositions while maintaining a greater degree of musical validity than is achieved with detail-oriented encodings.

The magnitude of the measured improvements is extremely encouraging, and suggests that this may be an important new way for computer scientists to think about music composition software design. Furthermore, because these advances arise from creativity and cognition theory, rather than relying on domain-specific music theory, this same approach should be adaptable to other media as well, such as visual arts, sculpture or narrative text.

In the years to come, I hope to expand upon this work, developing user interfaces that combine structured representations with exploratory workflows in a variety of media contexts, allowing more people to be more creative more often, just as Shneiderman challenged us to do.

REFERENCES

- [1] Mayumi Adachi and Yukari Chino. Inspiring creativity through music. pages 305–340. World Scientific Publishing Co. Pte. Ltd., 2004.
- [2] Michael Agustin, Gina Chuang, Albith Delgado, Anthony Ortega, Josh Seaver, and John W. Buchanan. Game sketching. In *DIMEA '07: Proceedings of the 2nd international conference on Digital interactive media in entertainment and arts*, pages 36–43, New York, NY, USA, 2007. ACM Press.
- [3] E. M. L. S. Alencar. Obstacles to personal creativity among university students. *Gifted Education International*, 15:133–140, 2002.
- [4] T. M. Amabile. *The Social Psychology of Creativity*. Springer–Verlag, New York, NY, 1983.
- [5] Guido Arezzo. Micrologus (circa 1026). In Claude V. Palisca, editor, *Hucbald, Guido And John On Music*, pages 57–86. Yale University Press, New Haven, Conn., 1978. Translated by Warren Babb.
- [6] G. Assayag, C. Agnon, J. Fineberg, and P. Hanappe. An object-oriented visual environment for musical composition. In *Proceedings of the 1998 International Computer Music Conference*, pages 364 – 367. International Computer Music Association, 1998.
- [7] S.P. Besemer and K. O’Quin. Assessing creative products: Progress and potentials. In S.G. Isaksen, M.C. Murdock, R.L. Firestien, and D.J. Treffinger, editors, *Nurturing and Developing Creativity: The emergence of a discipline*. Ablex, 1993.
- [8] John A. Biles. Genjam: A genetic algorithm for generating jazz solos. In *ICMC Proceedings 1994*, 1994.
- [9] Margaret Boden. *The Creative Mind: Myths and Mechanisms, Second Edition*. Sphere Books, 2004.
- [10] J. Peter Burkholder, Andreas Giger, and David C. Birchler, editors. *Musical Borrowing: An Annotated Bibliography*. Jacobs School of Music, Indiana University, Bloomington, IN, June 2003. <http://www.chmtl.indiana.edu/borrowing/browsest.html> (accessed on Sept. 25, 2009).
- [11] Allen Cadwallader. Schenker’s unpublished graphic analysis of brahms’s intermezzo op. 117, no. 2: Tonal structure and concealed motivic repetition. *Music Theory Spectrum*, 6:1–13, 1984.
- [12] Jos Campos and A. Dias De Figueiredo. Programming for serendipity. In *Proceedings of the 2002 AAAI Fall Symposium on Chance Discovery the Discovery and Management of Chance Events*, AAAI, 2002.
- [13] Linda Candy and Ernest Edmonds. Creativity enhancement with emerging technologies. *Communications of the ACM*, 43(8):62–65, 2000.
- [14] Linda Candy and Ernest Edmonds. Modeling co-creativity in art and technology. *Proceedings of the 4th conference on creativity and cognition*, pages 134–141, 2002.

- [15] Erin A. Carrol, Celine Latulipe, Richard Fung, and Michael Terry. Creativity factor evaluation: Towards a standardized survey metric for creativity support. In *Creativity and Cognition '09*, Berkeley, CA, October 2009. ACM Press.
- [16] D. Cope. Computer modelling of musical intelligence in EMI. *Computer Music Journal*, 16(2):69–83, 1992.
- [17] David Cope. *Experiments In Musical Intelligence*. A-R Editions, Inc, Madison, Wisconsin, 1996.
- [18] David Cope. *Virtual Music: Computer Synthesis Of Musical Style*. MIT Press, 2001.
- [19] David Cougar. *Creativity and Innovation in Information Systems Organizations*. Boyd and Fraser Publishing Co., Danvers, MA, 1996.
- [20] Mihaly Csikszentmihalyi. *Flow : The Psychology Of Optimal Experience*. Harper and Row, New York, 1990.
- [21] Mihaly Csikszentmihalyi. *Creativity : Flow And The Psychology Of Discovery And Invention*. HarperCollinsPublishers, New York, 1996.
- [22] T. Dartnell. Artificial intelligence and creativity: An introduction. *Artificial Intelligence and the Simulation of Intelligence Quarterly*, 5, 1993.
- [23] Daniel Clement Dennett. *Brainstorms: Philosophical Essays on Mind and Psychology*. MIT Press, 1981.
- [24] Frederick Dorian. *The Musical Workshop*. Harper & Row, New York, NY, 1947.
- [25] W. Jay Dowling. Scale and contour: Two components of a theory of memory for melodies. *Psychological Review*, 85(4):341–354, 1978.
- [26] Barry Eaglestone and Nigel Ford. Computer support for creativity: help or hindrance. 2002.
- [27] Barry Eaglestone, Nigel Ford, Guy Brown, and Adrian Moore. Information systems and creativity: an empirical study. *Journal of Documentation*, 63(4), 2007.
- [28] Barry Eaglestone, Nigel Ford, Ralf Nuhn, Adrian Moore, and Guy Brown. Composition systems requirements for creativity: what research methodology? In C. L. Buyoli and R. Louyeiro, editors, *Proceedings of Mozart Workshop on Current Research Directions in Computer Music*, pages 7–16, Barcelona, Spain, 2001.
- [29] Kemal Ebcioğlu. An expert system for harmonising chorales in the style of J.S. Bach. *Journal of Logic Programming*, 8:145–185, 1988.
- [30] Morwaread Farbood, Egon Pasztor, and Kevin Jennings. Hyperscore: a graphical sketchpad for novice composers. *IEEE Computer Graphics and Applications*, 24(1), 2004.
- [31] Gerhard Fischer, Elisa Giaccardi, Hal Eden, Masanori Sugimoto, and Yunwen Ye. Beyond binary choices: Integrating individual and social creativity. *International Journal of Human-Computer Studies*, 63:482–512, 2005.
- [32] Rajmil Fischman. Clouds, pyramids, and diamonds: Applying schrodinger’s equation to granular synthesis and compositional structure. *Computer Music Journal*, 27:47–69, Summer 2003.
- [33] Harold E. Fiske. *Understanding Musical Understanding: The Philosophy, Psychology, and Sociology of the Musical Experience*. The Edwin Mellen Press, Lewiston, NY, 2008.
- [34] Kenneth D. Forbus, Ronald W. Ferguson, and Jeffery M. Usher. Towards a computational model of sketching. In *IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces*, pages 77–83, New York, NY, USA, 2001. ACM.

- [35] Johann Joseph Fux. The study of counterpoint from johann joseph fux’s gradus ad parnassum, 1943 (1725).
- [36] Liane Gabora. Cognitive mechanisms underlying the creative process. In *Proc. of the 4th Conf. on Creativity and Cognition*, pages 126–133. ACM Press, 2002.
- [37] Brewster Ghiselin. *The creative process: a symposium*. University of California Press, 1985.
- [38] Michael Gogins. Iterated function systems music. *Computer Music Journal*, 15:40–48, 1991.
- [39] Catherine Golden. Composition: Writing and the visual arts. *Journal of Aesthetic Education*, 20:59–68, Autumn 1986.
- [40] Saul Greenberg. Toolkits and interface creativity. *Multimedia Tools and Applications*, 32(2):139–159, 2007.
- [41] Saul Greenberg and Bill Buxton. Usability evaluation considered harmful (some of the time). In *CHI 2008*, pages 111–120, Florence, Italy, 2008. ACM Press.
- [42] Dane L. Harwood. Universals in music: A perspective from cognitive psychology. *Ethnomusicology*, 20(3):521–533, 1976.
- [43] Tom Hewett, Mary Czerwinski, Michael Terry, Jay Nunamaker, Linda Candy, Bill Kules, and Elisabeth Sylvan. Creativity support tool evaluation and metrics. In Ben Shneiderman, Gerhard Fischer, Mary Czerwinski, Brad Myers, and Mitch Resnick, editors, *Creativity Support Tools*. Washington, DC, June 2005.
- [44] Keiji Hirata and Tatsuya Aoyagi. Computational music representation base on the generative theory of tonal music and the deductive object-oriented database. *Computer Music Journal*, 27:73–89, Fall 2003.
- [45] John Holland. *Adaptation In Natural And Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.
- [46] Amy K. Hoover and Kenneth O. Stanley. Exploiting functional relationships in musical composition. *Connection Science Special Issue on Music, Brain and Cognition*, 21(2), 2009.
- [47] Michael J. A. Howe, Jane W. Davidson, and John A. Sloboda. Innate talents: Reality or myth? *Behavioral and Brain Sciences*, 21(3):399–407, 1994.
- [48] Paul Hudak, Tom Makucevich, Syam Gadde, and Bo Whong. Haskore music notation - an algebra of music. In *Journal of Functional Programming*, volume 6, 1996.
- [49] David Huron. *Sweet Anticipation: Music And The Psychology Of Expectation*. MIT Press, 2006.
- [50] P. Jackson and S. Messick. The person, the product and the response: Conceptual problems in the assessment of creativity. *Journal of Personality*, 3, 1965.
- [51] Bruce L. Jacob. Algorithmic composition as a model of creativity. *Organised Sound*, 1:157–165, 1996.
- [52] Julian Jaynes. *The Origin of Consciousness in the Breakdown of the Bicameral Mind*. Houghton Mifflin, Boston, 1976.
- [53] Lile Jia, Edward R. Hirt, and Samuel C. Karpen. Lessons from a faraway land: The effect of spatial distance on creative cognition. *Journal of Experimental Social Psychology*, 45(5):1127 – 1131, 2009.
- [54] Fred H. Speece Jr. Assessing the quality of earnings and management. In *AIMR Conference Proceedings: CFA Equity Research and Valuation Techniques*, pages 16–20, May 1998.

- [55] John. J. Kao. The art & discipline of business creativity. *Strategy & Leadership*, 25(4):6–11, 1997.
- [56] Ai-Tee Koh. Linking learning, knowledge creation, and business creativity: A preliminary assessment of the east asian quest for creativity. *Technological Forecasting and Social Change*, 64(1):85–100, 2000.
- [57] Shulamith Kreitler and Hernan Casakin. Self-perceived creativity: The perspective of design. *European Journal of Psychological Assessment*, 25, 2009.
- [58] J. A. Landay and B. A. Myers. Interactive sketching for the early stages of user interface design. *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 43–50, 1995.
- [59] J. A. Landay and B. A. Myers. Sketching interfaces: Toward more human interface design. *Computer*, 34, Mar 2001.
- [60] M. Laurson and J. Duthen. Patchwork: A graphical language in PreForm. In *Proceedings of the 1989 International Computer Music Conference*, pages 172 – 175. International Computer Music Association, 1989.
- [61] Roger Leenders, Jo van Engelen, and Jan Kratzer. Virtuality, communication and new product team creativity: a social perspective. *Journal of Engineering and Technology Management*, 20(1–2):69–92, 2003.
- [62] T. H. Lentz. Acquiescence as a factor in the measurement of personality. *Psychological Bulletin*, 35, 1938.
- [63] Fred Lerdahl and Ray Jackendoff. *A Generative Theory Of Tonal Music*. MIT Press, 1983.
- [64] Fred Lerdahl and Ray Jackendoff. Hierarchical structure in music. In Stephan M. Schwanauer and David A. Levitt, editors, *Machine Models of Music*, pages 289–312. MIT Press, Cambridge, MA, 1993.
- [65] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8), 1965.
- [66] Daniel J. Levitin and Vinod Menon. Musical structure is processed in “language” areas of the brain: a possible role for Brodmann Area 47 in temporal coherence. *NeuroImage*, 20(4):2142 – 2152, 2003.
- [67] Yin Liu and Jeffrey A. Smith. A relational database model for text encoding. In *Computing in the Humanities Working Papers*. University of Toronto, 2008.
- [68] Kenneth R. MacCrimmon and Christian Wagner. Stimulating ideas through creativity software. *Management Science*, 40(11):1514–1532, Nov. 1994.
- [69] Brent MacGregor. Cybernetic serendipity revisited. *Proceedings of the 4th conference on Creativity and Cognition*, pages 11–13, 2002.
- [70] Stephen Malinowski. The music animation machine player, 2009. <http://www.musanim.com/player> (Accessed on Jan 19, 2009).
- [71] Lena Mamykina, Linda Candy, and Ernest Edmonds. Collaborative creativity. *Communications of the ACM*, 45(10):96–99, 2002.
- [72] Jeff Mauzy and Richard Harriman. *Creativity, Inc.: Building An Inventive Organization*. Harvard Business School Press, 2003.
- [73] Scott McCloud. *Understanding Comics: the Invisible Art*. HarperPerennial, Inc., 1994.

- [74] Ann McCutchan. *The Muse that Sings: Composers Speak about the Creative Process*. Oxford University Press, 1999.
- [75] Donald Meichenbaum. Enhancing creativity by modifying what subjects say to themselves. *American Educational Research Journal*, 1975.
- [76] Brian C. J. Moore. *An Introduction to the Psychology of Hearing*. Elsevier Press, 5th edition, 2004.
- [77] Edmund Morris. *Beethoven: the universal composer*. Harper Collins, 2005.
- [78] Wolfgang Amadeus Mozart. *Musikalisches Würfelspiel, K516f, K294d, K. Anhang C 30.01*. Schott, Sohne, Mainz, 1956.
- [79] Brad Myers, Scott E. Hudson, and Randy Pausch. Past, present and future of software interface tools. *ACM Transactions on Computer-Human Interaction*, 7(1):3–28, 2000.
- [80] Martin L. Nass. On hearing and inspiration in the composition of music. *Psychoanalytic Quarterly*, 44:431–449, 1975.
- [81] Charlan Jeanne Nemeth and Margaret Ormiston. Creative idea generation: Harmony versus stimulation. *European Journal of Social Psychology*, 37(3), 2006.
- [82] Bruno Nettl. An ethnomusicologist contemplates universals in musical sound and musical culture. In Nils L. Wallin, Björn Merker, and Steven Brown, editors, *The Origins of Music*, page 512. MIT Press, 2001.
- [83] A. Newell, J. Shaw, and H. Simon. The processes of creative thinking. In H. Gruber, G. Terrell, and M. Wertheimer, editors, *Contemporary approaches to creative thinking*. Atherton Press, 1962.
- [84] R. S. Nickerson. Enhancing creativity. In R. J. Sternberg, editor, *Handbook of Creativity*. Cambridge University Press, 1999.
- [85] Nadjé Noordhuis. *A Beginner's Guide to Composing: How to Start Writing Music at Any Age*. Bloomingdale School of Music, New York, NY, 2008. <http://www.bsmny.org/features/composing/index.php> (accessed on Sept. 25, 2009).
- [86] Ralf Nuhn, Barry Eaglestone, Nigel Ford, Adrian Moore, and Guy Brown. A qualitative analysis of composers at work. In *Proceedings of the International Computer Music Conference (ICMC) 02*, pages 572–580, Gothenburg, Sweden, 2002. International Computer Music Association.
- [87] Ken Perlin. An image synthesizer. *SIGGRAPH Comput. Graph.*, 19(3):287–296, 1985.
- [88] Jonathan A. Plucker and Joseph S. Renzulli. Psychometric approaches to the study of human creativity. In Robert J. Sternberg, editor, *Handbook of Creativity*. Cambridge University Press, 1999.
- [89] Sam Reese. Tools for thinking in sound. *Music Educators Journal*, 88(1):42–46, 2001.
- [90] Steve Reich. *Clapping Music for Two Performers*. The Press of the Nova Scotia College of Art and Design (Co-published by: New York University Press), 1980.
- [91] Mitch Resnick, Brad Myers, Randy Pausch, Kumiyo Nakakoji, Ben Shneiderman, Ted Selker, and Mike Eisenberg. Design principles for tools to support creative thinking. In *Creativity Support Tools*, pages 25–36, 2005.
- [92] Mitchel Resnick, Brad Myers, Kumiyo Nakakoji, Ben Shneiderman, Randy Pausch, Ted Selker, and Mike Eisenberg. Design principles for tools to support creative thinking, June 2005. Presented at Workshop on Creativity Support Tools (www.cs.umd.edu/hcil/CST).

- [93] F. Raymond Ritter. Robert schumann. *Dwight's Journal of Music*, 37(4), May 1877.
- [94] Curtis Roads. *Computer Music Tutorial*. MIT Press, Cambridge, MA, 1996.
- [95] Curtis Roads. Sound composition with pulsars. *Journal of the Audio Engineering Society*, 49(3):134–147, 2001.
- [96] Michael Robinson. Composing with computer instruments and ragas. *The IDEA*, (6), 2002.
- [97] I. Wayan Sadra, Jody Diamond, Ayi Solehuddin, and Theresa Rohlck. "komposisi baru": On contemporary composition in indonesia. *Leonardo Music Journal*, 1(1):19–24, 1991.
- [98] Carla Scaletti. Computer music languages, kyma, and the future. *Computer Music Journal*, 26(4):69–82, Winter 2002.
- [99] Heinrich Schenker. Five graphic music analyses, 1969.
- [100] Christina E. Shalley, Lucy L. Gilson, and Terry C. Blum. Matching creativity requirements and the work environment: Effects on satisfaction and intentions to leave. *The Academy of Management Journal*, 43(2):215–223, 2000.
- [101] Ben Shneiderman. Supporting creativity with advanced, information-abundant user interfaces. Technical report, Institute for Systems Research, University of Maryland, College Park, MD, 7 1999.
- [102] Ben Shneiderman. Creating creativity: User interfaces for supporting innovation. *ACM Transactions in Computer Human Interaction*, 7(1):114–138, 2000.
- [103] Ben Shneiderman. Creativity support tools: A grand challenge for HCI researchers. In Miguel Redondo, Crescencio Bravo, and Manuel Ortega, editors, *Engineering The User Interface*, pages 1–9. Springer-Verlag London Limited, 2009.
- [104] Karl Sims. Artificial evolution for computer graphics. In *Proceedings of SIGGRAPH '91*, volume 25, pages 319–328. ACM Press, July 1991.
- [105] Jeffrey A. Smith. E-publishing: Politics and pragmatics. chapter A First-Principles Reinvention of Software Tools for Creative Writing in the 21st Century. University of Toronto, 2010. In press.
- [106] Jeffrey A. Smith, David Mould, and Mark Daley. Constructures: Supporting human creativity in software. *Digital Creativity*, 20(1), 2009.
- [107] László Somfai. *Béla Bartók: Composition, Concepts and Autograph Sources*. University of California Press, 1996.
- [108] Neil Sorrell. *A Guide To The Gamelan*. Amadeus Press, Portland, Oregon, 1990.
- [109] M. I. Stein. *Stimulating Creativity, Volume 1*. Academic Press, 1974.
- [110] Rick Taube. Common music: A music composition language in common lisp and clos. *Computer Music Journal*, 15:21–32, 1991.
- [111] Belinda Thom. Bob: An interactive improvisational music companion. In *AGENTS '00: Proceedings of the fourth international conference on Autonomous agents*, pages 309–316. ACM Press, 2000.
- [112] Wyndham Thomas. *Composition – Performance – Reception: Studies in the Creative Process in Music*. Ashgate Publishing Limited, Hants, England, 1998.
- [113] E.P. Torrance and K. Goff. A quiet revolution. *Journal of Creative Behavior*, 23, 1989.

- [114] Yaacov Trope and Nira Liberman. Temporal construal. *Psychological Review*, 110(3):403–421, 2003.
- [115] IM Verstijnen, C van Leeuwen, G Goldschmidt, R Hamel, and JM Hennessey. Sketching and creative discovery. *Design Studies*, 19(4):519 – 546, 1998.
- [116] Željko Obrenovic, Dragan Gašević, and Anton Eliëns. Stimulating creativity through opportunistic software development. *IEEE Software*, 25(6):64–70, 2008.
- [117] G. Wallas. *The Art of Thought*. Harcourt, Brace and World, New York, NY, 1926.
- [118] Ge Wang and Perry R. Cook. On-the-fly programming: Using code as an expressive musical instrument. In *Proceedings of the 2004 Conference on New Interfaces for Musical Expression*, 2004.
- [119] Roger Watt and Sandra Quinn. Analysis of local and global timing and pitch change in ordinary melodies. *Proceedings of the 9th International Conference on Music Perception and Cognition*, pages 30–37, 2006.
- [120] G. A. Wiggins. Towards a more precise characterisation of creativity in ai. In R. Weber and C. G. von Wangenheim, editors, *Case-Based Reasoning: Papers from the Workshop Programme at ICCBR'01*, pages 113–120. Washington, DC: Naval Research Laboratory, Navy Centre for Applied Research in Artificial Intelligence, 2001.
- [121] Yin Yin Wong. Rough and ready prototypes: lessons from graphic design. In *CHI '92: Posters and short talks of the 1992 SIGCHI conference on Human factors in computing systems*, pages 83–84, New York, NY, USA, 1992. ACM.
- [122] Iannis Xenakis. *Formalized Music: Thought And Mathematics In Music*. Indiana University Press, Bloomington, Indiana, 1971.
- [123] D. Zicarelli, 2007. <http://www.cycling74.com/story/2007/9/28/105551/882> Accessed on July 7, 2008.

APPENDIX A

WSNG OPERATORS

Interactive help output from WSNT and WSNG operator transforms.

A.1 Generative

Operators that generate musical content from scratch.

A.1.1 wsngscale

`wsngscale [--out1] outfile`

The `wsngscale` program is a member of the WheelSong music composition suite of tools. Use it to generate a simple, ascending scale melody in WSNG format and output it to `outfile` (also in `wsg` format).

OPTIONS

- `-t <tonic note>`
The pitch value of the starting note (default = 24).
- `-m <mode>`
The scale mode to construct from the tonic.
Accepted modes are:
 - 'M' or 0 (major) (This is the default.)
 - 'm' or 1 (natural minor)
 - 'b' or 3 (blues)
 - 'c' or 2 (chromatic)
 - 'o' or 4 (octatonic)
 - 'x' or 5 (octatonic)
 - 's' or 6 (slendro)
 - 'S' or 7 (Sorrel's slendro)
 - 'B' or 8 (Another blues scale)
 - 'H' or 9 (Hungarian folk)
- `-h` Display command help information.
- `-I` Display the interface of this command, in a format expected by discovery applications.
- `-q` Quiet. Decrease the amount of verbosity in the output.
- `-v` Verbose. Provide more detail in the output.

A.1.2 wsngrandom

`wsngrandom [options] [--out1] outfile`

The `wsngrandom` program is a member of the WheelSong music composition suite of tools. Use it to generate melodic notes (in `wsg` format) within a start and end time range that exhibit stepwise movement. Output it to `outfile`.

OPTIONS

- `-c` Clamp pitch values to integers (Default off.)
- `-S <seed>`
Initialization seed for random generator. (Defaults to 17)
- `-s <start>`
Start melody at time `<start>`. (Defaults to 0)
- `-e <end>`
End melody at time `<end>`. (Defaults to 5.0)
Defaults to end of file.
- `-g <rate>`
Stagger the notes by doubling or halving the note durations on the fly

- with likelihood <rate> (Default 0.0)
- p <min pitch>
- P <max pitch>
 - Assign all notes a pitch between <min pitch> and <max pitch>
 - (Defaults to min 60 and max 84.)
- d <dur>
 - Assign all notes a duration of <dur>. (Defaults to 1.0)
- x <vox>
 - Assign all notes to voice <vox>. (Defaults to 0)
- r <rate>
 - Assign a fraction of the notes (given by <rate>) to be rests.
 - (Defaults to zero.)
- h Display command help information.
- I Display the interface of this command, in a format expected by discovery applications.
- q Quiet. Decrease the amount of verbosity in the output.
- v Verbose. Provide more detail in the output.

A.2 Transformative

Operators that transform one or more musical input streams into an output stream.

A.2.1 wsngfromscale

```
wsngfromscale [options] -s scalefile [--in1] infile [--out1] outfile
```

The `wsngfromscale` program is a member of the WheelSong music composition suite of tools. Use it to transform the pitches of `infile` (in `wsng` format) by treating them as indices into a table of pitch values in a second file and output it to `outfile` the results (also in `wsng` format).

There are two basic notation schemes possible: modulus and fractional. In the modulus scheme, the integer component of the index is used to index into the scale, modulus the scale size. So, if a scale has seven pitches, then a value of 1 will be the first, tonic note of the scale, 2 will index to the second and so on. When the value reaches 8, that will wrap around, back to the tonic, but one octave higher. It should be obvious that this scheme mimics the traditional tonic, second, third... system used in western scale degrees.

Within the modulus scheme, any pitch values that contain fractional amounts will be sharpened after the scale lookup is done, proportional to the distance between the indicated scale degree and the next. So, if a scale contains the pitch values 10, 16, 24 then a lookup for 1.0 will resolve as 10, a lookup of 1.5 will resolve as 13 and a lookup of 1.333 will resolve as 12, since the fractional component is doing its best to interpolate between the 10 and the 16 which are the 1st and 2nd degrees of the scale. Note that specifying notes in negative modulus notation is frowned upon, since it involves skipping the value 0. Some attempt has been made to do the right thing, but it breaks easily. You're better off just not using negatives. Plan accordingly.

In fractional notation, a scheme more familiar to computer scientists is used. The integer component specifies the octave and the fractional component specifies the interpolation within the given octave. So instead of fractions interpolating between two adjacent degrees of the scale, it interpolates the entire octave. For example, if the pitch values of the scale are 60, 62, 64, 65, 67, 69, 71, 72 (which is the C Major scale if we're outputting to MIDI) then those notes are indexed as 4.0, 4.125, 4.25, 4.375, 4.5, 4.625, 4.75, 4.875 and 5.0. In addition, the presence of a + sign at the beginning of the index will sharpen the resulting note by a semitone and a - will flatten it.

OPTIONS

```
-s <scalefile>
```

Take the pitch values from scalefile as the input scale

- f Use fractional index notation instead of integer. (Default = integer)
- h Display command help information.
- I Display the interface of this command, in a format expected by discovery applications.
- q Quiet. Decrease the amount of verbosity in the output.
- v Verbose. Provide more detail in the output.

A.2.2 wsngrandomize

wsngrandomize [options] [--in1] infile [--out1] outfile

The wsngrandomize program is a member of the WheelSong music composition suite of tools. Add or subtract a random value to the different channel values in a file (in wsng format).

OPTIONS

- P <value> Modify pitch start channel (Default = 3)
- S <value> Modify pitch end (slur) channel
- T <value> Modify time channel
- D <value> Modify duration channel (Default = 4)
- V <value> Modify volume start channel
- C <value> Modify volume end (crescendo) channel
- X <value> Modify voice channel
- Y <value> Modify voice change channel
<value> is either a constant value, such as "14"
or a relative value, such as "5%"
Modification will be by a random value within the range
-<value> to +<value>
- h Display command help information.
- I Display the interface of this command, in a format expected by discovery applications.
- q Quiet. Decrease the amount of verbosity in the output.
- v Verbose. Provide more detail in the output.

A.2.3 wsnginterpolate

wsnginterpolate [options] [--in1] infile [--out1] outfile

The wsnginterpolate program is a member of the WheelSong music composition suite of tools. Use it to break long notes into shorter notes in the infile (in wsng format) and output it to outfile (also in wsng format).

OPTIONS

- p Interpolate pitches as well as times. (Default is off.)
- g <old>
Interpolate all notes with duration greater than <old> (Default 1.0)
- n <new>
Replace interpolated notes with shorter notes of duration <new>
- P <patternfile>
Subdivide each note given the duration/time pattern specified in patternfile. Disables options -g and -n.
- h Display command help information.
- I Display the interface of this command, in a format expected by discovery applications.
- q Quiet. Decrease the amount of verbosity in the output.
- v Verbose. Provide more detail in the output.

A.2.4 wsngretrograde

wsngretrograde [options] [--in1] infile [--out1] outfile

The `wsngretrograde` program is a member of the WheelSong music composition suite of tools. Use it to reverse the order of music events of `infile` (in `wsng` format) and output it to `outfile` (also in `wsng` format).

OPTIONS

- h Display command help information.
- I Display the interface of this command, in a format expected by discovery applications.
- q Quiet. Decrease the amount of verbosity in the output.
- v Verbose. Provide more detail in the output.

A.2.5 `wsnginvert`

`wsnginvert` [options] [--in1] `infile` [--out1] `outfile`

The `wsnginvert` program is a member of the WheelSong music composition suite of tools. Use it to invert the contents of `infile` (in `wsng` format) around some median pitch and output it to `outfile` (also in `wsng` format).

OPTIONS

- m <pitch value>
The 'mirror' point around which pitches will be 'reflected'. Defaults to the first pitch in the file.
- h Display command help information.
- I Display the interface of this command, in a format expected by discovery applications.
- q Quiet. Decrease the amount of verbosity in the output.
- v Verbose. Provide more detail in the output.

A.2.6 `wsngsequence`

`wsngsequence` [options] -i `infile` [[-i `infile`]...] [--out1] `outfile`

The `wsngsequence` program is a member of the WheelSong music composition suite of tools. Use it to sequence multiple input files into consecutive playback order.

OPTIONS

- h Display command help information.
- s Simultaneous playback. Suppress sequencing and play all files at once.
- i filename
Add the given file to the input sequence, in the order they appear in the argument list.
- I Display the interface of this command, in a format expected by discovery applications.
- q Quiet. Decrease the amount of verbosity in the output.
- v Verbose. Provide more detail in the output.

A.2.7 `wsngloft`

`wsngrepeat` [options] [-in1] `infile` [--out1] `outfile`

The `wsngrepeat` program is a member of the WheelSong music composition suite of tools. Use it to loop through an input file some integer number of times.

OPTIONS

- h Display command help information.
- c <n>

Repeat the input file <n> times. (Default = 2)

-C loftfile
Repeat once for each note in the loftfile, transposing the repeat by the loft note value -I Display the interface of this command, in a format expected by discovery applications.

-q Quiet. Decrease the amount of verbosity in the output.

-v Verbose. Provide more detail in the output.

EXAMPLES

Lofting: with WSNB data = P60 P65 P70 and loft data = P0 P9 P-3
wsngrepeat -C loft.wsnb -i data.wsnb should produce:
P60 P65 P70 P69 P74 P79 P57 P62 P67

A.2.8 wsngharmonize

wsngharmonize [options] [--in1] infile [--out1] outfile

The wsngharmonize program is a member of the WheelSong music composition suite of tools. Use it to generate a sequence of chords constructed over the incoming melodic note sequence. This does NOT compute automatic harmonies. It creates the requested chord forms specified in the argument list.

OPTIONS

-a Render chords as descending arpeggios

-A Render chords as ascending arpeggios

-r Repeat chord sequence infinitely. (Defaults to off.)

-p <chord progression file>
A filename in WSNB format whose pitch values are interpreted as chord forms. These chord forms are applied in order to the root notes provided in the infile.

-h Display command help information.

-I Display the interface of this command, in a format expected by discovery applications.

-q Quiet. Decrease the amount of verbosity in the output.

-v Verbose. Provide more detail in the output.

Example: For file progression.wsnb containing "P0 P3 P5"
echo "P60 P62 P64 P65" | wsngharmonize -p progression.wsnb
will produce CMaj, Dmin7 and Eaug7, assuming pitch values are in MIDI form
Note that since the '-r' argument was not provided, no chord will be constructed over the F note.

The following chord forms can be specified by name or by integer value. (Names must be typed exactly as shown here.)

Maj (or 0)
min (or 1)
Maj7 (or 2)
min7 (or 3)
dom7 (or 4)
aug7 (or 5)
dim7 (or 6)
aug (or 7)
dim (or 8)

A.2.9 wsngbeatstuffer

wsngbeatstuffer [options] [--in1] infile [--out1] outfile

The wsngbeatstuffer program is a member of the WheelSong music composition suite of tools. Use it to conform the note events from one file (in wsnb format) to the beat structure of another file (also in wsnb format). The time values of events in the infile are taken as references to the beat number in beatdatafile.

-b <beatdatafile>

- Take the timing values from beatdatafile
- L Take the loudness info as well as timing (default = off).
- h Display command help information.
- I Display the interface of this command, in a format expected by discovery applications.
- q Quiet. Decrease the amount of verbosity in the output.
- v Verbose. Provide more detail in the output.

A.2.10 wsngstomp

wsngstomp [options] [--in1] infile [--out1] outfile

The wsngstomp program is a member of the WheelSong music composition suite of tools. Use it to import component values from one file (in wsnng format) into another file (also in wsnng format) overwriting whatever component values were in file2. A typical use of this tool is to apply timing values from one file to the melody of another. The output file will have the same number of note events as the input file.

OPTIONS

- s <stompdatafile>
Take the component values from stompdatafile as the new values
- c <components>
List of component field flags to import from stompdatafile
T=time, D=duration, P=pitch, S=slur, V=volume, C=volume out, X=voice
- h Display command help information.
- I Display the interface of this command, in a format expected by discovery applications.
- q Quiet. Decrease the amount of verbosity in the output.
- v Verbose. Provide more detail in the output.

A.2.11 wsnngmetronome

wsnngmetronome [options] [--out1] outfile

The wsnngmetronome program is a member of the WheelSong music composition suite of tools. Use it to generate a series of beat events in wsnng format. Beat events are like note events except they have no pitch value. They have an onset time and a duration. This can be used to, for example, specify quarter note events but to space them a half-note apart.

OPTIONS

- b num Set the begin-time value for the first beat (default = 1)
- d num Set the duration for each beat (default = 1)
Use multiple -d tags to specify a repeating sequence of durations.
- e num Set the end-time value for the last beat (default = 100)
- s num Set the spacing between beats (default = 0 Beats are adjacent)
- L num Set the loudness to use for each beat (default = unset)
- h Display command help information.
- I Display the interface of this command, in a format expected by discovery applications.
- q Quiet. Decrease the amount of verbosity in the output.
- v Verbose. Provide more detail in the output.

A.2.12 wsnngmetrify

wsnngmetrify [options] [--in1] infile [--out1] outfile

The wsnngmetrify program is a member of the WheelSong music composition suite of tools. Use it to impose cyclical dynamic values on beats in a file (in wsnng format) by multiplying by a rotating list of scalar values.

OPTIONS
 -P scalar
 -S scalar
 -T scalar
 -D scalar
 -V scalar
 -C scalar
 -X scalar
 -Y scalar
 Multiply channel A by scalar
 T=time, D=duration, P=pitch, S=slur, V=volume, C=volume out, X=voice, Y=2nd
 voice

 -h Display command help information.

 -I Display the interface of this command, in a format expected by
 discovery applications.

 -q Quiet. Decrease the amount of verbosity in the output.

 -v Verbose. Provide more detail in the output.

 Example: `wsngmetrify -V 1.5 -V 0.75 -V 1.0 -V 0.75 file.wsg`
 This will impose a Strong, Weak, MidStrong, Weak beat cadence in 4/4 time.

A.3 Evaluative

Operators that transform an input stream into some other musical or visual format.

A.3.1 `wsnt2wsg`

`wsnt2wsg` [options] infile outfile

The `wsnt2wsg` program is a member of the WheelSong music composition suite of tools. Use it to evaluate a WSNT file and render it into WSG format.

OPTIONS
 -h Display command help information.
 -I Display the interface of this command, in a format expected by
 discovery applications.
 -r <dirpath>
 Use dirpath as root for all temp files (Default = /tmp)
 -S Simulated only. Show commands that would be run but do not execute.
 -q Quiet. Decrease the amount of verbosity in the output.
 -v Verbose. Provide more detail in the output.

A.3.2 `wsngtomidi`

`wsngtomidi` [options] [--in1] infile [--out1] outfile

The `wsngtomidi` program is a member of the WheelSong music composition suite of tools. Use it to convert WSG files into MIDI format. Note that this is a lossy conversion. Much of the musical events that can be expressed in WSG have no corresponding representation in MIDI.

OPTIONS
 -h Display command help information.
 -x <j>
 Each occurrence sets the next MIDI instrument number to <j>
 First occurrence sets voice 0, second occurrence sets voice 1...
 There are a total of 16 possible voices.
 (All voices default to instrument 0.)

- D <n> Use MIDI instrument ID n as default voice.
- d Dump human readable note information, instead of MIDI binary
- q Quiet. Decrease the amount of verbosity in the output.
- v Verbose. Provide more detail in the output.

A.3.3 wsngtoly

wsngtoly [options] [--in1] infile [--out1] outfile

The wsngtoly program is a member of the WheelSong music composition suite of tools. Use it to convert WSNM files into Lilypond format. Note that this is a lossy conversion. Some of the musical events that can be expressed in WSNM have no corresponding representation in Lilypond.

This version is particularly sketchy, since it doesn't yet even support all of the WSNM file spec either.

OPTIONS

- h Display command help information.
- I Display the interface of this command, in a format expected by discovery applications.
- q Quiet. Decrease the amount of verbosity in the output.
- v Verbose. Provide more detail in the output.

A.3.4 wsnmpianoroll

wsnmpianoroll [options] [--in1] infile [--out1] outfile

The wsnmpianoroll program is a member of the WheelSong music composition suite of tools. Use it to convert WSNM files into PNM or PNG format displaying a piano roll notation of the music.

This version is particularly sketchy, since it doesn't yet even support all of the WSNM file spec either.

OPTIONS

- g Output in PNG format.
- m Output in PNM format (default).
- x Output in TXT format.
- a Display absolute note/pitch values (default)
- r Display relative or 'intervalic' note/pitch values
- c Display chromatic pitch-class values
- t n
Time resolution expressed in fractional beats. Default = 1.0
- h Display command help information.
- I Display the interface of this command, in a format expected by discovery applications.
- q Quiet. Decrease the amount of verbosity in the output.
- v Verbose. Provide more detail in the output.

A.3.5 wsnmvoiceflow

wsnmvoiceflow (options) infile outfile

Reads infile (in WSNM fmt) and outputs outfile (in Lilypond fmt)

There are two senses of time to keep straight. Time in the WSNM

Produces frame images. Integrate the frames with ffmpeg?

Convert all png output files to 8bit using: convert in -depth 8 out

Use something like: ffmpeg -f image2 -sameq -r 30 -i frame_%d.png video.mp4
file is unitless. Time in the output video is in seconds.

- l n Produce n frames of video for every 1 unit of WSNM time
- f n Produce n frames of video per video second (requires -s)
- s n Produce frames for n seconds at given fps rate (n is float)
- b n Render beginning at WSNM time unit n
- e n Stop rendering at WSNM time unit n

-k n Use n (in #FFF format) as background color (default = black)
-n Render an SVG snapshot of the video at starttime
-x Experiment. Dump frame info but don't output anything
-h Print this help
-? Print bad help

A.3.6 wsnt2dot

wsnt2dot [options] infile outfile

The wsnt2dot program is a member of the WheelSong music composition suite of tools. Use it to convert a WSNT file and render it into DOT format for use with the GraphViz structure visualization renderer.

OPTIONS

-h Display command help information.
-I Display the interface of this command, in a format expected by discovery applications.
-q Quiet. Decrease the amount of verbosity in the output.
-v Verbose. Provide more detail in the output.

APPENDIX B

COMPOSITIONS

WSNT-format scores for all compositions included in the *Wheelsongs* CD.

B.1 Resolving Our Voices

```
line1 = <P70T0D2X0 P65 P69D1 P68 P69D1V-0 P60D1 P63D2 P53 P53D4V-0> % 16
line2 = <P75T0D2V-0X1 P75D1.5 P75D0.5 P75D1V-0 P65 P65V-0 P65 P67D2 P46 P55D4 P55D2V
-0> %18
line3 = <P39T0D2X2 P39D3V-0 P60D1 P63 P63V-0 P70D2 P62D2 P63D4 P63D4V-0> %20
line4 = <P63T0D2X3 P71 P72D1 P72V-0 P58 P44 P46D2 P46V-0 P39D4 P39D14V-0> % 30
line5 = <P60T0D2V-0X4 P43 P44D1 P44V-0 P67 P56 P56D2V-0 P56 P56D12V-0> % 24
line6 = <P67T0D2X5 P67V-0 P63D1 P41 P43 P43D3V-0 P58D2 P58D4 P58D16V-0> %32

% add one to each loop so that it restates the theme again at the end
wsngrepeat [loop1] { -c { 91 } -i { line1 } }
wsngrepeat [loop2] { -c { 81 } -i { line2 } }
wsngrepeat [loop3] { -c { 73 } -i { line3 } }
wsngrepeat [loop4] { -c { 49 } -i { line4 } }
wsngrepeat [loop5] { -c { 61 } -i { line5 } }
wsngrepeat [loop6] { -c { 46 } -i { line6 } }

wsngsequence [singles] {
-i {line1}
-i {line2}
-i {line3}
-i {line4}
-i {line5}
-i {line6}
}

wsngselect [play] {
-s {1003}
%s {1430} % this line auditions just the final resolution
-i {
  wsnglayer [merge] {
    -i { loop1 }
    -i { loop2 }
    -i { loop3 }
    -i { loop4 }
    -i { loop5 }
    -i { loop6 }
  }
}
}}
```

B.2 The Lament

```
A = <P1D3 P5 P4 P3>
A2 = <P1D3 P5 P4 P5>
A3 = <P1D3 P3 P2 P1 >
```

```
wsngetrify [ThreeQuarterTime] {
  -i {
    wsngetronome [pulse] {
      -e { 200 }
      %-d { 1.0 } -d { 0.5 } -d { 0.5 } -d { 0.5 } % happy accident
      -d { 1.0 } -d { 0.5 } -d { 1 } -d { 0.5 } -d {0.5} -d {1} % happy
      accident
      %-d { 1.0 } % original
      -L { 200 }
    }
  }
  -V { 1.0 }
  -V { 0.35 }
  -V { 0.35 }
}
```

```
wsngetranspose [melody] {
  -n { 7 }
  -i {
    wsngetornament [orn] {
      -i {
        wsngetsequence [mels] {
          -i { A }
          -i { A2 }
          -i { A }
          -i { A3 }
        }
      }
      -P { "0,2,2" }
      -P { "0,-1,-1" }
      -P { "0,-1,-1" }
      -P { "0,-1,-1" }
      -P { "0,2,2" }
      -P { "0,-1,-1" }
      -P { "0,-1,-1" }
      -P { "0" }
      -P { "0,2,2" }
      -P { "0,-1,-1" }
      -P { "0,-1,-1" }
      -P { "0,-1,-1" }
      -P { "0,2,2" }
      -P { "2,-1,-1" }
      -P { "2,-1,-1" }
      -P { "0" }
    }
  }
}
```

```
wsngetsequence [overlay] {
  -s
  -i { mels }
  -i { melody }
}
```

```
wsngetsequence [intro] {
%  -i { mels }
  -i { overlay }
}
```

```
wsngetfromscale [theme] {
```

```

-s { wsngscale [dminor] {
    -t { 50 }
    -m { 0 } %1
  }
-i { intro }
}

wsngchanmult [arranged] {
-t { 1.0 }
-i {
  theme
}
}

wsngbeatstuffer [expressed] {
-b { ThreeQuarterTime }
-i { arranged }
}

% this is a dummy so that we can selectively play different structures
wsngtranspose [play] {
-n { 0 }
%-i { arranged }
-i { expressed }
}

```

B.3 Telco Jackhammer

```
% File A: ds2.wsnt
% File B: ds6.wsnt

% Orig fileB without a play token
wsngtranspose {
  -n {24}
  -i { wsngfromscale [Fnode39] {
    -s { wsngscale [Fnode38] {
      -m {5}
    }}
  }}
  -i { wsngsequence [Fnode2] {

    -i { wsngloft [Fnode26] {
      -C { wsngscale [Fnode25] {
        -m {8}
      }}
    }
    -i { wsngloft [Fnode23] {
      -C { wsngscale [Fnode22] {
        -m {7}
      }}
    }
    -i { wsngrepeat [Fnode20] {
      -c {2}
      -i { wsngtranspose [Fnode19] {
        -n {-1}
        -i { wsnginterpolate [Fnode18] {
          -n {0.226658}
          -g {0.617233}
          -i { wsngfromscale [Fnode17] {
            -s { wsngscale [Fnode16] {
              -m {1}
            }}
          }}
          -i { wsngloft [Fnode14] {
            -C { wsngrandom [Fnode13] {
              -S {51}
              -g {0.0725062}
              -r {0.452963}
            }}
            -i { wsnginterval [Fnode9] {
              -n {46}
              -i {-3}
            }}
          }}
        }}
      }}
    }}
  }}
  -i { wsngrandom [Fnode30] {
    -S {494}
    -g {0.0380116}
    -r {0.0335226}
  }}

  -i { wsngrandom [Fnode34] {
    -S {381}
    -g {0.151985}
    -r {0.177053}
  }}

  -i { wsngscale [Fnode36] {
    -m {6}
  }}
}
```

```

    }}
  }}
}

% Beginning of FileA
SubFrag1A = <P4.0D1 P4.143 P4.286 P4.429>
SubFrag1B = <P4.143D1 P4.286 P4.0>
Frag2 = <P4.571D2 P5.0 P4.857 P5.0>
Frag4 = <P3.571D2 P2.571> % drop octave eighths

% create some minor fragments
PauseOneSixteenth = <P4.0D1V-0>
PauseOneHalf = <P4.0D8V-0>
PauseOneQuarter = <P4.0D4V-0>

wsngsequence [Frag1] {
  -i { PauseOneSixteenth }
  -i { SubFrag1A }
  -i { SubFrag1B }
}

wsngsequence [bar1treble] {
  -i { Frag1 }
  -i { Frag2 }
}

wsngtranspose [bar1bass] {
  -n {-1.0}
  -i {
    wsngsequence {
      -i { PauseOneHalf }
      -i { Frag1 }
    }
  }
}

wsngtranspose [bar2treble] {
  -n { 0.571 }
}

% New tree from FileB with token declarations removed
-i {
  wsngloft {
    -C {
      wsngscale {
        -m {8}
      }
    }
    -i {
      wsngloft {
        -C {
          wsngscale {
            -m {7}
          }
        }
        -i {
          wsngrepeat {
            -c {2}
            -i {
              wsngtranspose {
                -n {-1}
                -i {
                  wsnginterpolate {
                    -n {0.226658}
                    -g {0.617233}
                    -i {
                      wsngfromscale {
                        wsngscale {
                          -s {
                            -m {1}
                          }
                        }
                      }
                    }
                    -i {
                      wsngloft {
                        -C {
                          wsngrandom {
                            -S {51}
                            -g {0.0725062}
                            -r {0.452963}
                          }
                        }
                      }
                    }
                    -i {
                      wsnginterval {
                        -n {46}
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}

```

```

-i {-3}
}} }} }} }} }} }} }} }} }}
% End of FileA
}

wsngsequence [bar2bass] {
-i { Frag4 }
-i { PauseOneQuarter }
-i { wsngtranspose {
-n { 0.571 }
-i { Frag1 }
} } }

wsngsequence [line1treble] {
-i { bar1treble }
-i { bar2treble }
}
wsngsequence [line1bass] {
-i { bar1bass }
-i { bar2bass }
}

wsngsequence [treble] { -i { line1treble } }
wsngsequence [bass] { -i { line1bass } }

wsngchanmult [play] {
-t { 0.6 }
-i { wsngselect {
-p
-s { 0.0 }
-e { 1000.0 }
-i { wsngfromscale {
-f
-s { wsngscale { -m {0} } }
-i {
wsngsequence {
-s
-i { treble }
-i { bass }
} } } } } } }
} } } } } } }

```

B.4 Invention Sketch

```

% The identified patterns
green = <P22D1 P23 P24 P25>
dblue = <P23D1 P24 P22>
orange = <P26D2 P29 P28 P29>
red = <P22D2 P15>
brown = <P27D2 P23>
dgrey = <P28D0.5 P29 P30D1>
lgrey = <P29D3 P30D1>

wsngsequence [purple] { -i {green} -i {dblue}}
wsngsequence [lblue] { -i {purple} -i {orange}}
wsngselect [pink] { -e {3} -i {green}}
wsngchanmult [doublegreen] { -t {2} -i {green}}
wsngchanmult [doubledblue] { -t {2} -i {dblue}}
wsnginvert [greeni] {-i {green}}
wsngchanmult [doublegreeni] { -t {2} -i {greeni}}

% some useful rests
pause16 = <P1D1V-0>
pause8 = <P1D2V-0>
pause6 = <P1D6V-0>
pause4 = <P1D4V-0>
pause2 = <P1D8V-0>

% some useful intervals and sequences
down5 = <P0D1 P4>
climb7 = <P4 P5>
lowclimb = <P-7 P-3>
lowdrop = <P-1 P-3 P-5>
lowdrop7 = <P-10 P-6>
lowclimb9 = <P5 P6>
lowdrop11 = <P6 P4 P2 P0>
greendrop = <P7 P5 P3>
drop = <P12 P10 P8 P6>

wsngsequence [lbluepat] {-i {pause16} -i {lblue}}
wsngsequence [lbluepati] {-i {pause16} -i {wsnginvert {-i {lblue}}}}
wsngsequence [purplepat] {-i {pause16} -i {purple}}
wsngsequence [purplepati] {-i {pause16} -i {wsnginvert {-i {purple}}}}
wsngretrograde [dbluer] {-i {dblue}}
wsnginvert [dbluei] {-i {dblue}}
wsnginvert [dblueri] {-i {dbluer}}
wsngsequence [pausepurple] {-i {pause2} -i {purplepat}}
wsnginvert [pausepurplei] {-i {pausepurple}}
wsngtranspose [climb9] { -n {46} -i {climb7}}
wsngchanmult [greendouble] { -t {2} -i {green}}

% the final chord
wsngchanmult [chord] {
  -T { 0 }
  -D { 4 }
  -i { wsngsequence {
    -i { wsngtranspose { -n {-7} -i {red}}}
    -i { brown }
  } }}

% assembling the treble patterns
%
wsngsequence [treble] {
  % bar 1-2
  -i { wsngloft { -C {down5} -i {lbluepat}} }

  % bar 3-4
  -i { wsngloft [bar2t] { -C {drop} -i {purplepati}} }

```

```

% bar 5
-i { brown }
-i { lgrey }
-i { wsngrtranspose { -n {5} -i {purplepati} }}

% bar 6
-i { wsngrtranspose { -n {4} -i {dbluer} }}
-i { wsngrtranspose { -n {6} -i {dbluei} }}
-i { wsngrtranspose { -n {7} -i {dbluer} }}
-i { dgrey }
-i { pause16 }
-i { wsngrtranspose { -n {-2} -i {wsngretrograde [dgreyr2] { -i {wsngchanmult { -t
{2} -i {dgrey} } } } } }

% bar 7-8
-i { wsnглоft { -C {climb7} -i {pausepurple} }}

% bar 9-10
-i { wsnглоft { -C {climb9} -i {pausepurplei} }}

% bar 11-12
-i { pause8 }
-i { wsnглоft { -C {greendrop} -i {greendouble} }}
-i { pause6 }

% bar 13
-i { wsngrtranspose { -n {2} -i {purplepat} }}
-i { pause16 }
-i { wsngrtranspose { -n {7} -i {dbluei} }}
-i { pause16 }
-i { wsngrtranspose { -n {6} -i {dbluei} }}

% bar 14
-i { pause8 }
-i { wsngrtranspose { -n {11} -i {dbluer} }}
-i { wsngrtranspose { -n {8} -i {dblue} }}
-i { pause16 }
-i { wsngrtranspose { -n {8} -i {wsngretrograde { -i {pink} } } } }
-i { wsngrtranspose { -n {-1} -i {dgreyr2} }}

% bar 15-16
-i { wsngrtranspose { -n {12} -i {purplepati} }}
-i { pause2 }
-i { wsngrtranspose { -n {9} -i {purplepat} }}
-i { pause2 }

% bar 17-19
-i { wsngrtranspose { -n {11} -i {purplepati} }}
-i { pause2 }
-i { wsngrtranspose { -n {8} -i {purplepat} }}
-i { pause2 }
-i { wsngrtranspose { -n {7} -i {purplepat} }}
-i { wsngrtranspose { -n {9} -i {purplepat} }}

% bar 20-22
-i { wsngrtranspose { -n {11} -i {purplepat} }}
-i { pause4 }
-i { wsngrtranspose { -n {1} -i {dgreyr2} }}
-i { wsngrtranspose { -n {6} -i {purplepati} }}
-i { wsngrtranspose { -n {5} -i {pink} }}
-i { pause8 }
-i { wsngrtranspose { -n {2} -i {wsngchanmult [brownh] { -t {0.5} -i {brown} } } } }

% assembling the bass patterns
%
wsngsequence [bass] {

```



```

% bar 1-2
%-i { wsnngloft { -C {lowclimb} -i {pausepurple}}}
-i { pause2 }
-i { wsnngtranspose { -n {-7} -i {purplepat}}}
-i { wsnngtranspose { -n {-3} -i {red}}}
-i { pause4 }
-i { wsnngtranspose { -n {-3} -i {purplepat}}}

% bar 3-4
-i {pause8}
-i { wsnngloft { -C {lowdrop} -i {doublegreen}}}
-i {pause6}

% bar 5-6
-i { wsnngtranspose { -n {-6} -i {purplepat}}}
-i {pause4}
-i { wsnngtranspose { -n {-7} -i {doublegreen}}}
-i {pause4}
-i { wsnngtranspose { -n {-15} -i {lgrey}}}
-i { wsnngtranspose { -n {-6} -i {red}}}

% bar 7-8
-i { wsnngloft { -C {lowdrop7} -i {lbluepat}}}

% bar 9-10
-i { wsnngtranspose { -n {4} -i {purplepati}}}
-i {pause8}
-i { wsnngtranspose { -n {1} -i {doubledblue}}}
-i { wsnngtranspose { -n {5} -i {purplepati}}}
-i {pause8}
-i { wsnngtranspose { -n {2} -i {doubledblue}}}

% bar 11-12
-i { wsnngloft { -C {lowdrop11} -i {purplepati}}}

% bar 13-14
-i { wsnngtranspose { -n {-6} -i {brown}}}
-i { wsnngtranspose { -n {-6} -i {lgrey}}}
-i { greeni }
-i {pause16}
-i { wsnngtranspose { -n {-3} -i {dbluei}}}
-i { wsnngtranspose { -n {-2} -i {dbluer}}}
-i { dbluei }
-i { wsnngtranspose { -n {1} -i {dbluer}}}
-i {pause16}
-i { wsnngtranspose { -n {-3} -i {wsngretrograde [brownr] { -i {brown}}}}}
-i {pause8}

% bar 15-16
-i { wsnngtranspose { -n {-2} -i {red}}}
-i {pause4}
-i { wsnngtranspose { -n {2} -i {purplepati}}}
-i {pause2}
-i { wsnngtranspose { -n {-2} -i {purplepat}}}

% bar 17-19
-i {pause2}
-i { purplepati }
-i {pause2}
-i { wsnngtranspose { -n {-3} -i {purplepat}}}
-i {pause8}
-i { wsnngtranspose { -n {-1} -i {doublegreeni}}}
-i { wsnngtranspose { -n {1} -i {wsnginvert [doublepink] { -i { wsnngchanmult [
doublepink] { -t {2} -i {pink} }}}}}}

% bar 20-21
-i {pause8}
-i { wsnngtranspose { -n {3} -i {doublepink}}}

```

```

-i { wsngrtranspose { -n {-6} -i {purplepat}}}
-i {pause8}
-i { wsngrtranspose { -n {-7} -i {doublepink}}}
-i {pause16}
-i { wsngrtranspose { -n {-6} -i {pink}}}
-i { wsngrtranspose { -n {-3} -i {red}}}

% bar 22
-i { chord }
}

wsngchanmult [play] {
-t {0.6} % Set the final tempo
-i {
  wsngrfromscale { % set the final note palette
    -s { wsngrscale { -m {0}}}
    -i {
      wsngrsequence {
        -s
        -i { treble }
        -i { bass }
      } } } } }
}

```

B.5 Tokyo Invention

```

% The identified patterns
%green = <P22D1 P23 P24 P25>
%green = <P22D1 P23 P24D0.5 P23 P24D1>
green = <P22D1 P21D0.5 P20 P19 P22 P22 P19>

%dblue = <P23D1 P24 P22>
%dblue = <P22D1 P22 P20>
dblue = <P26D1 P22D2>

%orange = <P26D2 P29 P28 P29>
%orange = <P24D2 P22 P22 P24>
orange = <P24D2 P24 P25 P25>

red = <P22D2 P15>
brown = <P27D2 P23>
dgrey = <P28D0.5 P29 P30D1>
lgrey = <P29D3 P30D1>

wsngsequence [purple] { -i {green} -i {dblue}}
wsngsequence [lblue] { -i {purple} -i {orange}}
wsngselect [pink] { -e {3} -i {green}}
wsngchanmult [doublegreen] { -t {2} -i {green}}
wsngchanmult [doubledblue] { -t {2} -i {dblue}}
wsnginvert [greeni] {-i {green}}
wsngchanmult [doublegreeni] { -t {2} -i {greeni}}

% some useful rests
pause16 = <P1D1V-0>
pause8 = <P1D2V-0>
pause6 = <P1D6V-0>
pause4 = <P1D4V-0>
pause2 = <P1D8V-0>

% some useful intervals and sequences
down5 = <P0D1 P4>
climb7 = <P4 P5>
lowclimb = <P-7 P-3>
lowdrop = <P-1 P-3 P-5>
lowdrop7 = <P-10 P-6>
lowclimb9 = <P5 P6>
lowdrop11 = <P6 P4 P2 P0>
greendrop = <P7 P5 P3>
drop = <P12 P10 P8 P6>

wsngsequence [lbluepat] {-i {pause16} -i {lblue}}
wsngsequence [lbluepati] {-i {pause16} -i {wsnginvert {-i {lblue}}}}
wsngsequence [purplepat] {-i {pause16} -i {purple}}
wsngsequence [purplepati] {-i {pause16} -i {wsnginvert {-i {purple}}}}
wsngretrograde [dbluer] {-i {dblue}}
wsnginvert [dbluei] {-i {dblue}}
wsnginvert [dblueri] {-i {dbluer}}
wsngsequence [pausepurple] {-i {pause2} -i {purplepat}}
wsnginvert [pausepurplei] {-i {pausepurple}}
wsngtranspose [climb9] { -n {46} -i {climb7}}
wsngchanmult [greendouble] { -t {2} -i {green}}

% the final chord
wsngchanmult [chord] {
  -T { 0 }
  -D { 4 }
  -i { wsngsequence {
    -i { wsngtranspose { -n {-7} -i {red}}}
    -i { brown }
  }
}

```

```

}}
% assembling the treble patterns
%
wsngsequence [treble] {
  % bar 1-2
  -i { wsngloft { -C {down5} -i {lbluepat} } }

  % bar 3-4
  -i { wsngloft [bar2t] { -C {drop} -i {purplepati} }}

  % bar 5
  -i { brown }
  -i { lgrey }
  -i { wsngtranspose { -n {5} -i {purplepati} }}

  % bar 6
  -i { wsngtranspose { -n {4} -i {dbluer} }}
  -i { wsngtranspose { -n {6} -i {dbluei} }}
  -i { wsngtranspose { -n {7} -i {dbluer} }}
  -i { dgrey }
  -i { pause16 }
  -i { wsngtranspose { -n {-2} -i {wsngretrograde [dgreyr2] { -i {wsngchanmult { -t
    {2} -i {dgrey} } } } } } }

  % bar 7-8
  -i { wsngloft { -C {climb7} -i {pausepurple} }}

  % bar 9-10
  -i { wsngloft { -C {climb9} -i {pausepurplei} }}

  % bar 11-12
  -i { pause8 }
  -i { wsngloft { -C {greendrop} -i {greendouble} }}
  -i { pause6 }

  % bar 13
  -i { wsngtranspose { -n {2} -i {purplepat} }}
  -i { pause16 }
  -i { wsngtranspose { -n {7} -i {dbluei} }}
  -i { pause16 }
  -i { wsngtranspose { -n {6} -i {dbluei} }}

  % bar 14
  -i { pause8 }
  -i { wsngtranspose { -n {11} -i {dbluer} }}
  -i { wsngtranspose { -n {8} -i {dbluei} }}
  -i { pause16 }
  -i { wsngtranspose { -n {8} -i {wsngretrograde { -i {pink} } } } }
  -i { wsngtranspose { -n {-1} -i {dgreyr2} }}

  % bar 15-16
  -i { wsngtranspose { -n {12} -i {purplepati} }}
  -i { pause2 }
  -i { wsngtranspose { -n {9} -i {purplepat} }}
  -i { pause2 }

  % bar 17-19
  -i { wsngtranspose { -n {11} -i {purplepati} }}
  -i { pause2 }
  -i { wsngtranspose { -n {8} -i {purplepat} }}
  -i { pause2 }
  -i { wsngtranspose { -n {7} -i {purplepat} }}
  -i { wsngtranspose { -n {9} -i {purplepat} }}

  % bar 20-22
  -i { wsngtranspose { -n {11} -i {purplepat} }}
  -i { pause4 }

```

```

-i { wsngrtranspose { -n {1} -i {dgreyr2}}}
-i { wsngrtranspose { -n {6} -i {purplepati}}}
-i { wsngrtranspose { -n {5} -i {pink}}}
-i {pause8}
-i { wsngrtranspose { -n {2} -i {wsngchanmult [brownh] { -t {0.5} -i {brown}}}}}
}

% assembling the bass patterns
%
wsngsequence [bass] {

% bar 1-2
-i { wsngrloft { -C {lowclimb} -i {pausepurple}}}
-i { pause2 }
-i { wsngrtranspose { -n {-7} -i {purplepat}}}
-i { wsngrtranspose { -n {-3} -i {red}}}
-i { pause4 }
-i { wsngrtranspose { -n {-3} -i {purplepat}}}

% bar 3-4
-i {pause8}
-i { wsngrloft { -C {lowdrop} -i {doublegreen}}}
-i {pause6}

% bar 5-6
-i { wsngrtranspose { -n {-6} -i {purplepat}}}
-i {pause4}
-i { wsngrtranspose { -n {-7} -i {doublegreen}}}
-i {pause4}
-i { wsngrtranspose { -n {-15} -i {lgrey}}}
-i { wsngrtranspose { -n {-6} -i {red}}}

% bar 7-8
-i { wsngrloft { -C {lowdrop7} -i {lbluepat}}}

% bar 9-10
-i { wsngrtranspose { -n {4} -i {purplepati}}}
-i {pause8}
-i { wsngrtranspose { -n {1} -i {doubledblue}}}
-i { wsngrtranspose { -n {5} -i {purplepati}}}
-i {pause8}
-i { wsngrtranspose { -n {2} -i {doubledblue}}}

% bar 11-12
-i { wsngrloft { -C {lowdrop11} -i {purplepati}}}

% bar 13-14
-i { wsngrtranspose { -n {-6} -i {brown}}}
-i { wsngrtranspose { -n {-6} -i {lgrey}}}
-i { greeni }
-i {pause16}
-i { wsngrtranspose { -n {-3} -i {dbluei}}}
-i { wsngrtranspose { -n {-2} -i {dbluer}}}
-i { dbluei }
-i { wsngrtranspose { -n {1} -i {dbluer}}}
-i {pause16}
-i { wsngrtranspose { -n {-3} -i {wsngretrograde [brownr] { -i {brown}}}}}
-i {pause8}

% bar 15-16
-i { wsngrtranspose { -n {-2} -i {red}}}
-i {pause4}
-i { wsngrtranspose { -n {2} -i {purplepati}}}
-i {pause2}
-i { wsngrtranspose { -n {-2} -i {purplepat}}}

% bar 17-19
-i {pause2}

```

```

-i { purplepati }
-i {pause2}
-i { wsngrtranspose { -n {-3} -i {purplepat}}}
-i {pause8}
-i { wsngrtranspose { -n {-1} -i {doublegreeni}}}
-i { wsngrtranspose { -n {1} -i {wsnginvert [doublepink] { -i { wsnchanmult [
  doublepink] { -t {2} -i {pink} }}}}}}

% bar 20-21
-i {pause8}
-i { wsngrtranspose { -n {3} -i {doublepink}}}
-i { wsngrtranspose { -n {-6} -i {purplepat}}}
-i {pause8}
-i { wsngrtranspose { -n {-7} -i {doublepink}}}
-i {pause16}
-i { wsngrtranspose { -n {-6} -i {pink}}}
-i { wsngrtranspose { -n {-3} -i {red}}}

% bar 22
-i { chord }
}

wsnchanmult [play] {
-t {0.6} % Set the final tempo
-i { wsngrtranspose { -n {-36}
  -i { wsngrfromscale { % set the final note palette
    -s { wsngrscale { -m {6}}}
    -i { wsngrsequence {
      -s
      -i { treble }
      -i { bass }
    } } } } } } } } } } } }

```

B.6 Bach Alley Shuffle

```

% The identified patterns
%green = <P22D1 P23 P24 P25>
%green = <P22D1 P23 P24D0.5 P23 P24D1>
green = <P29D0.5 P22 P21D0.5 P20D1 P19D0.5 P22D1>

%dblue = <P23D1 P24 P22>
%dblue = <P22D1 P22 P20>
dblue = <P26D1 P22D2>

%orange = <P26D2 P29 P28 P29>
%orange = <P24D2 P22 P22 P24>
orange = <P24D2 P24 P25 P25>

red = <P22D2 P15>
brown = <P27D2 P23>
dgrey = <P28D0.5 P29 P30D1>
lgrey = <P29D3 P30D1>

wsnginvert [greeni] {-i {green}}
wsngretrograde [dbluer] {-i {dblue}}
wsnginvert [dbluei] {-i {dblue}}
wsnginvert [dblueri] {-i {dbluer}}
wsngsequence [purple] {-i {dblueri} -i {greeni} }
wsngsequence [lblue] {-i {purple} -i {orange}}
wsngselect [pink] {-e {3} -i {greeni}}
wsngchanmult [doublegreen] {-t {2} -i {greeni}}
wsngchanmult [doubledblue] {-t {2} -i {dbluei}}
wsngchanmult [doublegreeni] {-t {2} -i {greeni}}

% some useful rests
pause16 = <P1D1V-0>
pause8 = <P1D2V-0>
pause6 = <P1D6V-0>
pause4 = <P1D4V-0>
pause2 = <P1D8V-0>

% some useful intervals and sequences
down5 = <P0D1 P4>
climb7 = <P4 P5>
lowclimb = <P-7 P-3>
lowdrop = <P-1 P-3 P-5>
lowdrop7 = <P-10 P-6>
lowclimb9 = <P5 P6>
lowdrop11 = <P6 P4 P2 P0>
greendrop = <P7 P5 P3>
drop = <P6 P8 P10 P12>

wsngsequence [lbluepat] {-i {pause16} -i {lblue}}
wsngsequence [lbluepati] {-i {pause16} -i {wsnginvert {-i {lblue}}}}
wsngsequence [purplepat] {-i {pause16} -i {purple}}
wsngsequence [purplepati] {-i {pause16} -i {wsnginvert {-i {purple} }}}
wsngsequence [pausepurple] {-i {pause2} -i {purplepat}}
wsnginvert [pausepurplei] {-i {pausepurple}}
wsngtranspose [climb9] {-n {46} -i {climb7}}
wsngchanmult [greendouble] {-t {2} -i {green}}

% the final chord
wsngchanmult [chord] {
  -T { 0 }
  -D { 4 }
  -i { wsngsequence {
    -i { wsngtranspose { -n {-7} -i {red} }}
    -i { brown }
  } }
}

```

```

% assembling the treble patterns
%
wsgsequence [treble] {
  % bar 1-2
  -i { wsgloft { -C {down5} -i {lbluepat}} }

  % bar 3-4
  -i { wsgloft [bar2t] { -C {drop} -i {purplepati} }}

  % bar 5
  -i { brown }
  -i { lgrey }
  -i { wsgtranspose { -n {5} -i {purplepati} }}

  % bar 6
  -i { wsgtranspose { -n {4} -i {dbluer}}}
  -i { wsgtranspose { -n {6} -i {dbluei}}}
  -i { wsgtranspose { -n {7} -i {dbluer}}}
  -i { dgrey }
  -i { pause16 }
  -i { wsgtranspose { -n {-2} -i {wsgretrograde [dgreyr2] { -i {wsgchanmult { -t
    {2} -i {dgrey} }}}} }}

  % bar 7-8
  -i { wsgloft { -C {climb7} -i {pausepurple}}}

  % bar 9-10
  -i { wsgloft { -C {climb9} -i {pausepurplei}}}

  % bar 11-12
  -i { pause8 }
  -i { wsgloft { -C {greendrop} -i {greendouble}}}
  -i { pause6 }

  % bar 13
  -i { wsgtranspose { -n {2} -i {purplepat}}}
  -i { pause16 }
  -i { wsgtranspose { -n {7} -i {dbluei}}}
  -i { pause16 }
  -i { wsgtranspose { -n {6} -i {dbluei}}}

  % bar 14
  -i { pause8 }
  -i { wsgtranspose { -n {11} -i {dbluer}}}
  -i { wsgtranspose { -n {8} -i {dblue}}}
  -i { pause16 }
  -i { wsgtranspose { -n {8} -i {wsgretrograde { -i {pink}}}}}
  -i { wsgtranspose { -n {-1} -i {dgreyr2}}}

  % bar 15-16
  -i { wsgtranspose { -n {12} -i {purplepati}}}
  -i { pause2 }
  -i { wsgtranspose { -n {9} -i {purplepat}}}
  -i { pause2 }

  % bar 17-19
  -i { wsgtranspose { -n {11} -i {purplepati}}}
  -i { pause2 }
  -i { wsgtranspose { -n {8} -i {purplepat}}}
  -i { pause2 }
  -i { wsgtranspose { -n {7} -i {purplepat}}}
  -i { wsgtranspose { -n {9} -i {purplepat}}}

  % bar 20-22
  -i { wsgtranspose { -n {11} -i {purplepat}}}
  -i { pause4 }
  -i { wsgtranspose { -n {1} -i {dgreyr2}}}

```



```

-i { wsngrtranspose { -n {6} -i {purplepati}}}
-i { wsngrtranspose { -n {5} -i {pink}}}
-i {pause8}
-i { wsngrtranspose { -n {2} -i {wsngchanmult [brownh] { -t {0.5} -i {brown}}}}}}

% assembling the bass patterns
%
wsngsequence [bass] {

% bar 1-2
%-i { wsngrloft { -C {lowclimb} -i {pausepurple}}}
-i { pause2 }
-i { wsngrtranspose { -n {-7} -i {purplepat}}}
-i { wsngrtranspose { -n {-3} -i {red}}}
-i { pause4 }
-i { wsngrtranspose { -n {-3} -i {purplepat}}}

% bar 3-4
-i {pause8}
-i { wsngrloft { -C {lowdrop} -i {doublegreen}}}
-i {pause6}

% bar 5-6
-i { wsngrtranspose { -n {-6} -i {purplepat}}}
-i {pause4}
-i { wsngrtranspose { -n {-7} -i {doublegreen}}}
-i {pause4}
-i { wsngrtranspose { -n {-15} -i {lgrey}}}
-i { wsngrtranspose { -n {-6} -i {red}}}

% bar 7-8
-i { wsngrloft { -C {lowdrop7} -i {lbluepat}}}

% bar 9-10
-i { wsngrtranspose { -n {4} -i {purplepati}}}
-i {pause8}
-i { wsngrtranspose { -n {1} -i {doubledblue}}}
-i { wsngrtranspose { -n {5} -i {purplepati}}}
-i {pause8}
-i { wsngrtranspose { -n {2} -i {doubledblue}}}

% bar 11-12
-i { wsngrloft { -C {lowdrop11} -i {purplepati}}}

% bar 13-14
-i { wsngrtranspose { -n {-6} -i {brown}}}
-i { wsngrtranspose { -n {-6} -i {lgrey}}}
-i { greeni }
-i {pause16}
-i { wsngrtranspose { -n {-3} -i {dbluei}}}
-i { wsngrtranspose { -n {-2} -i {dbluer}}}
-i { dbluei }
-i { wsngrtranspose { -n {1} -i {dbluer}}}
-i {pause16}
-i { wsngrtranspose { -n {-3} -i {wsngretrograde [brownr] { -i {brown}}}}}}
-i {pause8}

% bar 15-16
-i { wsngrtranspose { -n {-2} -i {red}}}
-i {pause4}
-i { wsngrtranspose { -n {2} -i {purplepati}}}
-i {pause2}
-i { wsngrtranspose { -n {-2} -i {purplepat}}}

% bar 17-19
-i {pause2}
-i { purplepati }
-i {pause2}

```

```

-i { wsngrtranspose { -n {-3} -i {purplepat}}}
-i {pause8}
-i { wsngrtranspose { -n {-1} -i {doublegreeni}}}
-i { wsngrtranspose { -n {1} -i {wsnginvert [doublepink] { -i { wsngrchanmult [
  doublepink] { -t {2} -i {pink} ]}}}}}

% bar 20-21
-i {pause8}
-i { wsngrtranspose { -n {3} -i {doublepink}}}
-i { wsngrtranspose { -n {-6} -i {purplepat}}}
-i {pause8}
-i { wsngrtranspose { -n {-7} -i {doublepink}}}
-i {pause16}
-i { wsngrtranspose { -n {-6} -i {pink}}}
-i { wsngrtranspose { -n {-3} -i {red}}}

% bar 22
-i { chord }
}

```

```

wsngrchanmult [play] {
  -t {0.7} % Set the final tempo
  -i {
wsngrtranspose { -n {-15} -i {
  wsngrfromscale { % set the final note palette
    -s { wsngrscale { -m {8}}}
    -i {
      wsngrsequence {
        -s
        -i { treble }
        -i { bass }
      }}}}
  }}}}
}

```

B.7 Morning Traffic

```
% This is the melody from the first 4 bars of the Cello Suite
scale = <P43T0D0.8 P50T0.8D0.8 P59T1.6D0.8 P57T2.4D0.8 P59T3.2D0.8 P50T4D0.8 P59T4.8
D0.8 P50T5.6D0.8 P43T6.4D0.8 P50T7.2D0.8 P59T8D0.8 P57T8.8D0.8 P59T9.6D0.8
P50T10.4D0.8 P59T11.2D0.8 P50T12D0.8 P43T12.8D0.8 P52T13.6D0.8 P60T14.4D0.8
P59T15.2D0.8 P60T16D0.8 P52T16.8D0.8 P60T17.6D0.8 P52T18.4D0.8 P43T19.2D0.8
P52T20D0.8 P60T20.8D0.8 P59T21.6D0.8 P60T22.4D0.8 P52T23.2D0.8 P60T24D0.8 P52T24
.8D0.8 P43T25.6D0.8 P53T26.4D0.8 P60T27.2D0.8 P59T28D0.8 P60T28.8D0.8 P53T29.6D0
.8 P60T30.4D0.8 P53T31.2D0.8 P43T32D0.8 P53T32.8D0.8 P60T33.6D0.8 P59T34.4D0.8
P60T35.2D0.8 P53T36D0.8 P60T36.8D0.8 P53T37.6D0.8 P43T38.4D0.8 P55T39.2D0.8
P59T40D0.8 P57T40.8D0.8 P59T41.6D0.8 P55T42.4D0.8 P59T43.2D0.8 P55T44D0.8 P43T44
.8D0.8 P55T45.6D0.8 P59T46.4D0.8 P57T47.2D0.8 P59T48D0.8 P55T48.8D0.8 P59T49.6D0
.8 P55T50.4D0.8>

simplerscale = <P43 P50 P52 P53 P55 P57 P59 P60>

SubFrag1A = <P4.0D1 P4.143 P4.286 P4.429>
SubFrag1B = <P4.143D1 P4.286 P4.0>
Frag2 = <P4.571D2 P5.0 P4.857 P5.0>
Frag4 = <P3.571D2 P2.571> % drop octave eighths
Frag5 = <P5.0D3 P5.143D1>
Frag6 = <P4.857D0.5 P5.0 P5.143D1>
Frag7 = <P4.714D2 P4.143>
CChord1st = <P5.0D16T-1 P4.286T-1 P4.571>
COct = <P3.0D16T-1 P2.0>

% create some minor fragments
PauseOneSixteenth = <P4.0D1V-0>
PauseOneEighth = <P4.0D2V-0>
PauseOneHalf = <P4.0D8V-0>
PauseOneQuarter = <P4.0D4V-0>
PauseNineSixteenths = <P4.0D9V-0>

wsngsequence [Frag1] {
  -i { PauseOneSixteenth }
  -i { SubFrag1A }
  -i { SubFrag1B }
}

wsnginvert [Frag1Inv] {
  -i {Frag1}
}

wsngchanmult [Frag3] {
  -t { 2.0 }
  -i { SubFrag1A }
}

wsngsequence [Frag8] {
  -i { Frag7 }
  -i { Frag5 }
}

wsngsequence [bar1treble] {
  -i { Frag1 }
  -i { Frag2 }
}

wsngtranspose [bar1bass] {
  -n {-1.0}
  -i { wsngsequence {
    -i { PauseOneHalf }
    -i { Frag1 }
  } }
}

wsngtranspose [bar2treble] {
```

```

    -n { 0.571 }
    -i { bar1treble }
}

wsngsequence [bar2bass] {
    -i { Frag4 }
    -i { PauseOneQuarter }
    -i { wsngtranspose {
        -n { 0.571 }
        -i { Frag1 }
    } } }

wsngtranspose [bar3a] {
    -n { 1.714 }
    -i { Frag1Inv }
}
wsngtranspose [bar3b] {
    -n { 1.429 }
    -i { Frag1Inv }
}
wsngtranspose [bar4a] {
    -n { 1.143 }
    -i { Frag1Inv }
}
wsngtranspose [bar4b] {
    -n { 0.857 }
    -i { Frag1Inv }
}

wsngsequence [line1treble] {
    -i { bar1treble }
    -i { bar2treble }
}
wsngsequence [line1bass] {
    -i { bar1bass }
    -i { bar2bass }
}

wsngsequence [line2treble] {
    -i { bar3a }
    -i { bar3b }
    -i { bar4a }
    -i { bar4b }
}

wsngtranspose [bar3abass] {
    -n { -0.143 }
    -i { Frag3 }
}

wsngtranspose [bar3bbass] {
    -n { -0.429 }
    -i { Frag3 }
}

wsngtranspose [bar4abass] {
    -n { -0.714 }
    -i { Frag3 }
}

wsngsequence [line2bass] {
    -i { PauseOneEighth }
    -i { bar3abass }
    -i { bar3bbass }
    -i { bar4abass }
    -i { PauseOneEighth } % bar4b is empty
    -i { PauseOneQuarter }
}

```

```

}

wsngsequence [line3treble] {
  -i { Frag8 }
  -i { wsngtranspose {
    -n { 0.714 }
    -i { Frag1Inv }
  } }
  -i { wsngtranspose {
    -n { 0.571 }
    -i {
      wsngretrograde {
        -i {SubFrag1B }
      } } }
  -i { wsngtranspose {
    -n { 0.857 }
    -i { wsnginvert {
      -i {SubFrag1B }
    } } }
  -i { wsngtranspose {
    -n { 1.0 }
    -i { wsngretrograde {
      -i {SubFrag1B }
    } } }
  -i { Frag6 }
  -i { PauseOneSixteenth }
  -i { wsngtranspose {
    -n { -0.286 }
    -i { wsngretrograde {
      -i { wsngchanmult {
        -t {2.0}
        -i { Frag6 }
      } } } } }
}

wsngsequence [line3bass] {
  -i { wsngtranspose {
    -n { -0.857 }
    -i { Frag1 }
  } }
  -i { PauseOneEighth }
  -i { wsngtranspose {
    -n { -1.143 }
    -i { Frag3 }
  } }
  -i { wsngtranspose {
    -n { -0.714 }
    -i { wsngchanmult {
      -t {2.0}
      -i { SubFrag1B }
    } } }
  -i { wsngtranspose {
    -n { -2.143 }
    -i { Frag5 }
  } }
  -i { wsngtranspose {
    -n { -0.429 }
    -i { Frag4 }
  } }
}

wsngsequence [line4treble] {
  -i { PauseOneHalf }
  -i { wsngtranspose {
    -n { 0.571 }
    -i { Frag1 }
  } }
  -i { PauseOneHalf }
  -i { wsngtranspose {

```

```

        -n { 0.714 }
        -i { Frag1 }
    } } }

wsngsequence [line4bass] {
    -i { wsngtranspose {
        -n { -1.429 }
        -i { bar1treble }
    } }
    -i { wsngtranspose {
        -n { -0.857 }
        -i { bar1treble }
    } } }

wsngsequence [line5treble] {
    -i { PauseOneHalf }
    -i { wsngtranspose {
        -n { 1.143 }
        -i { Frag1Inv }
    } }
    -i { PauseOneHalf }
    -i { wsngtranspose {
        -n { 1.286 }
        -i { Frag1Inv }
    } } }

wsngsequence [bar9bass] {
    -i { wsngtranspose {
        -n { 0.571 }
        -i { Frag1Inv }
    } }
    -i { wsngtranspose {
        -n { -0.571 }
        -i { wsngretrograde {
            -i { Frag2 }
        } } } } }

wsngsequence [line5bass] {
    -i { bar9bass }
    -i { wsngtranspose {
        -n { 0.143 }
        -i { bar9bass }
    } } }

wsngsequence [line6treble] {
    -i { PauseOneEighth }
    -i { wsngtranspose {
        -n { 1.0 }
        -i { Frag3 }
    } }
    -i { wsngtranspose {
        -n { 0.714 }
        -i { Frag3 }
    } }
    -i { wsngtranspose {
        -n { 0.429 }
        -i { Frag3 }
    } }
    -i { PauseOneEighth }
    -i { PauseOneQuarter }
}

wsngsequence [line6bass] {
    -i { wsngtranspose {
        -n { 0.854 }
        -i { Frag1Inv }
    } }
    -i { wsngtranspose {

```

```

        -n { 0.571 }
        -i { Frag1Inv }
    } }
-i { wsngrtranspose {
    -n { 0.286 }
    -i { Frag1Inv }
} }
-i { Frag1Inv }
}

wsngsequence [line7treble] {
-i { wsngrtranspose {
    -n { 0.286 }
    -i { Frag1 }
} }
-i {PauseOneSixteenth}
-i { wsngrtranspose {
    -n {1.0}
    -i { wsngrinvert {
        -i { SubFrag1B }
    } } } }
-i {PauseOneSixteenth}
-i { wsngrtranspose {
    -n {0.854}
    -i { wsngrinvert {
        -i { SubFrag1B }
    } } } }
-i {PauseOneSixteenth}
-i { wsngrtranspose {
    -n {1.571}
    -i { wsngrinvert {
        -i { SubFrag1B }
    } } } }
-i {PauseOneSixteenth}
-i { wsngrtranspose {
    -n {1.143}
    -i { SubFrag1B }
} }
-i {PauseOneSixteenth}
-i { wsngrtranspose {
    -n {1.429}
    -i { wsngrinvert { -i { SubFrag1A } }
} } }
-i {PauseOneEighth}
-i {PauseOneSixteenth}
}

wsngsequence [line7bass] {
-i { wsngrtranspose {
    -n {-0.857}
    -i { Frag8 }
} }
-i { wsngrtranspose {
    -n {-0.143}
    -i { Frag1Inv }
} }
-i { wsngrtranspose {
    -n {-0.286}
    -i { wsngrretrograde { -i {SubFrag1B} } }
} }
-i { wsngrinvert { -i {SubFrag1B} } }
-i { wsngrtranspose {
    -n {0.143}
    -i { wsngrretrograde { -i {SubFrag1B} } }
} }
-i {PauseOneEighth}
-i {PauseOneSixteenth}
}

```

```

-i { wsntranspose {
    -n { 0.714 }
    -i { Frag4 }
} } }

wsngsequence [line8treble] {
-i { wsntranspose {
    -n { 1.714 }
    -i { Frag1Inv }
} } }
-i {PauseOneHalf}
-i { wsntranspose {
    -n { 1.286 }
    -i { Frag1 }
} } }
-i {PauseOneHalf}
}

wsngsequence [line8bass] {
-i { wsntranspose {
    -n { 0.143 }
    -i { Frag4 }
} } }
-i {PauseOneQuarter}
-i { wsntranspose {
    -n {0.286}
    -i { Frag1Inv }
} } }
-i {PauseOneHalf}
-i { wsntranspose {
    -n {-0.286}
    -i { Frag1 }
} } }
} } }

wsngsequence [line9treble] {
-i { wsntranspose {
    -n { 1.571 }
    -i { Frag1Inv }
} } }
-i {PauseOneHalf}
-i { wsntranspose {
    -n { 1.143 }
    -i { Frag1 }
} } }
-i {PauseOneHalf}
-i { wsntranspose {
    -n { 1.0 }
    -i { Frag1 }
} } }
-i { wsntranspose {
    -n { 1.286 }
    -i { Frag1 }
} } }
} } }

wsngsequence [line9bass] {
-i {PauseOneHalf}
-i { wsntranspose {
    -n {0.143}
    -i { Frag1Inv }
} } }
-i {PauseOneHalf}
-i { wsntranspose {
    -n {-0.429}
    -i { Frag1 }
} } }
-i {PauseOneEighth}
-i { wsntranspose {

```



```

        -n { -0.143 }
        -i { wsnginvert { -i {Frag3 } } }
    } }
-i { wsngtranspose {
    -n { 0.143 }
    -i { wsnginvert { -i {Frag3 } } }
} } }

wsngsequence [line10treble] {
-i { wsngtranspose {
    -n { 1.429 }
    -i { Frag1 }
} }
-i { PauseOneQuarter }
-i { wsngtranspose {
    -n { 0.143 }
    -i { wsngretrograde {
        -i { wsngchanmult {
            -t {2.0}
            -i { Frag6 }
        } } } }
} }
-i { wsngtranspose {
    -n { 0.857 }
    -i { Frag1Inv }
} }
-i { PauseOneHalf }
-i { PauseOneSixteenth }
-i { CChord1st }
}

wsngsequence [line10bass] {
-i {PauseOneHalf}
-i { wsngtranspose {
    -n { -0.857 }
    -i { Frag1 }
} }
-i {PauseOneEighth}
-i { wsngtranspose {
    -n { -1.0 }
    -i { Frag3 }
} }
-i {PauseOneEighth}
-i { Frag4 }
-i { COct }
}

wsngsequence [treble] {
-i { line1treble }
-i { line2treble }
-i { line3treble }
-i { line4treble }
-i { line5treble }
-i { line6treble }
-i { line7treble }
-i { line8treble }
-i { line9treble }
-i { line10treble }
}

wsngsequence [bass] {
-i { line1bass }
-i { line2bass }
-i { line3bass }
-i { line4bass }
-i { line5bass }
-i { line6bass }
}

```


B.8 River of Light

```

% create several genpan fragments, each from a slightly different note palette
%

GenPan2Scale = <P7 P8 P9 P11 P12>
GenPan1Scale = <P6 P7 P8 P9 P12 P13>
GenPan4Scale = <P8 P9 P11 P12 P13>
GenPan3Scale = <P7 P9 P11 P12 P13>

% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
wsngfromscale [GenPan1] {
  -i { wsngrandom [ii] { -c -S {113} -g {0.03} -s {0} -e {8} -p {0} -P {4} -x {0} }
  }
  -s { GenPan1Scale }
}
wsngfromscale [GenPan2] {
  -i { wsngrandom [iii] { -c -S {123} -g {0.03} -s {0} -e {8} -p {0} -P {5} -x {0}
  } }
  -s { GenPan2Scale }
}
wsngfromscale [GenPan3] {
  -i { wsngrandom [iiii] { -c -S {131} -g {0.03} -s {0} -e {8} -p {0} -P {4} -x {0}
  } }
  -s { GenPan3Scale }
}
wsngfromscale [GenPan4] {
  -i { wsngrandom [iiiii] { -c -S {153} -g {0.03} -s {0} -e {8} -p {0} -P {4} -x
  {0} } }
  -s { GenPan4Scale }
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% merge the genpan fragments
%

wsngrepeat [genpancommon] {
  -c { 3 } -i { GenPan1 }
}
wsngsequence [genpanraw] {
  -i { genpancommon } -i { GenPan2 }
}
wsngsequence [genpanraw2] {
  -i { genpancommon } -i { GenPan3 }
}
wsngsequence [genpanraw3] {
  -i { genpancommon } -i { GenPan4 }
}
wsngrepeat [genpanbits] {
  -c { 2 } -i { genpanraw }
}

wsngtranspose [genpan] {
  -n { 12 }
  -i { wsngsequence [genmain] {
    -i { genpanbits }
    -i { genpanraw2 } -i { genpanraw3 }
    -i { genpanbits } -i { genpanraw2 }
    -i { genpanraw }
  } } }

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create several genbar fragments, each from a slightly different note palette
%

```

```

GenBar2Scale = <P6 P7 P8 P9 P11 P12>
GenBar1Scale = <P5 P6 P7 P11 P12>
GenBar4Scale = <P3 P5 P6 P7 P8 P9 P11 P12 P13>
GenBar5Scale = <P6 P7 P8 P9 P11 P12 P13 P14 P15>
GenBar3Scale = <P3 P5 P6 P7 P8 P9 P11 P12 P13>
% GenBar6Scale = GenBar3Scale

% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
wsngfromscale [GenBar1] {
  -i { wsngrandom [jj] { -c -S {17} -s {0} -e {16} -p {0} -P {5} -d {2} -x {1} } }
  -s { GenBar1Scale }
}
wsngfromscale [GenBar2] {
  -i { wsngrandom [jjj] { -c -S {117} -s {0} -e {16} -p {0} -P {4} -d {2} -x {1} } }
  -s { GenBar2Scale }
}
wsngfromscale [GenBar3] {
  -i { wsngrandom [jjjj] { -c -S {147} -s {0} -e {48} -p {0} -P {8} -d {2} -x {1} } }
  -s { GenBar3Scale }
}
wsngfromscale [GenBar4] {
  -i { wsngrandom [jjjjj] { -c -S {167} -s {0} -e {32} -p {0} -P {8} -d {2} -x {1} } }
  -s { GenBar4Scale }
}
wsngfromscale [GenBar5] {
  -i { wsngrandom [jjjjjj] { -c -S {171} -s {0} -e {32} -p {0} -P {8} -d {2} -x {1} } }
  -s { GenBar5Scale }
}
wsngfromscale [GenBar6] {
  -i { wsngrandom [jjjjjjj] { -c -S {173} -s {0} -e {64} -p {0} -P {8} -d {2} -x {1} } }
  -s { GenBar3Scale }
}
wsngrepeat [genbarcommon] {
  -c { 2 } -i { GenBar1 }
}

wsngtranspose [genbar] {
  -n {0}
  -i { wsnsequence [genbarset] {
    -i { genbarcommon }
    -i { GenBar2 } -i { GenBar3 }
    -i { GenBar4 } -i { genbarcommon }
    -i { GenBar5 } -i { GenBar6 }
  } } }

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create several Gambang fragments, each from a slightly different note palette
%
Gambang2Scale = <P6 P7 P8 P9 P11 P12 P13 P14>
Gambang1Scale = <P9 P11 P12 P13 P14 P15>
Gambang4Scale = <P6 P7 P8 P9 P11 P12 P13 P14 P15>
Gambang5Scale = <P8 P9 P11 P12 P13 P14 P15>
Gambang3Scale = <P3 P5 P6 P7 P8 P9 P11 P12 P13 P14 P15>

% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
% I WASN'T GETTING ENOUGH VARIETY FROM THE MELODY ON THIS CHANNEL

```

```

% SO I DOUBLED ITS -P VALUE, SO THAT IT WILL DRAW FROM A WIDER RANGE OF NOTES
wsngfromscale [Gambang1] {
  -i { wsngrandom [kk] { -c -S {15} -g {0.07} -s {0} -e {48} -p {0} -P {14} -d {1}
    -x {2} } }
  -s { Gambang1Scale }
}
wsngfromscale [Gambang2] {
  -i { wsngrandom [kkk] { -c -S {151} -g {0.07} -s {0} -e {48} -p {0} -P {10} -d
    {1} -x {2} } }
  -s { Gambang2Scale }
}
wsngfromscale [Gambang3] {
  -i { wsngrandom [kkkk] { -c -S {159} -g {0.07} -s {0} -e {48} -p {0} -P {16} -d
    {1} -x {2} } }
  -s { Gambang3Scale }
}
wsngfromscale [Gambang4] {
  -i { wsngrandom [kkkkk] { -c -S {143} -g {0.07} -s {0} -e {48} -p {0} -P {12} -d
    {1} -x {2} } }
  -s { Gambang4Scale }
}
wsngfromscale [Gambang5] {
  -i { wsngrandom [kkkkkk] { -c -S {159} -g {0.07} -s {0} -e {64} -p {0} -P {20} -d
    {1} -x {2} } }
  -s { Gambang5Scale }
}

wsngsequence [gambangs] {
-i { Gambang1 }
-i { Gambang2 }
-i { Gambang3 }
-i { Gambang4 }
-i { Gambang5 }
}

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create several Bonbar fragments, each from a slightly different note palette
%

```

```

BonBar2Scale = <P6 P11 P12 P13>
BonBar1Scale = <P9 P11 P12 P13>
BonBar4Scale = <P6 P8 P9 P11 P12 P13>
BonBar5Scale = <P6 P11 P12 P13>
BonBar3Scale = <P5 P6 P7 P8 P9 P13>

```

```

% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
wsngfromscale [BonBar1] {
  -i { wsngrandom [mm] { -c -S {1101} -s {0} -e {48} -p {0} -P {3} -d {2} -r {0.25}
    -x {3} } }
  -s { BonBar1Scale }
}
wsngfromscale [BonBar2] {
  -i { wsngrandom [mmm] { -c -S {1102} -s {0} -e {48} -p {0} -P {3} -d {2} -r
    {0.25} -x {3} } }
  -s { BonBar2Scale }
}
wsngfromscale [BonBar3] {
  -i { wsngrandom [mmmm] { -c -S {1103} -s {0} -e {48} -p {0} -P {5} -d {2} -r
    {0.25} -x {3} } }
  -s { BonBar3Scale }
}
wsngfromscale [BonBar4] {
  -i { wsngrandom [mmmmm] { -c -S {1104} -s {0} -e {48} -p {0} -P {3} -d {2} -r
    {0.25} -x {3} } }

```

```

-s { BonBar4Scale }
}
wsngfromscale [BonBar5] {
-i { wsngrandom [mmmmmm] { -c -S {1105} -s {0} -e {64} -p {0} -P {5} -d {2} -r
{0.25} -x {3} } } }
-s { BonBar5Scale }
}

wsngsequence [bonbars] {
-i { BonBar1 }
-i { BonBar2 }
-i { BonBar3 } % okay range
-i { BonBar4 }
-i { BonBar5 } % okay range
}

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create several BonPan fragments, each from a slightly different note palette
%
% CURRENTLY IGNORING THE BONPANS - JAS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

BonPan2Scale = <P6 P11 P12 P13>
BonPan1Scale = <P6 P9 P11 P12>
BonPan5Scale = <P9 P11 P12 P13>
BonPan4Scale = <P6 P9 P11 P12>

BonPanFinish = <P6D1X7V-0 P6 P6 P7>

```

```

wsngfromscale [BonPan1] {
-i { wsngsequence [cxkx] {
-i { wsngrepeat [aoiwq] {
-c { 4 }
-i { wsngrandom [vv] { -c -S {121401} -s {0} -e {4} -p {0} -P {3}
-d {1} -r {0.75} -x {7} }
} } }
-i { wsngrepeat [aowq] {
-c { 4 }
-i { wsngrandom [vvv] { -c -S {121402} -s {0} -e {4} -p {0} -P {3}
-d {1} -r {0.25} -x {7} }
} } }
-i { wsngrepeat [oiwq] {
-c { 4 }
-i { wsngrandom [vvvv] { -c -S {12143} -s {0} -e {4} -p {0} -P {3}
-d {1} -r {0.25} -x {7} }
} } } } }
-s { BonBar1Scale }
}

```

```

wsngfromscale [BonPan2] {
-i { wsngsequence [cxky] {
-i { wsngrepeat [aoiq] {
-c { 4 }
-i { wsngrandom [uu] { -c -S {11731} -s {0} -e {4} -p {0} -P {3} -
d {1} -r {0.25} -x {7} }
} } }
-i { wsngrepeat [aow] {
-c { 4 }
-i { wsngrandom [uuu] { -c -S {11732} -s {0} -e {4} -p {0} -P {3}
-d {1} -r {0.25} -x {7} }
} } }
-i { wsngrepeat [oiq] {
-c { 4 }
-i { wsngrandom [uuuu] { -c -S {11733} -s {0} -e {4} -p {0} -P {3}
-d {1} -r {0.25} -x {7} }
} } } } }
}

```

```

    -s { BonBar2Scale }
}

wsngsequence [BonPan3] {
    -i { wsngretrograde [blap] {
        -i { BonPan1 }
    } } }

wsngfromscale [BonPan4] {
    -i { wsngsequence [asuix] {
        -i { wsngrepeat [lksaoi] {
            -c { 4 }
            -i { wsngrandom [hh] { -c -S {131401} -s {0} -e {4} -p {0} -P {3}
                -d {1} -r {0.75} -x {7} }
            } } }
        -i { wsngrepeat [nhg] {
            -c { 4 }
            -i { wsngrandom [hhh] { -c -S {131402} -s {0} -e {4} -p {0} -P {3}
                -d {1} -r {0.25} -x {7} }
            } } }
        -i { wsngrepeat [uyu] {
            -c { 4 }
            -i { wsngrandom [hhhh] { -c -S {13143} -s {0} -e {4} -p {0} -P {3}
                -d {1} -r {0.25} -x {7} }
            } } } } }
    -s { BonBar4Scale }
}

wsngfromscale [BonPan5] {
    -i { wsngsequence [kx] {
        -i { wsngrepeat [csfse] {
            -c { 4 }
            -i { wsngrandom [dd] { -c -S {11401} -s {0} -e {4} -p {0} -P {3} -
                d {1} -r {0.75} -x {7} }
            } } }
        -i { wsngrepeat [zdsa] {
            -c { 4 }
            -i { wsngrandom [ddd] { -c -S {11402} -s {0} -e {4} -p {0} -P {3}
                -d {1} -r {0.25} -x {7} }
            } } }
        -i { wsngrepeat [gdf] {
            -c { 4 }
            -i { wsngrandom [dddd] { -c -S {1143} -s {0} -e {4} -p {0} -P {3}
                -d {1} -r {0.25} -x {7} }
            } } }
        -i { wsngrepeat [erwg] {
            -c { 4 }
            -i { wsngrandom [dddd] { -c -S {1143} -s {0} -e {4} -p {0} -P {3}
                -d {1} -r {0.25} -x {7} }
            } } } } }
    -s { BonBar5Scale }
}

wsngtranspose [bonpans] {
    -n { 12 }
    -i { wsngsequence [bonpanset] {
        -i { BonPan1 }
        -i { BonPan2 }
        -i { BonPan3 }
        -i { BonPan4 }
        -i { BonPan5 }
    } } }
}

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create several SarBar fragments, each from a slightly different note palette
%

```

```

SarBar2Scale = <P12 P13>
SarBar1Scale = <P9 P11 P12 P13>
SarBar4Scale = <P8 P9 P11 P12 P13>
SarBar5Scale = <P11 P12 P13>
SarBar3Scale = <P7 P8 P9 P12 P13>

% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
wsngfromscale [SarBar1] {
  -i { wsngrandom [pp] { -c -S {21201} -s {0} -e {48} -p {0} -P {1} -d {8} -r {0.5}
    -x {5} } } }
  -s { SarBar1Scale }
}
wsngfromscale [SarBar2] {
  -i { wsngrandom [ppp] { -c -S {21202} -s {0} -e {48} -p {0} -P {3} -d {8} -x {5}
    } } }
  -s { SarBar2Scale }
}
wsngfromscale [SarBar3] {
  -i { wsngrandom [pppp] { -c -S {21203} -s {0} -e {48} -p {0} -P {4} -d {8} -x {5}
    } } }
  -s { SarBar3Scale }
}
wsngfromscale [SarBar4] {
  -i { wsngrandom [ppppp] { -c -S {21204} -s {0} -e {48} -p {0} -P {2} -d {8} -r
    {0.33} -x {5} } } }
  -s { SarBar4Scale }
}
wsngfromscale [SarBar5] {
  -i { wsngrandom [pppppp] { -c -S {21205} -s {0} -e {64} -p {0} -P {4} -d {8} -r
    {0.125} -x {5} } } }
  -s { SarBar5Scale }
}

wsngsequence [sarons] {
  -i { SarBar1 }
  -i { SarBar2 }
  -i { SarBar3 }
  -i { SarBar4 }
  -i { SarBar5 }
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create several Peking fragments, each from a slightly different note palette
%

Peking2Scale = <P11 P12>
Peking1Scale = <P11 P12 P13>
Peking4Scale = <P9 P11 P12 P13>
Peking3Scale = <P8 P9 P11 P12 P13>
Peking6Scale = <P11 P12 P13>
Peking5Scale = <P7 P8 P9 P12 P13>

% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
wsngfromscale [Peking1] {
  -i { wsngrandom [qq] { -c -S {11201} -s {0} -e {8} -p {0} -P {1} -d {2} -x {6} }
    } }
  -s { Peking1Scale }
}
wsngfromscale [Peking2] {
  -i { wsngrandom [qqq] { -c -S {11202} -s {0} -e {32} -p {0} -P {2} -d {2} -x {6}
    } } }
  -s { Peking2Scale }
}
wsngfromscale [Peking3] {

```



```

-i { wsngrandom [qqqq] { -c -S {11203} -s {0} -e {32} -p {0} -P {3} -d {2} -x {6}
} }
-s { Peking3Scale }
}
wsngfromscale [Peking4] {
-i { wsngrandom [qqqqq] { -c -S {11204} -s {0} -e {48} -p {0} -P {4} -d {2} -x
{6} } }
-s { Peking4Scale }
}
wsngfromscale [Peking5] {
-i { wsngrandom [qqqqqq] { -c -S {11205} -s {0} -e {48} -p {0} -P {2} -d {2} -x
{6} } }
-s { Peking5Scale }
}
wsngfromscale [Peking6] {
-i { wsngrandom [qqqqqqq] { -c -S {11206} -s {0} -e {64} -p {0} -P {4} -d {2} -x
{6} } }
-s { Peking6Scale }
}

wsngrepeat [pekingfrag1] {
-c { 4 } -i { Peking1 }
}
wsngsequence [pekingfrag2] {
-i { Peking2 }
-i { Peking3 }
-i { Peking4 }
-i { Peking5 }
-i { Peking6 }
}

wsngtranspose [pekings] {
-n { 0 }
-i { wsngsequence [pekingfrags] {
-i { pekingfrag1 }
-i { pekingfrag2 }
} } }

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create several Kendang fragments, each from a slightly different note palette
%
KendangScale = <P1 P5 P6>

% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
wsngfromscale [Kendang1] {
-i { wsngrandom [ww] { -c -S {1701} -s {0} -e {48} -p {0} -P {2} -d {2} -x {8} }
}
-s { KendangScale }
}
wsngfromscale [Kendang2] {
-i { wsngrandom [www] { -c -S {1702} -s {0} -e {32} -p {0} -P {2} -d {2} -x {8} }
}
-s { KendangScale }
}
wsngfromscale [Kendang3] {
-i { wsngrandom [wwww] { -c -S {1703} -s {0} -e {32} -p {0} -P {2} -d {2} -x {8}
} }
-s { KendangScale }
}
wsngfromscale [Kendang4] {
-i { wsngrandom [wwwww] { -c -S {1704} -s {0} -e {48} -p {0} -P {2} -d {2} -x {8}
} }
-s { KendangScale }
}

```

```

wsngfromscale [Kendang5] {
  -i { wsngrandom [wwwww] { -c -S {1705} -s {0} -e {64} -p {0} -P {2} -d {2} -x
    {8} } } }
  -s { KendangScale }
}

```

```

wsngchanmult [drums] {
  -V {0.31}
  -i { wsngsequence [drumparts] {
    -i { Kendang1 }
    -i { Kendang2 }
    -i { Kendang3 }
    -i { Kendang4 }
    -i { Kendang5 }
  } } }

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create Gong fragment
%
```

```

gongpart = <P1D12X4V240 P1D12 P1D12 P1D188V0>

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% now pull the different voices together
%
```

```

wsngsequence [genders] {
  -s
  -i { genpan }
  -i { genbar }
  -i { gambangs }
}

```

```

% combine bonangs to play together

```

```

wsngsequence [bonangs] {
  -s
  -i { bonpans }
  -i { bonbars }
  -i { sarons }
}

```

```

wsngsequence [colotomics] {
  -s
  -i { wsngsequence [colotomics2]{
    -s
    -i { gongpart }
    -i { gongpart }
    -i { gongpart }
  } }
  -i { drums }
}

```

```

wsngsequence [melodycycle] {
  -s
  -i { genders }
  -i { bonangs }
  -i { pekings }
}

```

```

% gradually introduce each instrument

```

```

wsngsequence [melodyintro] {
  -s
  -i { wsngselect {
    -s {83} -p -x {0} % 40 s

```

```

    -i {genders}
  }}
-i { wsngselect {
  -s {71} -p -x {1} % 50 s
  -i {genders}
  }}
-i { wsngselect {
  -s {59} -p -x {2} % 60 s
  -i {genders}
  }}
-i { wsngselect {
  -s {37} -p -x {6} % 70 s
  -i {pekings}
  }}
-i { wsngselect {
  -s {23} -p -x {7} -x {3} -x {5}
  -i {bonangs}
  }}
}

% gradually silence each instrument until only the beat and gongs are left
wsngsequence [melodyoutro] {
-s
-i { wsngselect {
  -e {101} -p -x {0} % 40 s
  -i {genders}
  }}
-i { wsngselect {
  -e {121} -p -x {1} % 50 s
  -i {genders}
  }}
-i { wsngselect {
  -e {153} -p -x {2} % 60 s
  -i {genders}
  }}
-i { wsngselect {
  -e {171} -p -x {6} % 70 s
  -i {pekings}
  }}
-i { wsngselect {
  -e {200} -p -x {7} -x {3} -x {5} % 80 s
  -i {bonangs}
  }} }

wsngsequence [verses] {
  -i { melodyintro }
  -i { wsngretrograde [secondbest] { -i {melodycycle} } }
  -i { melodycycle }
  -i { melodyoutro }
}

wsngrepeat [rhythm] {
  -c {5}
  -i { colotomics }
}

wsngsequence [majorsections] {
  -i { wsngfromscale [render] {
    -s { wsngscale [slendroSorrel] {
      -m { 7 } % 7 and 8 are nice, 3 is a bit dark and mysterious
    } }
    -i { wsngsequence {
      -s
      -i {verses}
      -i {rhythm}
    } } } } }
}

```

```

% this is a dummy so that we can selectively play different structures
wsngtranspose [play] {
  -n { -6 } % need to use 12 or 24 for western keys
  -i { wsngchanmult [tempoadjust] {
    -t { 1.1 }
    %i { wsngrandomize [amateurs] {-T {0.02} -V {20} -i { majorsections }} }
    -i { majorsections }
  } } }

```

B.9 Clockhouse

```
% create several genpan fragments, each from a slightly different note palette
%
```

```
GenPan1Scale = <P7 P8 P9 P11 P12>
GenPan2Scale = <P6 P7 P8 P9 P12 P13>
GenPan3Scale = <P8 P9 P11 P12 P13>
GenPan4Scale = <P7 P9 P11 P12 P13>
```

```
% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
```

```
wsgfromscale [GenPan1] {
  -i { wsngrandom [ii] { -c -S {13} -s {0} -e {8} -p {0} -P {4} -x {0} } }
  -s { GenPan1Scale }
}
wsgfromscale [GenPan2] {
  -i { wsngrandom [iii] { -c -S {23} -s {0} -e {8} -p {0} -P {5} -x {0} } }
  -s { GenPan2Scale }
}
wsgfromscale [GenPan3] {
  -i { wsngrandom [iiii] { -c -S {31} -s {0} -e {8} -p {0} -P {4} -x {0} } }
  -s { GenPan3Scale }
}
wsgfromscale [GenPan4] {
  -i { wsngrandom [iiiii] { -c -S {53} -s {0} -e {8} -p {0} -P {4} -x {0} } }
  -s { GenPan4Scale }
}
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% merge the genpan fragments
%
```

```
wsgrepeat [genpancommon] {
  -c { 3 } -i { GenPan1 }
}
wsgsequence [genpanraw] {
  -i { genpancommon } -i { GenPan2 }
}
wsgsequence [genpanraw2] {
  -i { genpancommon } -i { GenPan3 }
}
wsgsequence [genpanraw3] {
  -i { genpancommon } -i { GenPan4 }
}
wsgrepeat [genpanbits] {
  -c { 2 } -i { genpanraw }
}

wsgtranspose [genpan] {
  -n { 12 }
  -i {
    wsgsequence [genmain] {
      -i { genpanbits }
      -i { genpanraw2 } -i { genpanraw3 }
      -i { genpanbits } -i { genpanraw2 }
      -i { genpanraw }
    }
  }
}
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% create several genbar fragments, each from a slightly different note palette
%
```

```
GenBar1Scale = <P6 P7 P8 P9 P11 P12>
```

```

GenBar2Scale = <P5 P6 P7 P11 P12>
GenBar3Scale = <P3 P5 P6 P7 P8 P9 P11 P12 P13>
GenBar4Scale = <P6 P7 P8 P9 P11 P12 P13 P14 P15>
GenBar5Scale = <P3 P5 P6 P7 P8 P9 P11 P12 P13>
% GenBar6Scale = GenBar3Scale

% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
wsngfromscale [GenBar1] {
  -i { wsngrandom [jj] } { -c -S {7} -s {0} -e {16} -p {0} -P {5} -d {2} -x {1} } }
  -s { GenBar1Scale }
}
wsngfromscale [GenBar2] {
  -i { wsngrandom [jjj] } { -c -S {17} -s {0} -e {16} -p {0} -P {4} -d {2} -x {1} } }
  -s { GenBar2Scale }
}
wsngfromscale [GenBar3] {
  -i { wsngrandom [jjjj] } { -c -S {47} -s {0} -e {48} -p {0} -P {8} -d {2} -x {1} } }
  -s { GenBar3Scale }
}
wsngfromscale [GenBar4] {
  -i { wsngrandom [jjjjj] } { -c -S {67} -s {0} -e {32} -p {0} -P {8} -d {2} -x {1} } }
  -s { GenBar4Scale }
}
wsngfromscale [GenBar5] {
  -i { wsngrandom [jjjjjj] } { -c -S {71} -s {0} -e {32} -p {0} -P {8} -d {2} -x {1} } }
  -s { GenBar5Scale }
}
wsngfromscale [GenBar6] {
  -i { wsngrandom [jjjjjjj] } { -c -S {73} -s {0} -e {64} -p {0} -P {8} -d {2} -x
    {1} } }
  -s { GenBar3Scale }
}
wsngrepeat [genbarcommon] {
  -c { 2 } -i { GenBar1 }
}

wsngtranspose [genbar] {
  -n {0}
  -i {
    wsngsequence [genbarset] {
      -i { genbarcommon }
      -i { GenBar2 } -i { GenBar3 }
      -i { GenBar4 } -i { genbarcommon }
      -i { GenBar5 } -i { GenBar6 }
    }
  }
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create several Gambang fragments, each from a slightly different note palette
%

Gambang1Scale = <P6 P7 P8 P9 P11 P12 P13 P14>
Gambang2Scale = <P9 P11 P12 P13 P14 P15>
Gambang3Scale = <P6 P7 P8 P9 P11 P12 P13 P14 P15>
Gambang4Scale = <P8 P9 P11 P12 P13 P14 P15>
Gambang5Scale = <P3 P5 P6 P7 P8 P9 P11 P12 P13 P14 P15>

```

```

% the -P value is based on the number of elements in its input scale

```

```

% the -e value is based on the phrase lenght of the original Wilujeng encoding
% I WASN'T GETTING ENOUGH VARIETY FROM THE MELODY ON THIS CHANNEL
% SO I DOUBLED ITS -P VALUE, SO THAT IT WILL DRAW FROM A WIDER RANGE OF NOTES
wsngfromscale [Gambang1] {
  -i { wsngrandom [kk] { -c -S {5} -s {0} -e {48} -p {0} -P {14} -d {1} -x {2} } }
  -s { Gambang1Scale }
}
wsngfromscale [Gambang2] {
  -i { wsngrandom [kkk] { -c -S {51} -s {0} -e {48} -p {0} -P {10} -d {1} -x {2} } }
  -s { Gambang2Scale }
}
wsngfromscale [Gambang3] {
  -i { wsngrandom [kkkk] { -c -S {59} -s {0} -e {48} -p {0} -P {16} -d {1} -x {2} } }
  -s { Gambang3Scale }
}
wsngfromscale [Gambang4] {
  -i { wsngrandom [kkkkk] { -c -S {43} -s {0} -e {48} -p {0} -P {12} -d {1} -x {2} } }
  -s { Gambang4Scale }
}
wsngfromscale [Gambang5] {
  -i { wsngrandom [kkkkkk] { -c -S {59} -s {0} -e {64} -p {0} -P {20} -d {1} -x {2} } }
  -s { Gambang5Scale }
}

wsngsequence [gambangs] {
-i { Gambang1 }
-i { Gambang2 }
-i { Gambang3 }
-i { Gambang4 }
-i { Gambang5 }
}

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create several Bonbar fragments, each from a slightly different note palette
%

```

```

BonBar1Scale = <P6 P11 P12 P13>
BonBar2Scale = <P9 P11 P12 P13>
BonBar3Scale = <P6 P8 P9 P11 P12 P13>
BonBar4Scale = <P6 P11 P12 P13>
BonBar5Scale = <P5 P6 P7 P8 P9 P13>

```

```

% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
wsngfromscale [BonBar1] {
  -i { wsngrandom [mm] { -c -S {101} -s {0} -e {48} -p {0} -P {3} -d {2} -r {0.25}
    -g {0.05} -x {3} } }
  -s { BonBar1Scale }
}
wsngfromscale [BonBar2] {
  -i { wsngrandom [mmm] { -c -S {102} -s {0} -e {48} -p {0} -P {3} -d {2} -r {0.25}
    -g {0.05} -x {3} } }
  -s { BonBar2Scale }
}
wsngfromscale [BonBar3] {
  -i { wsngrandom [mmmm] { -c -S {103} -s {0} -e {48} -p {0} -P {5} -d {2} -r
    {0.25} -g {0.05} -x {3} } }
  -s { BonBar3Scale }
}
wsngfromscale [BonBar4] {

```

```

-i { wsngrandom [mmmmmm] { -c -S {104} -s {0} -e {48} -p {0} -P {3} -d {2} -r
  {0.25} -g {0.05} -x {3} } }
-s { BonBar4Scale }
}
wsngfromscale [BonBar5] {
-i { wsngrandom [mmmmmm] { -c -S {105} -s {0} -e {64} -p {0} -P {5} -d {2} -r
  {0.25} -g {0.05} -x {3} } }
-s { BonBar5Scale }
}

wsngsequence [bonbars] {
-i { BonBar1 }
-i { BonBar2 }
-i { BonBar3 } % okay range
-i { BonBar4 }
-i { BonBar5 } % okay range
}

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create several BonPan fragments, each from a slightly different note palette
%
% CURRENTLY IGNORING THE BONPANS - JAS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

BonPan1Scale = <P6 P11 P12 P13>
BonPan2Scale = <P6 P9 P11 P12>
BonPan4Scale = <P9 P11 P12 P13>
BonPan5Scale = <P6 P9 P11 P12>

```

```

BonPanFinish = <P6D1X7V-0 P6 P6 P7>

```

```

wsngfromscale [BonPan1] {
-i { wsnsequence [cxkx] {
  -i { wsnrepeat [aoiwq] {
    -c { 4 }
    -i { wsngrandom [vv] { -c -S {1401} -s {0} -e {4} -p {0} -P {3} -d
      {1} -r {0.75} -x {7} } }
  } }
-i { wsnrepeat [aowq] {
  -c { 4 }
  -i { wsngrandom [vvv] { -c -S {1402} -s {0} -e {4} -p {0} -P {3} -
    d {1} -r {0.25} -x {7} } }
  } }
-i { wsnrepeat [oiwq] {
  -c { 4 }
  -i { wsngrandom [vvvv] { -c -S {143} -s {0} -e {4} -p {0} -P {3} -
    d {1} -r {0.25} -x {7} } }
  } } } }
-s { BonBar1Scale }
}

```

```

wsngfromscale [BonPan2] {
-i {
  wsnsequence [cxky] {
    -i { wsnrepeat [aoiq] {
      -c { 4 }
      -i { wsngrandom [uu] { -c -S {1731} -s {0} -e {4} -p {0} -P {3} -d
        {1} -r {0.25} -x {7} } }
    } }
    -i { wsnrepeat [aow] {
      -c { 4 }
      -i { wsngrandom [uuu] { -c -S {1732} -s {0} -e {4} -p {0} -P {3} -
        d {1} -r {0.25} -x {7} } }
    } }
    -i { wsnrepeat [oiq] {
      -c { 4 }
    } }
  } }
}

```



```

        -i { wsngrandom [uuuu] { -c -S {1733} -s {0} -e {4} -p {0} -P {3}
            -d {1} -r {0.25} -x {7} } }
    } } }
-s { BonBar2Scale }
}

wsngsequence [BonPan3] {
    -i { wsngretrograde [blap] {
        -i { BonPan1 }
    } } }

wsngfromscale [BonPan4] {
    -i { wsngsequence [asuix] {
        -i { wsngrepeat [lksaoi] {
            -c { 4 }
            -i { wsngrandom [hh] { -c -S {1401} -s {0} -e {4} -p {0} -P {3} -d
                {1} -r {0.75} -x {7} } }
        } } }
        -i { wsngrepeat [nhg] {
            -c { 4 }
            -i { wsngrandom [hhh] { -c -S {1402} -s {0} -e {4} -p {0} -P {3} -
                d {1} -r {0.25} -x {7} } }
        } } }
        -i { wsngrepeat [uyu] {
            -c { 4 }
            -i { wsngrandom [hhhh] { -c -S {143} -s {0} -e {4} -p {0} -P {3} -
                d {1} -r {0.25} -x {7} } } }
    } } }
-s { BonBar4Scale }
}

wsngfromscale [BonPan5] {
    -i { wsngsequence [kx] {
        -i { wsngrepeat [csfse] {
            -c { 4 }
            -i { wsngrandom [dd] { -c -S {1401} -s {0} -e {4} -p {0} -P {3} -d
                {1} -r {0.75} -x {7} } }
        } }
        -i { wsngrepeat [zdsa] {
            -c { 4 }
            -i { wsngrandom [ddd] { -c -S {1402} -s {0} -e {4} -p {0} -P {3} -
                d {1} -r {0.25} -x {7} } }
        } }
        -i { wsngrepeat [gdf] {
            -c { 4 }
            -i { wsngrandom [dddd] { -c -S {143} -s {0} -e {4} -p {0} -P {3} -
                d {1} -r {0.25} -x {7} } }
        } }
        -i { wsngrepeat [erwg] {
            -c { 4 }
            -i { wsngrandom [dddd] { -c -S {143} -s {0} -e {4} -p {0} -P {3} -
                -d {1} -r {0.25} -x {7} } }
        } } } }
-s { BonBar5Scale }
}

wsngtranspose [bonpans] {
    -n { 12 }
    -i {
        wsngsequence [bonpanset] {
            -i { BonPan1 }
            -i { BonPan2 }
            -i { BonPan3 }
            -i { BonPan4 }
            -i { BonPan5 }
        }
    }
}

```

```

}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create several SarBar fragments, each from a slightly different note palette
%

SarBar1Scale = <P12 P13>
SarBar2Scale = <P9 P11 P12 P13>
SarBar3Scale = <P8 P9 P11 P12 P13>
SarBar4Scale = <P11 P12 P13>
SarBar5Scale = <P7 P8 P9 P12 P13>

% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
wsngfromscale [SarBar1] {
  -i { wsngrandom [pp] { -c -S {201} -s {0} -e {48} -p {0} -P {1} -d {8} -r {0.5}
    -x {5} } } }
  -s { SarBar1Scale }
}
wsngfromscale [SarBar2] {
  -i { wsngrandom [ppp] { -c -S {202} -s {0} -e {48} -p {0} -P {3} -d {8} -x {5} } }
  -s { SarBar2Scale }
}
wsngfromscale [SarBar3] {
  -i { wsngrandom [pppp] { -c -S {203} -s {0} -e {48} -p {0} -P {4} -d {8} -x {5} } }
  -s { SarBar3Scale }
}
wsngfromscale [SarBar4] {
  -i { wsngrandom [ppppp] { -c -S {204} -s {0} -e {48} -p {0} -P {2} -d {8} -r
    {0.33} -x {5} } } }
  -s { SarBar4Scale }
}
wsngfromscale [SarBar5] {
  -i { wsngrandom [pppppp] { -c -S {205} -s {0} -e {64} -p {0} -P {4} -d {8} -r
    {0.125} -x {5} } } }
  -s { SarBar5Scale }
}

wsngsequence [sarons] {
  -i { SarBar1 }
  -i { SarBar2 }
  -i { SarBar3 }
  -i { SarBar4 }
  -i { SarBar5 }
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create several Peking fragments, each from a slightly different note palette
%

Peking1Scale = <P11 P12>
Peking2Scale = <P11 P12 P13>
Peking3Scale = <P9 P11 P12 P13>
Peking4Scale = <P8 P9 P11 P12 P13>
Peking5Scale = <P11 P12 P13>
Peking6Scale = <P7 P8 P9 P12 P13>

% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
wsngfromscale [Peking1] {
  -i { wsngrandom [qq] { -c -S {1201} -s {0} -e {8} -p {0} -P {1} -d {2} -x {6} } }
  -s { Peking1Scale }
}
wsngfromscale [Peking2] {

```

```

    -i { wsngrandom [qqq] { -c -S {1202} -s {0} -e {32} -p {0} -P {2} -d {2} -x {6} }
    }
    -s { Peking2Scale }
}
wsngfromscale [Peking3] {
    -i { wsngrandom [qqqq] { -c -S {1203} -s {0} -e {32} -p {0} -P {3} -d {2} -x {6}
    } }
    -s { Peking3Scale }
}
wsngfromscale [Peking4] {
    -i { wsngrandom [qqqqq] { -c -S {1204} -s {0} -e {48} -p {0} -P {4} -d {2} -x {6}
    } }
    -s { Peking4Scale }
}
wsngfromscale [Peking5] {
    -i { wsngrandom [qqqqqq] { -c -S {1205} -s {0} -e {48} -p {0} -P {2} -d {2} -x
    {6} } }
    -s { Peking5Scale }
}
wsngfromscale [Peking6] {
    -i { wsngrandom [qqqqqqq] { -c -S {1206} -s {0} -e {64} -p {0} -P {4} -d {2} -x
    {6} } }
    -s { Peking6Scale }
}

wsngrepeat [pekingfrag1] {
    -c { 4 } -i { Peking1 }
}
wsngsequence [pekingfrag2] {
    -i { Peking2 }
    -i { Peking3 }
    -i { Peking4 }
    -i { Peking5 }
    -i { Peking6 }
}

wsngtranspose [pekings] {
    -n { 0 }
    -i {
        wsngsequence [pekingfrags] {
            -i { pekingfrag1 }
            -i { pekingfrag2 }
        }
    }
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create several Kendang fragments, each from a slightly different note palette
%
KendangScale = <P1 P5 P6>

% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
wsngfromscale [Kendang1] {
    -i { wsngrandom [ww] { -c -S {701} -s {0} -e {48} -p {0} -P {2} -d {2} -x {8} } }
    -s { KendangScale }
}
wsngfromscale [Kendang2] {
    -i { wsngrandom [www] { -c -S {702} -s {0} -e {32} -p {0} -P {2} -d {2} -x {8} }
    }
    -s { KendangScale }
}
wsngfromscale [Kendang3] {
    -i { wsngrandom [wwww] { -c -S {703} -s {0} -e {32} -p {0} -P {2} -d {2} -x {8} }
    }
}

```

```

    -s { KendangScale }
}
wsngfromscale [Kendang4] {
    -i { wsngrandom [wwwww] { -c -S {704} -s {0} -e {48} -p {0} -P {2} -d {2} -x {8}
        } }
    -s { KendangScale }
}
wsngfromscale [Kendang5] {
    -i { wsngrandom [wwwww] { -c -S {705} -s {0} -e {64} -p {0} -P {2} -d {2} -x {8}
        } }
    -s { KendangScale }
}

wsngchanmult [drums] {
    -V {0.31}
    -i {
        wsngsequence [drumparts] {
            -i { Kendang1 }
            -i { Kendang2 }
            -i { Kendang3 }
            -i { Kendang4 }
            -i { Kendang5 }
        }
    }
}

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create Gong fragment
%
```

```
gongpart = <P1D12X4V240>
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create the Finale segment
%
```

```

GenPanLast = <P12D1X0>
GenBarLast = <P12D2X1T-1 P6>
GambangLast = <P6D1X2>
BonBarLast = <P6D2X3>
BonPanLast = <P6D1X7>
SarBarLast = <P12D8X5>
PekingLast = <P12D2X6>
KendangLast = <P1D2V80X8>

```

```

wsngsequence [finale] {
    -s
    -i { gongpart }
    -i { GenPanLast }
    -i { GenBarLast }
    -i { GambangLast }
    -i { BonPanLast }
    -i { BonBarLast }
    -i { PekingLast }
    -i { SarBarLast }
    -i { KendangLast }
}

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% now pull the different voices together
%
```

```

wsngsequence [genders] {
    -s

```

```

-i { genpan }
-i { genbar }
-i { gambangs }
}

% combine bonangs to play together
wsngsequence [bonangs] {
-s
-i { bonpans }
-i { bonbars }
-i { sarons }
}

wsngsequence [colotomics] {
-s
-i {
wsngsequence [multigongs] {
-s
-i { gongpart }
-i { gongpart }
-i { gongpart }
}
}
-i { drums }
}

wsngsequence [onecycle] {
-s
-i { genders }
-i { bonangs }
-i { pekings }
-i { colotomics }
}

% this is a dummy so that we can selectively play different structures
wsngtranspose [play] {
-n { 0 } % need to use 12 or 24 for western keys
-i { wsngsequence [majorsections] {
-i { wsngfromscale [render] {
-s { wsngscale [slendroSorrel] {
-m { 7 } % 8 is nice, 3 is a bit dark and mysterious
}
}
-i { wsngsequence [verses] {
-i { onecycle }
% -i { wsngretrograde [second] { -i { onecycle } } }
-i { onecycle }
} } } }
-i { wsngtranspose [bluesy] {
-n {24}
-i { wsngfromscale [rendermore] {
-s { wsngscale [darktones] {
-m { 3 } % 8 is nice, 3 is a bit dark and
mysterious
}
}
-i { verses }
} } } }
-i { wsngtranspose [halfback] {
-n {0}
-i { wsngfromscale [renderstillmore] {
-s { wsngscale [offtone] {
-m { 6 } % 8 is nice, 3 is a bit dark and
mysterious
}
}
-i { verses }
} } } }
} } } }

```


B.10 Incident on Pier 6

```

% create several genpan fragments, each from a slightly different note palette
%

GenPan1Scale = <P7 P8 P9 P11 P12>
GenPan2Scale = <P6 P7 P8 P9 P12 P13>
GenPan3Scale = <P8 P9 P11 P12 P13>
GenPan4Scale = <P7 P9 P11 P12 P13>

% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
wsngfromscale [GenPan1] {
  -i { wsngrandom [ii] { -c -S {103} -g {0.03} -s {0} -e {8} -p {0} -P {4} -x {0} }
  }
  -s { GenPan1Scale }
}
wsngfromscale [GenPan2] {
  -i { wsngrandom [iii] { -c -S {203} -g {0.03} -s {0} -e {8} -p {0} -P {5} -x {0}
  } }
  -s { GenPan2Scale }
}
wsngfromscale [GenPan3] {
  -i { wsngrandom [iiii] { -c -S {301} -g {0.03} -s {0} -e {8} -p {0} -P {4} -x {0}
  } }
  -s { GenPan3Scale }
}
wsngfromscale [GenPan4] {
  -i { wsngrandom [iiiiii] { -c -S {530} -g {0.03} -s {0} -e {8} -p {0} -P {4} -x
  {0} } }
  -s { GenPan4Scale }
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% merge the genpan fragments
%

wsngrepeat [genpancommon] {
  -c { 3 } -i { GenPan1 }
}
wsngsequence [genpanraw] {
  -i { genpancommon } -i { GenPan2 }
}
wsngsequence [genpanraw2] {
  -i { genpancommon } -i { GenPan3 }
}
wsngsequence [genpanraw3] {
  -i { genpancommon } -i { GenPan4 }
}
wsngrepeat [genpanbits] {
  -c { 2 } -i { genpanraw }
}

wsngtranspose [genpan] {
  -n { 12 }
  -i {
    wsngchanmult [genmain] {
      -V {0.8}
      -i {
        wsngsequence {
          -i { genpanbits }
          -i { genpanraw2 } -i { genpanraw3 }
          -i { genpanbits } -i { genpanraw2 }
          -i { genpanraw }
        }
      }
    }
  }
}

```

```

}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create several genbar fragments, each from a slightly different note palette
%

GenBar1Scale = <P6 P7 P8 P9 P11 P12>
GenBar2Scale = <P5 P6 P7 P11 P12>
GenBar3Scale = <P3 P5 P6 P7 P8 P9 P11 P12 P13>
GenBar4Scale = <P6 P7 P8 P9 P11 P12 P13 P14 P15>
GenBar5Scale = <P3 P5 P6 P7 P8 P9 P11 P12 P13>
% GenBar6Scale = GenBar3Scale

% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
wsngfromscale [GenBar1] {
  -i { wsngrandom [jj] { -c -S {70} -s {0} -e {16} -p {0} -P {5} -d {2} -x {1} } }
  -s { GenBar1Scale }
}
wsngfromscale [GenBar2] {
  -i { wsngrandom [jjj] { -c -S {101} -s {0} -e {16} -p {0} -P {4} -d {2} -x {1} } }
  -s { GenBar2Scale }
}
wsngfromscale [GenBar3] {
  -i { wsngrandom [jjjj] { -c -S {470} -s {0} -e {48} -p {0} -P {8} -d {2} -x {1} } }
  -s { GenBar3Scale }
}
wsngfromscale [GenBar4] {
  -i { wsngrandom [jjjjj] { -c -S {670} -s {0} -e {32} -p {0} -P {8} -d {2} -x {1} } }
  -s { GenBar4Scale }
}
wsngfromscale [GenBar5] {
  -i { wsngrandom [jjjjjj] { -c -S {1071} -s {0} -e {32} -p {0} -P {8} -d {2} -x {1} } }
  -s { GenBar5Scale }
}
wsngfromscale [GenBar6] {
  -i { wsngrandom [jjjjjjj] { -c -S {1703} -s {0} -e {64} -p {0} -P {8} -d {2} -x {1} } }
  -s { GenBar3Scale }
}
wsngrepeat [genbarcommon] {
  -c { 2 } -i { GenBar1 }
}

wsngchanmult [genbar] {
  -V {0.8}
  -i {
    wsngsequence [genbarset] {
      -i { genbarcommon }
      -i { GenBar2 } -i { GenBar3 }
      -i { GenBar4 } -i { genbarcommon }
      -i { GenBar5 } -i { GenBar6 }
    }
  }
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create several Gambang fragments, each from a slightly different note palette
%

```



```

Gambang1Scale = <P6 P7 P8 P9 P11 P12 P13 P14>
Gambang2Scale = <P9 P11 P12 P13 P14 P15>
Gambang3Scale = <P6 P7 P8 P9 P11 P12 P13 P14 P15>
Gambang4Scale = <P8 P9 P11 P12 P13 P14 P15>
Gambang5Scale = <P3 P5 P6 P7 P8 P9 P11 P12 P13 P14 P15>

% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
% I WASN'T GETTING ENOUGH VARIETY FROM THE MELODY ON THIS CHANNEL
% SO I DOUBLED ITS -P VALUE, SO THAT IT WILL DRAW FROM A WIDER RANGE OF NOTES
wsngfromscale [Gambang1] {
  -i { wsngrandom [kk] { -c -S {13050} -s {0} -e {48} -p {0} -P {14} -d {0.5} -x
    {2} } }
  -s { Gambang1Scale }
}
wsngfromscale [Gambang2] {
  -i { wsngrandom [kkk] { -c -S {5101} -s {0} -e {48} -p {0} -P {10} -d {1} -x {2}
    } }
  -s { Gambang2Scale }
}
wsngfromscale [Gambang3] {
  -i { wsngrandom [kkkk] { -c -S {71590} -r {0.3} -s {0} -e {48} -p {0} -P {16} -d
    {0.5} -x {2} } }
  -s { Gambang3Scale }
}
wsngfromscale [Gambang4] {
  -i { wsngrandom [kkkkk] { -c -S {3104} -r {0.3} -s {0} -e {48} -p {0} -P {12} -d
    {0.5} -x {2} } }
  -s { Gambang4Scale }
}
wsngfromscale [Gambang5] {
  -i { wsngrandom [kkkkkk] { -c -S {18035} -s {0} -e {64} -p {0} -P {20} -d {1} -x
    {2} } }
  -s { Gambang5Scale }
}

wsngchanmult [gambang5] {
-V {0.9}
-i {
  wsngsequence {
    -i { Gambang1 }
    -i { Gambang2 }
    -i { Gambang3 }
    -i { Gambang4 }
    -i { Gambang5 }
  }
}}

% create several Bonbar fragments, each from a slightly different note palette
%

BonBar1Scale = <P6 P11 P12 P13>
BonBar2Scale = <P9 P11 P12 P13>
BonBar3Scale = <P6 P8 P9 P11 P12 P13>
BonBar4Scale = <P6 P11 P12 P13>
BonBar5Scale = <P5 P6 P7 P8 P9 P13>

% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
wsngfromscale [BonBar1] {
  -i { wsngrandom [mm] { -c -S {11301} -s {0} -e {48} -p {0} -P {3} -d {2} -r
    {0.25} -x {3} } }
  -s { BonBar1Scale }
}
wsngfromscale [BonBar2] {

```



```

wsngfromscale [BonPan2] {
  -i { wsnsequence [cxky] {
    -i { wsnrepeat [aoiq] {
      -c { 4 }
      -i { wsnrandom [uu] { -c -S {117431} -s {0} -e {4} -p {0} -P {3}
        -d {1} -r {0.25} -x {7} } } } }
    -i { wsnrepeat [aow] {
      -c { 4 }
      -i { wsnrandom [uuu] { -c -S {141732} -s {0} -e {4} -p {0} -P {3}
        -d {1} -r {0.25} -x {7} } } } }
    -i { wsnrepeat [oiq] {
      -c { 4 }
      -i { wsnrandom [uuuu] { -c -S {411733} -s {0} -e {4} -p {0} -P
        {3} -d {1} -r {0.25} -x {7} } } } } } }
  -s { BonBar2Scale }
}

wsngsequence [BonPan3] {
  -i { wsnretrograde [blap] {
    -i { BonPan1 }
  } } }

wsngfromscale [BonPan4] {
  -i { wsnsequence [asuix] {
    -i { wsnrepeat [lksaoi] {
      -c { 4 }
      -i { wsnrandom [hh] { -c -S {13141} -s {0} -e {4} -p {0} -P {3} -
        d {1} -r {0.75} -x {7} } } } }
    -i { wsnrepeat [nhg] {
      -c { 4 }
      -i { wsnrandom [hhh] { -c -S {13402} -s {0} -e {4} -p {0} -P {3}
        -d {1} -r {0.25} -x {7} } } } }
    -i { wsnrepeat [uyu] {
      -c { 4 }
      -i { wsnrandom [hhhh] { -c -S {1143} -s {0} -e {4} -p {0} -P {3}
        -d {1} -r {0.25} -x {7} } } } } } }
  -s { BonBar4Scale }
}

wsngfromscale [BonPan5] {
  -i { wsnsequence [kx] {
    -i { wsnrepeat [csfse] {
      -c { 4 }
      -i { wsnrandom [dd] { -c -S {181401} -s {0} -e {4} -p {0} -P {3}
        -d {1} -r {0.75} -x {7} } } } }
    -i { wsnrepeat [zdsa] {
      -c { 4 }
      -i { wsnrandom [ddd] { -c -S {811402} -s {0} -e {4} -p {0} -P {3}
        -d {1} -r {0.25} -x {7} } } } }
    -i { wsnrepeat [gdf] {
      -c { 4 }
      -i { wsnrandom [dddd] { -c -S {11843} -s {0} -e {4} -p {0} -P {3}
        -d {1} -r {0.25} -x {7} } } } }
      -i { wsnrepeat [erwg] {
        -c { 4 }
        -i { wsnrandom [dddd] { -c -S {18143} -s {0} -e {4} -p {0} -P
          {3} -d {1} -r {0.25} -x {7} } } } } } }
    -s { BonBar5Scale }
  }
}

wsngtranspose [bonpans] {
  -n { 12 }
  -i {
    wsnchanmult [bonpanset] {
      -V {0.78}
    }
    -i {
      wsnsequence {

```

```

        -i { BonPan1 }
        -i { BonPan2 }
        -i { BonPan3 }
        -i { BonPan4 }
        -i { BonPan5 }
    }
}
}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create several SarBar fragments , each from a slightly different note palette
%

SarBar1Scale = <P12 P13>
SarBar2Scale = <P9 P11 P12 P13>
SarBar3Scale = <P8 P9 P11 P12 P13>
SarBar4Scale = <P11 P12 P13>
SarBar5Scale = <P7 P8 P9 P12 P13>

% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
wsngfromscale [SarBar1] {
    -i { wsngrandom [pp] { -c -S {2121} -s {0} -e {48} -p {0} -P {1} -d {8} -r {0.5}
        -x {5} } }
    -s { SarBar1Scale }
}
wsngfromscale [SarBar2] {
    -i { wsngrandom [ppp] { -c -S {2122} -s {0} -e {48} -p {0} -P {3} -d {8} -x {5} } }
    -s { SarBar2Scale }
}
wsngfromscale [SarBar3] {
    -i { wsngrandom [pppp] { -c -S {19203} -s {0} -e {48} -p {0} -P {4} -d {8} -x {5} } }
    -s { SarBar3Scale }
}
wsngfromscale [SarBar4] {
    -i { wsngrandom [ppppp] { -c -S {221204} -s {0} -e {48} -p {0} -P {2} -d {8} -r
        {0.33} -x {5} } }
    -s { SarBar4Scale }
}
wsngfromscale [SarBar5] {
    -i { wsngrandom [pppppp] { -c -S {221205} -s {0} -e {64} -p {0} -P {4} -d {8} -r
        {0.125} -x {5} } }
    -s { SarBar5Scale }
}

wsngchanmult [sarons] {
-V {0.8}
-i {
wsngsequence {
    -i { SarBar1 }
    -i { SarBar2 }
    -i { SarBar3 }
    -i { SarBar4 }
    -i { SarBar5 }
} } }

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create several Peking fragments , each from a slightly different note palette
%

Peking1Scale = <P11 P12>
Peking2Scale = <P11 P12 P13>
Peking3Scale = <P9 P11 P12 P13>
Peking4Scale = <P8 P9 P11 P12 P13>

```

```
Peking5Scale = <P11 P12 P13>
Peking6Scale = <P7 P8 P9 P12 P13>
```

```
% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
wsngfromscale [Peking1] {
  -i { wsngrandom [qq] { -c -S {311201} -s {0} -e {8} -p {0} -P {1} -d {2} -x {6} }
  }
  -s { Peking1Scale }
}
wsngfromscale [Peking2] {
  -i { wsngrandom [qqq] { -c -S {12102} -s {0} -e {32} -p {0} -P {2} -d {2} -x {6}
  } }
  -s { Peking2Scale }
}
wsngfromscale [Peking3] {
  -i { wsngrandom [qqqq] { -c -S {11032} -s {0} -e {32} -p {0} -P {3} -d {2} -x {6}
  } }
  -s { Peking3Scale }
}
wsngfromscale [Peking4] {
  -i { wsngrandom [qqqqq] { -c -S {12014} -s {0} -e {48} -p {0} -P {4} -d {2} -x
  {6} } }
  -s { Peking4Scale }
}
wsngfromscale [Peking5] {
  -i { wsngrandom [qqqqqq] { -c -S {12105} -s {0} -e {48} -p {0} -P {2} -d {2} -x
  {6} } }
  -s { Peking5Scale }
}
wsngfromscale [Peking6] {
  -i { wsngrandom [qqqqqqq] { -c -S {10126} -s {0} -e {64} -p {0} -P {4} -d {2} -x
  {6} } }
  -s { Peking6Scale }
}
}
```

```
wsngrepeat [pekingfrag1] {
  -c { 4 } -i { Peking1 }
}
wsngsequence [pekingfrag2] {
  -i { Peking2 }
  -i { Peking3 }
  -i { Peking4 }
  -i { Peking5 }
  -i { Peking6 }
}
}
```

```
wsngtranspose [pekings] {
  -n { 0 }
  -i { wsnchanmult [pekingfrags] {
    -V {0.8}
    -i { wngsequence {
      -i { pekingfrag1 }
      -i { pekingfrag2 }
    } } } }
}
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create several Kendang fragments, each from a slightly different note palette
%
KendangScale = <P1 P5 P6>
```

```
% the -P value is based on the number of elements in its input scale
% the -e value is based on the phrase length of the original Wilujeng encoding
wsngfromscale [Kendang1] {
```

```

    -i { wsngrandom [ww] { -c -S {17501} -s {0} -e {48} -p {0} -P {2} -d {2} -x {8} }
    }
    -s { KendangScale }
}
wsngfromscale [Kendang2] {
    -i { wsngrandom [www] { -c -S {51702} -s {0} -e {32} -p {0} -P {2} -d {2} -x {8}
    } }
    -s { KendangScale }
}
wsngfromscale [Kendang3] {
    -i { wsngrandom [www] { -c -S {17503} -s {0} -e {32} -p {0} -P {2} -d {2} -x {8}
    } }
    -s { KendangScale }
}
wsngfromscale [Kendang4] {
    -i { wsngrandom [www] { -c -S {17054} -s {0} -e {48} -p {0} -P {2} -d {2} -x
    {8} } }
    -s { KendangScale }
}
wsngfromscale [Kendang5] {
    -i { wsngrandom [www] { -c -S {15705} -s {0} -e {64} -p {0} -P {2} -d {2} -x
    {8} } }
    -s { KendangScale }
}

wsngchanmult [drums] {
    -V {0.31}
    -i { wsnsequence [drumparts] {
        -i { Kendang1 }
        -i { Kendang2 }
        -i { Kendang3 }
        -i { Kendang4 }
        -i { Kendang5 }
    } } }

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% create Gong fragment
%
gongpart = <P1D12X4V200>

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% now pull the different voices together
%

wsngsequence [genders] {
    -s
    -i { genpan }
    -i { genbar }
    -i { gambangs }
}

% combine bonangs to play together
wsngsequence [bonangs] {
    -s
    -i { bonpans }
    -i { bonbars }
    -i { sarons }
}

wsngtranspose [colotomics] {
    -n {12}
    -i {
wsngsequence {

```

```

-s
-i {
    wsnsequence [multigongs] {
        -s
        -i { gongpart }
        -i { gongpart }
        -i { gongpart }
        } }
-i { drums }
}}

```

```

wsnsequence [melodycycle] {
    -s
    -i { genders }
    -i { bonangs }
    -i { pekings }
}

```

% gradually introduce each instrument

```

wsnsequence [melodyintro] {
-s
-i {
    wsnselect {
        -s {83} -p -x {0} % 40 s
        -i {genpan}
    } }
-i {
    wsnselect {
        -s {71} -p -x {1} % 50 s
        -i {genbar}
    } }
-i {
    wsnselect {
        -s {59} -p -x {2} % 60 s
        -i {gambangs}
    } }
-i {
    wsnselect {
        -s {37} -p -x {6} % 70 s
        -i {pekings}
    } }
-i {
    wsnselect {
        -s {23} -p -x {7} -x {3} -x {5}
        -i {bonangs}
    } }
}

```

% gradually silence each instrument until only the beat and gongs are left

```

wsnsequence [melodyoutro] {
-s
-i {
    wsnselect {
        -e {101} -p -x {0} % 40 s
        -i {genpan}
    } }
-i {
    wsnselect {
        -e {121} -p -x {1} % 50 s
        -i {genbar}
    } }
-i {
    wsnselect {
        -e {153} -p -x {2} % 60 s
        -i {gambangs}
    } }
}

```

```

-i {
  wsngselect {
    -e {171} -p -x {6} % 70 s
    -i {pekings}
  }
}
-i {
  wsngselect {
    -e {200} -p -x {7} -x {3} -x {5} % 80 s
    -i {bonangs}
  }
}

wsngsequence [verses] {
  -i { melodyintro }
  -i { wsngretrograde [secondbest] { -i {melodycycle} } }
  -i { melodycycle }
  -i { melodyoutro }
}

wsngsequence [rhythm] {
-i {
  wsngsequence {
    -i { colotomics }
    -i { colotomics }
    -i { colotomics }
    -i { colotomics }
    -i {
      wsngchanadd {
        -T {80}
        -P {12}
        -i { multigongs }
      }
    }
  }
}
}

wsngsequence [majorsections] {
  -i {
    wsngfromscale [render] {
      -s { wsngscale [slendroSorrel] {
        -m { 1 } % 7 and 8 are nice , 3 is a bit dark and mysterious
      } }
    }
    -i {
      wsngsequence {
        -s
        -i {verses}
        -i {rhythm}
      } } } }
}

% this is a dummy so that we can selectively play different structures
wsngselect [play] {
-s { 0 }
% -x {0}
-x {0} -x {2} -x {4} -x {5} -x {7} -x {8}
-i {
wsngtranspose {
  -n { -13 } % need to use 12 or 24 for western keys
  -i {
    wsngchanmult [tempoadjust] {
      -t { 0.9 }
      %-i { wsngrandomize [amateurs] {-T {0.03} -V {20} -i { majorsections } } }
      -i { majorsections }
    } } } } }
}

```


B.11 Ladrang Wilujeng

```
% cool midi instruments for gamelan
% 09 glockenspiel
% 10 musicbox
% 11 vibes
% 12 marimba
% 13 xylophone
% 14 tubular bells (GREAT for gong)
% 77 shakazul
% 107 koto
% 108 kalimba
% 110 fiddle
% 112 carillon
% 113 agogo
% 114 steel drum
% 115 woodblock
% 116 taiko drums (use pitches 56, 60 and 64)

% channel X9 appears to be the drum channel

GenPan1 = <P12D1X0 P8 P9 P8 P7 P8 P9 P11>
GenPan2 = <P12D1X0 P8 P9 P8 P7 P6 P13 P6>
GenPan3 = <P12D1X0 P9 P11 P12 P13 P12 P9 P8>
GenPan4 = <P12D1X0 P13 P12 P11 P9 P7 P13 P7>
GenPanLast = <P12D1X0>

wsngrepeat [genpancommon] {
  -c { 3 } -i { GenPan1 }
}
wsngsequence [genpanraw] {
  -i { genpancommon } -i { GenPan2 }
}
wsngsequence [genpanraw2] {
  -i { genpancommon } -i { GenPan3 }
}
wsngsequence [genpanraw3] {
  -i { genpancommon } -i { GenPan4 }
}
wsngrepeat [genpanbits] {
  -c { 2 } -i { genpanraw }
}

wsngtranspose [genpan] {
  -n { 12 }
  -i {
    wsngsequence [genmain] {
      -i { genpanbits }
      -i { genpanraw2 }
      -i { genpanraw3 }
      -i { genpanbits }
      -i { genpanraw2 }
      -i { genpanraw }
    }
  }
}

GenBar1 = <P12D4X1 P9D2 P11 P8D1 P7 P11D2T-1 P6 P9T-1 P6 P11T-1 P6>
GenBar2 = <P12D2X1 P11 P12 P11T-1 P5 P6 P11T-1 P7 P12 P11T-1 P7>
GenBar3 = <P13D2X1 P11 P12T-1 P5 P11T-1 P3 P13T-1 P5 P11T-1 P6 P12T-1 P7 P13T-1 P8
P12T-1 P6 P11 P12 P11T-1 P6 P7 P11T-1 P8 P12 P11T-1 P8 P9 P12T-1 P6 P11T-1 P5
P12T-1 P3 P9 P12T-1 P6 P11T-1 P7 P12T-1 P8>
GenBar4 = <P13D2X1T-1 P7 P12 P11T-1 P8 P12T-1 P6 P13T-1 P9 P11T-1 P7 P12 P13T-1 P8
P12T-1 P6 P13 P14T-1 P7 P13T-1 P6 P15T-1 P7 P13T-1 P8 P14T-1 P9 P13T-1 P11>
```

```

GenBar5 = <P12D2X1 P11 P12 P11T-1 P5 P9T-1 P6 P11T-1 P7 P12 P11T-1 P7 P13T-1 P6D1V
-0 P6 P5D2 P12 P3 P5T-1 P13 P6 P7T-1 P12 P8T-1 P13>
GenBar6 = <P12D2X1T-1 P6 P13 P7 P12 P13T-1 P9 P12T-1 P8D1V-0 P8 P11D2T-1 P7 P12T-1
P7 P13 P12 P8 P12T-1 P9 P13 P8 P12T-1 P9 P13T-1 P11 P12T-1 P8 P11 P12 P11T
-1 P5 P6 P11T-1 P7 P12 P11T-1 P7 P13 P11 P12T-1 P5 P11T-1 P3 P13T-1 P5
P11T-1 P6 P12T-1 P7 P13T-1 P8>

GenBarLast = <P12D2X1T-1 P6>

wsngrepeat [genbarcommon] {
  -c { 2 } -i { GenBar1 }
}
wsngsequence [genbar] {
  -i { genbarcommon }
  -i { GenBar2 }
  -i { GenBar3 }
  -i { GenBar4 }
  -i { genbarcommon }
  -i { GenBar5 }
  -i { GenBar6 }
}

Gambang1 = <P6D1X2 P6 P6 P6 P7 P8 P9 P11 P12 P12 P12 P12 P12 P12 P12 P11 P12 P11
P9 P8 P9 P8 P9 P11 P12 P11 P12 P11 P13 P12 P11 P13 P12 P14 P13 P12 P13 P12
P13 P14 P15 P14 P13 P12 P13 P12 P13 P14>

Gambang2 = <P15D1X2 P14 P13 P14 P15 P14 P13 P14 P15 P14 P15 P14 P13 P12 P12
P12 P12 P14 P13 P12 P13 P12 P13 P14 P15 P14 P13 P12 P13 P12 P13 P14 P15 P14
P13 P12 P11 P9 P9 P9 P9 P9 P11 P9 P11 P12 P11 P12>

Gambang3 = <P13D1X2 P14 P13 P14 P15 P14 P13 P15 P14 P12 P13 P12 P11 P9 P11 P12
P9 P9 P11 P12 P13 P12 P13 P14 P12 P11 P9 P8 P6 P7 P6 P7 P8 P6 P6 P6 P7 P8
P9 P11 P12 P12 P12 P12 P12 P12 P12 P11>

Gambang4 = <P12D1X2 P11 P9 P8 P9 P8 P9 P11 P12 P11 P12 P11 P13 P12 P11 P13
P12 P14 P13 P12 P13 P12 P13 P14 P15 P14 P13 P12 P13 P12 P13 P14 P15 P14
P13 P14 P15 P14 P13 P14 P15 P14 P15 P14 P13 P12 P12 P12>

Gambang5 = <P12D1X2 P13 P13 P13 P13 P12 P11 P12 P13 P12 P11 P9 P11 P9 P11 P12
P13 P15 P15 P13 P14 P12 P12 P9 P9 P8 P7 P6 P7 P8 P8 P8 P8 P8 P7 P6 P7 P6 P7
P8 P9 P8 P7 P6 P8 P7 P6 P8 P7 P7 P8 P7 P8 P7 P6 P5 P3 P3 P5 P3 P5 P6 P3
P5>

GambangLast = <P6D1X2>

BonPan1Wrong = <P7T1D1X4 P7 P7 P7V-0 P7 P7 P7V-0 P7V-0 P7 P7 P7 P7V-0 P7 P7 P7V-0
P7V-0 P9 P8 P9 P7V-0 P9 P8 P9 P7V-0 P9 P8 P9 P7V-0 P9 P8 P9 P7V-0 P8 P7 P8 P7V-0
P8 P7 P8 P7V-0 P8 P7 P8 P7V-0 P8 P7 P8>

BonPanTriplet = <P6D1X4V-0 P6 P6 P6D1T-1 P12>
BonPanDoublet = <P6D1X4V-0 P6 P6D1T-1 P12>
BonPanRestStrike = <P12D2X4V-0 P6D1T-1 P12>
BonPanRestStrike2 = <P12D1X4V-0 P6D1T-1 P12>
BonPanDip2 = <P13D1X4V-0 P13 P11 P13>
BonPanDip1 = <P13D1X4V-0 P13 P12 P13>
BonPanHill1 = <P12D1X4V-0 P12 P13 P12>
BonPanHill2 = <P9D1X4V-0 P9 P11 P9>
BonPanFinish = <P6D1X4V-0 P6 P6 P7>
BonPanLast = <P6D1X4>
SixteenthRest = <P13D1V-0>
EighthRest = <P13D2V-0>

wsngsequence [BonPan5] {
  -i { BonPanTriplet }
  -i { BonPanDoublet }
  -i { SixteenthRest }
}

```

```

-i { BonPanTriplet }
-i { BonPanDoublet }
-i { wsgrepeate [arky] {
    -c { 4 }
    -i { wsgtranspose [mijh] {
        -n { -4 }
        -i { BonPanDip1 }
    }
    }
}
-i { wsgrepeate [amlr] {
    -c { 4 }
    -i { wsgtranspose [djik] {
        -n { -5 }
        -i { BonPanDip1 }
    }
    }
}
-i { wsgrepeate [qytr] {
    -c { 3 }
    -i { wsgtranspose [picn] {
        -n { -4 }
        -i { BonPanHill2 }
    }
    }
}
-i { BonPanFinish }
}

wsgsequence [BonPan1] {
-i { BonPanTriplet }
-i {
    wsgrepeate [lum] {
        -c { 9 }
        -i { BonPanRestStrike }
    }
}
-i { SixteenthRest }
-i {
    wsgrepeate [klup] {
        -c { 4 }
        -i { BonPanDip2 }
    }
}
}

wsgsequence [BonPan2] {
-i {
    wsgrepeate [clak] {
        -c { 4 }
        -i { BonPanDip1 }
    }
}
-i {
    wsgrepeate [claq] {
        -c { 4 }
        -i { BonPanHill2 }
    }
}
-i {
    wsgrepeate [qlak] {
        -c { 4 }
        -i { BonPanHill1 }
    }
}
}

```

```

}
wsngsequence [BonPan3] {
  -i {
    wsngrepeat [blap] {
      -c { 4 }
      -i {
        wsngtranspose [lorp] {
          -n { -1 }
          -i { BonPanDip1 }
        }
      }
    }
  }
  -i {
    wsngrepeat [blup] {
      -c { 4 }
      -i {
        wsngtranspose [lyrp] {
          -n { -4 }
          -i { BonPanDip1 }
        }
      }
    }
  }
  -i { BonPanTriplet }
  -i {
    wsngrepeat [lym] {
      -c { 4 }
      -i { BonPanRestStrike }
    }
  }
}

```

```

wsngsequence [BonPan4] {
  -i { BonPanRestStrike }
  -i { BonPanRestStrike2 }
  -i {
    wsngrepeat [glim] {
      -c { 3 }
      -i { BonPanRestStrike }
    }
  }
  -i { EighthRest }
  -i {
    wsngrepeat [jlup] {
      -c { 4 }
      -i { BonPanDip2 }
    }
  }
  -i {
    wsngrepeat [ghil] {
      -c { 4 }
      -i { BonPanDip1 }
    }
  }
}

```

BonBar1Wrong = <P7T2D2X3 P7 P7 P7V-0 P7 P7 P7V-0 P7V-0 P9 P8 P9 P7V-0 P7V-0 P8 P9 P7V-0 P8 P7 P7D0.5 P5D0.5 P7D1 P5D2 P5 P7>

BonBar1 = <P6D2V-0X4 P6 P6 P6T-1 P12 P12D4V-0 P6D2T-1 P12 P12D4V-0 P6D2T-1 P12 P12D4V-0 P6D2T-1 P12 P12D4V-0 P6D2T-1 P12 P12V-0 P13 P11 P13 P13D4V-0 P11D2 P13>

% ? These have strong fragmentables JAS

```

%
BonBar2 = <P13D2X4V-0 P13 P12 P13 P13D4V-0 P12D2 P13 P13V-0 P9 P11 P9 P9D4V-0
P11D2 P9 P9V-0 P12 P13 P12 P12D4V-0 P13D2 P12>

BonBar3 = <P13D2X4V-0 P12 P11 P12 P12D4V-0 P11D2 P12 P12V-0 P9 P8 P9 P12D4V-0
P8D2 P9 P9V-0 P6 P6 P6T-1 P12 P12D4V-0 P6D2T-1 P12 P6V-0>

BonBar4 = <P12D2X4V-0 P6T-1 P12 P6D4V-0 P6D2T-1 P12 P6D4V-0 P6D2T-1 P12 P6V-0
P13 P11 P13 P13D4V-0 P11D2 P13 P13V-0 P13 P12 P13 P13D4V-0 P12D2 P13>

BonBar5 = <P7D2X4V-0 P7 P7 P13T-1 P7 P7V-0 P7 P13T-1 P7 P7V-0 P7V-0 P9 P8 P9
P9D4V-0 P8D2 P9 P9V-0 P8 P7 P7D0.5 P5 P7D1 P5D2 P5 P6 P6V-0 P5V-0 P5 P6
P5 P5V-0 P6 P6 P7>

BonBarLast = <P6D2X4>

SarBar1 = <P12D8X5 P12D16V-0 P12D8 P12V-0 P13>
SarBar2 = <P11D8X5 P13 P12 P9 P11 P12>
SarBar3 = <P13D8X5 P12 P11 P9 P8 P12>
SarBar4 = <P12D8X5 P12D16V-0 P13D8 P11 P13>
SarBar5 = <P12D8X5 P13 P13 P9 P8 P12V-0 P7 P8>
SarBarLast = <P12D8X5>

Peking1 = <P12D2X6 P12 P11 P11>

Peking2 = <P12D2X6 P12 P13 P13 P11 P11 P13 P13 P11 P11 P13 P13 P12 P12 P13 P13>

Peking3 = <P12D2X6 P12 P9 P9 P11 P11 P9 P9 P11 P11 P12 P12 P13 P13 P12 P12>

Peking4 = <P13D2X6 P13 P12 P12 P11 P11 P12 P12 P11 P11 P9 P9 P8 P8 P9 P9 P8 P8 P11
P11 P12 P12 P11 P11>

Peking5 = <P12D2X6 P12 P11 P11 P12 P12 P11 P11 P12 P12 P13 P13 P11 P11 P13 P13
P11 P11 P13 P13 P12 P12 P13 P13>

Peking6 = <P12D2X6 P12 P12 P12 P13 P13 P12 P12 P13 P13 P9 P9 P8 P8 P9 P9 P8 P8
P8 P8 P7 P7 P8 P8 P7 P7 P8 P8 P12 P12 P8 P8>

PekingLast = <P12D2X6>

wsngrepeat [pekingfrag1] {
  -c { 4 } -i { Peking1 }
}
wsngsequence [pekingfrag2] {
  -i { Peking2 }
  -i { Peking3 }
  -i { Peking4 }
  -i { Peking5 }
  -i { Peking6 }
}

Gong = <P1D12X2V240>

Kendang1 = <P1D2V80X8 P5 P6 P1 P6 P5 P1 P5 P6 P5 P5 P6 P1 P5 P6 P1 P5 P6 P1 P6
P5 P1 P5 P6>

Kendang2 = <P1D2V80X8 P5 P6 P5 P6 P5 P1 P5 P6 P5 P5 P6 P1 P5 P6 P1 P5 P6 P6
P5 P6 P5 P1 P5>

Kendang3 = <P6D2V80X8 P5 P6 P5 P6 P5 P6 P5 P6 P5 P1 P5 P6 P5 P6 P5 P1 P5 P6 P5 P1
P5 P6 P5>

Kendang4 = <P6D2V80X8 P5 P6 P6 P1 P5 P6 P5 P5 P5 P6 P5 P6 P5 P6 P5 P1 P5 P6 P5 P6
P6 P1 P5>

```

```
Kendang5 = <P6D2V80X8 P5 P5 P6 P1 P5 P6 P1 P5 P6 P1 P6 P5 P1 P5 P6 P1 P5 P6 P5
P6 P5 P6 P5 P6 P5 P6 P6 P1 P5 P6 P5>
```

```
KendangLast = <P1D2V80X8>
```

```
Rebab = <P12D8V200X11 P12D4V-0 P12D4 P12 P12D2V-0 P12D2 P12D4V-0 P12D4 P12D2V-0
P12D2 P12D4V-0 P13D2 P14>
```

```
wsgsequence [bonpanset] {
  -i { BonPan1 } %BonPan1Wrong
  -i { BonPan2 }
  -i { BonPan3 }
  -i { BonPan4 }
  -i { BonPan5 }
}
```

```
wsgsequence [bonbars] {
  -i { BonBar1 } %BonBar1Wrong
  -i { BonBar2 }
  -i { BonBar3 }
  -i { BonBar4 }
  -i { BonBar5 }
}
```

```
wsgtranspose [bonpans] {
  -n { 12 }
  -i { bonpanset }
}
```

```
wsgtranspose [pekings] {
  -n { 12 }
  -i {
    wsgsequence [pekingfrags] {
      -i { pekingfrag1 }
      -i { pekingfrag2 }
    }
  }
}
```

```
wsgsequence [gambang] {
  -i { Gambang1 }
  -i { Gambang2 }
  -i { Gambang3 }
  -i { Gambang4 }
  -i { Gambang5 }
}
```

```
% combine genpan and genbar to play together
```

```
wsgsequence [genders] {
  -s
  -i { genpan }
  -i { genbar }
  -i { gambangs }
}
```

```
wsgsequence [sarons] {
  -i { SarBar1 }
  -i { SarBar2 }
  -i { SarBar3 }
  -i { SarBar4 }
  -i { SarBar5 }
}
```

```
wsgsequence [drums] {
  -i { Kendang1 }
  -i { Kendang2 }
  -i { Kendang3 }
  -i { Kendang4 }
}
```

```

    -i { Kendang5 }
}

% combine bonangsto play together
wsngsequence [bonangs] {
  -s
  -i { bonpans }
  -i { bonbars }
  -i { sarons }
}

wsngtranspose [gongpart] {
  -n { 0 }
  -i { Gong }
}

wsngsequence [colotomics] {
  -s
  -i { gongpart }
  -i { gongpart }
  -i { gongpart }
  -i { gongpart }
  -i { gongpart }
  -i { drums }
}

wsngfromscale [gendersSlendro] {
  -s { wsngscale [slendroSorrel] {
    -m { 7 }
  }
}
  -i {
    wsngsequence [balungans] {
      -s
      -i { genders }
      -i { bonangs }
      -i { pekings }
      -i { colotomics }
    }
  }
}

%
}

wsngsequence [finale] {
  -s
  -i { wsngfromscale [ak] { -s { wsngscale [am] { -m {3}} } -i { GenPanLast } } }
  -i { wsngfromscale [bk] { -s { wsngscale [an] { -m {3}} } -i { GenBarLast } } }
  -i { wsngfromscale [ck] { -s { wsngscale [ao] { -m {3}} } -i { GambangLast } } }
  -i { wsngfromscale [dk] { -s { wsngscale [ap] { -m {3}} } -i { BonPanLast } } }
  -i { wsngfromscale [ek] { -s { wsngscale [aq] { -m {3}} } -i { BonBarLast } } }
  -i { wsngfromscale [fk] { -s { wsngscale [ar] { -m {3}} } -i { PekingLast } } }
  -i { wsngfromscale [gk] { -s { wsngscale [as] { -m {3}} } -i { SarBarLast } } }
  -i { wsngfromscale [hk] { -s { wsngscale [at] { -m {3}} } -i { KendangLast } } }
  -i { wsngfromscale [lk] { -s { wsngscale [au] { -m {3}} } -i { Gong } } }
}

% this is a dummy so that we can selectively play different structures
wsngchanmult [play] {
  -t { 2.0 } % double the playing time for all notes
  -i {
    wsngsequence [loopthenend] {
      -i { gendersSlendro }
      -i { gendersSlendro }
      -i { gendersSlendro }
      -i { finale }
    }
  }
}

```

} }